

D4.7 Final Tools

Marc Ehrig
Christoph Tempich (University of Karlsruhe)
Zharko Aleksovski (Vrije Universiteit Amsterdam)

with contributions from:
Steffen Staab (University of Karlsruhe)
Alexander Löser (Technical University Berlin)

Executive Summary.

SWAP EU IST-2001-34103 Project Deliverable D4.7

This deliverable presents the final tools which have been developed in the last six months of the project.

Document Id. SWAP/2004/D4.7/v0.2
Project SWAP EU IST-2001-34103
Date September 9, 2004
Distribution Restricted

SWAP Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2001-34103. The partners in this project are: Institute AIFB / University of Karlsruhe (coordinator, Germany), Vrije Universiteit Amsterdam VUA (Netherlands), Meta4 (Spain), empolis UK ltd. (UK), empolis Polska (Poland), and IBIT (Spain).

University of Karlsruhe

Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 3923, Fax: +49 721 608 6580
Contactperson: Steffen Staab
E-mail: staab@aifb.uni-karlsruhe.de

Meta4 Spain S.A.

Rozabella, 8 Centro Europa Empresarial
28230 Las Rozas (Madrid)
Spain
Tel: +34 91 634 85 00, Fax: +34 91 634 86 86
Contactperson: Emilio Martin Cros
E-mail: emiliom@meta4.com

empolis Polska Sp. z o. o.

Plocka 5a
01-231 Warsaw
Poland
Tel: +48 22 535 88 06, Fax: +48 22 535 88 14
Contactperson: Mariusz Olko
E-mail: mariusz.olko@empolis.pl

Vrije Universiteit Amsterdam (VUA)

Division of Mathematics and Informatics W&I
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 4447786, Fax: +31 20 4447653
Contactperson: Hans Akkermans
E-mail: hansakkermans@cs.vu.nl

empolis UK ltd.

Unit B, The Dorcan Complex, Faraday Road
SN3 5HQ Swindon
United Kingdom
Tel: +44 1793 485465, Fax: +44 1793 485451
Contactperson: Justine Taylor
E-mail: jt@empolis.co.uk

Fundación IBIT

Reverend Francesc Sitjar 1
07010 Palma de Mallorca
Spain
Tel: +34 971 177 271, Fax: +34 971 177 279
Contactperson: Esteve Lladó Martí
E-mail: esteve@ibit.org

Changes

Version	Date	Author	Changes
0.1	9.9.04	meh	document setup
0.2	27.9.04	meh	integration of matching section
1.0	21.10.04	cte	integration of routing section

Contents

1	Introduction	2
2	Approximate Matching of Classification Hierarchies	3
2.1	Semantic Matching	3
2.2	Approximate Matching	4
2.3	The Music Domain	5
2.4	Experiments	5
3	Layered based semantic routing	7
3.1	The Social Metaphors	7
3.2	Peer Selection and Ranking Model	9
3.2.1	Example	9
3.2.2	Default Network Layer	9
3.2.3	The Bootstrapping Layer	11
3.2.4	The Content Provider Layer	14
3.2.5	The Recommender Layer	18
3.2.6	The Community Layer	20
3.3	Protocol Scenario	22
3.3.1	Query model	23
3.3.2	Result model	23
3.3.3	Setting 1: Exact Query Forwarding	23
3.3.4	Setting 2: Query forwarding with Zoomle	24
4	Conclusion	30

Chapter 1

Introduction

During the last six months of the project main focus of the project lied on bug-fixing, test, and evaluation of the two SWAP applications Xarop and Bibster. Nevertheless research continued resulting in new methodologies (WP1) and methods (WP3). Some of these were already applied in new tools, which this document describes. Unfortunately these tools couldn't be included into the SWAP applications anymore, as their integration deadline for the systems was already in early April.

In specific this document covers a new matching tool. Instead of using exact semantic matching operators it allows approximate matching. An evaluation of it was performed for the music domain.

The second tool was the result of improved ontology based routing. Social networks have been the foundation for this.

Chapter 2

Approximate Matching of Classification Hierarchies

The work reported in this section has been carried out in tight collaboration with Philips Research Labs Eindhoven. A more detailed description is available as [AtKvH04].

2.1 Semantic Matching

Recently a method for automatically matching of classification hierarchies has been proposed [BSZ03]. The goal of this method is to find mappings between the concepts of two Concept Hierarchies. One can think of a concept hierarchy as a rooted tree where each node and each edge has a label. It has explicit purpose of classifying objects. They create mappings between concepts, where a concept is not defined by a node solely. Next to the label of the node, the position of the node in the hierarchy is used. The method works in three main phases:

- Linguistic interpretation: Using union, intersection and complement operators, they create logic formula (formally, it is a propositional logic formula with interpretation union, intersection and complement) for each node representing all the possible linguistic interpretations of that label. The linguistic interpretation is derived from a background knowledge source, such as WordNet.
- Contextualization: In this phase they determine the component of the meaning of a node's label that depends on its position in the hierarchy. For example, each node's meaning is considered in conjunction with its ancestor's meanings, i.e. each node's formula is enriched as intersection with its ancestor's formulas. This makes sense because we expect a superclass to contain everything that the subclasses contain.
- Semantic comparison: The problem of finding semantic relations between the nodes from two different concept hierarchies is encoded as satisfiability problem of propo-

sitional logic formula (comparing logic formulas). They determine one out of five possible outcomes for each pair of concepts from two different concept hierarchies: (i) they are equivalent, (ii) they are disjoint, (iii) they are not in subclass relation and have non empty intersection, (iv) the first is subclass of the second and (v) the second is subclass of the first.

We extended this approach with an approximate notion of mapping that is more suitable for domains with a high degree of heterogeneity and fuzzyness. In this chapter, we present our extension to the method described above and briefly summarize some experiences from applying it to the problem of computing matches between musical genres.

2.2 Approximate Matching

Following the approach of Bouquet (2003) we are using propositional logic formulas with interpretation union, intersection and complement of classes for each concept in the concept hierarchies. We shape the problem of subclass relation check between two propositional logic formulas into form that uses the normal forms of the formulas. Suppose we want to check whether left side formula is subclass of the right side formula. The left side formula is transformed into disjunctive normal form (union of intersections of classes) and the right side into conjunctive normal form (intersection of unions of classes). Then the problem of subclass check is transformed into checking whether subclass relation holds between each pair disjunct - conjunct of the two normal form formulas. If any disjunct on the left side is subclass of any conjunct on the right side then the first formula is subclass of the second, but if for only few of them that condition does not hold we say that the first formula is almost subclass of the second. Depending on how many pairs disjunct-conjunct satisfy the subclass relation we can measure how strongly the subclass relation between the two given formulas hold.

This idea of approximation can be improved further. Not all disjunct conjunct pairs are equally important for the outcome. Disjuncts and conjuncts can have different size and may consist of various atomic concepts. Using this information may result in more accurate approximations. We expect that bigger size of disjuncts and conjuncts should have less impact on the result. The intuition behind is that more parts in the disjunct make it a class of smaller size because it is intersection, so it's contribution to the class is smaller. Similar intuition holds for the conjuncts as well. "Rare" classes should have the more impact on the result. The intuition is that more general concept names we meet more often, and therefore matching of more general terms doesn't give confidence. When we meet the word "Rock" in two classes, and when we meet "Cajun" in two classes it is natural to expect stronger relation in the second case.

2.3 The Music Domain

Commercial providers offer most of the music metadata schemas on the Internet. They usually have schemas that are easy to browse. Often they contain classes with meaning outside of the music styles domain (Example: "Music Accessories"). Most of the schemas have some peculiarities. After considering several different music metadata schemas from the Internet, we have extracted and adopted seven of them: CDNow (Amazon), MusicMoz, CdBaby, ArtistDirectNetwork, AllMusic, YahooLaunchCast and ArtistGigs. After the extraction we have created 3 new versions of the data. We did preprocessing and normalization in order to make the data more accessible for experimenting. Another reason was to make sure that the results we get are not biased by typing mistakes, abbreviations or similar peculiarities.

The labels on most of the classes in the ontologies have one of the following meanings: style of music (the content of the music), geographic region with music style (region where it comes from) and time or historical period when was the music created. The sibling classes often have overlap (they are not disjoint). The nodes are often named with more than one word. They either denote intersection of the terms named by the separate words, or may contain a term which is multiword (example: "New Zealand Rock", New Zealand is a term and should not be considered as separate words). More often is the first case. There are no strong objective criteria how to classify music. As a result different providers often classify the same music entities (artists, albums, songs...) differently. Genre is not precisely defined. Largely used terms like Pop and Rock do not denote the same sets of artists on different portals. That's the case for even more specific styles of music like Speed Metal. We compared the classified shared artists in MusicMoz and ArtistDirectNetwork (artists that exist in both and are classified in both of the portals). Under the class named Rock in MusicMoz there are 471-shared classified artists, in ArtistDirectNetwork there are 245, and 196-shared artists are classified under Rock in both of them. From all the artists classified under Rock in at least one of the portals, roughly 38% are classified under Rock in both portals. Overall, there is a high degree of fuzziness present in the music domain. Therefore we expect that exact reasoning methods to create matching are not useful, and approximate methods would be more useful.

2.4 Experiments

The goal in these experiments was to get preliminary results and to test and get impression about the approximate matching method. The lexical interpretation (building of the formulas from the labels of the nodes) was done using simple NLP. No background knowledge was used. Special characters were treated as logical operators. When using background knowledge, each of the atomic concepts (in this case three) should be

replaced with union of the different senses for that concept. We made assumption that concepts with the same label have the same meaning. When comparing the disjunct-conjunct relations we only used one rule: a disjunct is subclass of a conjunct when at least one part in the disjunct (which is intersection of atomic concepts) is found in the conjunct (which is union of atomic concepts).

We did experiments with real data extracted from Internet. We tested our method on the data from MusicMoz and ArtistDirectNetwork. Most of the shared classified artists are classified under "Rock"-related classes (Alternative Rock, Glam Rock, Heavy Metal...). The data used to test is not substantial since its size is comparable to the number of the classes present in the concept hierarchies. We have tested for equivalence matching between the two concepts hierarchies, i.e. we were looking for equivalent classes from the two different CHs. We discover equivalent classes using the approximate method by checking if they are both subclass of each other. Since we have to check two times for subclass relation we may get two different values for the sloppiness. In order to assess the success of the matching we use value called significance. It is a ratio between cardinality of the intersection and cardinality of the union of the two classes. The significance is close to 0 when the two classes have no big overlap, i.e. relatively small amount of instances belong to their intersection. When the value is close to 1.0 (or 100%) then the two classes denote almost the same set of instances. We considered only the results from the equivalences where both of the classes had at least 10 instances.

The relations inferred on sloppiness less than 30% were mostly true matches but were not discovered on sloppiness of 0% (were not discovered as perfect match) because of lack of background knowledge. The manual check showed that the approximate matching method performed well considering the facts that no background knowledge was used, the simple sloppiness measure was used, and the data used to test the method is potentially a problem.

Chapter 3

Layered based semantic routing

The work reported in this section has been carried out in tight collaboration with Alexander Löser from the Technical University of Berlin.

3.1 The Social Metaphors

We introduce the metaphors which we exploit in the next section to define the different layers which are used to rank the set of known peers according to their different estimated capabilities to answer a given query. A core task is finding the right peer among the multitude of possible addressees such that this peer returns good answers to a given question. To do this efficiently and effectively we follow the principle of our REMIDNIN approach and build on the following social metaphors of how search human networks. We observe that a human who searches for answers to questions may distinguish between four types of persons in human networks:

1. A question is asked to a person that has answered the question in the past successfully. We call such a person a Content Provider and organize the corresponding peer in the *Content Provider Layer*.¹
2. If no person is known having answered the question in the past we search for a person that has issued similar questions in the past and learn from this person suitable Content Provider. We call such a person a Recommender. The corresponding peers form the *Content Provider Recommender Layer*.
3. If neither a person is known that has asked similar questions nor that has answered our question successfully in the past, we ask a person that has similar interests. Similar interests are observed by answering many of our questions in the past or by having asked many similar questions in the past. Persons with similar interests

¹This layer was introduced with the REMINDIN' approach.

form our community. The corresponding peers are organized in the *Community Layer*.²

4. If we don't know one of the persons above we ask a person that has a broad general knowledge or a person that has established a good social network to other persons over several domains. Such persons form our social bootstrapping network and the peers are related in the *Bootstrapping Layer*.

Our approach builds on the metaphors of peer-to-peer networks being like a human social network and adopts the above mentioned assumptions to the peer model and the query routing algorithm. Our peer model is based on the observation of interactions a peer has with other peers in the network. Peers monitor which other peers frequently respond suc-

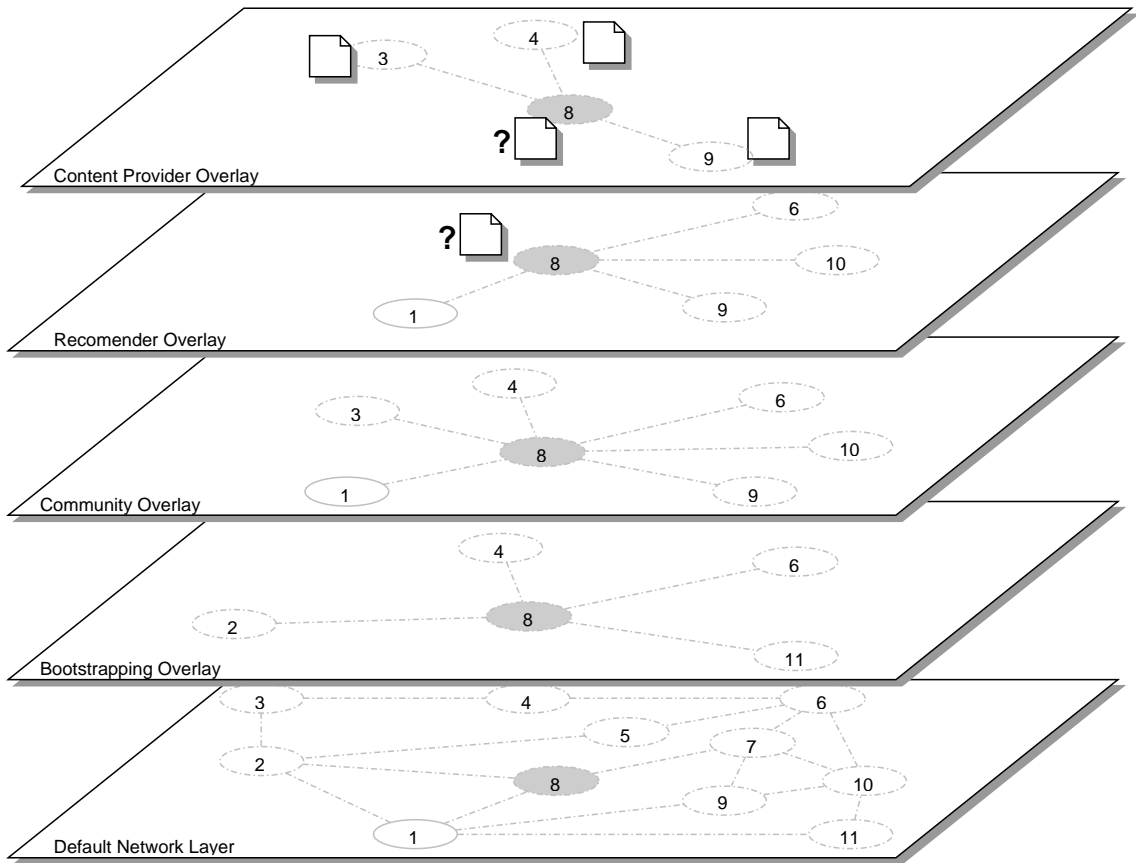


Figure 3.1: Adaptive Multi Overlay Network

cessfully to their requests for information, which peers ask similar questions, which peer provide many documents or which peer have asked many questions to a broad range of topics in the past. When a peer discovers such an information it will be locally stored in a shortcut. Each shortcut represents an additional link on top of the default network layer

²We introduce the community and the bootstrapping layer as an addition to REMINDIN' to facilitate the learning process.

of the peer-to-peer systems. Shortcuts are implemented independent from the physical network layer. Peers benefit from shortcuts by routing its queries directly to other peers along the shortcut overlay, therefore shortcut layers improve the query performance without affecting the scalability and correctness of the underlying default network topology.

Following the social metaphors from above we distinguish between four different shortcut overlays above the default network layer. Our routing algorithm exploits the information we can obtain from the different layers in our routing stack, *viz.* network, bootstrapping, community and content layer. Information from all five layers is eventually combined to decide to which peers a query will be send.

3.2 Peer Selection and Ranking Model

In this section we provided a detailed analysis of the metrics used to select a peer for a query depending on the knowledge the peer has acquired for each layer yet. The layer are described in the order of its short cut discovery.

3.2.1 Example

Before we go into details with the description of the overlay layers we here introduce a simplified peer setup which we will use in the following to exemplify our rating metrics. In table 3.1 the shared content of the peers P1, P2, P3 and P4 is summarized. We plot for each peer the number of documents it shares and the remote peers it knows, *eg.* P1 has 100 documents about “Semantic Web” and knows one peer – namely P2 – which also has information about this topic. Moreover, we introduce in table 3.2 the variables which are used throughout the rest of the paper.

3.2.2 Default Network Layer

The default network layer refers to the layer which provides the peers with the basic communication facilities. On this level for a peer *knowing* another peer means, that a peer can establish a physical connection with the other peer. Existing applications implement various approaches to enable the peers to get to know each other. The Gnutella protocol assumes that peers get to know each other on the physical layer via ping messages send around to a randomly selected number of IP addresses ([Kan99]). JXTA facilitates the retrieval of available peer IP addresses via so called rendezvous peers. They act as super peers to the extend that other peers can publish their IP addresses on them ([Gon01]). Furthermore, on this level each peer can be identified by a *unique address*, its PID. E.g. the

³LDs: Number of statements a peer has stored locally for the given topic
KNs: Number of remote peers a peer has stored locally for the given topic

Topic ³	Peer1		Peer2		Peer3		Peer4	
	# LDs	# KPs	# LDs	# KPs	# LDs	# KPs	# LDs	# KPs
Semantic Web	100	{P2} 1	10	{P1, P4} 2	50	0	1	{P1} 1
Peer-to-Peer	100	{P2, P3} 2	100	{P1, P3, P4} 3	50	0	1	0
Information Systems	10	{P2, P3} 2	100	{P1, P3, P4} 3	50	{P1} 1	1	0
Hiking	0	0	0	0	2	0	1	0
Total	210	2	210	3	150	1	4	1

Table 3.1: Peer Setup

Parameter	Description	Example
S	Statement	(uri1,hasTopic,Semantic-Web)
H	Entropy	$H(Q)$: Entropy of one Query
T	Topic	Semantic-Web
P	Peer	Peer of Christoph
\mathcal{P}	A set of peers	$\{P1, P2, \dots\}$
Q	A Query	(*,isAbout,Semantic-Web) returns all Statements which have "Semantic-Web" as topic
$Path$	The list of peers P which have received the Query	$\{P_3, P_7, P_1\}$
$ S_{P_i}^{Q_j} $	Number of Statements returned by Peer P_i given Query Q_j	Peer P1 has 10 resources with topic Semantic-Web
$p(S_{P_i}^{Q_j}) = \frac{ S_{P_i}^{Q_j} }{ S^{Q_j} }$	Fraction of statements returned by peer P_i given query Q_j	P1 returned 10, P2 90 statements, thus $p(S_{P_1}^{Q_j}) = 0.1$
\mathcal{P}^{Q_j}	The set of peers which have answered Query Q_j	$\{P_3, P_7\}$
P.BS	Bootstrapping capability of a peer	
P.CO	Community capability of a peer	
$ToCo$	Total number of content provider peers	$ToCo = 20$
$ToCo^{topic}$	Total number of content provider peers per topic	$ToCo^{topic} = 3$

Table 3.2: Parameters

JXTA project introduces the concept of virtual addressing, a dynamic mapping between virtual and physical network address, that allows to identify peers over multiple network sessions even if the peers use dynamic IP addresses.

3.2.3 The Bootstrapping Layer

We define *bootstrapping nodes* as peers that provide many statements for a broad range of topics and peers that know many remote peers with a broad range of topics. By routing a

query to bootstrapping nodes, we foster the probability to find a matching content provider for a query if no local knowledge could be used to route a query, e.g. when a new peer connects to the network or in the case the peer can not determine a set of peers to route a query using higher layers. Once an initial set of bootstrapping nodes is found a peer routes it's queries to the best found nodes and starts to acquire further knowledge, e.g. about content provider peers. While a peer is online, it continuously updates it's local bootstrapping index with information about other peers bootstrapping capability.

Discovery of Bootstrapping Short Cuts

When a peer joins the network it requests from its physical neighbors of the default network layer the bootstrapping capability. This set is pruned to the *top* k_{BS} bootstrapping nodes and forms the initial set of bootstrapping nodes of a peer. E.g. in figure 3.1 peers 2,4,6,11 are bootstrapping nodes for the peer 8. The bootstrapping index is updated with the bootstrapping information collected from each peer that provides statements for a query. Furthermore with each query a querying peer sends its bootstrapping information and updates the index as well. Hence the index of bootstrapping layer reflects the set of content provider peers with high bootstrapping capability, the set of initial peers and a set of 'random' peers. Please note, an important prerequisite for determine the bootstrapping capability is that each peer computes and publishes its local bootstrapping capability.

Influencing variables

The bootstrapping capability of a remote peer P_j is determined by two characteristics: By the capability of P_j to share many statements $|S|_{P_j}$ with a diverse knowledge of statements $H(S_{P_j})$ and the capability of P_j to collect a large set of peers $|P|_{P_j}$ that contributed answers for a diverse range of topics $H(P_{P_j})$, hence to collect a large and diverse knowledge about other peers in the network. For example, a peer with a high bootstrapping capability shares a large number of statements for a diverse number of topics and knows many peers that have answered a diverse set of queries. To represent its own bootstrapping capability each remote peer P_j computes locally a quadruple

$$\Psi_{P_j} = (|S|_{P_j}, H(S_{P_j}), |P|_{P_j}, H(P_{P_j}))$$

where:

- $|S|_{P_j}$ represents the total number of published statements by P_j
- $H(S_{P_j})$ determines the diversity of the statements published by P_j . We use the Shannon-Wiener index to compute the entropy of the statements. Intuitively a peer with a high value has a high diversity of statements, hence P_j has a broad knowledge. A low value indicates that P_j does only share statements for a narrowed

domain. The entropy is computed as:

$$H(S_{P_j}) = - \sum_{i=1}^{|shtopic|_{P_j}} \frac{|S_{P_j}(topic_i)|}{|S|_{P_j}} \cdot \log\left(\frac{|S_{P_j}(topic_i)|}{|S|_{P_j}}\right) \quad (3.1)$$

where $|topic|_{P_j}$ denotes the total number of topics shared on P_j and $|S_{P_j}(topic_i)|$ denotes the number of statements for a particular topic i shared by P_j .

- $|P|_{P_j}$ represents the total number of known content provider peers at P_j through shortcuts.
- $H(P_{P_j})$ determines the diversity of the topic statements among the known content provider at P_j through shortcuts. Again to compute the entropy we use the Shannon-Wiener index. A high value indicates that P_j knows other peers that share a broad knowledge, while a low value means that P_j only knows peers sharing knowledge for a narrowed domain. The entropy is computed as:

$$H(P_{P_j}) = - \sum_{i=1}^{|ptopic|_{P_j}} \frac{|P_{P_j}(topic_i)|}{|P|_{P_j}} \cdot \log\left(\frac{|P_{P_j}(topic_i)|}{|P|_{P_j}}\right) \quad (3.2)$$

where $|ptopic|_{P_j}$ denotes the total number of collected topics of remote peers at P_j and $|P_{P_j}(topic_i)|$ denotes the number of remote peers registered at P_j for a particular topic i .

Computation of the Bootstrapping Rank

Once a local peer P has collected Ψ_{P_j} from several remote peers it computes the bootstrapping capability for each P_j and select the *top* k_{BS} P_j with the best bootstrapping capability to forward a query. In all collected Ψ_{P_j} we search for the minimum number of shared statements $Min(S)_{P_j}$ and maximum number of statements $Max(S)_{P_j}$ a remote peer shares and compute the normalized number of shared statements for each P_j :

$$Norm(S)_{P_j} = \frac{Min(S)_{P_j} + |S_{P_j}|}{Max(S)_{P_j} - Min(S)_{P_j}} \quad (3.3)$$

Analogue, in all collected Ψ_{P_j} we search for the minimum $Min(P)_{P_j}$, the maximum number $Max(P)_{P_j}$ of peers that are known to a remote peer and compute the normalized number of known peers for each P_j :

$$Norm(P)_{P_j} = \frac{Min(P) + |P_{P_j}|}{Max(P)_{P_j} - Min(P)_{P_j}} \quad (3.4)$$

Finally, the bootstrapping capability of a remote peer $P.BS$ of P_j , with $0 \leq P.BS < \infty$, is computed by

$$P_j.BS = H(S_{P_j}) \cdot Norm(S)_{P_j} + H(P_{P_j}) \cdot Norm(P)_{P_j} \quad (3.5)$$

Topic	Ψ_{P_1}	Ψ_{P_2}	Ψ_{P_3}	Ψ_{P_4}
Total # statements $ S _{P_j}$	210	210	152	4
Total # peers $ S _{P_j}$	2	3	1	1
Entropy of statements $H(S_{P_j})$	1,23	1,23	1,67	2,00
Entropy of peers $H(P_{P_j})$	0,50	0,39	0	0

Table 3.3: Bootstrapping Measure

Peer P_n knows Peer P_m	P_2				P_3		...
	P_1	P_2	P_3	P_4	P_1	P_3	
Normalized # documents $Norm(S)_{P_j}$	1	1	0,72	0	1	0	...
Normalized # peers $Norm(P)_{P_j}$	0,5	1	0	0	1	0	...
Entropy of statements $H(S_{P_j})$	1,23	1,23	1,67	2,00	1,23	1,67	...
Entropy of peers $H(P_{P_j})$	0,50	0,39	0	0	0,50	0	...
Bootstrapping Capability $P.BS$	1,48	1,62	1,20	0	1,73	0	...

Table 3.4: Bootstrapping Capabilities

Example Computation

We will now show the computation of the bootstrapping capability on an example. Each time a peer sends a query to the network it attaches the vector Ψ . In table 3.3 we have summarized these values for our running example. The receiving peers can then rank the bootstrapping capability of the remote peers they know. In table 3.4 we have calculate the bootstrapping capabilities for different peers, assuming peer P_2 knows all other peers and P_3 knows only peer P_1 .

Index Update Strategy and Index Size

To maximize the bootstrapping capability in the local bootstrapping index with a fixed index size of $top\ k_{BS}$ we use a simple highest in/lowest out strategy: If the bootstrapping capability of a remote peer is higher as of the peer in the index with the lowest bootstrapping capability, the remote peer is added to the index, while the peer with the lowest bootstrapping capability is removed from the index.

3.2.4 The Content Provider Layer

The design of the content provider overlay departs from existing work recently published by [SMZ03] and exploits a simple, yet powerful principle called interest-based locality, which posits that if a *content provider peer* has a particular piece of content that one is interested in, it is very likely that it will have other items that one is interested in as well. The content provider short cut list will grow with each submitted query until the

maximum number of content provider peers is reached. New remote peers are added to the list, each time the querying peer receives an answer from a remote peer. The content provider peers are ranked according to the number of statements they could provide for the query.

Discovery of Content Provider Nodes Short Cuts

When a peer joins the system, it may not have any information about other peers interests. Its first attempt to locate statements is executed through lower layers. The lookup returns a set of peers that store the statement. These peers and the peers the query is forwarded to are potential candidates to be added to the content provider shortcut list. Subsequent queries for statements go through the content provider shortcut list. If a peer cannot find statements through the list, it again issues a lookup through lower layers, and repeats the process for adding new shortcuts. For example in figure 3.1 the peers 3,4,9 return statements for the query of peer 8.

Influencing Variables

The computation of the Content Provider Peer Ranking bases on the shortcuts a peer was able to collect with its past queries. Remember, that for each query a local peer stores shortcuts to remote peers that return statements matching a query. For each query a peer searches its local shortcuts for matching content provider peers. Given that there may be many shortcuts to remote peers $p \in P$ of the form $\Phi_p^Q = (Q, P, |S_p^Q|)$ for a query $q \in Q$ on the local shortcut list of the querying peer, we rank the peers based on their perceived content provider utility. For useful peers exist many shortcuts in the local shortcut list that match the query exact or with a high similarity. If peers are useful, they are ranked at the top of the list.

Content Similarity Function

We define the similarity function $sim : Q \times \Phi \mapsto [0; 1]$ between a query $q \in Q$ and a single shortcut $\phi_p^q \in \Phi_P^Q$, which are both represented as topics path in the same taxonomy, according to the similarity measure for hierarchical semantic networks defined by [LBM03] where l is the length of the shortest path between q and ϕ in the graph spanned by the *SubTopic* relation and h is the minimal level in the tree of either q or ϕ . $\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of shortest path length l and depth h , respectively. Based on their benchmark data set, the optimal values are $\alpha = 0.2, \beta = 0.6$.

$$sim(q, \phi_p^q) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } q \neq \phi_p^q \\ 1 & \text{otherwise} \end{cases} \quad (3.6)$$

Query Q	$ S_{p1}^q $	$ S_{p2}^q $	$ S_{p3}^q $	$ S_{p4}^q $	Total
Semantic Web	100	10	50	1	161
Peer-to-Peer	100	100	50	1	251
Total $ S^q $					412

Table 3.5: Answers to queries issued by peer P_4

$sim(q, \phi_p^q) \times$	$\frac{ S(\phi)_1^q }{ S^q }$	$\frac{ S(\phi)_2^q }{ S^q }$	$\frac{ S(\phi)_3^q }{ S^q }$	$\frac{ S(\phi)_4^q }{ S^q }$
$sim(InformationSystems, SemanticWeb) = 0,8$	0,19	0,02	0,10	0,00
$sim(InformationSystems, Peer - to - Peer) = 0,6$	0,15	0,15	0,07	0,00
$CPR(l, q, p)$	0,34	0,17	0,17	0,00

Table 3.6: Content Provider Rank

Computation of the Content Provider Rank

Since it is likely that some peers will be associated with many similar shortcuts for the query and others with some, we compute the *Content Provider Rank* CPR which is the aggregate weighted similarity of a peer to a given query. For computing the CBR we consider that the only shortcuts are useful for a query are those with a similarity to q greater than a user-provided threshold l . Shortcuts with lower similarity are unlikely to be useful, and therefore we ignore them. For computing the CPR we use the k-Nearest-Neighbor classification. The reason that we employ a Nearest Neighbor classification technique is that it is simple, and it has shown good accuracy in many different settings [KGZY02]. It has also been shown that the k-Nearest Neighbor classification has asymptotic error rate at most twice the Bayes error rate, independent of the distance metric used [DH73]. We weight the similarity between a query and a shortcut with the number of statements in the shortcut matching the query and normalize them with the total number of statements of all shortcuts of a peer matching the query above a similarity threshold. This allows us to rank higher the peers that returned more results. Given $|S_p^q|$ the total number of matching statements of a peer to a query above a similarity threshold l and $|S(\phi)_p^q|$ the number statements of a shortcut ϕ_p^q matching the query with a specific similarity $sim(q, \phi_p^q)$ a peer computes the Content Provider Rank $P_j.CPR(l, q, p)$ of a peer p for a query q .

$$P_j.CPR(l, q, p) = \sum_{\forall \phi_p^q | sim(q, \phi_p^q) > l} sim(q, \phi_p^q) \cdot \frac{|S(\phi)_p^q|}{|S^q|} \quad (3.7)$$

We compare this peer ranking function against data source selection techniques known from the distributed data base community. The better known for text based data bases, are gGloss[GGM95], Cori[FPV⁺98] and more recently InfoBeacon [Coo04].

Example Computation

We will now show the computation of the content capability, assuming that $l > 0,5$, and the similarity between $\text{sim}(\text{InformationSystems}, \text{SemanticWeb}) = 0,8$ and $\text{sim}(\text{InformationSystems}, \text{Peer-to-Peer}) = 0,6$. Peer P_4 queries for resources about “Information Systems”. In table 3.6 we first recall the number of statements the different remote peers have answered to peer P_4 with the ultimate queries. Then we calculate the *Content Provider Rank* for each peer. According to this metric peer P_1 is selected as the first peer to send a query to.

Index Update Strategy and Index Size

Over time a peer will get to know more and more peers which have answered his queries, hence the index storing the content provider peers will grow. However, as [SMZ03] has shown in its experiments that it is not reasonable to remember all remote peers but rather to limit the size of the index. In their case they found that adding a maximum five peers per query is a reasonable trade off for improving performance.

We distinguish to different index sizes. The first one - the *overall content provider index* - determines the total number of content provider peers $ToCo$ we want to remember, while the second one - the *topic specific content provider index* - sets the total number of content provider peers per topic $ToCo^{topic}$.

The index size for $ToCo^{topic}$ is subject to experimental evaluation and will probably depend on the overall size of the network. More difficult to find out is the optimal size of the overall content provider index, and in particular the selection of the least useful peers.

Once the index size is settled, and the index is filled, we must decide on the occurrence of a previously unknown peer whether we want to store the new remote peer and delete a known one or not. Therefore we must rank all known peers according to a given strategy. We can distinguish two kinds of strategies, the first type explores the usage patterns of the peers, the second type the provided content.

Usage based peer ranking In [VKMvS04] different usage based peer ranking methods are evaluated against each other. In particular they make a distinction between *Least recently used (LRU)*, *History* and *Popularity*. They found that the Popularity based approach shows very good results will it has a low memory footprint. In this approach each peer ranks the peers according to (1) the number of statement the peer has provided and (2) the last time it has provided results. We will evaluate this approach against a content based peer ranking.

Content based peer ranking We consider to rank the content provider peers according to the similarity of the content we have stored locally and the answers we received from

the remote peers. Furthermore we propose to generally prefer known peers to new ones. Store a content short cut only if we know the peer or if we have content for the query as well. However, only experimental results will reveal, whether this strategy actually outperforms the simpler usage based strategies.

3.2.5 The Recommender Layer

The name of this layer is an abbreviation of Content Provider Recommender Layer. We define the *Recommender Peer* as a peer that has a high overlap in its queries with our queries. If a peer can not determine a content provider peer for a given query, the peer searches its local recommender index for peers that have issued similar queries to the query and forwards the query to the “best” recommender peers.

Discovery of Recommender Peers

Remember, when a peer was not able to resolve a query by its local knowledge, the peer searches the shortcuts of remote peers via lower overlays. If a shortcut on a remote peer matches the query a peer learns transitive from the remote peers the content provider from the shortcut. Additionally, the peer learns, that the remote peer has issued the same query in the past and stores in its local index a *recommender shortcut* to the remote peer. Hence a prerequisite for using the recommender layer is that each peer *cooperates* with other peers, e.g. publishes its content provider shortcut list and analyzes the shortcut list of remote peers. E.g., in figure 3.1 the peers 1,6,9,10 have similar queries issued to peer 8 in the past for a particular document.

Influencing Variables

The recommender rank of a peer P_j to a query Q_i is determined by its number of matching recommender shortcuts $\Omega_{P_j}^{Q_i} = (Q_i, P_j, |rS_{P_j}^{Q_i}|)$, the similarity between a shortcut and the query $sim(Q_i, \Omega_{P_j}^{Q_i})$ and the number of *recommended* statements $|rS_{P_j}^{Q_i}|$ retrieved by a shortcut. The similarity between a query Q and a recommender shortcut $\Omega_{P_j}^{Q_i}$ is determined by using equation 3.6. The recommender shortcuts differ from the content provider shortcuts in one important aspect, namely the number of statements $|rS_{P_j}^{Q_i}|$ we take into account. Instead of counting the statements a remote peer directly provides to a query, we here sum up the statements which were provided by peers recommended by the remote peer. The recommender peer is always the penultimate peer in the message path.

Query Q	Message Path $Path$	$ rS_{p1}^q $	$ rS_{p2}^q $	$ rS_{p3}^q $	$ rS_{p4}^q $	Total
Semantic Web	$\{P_4, P_1, P_2\}$	60	0	0	100	160
Peer-to-Peer	$\{P_4, P_1, P_3\}$	150	0	0	100	250
	$\{P_4, P_1, P_2\}$					
Total $ rS^q $						410

Table 3.7: Recommended Statements

$sim(q, \phi_p^q) \times$	$\frac{ S(\phi)_1^q }{ S^q }$	$\frac{ S(\phi)_2^q }{ S^q }$	$\frac{ S(\phi)_3^q }{ S^q }$	$\frac{ S(\phi)_4^q }{ S^q }$
$sim(InformationSystems, SemanticWeb) = 0,8$	0,12	0,00	0,00	0,20
$sim(InformationSystems, Peer - to - Peer) = 0,6$	0,22	0,00	0,00	0,34
$RR(l, q, p)$	0,34	0,00	0,00	0,34

Table 3.8: Recommender Rank

Computation of the Recommender Rank

We rank the recommender peers by its *Recommender Rank*, peers with a high score are ranked on the top if the list. Analogue to the CPR we use the k-Nearest-Neighbor method. We weight the similarity $sim(Q_i, \Omega_{P_j}^{Q_i})$ between Q_i and $\Omega_{P_j}^{Q_i}$ with the normalized number of statements and normalize $|rS_{P_j}^{Q_j}|$ with the total number of statements $|rS^{Q_i}|$ of recommender shortcuts of all peers matching the query above a threshold l . We compute the $P_j.RR(l, Q_i, P_j)$ of a peer P_j for a query Q_i according to equation 3.8.

$$P_j.RR(l, Q_i, P_j) = \sum_{\forall \Omega_{P_j}^{Q_i} | sim(Q_i, \Omega_{P_j}^{Q_i}) > l} sim(Q_i, \Omega_{P_j}^{Q_i}) \cdot \frac{|rS_{P_j}^{Q_i}|}{|rS^{Q_i}|} \quad (3.8)$$

Example Computation

We will now show the computation of the recommender capability, assuming that $l > 0,5$, and the similarity between $sim(InformationSystems, SemanticWeb) = 0,8$ and $sim(InformationSystems, Peer - to - Peer) = 0,6$. Peer P_4 queries for resources about “Information Systems”. In table 3.5 the number of original answers to different queries is listed. Instead, in table 3.7 we assign the answered statements to the peers which have recommended the actual content provider peers. Than we calculate the *Recommender Rank* for each peer. According to this metric peer P_1 and peer P_4 are selected as the first peers to send a query to. Of course the local peer P_4 is removed from the set of selected peers.

Index Update Strategy

Similar to the indices defined for the Content Provider Layer we introduce two different indices for the Recommender Layer namely the Overall Recommender Index and the Topic specific Recommender Index. The index size is the same as for Content Provider indices. As in the Content Provider case we recalculate the rank of each known peer when we have received all answers for a query.

3.2.6 The Community Layer

Over the time each peer learns a subset of other peers that have been especially useful in the past, its community. If a peer is not able to select a particular peer for a query from its local knowledge, the query is forwarded to the “best” locally known members of its community. We define *Community Nodes* as peers that provided many statements for a peer’s queries in the past or as peers that have asked many similar queries. Hence, the community overlay clusters peers with similar interests and takes the assumption that peers which have answered successfully many of the peers previous queries they will do so also in future.

Discovery of the Community Short Cuts

The peers will acquire a useful list of community peers only after several interactions with remote peers. To assign a community rank to the known peers we only use the results remote peers answer to our queries. Thus, each time an answer to our query is returned, we update the community ranks of the known peers. In the start up phase of the system, or when a peer joins the network as a beginner, the peers with the highest rank in the bootstrapping layer will probably also form the community layer. After several interactions with the system though, the local peer will build up a list of remote peers which could provide meaningful results to most of its queries.

Influencing Variables

To calculate a remote peers community capability we consider two indicators that the peer has a shared interest with us and should thus be part of the local peers community: We require a remote peer to be able to answer many of our queries $Peer.CO^{content}$ in the past and we regard the recommendation of peers to other peers which can answer our queries as helpful $P.CO^{query}$. For example, a peer has a high community capability if the peer has served in the past as content provider for many of our queries and was able to recommend many other useful content provider for our queries. E.g. in figure 3.1 the peer 8 has retrieved statements from the peers 3,4,9 and has found similar queries at the peers

1,6,10,9, hence the union, with the peers 1,3,4,6,9,10, represents community overlay of peer 8.

Computation of the Community Rank

In contrast to the computation of the bootstrapping capability the community capability is computed complete locally on the basis of the information returned by each query from other peers. However, in contrast to bootstrapping information that is calculated without any delay, computing the community capability requires to monitor the query results of remote peers over a longer training phase. The community capabilities of a remote peer as seen from the local peer are updated each time a peer has received all answers⁴ for a query. It is calculated according to equation 3.9.

$$P.CO = P.CO^{content} + P.CO^{query} \quad (3.9)$$

The content community capability $P.CO^{content}$ of a peer P_i is calculated as the weighted sum of the current community capability and the share in statements returned by peer P_i in comparison to all statements returned for the current query Q_j (cf. 3.10). Using the fraction of statements ensures the independence of the metric with respect to the restrictiveness of the query. We weight this fraction with the entropy of the query. Hence we consider queries to which all selected peers have replied similarly more important to build the community than queries which could be answered by only a small subset of the peers.

$$P_i.CO^{content}(t) = \frac{P_i.CO(t-1) + p(S_{P_i}^{Q_j}) * H(Q_j)}{1 + H(Q_j)} \quad (3.10)$$

$$p(S_{P_i}^{Q_j}) = \frac{|S_{P_i}^{Q_j}|}{|S^{Q_j}|} \quad (3.11)$$

$$H(Q_j) = - \sum_{P_i \in \mathcal{P}} p(S_{P_i}^{Q_j}) \cdot \log(p(S_{P_i}^{Q_j})) \quad (3.12)$$

The query community capability $P.CO^{query}$ of a peer P_i is calculated in the same way as the content community capability (cf. equation 3.14). However, instead of calculating the fraction of statements returned by P_i himself, the input parameter for this metric are the number of statements returned by the peers P_i has recommended. Thus we introduce the function $penultimate(Path)$ (cf. equation 3.13) which returns the last but one peer in the message path, viz. P_{max-1} is the *Query provider* for the answer.

$$P^{max-1} := penultimate(Path) \quad (3.13)$$

⁴We consider only answers as relevant which are returned to the local peer within a given time frame.

$$P_i^{max-1}.COquery(t) = \frac{P_i^{max-1}.COquery(t-1) + p(S_{P_i^{max-1}}^{Q_j}) * H(Q_j)}{1 + H(Q_j)} \quad (3.14)$$

$$p(S_{P_i^{max-1}}^{Q_j}) = \frac{|S_{P_i^{max-1}}^{Q_j}|}{|S^{Q_j}|} \quad (3.15)$$

$$H(Q_j) = - \sum_{P_i \in \mathcal{P}_i^{max-1}} p(S_{P_i}^{Q_j}) \cdot \log(p(S_{P_i}^{Q_j})) \quad (3.16)$$

Example Computation We will now show the computation of the community capability on an example. Each time a peer has received all responses to a query from the remote peers it recalculates the community capability of each peer. In our example we assume that peer P_4 poses the queries to the network. In table 3.9 we have listed the query results, the probabilities as calculated according to equation 3.11 and the resulting entropy $H(Q_j)$, cf. equation 3.12. For the first two queries we assume that peer P_3 is offline. From this information we can calculate the *Community Capability* and the *Query Community Capability* of each peer as done in table 3.11. We observe that the *Community Capability* of peer P_1 for peer P_4 is the highest while peer P_2 has the highest *Bootstrapping Capability* cf. table 3.4.

Index Update Strategy and Index size

In the case of the index size for Community Index the ranking strategy is straightforward. We keep the N remote peers in the index with the highest community rank.

3.3 Protocol Scenario

Peer selection is based on the ratings a peer receives on the different overlay layers. The ratings a peer receives from the content and recommender overlay layer depend on the actual query. The query is forwarded to the peer with the highest rating. For the case the local peer has no exact information in it's shortcut index we use a cooperative zooming strategy, where the query is routed only to peers that content/recommender shortcuts similarity to the query is higher than the content/recommender shortcut similarity of forwarding peer. In the case no similar shortcuts could be found the query is routed via lower layers to the community of the peer and via the bootstrapping network.

3.3.1 Query model

We use a simple query model which is similar to the Gnutella query message[Kan99] Of course the query must contain the query itself. We currently support only queries for documents which are assigned to a certain topic. We currently working on a method to extend our framework to conjunctive queries in an ontology and to full text search.

Additionally to the query we add to each query message the bootstrapping capability of the querying peer. Hence, query receiving peers can quickly update their bootstrapping index.

Moreover we attach the message path to each query. The message path contains the unique ID's of the peers which have already received the query. This ensures, that one remote peer does not forward a query to another remote peer, which has already received the query. The message path is eventually copied to the answer message. In this way we can calculate the Recommender rank of a peer.

Furthermore each query has one almost unique QueryID⁵. This again ensures that a peer does not respond to a query it has already answered. This reduces network load.

The threshold zoomleT is transmitted with each query. As described in section 3.3.4 the receiving peer updates the threshold before it forwards the query. The threshold is used to narrow down the number of peers contacted at each forwarding step.

3.3.2 Result model

In the result we include the answer to the query. Currently we just consider results which contain statements pointing to documents matching the query. However, we are working on an extension to allow for arbitrary RDF being the response. As said before the message path is transmitted with the answer. Furthermore, cooperative peers can transmit their bootstrapping capabilities. We currently assume that all peers are cooperative or at least not malicious. This assumption is valid, since their exist algorithms like the one described in [KSGM03], which enable to reward only responses from not malicious peers.

3.3.3 Setting 1: Exact Query Forwarding

After having defined the query and result model we propose the *exact query routing strategy* that bases on a simple combination of the layers. The algorithm builds on the following metaphors:

- A query is forwarded to a person that can answer the query (*content layer*).

⁵Unique QueryID's can only be generated with a central service. Peer-to-Peer networks as JXTA use a random number generator which has a high probability to generate a unique number

- If the query could not be answered, the query is forwarded to a person that has issued the same query (*recommender layer*).
- If the query could still not be answered it is forwarded to the community (*community layer*) and to persons that have a broad knowledge about the network (*bootstrapping layer*).

For this setting we are only interested in exact matches between a content/recommender shortcut and a query, hence we set the similarity threshold $l = 1$. For the querying peer P_Q the query strategy in this setting works as follows:

- **Querying Peer: Forwarding** P_Q searches in its local content provider shortcuts for an exact match to the query. Peers with matching shortcuts are ranked according equation 3.7. The query is forwarded to the top_{CPR} content provider peers. If no content provider peer matches the query, P_Q searches in its local recommender peer index for matching peers and ranks them according equation 3.8. The query is forwarded to the top_{RR} recommender peers. If neither a content provider nor a recommender peer is found the query is routed to the top_C community peers and the top_{BS} bootstrapping peer of P_Q . In every case the forwarded query consists of the query message and the bootstrapping information of P_Q .
- **Remote Peer: Answering Queries** When a remote peer P_R receives a query from P_Q , P_R updates its bootstrapping index with the bootstrapping capability of P_Q received with the query. Next, it will try to answer the query and return the answer and its own bootstrapping information directly to P_Q .
- **Remote Peer: Forwarding Query** Query forwarding at a remote peer P_R is analogue to query forwarding at the querying peer. However, the query includes still the bootstrapping information of P_Q .
- **Querying Peer: Receiving Response** On the arrival of the answers at the querying peer P_Q , it analyzes the result and updates its content provider shortcut index, its recommender index, its community index and its bootstrapping index. Finally it presents the answers to the user.

The routing strategy stops after a query was routed over a maximum number $maxTTL$ of hops.

3.3.4 Setting 2: Query forwarding with Zoomle

Setting 1 works well in the case of an exact match between the query and a shortcut. However, most queries will only match a shortcut with a certain similarity. We have developed an algorithm 1 called *Zoomle*, that allows to forward a query even if no exact shortcut in the local index is found. For this algorithm we add the following metaphors:

- A content provider stores shortcuts to content providers with similar interests.
- A recommender stores shortcuts to recommenders with similar interests.
- A query should at match a shortcut with a minimum similarity threshold.

Analogue to the exact query strategy *Zoomle* searches in the local shortcut index and across the network. However, the algorithm starts with an initial threshold and increments the threshold with each found shortcut.

- **Querying Peer: Set Minimal Threshold** On the querying peer the minimal similarity threshold between the query and a shortcut is set, e.g. $l = 0.5$.
- **Querying Peer: Forwarding** Query forwarding is analogue to the exact query forwarding strategy. Additionally the similarity threshold l for the forwarded query is set above the initial threshold and above of the lowest similarity between a query and a matching shortcut.
- **Remote Peer: Answering Queries** When a remote peer P_R receives a query from P_Q , it will try to find an exact answer to the query and return the answer and its own bootstrapping information directly to P_Q .
- **Remote Peer: Forwarding** Query forwarding is analogue to the exact query forwarding strategy. Additionally the new similarity threshold l is set to the lowest similarity between a query and a shortcut above the old similarity threshold registered in the query.
- **Querying Peer: Receiving Response** Receiving response is analogue to the exact query strategy.

The following algorithm can be applied to both settings, however for setting one the initial threshold is set to $l = 1$ and for setting two to $0 < l < 1$, the maximum TTL is set to seven.

Algorithm 1 *Zoomle*

Require: Query Q, LocalShortCutIndex O**Ensure:** Q.getTTL() < maxTTL

// Update Local Bootstrapping Index

O.updateBoostrappingPeers(Q.getBoostrappingInfo())

// Update Query

Q.incTTL()

Q.addToMessagePath(this.PID)

// Initializing

Queue *selectedPeers* := \emptyset **Threshold** l = Q.getThreshold()

// Start Forwarding

if (| O.getContentProviderPeers(Q,l) | > 0) **then**

//Get matching Content Provider above threshold

selectedPeers.append(O.getContentProviderPeers(Q, l))

// Set New Similarity Threshold

Q.setThreshold(O.getContentProviderPeers(Q,l).getMinThreshold()+0.01)

else if (| O.getRecommenderPeers(Q,l) | > 0) **then**

// Get matching Recommender above threshold

selectedPeers.append(O.getRecommenderPeers(Q,l))

// Set New Similarity Threshold

Q.setThreshold((O.getRecommenderPeers(Q,l)).getMinThreshold()+0.01)

else

// Get Community Nodes

selectedPeers.append(O.getCommunityPeers())

// Get Bootstrapping Nodes

selectedPeers.append(O.getBoostrappingPeers())**end if****Return** *selectedPeers*.

Query Q	Answer from P_1 $\frac{ S_{P_1}^{Q_j} }{p(S_{P_1}^{Q_j})}$	Answer from P_2 $\frac{ S_{P_2}^{Q_j} }{p(S_{P_2}^{Q_j})}$	Answer from P_3 $\frac{ S_{P_3}^{Q_j} }{p(S_{P_3}^{Q_j})}$	Answer from P_4 $\frac{ S_{P_4}^{Q_j} }{p(S_{P_4}^{Q_j})}$	Total $ S_{Q_j} $	$H(Q)$
Semantic Web	100 0,90	10 0,09	0 0,00	1 0,01	111	0,51
Peer-to-Peer	100 0,50	100 0,50	0 0,00	1 0,00	201	1,04
Semantic Web	100 0,62	10 0,06	50 0,31	1 0,01	161	1,25
Peer-to-Peer	100 0,40	100 0,40	50 0,20	1 0,00	251	1,55

Table 3.9: Query Results: Peer P_4 queries the network

Query Q	Message Path $Path$	$P_1.CO$	$P_2.CO$	$P_3.CO$	$P_4.CO$
Semantic Web	$\{P_4, P_1, P_2\}$	0,90	0,09	0,00	0,01
Peer-to-Peer	$\{P_4, P_1, P_2\}$	0,70	0,30	0,00	0,01
Semantic Web	$\{P_4, P_1, P_2\}$	0,65	0,17	0,17	0,01
Peer-to-Peer	$\{P_4, P_1, P_3\}$	0,50	0,30	0,19	0,00
	$\{P_4, P_1, P_2\}$				
	$\{P_4, P_1, P_3\}$				

Table 3.10: Community Measure calculated at peer P_4 given the query hits summarized in table 3.9

Query Q	Message Path $Path$	$P_1.CO^c$	$P_2.CO^c$	$P_3.CO^c$	$P_4.CO^c$	$ S_{P_1}^{Q_j} $	$ S_{P_4}^{Q_j} $	$P_1.CO^q$	$P_4.CO^q$	$P_1.CO$	$P_2.CO$
Semantic Web	$\{P_4, P_1, P_2\}$	0,90	0,09	0,00	0,01	10	100	0,09	0,90	0,99	0,09
Peer-to-Peer	$\{P_4, P_1, P_2\}$	0,70	0,30	0,00	0,01	100	100	0,30	0,70	1,00	0,30
Semantic Web	$\{P_4, P_1, P_2\}$	0,65	0,17	0,17	0,01	60	160	0,33	0,67	0,98	0,17
Peer-to-Peer	$\{P_4, P_1, P_3\}$	0,50	0,30	0,19	0,00	150	250	0,47	0,53	0,97	0,30

Table 3.11: Community Measure calculated at peer P_4 given the query hits summarized in table 3.9

Chapter 4

Conclusion

This document presented a tool for matching of classification hierarchies and an improved tool for ontology-based routing of query messages.

With these tools the number of tools created in the context of SWAP is complete. Summarizing one can say that we developed 9 tools, of which the majority is directly or indirectly used in the SWAP applications Xarop and Bibster. As some of the research work done was only finished lately these have not been implemented in tools yet.

We hope and expect to continue the work begun in the recently started EU-projects SEKT and KnowledgeWeb.

Bibliography

- [AtKvH04] Zharko Aleksovski, Warner ten Kate, and Frank van Harmelen. Semantic coordination: a new approximation method and its application in the music domain. In *Workshop on Meaning Coordination at ISWC 2004*, Hiroshima, Japan, November 2004.
- [BSZ03] P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: a new approach and an application. In *2nd International Semantic Web Conference (ISWO'03)*, Sanibel Islands, Florida, USA, October 2003.
- [Coo04] Brian Cooper. Guiding queries to information sources with infobeacons. In *ACM/IFIP/USENIX 5th International Middleware Conference*, Toronto, 2004.
- [DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, Inc, 1973.
- [FPV⁺98] James C. French, Allison L. Powell, Charles L. Viles, Travis Emmitt, and Kevin J. Prey. Evaluating database selection techniques: A testbed and experiment. In *Research and Development in Information Retrieval*, pages 121–129, 1998.
- [GGM95] Luis Gravano and Héctor García-Molina. Generalizing GLOSS to vector-space databases and broker hierarchies. In *International Conference on Very Large Databases, VLDB*, pages 78–89, 1995.
- [Gon01] Li Gong. Project jxta: A technology overview. Technical report, Sun Microsystems Inc., 2001.
- [Kan99] Gene Kan. Gnutella. In AndyOram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, pages 94–122. O'Reilly, 1999.
- [KGZY02] Vana Kalogeraki, Dimitrios Gunopulos, and Demetris Zeinalipour-Yazti. A Local Search Mechanism for Peer-to-Peer Networks. In *Eleventh International Conference on Information and Knowledge Management (CIKM)*, McLean, VA, 2002.

- [KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the twelfth international conference on World Wide Web*, pages 640–651. ACM Press, 2003.
- [LBM03] Y. Li, Z.A. Bandar, and D. McLean. An Approach for measuring semantic similarity between words using semantic multiple information sources. In *IEEE Transactions on Knowledge and Data Engineering*, volume 15, 2003.
- [SMZ03] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient content location using interest based locality in peer-to-peer system. In *Infocom*. IEEE, 2003.
- [VKMvS04] Spyros Voulgaris, Anne-Marie Kermarrec, Laurent Massoulie, and Maarten van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *FTDCS Workshop*, 2004.