

Adaptive Querying to Knowledge Exchange*

Alessandro Agostini
ITC-IRST Trento

Paolo Avesani
ITC-IRST Trento

Abstract

The integration of the Interactive Querying Algorithm [AA04] into an existing peer-to-peer architecture—called KEx [BBMN03, BBMN02] is proposed. The integration improves the KEx’s functionalities at the level of knowledge peers. In particular, (a) each peer in the role of “the seeker” is provided with the ability to do queries by example, and (b) each peer in the role of “the provider” is provided with the ability to answer them. The interrelation between (a) and (b) is managed by a co-evolutive, selectionist process modeled as a kind of language game.

Keywords: 1. evolution, adaption and learning, 2. agent communication languages and protocols.

1 Introduction

Suppose that meaning conceptualization affects query formulation. For example, this would happen to an user who looks at the names of her data directories to decide the query expression to use. A situation like this is depicted in Figure 1, where the meaning of a query like “Sea Vacation” is modeled as a part of the concept hierarchy (see Section 3 for definition) on the left-hand side, and the answer is defined as the nodes reached by “destination.”

A question then arises out of how query meanings and query evolution and refinements are functional to achieve the query success. In this paper, we investigate this question. For a query to be appropriate, we assume that the user must know something about the information source’s schema. We rise the question in KEx [BBMN03, BBMN02], an innovative peer-to-peer architecture for Knowledge Exchange. In the current version of KEx, it is indeed possible and very common that the meaning of a query, advanced by an user selecting some “seeker” from a set of available peers, is conceptualised and interpreted by the seeker in a even very different way from the potential query solvers.

In this paper, we consider the problem of query answering as a problem of coordination or, more specifically, a problem of *schema matching* (see [RB01] for a survey). For example, “destination” in Figure 1 might be thought of as a matching

*This work was partially supported by the *Provincia Autonoma di Trento* Project EDAMOK (“Enabling Distributed and Autonomous Management of Knowledge”) under deliberation number 1060-4/5/2001. Electronic correspondence to: agostini@irst.itc.it, avesani@irst.itc.it.

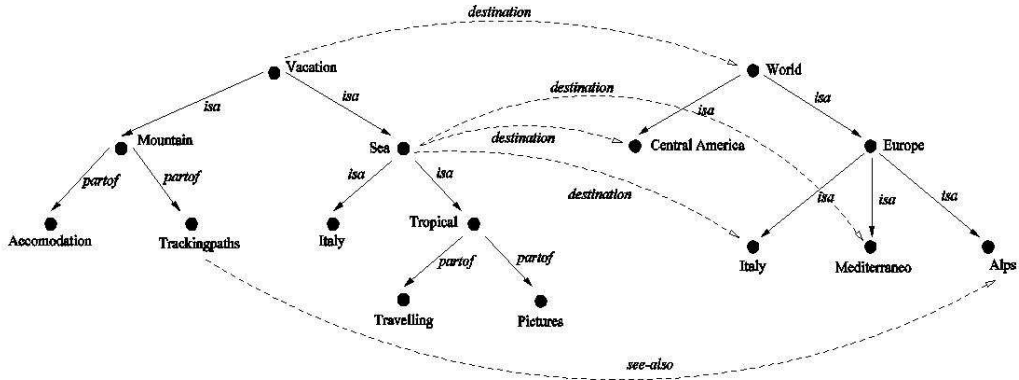


Figure 1: Concept hierarchies and querying.

function, or mapping, between two concept hierarchies [BRHM99], each representing a schema. The domain and codomain of “destination” may be used to express and solve, e.g., a query on the place to go for having sea holidays, or a mountaineering tracking in the European Alps.

We take the experimental conclusion of [MBR01] as a starting point, where it claims that “a robust solution [of the schema matching problem] will need a module to *incrementally learn synonyms and abbreviations* from [linguistic] mappings that are performed over time” (p. 9, our emphasis), and propose a model which integrates a Similar Document Service into KEx.

The model is strongly influenced by the philosophy of language games [Wit53], the methodology and seminal extensive work by Steels on language games (see for instance [SK02] and references cited) Moreover, it takes advantage of some ideas presented in [AA03a, AA03b].

In the next section we present the functionality of the service proposed. We overview KEx thereafter (Section 3). The basic components of the model are presented in Section 4. Section 5 focuses on the algorithm, which is presented as a kind of language game. Related work and the conclusion follow in Section 6 and Section 7.

2 Similar Documents Service

The functionality of the algorithm (or, **service**) of query answering we will present as the main result of this paper (see Section 5) is now illustrated through the use of an example. The service applies to information retrieval and knowledge discovery.

The scenario is that of an user in front of her desktop computer. The user is seeking for solutions to her current question: find some documents similar to that she is now pointing out by the computer’s mouse. The system installed on the computer is like a search engine, say a kind of virtual seeker employed by the user. The system is similar to KEx (Section 3), whose query functionality is to be

extended by the service we are describing, as the main conceptual result of this paper. The system’s interface shows a button, “*Similar Pages & Documents*,” that is used by the user to select the document as “the example” for the search.

The Service provides the user with a document retrieval semi-automatic “strategy”. When the user clicks on the *Similar Pages & Documents* link for a search result—say “give me a set of documents related to document d ”—, after a finite training period the service automatically scouts the system of peers, or a subset of previously selected (available, preferred, etc.) peers, for documents that are related (or, “similar,” in the sense made precise by Definition (6) below) to this result.

The more specialized a document chosen by the user as the example for the query, the fewer results the service will be able to find.

3 KEx Overview

KEx (for “Knowledge Exchange”, see [BBMN03, BBMN02]) is a distributed, knowledge management system implemented as a peer-to-peer architecture, where: “(i) each peer (called K-peer) provides all the services needed to create and organize local knowledge from an individual’s or a group’s perspective, and (ii) social structures and protocols of meaning negotiation are introduced to achieve semantic coordination among autonomous peers” [BBMN03, from the *Abstract*].

The user of KEx may employ a K-peer to play one of two roles: “the seeker” and “the provider.” Intuitively, a peer in the role of the seeker incorporates the functions dealing with knowledge management and query making. These functions allow the user to perform some manipulation of documents, mainly to create new contexts and thereby classifying documents. On the other hand, a A peer in the role of the provider has the additional ability to answer queries. Although a K-peer in KEx has many other functionalities, in this paper we focus on the two main modules of a K-peer, namely: the “query maker” and the “query solver” (cf. [BBMN03, Fig. 1]).

In KEx, “contexts” are the main data structures used to represent local knowledge. Local knowledge is the knowledge available to autonomous agents. KEx provides each individual agent or community with the mechanisms to represent and organize local knowledge according to its goals and interpretative perspective. Contexts are used to this aim. Precisely, contexts are used to conceptualize data according to three main components: an “unique identifier,” or context name; a set of “explicit assumptions,” that provide meta-information about the context (e.g., the context’s owner); and an “explicit representation.” Among possible reference models to explicitly represent a context’s content (e.g., first-order logic, description logics, graphs), KEx used concept hierarchies (see for instance [BRHM99]) also known as terminological ontologies. Since the difference between a context and a concept hierarchy is not relevant for the aim of this paper, from now on we leave out contexts’ names and meta-information, and we refer generally to a “context” as a special kind of concept hierarchy.

A **concept hierarchy** is an undirected, connected, acyclic graph $\langle C, E \rangle$, where C is the finite set of nodes of the graph—each node represents a concept, and E is the set of edges of the graph—each edge represents the relation between two concepts. A concept hierarchy is built from a set L_C of labels to denote concepts, and a set L_E of labels to denote edges. Observe that L_C and L_E are often thought of as disjoint sets.

As we mentioned, a context is a simplified version of a concept hierarchy. Thus, for the aim of this paper we define:

(1) DEFINITION: A **context** is a concept hierarchy $\langle C, E \rangle$ such that (a) each node in C is labelled by a label from language L_C , (b) each node is (or “classifies”) documents from a set \mathcal{D} , and (c) E induces a tree structure over C .

Figure 2 makes clearer the intuitive meaning of a context and shows a possible use of them to query answering. The query’s extensional meaning is the set of documents about Winter holidays in Trentino and Trento (light-lines in the picture), and the answer proposed by the system (represented by the bold arrow in the middle) are the documents about mountain sites in North Italy classified “for vacation during 2004” by the owner of the answering context.

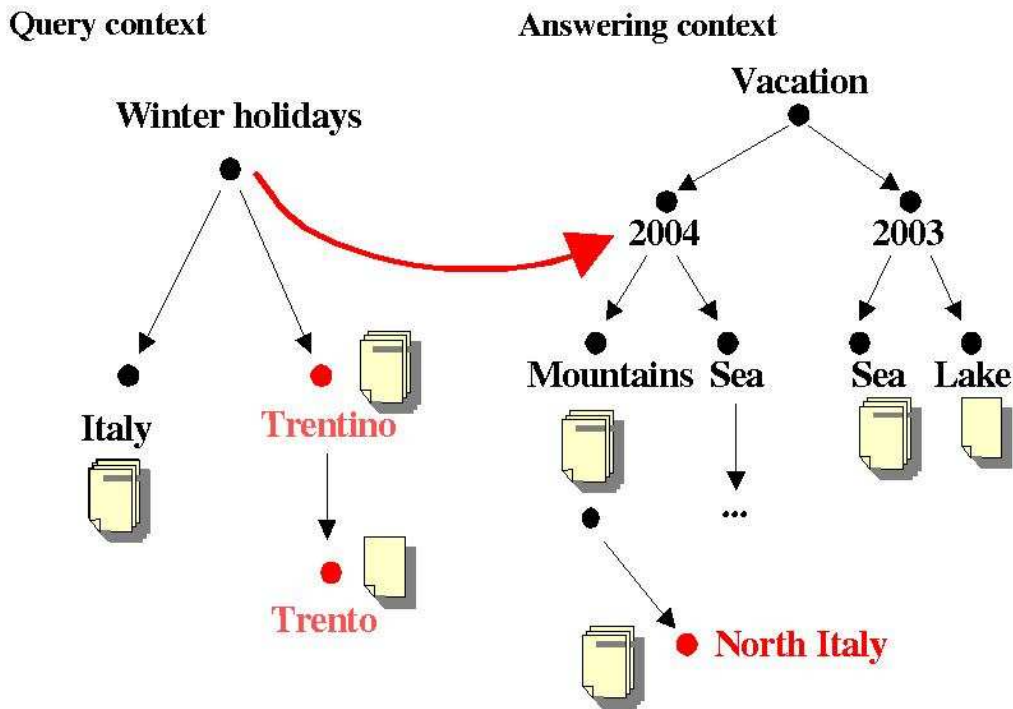


Figure 2: Query answering.

3.1 Concept names: linguistic analysis

Labels used in a context allow a number of linguistic expressions, for example one-word labels (e.g., common and proper nouns, abbreviations, adjectives, etc.), noun phrases (e.g., “Winter holidays,” cf. Figure 2), and many others expressions. We refer to [MSS02] for a thoughtful discussion. For the aim of this paper, however, what matters is the result of a single label analysis. According to this analysis, a label’s “meaning” is mainly obtained by shallow parsing.

A concept label in a context is analysed independently from other contexts or other labels in the same context. The result of applying linguistic analysis to a label l is a linguistic data structure providing the following key information about each “token” contained in l (we follow [MSS02, p. 45]; see there for details): l ’s identification numbers and tokens; morphological analysis (“lemmatization”), so to find all possible normal forms and lexical categories; “Part-of-Speech” (PoS) tagging, so to find the right lexical category for each token in the label; and functional decomposition, to reduce complex labels (e.g., noun and prepositional phrases) to simpler elements.

4 The Model

We image that a multi-agent system is available to an user. (From now we use the terms agent and peer as synonymous.) The user may employ a peer to play the role of the seeker or the provider. The peers have reasoning abilities, whose basic components are illustrated below.

4.1 Documents, contexts, language

The “knowledge domain,” or domain of data we consider is a set of documents—called the **universal set of documents**. We denote this set by \mathcal{D} . One might consider a document as any object provided with a textual content. For example, a document may be a web page, a journal article, any finite string of terms written in a previously specified language. For the aim of this paper, however, what matters is the interpretation of a document, say its semantics, and the local classification of the document, as we will see.

Each peer p has a **document repository**, consisting of a proper subset of the universal set of documents, denoted by: \mathcal{D}_p . Note that the subset is proper because, to the contrary, if p ’s repository and \mathcal{D} coincided, then it would not make sense for p to play the role of the seeker, because no documents to seek would exist in this case.

Each peer p has also a **context repository**, denoted by CX_p . By using contexts, the peer organizes its document repository. A typical example of a context repository is the set of all directories of a file system. Of course, it may be possible that some documents in a peer’s document repository are not classified according to any context. For example, think to some files we share with an user of a different file system. In this case, we can assume that the files are documents

in our repository, but they are classified according to a different context repository by a different user. It follows that classification of documents may be partial, and some documents in a peer’s document repository may be contained in any context—they are, so to say, “unclassified.” We denote by CX the collection of all contexts in the system—called the **universal set of contexts**—, that is, the union of all CX_p for each peer p in the system.

The peers share a minimal communication language \mathcal{L} . Intuitively, the shared language is composed by two different set of elements. The first set comes from context labels, that is, from some natural languages (e.g., “English,” “Italian”) or some combination thereof. As KEx, we assume that natural language in use is specified by looking at some meta-information in peers’ contexts. The second set of elements of communication language \mathcal{L} is defined according to documents. Let $\text{Text}(\cdot)$ be a text extraction function. We refer to the keywords extraction function of [TRPG03, Sec. 4]. Given a document d , $\text{Text}(d)$ lists all the keywords in d , precisely, the most frequent “bigrams” that were found in a list of frequent (Italian) bigrams. Thus, $\text{Text}(\cdot)$ combines simple statistical measures with elementary linguistic knowledge. Applied to a document d , $\text{Text}(\cdot)$ produces a set $\text{Text}(d)$ of expressions. We call $\text{Text}(d)$ the (textual) **semantics of d** .

To define minimal communication language \mathcal{L} , we fix some more notation. Given a natural language expression l , to be thought of as the **concept name** of some peer’s context node, we denote by $\text{sp}(l)$ (“sp” for “shallow parsing”) the linguistic data structure obtained by applying to l the procedures of linguistic analysis introduced in subsection 3.1. Given a context \mathcal{C} , we denote by $\text{sp}(\mathcal{C})$ the set of linguistic data structures obtained by applying $\text{sp}(\cdot)$ to each concept name l in \mathcal{C} . We are now ready to define minimal communication language \mathcal{L} as a combination of linguistic information from concept labels and textual expressions from documents.

(2) **DEFINITION:** Let universal set of documents \mathcal{D} , universal set of contexts CX , $\text{Text}(\cdot)$ and $\text{sp}(\cdot)$ be given. We define $\mathcal{L} = \text{Text}(\mathcal{D}) \cup \text{sp}(CX)$, where $\text{Text}(\mathcal{D}) = \bigcup_{d \in \mathcal{D}} \text{Text}(d)$ and $\text{sp}(CX) = \bigcup_{\mathcal{C} \in CX} \text{sp}(\mathcal{C})$.

Each peer p in the system may have an additional set of “words” to express autonomous concepts and queries out of language \mathcal{L} . These unspecified words are added to \mathcal{L} . The resulting union set defines p ’s language, denoted by \mathcal{L}_p . We note that additional words as those mentioned here play a central role in motivating our adaptive approach, where each peer may create new words and learn them from other peers using the new words in conversation.

4.2 Features

A peer manages and classifies its documents by using “features” and preferences over its “lexicon” (see subsection 4.5). Intuitively, a feature of a document is a pair composed by a (key)word in the document, to be thought of as a standardized text from procedure $\text{Text}(\cdot)$ of text extraction, and a relative value. For example, think to the frequency of occurrence of that word in the document. Following

[TRPG03], given a document d we rank a word $w \in \text{Text}(d)$ on the basis of a slightly better method than just counting the number of occurrences of the word in the document. More precisely, we use an equivalent formulation over contexts to the inverse document frequency [MS99].

Let context $\mathcal{C} = \langle C, E \rangle$ (cf. Definition (1)) and document $d \in \mathcal{D}$ be given. We write “ $d \in \mathcal{C}$ ” to mean that there is a concept $c \in C$ such that $d \in c$.

(3) DEFINITION: Let peer p , context $\mathcal{C} \in CX_p$ and word $w \in \text{Text}(d)$ for document $d \in \mathcal{D}$ be given. We define the **weight of w in d for p in \mathcal{C}** as follows.

$$W_p[w, d, \mathcal{C}] = N[w, d] \cdot \log \frac{\text{Card}(\mathcal{C})}{\text{doCX}_p[w]},$$

where $N[w, d]$ is the total number of occurrences of w in d , $\text{Card}(\mathcal{C})$ is the number of documents in \mathcal{C} , and $\text{doCX}_p[w]$ is the total number of documents in \mathcal{C} containing w . Note that $W_p[w, d, \mathcal{C}]$ is a real number, so $W_p[\cdot, \cdot]$ is a function from $\mathcal{L} \times \mathcal{D}$ to R .

$$\log \frac{\text{Card}(\mathcal{C})}{\text{doCX}_p[w]}$$

is called **context-inverse document frequency of w for p** , in symbols: $\text{IDF}_p[w, \mathcal{C}]$.

The intuitive meaning of $\text{IDF}_p[\cdot]$ is that terms which rarely occur over the collection of documents in the peer p 's chosen context are the most relevant. The importance of each term is assumed to be inversely proportional to the number of documents in the context that contain the term. Since $\text{IDF}_p[\cdot]$ represents word specificity in the view of a peer p , it is expected to improve the precision of p 's classification of documents.

We are now ready to define the “features of a document for a peer.” From now on, the collection of documents classified under a context \mathcal{C} is denoted by $\text{domain}(\mathcal{C})$. We say that $\text{domain}(\mathcal{C})$ is the **domain** (or “textual content”) of \mathcal{C} . In short, $\text{domain}(\mathcal{C})$ is the union set of \mathcal{C} 's nodes.

(4) DEFINITION: Let peer p , context $\mathcal{C} \in CX_p$ and document $d \in \mathcal{D}$ be given. The **features set of d for p in \mathcal{C}** is the set $F_{p,d,\mathcal{C}} = \{(w, v) \mid w \in \text{Text}(d), v = W_p[w, d, \mathcal{C}]\}$. The **features set for p** is the set $F_p = \bigcup_{\mathcal{C} \in CX_p, d \in \text{domain}(\mathcal{C})} F_{p,d,\mathcal{C}}$.

We call the pair (w, v) a **feature** (of d for p in \mathcal{C}). We say that w is the attribute (or name) of the feature (w, v) , and that v is its value. Some features are “distinctive.” Informally, features are distinctive when refer to the same attribute with different values. We rely on the following definition.

(5) DEFINITION: Let features $(w_1, v_1), (w_2, v_2)$ for a peer p be given. We say that (w_1, v_1) and (w_2, v_2) are **distinctive** if $w_1 = w_2$ and $v_1 \neq v_2$.

We will use distinctive features to distinguish documents.

4.3 Similarity

Documents may be organized by the peers in “clusters.” In particular, we have seen that documents may be aggregated by individual peers according to general schemas, or contexts, which are applied by each peer p locally to classify its document repository \mathcal{D}_p . Informally, a cluster is a set of similar documents. To define a cluster formally, we rely on the following terminology and notation.

Given peer p and document $d \in \mathcal{D}_p$, let V_d denote the finite sequence over real numbers—we call it **feature vector**, obtained by listing the value of each feature of d for p . That is, given $d \in \mathcal{D}_p$, we define feature vector $\langle v_1, v_2, \dots, v_k \rangle$, where v_i for $i = 1, 2, \dots, k = \text{Card}(F_{p,d})$ is the value of the i th feature of d for p .

For all peers in the system, we now define $\text{Sim}^p(\cdot, \cdot)$ to be a similarity measure defined over “normalized feature vectors.” To normalize two feature vectors, we order the attributes of each feature in each vector according to the same ordering relation, and extend the shorter vector by adding zeros. (See references in [TRPG03].)

(6) DEFINITION: Let peer p and documents $d, d' \in \mathcal{D}$ be given. We define

$$\text{Sim}^p(d, d') = \frac{\langle V_d, V_{d'} \rangle}{\|V_d\| \cdot \|V_{d'}\|},$$

where $\langle V_d, V_{d'} \rangle$ denotes the result of the scalar product of normalized feature vectors V_d and $V_{d'}$, and $\|V_d\| \cdot \|V_{d'}\|$ is the product of the vectors’ norm.¹ (See also [TRPG03, Sec. 5].)

Note that function $\text{Sim}^p(\cdot, \cdot)$ is partial recursive. We say that documents d, d' have similar content, or that are **similar for** peer p just in case $\text{Sim}^p(d, d') \neq 0$. We say documents d, d' are **similar** if d, d' are similar for some peer. Note that $\text{Sim}^p(d, d') \in [0, 1]$.

4.4 Distinctive feature sets and clusters

A “distinctive feature set” is a set of features distinguishing a document from a set of other documents. We formally define a “distinctive feature set of a document for a peer with respect to a set of documents,” in symbols: $D_{p,d}^C$ as follows. (Recall that $\text{domain}(\mathcal{C})$ denotes the union set of context \mathcal{C} ’s nodes.)

(7) DEFINITION: Let peer p , document $d \in \mathcal{D}$, context $\mathcal{C} \in CX_p$, set of documents $C \subseteq \text{domain}(\mathcal{C})$ and features set $F_{p,d,\mathcal{C}}$ be given. We define:

$$D_{p,d,\mathcal{C}}^C = \{(w, v) \in F_{p,d,\mathcal{C}} \mid \forall d' \in C, \nexists (w', v') \in F_{p,d',\mathcal{C}} \\ w' = w \text{ or } \exists (w, \tilde{v}) \in F_{p,d',\mathcal{C}} \tilde{v} \neq v\}.$$

In particular, note that $D_{p,d,\mathcal{C}}^C \subseteq F_{p,d,\mathcal{C}}$, $D_{p,d,\mathcal{C}}^{\{d\}} = \emptyset$, and $D_{p,d,\mathcal{C}}^\emptyset = F_{p,d,\mathcal{C}}$.

¹Recall that $\|V\| = \text{Sqr}(\langle V, V \rangle)$.

Of course, a distinctive feature set is a feature set. Thus, the elements of distinctive feature set $D_{p,d,\mathcal{C}}^C$ are all features suitable by the peer to distinguish a document d from *other* documents in a subset C of a context's domain.²

We now are ready to define clusters of documents in a context by using distinctive features. Given a peer p and a context $\mathcal{C} \in CX_p$, we denote by \overline{C}_p the **relative complement** $domain(\mathcal{C}) \setminus C$ of a subset C of $domain(\mathcal{C})$.

(8) DEFINITION: Let peer p and context $\mathcal{C} \in CX_p$ be given. A **cluster of p in \mathcal{C}** is the set $C \subseteq domain(\mathcal{C})$ such that for all $d, d' \in C$, the distinctive feature sets of d and d' for p with respect to \overline{C}_p are defined, and

$$D_{p,d,\mathcal{C}}^{\overline{C}_p} = D_{p,d',\mathcal{C}}^{\overline{C}_p}.$$

A question now is how clusters relate to contexts. An answer is provided by the next theorem.

(9) THEOREM: Let peer p and context $\mathcal{C} \in CX_p$ be given. Suppose that $\{c_1, c_2, \dots, c_n\}$ is the classification of $domain(\mathcal{C})$ defined by (the nodes of) \mathcal{C} . Then, for all $i = 1, 2, \dots, n$, c_i is a cluster of p in \mathcal{C} if and only if

$$c_i = \{d \in domain(\mathcal{C}) \mid \forall d' \in \bigcup_{j \neq i} c_j, \exists (w', v') \in F_{p,d',\mathcal{C}} \\ (w', \cdot) \in F_{p,d,\mathcal{C}} \text{ or } \exists (w, v) \in F_{p,d',\mathcal{C}} \text{ and } (w, v) \neq F_{p,d,\mathcal{C}}\}.$$

The proof of the theorem follows directly from Def. (7) and (8). We call the classification characterized by the theorem above **canonical**. From now to the end of this paper, we consider only **canonical contexts**, that is, contexts that induce a canonical classification of their textual domain.

We use distinctive feature sets to label clusters or, by Theorem (9), to introduce new concept names in canonical contexts. Definition (8) has an important consequence on this, as it allows us to define a way to assign a **recommended name** to a cluster by using preferences over documents' distinctive features. To illustrate, we need to define "preferences." Let $pow(X)$ denote the power set of X .

Given a peer p , let $Pref^p(\cdot, \cdot)$ be a partial function from $\mathcal{L}_p \times (pow(F_p) \cup pow(\mathcal{D}_p))$ to integers (denoted by Z). Positive, negative and neutral preferences are possible. For $n \in Z$, $Pref^p(w, X) = n$ means that peer p denotes the set X of either documents or features by linguistic expression w with preference n .

We note that, by the foregoing definition of clusters and of preferences, a peer's preferences over features that distinguish a document from the complement of a cluster do not depend on the choice of the document. So, the next definition gives us a sound way to decide how to name a cluster.

²If $d \in C$ then the features that define $D_{p,d,\mathcal{C}}^C$ are exactly those that define $D_{p,d,\mathcal{C}}^{C \setminus \{d\}}$.

(10) DEFINITION: Let peer p , cluster C of p in context $\mathcal{C} \in CX_p$ and word $w \in \mathcal{L}_p$ be given. We define:

$$\text{Pref}^p(w, C) = \text{Pref}^p(w, \overline{D_{p,d}^{C_p}}),$$

if there is $d \in C$ such that $\text{Pref}^p(w, \overline{D_{p,d}^{C_p}})$ is defined, and $\text{Pref}^p(w, \emptyset) = 0$. Otherwise, $\text{Pref}^p(w, C)$ is undefined. Observe: if \mathcal{C} is canonical then $\text{Pref}^p(w, C)$ is always defined for all d in every node C of the context.

We say that C is **labelled with w by p** just in case $\text{Pref}^p(w, C)$ is maximum, that is, $\text{Pref}^p(w', C) \leq \text{Pref}^p(w, C)$ for every $w' \in \mathcal{L}_p$ with $w' \neq w$.

(11) *Example:* Consider the two contexts depicted in Figure 2. Suppose that they are both canonical. The seeker s , as the owner of the left-hand side context, would prefer to name the first right-hand side node from the root by the linguistic data structure sp (“Trentino”). The seeker’s preference is $\text{Pref}^s(\text{sp}(\text{“Trentino”}), C)$, where C is a cluster by definition, since we have canonical contexts. C contains all or some of documents the seeker has in his repository about Trentino. The seeker’s preference to use label “Trentino” is maximum over all possible other (linguistic data structures for) labels available in his language. It is assumed that there is at least a document in the cluster “Trentino,” and this document is well discriminated (distinguished) from all other documents in the context.

4.5 Lexicon

Each agent has a “lexicon.” Given a peer p , we denote the lexicon of p by Lx_p . Informally, a peer’s lexicon is a set of pairs, where the first component is a linguistic element build from the peer’s language, and the second component is a feature set for the peer in some context. Formally:

(12) DEFINITION: The **lexicon of a peer p** is the set of pairs (w, F) for all $w \in \mathcal{L}_p$ and all $F \subseteq F_p$ such that $\text{Pref}^p(w, F)$ is defined.

A lexicon may be either empty or incomplete, that is, there may be some words with no associated feature set or some feature set with no associated word. Notice that there is not one lexicon for all peers, but each peer has its own local lexicon.

4.6 Query answering functions

Each peer p in the system has two functions, denoted by $query(\cdot, \cdot)$ and $answer(\cdot, \cdot)$. Informally, $query(\cdot, \cdot)$ provides a peer in the role of the seeker with the way to translate the local knowledge about a document to a language suitable to express the query and to interpret it. Since language interpretation is local to each peer, a question is how the query language eventually provides an external resource for the suitable information to interpret the query. For example, think of an external resource as the provider employed by a seeker to solve the query. For simplicity, from now on we assume that each peer has a *finite* language.

(13) DEFINITION: Let peer p , context $\mathcal{C} \in CX_p$ and document $d \in \text{domain}(\mathcal{C})$ be given. We define $\text{query}(F_{p,d,\mathcal{C}}, Lx_p)$ to be the finite sequence $\langle w_1, w_2, \dots, w_n \rangle$ over \mathcal{L}_p such that $(w_i, F_{p,d,\mathcal{C}}) \in Lx_p$ ($i = 1, 2, \dots, n$), and for all $i, j \leq n$, if $i \leq j$ then $\text{Pref}^p(w_i, F_{p,d,\mathcal{C}}) \geq \text{Pref}^p(w_j, F_{p,d,\mathcal{C}})$.

We call the finite sequence $\langle w_1, w_2, \dots, w_n \rangle$ **query expression**. Informally, by using $\text{query}(F_{p,d,\mathcal{C}}, Lx_p)$ the peer p exploits and, successively by separate action, communicates existing high-level knowledge of the user about document d in context \mathcal{C} to the selected query solver. This high-level knowledge is represented by $F_{p,d,\mathcal{C}}$ and Lx_p and is local to the peer.

Clusters' names and function $\text{query}(\cdot, \cdot)$ are employed by peers to convey clusters' meaning into a **query language**. The query language built by a peer p is defined as the set of tuples of the kind in Definition (13). How a cluster's name may be used to export some of the cluster's meaning to a query is the content of the next theorem.

(14) THEOREM: Let peer p , nonempty cluster C of p in context $\mathcal{C} \in CX_p$ be given. Suppose that C is labelled with word $w \in \mathcal{L}_p$ by p . There is a document $d \in C$ such that the first element of the finite sequence

$$\text{query}(D_{p,d,\mathcal{C}}^{\overline{C_p}}, Lx_p)$$

is equal to w .

To prove the theorem, let C be the cluster of the theorem and suppose that C be labelled with w by p . Then there is $d \in C$ such that $\text{Pref}^p(w, D_{p,d,\mathcal{C}}^{\overline{C_p}})$ is defined, hence $(w, D_{p,d,\mathcal{C}}^{\overline{C_p}}) \in Lx_p$ and $\text{query}(D_{p,d,\mathcal{C}}^{\overline{C_p}}, Lx_p)$ is defined. Moreover, $\text{Pref}^p(w, D_{p,d,\mathcal{C}}^{\overline{C_p}})$ is maximum over \mathcal{L}_p , so $\text{query}(D_{p,d,\mathcal{C}}^{\overline{C_p}}, Lx_p)$ is of the form $\langle w, \dots \rangle$.

Informally, $\text{answer}(\cdot, \cdot)$ allows a peer in the role of the provider to interpret a query, according to the provider's lexicon. We rely on the following definition. Let Λ denote the set of peers in the system.

(15) DEFINITION: Let peer p and finite sequence $\langle w_1, w_2, \dots, w_n \rangle$ over $\bigcup_{q \in \Lambda} \mathcal{L}_q$ be given. We define $\text{answer}(\langle w_1, w_2, \dots, w_n \rangle, Lx_p)$ to be the set $\{F_p^1, \dots, F_p^k\}$ of feature sets F_p^j for p such that for all $i = 1, 2, \dots, n$, $(w_i, F_p^j) \in Lx_p$.

So, each w_i is used by peer p as the input to find (“guess”) the cluster (if any) suitable to answer to the query expression $\langle w_1, w_2, \dots, w_n \rangle$.

We conclude this section with a remark.

(16) *Remark:* In contrast to function $\text{query}(\cdot, \cdot)$, which applies to single peers' local information to produce global information—that is, a query language that each other peer may interpret, even with different interpretations, sometimes also inconsistent each others—the function $\text{answer}(\cdot, \cdot)$ applies to global information to produce features that each peer can interpret and compare locally to features

of documents in the peer’s document repository. Importantly, the composition of $query(\cdot, \cdot)$ and $answer(\cdot, \cdot)$, that is, $answer(query(F_{p,d,\mathcal{C}}, Lx_p), Lx_q)$ defines a mapping for schema matching between peers p and q (see the Introduction).

5 Adaptive Querying

In normal operation, the user selects “the seeker” from the system of peers. The seeker selects one or more providers as the target peers for a query. The user, the seeker and the target provider(s) continuously go through a loop performing the following actions:

- Step 1. A context \mathcal{C} is selected by the user according to the query. The context classifies in a fixed way the documents in the seeker’s document repository. Intuitively, only the documents in the textual domain of the selected context are currently in the field of attention of the user. Without loss of generality, however, we assume $domain(\mathcal{C}) = \mathcal{D}_u = \mathcal{D}_s$ ($u =$ the user, $s =$ the seeker).
- Step 2. One document d in the context’s domain is chosen by the user according to the query to perform. In the hypothesis that context \mathcal{C} is canonical, this document determines the **query cluster**.³ The document may be thought of as “the example” of the user’s query.
- Step 3. The selected cluster C in \mathcal{C} is labelled by the seeker with a word w in the seeker’s language. The seeker’s action is performed with maximum preference $Pref^s(w, C)$. Intuitively, w is the recommended name of the cluster provided (with preference $Pref^s(w, C)$) by the seeker to the user.
- Step 4. The seeker sends to the target provider(s) the feature set $F_{s,d,\mathcal{C}}$ of the example and the sequence of words $\langle w_1, w_2, \dots, w_n \rangle$ (“query expression”) produced by applying function $query(\cdot, \cdot)$ to the seeker’s “knowledge” about d . Precisely, $\langle w_1, w_2, \dots, w_n \rangle = query(F_{s,d,\mathcal{C}}, Lx_s)$. The query expression is a kind of recommended query to the user.
- Step 5. An attempt is made by the target provider(s) to find possible features distinguishing the example from the set of all documents in its document repository. In other words, an attempt is made by the target provider(s) to find possible distinctive feature sets on the basis of $F_{s,d,\mathcal{C}}$ and the received query expression. The question the provider p poses is, roughly: “there are a context $\mathcal{C}' \in CX_p$, a cluster C' of p in \mathcal{C}' , some document(s) $d' \in domain(\mathcal{C}')$ and a distinctive feature set $D_{p,d',\mathcal{C}'}^{domain(\mathcal{C}') \setminus C'}$ such that $D_{p,d',\mathcal{C}'}^{domain(\mathcal{C}') \setminus C'} \neq \emptyset$ and $D_{p,d',\mathcal{C}'}^{domain(\mathcal{C}') \setminus C'} \in answer(\langle w_1, w_2, \dots, w_n \rangle, Lx_p)$?” If the answer is yes,

³This algorithm is defined only for queries from canonical contexts. As a consequence, a nonempty distinctive feature set suitable to form a cluster of documents exists. Otherwise, we may think to extend the model to allow more features that those of the form (w, v) we have considered in this paper.

then C' is the **answering cluster** of the provider that hopefully matches with the query cluster C of the seeker.

- Step 6. The provider p sends document(s) $d' \in \mathcal{D}_p$ to the seeker, who evaluates it against the example d by using $\text{Sim}^s(d, d')$. If the provider's result is relevant, that is, $\text{Sim}^s(d, d') > \delta$ for some fixed, user dependent relevance parameter δ , preferences over the peers' lexicon are updated to success. Otherwise, preferences over the peers' lexicon are updated to failure.

On the basis of the foregoing steps, that we refer to as the rules of the **query-answering game** $\mathcal{G} = \langle u, s, p, d, \mathcal{C} \rangle$, a number of query formation or query refinement situations arise. (In the following, C denotes the query cluster produced in Step 2.)

1. *The seeker is unable to express the example d by words* (Step 4 fails): Either the seeker's lexicon Lx_s is not developed enough or the seeker's naming function $query(\cdot, \cdot)$ is undefined over $(D_{s,d,\mathcal{C}}^{\text{domain}(\mathcal{C}) \setminus C}, Lx_s)$. The seeker (possibly by interacting with the user) then creates a new element $(w, D_{s,d,\mathcal{C}}^{\text{domain}(\mathcal{C}) \setminus C})$ in its lexicon with preference $\text{Pref}^s(w, D_{s,d,\mathcal{C}}^{\text{domain}(\mathcal{C}) \setminus C}) = 0$.
2. *The provider is unable to interpret the query from the seeker* (Step 5 fails): Either the provider is unable to find possible features distinguishing the example from the set of all documents in its database or the provider does not understand the query expression from the seeker. In the first case, the game ends in failure—the provider's response is an empty set of documents together with a message error.⁴ In the second case—suppose that the provider has found at least a set of distinguishing features, the provider extends its lexicon and creates associations between the seeker's query expression and each suitable set of features. More precisely, the following actions are done. (Let $\langle w_1, w_2, \dots, w_n \rangle$ be the query expression from the seeker and let D denote a set of distinguishing features found by the provider (Step 5). Observe that such set depends on d' and C' , so many distinguishing features sets may be found.) Then, for all $i = 1, \dots, n$:
 - the provider adds the new element D to its lexicon with fixed preference:

$$\text{Pref}^p(w_i, D) = 0.$$

3. *The seeker is able to send a query expression for the example and the provider is able to interpret it* (Step 6; no linguistic failures arise): We distinguish two cases, that are also the possible outcomes of the query-answering game \mathcal{G} .

⁴At this stage of the game, it would be possible for the provider to create new features to distinguish the example of the seeker from the provider's documents. Our model development at now, however, does not allow to do this. We have planned to extend our model to deal with features out of the form (w, v) . We leave it for future work.

- (a) *The provider’s response is compatible (“relevant”) with the seeker’s query.* This means that $\text{Sim}^s(d, d') > \delta$ for a fixed, previously defined, user dependent relevance parameter δ . The query succeeds, the game ends in success. Note that it is possible, indeed probable at the beginning of the game, that the query cluster and the answering cluster are labelled with different words and that the seeker and the provider(s) use different feature sets. The seeker increases the value of its lexicon preferences over the winning associations $(w_i, D_{s,d,\mathcal{C}}^{\overline{C_s}})$ and decreases lexicon preferences over competing associations $(w, D_{s,d,\mathcal{C}}^{\overline{C_s}})$ for all words w not present in the query expression. The seeker’s motivation to adapt preferences is to update cluster label. Similarly, the provider increases lexicon preferences over winning associations $(w_i, D_{p,d',\mathcal{C}'}^{\overline{C'_p}})$ and decreases preferences of competing associations $(w_i, D_{p,d',\mathcal{C}'}^{\overline{C'_p}})$ in its lexicon.
- (b) *The provider’s response is not compatible with the seeker’s query.* This means that $\text{Sim}^s(d, d') \leq \delta$. The query fails, the game ends in failure.

6 Related Work

The model presented in this paper refines and extends previous work [AA04] along two main directions. First, the model represents explicitly the KEx contexts. The relation between clusters and the classification defined by a context is made explicit. This has a major consequence on the implementation in KEx of the Similar Document Service. In technical report [AA04] such link has not been made explicit.

In the area of multiagent systems our work relates—even if sometimes only in the problem it addresses rather than in formal development—, to semantics and pragmatics of interaction, for example [PCd03] and work on query formulation and conversations in Web technologies (see for instance [SH03, AGP03], respectively).

Our approach to query answering is related to clustering of search results, as performed by Northern Light (*www.northernlight.com*), for example. Northern Light dynamically clusters search results into categories previously defined. These may be changed by new interactions to the user, who may narrow search results to any of these categories. More generally, there is a growing tendency of clustering techniques being used in web search (the system user - search engine is a query answering system, although not always cooperative!) to present the user with some query refinement options alongside the matching documents, that is, the documents supposed by the engine/provider to be relevant to the user. Other examples are Teoma (*www.teoma.com*) and AltaVista (*www.altavista.com*). However, an inherent problem of these techniques is naming of the clusters—that is, to select the right word or label to convey the cluster’s meaning or to select useful expressions that can serve as query refinement terms. A recent work [Kru03]

studies this problem in the specific case of document collections being searched are domain-specific or limited in size.

7 Conclusion

We have argued that query formulation is basically influenced by the language usage and it is not distinct from the categorization process behind query answering. To the aim of categorization of a query's meanings, we have used clusters to form canonical contexts. In order to investigate the dynamic refinement of a query and query flexibility, we have advanced a model in which queries and answers are a consequence of language evolution and meaning construction. The result of this paper is an adaptive algorithm aimed at identifying, after a finite number of steps and appropriate feedback from the user, potentially relevant answers with respect to an user's query. The algorithm defines a kind of mapping in the sense of schema matching. The mapping is the result of combination of query and answering as formally defined by functions *query*(\cdot, \cdot) and *answer*(\cdot, \cdot).

In summary, we have presented an integration of the Interactive Querying Algorithm [AA04] into an existing peer-to-peer architecture, KEx. The integration improves the KEx's functionalities at the level of knowledge peers in their main components of query maker and query solver. The inter-relation between queries and answers has managed by a co-evolutive, selectionist process modeled as a kind of language game.

References

- [AA03a] A. Agostini and P. Avesani. Advertising games for web services. In R. Meersman, Z. Tari, and D. Schmit, editors, *Eleventh International Conference on Cooperative Information Systems (CoopIS-03)*, pages 93–110, Berlin Heidelberg, 2003. Springer-Verlag LNCS 2888. Electronically available at <http://sra.itc.it/people/agostini/>.
- [AA03b] P. Avesani and A. Agostini. A peer-to-peer advertising game. In M. Orlowksa, M. Papazoglou, S. Weerawarana, and J. Yang, editors, *First International Conference on Service Oriented Computing (ICSOC-03)*, pages 28–42, Berlin Heidelberg, 2003. Springer-Verlag LNCS 2910.
- [AA04] A. Agostini and P. Avesani. On the discovery of the semantic context of queries by game-playing. Technical report, ITC-IRST, Trento, Italy, January 2004.
- [AGP03] L. Ardissono, A. Goy, and G. Petrone. Enabling conversations with Web Services. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 819–826, New York, NY, 2003. ACM Press.

- [BBMN02] M. Bonifacio, P. Bouquet, G. Marni, and M. Nori. KEx: A Peer-to-Peer solution for distributed knowledge management. In *Proceedings of the Fourth International Conference on Practical Aspects of Knowledge Management (PAKM-02)*, pages 490–500, Heidelberg, 2002. Springer-Verlag LNAI 2569.
- [BBMN03] M. Bonifacio, P. Bouquet, G. Marni, and M. Nori. Peer - mediated distributed knowledge management. In *Proceedings of AAAI Spring Symposium on Agent Mediated Knowledge Management (AMKM-03)*, Stanford, CA, 2003. Stanford University.
- [BRHM99] A. Büchner, M. Ranta, J. Hughes, and M. Mäntylä. Semantic information mediation among multiple product ontologies. In *Proceedings of the 4th World Conference on Integrated Design and Process Technology (IDPT-99)*, 1999.
- [Kru03] U. Kruschwitz. An adaptable search system for collections of partially structured documents. *IEEE Intelligent Systems*, 18(4):44–52, 2003.
- [MBR01] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In P. M. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB-01), September 11-14, 2001, Roma, Italy*, pages 49–58. Morgan Kaufmann, 2001.
- [MS99] C. D. Manning and H. Schtze, editors. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, 1999.
- [MSS02] B. Magnini, L. Serafini, and M. Speranza. Linguistic based matching of local ontologies. In P. Bouquet and M. Warglien, editors, *Working Notes of the AAAI-02 Workshop on Meaning Negotiation*, Technical Report WS-02-09, pages 42–50, Menlo Park, CA, 2002. AAAI Press.
- [PCd03] P. Pasquier and B. Chaib-draa. The cognitive coherence approach for agent communication pragmatics. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 544–551, New York, NY, 2003. ACM Press.
- [RB01] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
- [SH03] G. L Somlo and A. E. Howe. Using web helper agent profiles in query generation. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 812–818, New York, NY, 2003. ACM Press.

- [SK02] L. Steels and F. Kaplan. AIBO's first words. The social learning of language and meaning. *Evolution of Communication*, 4(1):3–32, 2002.
- [TRPG03] P. Tonella, F. Ricca, E. Pianta, and C. Girardi. Using keyword extraction for web site clustering. In Ken Wong, editor, *Fifth International Workshop on Web Site Evolution (WSE-03)*, pages 41–48, Amsterdam, The Netherlands, September 22, 2003. IEEE Computer Society.
- [Wit53] L. Wittgenstein. *Philosophical Investigations*. Blackwell, Oxford, UK, 1953.