Interoperability through Emergent Semantics. A Semiotic Dynamics Approach.

Luc Steels[1],[2] and Peter Hanappe[1]

[1] Sony Computer Science Laboratory, Paris, France
[2] Vrije Universiteit Brussel, Brussels, Belgium
steels@arti.vub.ac.be, hanappe@csl.sony.fr

Abstract. We study the exchange of information in collective information systems mediated by information agents, focusing specifically on the problem of semantic interoperability. We advocate the use of mechanisms inspired from natural language, that enable each agent to develop a repertoire of grounded categories and labels for these categories and negotiate their use with other agents. The communication system as well as its semantics is hence emergent and adaptive instead of predefined. It is the result of a self-organised semiotic dynamics where relations between data, labels for the data, and the categories associated with the labels undergo constant evolution.

1 Introduction

Many interactive information systems such as web browsers typically allow a user to taxonomically structure data and associate tags with this taxonomy. In the case of web browsers, the data consists of URLs to web pages, the taxonomy is the hierarchy of bookmark folders, and tags are names that users associate with folders and subfolders. We call the taxonomy created and maintained by a user the "owner taxonomy" and its tags the "owner tags". The taxonomy implies a particular way of categorising the data so that there is in fact a semiotic relation between data, tags, and categories, forming a semiotic triad (see figure 1). The semantics of the tags (i.e. the meaning of the categories) is usually not explicitly defined. We could either do this by defining the logical dependencies between categories (formal semantics), or by defining classifiers, i.e. computable functions capable of deciding whether the category applies to a data item or not (grounded semantics).

Categorisations and tagging by users is based on cognitive processes which are not accessible to information systems, and may not even be consciously known by the users themselves. For example, a user may decide to put all the songs he likes in one folder and the ones he does not like in another. This categorisation is completely subjective and can never be automated nor emulated by a machine. Similarly user tags often use natural language words but this is only suggestive and not necessarily accurate nor rational. For example, a user may have a folder tagged 'New York' but it could contain pictures of New York, pictures taken in New York, pictures taken while living in New York, etc.

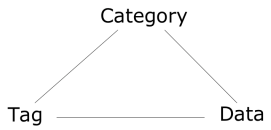

Fig. 1. Data, tags, and categories associated with tags form a semiotic relation. The paper proposes a system whereby agents autonomously establish such relations and coordinate them with others.

The taxonomies and tags of a user are usually local and private and this is unproblematic as long as there is no exchange between them. But there is now a rapidly growing number of collective information systems where users want to exchange data with each other, and they are therefore necessarily confronted with the problem that the taxonomy and tags imposed by one user are not necessarily the same as those of another. Moreover the information systems may be heterogenous in the sense that the conceptual schemata used by one information system for storing data and meta-data may be quite different from those used by another information system.

One type of collective information system are peer-to-peer systems which allow direct information exchange between peers without the need to go through a central server. Well-known examples for music file sharing are systems like Gnutella, Kazaa, or eMule that are used by millions of people today. Similar sharing networks are growing for movies or game software. Also in the domain of scientific data or educational materials, there are growing networks of peer-to-peer shared systems [8]. Another type of collective information systems form websites which encourage social sharing of data by allowing users to upload data and introduce tags for these data. Examples include www.flickr.com for exchanging pictures, www.citeulike.com for exchanging scientific papers, and del.icio.us for exchanging information about websites. Although these systems are not peer-to-peer in the strict sense, because they are managed from central servers, they nevertheless are highly distributed and the taxonomy is not imposed in a top-down manner. Users can at any time add or delete data, introduce or change their own tags, and thus impose taxonomies on their data.

The distributed creation of taxonomies and tags and the multiplicity of conceptual schemata generate the well known problem of *semantic interoperability*. One solution is to standardise. The different users of a collective information system could all agree a priori to use the same taxonomies to structure their data and to use the same conceptual schemata for their data and meta-data. The tags in the owner taxonomies can then act as a shared communication protocol between peers. For example, all users of web browsers could agree to use the taxonomies of Yahoo for organising their data, and adopt the labels used by Yahoo (possibly with translations into different languages). Unfortunately such a standardisation approach is unlikely to work for truely open-ended collective information systems in rapidly changing domains like music file sharing, picture exchange, medical imaging, scientific papers, etc. New topics and new kinds of

data come up all the time, styles shift, and interests of users diverge, so it is very hard to capture all this once and for all in a static taxonomy. Another issue is that taxonomies and conceptual schemata may be linked to specific proprietary software that others may not want to use.

Alternatively, it is possible that each peer has its own local taxonomy, but that these are translated into a (more) global taxonomy which is used for querying and information exchange and thus acts as an interlingua between peers. The translation to conceptual schemata of each peer could be aided by mediators [19] and achieved through automated schema matching based on finding structural similarities between schemas (see the survey in [9]). A promising recent variant of automated schema matching is based on ostensive interactions, in which agents send each other examples of the instances of schema elements so that the mapping can be made [16]. The difficulty with this approach is that a one-to-one mapping of taxonomies or conceptual schemata is not always possible. In these cases data semantics must be taken into account.

The first approach which is trying to do this is currently being explored by the Semantic Web initiative [4] and by advocates of CYC or Wordnet [7]. The data is associated with descriptions with a formal semantics, defined in terms of ontologies [5]. This approach is clearly highly valuable for closed domains, but there are known limitations when applied to open-ended information systems [1], [12], [11]. The ontologies do not capture the grounded semantics, they only constrain inference. Moreover the semantic web requires standardisation based on universal (or at least domain-wide) ontologies. But it is hard to imagine that a world-wide consensus is reachable and enforcable in every domain of human activity for which information systems are currently in use. Even in restricted domains this is hard because of an increasingly interconnected global world. Human activity and the information systems built for them are open systems. They cannot be defined once and for all but must be adaptable to new needs.

In this paper, we also take a semantics approach but pursue grounded as opposed to formal semantics. We view semantic interoperability as a coordination problem between the world, information systems, and human users, and propose to set up a semiotic dynamics that achieves this coordination. Rather than trying to map owner taxonomies or conceptual schemata directly onto each other, we propose that each information system has an associated agent. The agents self-organise an interlingua with labels whose underlying categories are grounded in the actual data and meta-data. The interlingua is not universal but coordinated among those agents that need to cooperate. The semiotic dynamics is user-driven in the sense that users continuously stimulate the formation of new labels and categories and steer the grounding of the categories by giving examples and counterexamples.

Our proposal has two components. On the one hand we try to orchestrate the same sort of semiotic dynamics that we see happening in natural languages or in social exchange websites like www.flickr.com, namely there is an emergent system of labels whose use is coordinated among the agents without central coordination. Second we try to achieve grounded semantics by programming the

agents so that they can develop operational classifiers grounding these labels. The semantics is emergent in the sense that it is derived by the system itself and it is dynamic because it continuously tracks the recategorisations that users inevitably carry out as they organise and reorganise their data.

Our proposals are strongly related to other approaches for achieving 'emergent semantics', notably [2], which also emphasises user orientation, and [10], which explores grounded semantics. Similar to [3] we focus on orchestrating a user-driven semiotic dynamics in information agents.

The work reported here relies on a decade of research into the origins and evolution of communication systems for robot-robot and robot-human communication [12], [14]. We have applied these ideas to the semantic interoperability problem and performed a case study in the domain of music file sharing. The development of classifiers can be done in many different ways but for this case study we have relied on recent work in the automatic construction of classifiers inspired by methods from genetic programming [18]. In the present paper, we use however a simpler example to explain the proposed mechanisms and study their behavior.

We are well aware of important limitations of the proposals discussed in this paper, and therefore see it as a first exploration rather than the final solution to semantic interoperability (if that ever could be found). More specifically, grounded semantics is only possible in domains where the meaning of a taxonomy *can* be grounded, which is only the case in well-delineated domains. For example, although it would be straightforward to develop a classifier for the tag 'black-and-white' (which would also work for 'bw', 'noir-et-blanc', etc.), it is quite impossible to develop a grounded semantics for 'New York' as the range of images where this tag might be applicable is vast and strongly varied. However it is only by defining and exploring the kinds of systems discussed in this paper that progress on the issues can be made.

The first part of the paper defines some of the terminology that we will use later. Then we describe the behaviors of the information agents, particularly as they pertain to the negotiation of a shared repertoire of labels and classifiers that constitute the meaning of the labels. Next we give example interactions from our implementation in the music domain.

2 Definitions

We assume that the information systems handle sets of *data*. The data can be files (text, music, movie, or image files) or any data stored by other means. For simplification, only one type of data is assumed (for example, only music files). Furthermore, we assume that every datum has a unique identifier that can be communicated between peers. The identifiers can be a URL that indicates where the data (or meta-data) can be found, or it can be an index into a database that

is accessible to everyone[3]. In the exchanges between agents, the identifiers are sent instead of the data. In the worst case, when no identifiers are available, the data itself will be transmitted. We will use $\mathcal{D}$ to indicate the set of all data.

We will use also *tags* and *labels*. Both are character strings. The tags are used by the owners in their taxonomy. The labels are used by the information agents. The set of all labels will be denoted by $\mathcal{L}$, and the set of all tags $\mathcal{N}$. Agents have also access to classifiers which are computable functions over data items. They are used to give grounded semantics to the labels.

2.1 Classifiers

Definition 1. *Classifier*

A classifier is a function, $c : \mathcal{D} \rightarrow 0, 1$, that, given a data element d, returns 1 or 0, depending on whether d belongs to a particular class (category) c or not. The set of classifiers is denoted by C.

For example, there might be a classifier that is able to detect whether a song sample contains a female voice or not, or whether a song was performed by The Beatles.

Classifiers use data and meta-data to decide what data belongs to the category and what not. The construction of classifiers is discussed in section 4.

Agents need the ability to discriminate a data set D_1 from another set D_2. For example, songs which have a female voice and those which do not. For this purpose, we introduce two functions, the *scope* and the *discriminative success* of a classifier.

Definition 2. *Scope of a classifier*

The scope of a classifier c for a set of data D, $\sigma(D, c)$, is the fraction of elements in D for which c returns 1:

$$\sigma(D, c) = \frac{\sum_{d \in D} c(d)}{|D|}$$

Definition 3. *Discriminative success of a classifier*

The discriminative success of a classifier c measures how well c can discriminate between two data sets D_1 and D_2:

$$\delta(D_1, D_2, c) = \sigma(D_1, c) - \sigma(D_2, c)$$

The categorisation process consists of finding the classifier with the highest discriminative success, given two data sets D_1 and D_2.

[3] For music files, the MusicBrainz database could be used (http://www.musicbrainz.org); for movies, the Internet Movie Database (http://www.imdb.com); for scientific papers, Citeseer (http://citeseer.ist.psu.edu).

Definition 4. *Categorise*

Let D_1 and D_2 be two sets of data and C a set of classifiers. We can order the classifiers $c \in C$ in descending order based on their discriminative success: $[\langle c_1, p_1\rangle, ..., \langle c_m, p_m\rangle]$ with $c_i \in C$, and $p_i = \delta(D_1, D_2, c_i)$, and $p_i, p_{i+1} \rightarrow p_i \geq p_{i+1}$. Clearly the better c_i distinguishes the elements in D_1 from the elements in D_2, the greater $\delta(D_1, D_2, c_i)$ will be and hence by taking the first element of the sequence above, we find the most discriminating category: $categorise(C, D_1, D_2) = first([\langle c_1, p_1\rangle, ..., \langle c_m, p_m\rangle])$.

In the case where several classifiers $c_i, \ldots, c_n$ have a maximum discriminative success, additional heuristics could be used in choosing the best classifier. For example, one may choose the classifier that has been used most successfully in previous exchanges with other agents. Classifiers will not only be used to distinguish between two sets but also to filter an existing data set, based on the following definition:

Definition 5. *Filter*

$$filter(D, c) = \{d \mid d \in D \text{ and } c(d) = 1\}$$

2.2 Dictionaries

The association of labels with classifiers and vice versa is stored in a *dictionary*.

Definition 6. *Dictionary*

A dictionary W is a two-way mapping from a set of labels L to a set of classifiers C. Each association between a classifier and a label has a certain strength $\gamma \in [0.0, 1.0]$. More formally: $W \subset \mathcal{L} \times \mathcal{C} \times [0.0, 1.0]$.

For example, there could be a label 'female-voice' which is associated with the classifier able to decide whether a song contains a female voice or not.

Given a classifier c and a dictionary W, we can construct the list of possible labels for a classifier c as an ordered set based on the strength γ of the relation between c and a label l:

$$labels(c, W) = ((l_1, \gamma_1), ..., (l_n, \gamma_n)) \text{ with } (c, l_i, \gamma_i) \in W \text{ and } \gamma_i \geq \gamma_{i+1}$$

Definition 7. *Coding of a classifier*

The label coding a classifier c is the first label from this set: $code(c, W) = first(labels(c, W))$.

The inverse operation, *decode*, is defined similarly. Given a label l and a dictionary W, we construct the list of possible classifiers as an ordered set based on the strength γ of the relation between l and a classifier c:

$$cats(l, W) = ((c_1, \gamma_1), ..., (c_n, \gamma_n)) \text{ with } (c_i, l, \gamma_i) \in W \text{ and } \gamma_i \geq \gamma_{i+1}$$

Definition 8. *Decoding of a label*
The decoded classifier is the first classifier from the ordered set: $decode(l, W) = first(cats(l, W))$.

This process can be easily extended to coding or decoding conjunctions (i.e. sets) of classifiers. Coding should seek the minimal number of words that cover the set of classifiers resulting from discrimination and decoding should reconstruct the minimal set of classifiers that are associated with each of the words.

2.3 Peer information system

Definition 9. *Peer information system*
A peer information system a at time t as $PI_{a,t} = \langle IS_{a,t}, O, IA_{a,t} \rangle$ *is defined as:*

- $IS_{a,t} = \langle D_{a,t}, N_{a,t}, M_{a,t}, MD \rangle$ *being an information system which consists of a set of data D, a set of tags N and a mapping* $M : N \rightarrow \mathcal{P}(D)$ *mapping tags to subsets of D by an extensional definition, and a set of meta-data MD.*
- *O a (human) owner.*
- $IA_{a,t} = \langle L_{a,t}, C_{a,t}, W_{a,t} \rangle$ *an information agent with a set of labels L, a set of classifiers C and a dictionary W.*

Definition 10. *Collective information network A collective information system IN consists of a set of information systems:* $IN = \{PI_1, ...PI_n\}$.

When the owner of one information system queries another information system, we call the information system of the querying peer the caller (or client) and the system providing information the callee (or server).

3 Agent behaviors

3.1 Interaction

We assume that the (human) owner of the caller initiates a query by identifying a set $G_c \subseteq D_c$ of data elements that are considered to be good examples of the kind of elements the owner is requesting from the callee. The owner can do this by using the tags of the owner taxonomy that remains fully under his control, or by explicitly identifying in some other way a subset of the examples in the information system's data set. We assume furthermore that the query is formulated within a specific context $K_c \subseteq D_c$, which consists of other data elements against which G_c is to be distinguished (counter-examples). The counter-examples should not overlap with the examples, i.e. $G_c \cap K_c = \emptyset$.

For example, the owner could choose a number of tags (like 'jazz', 'female-voice', 'piano') yielding a set of possible data elements based on the tagging system. If this is the user's first interaction, these data elements are viewed

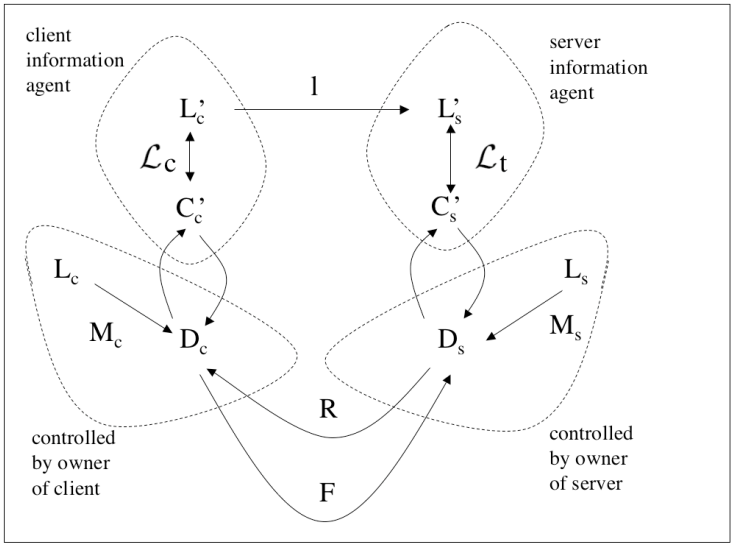

Fig. 2. The different entities and relations involved in collective information exchange and the items that are exchanged between the two. To the left are components of a caller and to the right those of a callee.

against the set of all other music files in his information system. We expect the information agent to come up with the best classifier for distinguishing the data elements against the context and then querying other peers to find the data that are the most compatible. Agents do not exchange the music files themselves, neither the classifiers directly (as different peers may use different libraries or programming languages for their classifiers), but rather labels that are progressively negotiated.

Five types of situations may occur and they are defined in the following subsections. In section 5, we give specific examples for each of these situations in the context of a music application.

3.2 Successful interaction

The following script gives an overview of the interaction in case of a successful query. The left side details the actions on the caller side and the right side the actions on the callee side. In the middle we show the items that are exchanged between caller and callee.

1. Caller owner chooses $G_c \subset D_c$ with K_c as context 2. Caller IA categorises G_c, K_c as cat_c 3. Caller IA codes cat_c as l	$\longrightarrow\ l\ \longrightarrow$	1. Callee IA decodes l as cat_s 2. Callee IA filters $R = filter(Ds, cat_s)$
4. Caller owner selects $F \subseteq R$ as relevant data 5. Caller inserts F in D_c and updates W_c	$\longleftarrow R \longleftarrow$ $\rightarrow success \rightarrow$	3. Callee updates W_s

If the interaction succeeds completely, the feedback F provided by the caller's owner is positive, in other words he received mainly good examples. This implies that the label l naming cat_c, as used by the caller, was compatible with the interpretation of this label by the callee as cat_s and compatible with the desires of the user. In this case, both caller and callee update their mappings from labels to categories so that the use of the label l for the categories cat_c and cat_s is reenforced in the future. This is done by increasing the relation between the used label and the used classifier with a quantity Δ_{inc}, and diminishing competing relations. Competitors are relations that either use another label for the same classifier, in which case they are decreased with Δ_{n-inh}, or that have associated another classifier with the same label, in which they are decreased with Δ_{o-inh}. More formally, $UpdateCaller(W_{c,t}, l, cat_c)$ is defined as:

$$\begin{aligned} W_{c,t+1} = &\{r_i | r_i = (c_i, l_i, \gamma_i) \in W_{c,t} \text{ with } c_i \neq cat_c \text{ and } l_i \neq l\} \quad \cup \\ &\{(cat_c, l, \gamma_i + \Delta_{inc}) \text{ for } w_c = (cat_c, l, \gamma_i) \in W_{c,t}\} \quad \cup \\ &\{r_j | r_j = (cat_c, l_j, \gamma_i + \Delta_{n-inh}) \text{ with } l_j \neq l\} \quad \cup \\ &\{r_j | r_j = (cat_j, l, \gamma_i + \Delta_{o-inh}) \text{ with } cat_j \neq cat_c\} \end{aligned}$$

Similarly, $UpdateCallee(\mathcal{W}_{s,t}, l, cat_s)$ is defined as:

$$\begin{aligned} W_{s,t+1} = &\{r_i | r_i = (c_i, l_i, \gamma_i) \in W_{s,t} \text{ with } c_i \neq cat_s \text{ and } l_i \neq l\} \quad \cup \\ &\{(cat_s, l, \gamma_i + \Delta_{inc}) \text{ for } w_s = (cat_s, l, \gamma_i) \in W_{s,t}\} \quad \cup \\ &\{r_j | r_j = (cat_s, l_j, \gamma_i + \Delta_{n-inh}) \text{ with } l_j \neq l\} \quad \cup \\ &\{r_j | r_j = (cat_j, l, \gamma_i + \Delta_{o-inh}) \text{ with } cat_j \neq cat_s\} \end{aligned}$$

3.3 The information agent fails to categorise

Consider next the situation in which the caller does not have a classifier to discriminate G_c from K_c (caller step 2 fails). In that case, the caller performs two steps:

1. The caller constructs a new classifier cat_n that distinguishes the elements in G_c from those in K_c.
2. The caller invents a new label l_n (a random string drawn from a sufficiently large alphabet) and extends W with a new relation between l_n and cat_n with an initial value for the strength being γ_{init}: $W'_c = W_c \cup \{(l_n, cat_n, \gamma_{init})\}$.

The interaction between caller and callee can now continue as before with the transmission of l_n as new label.

An important issue arises when the random string was already used by another agent for another classifier. This issue (known as homonymy: one label having different meanings) is dealt with by the dynamics of the system as presented here, but it can be minimised by using a technique that guarantees unique symbols even if generated in a distributed fashion, such as the universally unique identifiers (UUID) [6].

3.4 The callee does not know the label

The callee does not have the label l that was transmitted by the caller (callee step 1 fails). In that case, the callee signals failure to the caller. It then receives G_c and K_c as examples of what the caller is looking for and then goes through the following steps:

1. Callee IA categorises G_c as distinctive from the context K_c with the classifier cat_s. When this fails, the callee IA creates a new classifier (further called cat_s) and adds it to its categorial repertoire.
2. Callee IA extends W_s with a relation between cat_s, the label l, and an initial strength γ_{init}: $W'_s = W_s \cup \{(l, cat_s, \gamma_{init})\}$.

Then the callee continues the game as before. The interaction is summarised as follows:

1. Caller owner chooses G_c with K_c as context		
2. Caller IA categorises G_c, K_c as cat_c		
3. Caller IA codes cat_c as l		
	$\longrightarrow\ l\ \longrightarrow$	
		1. Callee IA fails to decode l
	$\leftarrow$ failure $\leftarrow$	
4. Caller IA transmits G_c, K_c		
	$\rightarrow l, G_c, K_c \rightarrow$	
		2b. Callee IA categorises G_c, K_c as cat_s
		2c. Callee IA stores $(cat_s, l, \gamma_{init})$
		3. Callee IA filters $R = filter(Ds, cat_s)$
	$\longleftarrow R \longleftarrow$	
5. Caller owner selects $F \subseteq R$		
6. Caller inserts F in D_c and updates W_c		
	$\rightarrow success \rightarrow$	
		4. Callee updates W_s

3.5 Handling partial success

The next case occurs when the results given by the callee to the caller is deemed to be partly irrelevant by the (owner of the) caller. A score can be computed which is simply the percentage of elements that was deemed appropriate by the owner. A percentage below θ_{fail} signals the failure. There are two causes for this problem: (1) The classifier used by the caller is not precise enough to capture the distinction that was intended by the user, or (2) the classifier associated with the transmitted label by the caller is different from the classifier associated with the same label by the callee.

The distinction between the two cases is done as follows. After the evaluation of the results by the owner, the calling agent is in possession of two sets of good examples (G_c and F) and two sets of counter-examples (K_c and B). With the extra information available, the agent can now try to find a classifier that has a higher discriminative success than the initially chosen classifier. If such a classifier can be found, the agent concludes that it has misinterpreted the intentions of its owner. If such a classifier cannot be found, it signal a communication failure, indicating that the callee has a different interpretation of the label that does not match with its own. The next two sections detail the interactions in both cases.

3.6 The caller has misinterpreted the owner's request

In case the calling agent can find a new classifier cat'_c with a higher discriminative success between the new sets $G_c \cup F$ and $K_c \cup B$, the interaction proceeds as

before using cat'_c instead of cat_c. The classifier cat'_c can either be an existing classifier or it can be a classifier that is newly created. The classifier cat'_c is coded as label l'.

1. Caller owner chooses G_c with K_c as context		
2. Caller IA categorises G_c, K_c as cat_c		
3. Caller IA codes cat_c as l		
	$\longrightarrow\ l\ \longrightarrow$	
		1. Callee IA categorises G_c, K_c as cat_s
		2. Callee IA filters $R = filter(Ds, cat_s)$
	$\longleftarrow R \longleftarrow$	
4. Caller owner selects $F \subseteq R$ as relevant data		
5. Caller inserts F in D_c		
6. The result score is too low		
6a. Caller IA categorises $G_c \cup F, K_c \cup B$ as cat'_c		
6b. Caller IA codes cat'_c as l'		
	$\longrightarrow l' \longrightarrow$	
		2. Callee IA decodes l' as cat'_s
		3. Callee IA filters $R = filter(Ds, cat'_s)$
	$\longleftarrow R' \longleftarrow$	
4. Caller owner selects $F' \subseteq R'$ as relevant data		
5. Caller inserts F' in D_c and updates W_c		
	$\rightarrow success \rightarrow$	
		3. Callee updates W_s

The second query, using label l', can be implemented as a recursive call to the `Query` function, described previously, using the sets $G_c \cup F$ and $K_c \cup B$. This call can fail for the same reasons as the first invocation. For example, the callee may not know the label l' and signal a failure. In that particular case, the interaction falls back to the situation discussed in section 3.4.

3.7 The caller and callee interpret the label differently

In this case caller and callee should try to coordinate their categories and labels so that exchange becomes possible or fruitful in the future. Actions are necessary both on the side of the caller and of the callee. First of all the strength of the labels they used in the failed communication are to be diminished:

1. Caller IA diminishes the association strength between cat_c and l in W_c by a factor Δ_{dec}. This will decrease the chance that the relation is coded in the future with this particular label.
2. Callee IA diminishes the association strength between cat_s and l in W_s by a factor Δ_{dec}. This will decrease the chance that l is decoded in the future with this relation.

If the caller was not able to come up with a better classifier than the one used in the first transaction, the caller IA can send examples of the objects of interest $G_c \cup F$ and the context $K_c \cup B$ so that the callee can attempt to acquire the right meaning by finding a distinctive classifier and by adding an association between this classifier and the label l. This case then becomes identical to the one discussed earlier (section 3.4, "the callee does not know the label").

1. Caller owner chooses G_c with K_c as context		
2. Caller IA categorises G_c, K_c as cat_c		
3. Caller IA codes cat_c as l		
	$\longrightarrow\ l\ \longrightarrow$	
		1. Callee IA categorises G_c, K_c as cat_s
		2. Callee IA filters $R = filter(Ds, cat_s)$
	$\longleftarrow R \longleftarrow$	
4. Caller owner selects $F \subseteq R$ as relevant data		
5. Caller inserts F in D_c		

6. The result score is too low		
6a. Caller IA fails to categorise $G_c \cup F, K_c \cup B$		
6b. fails to create a better classifier		
7. Caller updates W_c and signals failure		
	$\rightarrow failure \rightarrow$	
		3. Callee IA updates W_s
8. Caller IA transmits $l, G_c \cup F, K_c \cup B$		
	$\rightarrow l, G_c \cup F, K_c \cup B \rightarrow$	
		4a. Callee IA categorises $G_c \cup F, K_c \cup B$ as cat'_s
		4b. Callee IA stores $(cat'_s, l, \gamma_{init})$

3.8 Parameters

In summary, we find the following main parameters for the agent's adaptive mechanisms. Each time we give values for these parameters that have proven to yield adequate performance in large-scale tests of the system.

1. γ_{init} is the initial strength with which a new relation enters into the dictionary $\mathcal{L}$ of the agents. $\gamma_{init} = 0.5$.
2. Δ_{inc} is the increase of γ in the relation used, in case there is success. $\Delta_{inc} = 0.1$. [ENFORCEMENT]
3. Δ_{n-inh} is the decrease of relations with the same label (but different categories) in case of success. $\Delta_{n-inh} = -0.2$.
   Δ_{o-inh} is the decrease of relations with the same classifier (but different labels) in case of success. $\Delta_{o-inh} = -0.1$. [LATERAL INHIBITION]
4. Δ_{dec} is the decrease of γ in the relation used, in case when there is failure. $\Delta_{dec} = -0.1$. [DAMPING]
5. θ_{disc} is the threshold used in the categorisation. $\theta_{disc} = 0.5$
6. θ_{fail} is the threshold used to signal a failed exchange. $\theta_{fail} = 0.5$

There is some leeway with the exact value of these parameters. It is even possible to make all of them 0 (accept γ_{init}) but then all labels ever invented by any agent will propagate in the population and so we get a very large dictionary. If they are non-zero, then obviously $\Delta_{inc} > 0$ and $\Delta_{n-inh} < 0$, $\Delta_{o-inh} < 0$. Also $\Delta_{dec} < 0$ because otherwise a relation that is not successful would increase in strength. The importance of the parameters is summarised in figure 3, taken from simulation experiments. Adoption means that new labels propagate, enforcement means that the strength is increased in case of success, damping means that the strength is decreased in case of failure, and lateral inhibition means that the strength of competitors is decreased in case of success.

4 Categorisation

We recall that a classifier γ is a computable function that, when given a data element $d \in D$, returns $true$ (T) or $false$ (F): $\gamma : \mathbf{D} \mapsto \{T, F\}$. A classifier $C \in \mathbf{C}$ is defined by a classifier γ_c that returns $true$, if d belongs to the classifier, or $false$ otherwise. A classifier C is a set whose elements are specified by intensional definition, i.e. by the classifier γ_c. The elements of the classifier C are rarely known in advance, not by the user nor by the information system. Instead, the user provides a subset of C, the examples $K \subset C$, and a subset of C's complement, the counter-examples $K \subset (\mathbf{D} \setminus C)$. Thus, the user provides the information agent with an incomplete, extensional definition of C and $\mathbf{D} \setminus C$. The categorisation process then consist of generalizing this partial definition into an intensional definition in the form of a classifier γ. In the worst case, the classifier γ stores an enumeration of the elements that belong to the classifier. We define the two classifiers γ_{min} and γ_{max} as follows:

$$\gamma_{min} : \mathbf{D} \mapsto \{T, F\} : \gamma(d) = T \text{ if } d \in G,\ F \text{ otherwise}$$

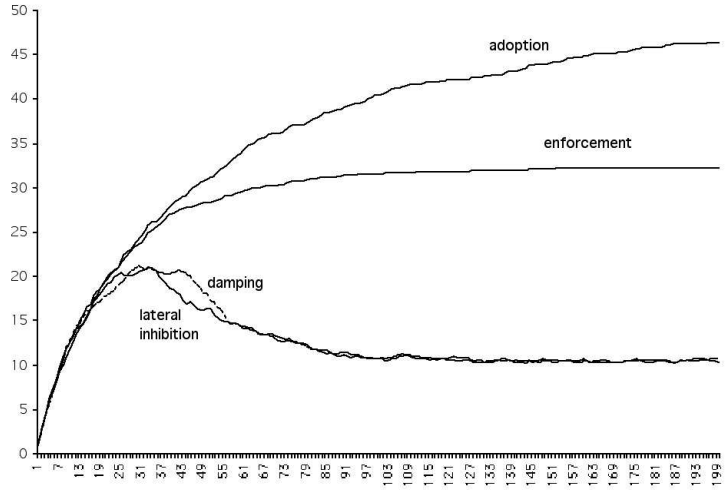

Fig. 3. The evolution in average dictionary size for labeling 10 objects in a population of 10 agents. Enforcement combined with lateral inhibition and damping leads to the most efficient dictionary, in which only 10 labels are used for 10 objects.

$$\gamma_{max} : \mathbf{D} \mapsto \{T, F\} : \gamma(d) = F \text{ if } d \in K,\ T \text{ otherwise}$$

The classifiers γ_{min} and γ_{max} are the simplest form of a classifier. If $G = C$ and $K = \mathbf{D} \setminus C$ then $\gamma_{min} = \gamma_{max}$. For any relevant classifier built by the information system it should hold that $D_{\gamma_{min}} \subseteq D_\gamma \subseteq D_{\gamma_{max}}$. A classifier γ converges if $D_\gamma \to C$, for $G \to C$ and $K \to \mathbf{D} \setminus C$. In other words, if we know all the elements in C and all the elements not in C, than the classifier should map exactly to C. If the user only provides additional good examples such that $G \to C$ but no counter-examples are added to K, the classifier may not converge: $D_\gamma \to D'_\gamma$, with $C \subseteq D'_\gamma \subseteq D_{\gamma_{max}}$. It will generally be necessary that the set of counter-examples contains enough elements in order to construct an efficient classifier.

There are in principle many ways to build classifiers. The approach that we have used for the music domain explored discussed further uses genetic programming techniques as described in more detail in [18].

5 An example: Music sharing

We now show a concrete example of these scripts at work for the case of p2p music sharing. The examples have been drawn from our experimental implementation. We introduce three users and their respective information agents. Each peer has local meta-data, displayed in the tables below. All songs have a unique identification number, common to all agents. To make the examples more readable we will use the title to indicate a song instead of its identification number.

The agents also have access to the name of the artists. However, these names are formatted differently for every agent. The other attributes in the tables are specific to every agent. $Agent_0$'s database contains a genre and a BPM (beats per minute) column. $Agent_1$ stores the release year of the recordings and $Agent_2$ has information on the *global energy* of the songs. The peers thus have different sets of data and meta-data, and the conceptual schemas used for storing data is different.

The meta-data of peer 0 is as follows:

ID	Name	Artist	Genre	BPM
15	Let's Spend the Night Together	The Rolling Stones	Rock	141
14	Ruby Tuesday	The Rolling Stones	Rock	102
13	Paint It Black	The Rolling Stones	Rock	160
12	And I Love Her	The Beatles	Rock 'n Roll	115
11	Another Girl	The Beatles	Rock 'n Roll	180
10	Twist And Shout	The Beatles	Rock 'n Roll	128
9	I'm Down	The Beatles	Rock 'n Roll	164
8	A Hard Day's Night	The Beatles	Rock 'n Roll	141
7	Norwegian Wood	The Beatles	Pop	61
27	Amazing Grace	Elvis Presley	Rock 'n Roll	63
6	Eleanor Rigby	The Beatles	Pop	138
26	Are You Lonesome Tonight	Elvis Presley	Rock 'n Roll	75
5	I Feel Fine	The Beatles	Rock 'n Roll	180
25	Love Me Tender	Elvis Presley	Rock 'n Roll	81
4	You Know My Name	The Beatles	Pop	96
24	Smoke on the Water	Deep Purple	Rock	120
3	Across The Universe	The Beatles	Pop	82
2	Helter Skelter	The Beatles	Rock	155
1	Blackbird	The Beatles	Pop	94
21	Billie Jean	Michael Jackson	Pop	119
0	Sie Liebt Dich	The Beatles	Rock 'n Roll	153
20	True Blue	Madonna	Pop	119

The meta-data of Peer 1 is:

ID	Name	Nom	Année
15	Let's Spend the Night Together	Rolling Stones, The	1967
14	Ruby Tuesday	Rolling Stones, The	1967
13	Paint It Black	Rolling Stones, The	1966
12	And I Love Her	Beatles, The	1964
11	Another Girl	Beatles, The	1965
10	Twist And Shout	Beatles, The	1963
9	I'm Down	Beatles, The	1965
8	A Hard Day's Night	Beatles, The	1964
7	Norwegian Wood	Beatles, The	1965
27	Amazing Grace	Presley, Elvis	1972
6	Eleanor Rigby	Beatles, The	1966
26	Are You Lonesome Tonight	Presley, Elvis	1956
5	I Feel Fine	Beatles, The	1964
25	Love Me Tender	Presley, Elvis	1957
4	You Know My Name	Beatles, The	1969
24	Smoke on the Water	Deep Purple	1972
3	Across The Universe	Beatles, The	1970
2	Helter Skelter	Beatles, The	1968
1	Blackbird	Beatles, The	1968
21	Billie Jean	Jackson, Michael	1983
0	Sie Liebt Dich	Beatles, The	1964
20	True Blue	Madonna	1986

The meta-data of Peer 2 is:

ID	Name	Band	Energy
15	Let's Spend the Night Together	stones	0.612
14	Ruby Tuesday	stones	0.322
13	Paint It Black	stones	0.571
12	And I Love Her	beatles	0.431
11	Another Girl	beatles	0.607
10	Twist And Shout	beatles	0.745
9	I'm Down	beatles	0.623
8	A Hard Day's Night	beatles	0.619
7	Norwegian Wood	beatles	0.494
27	Amazing Grace	the_king	0.38
6	Eleanor Rigby	beatles	0.51
26	Are You Lonesome Tonight	the_king	0.0
5	I Feel Fine	beatles	0.523
25	Love Me Tender	the_king	0.128
4	You Know My Name	beatles	0.502
24	Smoke on the Water	deep_purple	0.53
3	Across The Universe	beatles	0.55
2	Helter Skelter	beatles	0.642
1	Blackbird	beatles	0.191
21	Billie Jean	michael_jackson	0.533
0	Sie Liebt Dich	beatles	0.663
20	True Blue	madonna	0.619

As explained earlier, we assume that owners have imposed a taxonomy on their data shown in the following tables for peer 0, 1, and 2 respectively:

folder	files
beatles	Across The Universe, I Feel Fine, Norwegian Wood, Helter Skelter, You Know My Name, Eleanor Rigby, Blackbird
stones	Let's Spend the Night Together, Ruby Tuesday

folder	files
sixties	Twist And Shout, Paint It Black, I'm Down, And I Love Her
seventies	Smoke on the Water, Let's Spend the Night Together
eighties	Billie Jean, True Blue

folder	files
party music	Twist And Shout, I'm Down

We now describe four queries which illustrate the approach. The agents start from an initial state where they do not know any labels nor categories yet.

5.1 Query 1: Agent 0 asks Agent 1 for more "beatles"

In the first query, $user_0$ selects the folder "beatles" and asks its agent to seek similar songs. Since the user explicitly selects "beatles" and not "rolling stones", the agent interprets the request as: "find more beatles, not rolling stones". The search query then proceeds as follows:

- $Agent_0$ categorises the owner's request using the following set of examples: [Across The Universe], [I Feel Fine], [Norwegian Wood], [Helter Skelter], [You Know My Name], [Eleanor Rigby], [Blackbird], and counter-examples: [Let's Spend the Night Together], [Ruby Tuesday].
- Since the dictionary is empty, $agent_0$ fails to find a classifier that discriminates the examples from the counter-examples.
- $Agent_0$ constructs the new classifier, `Artist(The Beatles)`. It introduces a new label[4] `"6365915a"` and binds the name to the classifier with the default strength of 0.5.
- $Agent_0$ queries $agent_1$ with the label `"6365915a"`.
- $Agent_1$ fails to decode the label `"6365915a"` since its dictionary is empty. It returns a failure message to $agent_0$.
- $Agent_0$ transmits the examples and counter-examples of `"6365915a"`.
- $Agent_1$ creates the classifier `Nom(Beatles, The)` and binds the label `"6365915a"` to it with a default strength of 0.5.
- $Agent_0$ iterates the query again. This time, $agent_1$ successfully decodes `"6365915a"` as `Nom(Beatles, The)` and uses the classifier to filter its data collection. The results are: [And I Love Her], [Twist And Shout], [I'm Down].
- $Agent_0$ asks its owner to evaluate the results. All songs are considered good results. The query was a success.
- $Agent_0$ updates its dictionary by increasing the strength of the binding between `Artist(The Beatles)` and `"6365915a"`.
- $Agent_1$ updates its dictionary by increasing the strength of the binding between `Nom(Beatles, The)` and `"6365915a"`.

The following listing shows the same query in a more compact form. We will use this form of presentation in the remainder of the text.

```
- [0, Agent0]: search examples: [Across The Universe][I Feel
  Fine][Norwegian Wood][Helter Skelter][You Know My Name][Eleanor
  Rigby][Blackbird], counter-examples: [Let's Spend the Night
  Together][Ruby Tuesday]
- [1, Agent0]: categorisation failed
- [2, Agent0]: creates Category<Artist(The Beatles)>
- [3, Agent0]: binds 6365915a to Category<Artist(The Beatles)>
- [4, Agent1]: query for 6365915a
- [5, Agent1]: fails to decode 6365915a
- [6, Agent0]: transmits examples and count-examples of 6365915a
```

[4] We use randomly generated labels based on the UUID algorithm.

```
- [7, Agent1]: categorisation failed
- [8, Agent1]: creates Category<Nom(Beatles, The)>
- [9, Agent1]: binds 6365915a to Category<Nom(Beatles, The)>
- [10, Agent1]: query for 6365915a
- [11, Agent1]: decodes 6365915a as Category<Nom(Beatles, The)>
- [12, Agent1]: filter data: results: [I'm Down][And I Love Her][Twist
  And Shout]
- [13, Agent0]: owner evaluation: good (3 out of 3): [Twist And
  Shout][I'm Down][And I Love Her]
- [14, Agent0]: search sucessful
- [15, Agent0]: update dictionary
- [16, Agent1]: update dictionary
```

The dictionary of peer 0 is now:

Label	Category	Strength
6365915a	Artist(The Beatles)	0.6

The dictionary of peer 1 is:

Label	Category	Strength
6365915a	Nom(Beatles, The)	0.6

5.2 Query 2: Agent 1 asks Agent 0 for more "sixties"

In the second query, $user_1$ select the folder "sixties" and requests for more songs like these. $Agent_1$ interprets the request as "find more sixties, not seventies nor eighties". It chooses to send the query to $agent_0$. The complete listing of interaction is given below. In step 1, $agent_1$ categorises the set of examples and counter-examples as `Nom(Beatles, The)`. This is a reasonable choice because three out of four of the examples are indeed Beatles songs. However, when $owner_1$ evaluates the results in step 6, the number of songs that are considered good is low. In this case, $agent_1$ concludes that it has misinterpreted the request of the owner. It tries to correct its mistake and creates a new classifier, `Année(from 1963 to 1966)`, that better reflects the request. A new label is introduced and bound to the classifier. The search is then repeated. $Agent_0$ does not know the new label, so examples and counter-examples are transmitted to indicate its meaning. $Agent_0$ categorises these examples as `Genre(Rock 'n Roll)` and filters its data set with it. The results are all considered valid by $owner_1$. The query thus ends successfully even though both agents use different categories and meta-data.

```
- [0, Agent1]: search examples: [Twist And Shout][Paint It Black][I'm
  Down][And I Love Her], counter-examples: [Billie Jean][Smoke on the
  Water][True Blue][Let's Spend the Night Together]
- [1, Agent1]: uses Category<Nom(Beatles, The)>
- [2, Agent1]: codes Category<Nom(Beatles, The)> as 6365915a
```

```
- [3, Agent0]: query for 6365915a
- [4, Agent0]: decodes 6365915a as Category<Artist(The Beatles)>
- [5, Agent0]: filter data: results: [Blackbird][Eleanor
  Rigby][Norwegian Wood][Across The Universe][You Know My Name][And I
  Love Her][Helter Skelter][I Feel Fine][Twist And Shout][I'm Down]
- [6, Agent1]: owner evaluation: good (4 out of 10): [Twist And
  Shout][I Feel Fine][I'm Down][And I Love Her], bad (6 out of 10):
  [Across The Universe][Norwegian Wood][Helter Skelter][You Know My
  Name][Eleanor Rigby][Blackbird]
- [7, Agent1]: owner request misinterpreted, uses new
  Category<Annee(from 1963.0 to 1966.0)> instead.
- [8, Agent1]: binds 5f6a1a0c to Category<Annee(from 1963.0 to 1966.0)>
- [9, Agent1]: transmits examples and count-examples of 5f6a1a0c
- [10, Agent0]: categorisation failed
- [11, Agent0]: creates Category<Genre(Rock 'n Roll)>
- [12, Agent0]: binds 5f6a1a0c to Category<Genre(Rock 'n Roll)>
- [13, Agent0]: query for 5f6a1a0c
- [14, Agent0]: decodes 5f6a1a0c as Category<Genre(Rock 'n Roll)>
- [15, Agent0]: filter data: results: [And I Love Her][I Feel
  Fine][I'm Down][Twist And Shout]
- [16, Agent1]: owner evaluation: good (4 out of 4): [Twist And
  Shout][I Feel Fine][I'm Down][And I Love Her]
- [17, Agent1]: search sucessful
- [18, Agent1]: update dictionary
- [19, Agent0]: update dictionary
```

The dictionary of agent 0 is now:

Label	Category	Strength
6365915a	Artist(The Beatles)	0.6
5f6a1a0c	Genre(Rock 'n Roll)	0.6

The dictionary of agent 1 is now:

Label	Category	Strength
6365915a	Nom(Beatles, The)	0.6
5f6a1a0c	Année(from 1963.0 to 1966.0)	0.6

5.3 Query 3: Agent 2 asks Agent 0 for more "party music"

In the third query, $owner_2$ is looking for more "party music". The request will be directed to $agent_0$. In this query, the context will be empty because there are no other folders from which to distinguish the selected "party music". In step 2 of the query, $agent_2$ creates the classifier `Band(beatles)` to describe the two example songs. $Agent_0$ learns the new label from $agent_2$ (step 5–8) and returns a list of Beatles songs (steps 9–11). As in the previous query, the information

agent concludes it has misinterpreted the selection of the owner. Using the sets of good and bad examples, it tries to create a better classifier than the one used. It introduces the classifier `Energy(from 0.523 to 0.745)`, binds a new label to it, and calls upon $agent_0$ again (steps 12–14). The new label is explained by transmitting the examples and counter-examples. $Agent_0$ finds that the existing classifier `Genre(Rock 'n Roll)` fits the description well and binds the new label to it. The corresponding songs in its data set are sent back to $agent_2$ (steps 15–20). Three out of four results are deemed relevant to $owner_2$ and the query is considered a success.

```
- [0, Agent2]: search examples: [Twist And Shout][I'm Down],
  counter-examples: none
- [1, Agent2]: categorisation failed
- [2, Agent2]: creates Category<Band(beatles)>
- [3, Agent2]: binds 8b85235d to
  Category<Band(beatles)>
- [4, Agent0]: query for 8b85235d
- [5, Agent0]: fails to decode 8b85235d
- [6, Agent2]: transmits examples and count-examples of 8b85235d
- [7, Agent0]: uses Category<Artist(The Beatles)>
- [8, Agent0]: binds 8b85235d to Category<Artist(The Beatles)>
- [9, Agent0]: query for 8b85235d
- [10, Agent0]: decodes 8b85235d as Category<Artist(The Beatles)>
- [11, Agent0]: filter data: results: [Twist And Shout][And I Love
  Her][Norwegian Wood][Helter Skelter][I'm Down][Blackbird][You Know
  My Name][Across The Universe][I Feel Fine][Eleanor Rigby]
- [12, Agent2]: owner evaluation: good (4 out of 10): [Twist And
  Shout][I Feel Fine][Helter Skelter][I'm Down], bad (6 out of 10):
  [Across The Universe][Norwegian Wood][You Know My Name][Eleanor
  Rigby][Blackbird][And I Love Her]
- [13, Agent2]: owner request misinterpreted, uses new
  Category<Energy(from 0.523 to 0.745)> instead.
- [14, Agent2]: binds f5af0ee6 to Category<Energy(from 0.523 to 0.745)>
- [15, Agent2]: transmits examples and count-examples of f5af0ee6
- [16, Agent0]: uses Category<Genre(Rock 'n Roll)>
- [17, Agent0]: binds f5af0ee6 to Category<Genre(Rock 'n Roll)>
- [18, Agent0]: query for f5af0ee6
- [19, Agent0]: decodes f5af0ee6 as Category<Genre(Rock 'n Roll)>
- [20, Agent0]: filter data: results: [And I Love Her][I'm Down][Twist
  And Shout][I Feel Fine]
- [21, Agent2]: owner evaluation: good (3 out of 4): [Twist And
  Shout][I Feel Fine][I'm Down], bad (1 out of 4): [And I Love Her]
- [22, Agent2]: search sucessful
- [23, Agent2]: update dictionary
- [24, Agent0]: update dictionary
```

The dictionary of agent 0:

Label	Category	Strength
6365915a	Artist(The Beatles)	0.6
8b85235d	Artist(The Beatles)	0.5
5f6a1a0c	Genre(Rock 'n Roll)	0.6
f5af0ee6	Genre(Rock 'n Roll)	0.5

The dictionary of agent 2 after query 3:

Label	Category	Strength
8b85235d	Band(beatles)	0.5
f5af0ee6	Energy(from 0.523 to 0.745)	0.6

5.4 Owner 0 and owner 2 make changes to their taxonomies

After the third query, $owner_0$ and $owner_2$ edit their data sets. The owner of the information system can intervene at any moment in the organisation of the music files, as it is under his control. In this example, $owner_0$ creates a new folder, named "elvis", and $owner_2$ adds more party music to its collection. The contents of the folders after the changes are shown below.

The taxonomy of $owner_0$ after the changes:

folder	files
beatles	Twist And Shout, Across The Universe, I Feel Fine, Norwegian Wood, Helter Skelter, I'm Down, You Know My Name, And I Love Her, Eleanor Rigby, Blackbird
stones	Let's Spend the Night Together, Ruby Tuesday
elvis	Are You Lonesome Tonight, Amazing Grace, Love Me Tender

The taxonomy of $owner_2$ after the changes:

folder	files
party music	Billie Jean, Twist And Shout, I Feel Fine, Helter Skelter, True Blue, I'm Down

5.5 Query 4: Agent 2 asks Agent 0 for more "party music"

The last query start similarly as the previous one. This time again, $agent_2$ is seeking more "party music" from $agent_0$. The initial request of the owner is suitably categorised as `Energy(from 0.523 to 0.745)` and the same label as in query 3 is reused (steps 1–3). $Agent_0$ decodes the label as `Genre(Rock 'n Roll)` and uses this classifier to filter its data set (steps 4–5). This time, however, a large number of the result are considered unrelevant to the owner. The Elvis' songs

that were added since the previous query are not retained by $owner_2$ and the query is deemed unsuccessful. $Agent_2$ concludes that the failing communication is due to a misinterpretation of the label by $agent_0$. The label is therefore considered unreliable and the strength of its binding to the selected classifier is decreased by both agents (steps 6–9). $Agent_2$ then explains the use of the label by pointing $agent_0$ to the set of examples and counter-examples. $Agent_0$ concludes that the classifier `Genre(Rock 'n Roll)` unsufficiently discriminates between both sets and, as a result, introduced a new classifier `BPM(from 119.0 to 180.0)` (steps 10–13). The query is re-launched and yields many good results.

```
- [0, Agent2]: search examples: [Billie Jean][Twist And Shout][I Feel
  Fine][Helter Skelter][True Blue][I'm Down], counter-examples: none
- [1, Agent2]: uses Category<Energy(from 0.523 to 0.745)>
- [2, Agent2]: codes Category<Energy(from 0.523 to 0.745)> as f5af0ee6
- [3, Agent0]: query for f5af0ee6
- [4, Agent0]: decodes f5af0ee6 as Category<Genre(Rock 'n Roll)>
- [5, Agent0]: filter data: results: [And I Love Her][I Feel
  Fine][Amazing Grace][Twist And Shout][I'm Down][Are You Lonesome
  Tonight][Love Me Tender]
- [6, Agent2]: owner evaluation: good (3 out of 7): [Twist And
  Shout][I Feel Fine][I'm Down], bad (4 out of 7): [Are You Lonesome
  Tonight][Amazing Grace][Love Me Tender][And I Love Her]
- [7, Agent2]: search failed
- [8, Agent2]: decreasing binding strength
  [f5af0ee6,Category<Energy(from 0.523 to 0.745)>,0.5]
- [9, Agent0]: decreasing binding strength
  [f5af0ee6,Category<Genre(Rock 'n Roll)>,0.4]
- [10, Agent2]: transmits examples and count-examples of f5af0ee6
- [11, Agent0]: categorisation failed
- [12, Agent0]: creates Category<BPM(from 119.0 to 180.0)>
- [13, Agent0]: binds f5af0ee6 to Category<BPM(from 119.0 to 180.0)>
- [14, Agent0]: query for f5af0ee6
- [15, Agent0]: decodes f5af0ee6 as Category<BPM(from 119.0 to 180.0)>
- [16, Agent0]: filter data: results: [I'm Down][I Feel Fine][Let's
  Spend the Night Together][Eleanor Rigby][Twist And Shout][Helter
  Skelter]
- [17, Agent2]: owner evaluation: good (5 out of 6): [Twist And
  Shout][I Feel Fine][Helter Skelter][Let's Spend the Night
  Together][I'm Down], bad (1 out of 6): [Eleanor Rigby]
- [18, Agent2]: search sucessful
- [19, Agent2]: update dictionary
- [20, Agent0]: update dictionary
```

The dictionary of $agent_0$ after query 4:

Label	Category	Strength
6365915a	Artist(The Beatles)	0.6
8b85235d	Artist(The Beatles)	0.5
5f6a1a0c	Genre(Rock 'n Roll)	0.6
f5af0ee6	Genre(Rock 'n Roll)	0.4
f5af0ee6	BPM(from 119.0 to 180.0)	0.5

The dictionary of $agent_1$ after query 4:

Label	Category	Strength
6365915a	Nom(Beatles, The)	0.6
5f6a1a0c	Année(from 1963.0 to 1966.0)	0.6

The dictionary of $agent_2$ after query 4:

Label	Category	Strength
8b85235d	Band(beatles)	0.5
f5af0ee6	Energy(from 0.523 to 0.745)	0.6

6 Conclusions

This paper considered the question of semantic interoperability in collective information exchange. We advocated the creation of a semiotic dynamics whereby information agents coordinate the use of labels, similar to the way this is now done by human users in social exchange websites, and they develop an emergent grounded semantics for these labels in terms of classifiers that are functions over data or meta-data. We illustrated this for the domain of electronic music distribution.

On the positive side, the examples in section 5 show how the agents "bootstrap" their dictionaries. The only data exchanged between the agents are the unique identifiers of the data and the labels of the query. No meta-data is exchanged nor any indication of the owner's data organisation. This makes it possible for the taxonomies and the meta-data to be completely local to each information system. We have seen cases of successful communication but also how failure is handled in two situations: when the calling agent misunderstands the request of its owner, and when the called agent misinterprets the label of the query. Agents are not using the tags of the folders in the interpretation of the owner's request. So there is no attempt to do taxonomy or schema matching. In fact, the owner's taxonomy plays only a marginal role, it was mainly used to define the initial set and the context for the query. The definitions of classifiers does not depend on them.

The idea of bootstrapping semantic interoperability from local interactions in a bottom-up fashion is not new (see in partiular [2]). The novelty of the presented work resides in the fact that the peers do not exchange the organisation of their meta-data (or database schema's) and that no direct mapping is built between these schema's. Instead, the peers locally maintain a bi-directional mapping between classifiers and tags. In addition, the classifiers nor the mapping are

established by human experts but are introduced by the agents through the exchange of examples and counter-examples. The system is continuously adapting based on the validity of the results. This validation is not done automatically, as has been proposed in the literature, but by the user. The input from the end user, not necessarily an expert, remains a key element of the system.

Although we believe that the approach advocated in this paper provides an interesting alternative to information exchange without semantics or the semantic web, we want to stress the limits of the approach. It will not always be possible to have a grounded semantics, partly because user behavior may be too erratic and subjective to construct classifiers, and partly because the building blocks available for grounding (such as the signal processing primitives in the case of music) or the machine learning methods (in this case genetic programming) may not be effective enough to achieve an adequate grounded semantics. We therefore see the grounding and negotiation of labels for classifiers as one of the building blocks to achieve emergent semantics. Other building blocks consist of exploiting the co-occurrence of tags (as displayed by tag clouds), which establish associative relations that narrow down the set of data elements corresponding to a tag, or the query path of a user that establishes additional context [10].

References

1. Aberer, K.,et.al. (2003) Emergent Semantics. Principles and Issues. To appear in Proc. of the International Conference on Semantics of a Networked World. www.ipsi.fraunhofer.de/ risse/pub/P2004-01.pdf
2. Aberer, K.,et.al. (2003) The Chatty Web: Emergent Semantics through Gossiping. In: Proceedings of the 12th World Wide Web Conference. citeseer.ist.psu.edu/aberer03chatty.html
3. Agostini, A. and P. Avesani (2003) A Peer-to-Peer Advertising game.. July 2003, 15 pages. In: Proceedings of the First International Conference on Service Oriented Computing (ICSOC-03), Springer-Verlag LNCS 2910, pp. 28-42
4. Berners-Lee, T., J. Hendler, and O. Lassila. (2001) The Semantic Web. Scientific American. May 2001.
5. Davies, J., Fensel, D., & Harmelen, F. van. (2003). Towards the semantic web: Ontology driven knowledge management. Chicester, UK: John Wiley & Sons
6. Leach, P., Mealling, M., and R. Salz (2004) A UUID URN Namespace. The Internet Engineering Task Force. Internet drafts. http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt
7. Lenat, D., George A. Miller and T. Yokoi. "CYC, WordNet and EDR — critiques and responses — discussion." In:Communications of the ACM 38 (11), November 1995, pp. 45-48. http://www.acm.org/pubs/articles/journals/cacm/1995-38-11/p24-lenat/p45-lenat.pdf
8. Nejdl, W. et.al. (2003) RDF-based Peer-to-Peer-Networks for Distributed (Learning) Repositories. VLDB journal www.kbs.uni-hannover.de/Arbeiten/Publikationen/2002/
9. Rahm, E., and Philip A. Bernstein (2001) A Survey of Approaches to Automatic Schema Matching VLDB Journal: Very Large Data Bases. 10: 334-350 http://citeseer.ist.psu.edu/rahm01survey.html

10. Santini, S., A. Gupta and R. Jain (2001) Emergent Semantics Through Interaction in Image Databases IEEE Transaction of Knowledge and Data Engineering, summer 2001. www.sdsc.edu/ gupta/publications/kde-sp-01.pdf
11. Staab, S. (2002) Emergent Semantics. IEEE Intelligent Systems. pp. 78-86. www.cwi.nl/ media/publications/nack-ieee-intsys-2002.pdf
12. L. Steels, "The Origins of Ontologies and Communication Conventions in Multi-Agent Systems," Autonomous Agents and Multi-Agent Systems, vol. 1, no. 1, Oct. 1998, pp. 169-194. http://www3.isrl.uiuc.edu/ junwang4/langev/localcopy/pdf/steels98theOrigins.pdf
13. Steels, L. (2002) Emergent Semantics. IEEE Intelligent Systems. Trends and Controversies. p. 83-85. www.cwi.nl/ media/publications/nack-ieee-intsys-2002.pdf
14. Steels, L. (2003) Evolving grounded communication for robots. Trends in Cognitive Science. Volume 7, Issue 7, July 2003 , pp. 308-312. www.csl.sony.fr/downloads/papers/2003/steels-03c.pdf
15. Steels, L. and Kaplan, F. Collective learning and semiotic dynamics. In Floreano, D. and Nicoud, J-D and Mondada, F., editor, Advances in Artificial Life (ECAL 99), Lecture Notes in Artificial Intelligence 1674, pages 679-688, Berlin, 1999. Springer-Verlag.
16. Tzitzikas, Y. and Meghini, C. (2003) Ostensive Automatic Schema Mapping for Taxonomy-based Peer-to-Peer Systems. Proc. of CIA-2003, the Seventh International Workshop on Cooperative Information Agents - Intelligent Agents for the Internet and Web. Lecture Notes in Artificial Intelligence n. 2782, pages 78-92. August 2003 http://www.csi.forth.gr/ tzitzik/publications/Tzitzikas_CIA_2003.pdf
17. Zhang, H., B. Croft, B. Levine, V. Lesse (2004) A Multi-agent Approach for Peer-to-Peer based Information Retrieval System In Proceedings of the 2004 Multi-Agent Conference, AAMAS. New York. http://www.aamas2004.org/proceedings/057_zhangh-p2pir.pdf
18. Zils, A. and Pachet, F. (2004) Automatic Extraction of Music Descriptors from Acoustic Signals using EDS. In Proceedings of the 116th AES Convention, May 2004. http://www/downloads/papers/uploads/zils-04a.pdf
19. Wiederhold, G. (1992) Mediators in the Architecture of Future Information Systems In: IEEE Computer, March 1992, pages 38-49. http://www-db.stanford.edu/pub/gio/1991/afis.ps