Abstract. We present in this paper some attempts to design a Machine Learning method to predict preference knowledge in a multi-agents context. This approach is applied to a corporate knowledge management system.

1 Introduction

In this paper we will present some attempts to design a Machine Learning method to predict preference knowledge in a multi-agents context. Here we define preference knowledge as knowledge about a preference between elements of a set.

For instance, the documents found by a search engine on the web are ordered according to a preference function computed from the user request. Thus, they can be considered as ordered according to a preference relation.

The framework that gave birth to this work is a joint research project, CoMMA\(^1\), dedicated to corporate memory management in an intranet. The main objective of the project is to implement and test a Corporate Memory management framework integrating several emerging technologies in order to optimize its maintenance and ease the search inside it and the use of its content by the members of the organization.

The main challenge is to create a coherent system that relies upon several promising new technologies which are in the middle of their struggle to become standards:

\begin{itemize}
  \item Multi-agent architecture: it is well suited to the heterogeneity of the Corporate Memory; its flexibility eases the system maintenance and keeps the rhythm with the dynamics and evolution of the Corporate Memory; cooperating and adaptive agents assure a better working together with the user in his pursuit to more effectively achieve his goals. The FIPA standard, supported by the CoMMA project, offers the specifications for interoperable intelligent multi-agent systems.
\end{itemize}

\(^1\) This work was supported by the CoMMA (Corporate Memory Management through Agents) project [Con00] funded by the European Commission under Grant IST-1999-12217, which started beginning of February 2000.
– XML: is a standard recommended by the World Wide Web Consortium intended to offer a human and machine understandable description language: a good choice if it is important to ensure an easy maintenance, and seamless flow through various information processing systems that evolve in time.

– RDF/RDFS another W3C recommendation, that creates a semantic level on the top of XML formal description. RDF annotations allow having an integrated, global view of the Corporate Memory keeping untouched (in terms of storage, maintenance) the heterogeneous and distributed nature of the actual info sources. RDF also allows us to create a common ontology to represent the enterprise model. The ontological commitment, a fundamental choice in our approach to design the Corporate Memory, is motivated by our belief that the community of corporate stakeholders is sharing some common global views of the world that needs to be unified and formalized (RDFS) to form the basis of the entire information system.

– Machine Learning Techniques makes the system adaptive to the user, and comes even more naturally due to the previous choices, as presented in the following section.

2 The role of preference in Knowledge Management

The purpose of this section is to discuss how our work (preference learning) fits into the Knowledge Management system.

2.1 Getting the user profile

One of the advantages of an enterprise that should be exploited by such a corporate information management system is that the users (i.e. the employees) can be known (their domains of interest/competence, their current activities/tasks). This can be especially useful in some cases where users are likely to be overwhelmed by the quantity of information to process and navigate themselves through (new employees during accommodation, technology monitoring scientists) who would appreciate personalized, automated help in their process of information retrieval.

2.2 Using semantic annotations

On the other hand, we have “human and machine understandable” semantic information upon the corporate knowledge offered by the RDF formalization, based upon an “enterprise ontology” (RDF schema).

The combination of these two sources of information can provide a rich ground to infer knowledge about the users probable/possible preferences. This combination is made possible due to the fact that we use the same RDF standard for formalizing the user profile; the same base ontology for the enterprise and user models.
It can be imagined that the information combined from these sources will form sets of attributes that will be used as input for an Machine Learning mechanism.

In order to set up such a ML mechanism, there are two main tasks to carry out:

1. Getting and formalizing the information to be decomposed as attributes to feed the ML mechanism.
2. Defining the ML methodology to process this info

2.3 Collecting the information to create a set of most meaningful attributes

We will need to answer the following question: *Why does a user prefer a document?*

In our attempt to give an example of some possible answers, we are gradually going deeper and deeper into details in case of complex answers: *The document is interesting.*

– Because it has been stated so:
  • By the user himself (the user has already seen the document, and “told” the system, that he is interested in)
  • By someone else (someone, maybe “close” to the user, wanted to share a favorable opinion about a document)
– Because it concerns a topic close to the users interest fields:
  • by the relation with the user:
    * Personal interest fields
    * Professional interest fields (known by the his role in the enterprise)
  • by the way they are obtained:
    * Declared interest fields (the user has stated his interest documents concerning a topic)
    * Implied interest fields (the user is included in a community of interest which is close to a topic)

The second question, that introduces the notion of temporality into the preference: Why does a user prefer a document at a given moment?

In other words, to make the difference from the first question: *Why does a user prefer a document at a given moment, and does not prefer it at another moment?*

The document is interesting only if seen the first time (or the first few times)

– It is interesting during a certain period:
– When the user performs a certain activity
– Etc.
These answers are just some samples, one can think of many other possible reasons. Though, we realize that it is a very important to find the right questions and answers, that include the majority of possible situations. Indeed, getting the right questions and answers and translating them into quantifiable attributes, and making sure that the highest number of possible situations are observed is a key to the success of such a learning mechanism, that may even outclass in importance the chosen learning technique.

Nevertheless, we will present our approach in the Comma project to choose some typical answers and attributes, but we will focus more on the second issue: the preference learning methodology.

3 The design of the CoMMA system

The design of CoMMA, presented in this section, can be viewed as our attempt to implement the Knowledge Management.

3.1 The agent architecture

The chosen MAS consists of a society of coarse-grained agents, that fulfill in general multiple roles, and are organized in a small number of functional sub-societies. The MAS architecture was designed in order to optimize task-division, flexibility and robustness of the system, and network layout (extensibility, scalability, traffic optimization).

For the implementation of the prototype system, the Jade agent platform was chosen, which is an Open Source Project developed by project partners, University of Parma and CSELT. Jade is a FIPA compliant agent platform, implemented in Java, and has also the advantage of a wide opening towards Internet and the Web, interoperability with other MAS-s, and future systems.

In the current status of implementation, the CoMMA system will help the user in three main tasks:

– insertion and RDF annotation of documents,
– search of existing documents, and
– autonomous document delivery in a push fashion to provide her/him with information about new interesting documents.

The agent categories present in the system can be classified into four main areas:

1. Document and annotation management.
2. Ontology (Enterprise and User Models) management.
3. User management.
4. Agent interconnection and matchmaking.

The agents from the document dedicated sub-society are concerned with the exploitation of the documents and annotations composing the corporate memory,
they will search and retrieve the references matching the query of the user with
the help of the ontological agents.

The agents from the ontology dedicated sub-society are concerned with the
management of the ontological aspects of the information retrieval activity es-
pecially the queries about the hierarchy of concepts and the different views.

The agents from user dedicated sub-society are concerned with the interface,
the monitoring, the assistance and the adaptation to the user. Finally the agents
from the interconnection dedicated sub-society are in charge of the matchmaking
of the other agents based upon their respective needs.

We have already experimented such an architecture in Network Supervision
[EDQ96], with Machine Learning abilities [QEN97].

3.2 The learning agent

The first context to assess preference learning was chosen to be the document
retrieval scenario, via semantic annotations. The search engine used for docu-
ment retrieval in the CoMMA system is an inference engine called CORESE
[CDH00] developed by INRIA, one of the partners of the project. CORESE uses
Conceptual Graphs and combines the advantages of using the RDF language for
expressing and exchanging metadata, and the query and inference mechanisms
available in CG formalism. In order to produce inferences, CORESE exploits
the common aspects between CG and RDF: it defined a mapping from annotation
statements (RDF triples) to Conceptual Graphs and vice-versa.

One of the shortcomings of such a query retrieval engine is that there is no
standard method to sort the information returned, such as keyword frequency
in keyword-based search engines. The returned data set must be post-processed,
filtered and sorted to present the user with the relevant information. Here comes the aid offered by our ML mechanism.

In the CoMMA system, information that feeds the ML comes from several sources: The document sub-society (the annotations accompanying a query response), the user sub-society (user monitoring and explicit user feedback), and ontology sub-society (to help getting the meaning of the results). And of course the user profile. Therefore, the learning behavior was awarded to the User Profile Manager agent, which belongs to the connection dedicated sub-society, and performs notably a role of middleman between agents. This decision was justified also by network traffic optimization reasons, especially because in reaction to a user action (query), several interactions can be triggered between agents of different roles.

For example, during a query retrieval, the main interactions are as described in the following figure.
3.3 The learning cycle

The goal of the ML component is to produce a set of rules that will be used to produce predictions about user preferences. It can be supposed that the system starts with a set of predefined rules, that will be gradually improved during the process of adaptation to users. Otherwise, the system may start with an empty knowledge base, and will be undergo a training period, to accumulate sufficient knowledge to allow its deployment.

In our approach, user adaptability comes from both implicit observation of the user (user monitoring subsystem), and explicit user feedback.

We use a ”predict-or-learn” type protocol, that is, when the input is coming from query answers, the system tries to use its knowledge to predict, otherwise, when the input comes from the user in the form of negative feedback, the system tries to update its rules set.

3.4 A sample set of attributes

A sample set of attributes we used to create instances from the answers we gave as examples to the question of preference is listed in the followings. These attributes may not be the most relevant ones, we only tried to make it diverse and for most of them restricted the scope as much as possible for the sake of simplicity of our prototype. We also tried to stick close to the concepts defined in the prototype ontology we used as a basis for our first enterprise model.

Is the document related to the user’s role Supposing that the users, depending on their roles, will be assigned certain interest fields, that will be recorded in their profile, and the documents have a generic ”Concerns” property, that associates them with such topics, we can extract this information and combine it with the data extracted from the document annotation. We can define this attribute as taking a binary value.

As an observation, the notion interest field is used in a general sense to categorise documents.

In a complete implementation it is likely that this relationship can be seen as more complex and eventually be split into several attributes, or at least takes a wider range of values.

Is the document related to the user’s COINs (communities of interests) This attribute reflects one of the differences we have made in interest topics when answering the preference question: topics can be assigned or chosen by the user; inherited or inferred by the system, etc.

In the same conditions as for the attribute above, the attribute will take a binary value, and in case we define relationships, we can use the same procedure as above.
User experience at the company Since in our project addressing the New Employee scenario is particularly focused upon, we considered important making the system behave differently towards new versus experienced employees. We have segmented the range of values so that this may take such as: less than 1 week, less than 1 month, less than 2 months, ... all these regarding the novice user, and expert. Note that such segmentation (in case they are relevant at all) may vary from case to case and can be defined in the enterprise model.

The following subset of attributes are related to user monitoring.

In our example we have imagined a simple user monitoring scenario, that supposes tracking each consultation of a document by the user, and building a navigation history (or consultation trace).

**Document last seen** Usually it is important if a document was seen before or not, and if it was then how long before. After extraction from the user’s navigation history it should also be discretised into several intervals: less-than-1-hour, less-than-1-day, less-than-1-week, ... , never.

**Average return frequency** In certain cases the user may return more often to a document, in other cases an information can only present interest when first seen. This attribute may also tell something about the situation for the current case. The value will also result from processing the navigation history, and it would probably be enough to use some rough intervals (like once, small, large, etc.)

**Document category touch frequency** It might present an interest to extend the above attribute for categories the document belongs to. In this case the value will be processed the same way as for the previous attribute. In case a specific strategy for creating implicit communities of interest is envisaged, it should be checked if there are possible conflicts with the use of this attribute.

The next attributes contain specific information about the particular document being analysed:

**The user’s rating for the document** For some specific documents, the user may explicitly wish to manifest his interest or non-interest. In this case the user profile should allow storing this information. It is a general vote for the document, and does not have the temporal aspect implied by the output of this classifier.

**Public ratings** Sometimes a user would like to share his opinion about a document with others. In this case it must be foreseen in the enterprise model so that it may be stored in the form of an annotation, that can be used also by
our classifier. A method should be formalised to allow storing information about people possibly interested about this opinion.

This was a list we used in our first trials. Once again we make the remark that the list is still open, and checking the completeness and exhaustiveness of it should be an important and ongoing task. In other words, watching that these are really the factors that contribute to a document being seen as more or less relevant in a situation, and that there are not other decisive factors that were not, or can not be represented. Because in either case, the system might make major mistakes in certain situations, whatever learning algorithm was used.

4 The use of Preference knowledge

Here we define preference knowledge as knowledge about a preference between elements of a set. Such knowledge can be stated in various forms: a numerical value assigned for each item, a total ordering relation, a partial ordering relation or even a preordering of the set.

Logical models have been proposed to deal with such knowledge, some dealing directly with the comparison abilities [Sch96] and others inspired from discrete linear-time temporal logics [SR99].

It is generally admitted that a preference stands for an order that maybe partial, even a preorder, but that it is often convenient to represent it by a linear extension (which is a total order) or a numeric value compatible with the known orderings.

Then, in terms of Machine Learning, different strategies may be used, depending on the form of the preference knowledge.

4.1 Numeric labelling

A numeric labeling, i.e. a mapping of our set of examples into a set of real numbers, is a convenient way to summarize a preference relation. Some Machine Learning methods are available to learn numerical variables [Gas89,Bre96b,Bre96a].

Generally, the methods for learning to predict a numerical variable $v$ measure the quality of a predictive rule $R$ by the standard deviation $\sigma^2(v)$ of the value of the variable among the set of objects verifying the concept $R$ to be tested.

$$Q(R, v) = \frac{1}{|R(\text{true})|} \sum_{x}^{R(x)\text{true}} (v(x) - \bar{v})^2$$

The lower $Q(R, v)$ is, the better $R$ is to predict $v$, because the mean value of $v$ can be used with less error. With such criteria, any learning method will lead to grouping several neighbour values around their mean. Then, the learnt rules will not be very different from rules learnt from examples roughly rated with a finite set of values.
4.2 The order relation

By definition, a binary relation, which we note $\prec$, is an order if it has the following properties:

- reflexive: $x \prec x$,
- transitive: if $x \prec y$ and $y \prec z$, then $x \prec z$,
- antisymmetric: if $x \prec y$ and $y \prec x$, then $x = y$.

Then, we can imagine to learn the binary relation by learning each of its elements, that is, learn on each couple of objects $(a, b)$ such that $a \prec b$. Then, let us summarize the suitable properties of such a learning set for this approach to work correctly.

First, if $(a, b)$ with $a \prec b$ is an example, then $(b, a)$ is a counter-example. Then, what happens to $(a, a)$? We can see that they would be both examples and counter-examples, then it is better to consider the strict order relation, and eliminate diagonal elements.

With these hypotheses, the description of an example is made of 2 parts: the attributes which are modified between $a$ and $b$, and those which keep the same value. We can notice here that these attributes are the same as those involved in the sorting tree of our examples.

Then, our method appears to be “half lazy”, in comparison with lazy learning methods, like kNN or LWR [Aha92]. Our learned knowledge is partly explicit, but in the classification step, we need to compare a new instance with several elements of the learning set (maybe in a dichotomic way) to put it in the right place.

4.3 Statistical evaluation criteria

Usually in Machine Learning, particularly for building of decision trees, the learned classification rules are evaluated by their similarity to the desired classification.

We can use the same principle here, and we have two possible families of criteria. If we can compute a rank for each element, the similarity is computed by measuring the rank correlation to the expected ranking. Otherwise, each pair must be given: then we use a pairwise comparison between the expected order and the learnt order.

Several measures of similarity between 2 different orderings of the same data have been proposed. In each case, one has to deal with tied elements.

The Spearman rank order correlation The Spearman rank order correlation $r_s$ is a correlational measure that is used when both variables are ordinal. The traditional formula for calculating the Spearman rank-order correlation is

$$Corr(r, r') = 1 - \frac{6 \sum_{i=1}^{n} (r_i - r'_i)^2}{n(n^2 - 1)}$$
where \( r \) and \( r' \) are the ranks to compare of paired ranks. When there are tied cases they should be assigned the mean of their ranks. The mean of the ranks from \( p + 1 \) to \( p + n \) is \( \frac{1}{n} \left( \frac{(n+p)(n+p+1)}{2} - \frac{p(p+1)}{2} \right) \), which become after simplification \( \frac{n^2 + 2p+1}{2} \).

**The Kendall pairwise \( \tau \) criterion** When we have to compare each pair of data, they can be classified as either tied (T), concordant (P), or discordant (Q). The best measure for this case is Kendall’s \( \tau_b \) which takes into account a correction for tied pairs. Its formula is

\[
\tau_b = \frac{P - Q}{\sqrt{(P + Q + T_x)(P + Q + T_y)}}
\]

where \( T_x \) is the number of pairs tied on X but not Y, and \( T_y \) is the number of pairs tied on Y but not X.

### 4.4 Verifying the consistency of the method

In the case we perform a pairwise learning of the order relation, we can notice that a fundamental property, the transitivity, can be guaranteed by the learning process itself, as we show below for a version space method [Mit97]. We can check that, if, for 3 examples the transitivity holds, then it is not necessary to add the 3rd pair as example to learn the relation:

- let \((a, b) = (a_1 \ldots a_n, b_1 \ldots b_n)\) and \((b, c) = (b_1 \ldots b_n, c_1 \ldots c_n)\). Then, \( S \) is of the form \( L \land R \), with the left part \( L \) as a generalisation of both \( a \) and \( b \), and the right part \( R \) of both \( b \) and \( c \). Then, as \( L \) is a generalisation of \( a \) and \( R \) of \( c \), \( S \) is a generalisation of \((a, c)\).
- with the same conventions, \( G \) has a disjunctive form whose elements reject all the examples, then, if we represent any of its elements as \( L \land R \). If \((b,a)\) and \((c,b)\) are rejected, it means that \( L \) rejects \( b \) or \( R \) rejects \( a \), and \( L \) rejects \( c \) or \( R \) rejects \( b \). But \( G \) must also be a generalisation of \( S \).

Of course, this is only a scheme of the proof, and is, strictly speaking, only available for version-space-like learning. In a more general case, like decision tree learning, we can only make the hypothesis that it is true. We concluded that we could learn directly a sorting rule (in a greedy way, like decision trees) and evaluate the obtained rule with the \( \tau \) criteria defined in section 4.3.

### 4.5 Document ranking/sorting

In our case, the first goal is to sort documents in a query response by the order of predicted user preference. The two principal strategies that can be used to achieve that, are: numeric labeling and the order relation.

If we use numeric labeling, associating numeric values from a finite set to categories of documents representing their importance (ranking), there is a risk,
that in case of large number of items to class, many of them will fall into the
same class, and there will be no further means to differentiate them.

On the other hand, if an order relation is used, we will have no distance
measure to separates values. That is, to give an idea about for instance "how
much a document is more important then another".

Our idea is that maybe we could use both methods in a complementary way:
1. in serial-coupling (one method used to pre-process, the second to post-
process)
2. in parallel (eventually in distinct agents) then putting together the results
and solving conflicts.

5 Conclusion

The Machine Learning and user adaptability component is a very important
aspect in the CoMMA project. In this paper we presented the advances that
we have made in this domain, especially focussing on the learning of preference
data for document retrieval.

The main choice we focus on does not only present the usefulness of Machine
Learning, but also tries to overcome some of the gaps and limitations of semantic
based information retrieval systems.

In our opinion, this choice will give fast results during the first trial we
planned in the project, then allows an experimental validation through feedback
from the user.

The implementation has started and is well advancing, and we begin to have
some experimental results.

References

[Aha92] D. Aha. Tolerating noisy, irrelevant, and novel attributes in instance-
based learning algorithms. *International Journal of Man Machine Studies*,


[CDH00] Olivier Corby, Rose Dieng, and C. Hébert. A Conceptual Graph Model for
W3C Resource Description Framework. In *the 8th International Conference
on Conceptual Structures (ICCS’00)*, number LNCS 1867 in Lecture Notes in
Artificial Intelligence, Darmstadt, Germany, 2000. Springer Verlag, Springer
Verlag.

[Con00] CoMMA Consortium. Corporate Memory Management through Agents. In

[EDQ96] Babak Esfandiari, Gilles Deflandres, and Joël Quinqueton. An interface agent
for network supervision. In *Intelligent Agents for Telecommunication Appli-

[Gas89] O. Gascuel. A conceptual regression method. In Katharina Morik, editor,
*EWSL-89, 4th European Working Session on Learning*, pages 81–90, Mont-
pellier, France, Décembre 1989. Jean Sallantin and Joel Quinqueton, CRIM,
Pitman, Morgan Kaufman.

