A bidding mechanism for Web-based agents involved in information classification

Rajeev R. Raje[a], Snehasis Mukhopadhyay[a], Michael Boyles[a], Artur Papiez[a], Nila Patel[a], Mathew Palakal[a] and Javed Mostafa[b]

[a] *Department of Computer and Information Science, Indiana University Purdue University, Indianapolis, IN 46202, USA*
E-mail: rraje@cs.iupui.edu
[b] *School of Library and Information Science, Indiana University, Bloomington, IN 47405, USA*

The impact of the World Wide Web on providing an easy information access is clearly evident in all aspects of today's life. As attractive as the information availability is, due to its sheer volume, it creates an information-overload on users. Agent-based collaborative filtering is a technique used to effectively counter this burden. For agents to collaborate successfully, and maintain an overall progress of the entire interconnected environment, a governing mechanism is necessary. In this article, we present an economic framework based on a bidding mechanism for agents to communicate and negotiate with each other, thereby achieving collaborative information classification. D-SIFTER, a system developed using this economic framework, is described along with various experiments and their results.

1. Introduction

As the World Wide Web (WWW) evolves into a massive infrastructure for information, the advancement of Web technologies is becoming a primary interest to industry and academia. One of the prominent areas of development concerning the Web involves the design and implementation of *software agents*, which are defined by Genesereth and Ketchpel [1994] as, "Software agents are application programs which communicate with peers by exchanging messages in an expressive agent communication language."

Software agents have been developed for diverse application domains. Our research is targeted at using Web-based agents in distributed (or collaborative)[1] information filtering. Each agent in such an environment accomplishes tasks of *representation*, *classification*, *sorting*, and *presentation* of incoming documents in an efficient and accurate manner for its user (also referred to as owner). Thus, agents in a distributed information filtering environment aim at reducing the information overload of their owners. Through previous investigations, we have found that a collaboration among filtering agents on the WWW leads to a drastic improvement in the number of successful classifications [D-SIFTER 1997].

In order to allow agents to communicate effectively, they must be organized under a governing system. One attractive alternative for such a system uses the concept of economic framework. This is emphasized by Genesereth and Ketchpel [1994]:

> "As the Internet becomes increasingly commercialized, we envision a world where agents act on behalf of their creators to make a profit. Agents will seek payment for services provided and may negotiate with one another to maximize their expected utility, which might by measured in a form of electronic currency."

[1] Throughout this article, we will use the words collaborative and distributed interchangeably.

This article describes a bidding mechanism which allows an agent to compute the price of its service (the "bid") as offered to others and also to choose a service from a particular agent, in the context of distributed information filtering. Here, the service offered by an agent (in response to a request from another agent), involves classifying documents from the domain of computer science. Specifically, the paper is organized as follows.

We begin by providing a detailed overview of our previous work with distributed classification agents. We then describe economic models and the bidding mechanism. Next, details of the experiments investigating the effects of various parameters are discussed. The paper ends with a brief description of related efforts, our plans for future work and conclusions.

2. Previous work

The overall focus of our research is to develop an agent-based system capable of providing personalized information services to an user with as little user intervention as possible. The digital library projects (at Stanford University, Michigan University, the University of Illinois and University of California-Santa Barbara) were a direct source of inspiration for this research. It is not our intention to duplicate the efforts of the digital library initiatives, but rather to complement them by focusing on the agent-based filtering aspects, which we believe should be an important component in any digital library project.

We classify information filtering systems into two categories: *single-agent* and *multi-agent* environments. A single-agent system called SIFTER [Lam *et al.* 1996;

Sifter Research Group 1997] has been developed and implemented at IUPUI (Indiana University Purdue University Indianapolis). Our contributions include the enhancement of SIFTER by creating a distributed multi-agent environment, called D-SIFTER. This section elaborates features of both these systems and provides the results of our earlier experiments with D-SIFTER.

2.1. SIFTER

As stated above, SIFTER (Smart Information Filtering Technology for Electronic Resources) is a single-agent isolated system dedicated to classify and present the incoming textual documents to its user in an ordered scheme which is based upon its user's interests. The interests of its user are captured in a *knowledgebase*. This knowledgebase is made up of many keywords drawn from authoritative sources such as the ACM Computing Reviews Classification Scheme for documents in the domain of computer science. SIFTER is composed of three components: a *representation module*, a *classification module*, and a *user profile learning module*.

2.1.1. The representation module

The first component, the representation module, converts documents into structures that can efficiently be parsed without the loss of vital content. It uses the vector-space model [Salton 1988] for document representation because it has been widely tested and is general enough to support other computational requirements of the filtering environment. This relies on a thesaurus management submodule. At the core of the latter is a set of technical terms or concepts culled from authoritative sources representing a given area (presently SIFTER uses ACM Computing Science Classification Scheme and American Society of Information Science Thesaurus for filtering documents). The thesaurus management sub-module is also used for pruning of common functional terms, as well as several term normalization tasks including synonymy control and singular-plural form control. The popular *tf.idf* (term frequency multiplied by inverse document frequency) technique is applied to establish the particular degree of importance for each concept in a document. In the on-line filter application mode, another table is generated containing the frequencies of all unique terms found in newly arrived documents. Then the following equation is used to derive appropriate weights for terms in each document:

$$W_{ik} = T_{ik} * \log(N/n_k),$$

where T_{ik} is the number of occurrences of term T_k in document i, $I_k = \log(N/n_k)$ is the inverse document frequency of term T_k in the document base, N is the total number of documents in the document base, and n_k is the number of documents in the base that contain the given term T_k.

2.1.2. The classification module

The classification module consists mainly of two processing stages: an unsupervised cluster learning stage and a vector classification stage. During the learning stage, initial cluster hypotheses $[C^1, \ldots, C^k]$ are generated from a representative sample of document vectors $[S^1, \ldots, S^N]$. Each cluster C^i is then represented by its centroid, Z^i. During the classification stage, an incoming document vector V^i is classified into a particular class C^k using the learned centroids from stage 1. The learning of cluster centroids is done in an off-line batch mode while classification is carried out continuously as documents arrive.

A simple heuristic unsupervised clustering algorithm, called the *Maximum-distance* algorithm [Tou and Gonzalez 1974], is used to determine the centroids over the document vector space. The measure used for computing the distance between two document vectors is the *cosine similarity measure* [Salton 1988]. Given two non-null document vectors $X = [x_1, \ldots, x_t]^T$ and $Y = [y_1, \ldots, y_t]^T$, such a similarity measure represents the cosine of the angle between them and is given by

$$\sum_{i=1}^{t} x_i y_i \Big/ \sqrt{\left(\sum_{i=1}^{t} x_i^2\right)\left(\sum_{i=1}^{t} y_i^2\right)}.$$

The distance is then computed as 1 minus the similarity.

2.1.3. The user-profile learning module

The function of the user profile learning module is to determine the user's preference for the different classes of information C^i $(i = 1, \ldots, k)$, and prioritize the incoming documents based on their classes as well as the estimated user preferences for the classes. To accomplish this task, the learning agent maintains and updates a simplified model of the user, based on the relevance feedbacks. The algorithm currently used to learn the user model is based on a reinforcement learning algorithm studied in the area of Learning Automata [Narendra and Thathachar 1989] in AI and Mathematical Psychology communities.

Let d_i denote the underlying (unknown) expected user preference (relevance) for the class C^i. The learning agent maintains and updates two vectors of dimensions equal to the number of classes. The first is the *estimated relevance probability vector*, with elements $\hat{d}_i$ $(i = 1, \ldots, n)$, which is an estimate of d_i. The second is an *action probability vector* $p = [p_i]$, such that p_i represents the probability of the class C^i being selected by the filter as the most relevant class. Both p and $\hat{d}$ vectors are continuously updated during the learning process on the basis of user relevance feedback.

The learning algorithm (i.e., the algorithm for updating $p(k)$ and $\hat{d}(k)$) is described briefly as follows [Thathachar and Sastry 1985]. $\hat{d}^i(k)$ $(i = 1, \ldots, n)$ at any instant is the running average of the relevance values given by the user for documents belonging to class i. Denoting the currently maximum element of $\hat{d}$ vector as having the index l, a unit vector $E(k)$ is created of dimension n whose lth element is 1, and whose all other elements are zero. Then $P_i(k)$ ($i = 1, \ldots, n$) is updated as

$$p_i(k+1) = p_i(k) + \eta\bigl(E_i(k) - p_i(k)\bigr),$$

where $0 < \eta < 1$ is a suitably chosen step-size. Thus, the p vector is moved by a small distance towards the optimal unit vector. The convergence properties of this algorithm have been well investigated [Thathachar and Sastry 1985]. For more information on either of the three modules, the reader is referred to [Lam *et al.* 1996; Mostafa *et al.* 1997; Sifter Research Group 1997].

2.2. D-SIFTER

In SIFTER, if an agent cannot classify a document due to its inadequate knowledgebase and/or limited expertise and/or less comprehensive training set of documents, it marks that document as a failure. This is an undesirable situation. D-SIFTER (Distributed SIFTER) provides the necessary mechanism to prevent such a situation, by allowing an agent to collaborate with other agents. This collaboration helps to achieve better classifications of the documents for users. Using SIFTER as the basis for document representation and classification, we implemented D-SIFTER.

2.2.1. Implementation using Java-RMI

We chose to implement D-SIFTER in Java [Cornell and Horstmann 1997; Flanagan 1996; Siyan and Weaver 1997] because of its portability, object-oriented features, APIs for GUI and RMI (Remote Method Invocation). The collaborative actions in D-SIFTER are implemented as remote calls over the network using RMI.

2.2.2. System organization

As D-SIFTER enhances SIFTER, it was logical to base its design on the model of SIFTER. The representation and classification modules of D-SIFTER are identical to their counter-parts from SIFTER.[2] The collaborative features of D-SIFTER are provided by the *distributed module*. This module is responsible for the communication among agents. Our research involves the exploration (through experimentation) of the effective paradigm for the distributed module. The next subsection describes two such paradigms, which we have incorporated for the agent-based distributed module.

2.3. Collaborative paradigms

Figure 1 shows the architecture of the distributed module. Currently, all agents are identical and they continuously perform the task of document classification on behalf of their owners. The communication among different agents takes place in an indirect manner through a common server. It should be noted that the role of the server is just to

[2] The incorporation of the user profile learning module is currently under investigation.

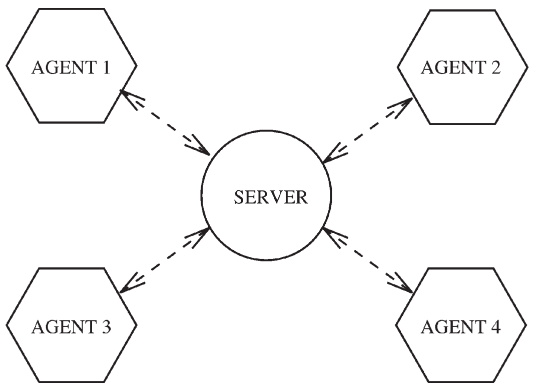

Figure 1. Distributed module communication architecture.

provide a channel for broadcasting the relevant information to all the agents. The classification, ranking and presentation of documents is carried in a distributed fashion by the agents and the server is not responsible for executing any of these tasks. With the intent of studying collaborative ideas, we chose to implement two types of collaborative schemes: the *single-* and *multiple-opinion* models. Each agent in both models follows the same general algorithm:[3]

```
1. Attempt to classify the next document.
2. If successful, present the result
      to the owner.
   If not then request remote assistance.
3. If a pending remote request exists
      then check its status.
   If the result is available, fetch it and
      present it to the owner.
4. Check to see if another agent
      needs assistance.
   If so, help that agent.
5. Go back to 1.
```

While the general algorithm is similar, the difference arises from the number of remote collaborators which are allowed to assist a requesting agent. The single-opinion model allows only one remote agent to assist another agent while the multiple-opinion model allows more than one remote agent the chance at remote collaboration.

2.3.1. Single-opinion

The strength of this model lies in its simplicity. It is easy to implement and as only one remote agent is chosen to assist with the classification, the necessary turn-around time for remote collaboration is small.

The simplicity of the model also results in its drawbacks. The single-opinion model allows only one agent to perform the remote service. Currently, the collaborator is chosen randomly. This can be a serious drawback causing the document never to be classified. Also, even if the remote agent can perform a classification, how can the owner be assured that he/she received the most accurate classification that was possible?

[3] For the complete algorithms, see [Raje *et al.* 1997c].

2.3.2. Multiple-opinion

The multiple-opinion model aims to compensate the drawbacks of the single-opinion model. It represents the real-world by allowing different agents (with varied expertise) to participate in the collaboration. Such a collaboration will lead to better classifications [D-SIFTER 1997]. Also, this model elegantly handles the discovery of new keywords, which form the knowledgebase. By providing agents with the facilities to communicate with others, keywords can be passed along which will help in improving knowledgebases of some agents and reducing the need for remote assistance. Also, as with any distributed system, more agents lead to a more stable environment, thus failures can be handled gracefully.

Most difficult problems of this model involve managing multiple remote agents. The critical issues include waiting for all agents to complete their classification, waiting for all agents to produce a bid (in a market driven system), and an added overhead of determining the most appropriate agent's classification (also known as *selection criteria*).

2.3.3. Single- and multiple-opinion preliminary results

We ran experiments using each model to observe the change in the number of successful classifications. Results from both models showed an increase in the total number (local and remote) of classifications that were successful. The single-opinion model showed an improvement of 42% while the multiple-opinion increased the classifications by as much as 61% [D-SIFTER 1997].

Another experiment involved only the single-opinion model. When only one remote agent is requested to assist, we decided to add an additional constraint called, *time-out period*, i.e., how long should an agent wait to obtain the remote classification? We studied the relation between the time-out period and its effect on the total number of successful classifications. The results were as expected: the number of successful classification increased until the time-out period reached a peak value. After the peak value, the number of successful classifications did not change [Raje *et al.* 1997b]. For more information on either the single- or multiple-opinion models, the reader is referred to [D-SIFTER 1997].

As stated earlier, as the Internet becomes more commercialized, users will want their agents to act only for compensation. This approach is best described within a market-based environment. The next subsection outlines the economic models we have used in D-SIFTER.

2.4. Economic models

As stated in the introduction, we are interested in investigating the suitability of economic models for collaborative information filtering. To explore this area in detail, we have implemented two economic models, *user-centric* and *money-centric*, for the single-opinion version of D-SIFTER. Throughout the rest of this paper, the term *information dollar* or *money* refers to a form of electronic currency that is agreed upon by the agents/users.

In the user-centric model, the quality of service offered by the agent to its owner is the top priority. The agent performs remote services only if it has no documents to classify for its owner.

The money-centric model takes a greedy approach towards remote assistance. In this model, an agent performs remote assistance whenever it sees an opportunity.

We conducted many experiments with both these models. As expected, the agents in the money-centric model gained money quicker and it took much longer for their local documents to be classified. The details of these experiments and their results are described in [Raje *et al.* 1997a].

3. A bidding mechanism for a multiple-opinion scenario

There are many possible approaches to the multiple-opinion scenario. For example, an agent requesting a service can contact other agents in a round-robin fashion, get their quotes and divide its tasks in an uniform fashion. Such an approach is simplistic and may result in an uniform load distribution. However, we think that this technique has a few drawbacks. First, it requires an agent to be aware of all other collaborators in the system. Second, there is always a trade-off between balancing the load uniformly and getting the most economical service. Third, a round-robin selection by each agent may or may not result in a uniform load distribution in the entire system. Fourth, each agent wanting to offer an service, may be required to indicate its current load to all of its clients, so that the clients can make appropriate selections. Finally, we think that this simplistic mechanism does not emulate the real-world situation.

For the multiple-opinion scenario, we are proposing a bidding mechanism, in which each agent specifies a quote for its service and the agent requesting an assistance makes a selection based upon these quotes.

3.1. Issues in a bidding mechanism

The bidding mechanism is a natural extension of the single-opinion economic framework. However, the presence of multiple agents gives rise to many interesting questions: *Why (or when) should an agent bid? How much time should the requesting agent wait? How much should an agent bid? How should the requesting agent decide which remote agent to choose?* There are many valid answers to these questions. In the following paragraphs, we will discuss possible alternatives and present our approaches.

3.1.1. Why should an agent post its bid?

It is possible that an agent may want to become a "free rider", i.e., choose not to bid. Some of the factors influencing such a decision include addressing questions such as: *Is* *the expected revenue larger than expected costs? Is the expected effort worth the bid?* In D-SIFTER, we have taken a simplistic view, called as the *principle of ever-readiness*, i.e., each agent will always be ready to offer assistance by posting its bid. The reason for this assumption is the very basis of distributed filtering. In a decentralized system, such as D-SIFTER, it is conceivable to imagine a situation in which a single agent cannot complete the task of classification for its owner. However, through collaboration with other agents, this task could be successfully executed. In a market based system, each assistance requested has a price associated with it and hence, in order to seek collaboration, agents must have sufficient amount of money. As agents receive documents continuously, it is obvious that irrespective of the initial financial situation of each agent, there is a need for an alternative to earn money. The only way an agent can earn money is by offering its assistance to others. In addition to improving each agent's financial state, the principle of ever-readiness ensures a constant progress of the entire collaborative environment.[4]

3.1.2. What is the effect of time on a bid?

To address the effect of time, an agent may impose a time limit along with its bid. This time-limit will decide the validity of the bid. The validity of a bid could be static or dynamic, i.e., does not change over a substantial amount of time or change very frequently. In D-SIFTER, each agent calculates its bid in every cycle, where a cycle indicates one iteration of the algorithm mentioned in section 2.3. This bid is valid only until the next cycle. Whenever an agent needs assistance, it checks the current bids posted by all other agents and makes a decision about selecting a collaborator. The implementational details of an agent are discussed later.

3.1.3. What should be the value of a bid and how to select a bid?

The calculation of the amount of a bid is complex, as it depends upon many internal and external factors. The internal factors are based upon each agent's (and thereby its owner's) desires, while the external factors depend upon the overall filtering environment. Also, the selection of a bid from all the available ones, depends upon issues such as the magnitude of the bid and assured quality of service (or confidence about a collaborator's competence).

In D-SIFTER, we have used a model, in which the amount of a bid is dependent upon *the initial dollar value, the current financial state and the history of document classifications*. While selecting a bid, we have assumed a uniform guarantee about the quality of service and hence, an agent selects a collaborator whose bid is the least expensive.

Next subsections describe the factors influencing a bid and our approach for calculating a bid in a market-driven information classification environment.

[4] A continuous participation of agents is a critical concern expressed by experts at the collaborative filtering workshop [Avery 1996].

3.2. Factors influencing a bid

As stated previously, internal and external factors influence a bid. When an agent makes a bid, that bid is a reflection of its present state, which is dependent upon two factors: *the current information dollar value and the probability of the next document needing a remote classification*.

The first factor is an agent's current dollar amount. This is important because, in a market driven environment, an agent only earns money when it performs a service and pays money when it needs a service. Hence, the current financial state of an agent is directly controlled by the number of remote classifications it asks and provides. The second factor is the number of documents that the agent is not able to classify locally. The more documents the agent needs to request help with, the more money it will need.

Once we determined these two factors, we considered extreme cases involving them. If an agent has a small amount of money and a small amount of expected remote classifications (a factor which will be discussed in detail soon), it will need as many jobs as it can receive, and hence, it will bid a minimal amount. On the other hand, if the agent has a lot of money and a small number of expected remote classifications, it can surely afford to lose some jobs. This would cause the agent to bid a high amount.

The other extreme scenario occurs when an agent has a large number of expected remote classifications and only a limited amount of money to pay for these services. In such a situation, the agent cannot afford to miss any job, and hence would charge only the minimal amount. On the other hand, having a large number of expected remote classifications, and a large amount of money, an agent should bid high as it is not bothered with the cost of remote services.

Using these two factors, we formulated a bidding function for all agents which is described in the next section.

3.3. The bidding function

We began by introducing an upper and a lower limit for all bids. The lower limit (minimal amount) was set to zero and the upper limit (maximum amount) was set as a parameter, *MaxBid*. The actual bid is computed as a product of the *bidding factor* and the *MaxBid*. The bidding factor is a function of the two parameters discussed earlier – the current financial state of an agent and the probability of next document needing remote classification. The bidding factor is a weighted sum of these two parameters. The weights are denoted as α and $(1-\alpha)$.

Figure 2 shows a schematic of the computational mechanism. This diagram produces the following formula:

$$\mathit{ActualBid} = \mathit{MaxBid}\big((1-\alpha)P_{\mathrm{r}} + \alpha P_{\$}\big),$$

where *MaxBid* is the maximum amount an agent may bid, α and $(1-\alpha)$ are weighting factors, P_{r} is the probability that the next document cannot be classified locally, and $P_{\$}$

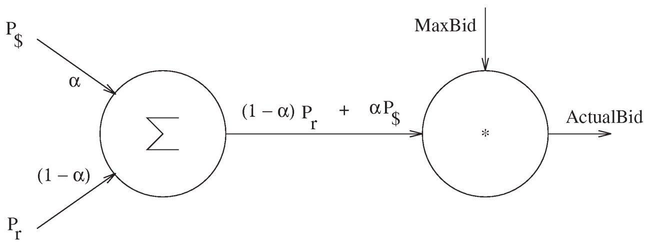

Figure 2. Schematic of bidding function.

is the ratio of current dollars to initial dollars, which is the indication of each agent's current financial state.[5]

As stated previously, P_{r} is the probability that the next document will need remote collaboration. As future cannot be known in advance, we can only make predictions about it. By assuming that an user's interests do not change very rapidly, we used the *principle of locality* and adopted a *working-set model* (or a *working-set window*). The principle of locality says that the future behavior of the system will most likely be similar to its past, i.e., the probability of the next document needing a remote assistance will be similar to the experience with the past documents, i.e., *history of classification*. The working-set window defines how many documents constitute the history. If the window is small, the agent would rely on its most recent past to make predictions about the future. On the other hand, if the window is very large, the prediction of future is influenced by the nature of the documents from distant as well as recent past. Once the window size is specified by the user, P_{r} is calculated by dividing the number of documents needing remote collaboration within the window by the total number of documents within the window.

α is the weighting factor. It can be adjusted according to each user interests. Its value is between zero and one and indicates the influence of each of the two parameters involved in the bidding calculations on the actual bid. A value of zero for α would reduce the bidding factor to P_{r}, while a value of one makes the bidding factor to be equal to $P_\$$.

3.4. Implementation

In order to experiment and validate our model, we implemented the bidding mechanism onto D-SIFTER. Each agent in our prototype continuously executes the following algorithm:

```
1. Calculate a current bid based upon
   bidding formula and post it.
2. Attempt to classify a local document,
   if successful, display the result
   to your owner.
3. If necessary, obtain remote assistance
   from lowest bidding agent.
4. If remote agent has selected you,
   classify its document.
5. Check to see if your collaborator
   has posted the result for you,
   if it has, then fetch
   and display that result to your owner.
6. Go back to 1.
```

[5] It is obvious that in some cases, $P_\$$ can be larger than 1.

The next section describes the experiments we conducted after implementing the mechanism previously described.

4. Experiments and observations

We conducted two sets of experiments using SUN Sparc4 or SUN Ultra1 machines running the Solaris Operating System and connected via an Ethernet. The first set of experiments included three agents and the document set consisted of 1064 documents from the domain of computer science. Each agent had a disjoint knowledgebase. The second set of experiments included five agents and the document set consisted of 5335 documents from the domain of computer science. The knowledgebases of agents were overlapping in nature.

4.1. Three agents

Table 1 shows the number of local and remote classifications possible for each of three agents. The second column in table 1 contains the total number of documents (out of the possible 1064) that can be classified locally and the third column indicates the remaining documents that must be classified remotely. Only 66 of the original 1064 can not be classified by any of the three agents. This experiment was conducted earlier during the preliminary work. For more information on it or the other preliminary experiments, the reader is referred to [Raje *et al.* 1997b].

The results in table 1 provided a clear picture about the classification abilities of all three agents. These results acted as a starting point for our experiments with the bidding mechanism.

4.1.1. Constant α and constant window size

The first set of experiments studied the behavior of the three agents with the same values of α and a constant size window of 50 documents. We ran experiments with three values of α, 0.0, 0.5 and 1.0. The minimum bidding value was \$2 and the maximum bidding value was \$10. Each agent was given \$500 at the beginning.

Tables 2 and 3 show the data collected for the experiment with $\alpha = 0.0$. Table 2 indicates the data related to the local classifications of each agent, while table 3 shows the

Table 1
Statistics for the multiple-opinion version of D-SIFTER.

Knowledgebase	Successful local	Successful remote	Failed remote
Agent 1	676	322	66
Agent 2	350	648	66
Agent 3	636	362	66

Table 2
Data related to local documents of each agent.

Agent	Local successful	Remote successful	Remote unsuccessful	Remote un-affordable
1	676	259	129	0
2	350	132	55	527
3	636	300	128	0

Table 3
Data related to remote documents received by each agent for the other agents.

Agent	Total # received	Successful	Failed	# of A1's docs.	# of A2's docs.	# of A3 docs.
1	612	431	181	–	184	428
2	2	1	1	2	–	0
3	389	259	130	186	3	–

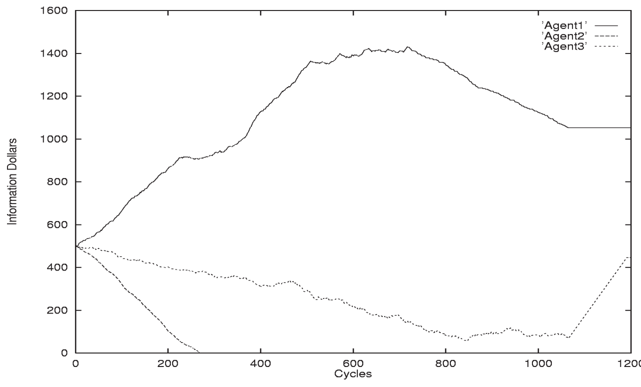

Figure 3. Cash-flow for $\alpha = 0.0$.

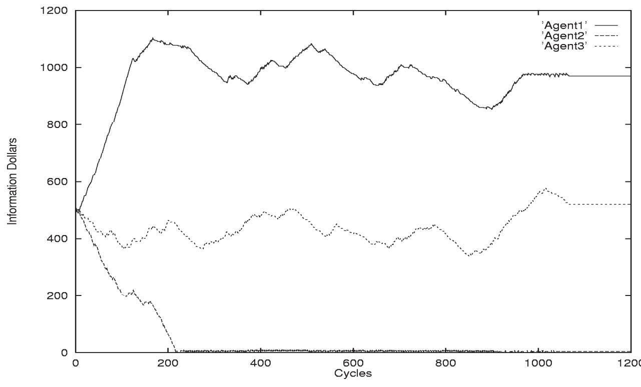

Figure 4. Cash-flow for $\alpha = 0.5$.

data related to the remote classifications performed by each agent for the other agents.

Figure 3 is the graph of information dollars as a function of each execution cycle[6] for $\alpha = 0.0$. As seen from it, Agent 1 clearly dominates the others and shows a substantial gain in its information dollars, while Agent 2 quickly drops to a minimal amount. This behavior is hardly surprising, as Agent 1, which gained the most, has the *best* knowledgebase, while Agent 2, which lost all its money, has the *worst* knowledgebase.[7] The term best/worst is an indication of the completeness of a knowledgebase in assisting an agent to achieve a large/small number of local classifications. The highest gain achieved by Agent 1 is due to a combination of two factors: (a) it requires the least number of remote classifications, and (b) as α is 0.0, its bid is always the least (as the bid in this case depends only on P_r). Similarly, the disaster of Agent 2 is a compound effect of its need for a large number of remote classifications and a highest bid which it offers. The impact of economic downfall of Agent 2 is evident in table 2 – Agent 2 sought assistance of others for only 132 (successful) +55 (unsuccessful) documents, and it could not afford seeking assistance for 527 (which is 49.5% of total 1064) documents. The knowledgebase of Agent 3 is equally good

[6] Each execution cycle corresponds to an iteration of the filtering algorithm discussed earlier.
[7] This observation is evident from the table 1.

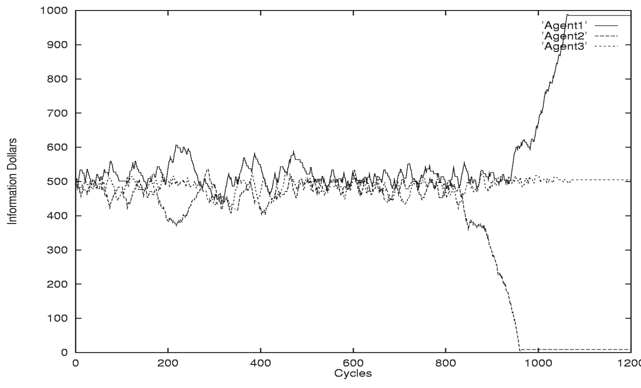

Figure 5. Cash-flow for $\alpha = 1.0$.

as that of Agent 1, and hence, its bids were comparable to those of Agent 1. However, on an average, bids of Agent 1 were less than those of Agent 3, thus, Agent 1 gained more than Agent 3.

Figure 4 indicates the situation for α = 0.5. Here also, Agent 1 gains the most, Agent 2 loses the most and Agent 3 stabilizes to the initial dollar amount.

Figure 5 demonstrates a slightly different behavior. As this graph corresponds to $\alpha = 1.0$, the agents rely solely on their current dollar amount for calculating a bid. Since all agents start with the same initial amount ($500), the bids are competitive and all agents maintain a similar amount of money for a substantial period. Towards the end they begin

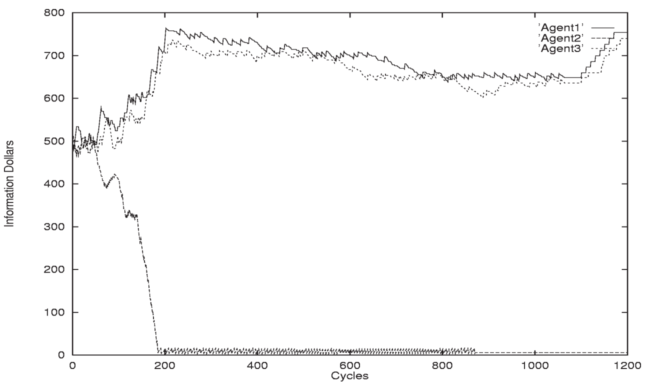

Figure 6. Cash-flow for $\alpha = 1.0$.

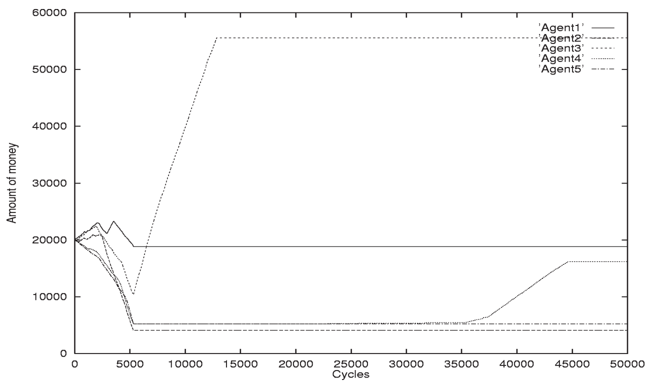

Figure 7. Cash-flow for $\alpha = 0.0$.

to break from one another. The sharp incline for Agent 1 is caused by the current implementation of the environment. Once an agent goes over the initial amount, we consider that agent in the *safe zone* and its $P_\$$ amount is set to one, which enforces the upper limit (as specified by *MaxBid*) on any bid. This meant that both Agent 1 and Agent 3 produced the same bid even though, Agent 1 had gained more than \$500, thus creating a tie situation. The current implementation took Agent 1 as the service provider in the case of a bidding-tie. As this is not a desirable situation, we conducted another experiment with the same value of α but neglected the upper limit for bidding amount by not truncating the value of $P_\$$.

Figure 6 shows the new results. This time the behaviors of agents were similar to the first two experiments. Agent 2 quickly dropped to a low value due to its lack of ability to classify documents locally while the other two fought over the remote work. This phenomenon did not occur in the previous experiments because the bid was determined partly or totally by P_r. When $\alpha = 1.0$, P_r plays no role in the bidding process and since Agents 1 and 3 had similar knowledgebases, the distribution of the work to Agents 1 and 3 was fairly even. Hence, the gains at the end of the experiment for Agents 1 and 3 were comparable.

4.2. Five agents

We ran the second set of experiments with five agents and a larger document set consisting of 5335 documents to investigate the scalability of the system. Before dividing the knowledgebase among five agents, we conducted a preliminary experiment to check the comprehensiveness of the complete knowledgebase. The full knowledgebase was able to classify 3728 out of 5335 documents (nearly 70%).

Next, we observed the behavior of agents in three different scenarios. The first scenario was with the same value of α. We ran experiments for three values of α: 0.0, 0.5, 1.0. In the second scenario, we studied the behavior of agents with each having a different value of α and the constant window size of 50 documents. Values of α were set to 0.0, 0.25, 0.50, 0.75, 1.0 for Agents 1–5, respectively. In the third scenario, agents retained their α values from the second scenario. However, their window sizes differed from each other. Agents 1–5 had their window sizes set to 250, 200, 150, 100 and 50, respectively. We selected the minimum bidding value to be \$2 with no limit set on the maximum bid. Each agent was given the \$20000 at the beginning.

4.2.1. Constant α and constant size window

Tables 4 and 5 show the data collected for the experiment with $\alpha = 0.0$. Figure 7 is the graph of information dollars as a function of each execution cycle for $\alpha = 0.0$. As seen from the tables, Agent 3 has the best knowledgebase out of all the five agents. It is able to classify locally roughly twice as many documents as other agents. The highest gain achieved by Agent 3 is a result of combination of the least number of remote classifications and the lowest bid on average. As $\alpha = 0.0$, the bid depends solely on P_r. Similarly, Agents 2, 4 and 5 could not compete with Agent 3, because their P_r was higher and thus they finished with a lesser amount of money. A rather high number of unsuccessful remote classifications for all agents is not surprising, as each agent owns nearly only one fifth of the whole thesaurus.

Figure 8 indicates the situation for $\alpha = 0.5$. The value of α forces the bid to be dependent on the current dollar amount as well as the probability that the next document will need remote collaboration. Agent 5 has the worst knowledgebase, therefore it spends most money and the probability that its next document will not be classified locally is high. The least amount of money causes its bid to be the lowest, which in turn brings in most work. The losses of Agents 1, 2 and 4 are due to their lack of competitive bids and poor knowledgebase. These two factors cause low income and high spending.

Table 4
Data related to local documents of each agent.

Agent	Local successful	Remote successful	Remote unsuccessful
1	1330	1371	2634
2	1187	1263	2885
3	2200	436	2699
4	1334	1006	2995
5	1078	1000	3257

Table 5
Data related to remote documents received by each agent for the other agents.

Agent	Total received	Successful	Failed	A1's docs	A2's docs	A3's docs	A4's docs	A5's docs
1	3885	896	2989	–	401	1544	839	1101
2	1440	382	1058	260	–	211	392	577
3	12740	3710	9030	3714	3725	–	2770	2531
4	1472	88	1384	31	21	1372	–	48
5	9	0	9	0	1	8	0	–

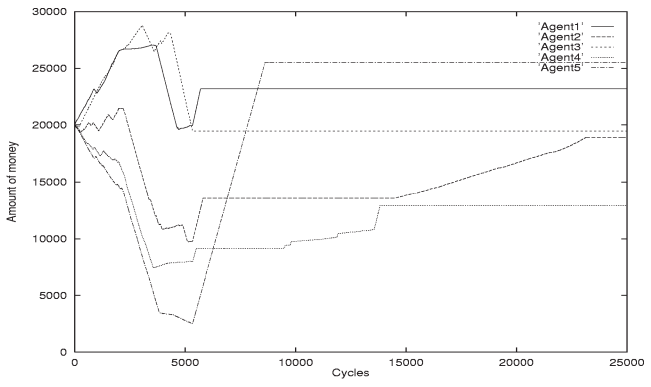

Figure 8. Cash-flow for $\alpha = 0.5$.

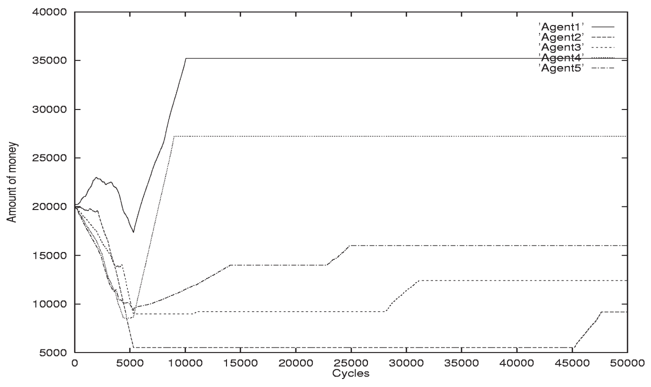

Figure 10. Cash-flow for 5 agents with different α each.

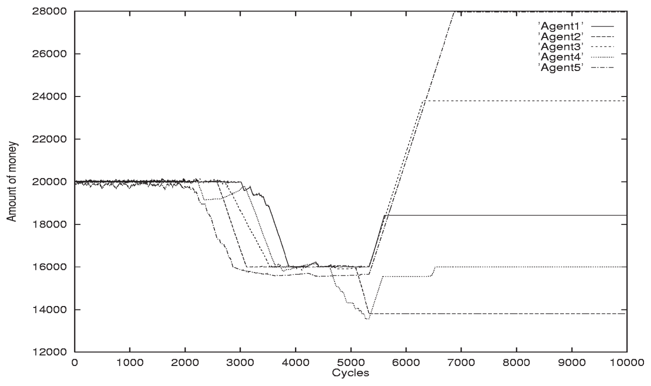

Figure 9. Cash-flow for $\alpha = 1.0$.

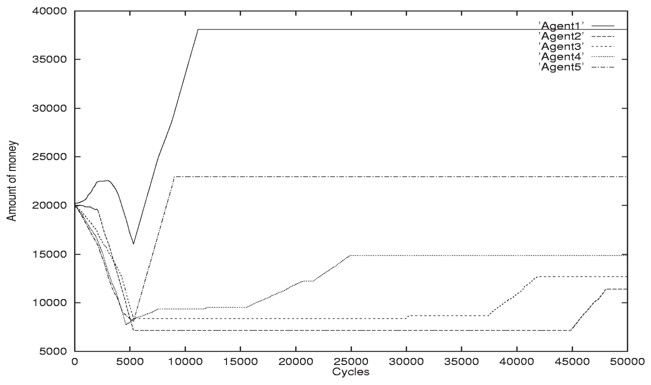

Figure 11. Cash-flow for 5 agents with different α and different size window each.

Figure 9 demonstrates different behavior. As this graph corresponds to $\alpha = 1.0$, the agents rely solely on their current dollar amount for calculating the bid. Once again, Agent 5, who had to seek remote help more times than other agents, was posting the lowest bid, thus bringing in most work.

4.2.2. *Different α and constant size window*

Figure 10 shows the cash-flow for agents with different α values. Agents 1–5 have their α values set to 0, 0.25, 0.5, 0.75 and 1.0, respectively. Only Agent 1 and Agent 4 were able to finish with profits. Due to the fact that $\alpha = 0.0$ for Agent 1, its bid is dependent on the P_r only. In that case, the lower the probability that the next document will require a remote assistance, the lower the bid. Since the knowledgebase of Agent 1 is only considerably inferior to that of Agent 3 (as seen from table 4), its P_r is lower than other agent's P_r (except for the Agent 3). This and the fact that the positive income does not affect the bid, make the offer of Agent 1 the most competitive causing the highest gain.

Agent 4 achieved the second highest gain. Its success is the effect of the combination of the low $P_\$$ (which dominates the bid estimate when $\alpha = 0.75$) and the P_r comparable to the one of Agent 1 (as seen from the table 4). Agent 3, which in the past examples was always one of the best performers, suffered a considerable loss this time. This economic downfall was provoked by the combination of its low $P_\$$ and the low P_r. These two factors made the bid of Agent 3 unattractive to other agents.

4.2.3. *Different α and varying size window*

Figure 11 shows the results obtained by assigning different α and different window sizes for all five agents. Agents 1–5 have their α values set to 0, 0.25, 0.5, 0.75, 1.0 and their window size set to 250, 200, 150, 100, 50, respectively. The results are similar to those in figure 10 with one striking difference. This time, Agent 5 finished second while Agent 4 finished third. Everything else remained similar. The $\alpha = 1.0$ and the window size of 250 documents increased the competitiveness of Agent 5 over Agent 4. Since the bid of Agent 5 depends solely on P_r, the increased size of the window (from 50 documents to 250 documents) has decreased the probability that the next document will require a remote assistance. Lower P_r caused the bid to be lower making it more competitive than it was in the case of the window size of 50 documents. The bid of Agent 4 which had $\alpha = 0.75$ and the window size of 200 documents was determined by the combination of these two factors. Agent 4 was less competitive than Agent 5, because Agent 5 had a lower P_r due to its bad knowledge-base.

5. Related work

Currently, there exists many collaborative information assistance tools and each has a very different focus. This section briefly describes only a small number of systems. It should however be noted that the concept of collaboration (or distribution) in D-SIFTER is slightly different from the one used in other research efforts described in [Avery 1996]. A majority of these efforts provide prediction by aggregating the individual recommendations at one central place, while our approach relies more on decentralized forum, in which autonomous agents negotiate, learn and assist each others.

Laskari *et al.* [1994] describes interface agents providing personalized services such as meeting scheduling, e-mail handling, news filtering, and entertainment selection. They provide [Laskari *et al.* 1994] an illustration of experienced interface agents helping a new agent to deal with unfamiliar situations as well as to speed up the learning process.

Implemented at Bellcore [Avery 1996; Bellcore 1997], *GAB* uses hierarchical decompositions made by users and provides browsing facilities over Web pages. GAB's mission is to provide browsable interfaces for pages which are represented by equivalence classes. A summary of GAB's features indicate that GAB could be very useful for the middle-level user but disappointing for the novice or expert [Avery 1996]. One of the biggest drawbacks of GAB includes its ability to reach into someone else's bookmarks. This raises privacy issues.

Phoaks is yet another web-based filter application. Phoaks uses the frequency of mention data within Usenet news groups. The AT&T Research group is now investigating how far they can go without asking the user for data. They are also exploring human interface issues. For more information on the Phoaks project, please see [Avery 1996; AT&T Research Group 1997].

Developed by John Riedl and Paul Resnick [Avery 1996; Riedl and Resnick 1997], *GroupLens* uses Usenet newsgroups as a domain to produce quality predictions about certain articles. The open architecture feature allows developers to create clients which work with GroupLens servers or even replace them if necessary. One of the biggest drawbacks about the system is the size of its domain. Usenet offers the potential of approximately 22 million user. With this many users reading and ratings articles, the database is simply huge [Avery 1996].

Andreoli *et al.* [1995] describe a constraint-based agent framework that has direct application in complex information environments such as the WWW. The framework employs Constraint-based Knowledge Brokers (CBKBs) to conduct knowledge-intensive tasks by exploiting constraints and reusing information. Using this framework, for example, it is possible to handle queries generated in the WWW by transforming them into constraints on brokers, with "specialization" or prior knowledge of diverse sources, that ultimately cooperate and compose a solution-set to the queries. Several types of agent-interaction protocols (e.g., direct request-subrequest, broadcast with local caching, problem-tuned, etc.) are supported along with an elegant model of cooperative agents (brokers). The strength of the CBKB framework is that it is extremely general in nature, with a solid foundation in the distributed problem solving (DPS) paradigm, that can be adapted to a variety of agent negotiation strategies.

6. Future work and conclusion

Many avenues of future work stem out from our current research. One interesting possibility is to allow subcontracting and extracting commissions, while performing the classifications. Other alternatives to be explored include associating a time period with each bid and allowing a bidding war. A bidding war would permit an agent to adjust its bid based upon the bids of other agents. As described in this paper, the bids are computed by using the history of local/remote classifications and current financial state of the agent. We plan to integrate other factors such as the quality of service, past experiences with the service providers and load-distribution in the bidding calculations. We also look forward to investigating a more complicated mechanism for collaborator selection. At present, D-SIFTER is tested only with documents from the domain of computer science, we plan to use D-SIFTER to classify documents from an entirely different field.

An important component which is present in SIFTER, but not yet implemented in D-SIFTER is the user-profile module. An incorporation of this module into D-SIFTER will enhance the architecture of agents and allow agents to learn about the behavioral changes in the interests of their owners.

At present, D-SIFTER only includes one aspect of collaboration, namely, asking for assistance when it cannot classify documents locally. It is our intent to add the second (and more complex) component of collaboration – an ability to seek and negotiate useful information by a continuous interaction with other agents over the Web.

In the information age, agents are expected to assume a large responsibilities on behalf of their owners. In order to satisfactorily serve its owner, an agent must not only know owner's interests but also have an ability to negotiate with other agents in a successful manner. In this article, we have described an effective bidding framework which allows agents to achieve a collaborative classification of incoming documents for their owners. The bidding function, which we have used considers the impact of the present financial state of an agent and its prediction about future. Our preliminary results emphasize the feasibility of a market-based economic system in a distributed information classification environment.

References

AT&T Research Group (1997), "Phoaks Web Site." `http://www.phoaks.com/phoaks/.`

Avery, C. (1996) "Summary of Proceedings," In *Collaborative Filtering Workshop*, Berkeley, CA. `http://www.sims.berkeley.edu/resources/collab/conferences/berkeley96/collab-announce.html/.`

Bellcore (1997), "Web Site." `http://www.bellcore.com/WWWCONF/ARTICLES/04/adaptx.html.`

Cornell, G. and C. Horstmann (1997), *Core Java*, 2nd Edition, Sunsoft Press, Mountain View, CA.

D-SIFTER (1997), "D-SIFTER Research Project @ IUPUI." `http://klingon.cs.iupui.edu/~sifter/.`

Flanagan, D. (1997), *Java in a Nutshell*, 2nd Edition, O'Reilly & Associates, Sebastopol, CA.

Genesereth, M. and S. Ketchpel (1994), "Software Agents," *Communications of the ACM 33*, 7, 48–53.

Karanjit, S.S. and J.L. Weaver (1997), *Inside Java*, New Riders Publishing, Indianapolis, IN.

Lam, W., S. Mukhopadhyay, J. Mostafa and M. Palakal (1996), "Detection of Shifts in User Interface for Personalized Information Filtering," In *Proceedings of 19th International SIGIR Conference on Research and Developement in Information Retrieval*, ACM, New York, NY, pp. 317–325.

Laskari, V., M. Metral and P. Maes (1994), "Collaborative Interface Agents," In *Proceedings of AAAI Conference*, AAAI Press, Menlo Park, CA, pp. 444–449.

Mostafa, J., S. Mukhopadhyay, W. Lam and M. Palakal (1997), "A Multilevel Approach to Intelligent Information Filtering: Model, System and Evaluation," *ACM Transactions on Information Systems 15*, 4, 368–399.

Narendra, K.S. and M.A.L. Thathachar (1989), *Learning Automata – An Introduction*, Prentice-Hall, Englewood Cliffs, NJ.

Raje, R., S. Mukhopadhyay, M. Boyles and A. Papiez (1997a), "An Economic Framework for a Web-Based Collaborative Information Classifier," In *Proceedings of the International Association of Science and Technology for Development, SE'97 Conference*, IASTED/ACTA Press, Anaheim, CA, pp. 362–366.

Raje, R., S. Mukhopadhyay, M. Boyles, N. Patel and J. Mostafa (1997b), "D-SIFTER: A Collaborative Information Classifier," In *Proceedings of the First International Conference on Information, Communications, and Signal Processing*, IEEE Singapore Section, Singapore, pp. 820–824.

Raje, R., S. Mukhopadhyay, M. Boyles, N. Patel and J. Mostafa (1997c), "On Designing and Implementing a Collaborative System Using Java-RMI," In *Proceedings of the Fifth International Conference on Advanced Computing*, Tata–McGraw Hill, New Delhi, India, pp. 404–411.

Riedl, J. and P. Resnick (1997), "GroupLens Web Site." `http://www.cs.umn.edu/Research/GroupLens.`

Salton, G. (1988), *Automatic Text Processing*, Addison-Wesley, Reading, MA.

Sifter Research Group (1997), "SIFTER." `http://129.79.33.62/jmdocs/restext.html.`

Thathachar, M.A.L. and P.S. Sastry (1985), "A New Apporach to the Design of Reinforcement Schemes for Learning Automata," In *IEEE Transactions on Systems, Man, and Cybernetics 15*, 168–175.

Tou, J.T. and R.C. Gonzalez (1974), *Pattern Recognition Principles*, Addison-Wesley, Reading, MA.