Clustering and Information Sharing in an Ecology of Cooperating Agents

Leonard N Foner
Agents Group, MIT Media Lab
E15-305, 20 Ames St
Cambridge, MA 02139
foner@media.mit.edu

Abstract

Software agents have become increasingly popular for a variety of applications, many of which benefit from distribution on a network. However, many common approaches scale poorly in such an environment, because they assume the feasibility of knowing about all or most of any given agent's peers.

We investigate here some approaches and applications which will serve as a testbed for evaluating solutions to the above problems. These approaches involve clustering agents into clumps or "niches" that need only local knowledge; this work hence has relevance in the fields of learning, knowledge sharing, and distributed AI.

Parts of the ideas above have been implemented, but many of them are speculative at this point, and feedback is actively encouraged.

1 Introduction

Software agents have become an increasingly popular approach in dealing with information discovery and information filtering in a networked environment, either for the purposes of utility (e.g., [Maes 94], [Lashkari et al 94], etc), or entertainment (e.g., [Mauldin 94] [Foner 93]).

Many foreseeable future applications for software agents (such as extensions of collaborative interface agents [Lashkari et al 94] or social filtering [Shardanand 95], to pick just two of many) involve large numbers of agents interacting with each other. Users may have a number of agents operating on their behalf, and agents of any particular user may have to communicate with other agents elsewhere on the network in order to share information. The overall organization may resemble a *computational ecology* [Huberman 88], in which agents have only local knowledge, but self-organize into larger units.

Such ecologies of agents present several scaling issues, however, one of which is how agents are supposed to find each other. Each agent should not have to know about (and, indeed, probably *cannot* know about) every other agent, user, or resource on the network. Perhaps a worst-case for such systems involve *matchmaker*-like problems, in which, from a set of peers, we must find good matches based on some attribute. Straightforward approaches to this problem (e.g., [Shardanand 95]) generally require a quadratic-order matching step somewhere, which makes them scale quite poorly. Often, this matching step must be performed in a single place, worsening the problems of bottlenecks—such centralization is often the result of there being no effective way for peers to find each other, or no effective way for any clump of peers to know whether a clump somewhere else is related to them in any way.

Other solutions have certainly been proposed and adopted. For example, hierarchical organization of the entities, as is done with resource records in the Internet domain name system [Mockapetris 87] or with newsgroup topics in the Usenet [Horton 87], can help to reduce the inherently quadratic problem into a logarithmic one. However, such approaches depend on some inherent organizational principle that is established in advance, which is neither always optimal nor always convenient (for example, consider the number of crossposted Usenet articles, a clear indication that single-inheritance hierarchies are not necessarily a good match to the underlying topic space).

2 A motivating example: Matchmaking

The motivating example for this research is that of a matchmaker, which can be in one of several domains. In this paper, we will investigate the domain of newspaper classified ads, transported to a an electronic context. Individual users (via their agents) submit either "buy" or "sell" ads (akin to messages in a for-sale newsgroup); these ads are then matched with each other.

We assume here that each agent has a particular ad that corresponds to some request of its user. Both buyers and sellers post ads. We use the term "ad" and "agent" somewhat interchangeably when talking about communication; in particular, when we talk about an ad communicating with another ad, we really mean the agent corresponding to that ad. For our purposes, it does not matter whether the agent (and its corresponding ad) stays on a particular workstation and simply opens network connections to other agents, or instead the agent itself (e.g., its thread of control) moves around the network a la Telescript [Telescript 94]. However, for clarity of presentation, we assume that the agent stays put and opens network connections.

From: AAAI Workshop on Information Gathering in Distributed, Heterogenous Environments '95

2.1 Problems with typical approaches

Clearly, every agent on the network should not have to see every ad, nor should there be a single clearinghouse of such ads or of all agents—neither approach scales. Newspapers generally force division into categories when running classifieds, but such categories are often either too wide, too narrow, or off the mark in some other way for their intended readership. Further, by centralizing the distribution of the ads, newspapers introduce artifacts of timing, geographical region, and so forth, which may also be a poor match for particular readers.

Posting and reading ads on the Usenet suffers from similar problems. Usenet posters by definition flood any post to all machines, in the same way that every recipient of a newspaper gets every page of the newpaper's classified ads. Because many topics are simultaneously too narrow along one dimension (causing crossposting to other groups) and too wide along others (causing many dissimilar ads to be grouped into the same newsgroup), potential readers must scan every article in a possibly large number of newsgroups to be sure of finding relevant ads. (If one does not a priori trust the imposed hierarchical clustering to cause advertisers to correctly cluster, then one may be reduced to scanning *all* newsgroups, a 70-meg-per-day proposition. Likewise, if one does not trust one's readers to read appropriate newsgroups, one is tempted to crosspost to increasingly irrelevant newsgroups, as the recent "green card lawyers" incident demonstrates.)

2.2 Some potential solutions

This paper takes the problem of matching classified ads to their intended readers in a somewhat different direction. The fundamental idea is to *dynamically cluster* ads into clumps located in a space. Once a cluster (or clump) is formed, we need only search a relatively small space to find relevant ads, rather than the space of all ads.

Both buyers and sellers post ads. Ads only talk to nearby neighbors in the space. Since the clumps are dynamically created, a predefined hierarchy is avoided. Since communication is strictly local, many scaling effects are likewise reduced. This approach requires that we define a distance metric between any pair of ads, and a method of clustering ads which have small distances between them. We actually have two spaces here: a *concept* space, consisting of how ads would cluster if every ad knew about every other ad, and a *clump* space, consisting of how the ads are actually clustered, given their local knowledge, network connectivity, and so forth. The clump space is similar to but not exactly the same as the concept space.

2.2.1 The concept space

There are several possibilities for establishing a distance metric between ads and hence a concept space.

Simple keyword matching defines a very irregular feature space; "house" and "apartment" are very far apart in a strict keyword match, but are semantically close nonetheless. Despite this shortcoming, keyword matching is quite fast, yields a fairly simple one-dimensional distance between pairs of ads, and is usable with only very small knowledge of what words mean in the domain (e.g., we need to stem words, but not know what they mean). For initial experiments, we are therefore experimenting with SMART [Yanoff 71] to establish a distance metric.

Other alternatives certainly exist, and will be investigated soon. For example, a semantic-net approach can yield less "discontinuous" matching (wherein "house and "apartment" are close in the concept space), at the possible cost of increased computation and a larger requirement that the agent know something about the domain of the ad. In addition, a semantic-net approach offers promise of more interesting distance metrics between ads than simple similarity of keywords, such as via antonymical relationships—for example, we could use WordNet [Miller 93], which specifies a semantic net of common words, with help from part-of-speech tagging [Cutting et al 92], to find, e.g., negations such as "I do *not* want to live in an apartment."

2.2.2 The clump space

We turn now to actually using a distance metric to cause ads to agglutinate in clump space. The aim is to cluster ads on similar topics together, so that buyers need only search through ads which relate to what they wish to buy, and so forth. (A semantic-net approach could also allow us to split such a clump of "buy" and "sell" ads into two subclumps, in which all the buyers are likewise clustered apart from the sellers—we ignore such a refinement here for the moment.)

Consider, for a moment, an individual ad. It "talks" only to neighboring ads. It can, however, pick up and move to a different neighborhood along some dimension, if it finds out from a neighbor that some nearby ad is closer to its meaning. (Again, it does not matter whether such "movement" corresponds to a literal movement of the agent's computation to another workstation, or instead corresponds to opening a network connection and talking to an agent that way.)

In this way, ads migrate around the clump space, always talking only to local neighbors, gradually forming clusters. Each ad can talk to multiple different local neighbors at once, where each such neighbor is a neighbor along a different dimension in concept space. Thus, two ads, one of which says "I'd like to buy a house" and the other of which says "I'm in the market for an apartment" will tend to be close in two important dimensions: both of them are posted by a buyer, and both of them are housing-related. They will tend to form an easily-findable clump that sellers can therefore find, by following a gradient of links in concept space.[1]

Note that the scheme above assumes we have the concept-space distance metric mentioned earlier. It also assumes we have two more abilities:

- A way of finding peers
- A way of establishing a gradient in clump space

1. If there were only a few ads in the entire network, this particular example would not work without a semantic-net distance metric, since a keyword-matcher would not equate "house" and "apartment." However, if there are many ads in the system, we assume that some of them will use both "house" and "apartment" in the same ad, which will tend to cause ads mentioning only one or the other to clump up with ads that mention both. It remains to be seen how effective this strategy will be.

2.2.2.1 Finding peers

For ads to form clumps, they must be able to find other ads with which to clump. Any solution which scales cannot rely on a central registry of all ads or all agents, hence individual ads must have a way of joining an ad-hoc network of other ads.

To enable this, each agent keeps track of the location on the network of previous agents it has encountered (e.g., other agents which have either initiated or responded to communication). It need not keep a complete list of all such agents, but simply a large enough set to make it unlikely that all remembered agents form a clique in which there are no links to any agents outside the clique. (This is to make the probability of disconnected spaces small.) The exact setting of this parameter is unknown, but keeping track of the last 100-500 agent locations should be trivial and probably sufficient.

When an agent is first created, it must guess at contact information to find some other "connected" agent. There are many heuristics for accomplishing this, most of them outside of the scope of this paper. Easy techniques include asking the user for likely machines to contact, trying all locally-connected machines (via a broadcast or some other cheap mechanism) to see if any of them have agents, or consulting an ad-hoc or site-specific registry containing nearby likely addresses (such a registry need not know about all agents, of course, merely a very small number of them). From such contacts, an agent can bootstrap its way into the network by asking for *referrals*, as described below.

2.2.2.2 Establishing a gradient

In general, an ad attempting to find a suitable clump asks some other ad, "Are you close to me in concept space?" using some distance metric. If the answer is yes, the ad joins the clump. If the answer is no, however, the itinerant ad must ask for a *referral* to some other, more compatible ad.

If the referrals received were simply random (e.g., a complete listing of all ads previously encountered, or some random subset of them), forming a clump would resemble random diffusion and would likely be very slow. Instead, we attempt to establish a gradient in concept space that enables quicker formation of clumps.

We do this by having each agent remember, in addition to which agents it has recently encountered, something about their ads as well. For example, we could remember the entire text of the last *n* ads, or some summarization (such as a point in a vector space). When a new agent comes looking for a suitable referral, the agent being asked could find the nearest remembered ad from one of its partners, and redirect the new agent as appropriate.[2]

2. Such behavior is altruistic, in that any given agent has little incentive to remember prior ads, though if there is uniformity in the programming of each agent, the community as a whole will benefit, and so will each individual agent. Solutions which depend less on altruism are being investigated; however, they may not be strictly necessary. If the overhead of such referrals is relatively small, and their utility sufficiently obvious to the agents' user community, then few users will be motivated to eliminate them, even in the absence of peer pressure or more draconian, tit-for-tat responses.

3 Conclusions

By distributing the computation into a large number of small pieces, all of which are relatively local and do not require knowledge of the global state, this research has aimed to make large, networked information retrieval applications more tractable. It presents a typical example of an application that can benefit from distribution, and talks about ways to accomplish the distribution of the computation. There is still a long way to go, however, and feedback on these approaches in encouraged. Some of the more important unsolved pieces include:

- The details of the distribution. Which applications are best done with keyword-based matching, which with semantic-net-based matching, and which with completely different sorts of matching? What does this imply about the dimensionality of the resulting spaces? How important is visualization of the result?
- Issues of user privacy. Notice that a buyer must essentially advertise her interests to a large number of a priori unknown agents. If the item being sought is supposed to be secret, this is a problem. Parallel research is being conducted into ways of using cryptographic protocols (such as zero-knowledge proofs [Schneier 94] and various others) to help contain the resulting privacy exposures—the test domain for this research employs an application that attempts to match users by their interests. Unfortunately, a complete description of these approaches is well outside the scope of this paper.
- How to distribute such an application across the Internet in a way that makes it easiest for new users to install and use. Ease of installation is critical to quickly establishing a large user base, and such a large user base is essential to give the community of agents enough size to start doing useful work. Such distribution also interacts with the point above about privacy; mechanisms for ensuring that the software received is trustworthy are important.

All of these areas are currently under active development.

4 References

[Cutting et al 92]
"A Practical Part-of-Speech Tagger", Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun, Xerox Technical Report, 1992.

[Foner 93]
"What's an Agent, Anyway? A Sociological Case Study", Leonard Foner, *Agents Memo 93-01*, MIT Media Lab, Cambridge, MA, 1993. (ftp://media.mit.edu/pub/Foner/Papers/Agents--Julia.ps)

[Horton 87]
"Standard for interchange of USENET messages", Mark Horton and R Adams, Internet RFC1036. (ftp://ds.internic.net/rfc/rfc1036.txt.)

[Huberman 88]
The Ecology of Computation, B. A. Huberman, editor, Elsevier Science Publishers B.V., 1988.

[Lashkari et al 94]
"Collaborative Interface Agents", Yezdi Lashkari, Max Me-

tral, and Pattie Maes, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, MIT Press, Cambridge, MA, 1994.

[Maes 94]
"Agents that Reduce Work and Information Overload", Pattie Maes, *Communications of the ACM*, July 1994, Vol 37 #7.

[Mauldin 94]
"ChatterBots, TinyMuds, and the Turing Test: Entering the Loebner Prize Competition", Michael Mauldin, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, MIT Press, Cambridge, MA, 1994. (http://fuzine.mc.cs.cmu.edu/mlm/aaai94.html)

[Miller 93]
"Introduction to WordNet: An On-line Lexical Database", George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller, Princeton University Technical Report, 1993.

[Mockapetris 87]
"Domain names - implementation and specification", Paul Mockapetris, Internet RFC1034. (ftp://ds.internic.net/rfc/rfc1034.)

[Shardanand 95]
"Social Information Filtering: Algorithms for Automating 'Word of Mouth'", Upendra Shardanand and Pattie Maes, submitted to CHI-95, Denver, CO, May 1995.

[Schneier 94]
Applied Cryptography, Bruce Schneier, John Wiley & Sons, 1994.

[Telescript 94]
Telescript Technology: The Foundation for the Electronic Marketplace, General Magic, Inc, 1994.

[Zumoff 71]
"Users Manual for the SMART Information Retrieval System", Joel Zumoff, Cornell Technical Report 71-95.