Building Large Knowledge Bases by Mass Collaboration

Matthew Richardson
mattr@cs.washington.edu

Pedro Domingos
pedrod@cs.washington.edu

Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350

ABSTRACT

Acquiring knowledge has long been the major bottleneck preventing the rapid spread of AI systems. Manual approaches are slow and costly. Machine-learning approaches have limitations in the depth and breadth of knowledge they can acquire. The spread of the Internet has made possible a third solution: building knowledge bases by mass collaboration, with thousands of volunteers contributing simultaneously. While this approach promises large improvements in the speed and cost of knowledge base development, it can only succeed if the problem of ensuring the quality, relevance and consistency of the knowledge is addressed, if contributors are properly motivated, and if the underlying algorithms scale. In this paper we propose an architecture that meets all these desiderata. It uses first-order probabilistic reasoning techniques to combine potentially inconsistent knowledge sources of varying quality, and it uses machine-learning techniques to estimate the quality of knowledge. We evaluate the approach using a series of synthetic knowledge bases and a pilot study in the domain of printer troubleshooting.[1]

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge acquisition, Parameter learning*

General Terms

Algorithms

Keywords

knowledge acquisition, knowledge engineering, AI architectures, machine learning, collective knowledge bases

[1]This research was supported by an IBM Ph.D. Fellowship to the first author, and by ONR grant N00014-02-1-0408.

K-CAP'03, October 23–25, 2003, Sanibel Island, Florida, USA.

1. INTRODUCTION

Truly intelligent action requires large quantities of knowledge. Acquiring this knowledge has long been the major bottleneck preventing the rapid spread of AI systems. Two main approaches to this problem exist today. In the manual approach, exemplified by the Cyc project [10], human beings enter rules by hand into a knowledge base. This is a slow and costly process. Although the original goal was to complete Cyc in ten years, it has now been under development for seventeen.[2] In the machine learning approach, exemplified by programs such as C4.5 [16], rules are automatically induced from data. Although this approach has been extremely successful in many domains, it has not led to the development of the large, diverse knowledge bases necessary for truly intelligent behavior. Typical learning programs contain only very weak assumptions about the world, and as a result the rules they learn are relatively shallow – they refer only to correlations between observable variables, and the same program applied to two different data sets from the same domain will typically produce different rules. Recognizing this problem, researchers have invested substantial effort into developing learning programs that can incorporate pre-existing knowledge, in effect combining the manual and automatic approaches (e.g., Pazzani & Kibler (1992)). However, these programs have not been widely adopted, largely due to the difficulty and expense of capturing knowledge – the same bottleneck that has plagued purely manual solutions.

The rise of the Internet has made possible a third approach to the knowledge acquisition problem, one with the potential to greatly speed the spread of AI. The open-source software movement, enabled by the Internet, has shown that it is possible to develop very high quality software by accumulating contributions from thousands of volunteers [17]. This surprising outcome, exemplified by the success of the Linux operating system, is relevant to the construction of large-scale knowledge

[2]Our description of Cyc in this paper is based on the publications about it, made primarily in its founding years. Cyc has developed since, but these developments are not publicly available.

bases. If the work of a large number of volunteers can be properly coordinated, knowledge bases as large as Cyc or larger can be built in a much shorter period of time, at a fraction of the cost. Conversely, over a period of a decade a knowledge base dwarfing any built so far can be inexpensively developed.

However, while building knowledge bases by mass collaboration avoids some of the problems of the traditional approach, it greatly exacerbates others:

Quality. Ensuring the quality of knowledge contributed by many different sources, when little is known about most of them, is likely to be very difficult. We thus need mechanisms for automatically gauging the quality of contributions, and for making the best possible use of knowledge of widely variable quality. This includes taking advantage of redundant or highly overlapping contributions, when they are available.

Consistency. As the knowledge base grows in size, maintaining consistency between knowledge entered by different contributors, or even by the same contributor at different times, becomes increasingly difficult. In a traditional logic-based system, a single inconsistency is in principle enough to make all inference collapse. This has been a major issue in the development of Cyc, and will be a much more serious problem in a knowledge base built by many loosely-coordinated volunteers.

Relevance. The initial Cyc philosophy of simply entering knowledge regardless of its possible uses is arguably one of the main reasons it has failed to have a significant impact so far. In a distributed setting, ensuring that the knowledge contributed is relevant – and that volunteers' effort is productive – is an even more significant problem.

Scalability. To achieve its full potential, a collective knowledge base must be able to assimilate the work of an arbitrarily large number of contributors, without the need for centralized human screening, coordination, or control becoming a bottleneck. Likewise, the computational learning and reasoning processes carried out within the knowledge base should scale at worst log-linearly in the number of contributions. This implies making expressiveness/tractability trade-offs, approximations, etc.

Motivation of contributors. To succeed, collective knowledge bases will depend on the unpaid work of a large number of volunteers. Motivating these volunteers is therefore essential. Following the example of open-source software, collective knowledge bases should allow user-developers to enter knowledge that is first of all relevant to solving their own problems. Following the example of knowledge-sharing Web sites [4], collective knowledge bases should incorporate a fair mechanism for giving volunteers credit for their contributions.

This paper proposes an architecture for collective knowledge base development that addresses the five issues above. The next section describes the architecture. The following section describes the preliminary experimental evaluation of the architecture we have carried out. We conclude with a discussion of related and future work.

2. AN ARCHITECTURE FOR COLLECTIVE KNOWLEDGE BASES

Figure 1 shows an input-output view of the architecture. A collective knowledge base is a continuously-operating system that receives three streams of information:

Rules and facts from contributors. Rules and facts are expressed in the Horn clause subset of first-order logic. A first-order representation is used in the great majority of knowledge-based systems, including Cyc, and is clearly necessary to efficiently capture a broad range of real-world knowledge. Horn clauses are used in many expert system shells, and form the basis of the Prolog programming language. They are an effective trade-off between expressiveness and tractability, and using them takes advantage of extensive previous research on making Horn-clause inference efficient. Horn clauses also have the key feature of high modularity: a new rule can be input without knowing what other rules are already in the knowledge base. Note that, although our current system requires entering knowledge directly in Horn-clause form, this need not be the case in general. Allowing knowledge entry via menu-driven interfaces, ontology browsers, and restricted forms of natural language (e.g., using a particular syntax, or within specific domains) should greatly increase the number of individuals that are able to contribute. The extensive previous research on tools for knowledge base development (e.g., McGuinness *et al.* (2000)) should be useful here.

Queries and evidence from users. Following conventional usage, a query is a predicate with open variables (input directly, or obtained by translation from a user interface). Users also supply evidence relevant to the queries in the form of a set of facts (ground instances of predicates). These facts may be manually input by the user, or automatically captured from the outside system the query refers to. (For example, if the query is a request to diagnose a malfunctioning artifact such as a car or a computer, information on the state and configuration of the artifact may be captured directly from it.) As in many knowledge-sharing sites, queries can also have a "utility value" attached, reflecting how much the user is willing to "pay" (in some real or virtual unit) for the answer.

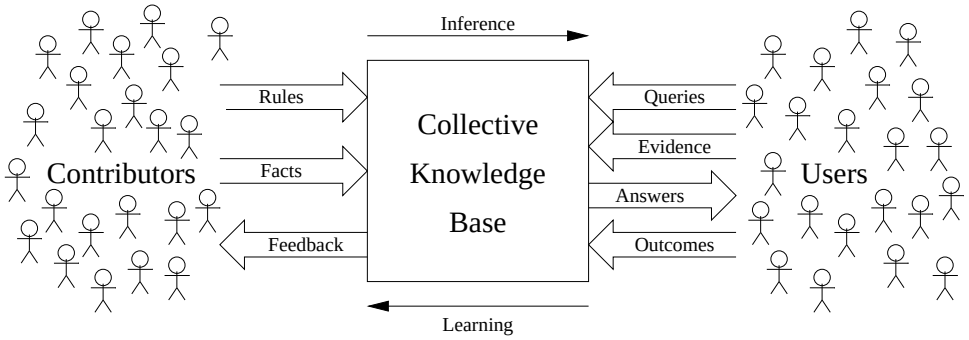

Figure 1: Input-output view of a collective knowledge base.

Feedback on the system's replies, from users. Given the answer or answers to a query, the user takes actions, observes their outcomes, and reports the results to the knowledge base. For example, if the query is "Where on the Web can I find X?" and the answer is a URL, the user can go to that URL and report whether or not X was found there. In a fault diagnosis problem, the user attempts to fix the fault where diagnosed, and reports the result. The outcome can also be in the form of a utility rating for the answer. This rating can be objective (e.g., time saved, number of "hits" in some task) or subjective (e.g., user's satisfaction on a five point scale).

In return, the collective knowledge base produces two streams of information:

Answers to queries. Answers to a query consist of instantiations of the open variables for which the query predicate holds true. They are sorted by probability of correctness, in the same way that search engines sort documents by relevance. Probabilities of correctness are computed as described below.

Credit to contributors. Contributors receive from the knowledge base feedback on the quality of their entries, in the form of accumulated (positive or negative) credit for their use in answering queries. The credit assignment computation is described below.

The collective knowledge base is thus involved in two continuous loops of interaction, one with contributors, and one with users (these two populations need not be disjoint). Contributors and users, as a result, interact via the knowledge base. This interaction is in general not one-to-one, but many-to-many: entries from many different contributors may be combined by inference to yield the answer(s) to a query, and the feedback from a query's outcome will in return be propagated to many different contributors. Conversely, a single contribution may be used in answering many different queries, and receive feedback from all of them.

The contributed rules are likely to be noisy, so associated with each are probabilistic parameters specifying quality and/or accuracy. The result is a probabilistic first-order representation. There are a variety of approaches for reasoning in such a representation; we chose to base our algorithms on *knowledge-based model construction* [24] (KBMC) (see Section 3), the most directly applicable method. In KBMC, queries are answered by first compiling the knowledge base (and associated rule weights) into a Bayesian network [15], then applying standard Bayesian network inference algorithms. Similarly, feedback is incorporated by using standard BN methods such as expectation-maximization [3] (EM) to find the rule weights which maximize the likelihood of the feedback. KBMC allows for efficient inference and learning (in space, time and the number of samples needed to accurately learn the parameters). We will provide more algorithmic details in Section 3.

A key feature of this architecture is that collective knowledge bases are built by an intimate combination of human work and machine learning, and the division of labor between them reflects their respective strengths and weaknesses. Human beings are best at making simplified, qualitative statements about what is true in the world, and using their judgment to gauge the quality of the end results produced by computers. They are notoriously poor at estimating probabilities or reasoning with them [23]. Machines are best at handling large volumes of data, estimating probabilities, and computing with them. Another key feature is that the knowledge base is not developed in open-loop mode, with the knowledge enterers receiving no real-world feedback on the quality and correctness of their contributions. Rather, the evolving knowledge base is subjected to constant reality checks in the form of queries and their outcomes, and the resulting knowledge is therefore much more likely to be both relevant and correct.

In current knowledge-sharing sites and knowledge management systems, questions are answered from an indexed repository of past answers, or routed to the appropriate experts. Thus the only questions that can be

answered automatically are those that have been asked and answered in the past. In contrast, the architecture we propose here allows chaining between rules and facts provided by different experts, and thus automatically answering potentially a very large number of questions that were not answered before. This can greatly increase the utility of the system, decrease the cost of answering questions, and increase the rewards of contributing knowledge.

The architecture answers the problems posed in the introduction:

Quality. By employing feedback and machine learning, we are able to determine which rules are of high quality, and which are not. Further, since we are tracking the utility of knowledge provided by users, they are more inclined to provide good rules.

Consistency. By using a probabilistic framework, we are able to handle inconsistent knowledge.

Relevance. Since the knowledge base is being built by users, for users, we expect the rules to be on topics that the users find relevant and interesting. The credit assignment process rewards those contributors whose rules are used (and produced a correct answer), which provides incentive to create rules that are relevant to users' needs.

Scalability. For both training and query-answering, the most expensive portion of the computation is the probabilistic inference on the Bayesian network. However, this computation depends only on the size of the network, not of the entire knowledge base. The Bayesian network is constructed out of only the relevant knowledge, which we expect (and confirm empirically in the experimental section) will lead to relatively small networks even for very large knowledge bases.

Motivation of contributors. By tracking the utility of rules and assigning credit to those which are used to answer queries, we provide the means for motivating contributors (e.g. listing the top-10, paying in some real or virtual currency, etc.)

In the next section, we present the CKB algorithms in more detail.

3. ALGORITHM

The internal workings of a collective knowledge base generally follow the knowledge-based model construction (KBMC) framework of Ngo and Haddawy [13]. This has been used by, among others, Koller and Pfeffer [9], and Kersting [8]. KBMC takes a Horn clause knowledge base and a query as inputs, and produces a Bayesian network relevant to answering the query. The advantage of KBMC over ordinary logical inference is that it allows for "noisy" knowledge (i.e., it takes into account that facts and rules may not be believed with certainty).

3.1 Review of KBMC

We begin by introducing KBMC for the common case of using noisy-or [15] to combine rules (Horn clauses) with the same consequent. Noisy-or is a probabilistic generalization of the logical OR function; it makes the assumption that the probability that one of the "causes" fails to produce the effect is independent of the success or failure of the other causes. In this case, with each clause is associated a parameter that specifies the probability that the consequent holds given that the antecedents hold. Table 1 gives an example set of Horn clauses and their associated parameters. For example, the probability that a person, say "mary", exercises is 0.8 if she owns a gym membership. Because the clause is defined for all X, this probability is the same for all people in the model. This parameter sharing facilitates both a compact representation, and learning. KBMC allows Horn clauses with relations of any arity.

To answer a query, KBMC extracts from the knowledge base a Bayesian network containing the relevant knowledge. Each grounded predicate that is relevant to the query appears in the Bayesian network as a node. Relevant predicates are found by using standard Prolog backward chaining techniques, except that rather than stopping when one proof tree is found, KBMC conceptually finds every possible proof tree. The multiple trees together form a proof DAG (directed acyclic graph) where each node of the DAG is a grounded predicate. For example, Figure 2 shows the Bayesian network that would result from the query "*healthy(mary)*?" given "*eats_well(mary)*" and "*gym_member(mary)*". Once the query has been converted into a Bayesian network, any standard BN inference technique may be used to answer the query.

When there are multiple relevant clauses that have the same grounded consequent, KBMC employs a *combination function* to compute the consequent's probability. For example, consider the second clause in Table 1: *eats_well(X)* $\leftarrow$ *eats(X,Y)*, *healthy_food(Y)*. If *eats(mary,carrots)* and *eats(mary,apples)*, both of which are *healthy_food(.)*, then what is the probability that *eats_well(mary)*? To answer this, an additional set of nodes are introduced to the Bayesian network, one for each clause (e.g, E1 and E2 in Figure 3). The node corresponding to a clause represents the proposition "All of the clause's antecedents are true," and is thus a deterministic AND function of those antecedents. For each (grounded) predicate, the probability that the predicate holds is a function of the "clause" nodes (e.g., E1) that have that predicate as the consequent. For example, the predicate can be a noisy-or of the clauses that have it as the consequent. In general, the combination func-

0.9	*healthy(X)*	←	*eats_well(X), exercises(X)*
0.7	*eats_well(X)*	←	*eats(X,Y), healthy_food(Y)*
0.8	*exercises(X)*	←	*gym_member(X)*

Table 1: Sample Horn clauses defining part of a knowledge base. The number specifies the probability of the consequent when all of the antecedents are true .

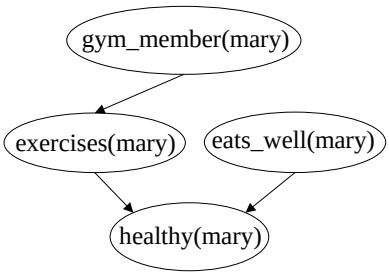

Figure 2: Example Bayesian network formed for a query on the knowledge base shown in table 1.

tion can be any model of the conditional distribution of a Boolean variable given other Boolean variables, and can be different for different predicates. Our implemented system supports three combination functions, all of which require one parameter per clause: noisy-or [15], linear pool [5][7], and logistic regression [1].

Notice that the size of the Bayesian network produced by KBMC in response to a query is only proportional to the number of rules and facts relevant to the query, not the size of the whole knowledge base; this is crucial to the scalability of the approach.

For some combination functions (such as a leaky noisy-or), a fact may have non-zero probability even if none of

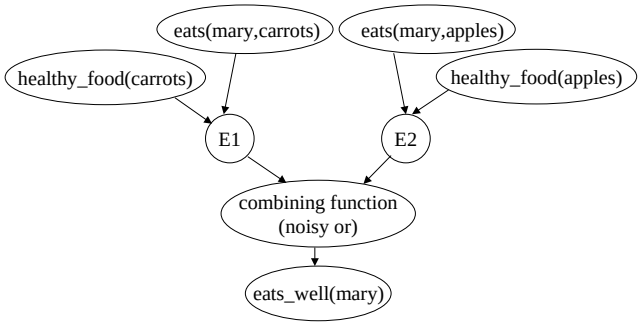

Figure 3: When there are multiple sources of evidence, they are combined using a node which performs a function such as noisy-or. Above is an example Bayesian network for the query "*eats_well(mary)*?".

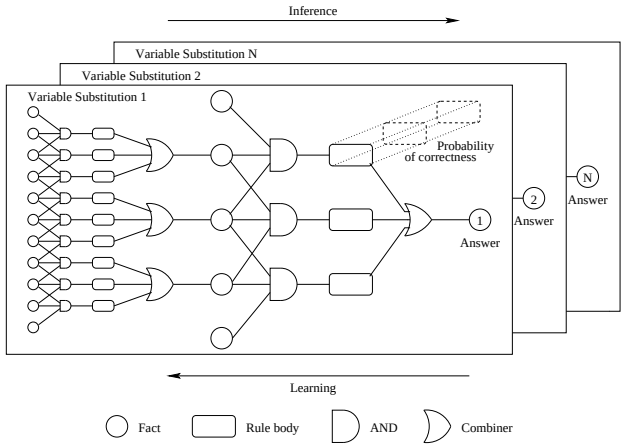

Figure 4: Internal functioning of the knowledge base.

the rules for which it is a consequent apply. When using such functions, the results of inference are not complete: probabilities are only computed for facts that have at least one applicable rule. Generally, non-derivable answers vastly outnumber the derivable ones, and ignoring them greatly reduces the computational complexity of query answering. Since these answers would presumably have low probability, and users will generally only want to look at the top few most probable answers, this should have very little effect on the end result.

3.2 Credit and Rule Weights

For each answer to a query there is a proof DAG (directed acyclic graph). Each subtree of the DAG rooted in the answer and with facts (evidence or contributed) as leaves corresponds to an alternate derivation of the answer. Given feedback on the correctness of the answer (entered by the user), the utility of the corresponding query is propagated throughout the proof DAG. Credit is divided equally among the proof trees, and within each tree among the rules and facts that were used in it. If a rule or fact was used in multiple proof trees, it accumulates credit from all of them. Over many queries, each rule or fact accumulates credit proportional to its overall utility, and each contributor accumulates credit from all the queries his/her knowledge has helped to answer. If a rule or fact tends to lead to incorrect answers, the consequent cost (or negative utility) will be propagated to it; this will encourage contributors to enter only knowledge they believe to be of high quality.

Feedback from users, in the form of confirmation of the system's answers, or the correct answers after they become known, is also used to learn the weights of rules and facts in the knowledge base. This is done using the EM algorithm [3, 9], with each example being the evidence and correct answer for a given query, and the missing information being the remaining facts. By de-

Table 2: The CKB algorithms.

Inputs:

E Set of grounded predicates (evidence)
KB Set of horn clauses with associated probabilities (knowledge base)
F Feedback set $\{f_1, f_2, \dots\}$, with $f_i = (\mathbf{E}, q, a)$: an evidence set, a query, and the correct answer.

AnswerQuery(KB, E, q**)**
N = GenerateNetwork(**KB**, **E**, q)
return $P(q|\mathbf{E})$ = RunBayesNetInference(N)

Train(KB, F)
do while **KB** probabilities have not converged:
for each f_i in **F**:
N = GenerateNetwork(**KB**, $f_i(\mathbf{E})$, $f_i(q)$)
SetEvidence(N, $f_i(q)$)
RunBayesNetInference(N)
AccumulateProbabilities(**KB**, N)
UpdateRuleProbabilities(**KB**)

GenerateNetwork(KB, E, q**)**
G = GenerateProofDAG(**KB**, **E**, q)
N = ConvertToBayesianNetwork(G)
return N = SetEvidence(N, **E**)

fault, each first-order rule has only one weight, obtained by tying (averaging) the weights of its ground instances. The weights of all the combination nodes corresponding to that rule in the propositionalized network are set to this value.

The CKB algorithms are summarized in Table 2. GenerateProofDAG() uses Prolog to find all possible proof trees for the given query, and then merges these proof trees into a proof DAG. ConvertToBayesianNetwork() constructs a Bayesian network of AND and combination nodes as described earlier. UpdateRuleProbabilities() updates the probabilities in the knowledge base so as to maximize the likelihood of the feedback.

4. EXPERIMENTAL EVALUATION

As a preliminary evaluation, we performed two sets of experiments using our implementation of a collective knowledge base. In the first, we generated synthetic first-order rules and facts. In the second, we built a printer troubleshooting knowledge base using contributions from real users. Logistic regression was used as the evidence combination function in both experiments.

4.1 Synthetic Knowledge Bases

To our knowledge, there is currently no publicly-available knowledge base of the scope that would be desirable for demonstrating the advantages of our system. We thus opted to simulate the contributions of many different volunteers to a collective knowledge base, in the form of first-order rules and facts.

We based our knowledge generation process on the assumption that a contributor is an expert in a particular topic. We thus first generated a random taxonomy of topics, each of which contained some number of predicates, variable types, and ground instances. An expert is likely to know not just the concepts in a given topic, but also the general concepts of more specialized subtopics. Each topic was thus divided into general and specific predicates. An expert could form rules for a topic[3] using as antecedents any of the topic's predicates, or any of the general predicates of the immediate sub-topics. We generated a random knowledge base of rules in this way.

We simulated an expert by choosing a random topic and sampling the knowledge base for rules with consequents from nodes in the vicinity of that topic in the hierarchy. The probability that an expert submitted a rule in a given topic decreased exponentially with the distance (number of hops) between that topic and the expert's one in the taxonomy. We randomly added and removed antecedents from an expert's rules to simulate noisy or incomplete knowledge.

Positive training and testing examples were generated by randomly choosing a consequent and backward-chaining through rules in the knowledge base to find evidence that supported them. A positive example was turned into a negative one by removing a single evidence item, which resulted in "near-miss" examples that are easy to mistake as positive. Note that some samples thus required only knowledge contained within one topic, while others required chains of inference that spanned topics and subtopics, which we believe is often the case in the real world.

We modeled the accumulation of knowledge as proportional to the number of contributors to the system, with 25 rules and 50 feedback instances per contributor. The ontology had 25 nodes, and the "true" knowledge base had 50 rules per category. These and other parameters (provided in an online appendix) were constant throughout the experiments, and set before seeing any test results. We tested with 500 queries. The results are shown in Figure 5, where "Averaged Experts" is the performance obtained by estimating the probability of an answer as the average of the probabilities predicted by the relevant experts (i.e., those that were able to answer the question). "Trained CKB" and "Untrained CKB" refer to the performance obtained by using the architecture presented in this paper, with or without using the feedback to train. The performance measure ("Accuracy") is the fraction of queries that were answered correctly (with unanswered queries counting as failures). The advantage of the collective knowledge base increases rapidly with the number of contribu-

[3]Rules for a topic are defined as rules whose consequent is a predicate belonging to that topic.

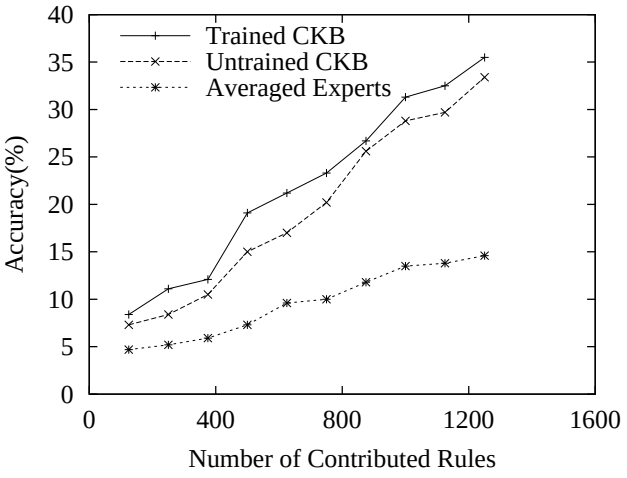

Figure 5: Results on synthetic knowledge bases.

tors. This is attributable to the increasing number of connections that can be made between different contributions as the knowledge base becomes increasingly densely populated. We also see that applying machine learning to estimate the quality of knowledge further improves performance.

Although not optimized for speed, our current system is fairly efficient. The time required to extract the Bayesian network for a query from the knowledge base was dominated by the time to run probabilistic inference on the network. The size of the extracted Bayesian network grew sub-linearly in the size of the knowledge base, from an average of 11 nodes for a collection of 10 experts to 24 nodes for a collection of 50 experts. The average time spent on probabilistic inference for a query asked of the pool of 50 experts was 400 milliseconds; the time required to run EM was approximately proportional to the product of this and the number of training examples.

4.2 Printer Troubleshooting

A significant portion of Usenet newsgroups, FAQs, and discussion forums is devoted to the task of helping others diagnose their computer problems. This suggests that an automated knowledge base for this domain would be in high demand. It is also potentially well suited to development using our collective knowledge base architecture, due to the availability of a large pool of willing experts with some degree of formal knowledge, the availability of objective outcomes for feedback purposes, the composable nature of the knowledge, the fact that evidence can potentially be captured automatically from the machines being diagnosed, etc. As a first step in this direction, we have carried out a pilot study demonstrating that knowledge obtained from real-world experts in this domain can be merged into a system that is more accurate than the experts in isolation.

We used the Microsoft printer troubleshooting Bayesian network as a model of the domain.[4] (In other words, examples generated from this network were used to simulate examples generated by the real world.) The network consists of 76 Boolean variables. Seventy of these are informational, such as "print spooling is enabled" and "fonts are installed correctly", and six are problem-related, such as "printing takes too long". Many of the variables are labeled as fixable and/or observable with associated costs. We considered any variable whose cost of observation was less than one to be *evidence*, and any proposition that was fixable but not evidence to be a *cause*, which resulted in seventeen evidence variables and twenty-three causes.

The system attempts to identify the most likely cause of a problem, given the evidence and problem nodes. To generate plausible problems a user may ask the system about, we generated random samples from the network and accepted only those where exactly one cause was at fault and at least one of the problem-related propositions was true. The system was then presented with the resulting evidence and problem nodes and asked to diagnose which proposition was the cause of the problem. As in the first-order domain, the system may elect not to answer a question. We report two measures of success. One is the fraction of queries whose cause is properly diagnosed (with unanswered queries counting as failures). The other is the average rank of the correct diagnosis in the list of probable causes returned (with the most probable cause having rank one). This corresponds to the number of actions a user needs to perform before the printer is functioning again.

We gave the definitions of the seventy-six variables to four volunteers, who were each asked to write rules describing the printer domain to the best of their ability in a limited amount of time. All four were computer users who have had experience printing but did not have any particular expertise or training on the subject. Table 3 shows for each volunteer the time spent contributing knowledge, the number of rules contributed, and the performance before and after learning rule weights. Two hundred examples were used for training. Random guessing would have achieved an accuracy of 4.5%.

Table 3 also shows the results of combining the experts. The row labeled "Average" is the result of averaging predictions as described before.[5] The "CKB" row shows the added advantage of the collective knowledge base: it achieves higher accuracy than a simple combination of the individual volunteers, both when the individual volunteers' rule coefficients have been trained and when

[4]Available at http://www.cs.huji.ac.il/~galel/Repository/-Datasets/win95pts/.

[5]This can be higher than the average accuracy of the experts, if different experts answer different questions, because an unanswered query is counted as answered incorrectly.

Table 3: Printer troubleshooting results. "Volunteer i" is the system using the ith volunteer's rules. "CKB" is the collective knowledge base. The accuracy of random guessing is 4.5%.

System	Time (mins)	Num. Rules	Accuracy (%)		Rank	
			Untrained	Trained	Untrained	Trained
Volunteer 1	120	79	11.1	15.8	11.9	6.7
Volunteer 2	30	32	2.6	4.5	9.4	8.6
Volunteer 3	120	40	2.7	10.3	13.6	10.3
Volunteer 4	60	34	3.9	6.3	13.0	12.0
Average	–	–	2.2	17.6	13.3	6.7
CKB	–	–	4.6	34.6	12.7	5.7

they have not. Thus we observe once again that the collective knowledge base is able to benefit from chaining between the rules of different volunteers.

5. RELATED WORK

Open Mind [22] (www.openmind.org) and MindPixel (www.mindpixel.com) are two recent projects that seek to build collective knowledge bases. However, neither of them addresses the issues of quality, consistency, relevance, scalability and motivation that are critical to the success of such an enterprise. The MindPixel site asks contributors to input natural language statements and states that they will be used to train a neural network, but it is not clear how this will be done, or how the results will be used. Open Mind appears to be mainly an effort to gather training sets for learning algorithms (e.g., for handwriting and speech recognition). Its "common sense" component [21] is similar to MindPixel. Cycorp (www.cyc.com) has recently announced its intention to allow contributions to Cyc from the public. However, its model is to have contributions screened by Cyc employees, which makes these a bottleneck preventing truly large-scale collaboration. There is also no mechanism for motivating contributors or ensuring the relevance of contributions. Another key difference between Cyc and our approach is that Cyc is an attempt to solve the extremely difficult problem of formally representing all common sense knowledge, while our goal is to build knowledge bases for well-defined, concrete domains where it should be possible to enter much useful knowledge using relatively simple representations.

The Semantic Web is a concept that has received increasing attention in recent times [2]. Its goal can be summarized as making machine-readable information available on the Web, so as to greatly broaden the spectrum of information-gathering and inference tasks that computers can carry out unaided. The Semantic Web can be viewed as complementary to the architecture described here, in that each can benefit from the other. The Semantic Web can provide much of the infrastructure needed for collective knowledge bases (e.g., standard formats for knowledge). In turn, the mechanisms described in this paper can be used to guide and optimize the development of the Semantic Web. Similar remarks apply to other ongoing efforts to support large-scale, distributed knowledge base development (e.g., the Chimaera project [11]).

Collaborative filtering systems [18] and knowledge-sharing sites [4] can be viewed as primitive forms of collective knowledge base. Their success is an indication of the promise of mass collaboration.

The representation we use is a form of probabilistic logic program [13]. Other recently-proposed probabilistic first-order formalisms include stochastic logic programs [12] and probabilistic relational models [6]. Stochastic logic programs are a generalization of probabilistic context-free grammars, and assume that for a given consequent only one rule can fire at a time. They are thus not applicable when multiple rules can function simultaneously as sources of evidence for their common consequent. Probabilistic relational models lack the modularity required for construction by many loosely-coordinated individuals. In the future we plan to explore ways of adapting these approaches for our purposes, and to compare them with probabilistic logic programs.

6. CONCLUSION

Knowledge acquisition is the key bottleneck preventing the wider spread of AI systems. Both current approaches to it — manual and automatic — have limitations that are hard to overcome. The Internet makes possible a new alternative: building knowledge bases by mass collaboration. While this approach can greatly reduce the time and cost of developing very large knowledge bases, it raises problems of quality, consistency, relevance, scalability and motivation. This paper proposes an architecture that addresses each of these problems. Experiments with large synthetic knowledge bases and a pilot study in the printer diagnosis domain show its promise.

Current and future research directions include: developing first-order probabilistic methods specifically for collective knowledge engineering, focusing particularly

on the restrictions on expressiveness needed for scalability; allowing users to provide different forms of knowledge (in [20], we merge statements from experts on the structure of a Bayesian network); studying different alternatives for credit assignment (e.g. using a web of trust [19]), leveraging results from the multi-agent systems literature; developing mechanisms for guiding contributors to where new knowledge would be most useful, using value-of-information computations; detecting and overcoming malicious users; using machine learning techniques to automatically propose to contributors refinements of their entries; developing methods for automatically translating between the ontologies used by different subcommunities of contributors; and deploying a pilot Web site for collective knowledge base construction, open to contributions from all sources, focusing initially on the domain of computer troubleshooting.

7. REFERENCES

[1] A. Agresti. *Categorical Data Analysis*. Wiley, New York, NY, 1990.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[4] M. Frauenfelder. Revenge of the know-it-alls: Inside the Web's free-advice revolution. *Wired*, 8(7):144–158, 2000.

[5] S. French. Group consensus probability distributions: A critical survey. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, and A. F. M. Smith, editors, *Bayesian Statistics 2*, pages 183–202. Elsevier, Amsterdam, Netherlands, 1985.

[6] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1300–1307, Stockholm, Sweden, 1999. Morgan Kaufmann.

[7] C. Genest and J. V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1:114–148, 1986.

[8] K. Kersting. *Bayesian Logic Programs*. PhD thesis, University of Freiburg, Freiburg, Germany, 2000.

[9] D. Koller and A. Pfeffer. Learning probabilities for noisy first-order rules. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 1316–1321, Nagoya, Japan, 1997. Morgan Kaufmann.

[10] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, Reading, MA, 1990.

[11] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, Breckenridge, CO, 2000. Morgan Kaufmann.

[12] S. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, Amsterdam, The Netherlands, 1996.

[13] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.

[14] M. Pazzani and D. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9:57–94, 1992.

[15] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[16] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[17] E. S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, Sebastopol, CA, 1999.

[18] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, New York, NY, 1994. ACM Press.

[19] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the Semantic Web. In *Proceedings of the Second International Semantic Web Conference*, Sanibel Island, FL, 2003.

[20] M. Richardson and P. Domingos. Learning with knowledge from multiple experts. In *Proceedings of the Twentieth International Conference on Machine Learning*, Washington, DC, 2003. Morgan Kaufmann.

[21] P. Singh. The public acquisition of commonsense knowledge. In *Proceedings of the AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*, Palo Alto, CA, 2002. AAAI Press.

[22] D. G. Stork. Using open data collection for intelligent software. *IEEE Computer*, 33(10):104–106, 2000.

[23] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185:1124–1131, 1974.

[24] M. P. Wellman, J. S. Breese, and R. P. Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, 7(1):35–53, 1992.