The Public Acquisition of Commonsense Knowledge

Push Singh

MIT Media Lab
20 Ames St.
Cambridge, MA 02139 USA

push@mit.edu

Abstract

The Open Mind Common Sense project is an attempt to construct a database of commonsense knowledge through the collaboration of a distributed community of thousands of non-expert netizens. We give an overview of the project, describe our knowledge acquisition and representation strategy of using natural language rather than formal logic, and demonstrate this strategy with a search engine application that employs simple commonsense reasoning to reformulate problem queries into more effective solution queries.

Introduction

Workers in artificial intelligence have long sought to teach computers enough about our world so that they could reason about it like we do, to give them the capacity for "common sense". However, the scale of the problem has been discouraging, for people seem to need a tremendous amount of knowledge of a very diverse variety to understand even the simplest children's story. As a result there have been few efforts to try to encode a broad range of human commonsense knowledge.

We believe that today this problem of scale can be addressed in a new way. The critical observation is that every ordinary person has common sense of the kind we want to give our machines. Given the advent of the World Wide Web, artificial intelligence projects now have access to the minds of millions. If we can find good ways to extract common-sense from people by prompting them, asking them questions, presenting them with lines of reasoning to confirm or repair, and so on, we may be able to accumulate many of the knowledge structures needed to give our machines the capacity for commonsense reasoning.

The *Open Mind Initiative* was created to support such large-scale collaborative efforts to build components of intelligent systems (Stork 1999). As part of this effort, we built a Web site called *Open Mind Common Sense*[1] to make it easy and fun for non-expert netizens to collaborate to construct a database of commonsense knowledge. Open Mind Commonsense is in the vein of projects like The Open Directory Project, The Mars Crater Mapping project, and similar "distributed human projects" where difficult problems are solved by distributing the work across thousands of volunteers across the Internet. Our purpose was to explore whether the commonsense knowledge acquisition problem could be cast in a way that non-experts could participate in supplying data, rather than having a small team doing careful knowledge engineering, as in Doug Lenat's well-known Cyc project (Lenat 1995).

This paper reports on our results so far. We give an overview of the project, describe our knowledge acquisition and representation strategy of using natural language rather than formal logic, and demonstrate this strategy with a search engine application that employs simple commonsense reasoning to reformulate problem queries into more effective solution queries.

The Diversity of Commonsense Knowledge

There is much disagreement over what kinds of knowledge are needed to build systems capable of commonsense reasoning. One popular view is that we need to encode general axiomatic formulations of different facets of the commonsense world (Hayes 1979). Others have argued that story-like and other forms of "concrete" representations are the right way to represent commonsense (Schank and Abelson 1977). Still others have argued that much of commonsense is less about deliberative cognition than it is routine behavioral activity that operates using purely procedural representations (Agre and Chapman 1987). Some have argued that new concepts are built through composition and specialization of lexical semantic primitives (Jackendoff 1983). And there are many who believe that representations should be constructed by analogy to representations grounded in the physical and social world (Lakoff and Johnson 1980).

Our view is that when it comes to commonsense reasoning, diversity is the secret to success. To do commonsense reasoning one needs *many* kinds of representations. The more variety in representation, the more likely it is you will have the appropriate units of knowledge to solve whatever commonsense problem you

[1] http://www.openmind.org/commonsense

are faced with at the moment. It is not so much a matter of choosing between such representations as it is a matter of finding way for them to work together in one system.

To this end we constructed a variety of activities for gathering knowledge. Each activity tries to make it simple for the user to teach a certain kind of knowledge. At present we have about 25 activities. Some of the kinds of knowledge we collect include:

- *Spatial.* Describe a simple movie of a spatial event.
- *Relation.* Explain relation between pair of words.
- *Images*. Describe a photo of typical scene.
- *Stories*. Supply a story given a story title.
- *Verbs*. Supply typical arguments of a verb.
- *Objects*. List objects that tend to appear together.
- *Explanation*. Explain an aspect of a short story.
- *Teleology*. Provide the functions of an object.
- *Causality*. Give the likely effects of an action.

Each activity has its own special interface for knowledge elicitation. We regularly add new activities that gather new kinds of knowledge.

Users prompted with story:

Bob had a cold.
Bob went to the doctor.

Users enter many kinds of knowledge in response:

Bob was feeling sick
Bob wanted to feel better
The doctor made Bob feel better
People with colds sneeze
The doctor wore a stethoscope around his neck
A stethoscope is a piece of medical equipment
The doctor might have worn a white coat
A doctor is a highly trained professional
You can help a sick person with medicine
A sneezing person is probably sick

Table 1: Typical Knowledge Entry Activity

Table 1 illustrates a typical knowledge entry activity, which gathers several kinds of background knowledge that would be useful for understanding the presented story: the likely prior situation, the likely posterior situation, some general facts about the characters involves and general rules about the domain. The result is a large amount of related knowledge about a specific kind of event.

Using natural language input

When we seek to obtain the greatest possible audience, we cannot assume from our users any special background in artificial intelligence or knowledge engineering. We considered using the Cyc representation language for our web-based effort, but because our target audience is the average layperson, we could not expect our users to learn a complex formal language like CycL and its associated vast ontology of terms.

We decided instead to allow our users to enter knowledge in plain English. English gives us a convenient representation which many people already understand, and much commonsense knowledge can be expressed in terms of simple generic English sentences. We encouraged our users to enter sentences that "even a child could understand". As a result, we have accumulated a large set of sentences in relatively simple English. Some of the kinds of knowledge our users have supplied include:

- *Taxonomic*: Cats are mammals
- *Causality*: Eating a sandwich makes you less hungry
- *Goals*: People want to be warm
- *Spatial*: You often find a hairdryer in a bathroom
- *Functional*: Chairs are for sitting on
- *Planning*: To become less hungry, eat a sandwich
- *Grammatical*. "is not" is the same as "isn't"

Improvements in the basic techniques of natural language processing over the past decade further suggested to us that a modern attempt to build a commonsense knowledgebase might reasonably be based on natural language input. Modern part-of-speech taggers claim over 99% accuracy on text (Voutilainen 1995). Dependency grammars achieve an attachment accuracy of 93% (Eisner 1996). Syntactic parsers achieve 90% average precision/recall when tested on text from the Wall Street Journal (Charniak 1999). Further arguments for using English for knowledge entry are also made in (Fuchs and Schwitter 1996) and (Pulman 1996), where it is suggested that using a controlled subset of English makes it far easier for domain experts to supply knowledge to systems in a way that is still computer processable.

In addition to free-form English, many of our activities use a template-based input system in which users are restricted to entering knowledge into narrow fields. These templates were designed to reflect the primitive concepts out of which many of the most useful representations could be built: goals, scripts, plans, structural descriptions, causal models for diagnostic reasoning, explanations for explanation-based reasoning, and others. For example:

- Functional. A hammer is for _____
- Goals. People want _____
- Scripts. The effect of eating a sandwich is _____
- Location. Somewhere you find a bed is _____
- Ontology. A typical activity is _____

While these sorts of clearly formed knowledge templates are often easier to parse than free form English, we worried using only this kind of input would restrict us to gathering only a tiny slice of the kinds of knowledge people could express. We wanted to avoid imposing too strongly our own notions of what kinds of knowledge were important, and instead try to learn from our users more about what sorts of commonsense people could actually express. So while we find that knowledge from templates is often clearer and easier to parse, we continue to use both forms of entry.

Gathering scripts and frames

In addition to simple sentences we also gather several kinds of larger structures:

Concrete episodes. We believe much reasoning is done not only with abstract logical inferences, but also by reasoning by analogy to concrete episodes. Therefore we have our users supply stories, either to illustrate some existing fact like "flashlight light up places" or in response to a given title like "going outside at night". Example:

It was too dark to see. I went and got my flashlight. I turned on my flashlight. I could see much better.

Concrete situations. In case-based reasoning, rather than inferring relationships from first principles one simply "looks and sees". This sort of knowledge is different from a fact or a rule in that it is simply a description of a partial state of the world. Therefore we have people supply descriptions of photos in plain English. Example:

A mother is holding her baby. The baby is smiling. They are looking into each other's eyes. The baby is happy. The mother is happy.

Visual events. To allow more sophisticated forms of spatial reasoning, we allow users to annotate movies of simple iconic spatial events. Our goal is to eventually learn translation rules that let us produce event descriptions from movies and vice-versa. This is a first step towards reasoning using multimodal representations. Example:

The small red ball rolls past the big blue ball.

Breakdown of collection knowledge

So far we have collected about 400,000 pieces of knowledge from about 8,000 users. For analysis we manually decomposed this knowledge into a set of about 90 groups, each extracted using a particular regular expression pattern. We have organized these groups into a small set of families, whose distribution in the database is shown in below in Table 2.

Class of Knowledge	% Collected
Scripts/Plans	14.4
Causal/Functional	11.9
Spatial/Location	10.18
Goals/Likes/Dislikes	5.5
Grammatical	5.5
Photo descriptions	5.4
Properties of people	4.8
Explanations	2.6
Story events	1.8
Other	33.7

Table 2: Breakdown of Collected Knowledge

Many of these patterns resulted from our initial set of input templates, but others just seems to be patterns that seem to occur naturally when people are asked to produce facts. About a third of the collected knowledge does not seem to fit into any of the patterns we have constructed, but we are presently working on decomposing these into further patterns. Our hope is to build a large set of inference rules based on these patterns.

Distribution of knowledgebase

Our goal is for this database to be a free resource for helping study how to build systems capable of commonsense reasoning. We have made the data publicly available for download at our web site in several formats.

Reasoning in English

The problem still remains that knowledge supplied by users in English must be parsed into a target representation that is as expressive as English itself, if we want to make use most of that knowledge. At this point we once again considered using the Cyc ontology, but we needed a way to parse natural language into that commonsense ontology, which has proven daunting even for the Cyc project.

We believe there is an option that has been largely overlooked within the field of knowledge representation, which is *to use English itself as the knowledge representation.* As observed in (Szolovitz, Hawkinson, and Martin 1977), "The greatest attraction of this approach is that it almost trivially satisfies our need for expressive power." The idea is that English can perhaps serve as itself the representation over which reasoning is done. The value of this approach is three-fold:

- We can avoid having to impose a novel ontological structure on the universe beyond that which English has already supplied us.
- There is no need to create and learn a massive ontology. We can just use familiar English words and expressions.
- We can avoid having to do difficult and error prone "semantic interpretation" before reasoning can begin.

Inference by Natural Language Reformulation

In this section we describe a first attempt at using some of the knowledge we have collected. Our goal was to develop a reasoning system that could operate on natural language expressions directly, without having to go through a difficult semantic interpretation stage where the sentences were converted to logical form.

REFORMULATOR is an inference system that operates on simple English sentences. It is a forward chaining rule-based system that pattern matches shallow parses of English sentences to produce new English sentences. REFORMULATOR uses English expressions to represent knowledge. For example, instead of the symbol *$PresidentOfUnitedStates* it simply uses the phrase *president of the united states*. Similarly for predicates,

instead of *($kissevent $John $Mary)*, it simply uses *John kissed Mary*. We now explain how REFORMULATOR can operate on natural language, by supplying it with five special kinds of reformulation rules: *paraphrase*, *disambiguation*, *splitting*, *merging*, and *inference*.

Reformulation Rules

We supply inference rules to REFORMULATOR by example, by first eliminating determiners and replacing words with their root forms, and then substituting with variables parenthesized phrases that match across the sentences of the pattern. For instance, the example

(A door) is a portal
(Bob) opens (the door)
→ (Bob) can go through (the door)

generalizes to the following syntactic pattern

(S (NP ?x) (VP is (NP ?y)))
(S (NP ?z) (VP opens (NP ?x)))
→ (S (NP ?z)
    (VP can go (PP through (NP ?x))))

For conciseness, in the following sections we give examples of reformulation rules rather than the rules themselves.

Disambiguation rules. When it comes to domains as broad as the world of human concerns, it is very hard to pin down the meanings of terms with ultimate precision. In REFORMULATOR, ambiguity within an expression is dealt with by reformulating it to less ambiguous forms. This can be done in several ways, including replacing an ambiguous word with a less ambiguous word or multi-word expression, by elaborating the expression with additional text, or by making structural modifications to the sentence. For example, the lexical ambiguity of the word *president* might be dealt with by reformulating to reasonably unambiguous multiword expressions:

President
→ president of the united states
→ president of a corporation

The structural ambiguity of the sentence *The chicken is ready to eat* can be dealt with by reformulating the sentence so that it uses a less ambiguous structure:

The chicken is ready to eat
→ The chicken is ready to eat something
→ The chicken is ready to be eaten

We call these *disambiguation* rules. In the case of lexical ambiguity we do word sense disambiguation by correlating the word's surrounding context with the terms in co-location vectors found for each of the word senses. These co-location vectors can be obtained through supplying the disambiguated item (e.g. "president of the united states") to a typical web search engine, and extracting content words correlated across the result pages. For the above examples, we might obtain:

president of the united states: America, Bush, election
president of a corporation: stock, CEO, investors

This disambiguation method is far from perfect, and for structural ambiguity the disambiguation problem is even more difficult. In general, more context and general common sense reasoning is required to do sense disambiguation.

Paraphrase rules. English is also ambiguous in a different sense, in that, unlike the Cyc ontology, there are generally many ways to say the same thing. One might try to formulate a "canonical English" in which there would be exactly one expression to correspond to any given idea. For instance, one could say only *A cat is a mammal*, and alternative forms like *Cats are mammals* or *A cat is a kind of mammal* would be ruled out. We decided that people would probably be nearly as unwilling to learn a canonical English as they would to learn a vast logical ontology.

REFORMULATOR avoids canonical expressions through the use of *paraphrasing* rules that allow it to move between similar ideas expressed in different ways. These different formulations need not be precisely identical in their meanings, and may each supply a slightly different viewpoint and emphasis, as in the following example:

Bob likes to eat apples.
↔ If Bob eats an apple, then he will enjoy it.
↔ Eating an apple will make Bob happy.
↔ If I were Bob, then I would like apples.

Paraphrasing rules are important because, lacking canonical forms, different inference rules supplied by different people may require different formulations of an expression to match.

Splitting and Merging Rules. While we ask our users to enter knowledge in simple ways, it is sometimes easiest to enter sentences of intermediate complexity, in order to convey the most information in fewest words. However, this makes inference difficult because expressions will not match if knowledge is spread across multiple sentences. Therefore REFORMULATOR allows complex sentences to be broken apart and recombined in many ways through the application of *splitting* and *merging* rules. For example:

- Splitting:

A tiger is a ferocious cat that lives in the jungle
→ Tigers are ferocious
   A tiger is a kind of cat
   Tigers live in the jungle

- Merging:

Tigers are ferocious
A tiger is a kind of cat
→ Tigers are ferocious cats

These two kinds of rules are the special case of paraphrasing between complex sentences and their sets of constituent phrases expressed as full sentences.

General Inference rules. The syntactic complexity of English prevents us from using uniform proof procedures such as resolution theorem proving. REFORMULATOR operates instead through the application of heuristic inference rules. Examples of such rules are listed below.

- Taxonomic inference:

  Cats are mammals
  Mammals are animals
  → *Cats are animals*

- Inference about requirements:

  You need air to live
  There is no air in space
  → *You cannot live in space*

- Inference about effects:

  Pushing a door will open the door
  Bob pushed the door
  → *Bob opened the door*

- Default inferences:

  Bob drove
  → *Bob drove in a car*
     Bob owns the car

One advantage of this approach is that we can easily read off the inference trace, which could lead to an improved ability to maintain and debug the knowledgebase.

A Search Engine with Common Sense

Let us now demonstrate how these ingredients can be combined to solve a real world problem in the area of web search. Using REFORMULATOR, we have built a simple search engine application that reformulates problem queries into more effective action queries.

For example, let us imagine the problem that your pet cat is ill. Our experience is that when faced with a search engine for the first time, a person not familiar with the limits of computer technology is likely to enter not an efficient query like *I want a veterinarian* but rather a statement of their problem, such as *My cat is sick*. They treat the search engine as if it were capable of making the commonsense inference that a sick cat requires a veterinarian.

REFORMULATOR acts to make this inference. It reformulates a user query which describes a problem into a more focused query that is a step towards solving the problem. Given a search query, it forward chains on its knowledge, until it encounters the following pattern:

  I want ?VP
  ?VP is a concrete action

The following is the chain of reasoning that results from entering the search query *My cat is sick*, given the initial database contents supplied below.

Query: *My cat is sick*

Initial database contents:

(a) *My cat is sick*
(b) *People care about their pets*
(c) *I am a person*
(d) *A cat is a kind of pet*
(e) *A veterinarian helps sick animals*
(f) *A cat is a kind of animal*
(g) *A veterinarian is a kind of person*

Chain of inference:

People care about their pets	(b)
I am a person	(c)
→ *(h) I care about my pets*	(1)
My cat is sick	(a)
A cat is a kind of pet	(d)
→ *(i) My pet is sick*	(2)
A veterinarian helps sick pets	(e)
→ *(j) A veterinarian heals sick pets*	(7)
A veterinarian heals sick pets	(j)
→ *(k) A veterinarian makes sick pets healthy*	(4)
A veterinarian is a kind of person	(g)
→ *(l) Calling a veterinarian is a concrete action*	(6)
I care about my pets	(h)
→ *(m) I want my pets to be healthy*	(3)
I want my pets to be healthy	(m)
My pet is sick	(i)
A veterinarian makes sick pets healthy	(k)
→ *(n) I want to call a veterinarian*	(5)
I want to call a veterinarian	(n)
Calling a veterinarian is a concrete action	(l)
→ *Search for "call a veterinarian"*	(8)

The system was endowed with the following set of reformulation rules.

(1) *People ?P their ?Q*   (Inference)
     I am a person
     → *I ?P my ?Q*

(2) *A ?NOUN1 is a kind of ?NOUN2*   (Inference)
     ?P ?NOUN1 ?Q
     → *?P ?NOUN2 ?Q*

(3) *I care about ?NOUN*   (Reformulation)
     → *I want ?NOUN to be healthy*

(4) *A ?P heals ?Q*   (Reformulation)
     → *A ?P makes ?Q healthy*

(5) *I want my ?NOUN1 to be ?ADJ1*   (Inference)
     my ?NOUN1 is ?ADJ2

A ?NOUN2 makes ?ADJ2 ?NOUN1 ?ADJ1
→ *I want to call a ?NOUN2*

(6) *A ?NOUN is a kind of person*   (Inference)
→ *Calling a ?NOUN is a concrete action*

(7) *A ?P helps ?Q*   (Disambiguation)
→ *A ?P heals ?Q*

(8) *I want to ?VP*   (Goal)
?VP is a concrete action
→ *Search for ?VP*

The end result is that the system reformulates the query *My cat is sick* into *Call a veterinarian*, in effect guessing at the real purpose behind the user's search query.

Conclusions

Open Mind Commonsense has gathered hundreds of thousands of small pieces of commonsense knowledge, and it continues to grow. We hope that the ideas in this paper will help us build an inference system capable of reasoning with this knowledge, and that we will find new ways to make available the full expressive power of a natural language for commonsense reasoning. We feel we have explored only the very surface of ways to extract commonsense knowledge from the general public, and hope that others will be inspired to follow with new approaches. We have received much feedback from our users saying how they very much enjoy entering knowledge and working with our system. We can only speculate that we are drawing on some basic human instinct to pass on our commonsense to our progeny.

Acknowledgements

References

Agre, P. and Chapman, D. 1987. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 268-272. Menu Park, Calif.: AAAI Press.

Charniak, E. 1999. A maximum-entropy-inspired parser. Technical Report CS-99-12, Brown University.

Eisner, J. M. 1996. An empirical comparison of probability models for dependency grammar. Technical report IRCS-96-11, Institute for Research in Cognitive Science, Univ. of Pennsylvania.

Fuchs, E. & Schwitter, R. 1996. Attempto Controlled English (ACE). In *Proceedings of The First International Workshop On Controlled Language Applications*. Katholieke Universiteit Leuven, pages 124-136, Belgium.

Hayes, P. J. The Naïve Physics Manifesto, in D. Michie (ed.), *Expert Systems in the Micro-Electronic Age* (Edinburgh University Press, 1979), pp. 242-70.

Jackendoff, R. 1983. *Semantics and Cognition*. The MIT Press, Cambridge, MA.

Lakoff, G. and Johnson, M. 1980. *Metaphors We Live By*. The University of Chicago Press. Chicago.

Lenat, D. B. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33-38.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313--330.

Pulman, S. 1996. Controlled language for knowledge representation. In *Proceedings of the first international workshop on controlled language applications*, Katholieke Universiteit Leuven, Belgium.

Schank, R. C., and Abelson, R. P. 1977. *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Sleator, D., and Temperley, D. 1991. Parsing English with a link grammar. Technical Report CMU-CS-91-196, Dept. of Computer Science, Carnegie Mellon University.

Stork, D. 1999. The OpenMind Initiative. *IEEE Intelligent Systems & their applications* 14(3):19-20.

Szolovits, P., Hawkinson, L. B., Martin, W. A. 1977. An overview of Owl, a language for knowledge representation. Technical Memo TM-86, Laboratory for Computer Science, MIT.

Voutilainen, A. 1995. A syntax-based part-of-speech analyser. In *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics*, Dublin.