CrossBridge: Finding Analogies using Dimensionality Reduction

Abstract

We present CROSSBRIDGE, a practical algorithm for retrieving analogies in large, sparse semantic networks. Other algorithms adopt a generate-and-test approach, retrieving candidate analogies by superficial similarity of concepts, then testing them for the particular relations involved in the analogy.

CROSSBRIDGE adopts a global approach. It organizes the entire knowledge space at once, as a matrix of small concept-and-relation subgraph patterns versus actual occurrences of subgraphs from the knowledge base. It uses the familiar mathematics of dimensionality reduction to reorganize this space along dimensions representing approximate semantic similarity of these subgraphs. Analogies can then be retrieved by simple nearest-neighbor comparison.

CROSSBRIDGE also takes into account not only knowledge directly related to the source and target domains, but also a large background Commonsense knowledge base. Commonsense influences the mapping between domains, preserving important relations while ignoring others. This property allows CROSSBRIDGE to find more intuitive and extensible analogies.

We compare our approach with an implementation of structure mapping and show that our algorithm consistently finds analogies in cases where structure mapping fails. We also present some discovered analogies.

Introduction

Analogies are comparisons that highlight commonalities between two seemingly dissimilar ideas. People frequently use analogies to teach others (Podolefsky and Finkelstein 2006), to solve problems (Gick and Holyoak 1980) and to understand unfamiliar situations. For example, the analogy "an atom is like the solar system" enables us to reason about protons and electrons using knowledge about the Sun and the Earth. The prevalence of analogies in human thought makes them an interesting topic for AI research, and many researchers believe that analogies are an important part of human cognition (Minsky 1988; Lakoff and Johnson 2003).

Analogies are an integral part of common sense reasoning, as analogies dictate how prior knowledge is applied to novel domains. People frequently solve problems using a nearest-neighbors search: they remember a similar problem, make an analogy to their current problem, then use the analogy to modify the old solution. Case-based reasoning systems implement this problem-solving process (Kolodner 1993). A crucial component of these systems is an analogy mechanism which retrieves and adapts solutions from a knowledge base.

Structure mapping theory (Gentner 1983), a well-known theory of analogy, describes analogies as mappings from a *source domain* to a *target domain*. Domains contain *concepts* and *relations*. In this paper, we represent domains using semantic networks, where concepts are vertices and relations are typed edges. An analogy is a correspondence between the concepts of two domains that preserves the relations between concepts. For example, consider the "bird domain" and the "car domain" shown in Figure 1. An analogy could include a mapping from "bird" to "car" and from "wing" to "wheel" because this mapping preserves the **PartOf** relation: **PartOf**(wing, bird) maps to **PartOf**(wheel, car). In fact, structure mapping theory states that two domains are analogous if their semantic networks are isomorphic.

Structure mapping clearly defines analogy, but finding analogies in real knowledge bases using structure mapping is difficult. The problem is a generalization of subgraph isomorphism, which is NP-complete. Further, structure mapping does not specify how to infer reasonable analogies from sparse knowledge bases, causing it to miss many plausible analogies. In our evaluation, we show that structure mapping misses many plausible analogies found by CROSSBRIDGE.

CROSSBRIDGE implements an approximate form of structure mapping that allows it to find plausible analogies even in sparse knowledge bases. CROSSBRIDGE's essential assumption is *the more isomorphic two domains are, the more analogous they are*. CROSSBRIDGE defines a similarity measure between domains which approximates graph isomorphism, then uses it as a measure of analogousness. The similarity measure is defined as the cosine similarity between domains in *domain space*, a vector space representation of domains. We use dimensionality reduction when constructing the vector space, which reorganizes the space based on global patterns in the knowledge base and allows the entire knowledge base to influence the analogy found

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

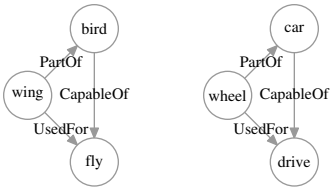

Figure 1: The bird domain and the car domain. An analogy between these domains maps "bird" to "car," "wing" to "wheel," and "fly" to "drive." This mapping is an analogy because it preserves the relations between the concepts in each domain.

between a pair of domains. Dimensionality reduction also improves the efficiency of scoring analogies. Analogies are found efficiently using a nearest-neighbors search in domain space. Our evaluation shows that CROSSBRIDGE is a practical algorithm for analogy retrieval in knowledge bases with thousands of concepts and tens of thousands of relations.

The Analogy Retrieval Problem

We consider the problem of retrieving analogies for a given target domain T from a large knowledge base G. Both the domain and the knowledge base are represented as semantic networks, that is, graphs with typed edges. Each subgraph of G forms a possible source domain for an analogy. The goal is to find possible source domains in G and analogies between these domains and T. For example, an instance of this problem is "What is analogous to the bird domain?" A legitimate response is a list of analogies to domains in the knowledge base, which in our case would include the car domain.

Our definition of analogy is based on structure mapping (Gentner 1983). According to structure mapping theory, an analogy is a correspondence between the concepts of two domains that preserves the system of relations within the domains. In a semantic network, an analogy is therefore a mapping from the vertices of a source domain S to the vertices of the target domain T that preserves the edges in S. That is, if edge (v_0, v_1, r) is part of S and an analogy maps v_0 to t_0 and v_1 to t_1, then T contains the edge (t_0, t_1, r).

Unfortunately, finding analogies according to this definition is NP-complete, as analogy retrieval is a generalization of subgraph isomorphism. CROSSBRIDGE approximates this isomorphism computation, returning a list of analogies ordered by plausibility. The plausibility score is related to the number of edges preserved by the analogical mapping, but is also influenced by correlations between relations and other statistical information from the entire knowledge base. These properties of the scoring function also allow CROSSBRIDGE to operate on sparse knowledge bases.

CrossBridge

CROSSBRIDGE, our algorithm for analogy retrieval, has two major steps. The first step is a preprocessing step in which it constructs *domain space* from a knowledge base. During this step, the knowledge base is divided into many fixed-size source domains, and each of these source domains is represented as a vector in domain space. These vectors capture the system of relations within a domain; therefore, the plausibility of an analogy is a function of the distance between the domains in domain space. The second step retrieves analogies for a target domain using similarity information from domain space and a heuristic search.

Domain Space

Domain space is a vector space containing a subset of the domains found in the knowledge base. Recall that a domain corresponds to a subgraph of the knowledge base, where the knowledge base is a large semantic network. In general, a graph contains an exponential number of subgraphs, so we therefore cannot place all domains into domain space. We therefore only include domains with l concepts and more than m relations. (The constraint on edges is reasonable because domains with very few edges are not likely candidates for analogies.) In our evaluation, we use $l = 3$ and $m = 2$.

Even with these limits, constructing domain space is an expensive operation, as it scans the entire knowledge base and extracts all viable domains. However, domain space must be constructed only once per knowledge base – it is an index which helps us efficiently find analogies. We also note that application-specific filters could reduce the number of viable domains, which would improve the efficiency of this process.

Domain space actually contains multiple copies of each domain, each with a different ordering of its vertices. Ordering the vertices of each domain helps us construct analogies between similar domains. To understand the purpose of this ordering, consider finding analogies for the bird domain. With the ordering, we would find that the bird domain matches the car domain only when the bird domain is ordered as say ("bird," "wing," "fly") and the car domain is ordered as ("car," "wheel," "drive"). To construct an analogy, we simply map "bird" to "car," "wing" to "wheel," etc. Every possible ordering of a source domain has its own vector representation in domain space.

To construct domain space, we search the knowledge base for appropriately-sized domains. We create a row of a binary matrix for each viable domain with each possible vertex ordering. The vector for each domain encodes the *relation structures* within the domain, which are characterizations of the relations in a domain. Once we have found all of the acceptable domains, we apply dimensionality reduction to the resulting matrix, which uses global cooccurrence patterns between relation structures to create a low-dimensional representation of each row. Domain space is the space containing these low-dimensional vectors.

Relation Structures The domain space vector for a domain encodes the system of relations within the domain so that nearly isomorphic domains have similar vectors. We create features for domains by extracting small systems of relations from them; each domain is then characterized by whether or not it contains each system of relations. We call these systems of relations *relation structures* because they characterize the "structure" of the domain which is preserved by structure mapping analogies. A relation structure is essentially a subset of the edges contained in a domain; al-

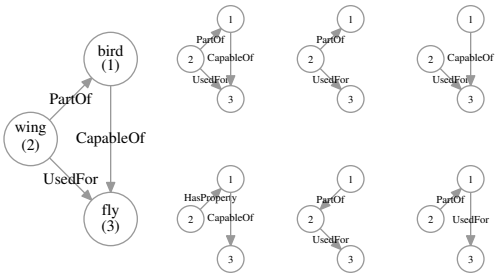

Figure 2: The bird domain, three relation structures in the domain (top), and three relation structures not in the domain (bottom). The vertices of the bird domain are ordered from top-to-bottom, and the position of each vertex is explicitly shown in parentheses.

ternatively, it is an l-ary relation formed by combining several binary relations. Relation structures are extracted by replacing each vertex with its position in the vertex ordering, then selecting a subset of the domain's edges. Figure 2 shows some of the relation structures which can be extracted from the bird domain.

We limit the size of the relation structures we use, as the number of possible relation structures grows exponentially in the number of edges in each structure. We therefore only use relation structures with between f_{min} and f_{max} edges. Note that relation structures with too many edges will match very few graphs, so rare structures are unlikely to help analogy retrieval. A typical setting is $f_{\text{min}} = 1$ and $f_{\text{max}} = 3$. However, in our evaluation, we set $f_{\text{min}} = f_{\text{max}} = 1$, as this setting gives the closest approximation of structure mapping.

Domain space represents each domain (with a particular vertex ordering) as a vector of binary variables that encodes the relation structures contained within the domain. Note that if two domains are isomorphic, they will share the same relation structures when their vertices are correctly ordered. However, when incorrectly ordered, they may not share any of the same relation structures. The two orderings describe a mapping between the vertices of the two domains, as we construct analogies by simply pairing off the vertices in each position. The vectors for two domains are only similar when this implied mapping is nearly an isomorphism.

Dimensionality Reduction Domain space is constructed by performing dimensionality reduction on the relation structure vectors. The previous section described how to characterize each extracted domain as a vector, where each entry of the vector is a binary value describing whether the domain contains a particular relation structure. To construct domain space, we place these vectors into a matrix M, then perform dimensionality reduction on M. Figure 3 shows a section of M before dimensionality reduction. The rows of M are graphs with ordered vertices, and the columns of M are relation structures. Each entry encodes whether the corresponding graph contains the corresponding relation structure.

Domain space is a reduced-dimensional approximation of M. We use a truncated singular value decomposition (SVD) for dimensionality reduction, which reorganizes the space

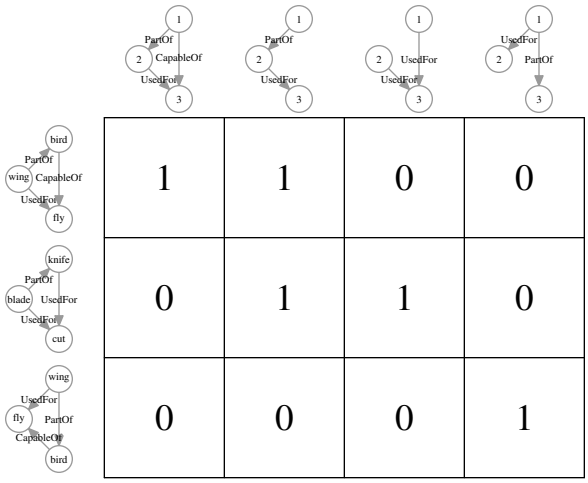

Figure 3: A section of the matrix M using concepts and relations from ConceptNet. The rows are subgraphs from the knowledge base, and the columns are relation structures. The vertices of the subgraphs are ordered from top-to-bottom. Entries of the matrix encode the relation structures contained in each subgraph.

based on the most important dimensions, and also helps smooth over sparsity. This use of dimensionality reduction is similar to Latent Semantic Analysis (Deerwester et al. 1990). The truncated SVD defines rank-k matrices U, Σ, V such that $M \approx U\Sigma V^T$. Domain space is then $U\Sigma$. Domain space represents every graph G with vertex ordering P as a k-dimensional vector, $u_{(G,P)}$. The dimensions of domain space capture global correlations between relation structures. Similar graphs in domain space therefore contain correlated, but not necessarily identical, relation structures. This property enables us to robustly approximate structure mapping using vector similarity in domain space. In our evaluation, we use $k = 50$.

Retrieving Analogies

There are two different procedures for retrieving analogies depending on the size of the target domain T. Recall that domain space only contains domains with exactly l concepts. If the target domain T also contains l concepts, we simply use a nearest-neighbors search in domain space to find similar domains. If T contains more than l concepts, we split T into several domains with l concepts, find analogies for these domains, then merge the resulting analogies using a heuristic search.

To retrieve analogies for an l-concept target domain T (e.g., the bird domain), we must represent it in domain space. We first choose an arbitrary vertex ordering P for T. If T is a subgraph of the knowledge base, then (T, P) is already represented in domain space. Otherwise, we check which relation structures are in T, construct the corresponding vector, then project this vector into domain space using the V matrix returned by the SVD. Let $u_{(T,P)}$ be the resulting vector representation of (T, P).

To find source domains, we compute the cosine similarity between $u_{(T,P)}$ and every other domain vector $u_{(S,Q)}$. We sort the results to find the most similar domains to T.

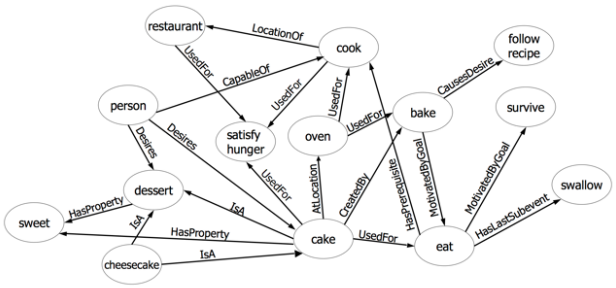

Figure 4: A portion of the ConceptNet semantic network.

Finally, we construct an analogy between T and each similar source domain S using the vertex orderings. Recall that the analogy simply pairs off the vertices in each position of the orderings P and Q. If we represent each ordering as a mapping from the vertices of the graph to $\{1, 2, ..., l\}$, then $P^{-1} \circ Q$ is an analogy from S to T. (Intuitively, use Q to map the vertices of S to $\{1, 2, ..., l\}$, then use P^{-1} to map $\{1, 2, ..., l\}$ to vertices of T.)

We find analogies for larger domains by combining similarity information from domain space with a heuristic search. We initially divide the target domain into several arbitrary l-concept subdomains and retrieve the 100 best analogies for each subdomain. We then repeatedly merge analogies containing at least two identical concept mappings. Intuitively, analogies that share several concept mappings are likely to refer to the same region of the source semantic network, which suggests that they can be combined. This heuristic seems to yield reasonable analogies in practice.

Evaluation

Evaluating analogy algorithms is difficult because it requires a subjective evaluation of analogy quality. To avoid this problem, we compare our algorithm to structure mapping. We define a set of test domains and show that CROSSBRIDGE retrieves analogies in cases where structure mapping fails. We then present some analogies retrieved by CROSSBRIDGE, but not structure mapping, to show that these additional analogies are reasonable.

Our evaluation uses ConceptNet (Havasi, Speer, and Alonso 2007), a large semantic network of common sense knowledge. ConceptNet is constructed by parsing simple sentences such as "a dog is a pet" into assertions such as **IsA**(dog, pet). Each assertion has an associated numerical quality score. These assertions belong to 21 different relation types. A portion of ConceptNet is shown in Figure 4. For our evaluation, we used a subset of ConceptNet containing all concepts involved in at least 5 assertions. We also required each assertion to have a score of at least 1. This semantic network contains 8426 vertices and 85440 edges.

We compare CROSSBRIDGE to two baseline algorithms based on structure mapping. Structure mapping was not designed for analogy retrieval, so we adapted it to retrieval for the sake of comparison. Our implementation naïvely tries to match the given target domain against every source domain in the knowledge base. For each source domain, it considers every possible mapping to the target domain, aborting the mapping as soon as it finds an unmapped edge. We call this algorithm STRUCTURE-MAP-0. We additionally defined a variant, STRUCTURE-MAP-1, that only aborts if two unmapped edges are found; this variant more closely resembles CROSSBRIDGE, as it allows some errors in the analogical mapping.

Test Set	CB	SM0	SM1
fly, bird, wing	237	0	86
fly, sky, bird, wing	418	0	72
fly, airplane, sky, bird, wing	548	0	3
fly, sky, wing, duck, airplane, bird	793	2	26
school, student, learn	245	1	26
school, book, student, learn	337	0	5
read, school, book, student, learn	872	0	5
school, read, text, book, student, learn	1297	0	1
wood, tree, forest	300	0	43
tree, wood, leaf, forest	477	1	31
tree, wood, leaf, forest, branch	995	1	32
leaf, tree, wood, forest, branch, snake	1919	3	31
table, eat, restaurant	231	79	13869
food, table, eat, restaurant	935	1	16
food, table, restaurant, eat, person	1799	1	20
plate, restaurant, food, person, table, eat	2814	0	8

Table 1: The number of analogies found by the analogy algorithms on each test set. These counts do not include the trivial analogy in which every concept is mapped to itself. Each test domain is the induced subgraph of ConceptNet containing the listed concepts.

Our first experiment shows that structure mapping frequently fails to find analogies. Table 1 shows the number of analogies found by each algorithm on a test set of target domains. Each domain is the subgraph of ConceptNet induced by the concepts in each row of the table. The target domains were chosen by examining ConceptNet for densely related sets of concepts that intuitively seemed like cohesive domains. Running structure mapping on all of ConceptNet was impractical for the larger test domains, so we restricted ConceptNet to assertions with a score of at least 2 for this experiment. As Table 1 shows, the variants of structure mapping find very few analogies on many of the test sets, while CROSSBRIDGE consistently finds a good number of analogies.

The additional analogies found by CROSSBRIDGE are also intuitively reasonable. Figures 5 and 6 show some analogies found by CROSSBRIDGE that were missed by both variants of structure mapping. This experiment uses the full version of ConceptNet, as structure mapping ran in a reasonable period of time for the chosen domains. The selected analogies are drawn from the top 50 results returned by CROSSBRIDGE and are chosen for diversity. (Many of the top source domains only differ from each other in one concept, as there are frequently multiple concepts which can replace each other in an analogy. For example, in the bird domain, we can replace "bird" with "plane," "airplane," etc. This behavior is reasonable for a computer, but not very

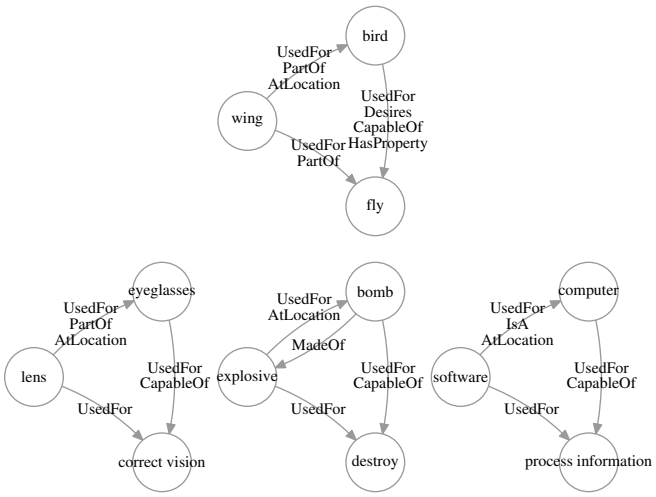

Figure 5: The bird domain (as it exists in ConceptNet) and three analogous source domains found by CrossBridge but not structure mapping.

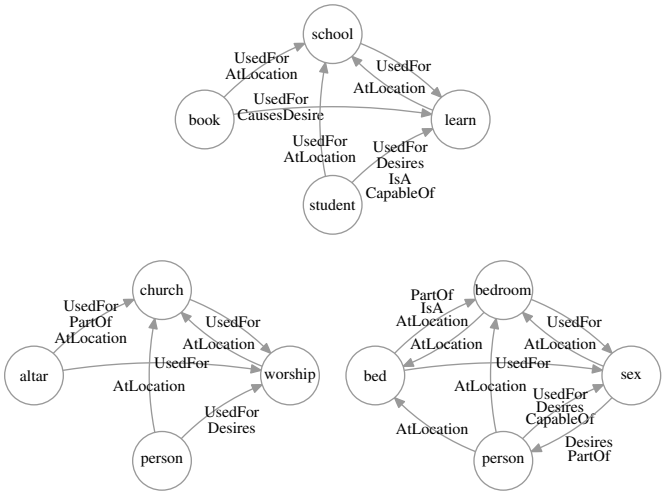

Figure 6: The school domain and two analogous source domains found by CrossBridge but not structure mapping.

interesting to humans.) The source domains in the figure are intuitively analogous to the target domain, but ConceptNet happens to represent each domain somewhat differently. These slight differences cause both structure mapping algorithms to fail, but CROSSBRIDGE is robust enough to handle the different representations.

Prior Work

Previous work on analogy has produced a rich literature on the subject, including several theories of analogy, and dozens of algorithms. In this section, we focus on the work most closely related to CROSSBRIDGE: analogy retrieval algorithms and analogy algorithms using dimensionality reduction. For a more complete discussion of analogy in AI, we refer the reader to (Melis and Veloso 1998).

Other algorithms for analogy retrieval, such as MAC/FAC (Gentner and Forbus 1991) and ARCS (Thagard et al. 1990), model the human analogy retrieval process, and therefore have a slightly different focus than CROSSBRIDGE. CROSSBRIDGE approximates structure mapping, and hence retrieves structurally similar analogies. Surprisingly, people tend not to retrieve structurally similar analogies, even though they prefer them (Gentner and Landers 1985). As structural similarity is useful for solving problems and transferring knowledge between domains (Gick and Holyoak 1980), we believe CROSSBRIDGE may outperform these algorithms in applications requiring knowledge transfer.

Of the analogy retrieval algorithms, our work is most closely related to MAC/FAC (Gentner and Forbus 1991), which also retrieves source domains using similarity in a vector space. MAC/FAC then uses the Structure Mapping Engine (Falkenhainer, Forbus, and Gentner 1989) to find a mapping from the target domain to each retrieved domain. However, the similarity measure used by MAC/FAC measures the superficial similarity of domains, which is the number of shared concepts and relations between the two domains. Although this choice emulates the human analogy retrieval process, it is not optimal for retrieving structurally similar analogies.

CROSSBRIDGE's use of dimensionality reduction resembles the Latent Relational Mapping Engine (LRME) (Turney 2008). LRME solves SAT analogy questions using a corpus of websites. The program determines the relationship between a pair of words by correlating the phrases that typically appear between the two words using the SVD. To solve an analogy, LRME finds the pair of target words that is most similar to the source words. LRME can also perform structure mapping using an exhaustive search over possible mappings. CROSSBRIDGE extends the technique used by LRME to arbitrarily large sets of concepts.

AnalogySpace (Speer, Havasi, and Lieberman 2008) also uses the SVD to find similar concepts in ConceptNet (Havasi, Speer, and Alonso 2007). However, the similarity of two concepts is based on their shared properties. AnalogySpace does not operate on sets of concepts, and therefore cannot perform structure mapping.

We believe that CROSSBRIDGE also addresses some previously mentioned problems with structure mapping. Structure mapping has previously been criticized because it does not construct its own representation of each domain (Chalmers, French, and Hofstadter 1991). Structure mapping assumes that analogous domains are input in the same format, with the same relations between concepts, etc. This assumption is unrealistic, as a domain can be represented in many different ways. Therefore, Chalmers et al. argue that an analogy algorithm must also learn a representation of each domain as part of the analogy process. CROSSBRIDGE is a step in this direction, as CROSSBRIDGE uses dimensionality reduction to learn a new representation of each domain in its knowledge base. Further, CROSSBRIDGE does not use a representation that is chosen by the researcher – although the data in ConceptNet is manually entered, the relations are not chosen to make certain domains analogous.

Discussion and Future Work

CROSSBRIDGE has several limitations which we would like to address in future work. For example, CROSSBRIDGE currently only operates in semantic networks, while past work has operated in the richer domain of propositional logic. We think it will be relatively easy to extend CROSSBRIDGE to operate on propositional logic statements; we only have to modify the process for selecting domains and constructing relation structures. However, this extension will be useful for practical applications, and will also allow us to incorporate the systematicity criterion from structure mapping theory (Gentner 1983). We could additionally include other types of constraints on the retrieved analogies using blending (Havasi et al. 2009).

We think common sense knowledge (like the knowledge in ConceptNet) is a particularly interesting domain for analogies. Since we expect people to possess this common sense knowledge, we expect them to understand analogies to common sense. We would also expect a corpus of common sense to contain all of the basic "types" of domains. One possible area for future work is to explain difficult concepts using analogies to common sense knowledge. For example, SuggestDesk (Lieberman and Kumar 2005) used analogies to explain technical problems and their solutions to users.

These applications will have to combine specialized knowledge with common sense knowledge. Blending (Havasi et al. 2009) is a powerful technique for combining data sets using the SVD. Using blending, we can construct domain space with both common sense knowledge and specialized, application-specific knowledge. This domain space will allow us to make analogies from specialized knowledge to common sense, and vice versa. We believe this technique could be useful for explaining technical ideas.

We believe that CROSSBRIDGE is a significant improvement over previous analogy algorithms because CROSSBRIDGE makes analogy a practical reasoning mechanism. CROSSBRIDGE can be considered a technique for performing approximate structure mapping in semantic networks, and is capable of efficiently retrieving analogies from large, sparse data sets. These properties suggest that CROSSBRIDGE may be a useful component of applications that solve problems by analogy.

An implementation of CROSSBRIDGE is available in Divisi, a toolkit for common sense reasoning. Divisi is available from `http://divisi.media.mit.edu/`.

References

Chalmers, D.; French, R.; and Hofstadter, D. 1991. High-level perception, representation, and analogy: A critique of artificial intelligence methodology.

Deerwester, S.; Dumais, S. T.; Furnas, G. W.; Thomas; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41:391–407.

Falkenhainer, B.; Forbus, K. D.; and Gentner, D. 1989. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence* 41(1):1–63.

Gentner, D., and Forbus, K. D. 1991. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science* 19:141–205.

Gentner, D., and Landers, R. 1985. Analogical reminding : A good match is hard to find. In *Proceedings of the International Conference on Systems, Man and Cybernetics*.

Gentner, D. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7(2):155–170.

Gick, M. L., and Holyoak, K. J. 1980. Analogical problem solving. *Cognitive Psychology* 12:306–355.

Havasi, C.; Speer, R.; Pustejovsky, J.; and Lieberman, H. 2009. Digital intuition: Applying common sense using dimensionality reduction. To appear in IEEE Intelligent Systems.

Havasi, C.; Speer, R.; and Alonso, J. 2007. Conceptnet 3: a flexible, multilingual semantic network for common sense knowledge. In *Proceedings of Recent Advances in Natural Languge Processing 2007*.

Kolodner, J. 1993. *Case-Based Reasoning*. San Mateo: Morgan Kaufmann.

Lakoff, G., and Johnson, M. 2003. *Metaphors We Live by*. University of Chicago Press.

Lieberman, H., and Kumar, A. 2005. Providing expert advice by analogy for on-line help. *Intelligent Agent Technology, IEEE / WIC / ACM International Conference on* 0:26–32.

Melis, E., and Veloso, M. 1998. Analogy in problem solving. In *Handbook of Practical Reasoning: Computational and Theoretical Aspects*. Oxford University Press.

Minsky, M. 1988. *Society of Mind*. Simon & Schuster.

Podolefsky, N. S., and Finkelstein, N. 2006. Use of analogy in learning physics: The role of representations. *Phys. Rev. ST Phys. Educ. Res.* 2(2).

Speer, R.; Havasi, C.; and Lieberman, H. 2008. Analogyspace: Reducing the dimensionality of common sense knowledge. In Fox, D., and Gomes, C. P., eds., *AAAI*, 548–553. AAAI Press.

Thagard, P.; Holyoak, K. J.; Nelson, G.; and Gochfeld, D. 1990. Analog retrieval by constraint satisfaction.

Turney, P. D. 2008. The latent relation mapping engine: Algorithm and experiments. NRC-50738.