The Goal of Language Understanding

Chapter 6: Analogy and case-based reasoning

John F. Sowa

VivoMind Research, LLC

1 December 2013

The Goal of Language Understanding

Outline:

1. Problems and challenges (goal.pdf)
2. Psycholinguistics and neuroscience (goal2.pdf)
3. Semantics of natural languages (goal3.pdf)
4. Wittgenstein's early and later philosophy (goal4.pdf)
5. Dynamics of language and reasoning (goal5.pdf)
6. Analogy and case-based reasoning
7. Learning by reading (goal7.pdf)

Each chapter is in a separate file. Later chapters make occasional references to earlier chapters, but they can be read independently.

6. Analogy and Case-Based Reasoning

Based on the same kind of pattern matching as perception:

- **Associative retrieval by matching patterns.**
- **Approximate pattern matching for analogies and metaphors.**
- **Precise pattern matching for logic and mathematics.**

Analogies can support informal, case-based reasoning:

- **Long-term memory can store large numbers of previous experiences.**
- **Any new case can be matched to similar cases in long-term memory.**
- **Close matches are ranked by a measure of semantic distance.**

Formal reasoning is based on a disciplined use of analogy:

- **Induction: Generalize multiple cases to create rules or axioms.**
- **Deduction: Match (unify) a new case with part of some rule or axiom.**
- **Abduction: Form a hypothesis based on aspects of similar cases.**

How can a computer understand language?

According to Alan Turing (1950),

If people can't tell the difference between what a computer does and and what a person does, then the computer is thinking the way people do.

A more implementable idea:

Human thinking uses analogies to find and relate patterns.

People understand language by finding analogies between patterns of words and patterns in what they see and remember.

Question:

How can we use analogies to make computers more human-like?

Describing Things in Different Ways

How can we describe what we see?

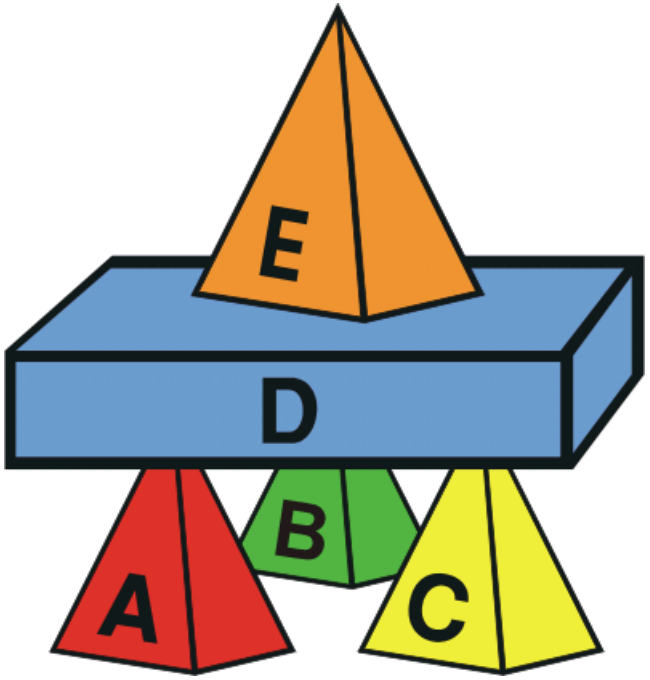
In ordinary language?

In some version of logic?

In a relational database?

In the Semantic Web?

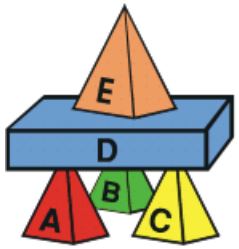
In a programming language?

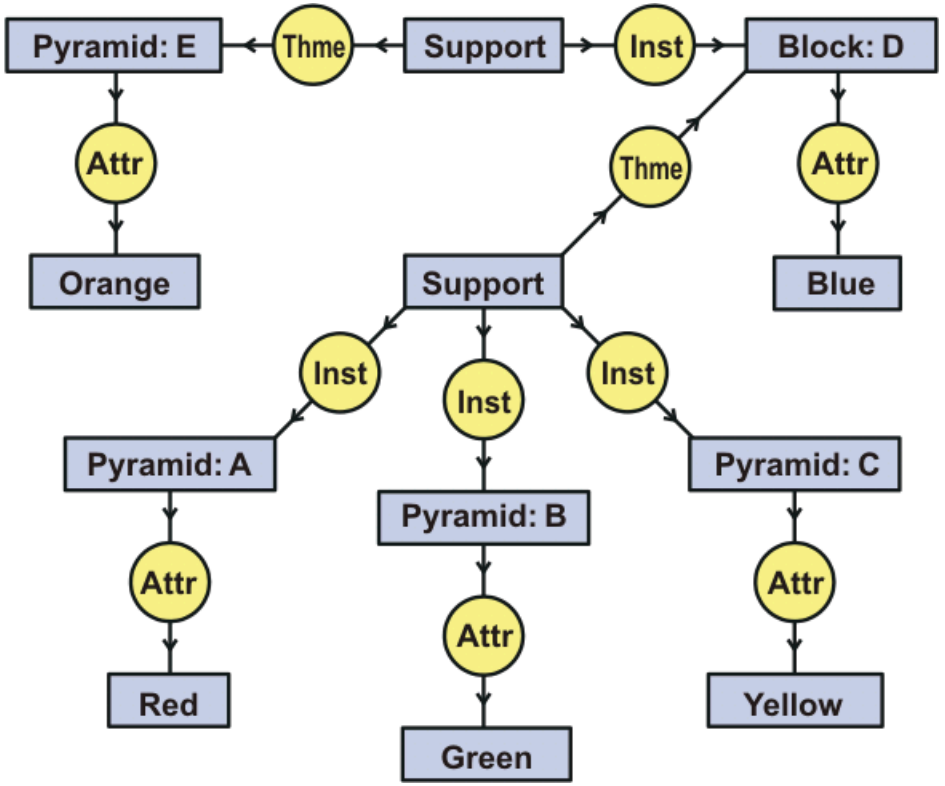


Even when people use the same language, they use different words and expressions.

How could humans or computers relate different descriptions to one another?

Mapping English to a Conceptual Graph



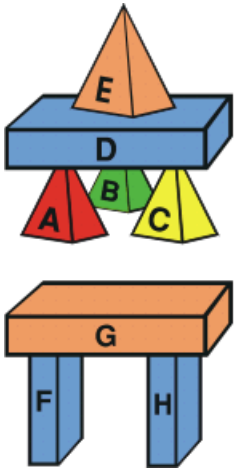


"A red pyramid A, a green pyramid B, and a yellow pyramid C support a blue block D, which supports an orange pyramid E."

The concepts (blue) are derived from English words, and the conceptual relations (yellow) from the case relations or thematic roles of linguistics.

Structured and Unstructured Representations

A description in tables of a relational database:



Objects

Entity	Shape	Color
A	pyramid	red
B	pyramid	green
C	pyramid	yellow
D	block	blue
E	pyramid	orange
F	block	blue
G	block	orange
H	block	blue

Supports

Supporter	Supportee
A	D
B	D
C	D
D	E
F	G
H	G

A description in English:

> **"A red pyramid A, a green pyramid B, and a yellow pyramid C support a blue block D, which supports an orange pyramid E."**

The database is called structured, and English is called unstructured.

Yet English has even more structure, but of a very different kind.

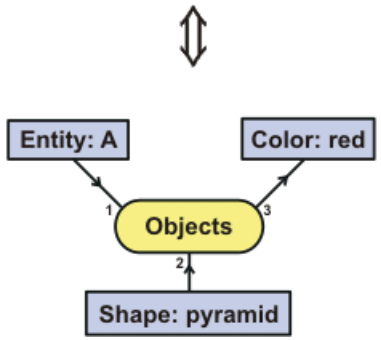
Mapping Database Relations to Conceptual Relations

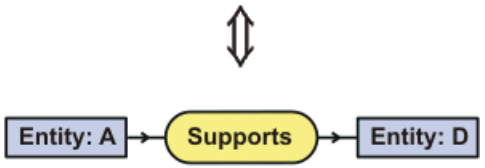
Objects

Entity	Shape	Color
A	pyramid	red
B	pyramid	green
C	pyramid	yellow
D	block	blue
E	pyramid	orange
F	block	blue
G	block	orange
H	block	blue

Supports

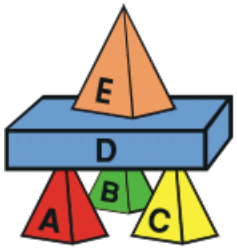
Supporter	Supportee
A	D
B	D
C	D
D	E
F	G
H	G

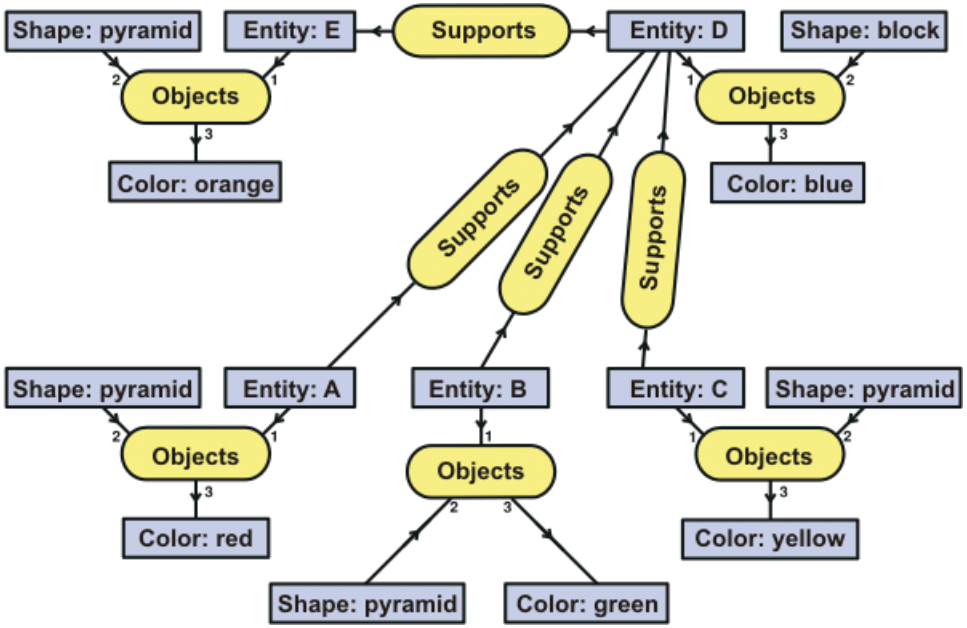




Each row of each table maps to one conceptual relation, which is linked to as many concepts as there are columns in the table.

Mapping an Entire Database to Conceptual Graphs

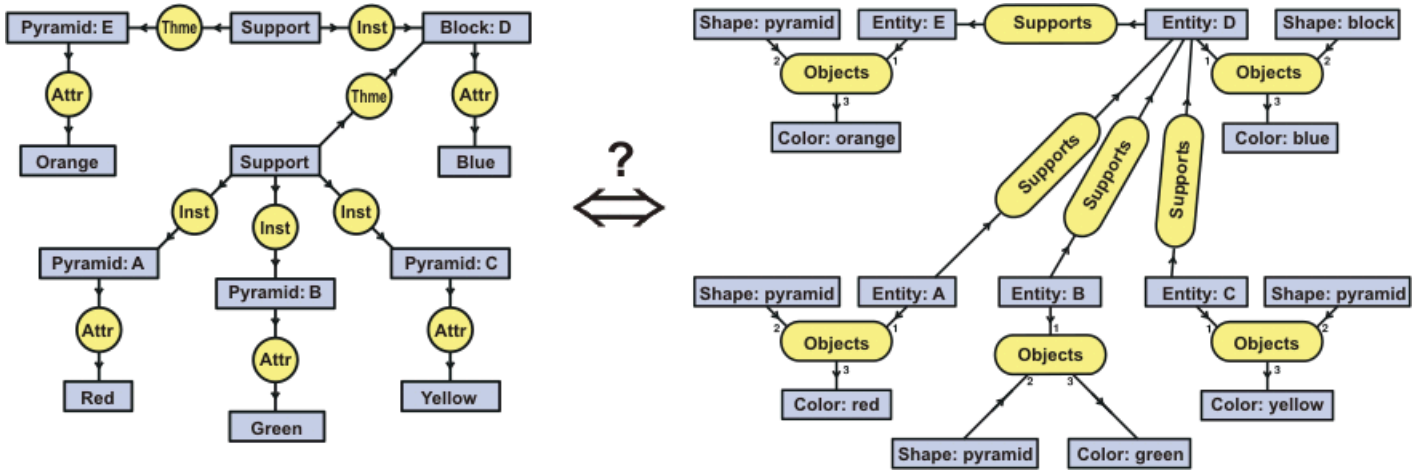




Join concept nodes that refer to the same entities.

Closely related entities are described by connected graphs.

Mapping the Two Graphs to One Another



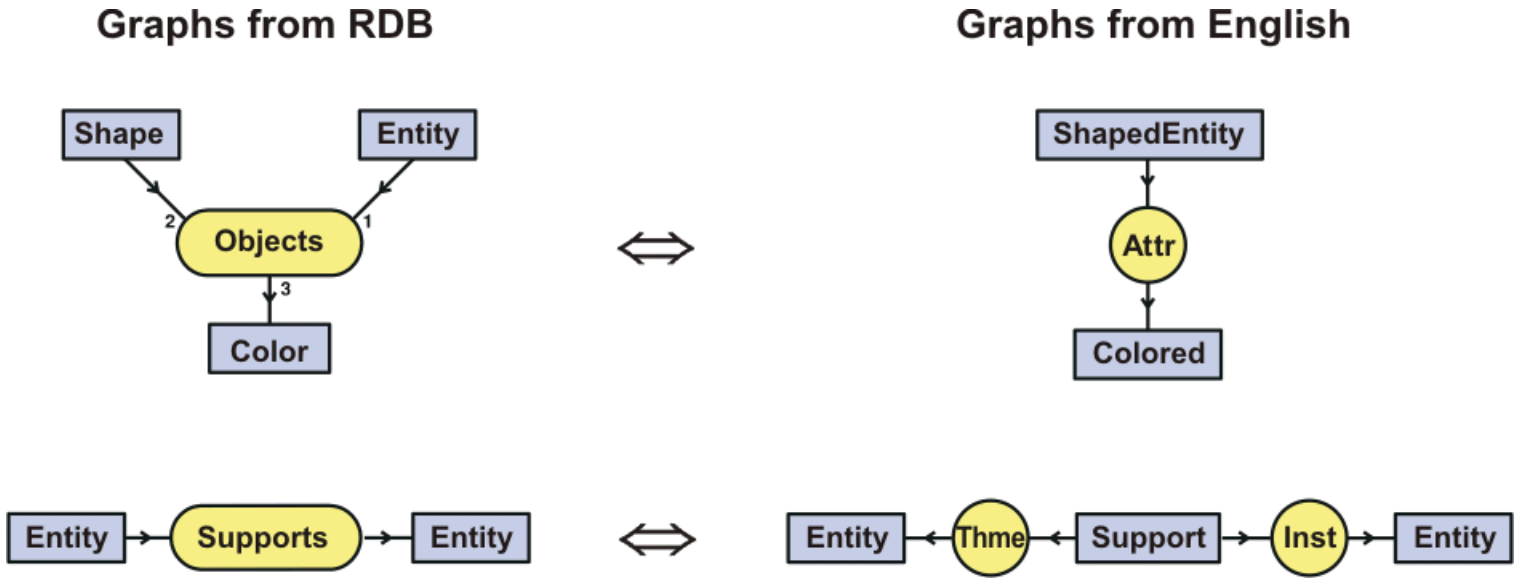
Very different structures: 12 concept nodes vs. 15 concept nodes, 11 relation nodes vs. 9 relation nodes, no similarity in type labels.

The only commonality is in the five names: A, B, C, D, E.

People can recognize the underlying similarities.

How is it possible for a computer to discover them?

Aligning Ontologies by Structure Mapping



Repeated application of these two transformations perform an exact mapping of all the nodes and arcs of each graph to the other.

This mapping, done by hand, is from an example by Sowa (2000), Ch 7.

The VivoMind Analogy Engine (VAE) found the mapping automatically.

Approximate Mapping

Example: ***How is a cat like a car?***

Cat	Car
Head	Hood
Eye	Headlight
Cornea	Glass plate
Mouth	Fuel cap
Stomach	Fuel tank
Bowel	Combustion chamber
Anus	Exhaust pipe
Skeleton	Chassis
Heart	Engine
Paw	Wheel
Fur	Paint

Metaphor and other aspects of language require approximations.

Approximations

Concepts are related by ontology or common associations:

- Eyes and headlights are related to light.
- Heart and engine are internal parts with a regular beat.
- Skeleton and chassis are structures for attaching parts.
- Paws and wheels support the body, and there are four of each.

One-to-one structure matching is preferred:

- Head → Eyes → Cornea.
- Hood → Headlights → Glass plate.

Approximate matches may skip some nodes (marked in red):

- Mouth → Esophagus → Stomach → Bowel → Anus.
- Fuel cap → Fuel tank → Combustion chamber → Muffler → Exhaust pipe.

Two factors determine the semantic distance between graphs:

- Ontology: How similar are the concept and relation types?
- Structure: How many nodes and arcs are added or deleted?

Computational Complexity

Research by Falkenhainer, Forbus, & Gentner:

- **Pioneers in finding analogies with their Structure Mapping Engine.**
- **Showed that SME algorithms take time proportional to N^3, where N is the number of frames (or graphs) in the knowledge base.**
- **MAC/FAC approach: Use a search engine to narrow down the number of likely candidates before using SME.**

VivoMind approach:

- **Encode graph structure and ontology in a Cognitive Signature™.**
- **For any graph, find closely matching signatures in log(N) time.**
- **Only graphs with similar signatures are likely candidates.**

For papers by Falkenhainer, Forbus, Genter, and colleagues,
http://www.qrg.northwestern.edu/papers/papers.html

Algorithms for Chemical Graphs

Graphs of organic molecules are similar to conceptual graphs:

- **Atoms ⇒ concept nodes labeled by the name of the element.**
- **Chemical bonds ⇒ relation nodes labeled by the name of the bond type.**
- **But conceptual graphs have many more types of concepts and relations.**

Chemical graphs inspired Peirce's existential graphs as representations of "the atoms and molecules of logic."

Some of the largest and most sophisticated systems for graph processing were developed by chemists, not computer scientists.

An early example was the use of chemical graph algorithms for building and searching hierarchies of conceptual graphs:

Robert A. Levinson, & Gerard Ellis (1992) Multilevel hierarchical retrieval, *Knowledge Based Systems* 5:3, pp. 233-244.

Finding Analogies

Find similar chemical graphs in logarithmic time:

- **Represent each graph by its unique International Chemical Identifier (InChI).**
- **Map the InChI codes to numeric vectors that encode both the graph structure and the labels of the atoms and bonds.**
- **Estimate the semantic distance between graphs by a measure based on both the graph structure and the labels on the nodes and arcs (atoms and bonds).**
- **Index the vectors by a locality-sensitive hashing (LSH) algorithm.**
- **Use the semantic distance measure to find the most similar graphs.**

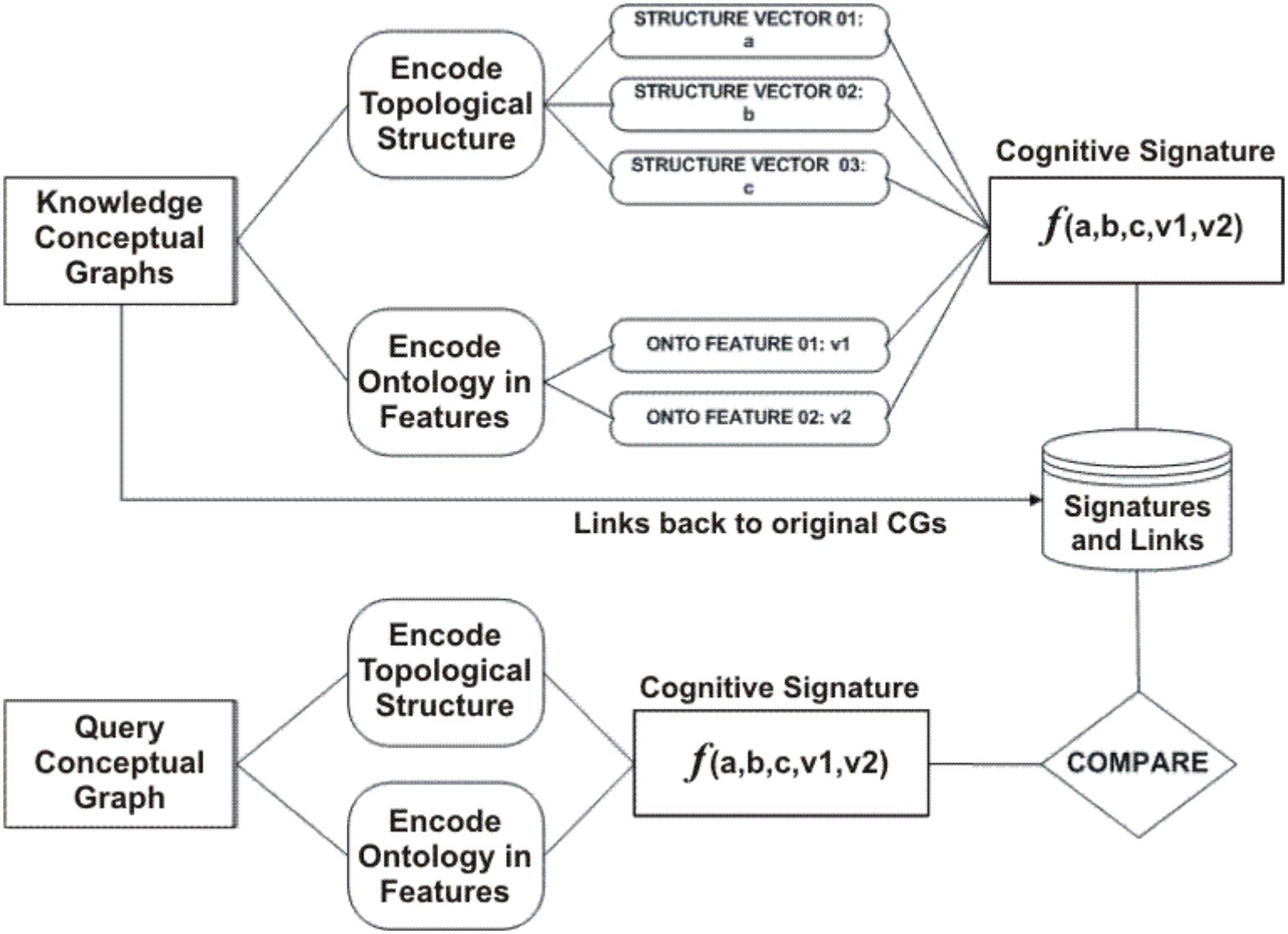
For details of the chemical algorithms, see

Mining patents using molecular similarity search, by James Rhodes, Stephen Boyer, Jeffrey Kreulen, Ying Chen, & Patricia Ordonez, http://psb.stanford.edu/psb-online/proceedings/psb07/rhodes.pdf

VivoMind algorithms use geometric algebras and Rvachev functions: US Patent 8566321, Relativistic concept measuring system.

For methods of pattern recognition with Rvachev functions, see Bougaev (2006): http://docs.lib.purdue.edu/dissertations/AAI3263546/

Cognitive Memory™



Basis for the VivoMind Analogy Engine (VAE)

Applications of VivoMind Software

General approach:

- VAE and Intellitex for analyzing English and other languages.
- Relate the customer's ontology (if any) to the lexical resources.
- Supplement the ontology with automated methods for deriving further information from NL documents. (No annotations required.)
- Analyze and relate any combination of NL documents and structured information from any sources.
- Translate the results to any notation or format the customer prefers.

Sample applications (see goal7.pdf for examples 2, 3, and 4):

1. Evaluate student answers in free-form English sentences.
2. Information extraction from research reports.
3. Legacy re-engineering: Analyze computer programs and relate them to the English documentation.
4. Extract information from textbooks and research reports about oil and gas fields, and answer English queries by a geologist.

Evaluating Student Answers

Multiple-choice questions are easy to evaluate by computer.

Long essays are often evaluated by statistical methods.

But short answers about mathematics are very hard to evaluate.

Sample question:

> ***The following numbers are 1 more than a square: 10, 37, 65, 82.***
> ***If you are given an integer N that is less than 200,***
> ***how would you determine whether N is 1 more than a square?***
> ***Explain your method in three or four sentences.***

An example of a correct answer:

> ***To show that N is 1 more than a square, show that N−1 is a square.***
> ***Find some integer x whose square is slightly less than N−1.***
> ***Compare N−1 to the squares of x, x+1, x+2, x+3, ...,***
> ***and stop when some square is equal to or greater than N−1.***
> ***If the last square is N−1, then N is one more than a square.***

Even experienced teachers must spend a lot of time checking and correcting such answers.

Publisher's Current Procedure

To evaluate new exam questions, the publisher normally gives the exam to a large number of students.

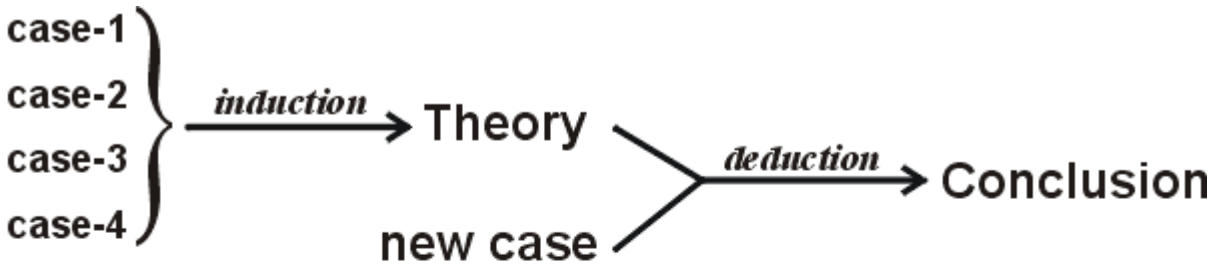
For each problem, they would get about 50 different answers:

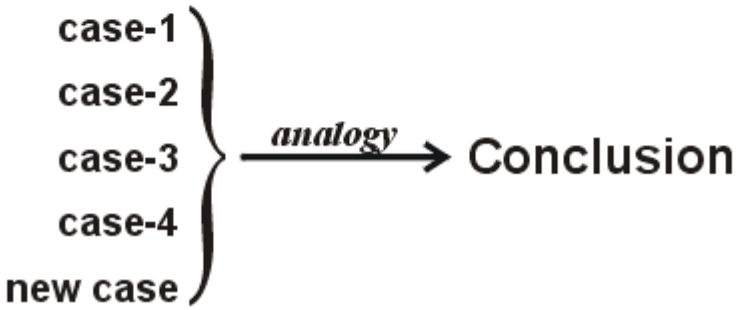
- **Some are completely correct**
  — but stated in different ways.
- **Some are partially correct**
  — and the teacher says what is missing.
- **Others are wrong**
  — in many different ways.

Result: 50 pairs of student answer and teacher's response.

Each answer-response pair is a case for case-based reasoning.

Case-Based Reasoning





Given the same cases, analogy takes one step to derive an answer, but induction and deduction take multiple steps.

Analogy is fast and flexible, but a theory is important if it can be used and reused in many applications.

Using Intellitex and VAE

Translate all answers to conceptual graphs (CGs):

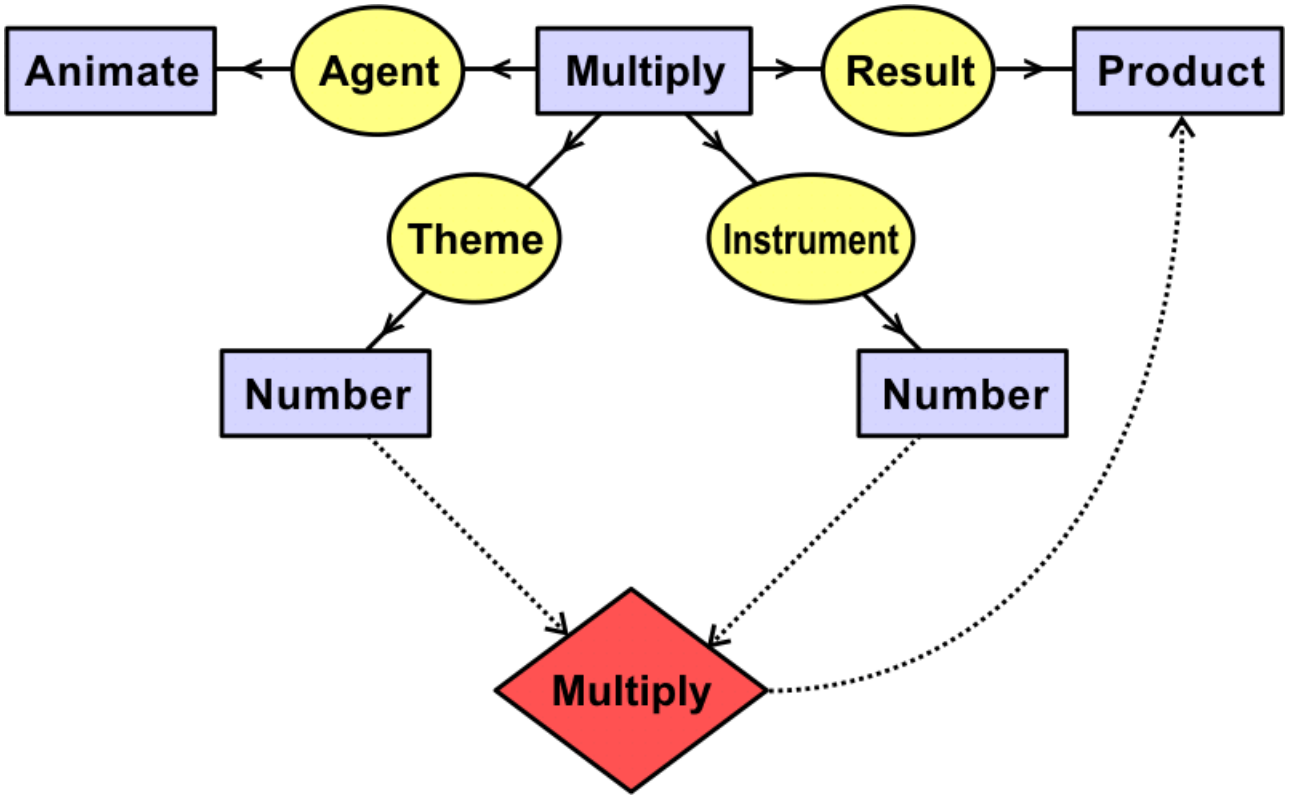
- Intellitex uses a link grammar and a variety of semantic knowledge.
- Whenever a new CG is derived, it is stored in Cognitive Memory.
- With high-speed search, any CG can be used to analyze any sentence.
- The next slide shows a formally defined CG for Multiply.

VAE compares each new answer to the 50 cases:

- CGs for the answers of all 50 cases are stored in Cognitive Memory.
- Compare the CG for each new answer to the CGs in Cognitive Memory.
- If there is a good match, print out the teacher's previous response.
- Otherwise, send the new student answer to some teacher to evaluate.
- Add the new answer-response pair to the collection of cases.

VAE supports both formal and informal reasoning.

Conceptual Graph for Multiply



This CG represents a pattern or schema for the concept Multiply:

[*Someone*] *multiplies a number by a number to get a product.*

The diamond node, called an actor, represents a function that multiplies two numbers to compute their product.

Using Syntax and Semantics

Intellitex uses Cognitive Memory to find conceptual graphs that approximately match the input phrases and sentences.

Any CG can be used to supplement the syntax:

- **Intellitex uses a link grammar to generate dependency graphs.**
- **Those graphs can be mapped to CGs, but the mapping is rarely one-to-one, and Intellitex usually requires more information.**
- **CGs from any source can be used to check constraints, provide default information, resolve ambiguities, or suggest alternatives.**

Many kinds of CGs can be used during the analysis:

- **IBM-CSLI verb ontology* for case roles and type constraints.**
- **CGs for an ontology of arithmetic (see the previous slide).**
- **CGs automatically derived from a textbook for the course.**
- **CGs automatically derived from answers written by the teachers.**

*** See http://lingo.stanford.edu/vso/**

Results

VAE found a good match for most student answers.

The answers were often poorly written and not precise enough for a formal parser and theorem prover.

But the stored CGs made the parsing more robust —

- **By resolving ambiguities and showing expected combinations,**
- **By correcting and compensating for errors in syntax,**
- **By providing defaults for missing arguments, and**
- **By relating multiple fragments to form complete sentences.**

Approximate matches within a reasonable semantic distance were adequate for evaluating student answers.

For matches outside the limits, student answers could be sent to a teacher, who would write a new evaluation.

IBM Watson System

Beat human experts in answering Jeopardy! questions.

Uses analogies to find relevant background knowledge:

- **Watson and VAE implement a version of structure mapping similar to the methods of Falkenhainer, Forbus, and Gentner. ***
- **Both systems use analogies and case-based reasoning to relate natural language to a large volume of unstructured information.**
- **They also use structured information stored in databases, knowledge bases, and the Semantic Web.**
- **But Watson does not have a method for finding analogies in logarithmic time.**
- **Instead, it uses a supercomputer with 2880 CPUs.**

*** J. William Murdock (2011) Structure mapping for Jeopardy! clues. *Proceedings of the 19th International Conference on Case Based Reasoning* (ICCBR'11), London.**
http://bill.murdocks.org/iccbr2011murdock_web.pdf

Role of Analogy

The basis for human reasoning and language understanding.

Logic is a disciplined special case of analogical reasoning:

- **Essential for precise reasoning in mathematics and science.**
- **Important for precision in any field.**
- **But even in science and engineering, analogy is necessary for knowledge discovery and innovation.**

Conceptual graphs support logical and analogical methods:

- **They are defined by the ISO/IEC standard 24707 for Common Logic.**
- **But they also support semantic distance measures for analogy.**
- **They provide a bridge between informal language and formal logic.**

CGs derived from English can be used for analogies.

But CGs used for formal logic should be derived from formal languages or be corrected by comparison to formal CGs.

What is Language Understanding?

Understanding a text in some language does not require a translation to a language of thought or logical form.

Instead, it requires an interpreter, human or robot, to relate the text to his, her, or its context, knowledge, and goals:

- **That process changes the interpreter's background knowledge.**
- **But the kind of change depends critically on the context, goals, and available knowledge.**
- **No two interpreters understand a text in exactly the same way.**
- **With different contexts, goals, or knowledge, the same interpreter may understand a text in different ways.**

The evidence of understanding is an appropriate response to a text by an interpreter in a given situation.

If a robot responds appropriately to a command, does it understand? What if it explains how and why it responded?

Related Readings

Future directions for semantic systems,
http://www.jfsowa.com/pubs/futures.pdf

From existential graphs to conceptual graphs,
http://www.jfsowa.com/pubs/eg2cg.pdf

Role of Logic and Ontology in Language and Reasoning,
http://www.jfsowa.com/pubs/rolelog.pdf

Fads and Fallacies About Logic,
http://www.jfsowa.com/pubs/fflogic.pdf

Conceptual Graphs for Representing Conceptual Structures,
http://www.jfsowa.com/pubs/cg4cs.pdf

Peirce's tutorial on existential graphs,
http://www.jfsowa.com/pubs/egtut.pdf

ISO/IEC standard 24707 for Common Logic,
http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007(E).zip

References

For more information about the VivoMind software:

Majumdar, Arun K., John F. Sowa, & John Stewart (2008) Pursuing the goal of language understanding, http://www.jfsowa.com/pubs/pursuing.pdf

Majumdar, Arun K., & John F. Sowa (2009) Two paradigms are better than one and multiple paradigms are even better, http://www.jfsowa.com/pubs/paradigm.pdf

Sowa, John F. (2002) Architectures for intelligent systems, http://www.jfsowa.com/pubs/arch.htm

Sowa, John F., & Arun K. Majumdar (2003) Analogical reasoning, http://www.jfsowa.com/pubs/analog.htm

Sowa, John F. (2003) Laws, facts, and contexts, http://www.jfsowa.com/pubs/laws.htm

Sowa, John F. (2005) The challenge of knowledge soup, http://www.jfsowa.com/pubs/challenge.pdf

Sowa, John F. (2006) Worlds, models, and descriptions, http://www.jfsowa.com/pubs/worlds.pdf

Sowa, John F. (2011) Cognitive architectures for conceptual structures, http://www.jfsowa.com/pubs/ca4cs.pdf

Related references:

Johnson-Laird, Philip N. (2002) Peirce, logic diagrams, and the elementary processes of reasoning, *Thinking and Reasoning* 8:2, 69-95. http://mentalmodels.princeton.edu/papers/2002peirce.pdf

Lamb, Sydney M. (2011) Neurolinguistics, Class Notes for Linguistics 411, Rice University. http://www.owlnet.rice.edu/~ling411

Harrison, Colin James (2000) *PureNet: A modeling program for neurocognitive linguistics*, http://scholarship.rice.edu/bitstream/handle/1911/19501/9969261.PDF

For other references, see the combined bibliography for this site:

http://www.jfsowa.com/bib.htm