Pursuing the Goal Of Language Understanding

John F. Sowa and Arun K. Majumdar
VivoMind Research, LLC

18 March 2009

How can a computer understand language?

According to Alan Turing,

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 is based on analogies.

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

Research project:

How can we use an analogy engine to make computers more human-like?

Describing Things in Different Ways

How can we describe what we see?

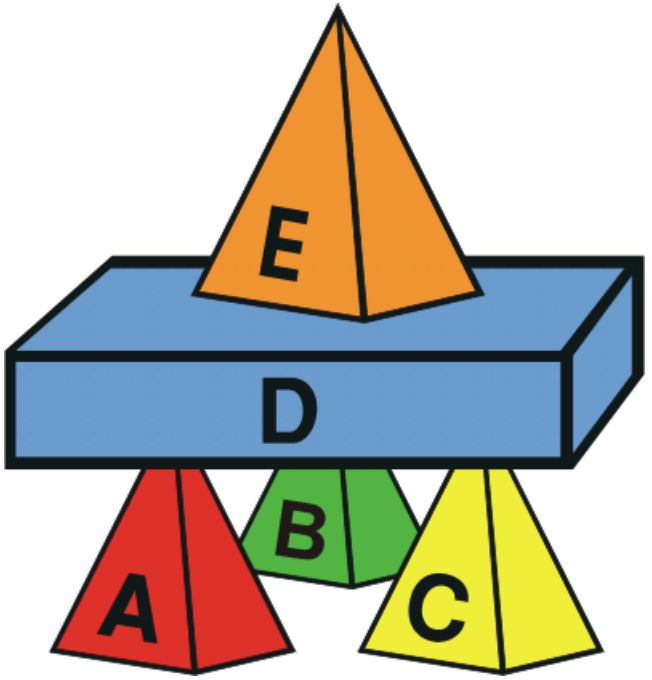
In ordinary language?

In some version of logic?

In a relational database?

In the Semantic Web?

In a programming language?

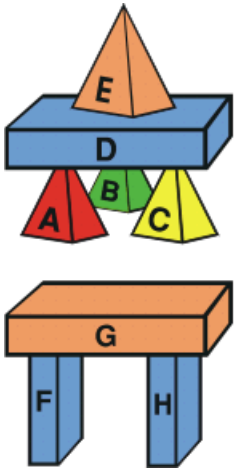


Even when people use the same language, they use different words and expressions.

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

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

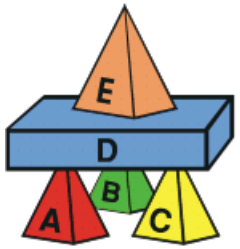
A description in English:

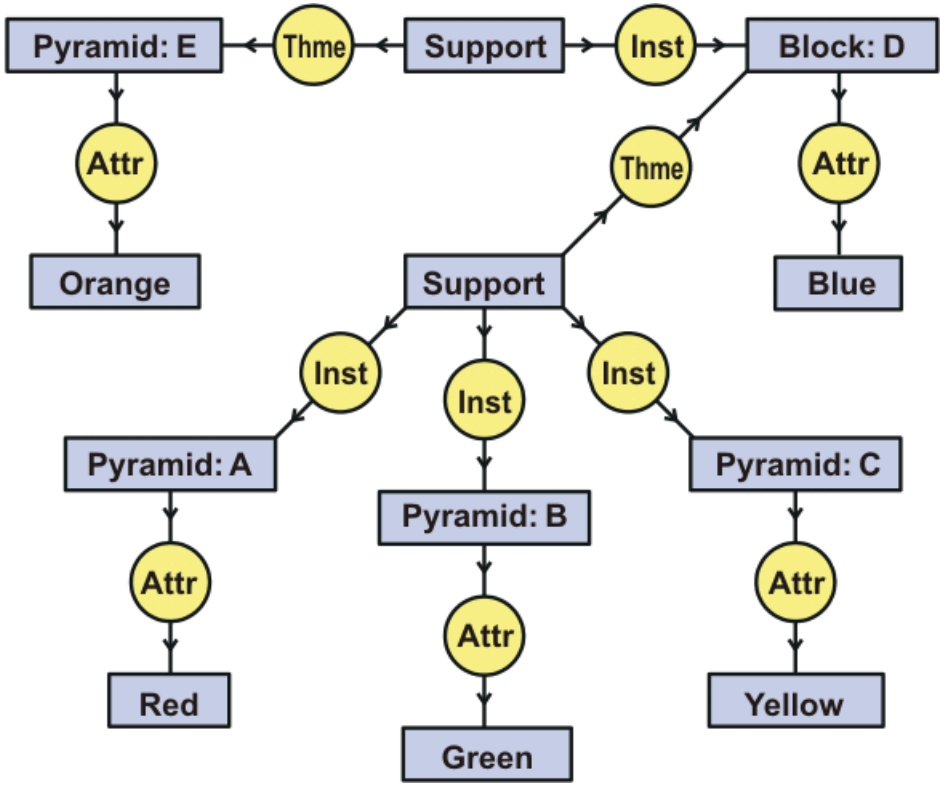
> **"A red pyramid A, a green pyramid B, and a yellow pyramid C support a blue block D, which supports an orange pyramid E."**

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

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

Mapping 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.

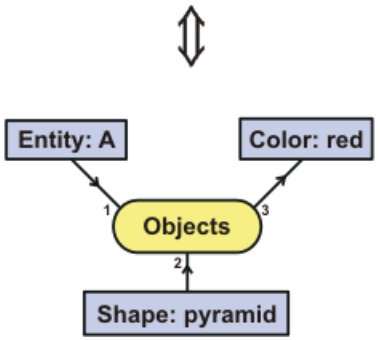
Mapping Database Relations to Conceptual Relations

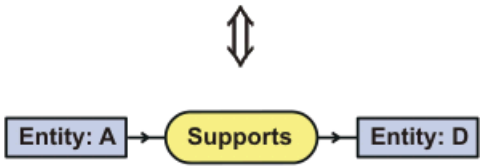
Objects

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

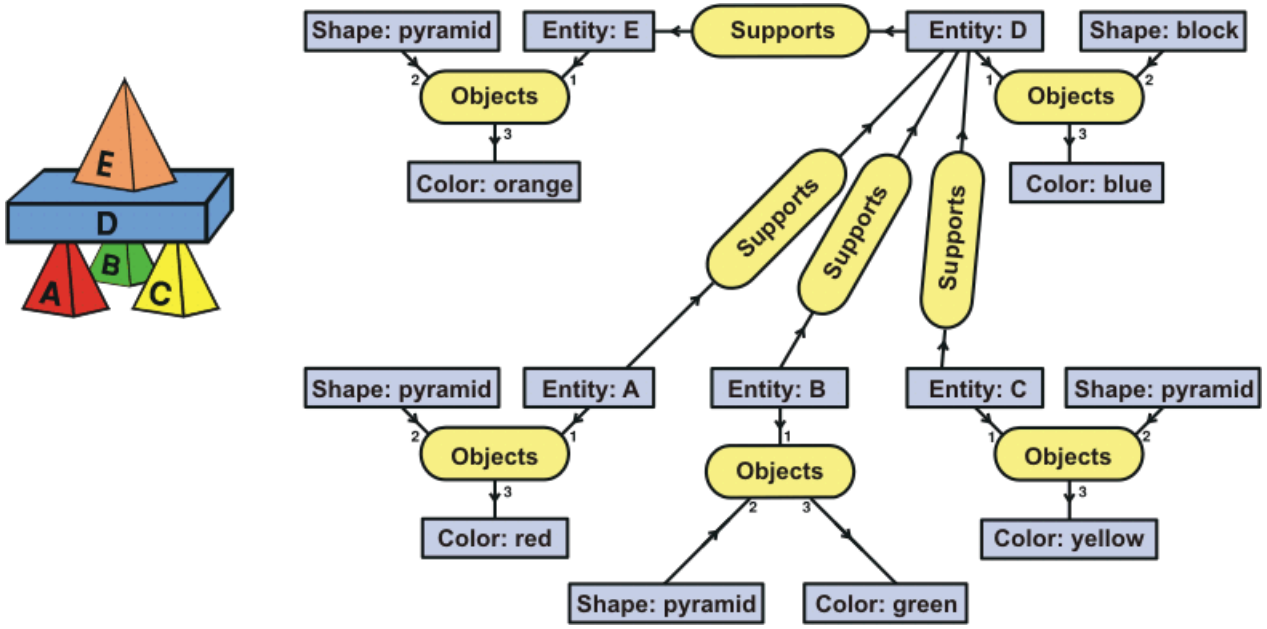
Supporter	Supportee
A	D
B	D
C	D
D	E
F	G
H	G





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

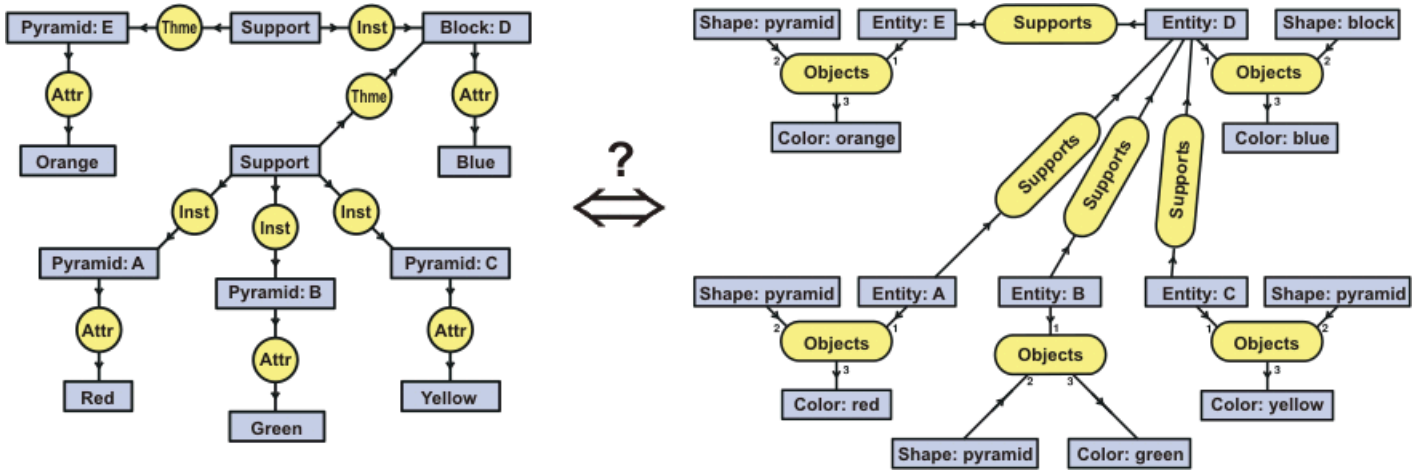
Mapping an Entire Database to Conceptual Graphs



Join concept nodes that refer to the same entities.

Closely related entities are described by connected graphs.

Mapping the Two Graphs to One Another



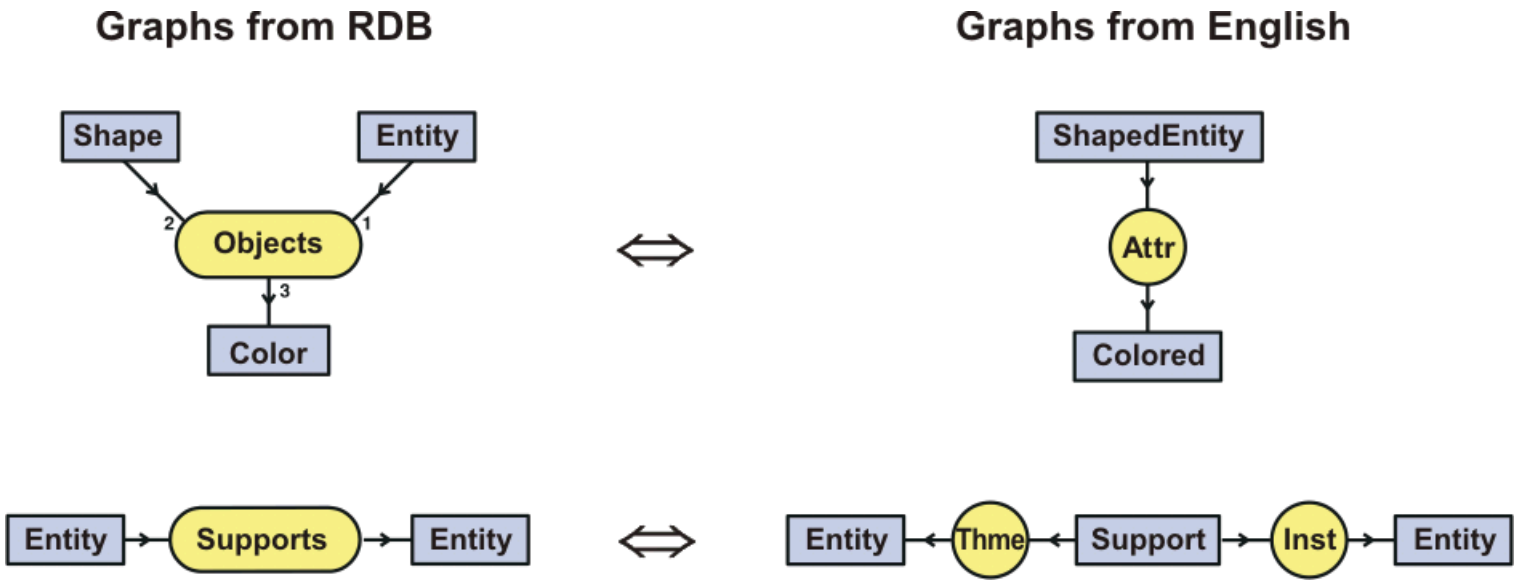
Very different ontologies: 12 concept nodes vs. 15 concept nodes, 11 relation nodes vs. 9 relation nodes, no similarity in type labels.

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?

Mapping the Graphs by Aligning the Ontologies



Repeated application of these two transformations completely map all 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.

Four Views of Analogy

1. **By logicians:**

   Deduction is reasoning from "first principles."

2. **By psychologists:**

   Analogy is a fundamental mechanism of human and animal cognition.

   All aspects of language understanding depend on analogy.

3. **Theoretical:**

   Analogy is a general pattern-matching mechanism, and all methods of formal logic — deduction, induction, and abduction — are special cases.

4. **Computational:**

   A powerful and flexible technique that can have important applications in reasoning, learning, and language processing.

   But practicality depends on finding analogies efficiently.

Computational Complexity

Research by Falkenhainer, Forbus, & Gentner:

Pioneers in finding analogies with their Structure Mapping Engine.

Demonstrated that the SME algorithms take time proportional to N-cubed, where N is the number of 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™.

Find the closest matching signatures in logarithmic time.

Use structure mapping only on a very small number of graphs.

Algorithms for Chemical Graphs

Graphs of organic molecules can be represented as 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 important application 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.

Chemical Graph Search Engine

Techniques similar to the methods for searching conceptual graphs:

- 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.
- Index the vectors by a locality-sensitive hashing (LSH) algorithm.
- Estimate the semantic distance between graphs by a measure of intermolecular similarity.
- Use the semantic distance measure to find the most similar graphs.

For a description of these 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

For indexing and searching over 4 million chemical graphs, see

https://chemsearch.almaden.ibm.com/chemsearch/SearchServlet

Three Applications for an Analogy Engine

1. **Educational Software**

   Evaluating student answers written in free-form English.

2. **Legacy Re-engineering**

   Comparing structured data to English documentation, finding which sentences describe which data, and detecting errors and inconsistencies between the data and the sentences.

3. **Oil and gas exploration**

   Reading English documents about oil and gas exploration and answering English questions written 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.***

How could a computer system evaluate such answers?

Determine whether they are correct, incorrect, or partially correct?

And make helpful suggestions about the incorrect answers?

Many Possible Answers

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.

How can a computer system evaluate them?

How can it make helpful suggestions for incorrect answers?

Publisher's Current Procedure

To evaluate new exam questions, the publisher normally gave 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.

Three Teams Addressed this Problem

Team #1 used a large deductive knowledge base.

- **Ontology for mathematical word problems.**
- **English parser to analyze student answers.**
- **Theorem prover to determine if the answers are correct.**

Team #2 used Latent Semantic Analysis (LSA).

- **Measure the similarity of student answers to correct answers.**

VivoMind team used case-based reasoning.

- **Intellitex to translate student answers to conceptual graphs.**
- **VAE to compare student CGs to correct and incorrect CGs.**

Results for Team #1

Extending the ontology for each problem type was too difficult for most high-school teachers.

Student answers had too many fragmentary and ungrammatical sentences to be parsed correctly.

The theorem prover could not reach a conclusion for most of the student answers.

The method was considered impractical.

Results for Team #2

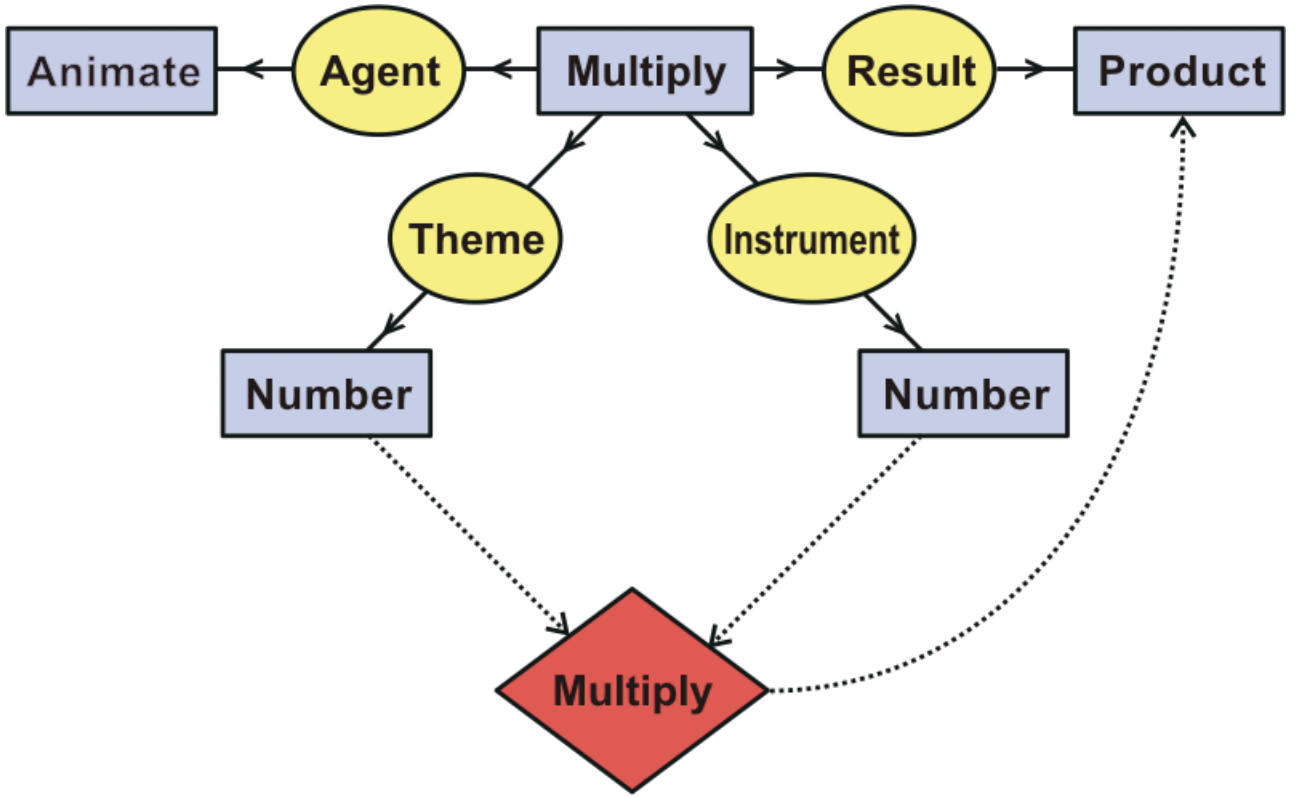
LSA methods often produce good results for measuring the similarity of texts that are longer than a single paragraph.

They are less reliable on texts of just a few sentences.

They cannot distinguish texts that interchange words or insert an extra word, such as *not*.

They were unreliable for distinguishing correct answers from incorrect answers.

Conceptual Graphs Relate English to Math

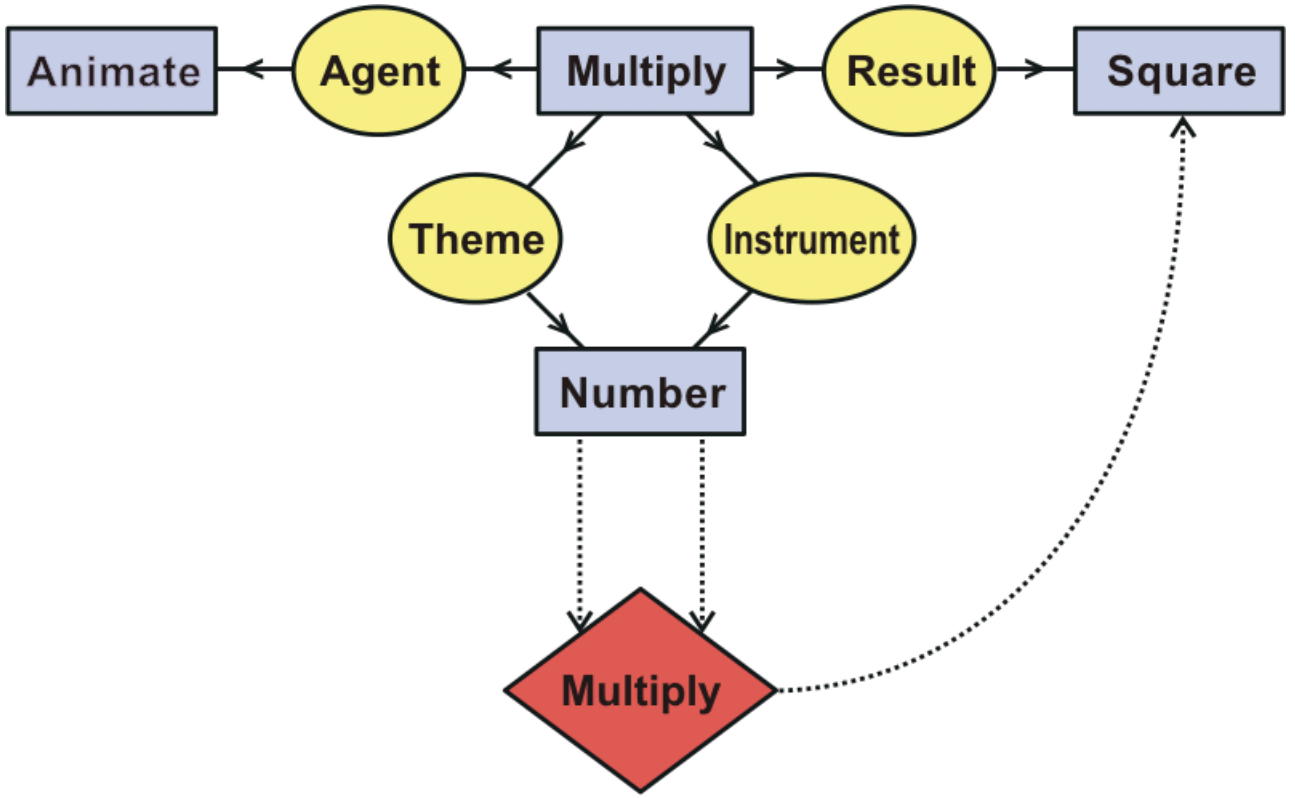


The boxes and circles represent an English sentence pattern:

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

The diamond node, called an actor, represents a function that computes the result of multiplying values inside the concept boxes.

Definition of the Word *square*



This conceptual graph is a specialization of the previous CG for the case when both inputs to the multiply actor are the same number.

This graph expresses the context that distinguishes the numerical square from the geometrical square.

Intellitex Parser

VivoMind developed a parser named Intellitex:

- **Uses a rather simple grammar.**
- **Depends on analogies for interpreting sentences.**
- **Generates conceptual graphs as output.**
- **Robust: always generates some CG as its best guess.**

These properties are important for analyzing student answers, which frequently have poor grammar and incomplete sentences.

Minor errors are not necessarily bad — provided that Intellitex makes the same errors consistently in all cases.

Using VAE to Evaluate Student Answers

VAE compares each new answer to the 50 cases:

1. For all 50 cases, translate student answer to conceptual graphs.
2. Translate each new answer to a new CG.
3. Compare the new CG to the 50 CGs for previous answers.
4. Use the semantic distance measure to determine the best match.
5. If there is a good match, print out the corresponding response.
6. Otherwise, send the new student answer to a teacher to evaluate.

Results:

- VAE found a good match for nearly all of the student answers.
- For each good match, the previous teacher's response was appropriate.
- When VAE failed to find a good match, the new case could be added to the list of cases in order to improve its coverage.
- No need for teachers to use any language other than English.

Problem for Legacy Re-engineering

Analyze the software and documentation of a large corporation.

Programs in daily use, some of which were up to 40 years old.

- **1.5 million lines of COBOL programs.**
- **100 megabytes of English documentation — reports, manuals, e-mails, Lotus Notes, HTML, and program comments.**

Goal:

- **Analyze the COBOL programs.**
- **Analyze the English documentation.**
- **Compare the two to generate**

  English glossary of all terms with index to the software.

  Structure diagrams of the programs, files, and data.

  List of discrepancies between the programs and documentation.

An Important Simplification

An extremely difficult, still unsolved problem:

- Translate English specifications to executable programs.

Much easier task:

- Translate the COBOL programs to conceptual graphs.
- Use the conceptual graphs from COBOL to interpret the English.
- Use the analogy engine to compare the graphs derived from COBOL to the graphs derived from English.
- Record the similarities and discrepancies.

Excerpt from the Documentation

The input file that is used to create this piece of the Billing Interface for the General Ledger is an extract from the 61 byte file that is created by the COBOL program BILLCRUA in the Billing History production run. This file is used instead of the history file for time efficiency. This file contains the billing transaction codes (types of records) that are to be interfaced to General Ledger for the given month.

For this process the following transaction codes are used: 32 — loss on unbilled, 72 — gain on uncollected, and 85 — loss on uncollected. Any of these records that are actually taxes are bypassed. Only client types 01 — Mar, 05 — Internal Non/Billable, 06 — Internal Billable, and 08 — BAS are selected. This is determined by a GETBDATA call to the client file.

Note that none of the files or COBOL variables are named.

By matching the English graphs to the COBOL graphs, VAE identified all the file names and COBOL variables involved.

Interpreting Novel Patterns

Many texts contain unusual or ungrammatical patterns.

They may be elliptical forms that could be stored in tables.

But some authors write them as phrases in a sentence:

- ***32 — loss on unbilled***
- ***72 — gain on uncollected***
- ***85 — loss on uncollected***

Intellitex generated a CG with a default relation (Link):

[Number: 32]→(Link)→[Punctuation: "–"]→(Link)→[Loss]→(On)→[Unbilled]

The value 32 was stored as a constant in a COBOL program.

The phrase "loss on unbilled" was written as a comment.

The value and the comment from COBOL were translated to a CG that was the closest match to the CG derived from the text.

Results

Job finished in 8 weeks by two programmers, Arun Majumdar and André LeClerc.

- **Four weeks for customization:**

  Design, ontology, and additional programming for I/O formats.

- **Three weeks to run Intellitex + VAE + extensions:**

  VAE handled matches with strong evidence (close semantic distance).

  Matches with weak evidence were confirmed or corrected by Majumdar and LeClerc.

- **One week to produce a CD-ROM with integrated views of the results:**

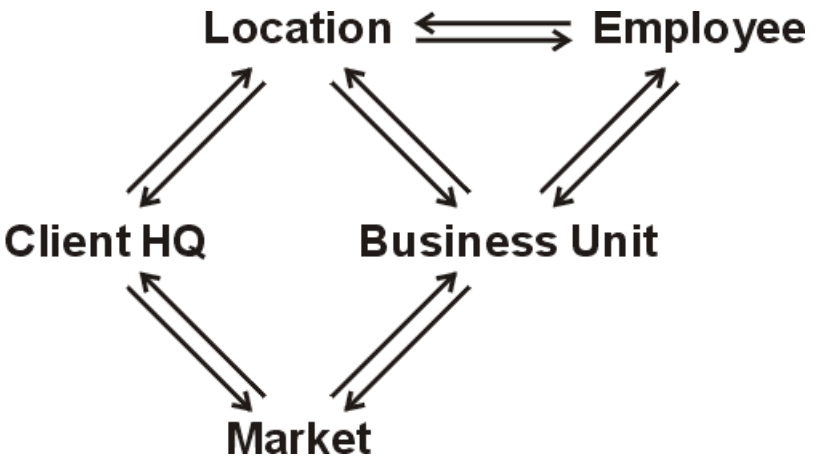
  Glossary, data dictionary, data flow diagrams, process architecture, system context diagrams.

A major consulting firm had estimated that the job would take 40 people two years to analyze the documentation and generate the cross references.

With VivoMind software, it was completed in 15 person weeks.

Mismatch Found by VAE

A diagram of relationships among data types in the database:



Question: Which location determines the market?

According to the documentation: Business unit.

According to the COBOL programs: Client HQ.

Management had been making decisions based on incorrect assumptions.

Contradiction Found by VAE

From the ontology used for interpreting English:

- **Every employee is a human being.**
- **No human being is a computer.**

From analyzing COBOL programs:

- **Some employees are computers.**

What is the reason for this contradiction?

Quick Patch in 1979

A COBOL programmer made a quick patch:

- Two computers were used to assist human consultants.
- But there was no provision to bill for computer time.
- Therefore, the programmer named the computers Bob and Sally, and assigned them employee ids.

For more than 20 years:

- Bob and Sally were issued payroll checks.
- But they never cashed them.

VAE discovered the two computer “employees.”

Processing Documents with VLP

VLP (VivoMind Language Processor) is a successor to Intellitex:

- Translate documents to conceptual graphs.
- Map structured data from SQL, RDF, etc., to conceptual graphs.
- Index the CGs in time proportional to (N log N), where N is the total number of nodes in all the graphs.

For a query stated as a paragraph in ordinary language,

- Translate the query to conceptual graphs.
- Find matching patterns in the source data and rank them in order of semantic distance. (Zero distance means an exact match.)
- For each match within a given threshold, determine which subgraphs are similar or different.
- As answer, return the English phrases in the original documents from which those graphs and subgraphs were derived.

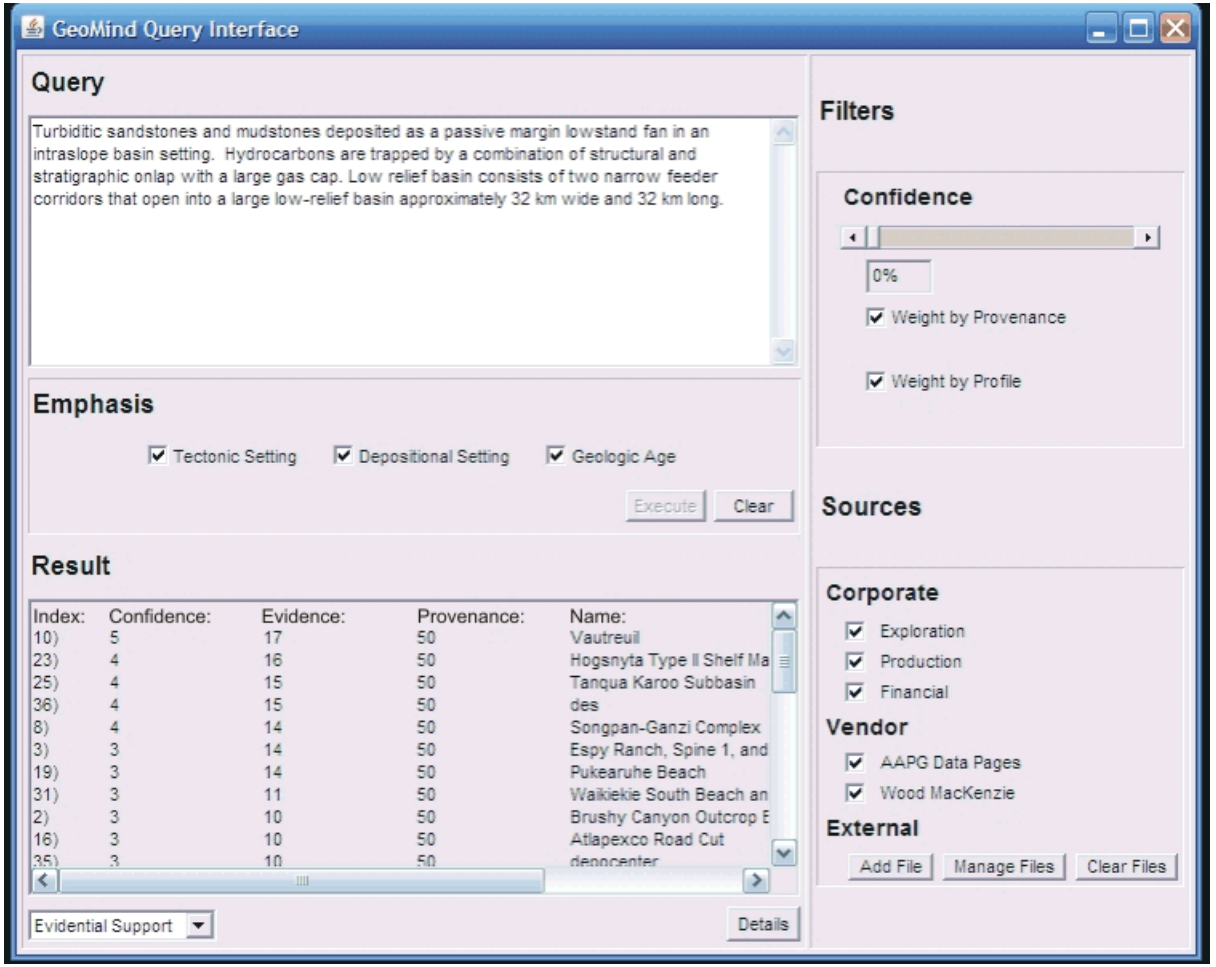
Application to Oil and Gas Exploration

Source material:

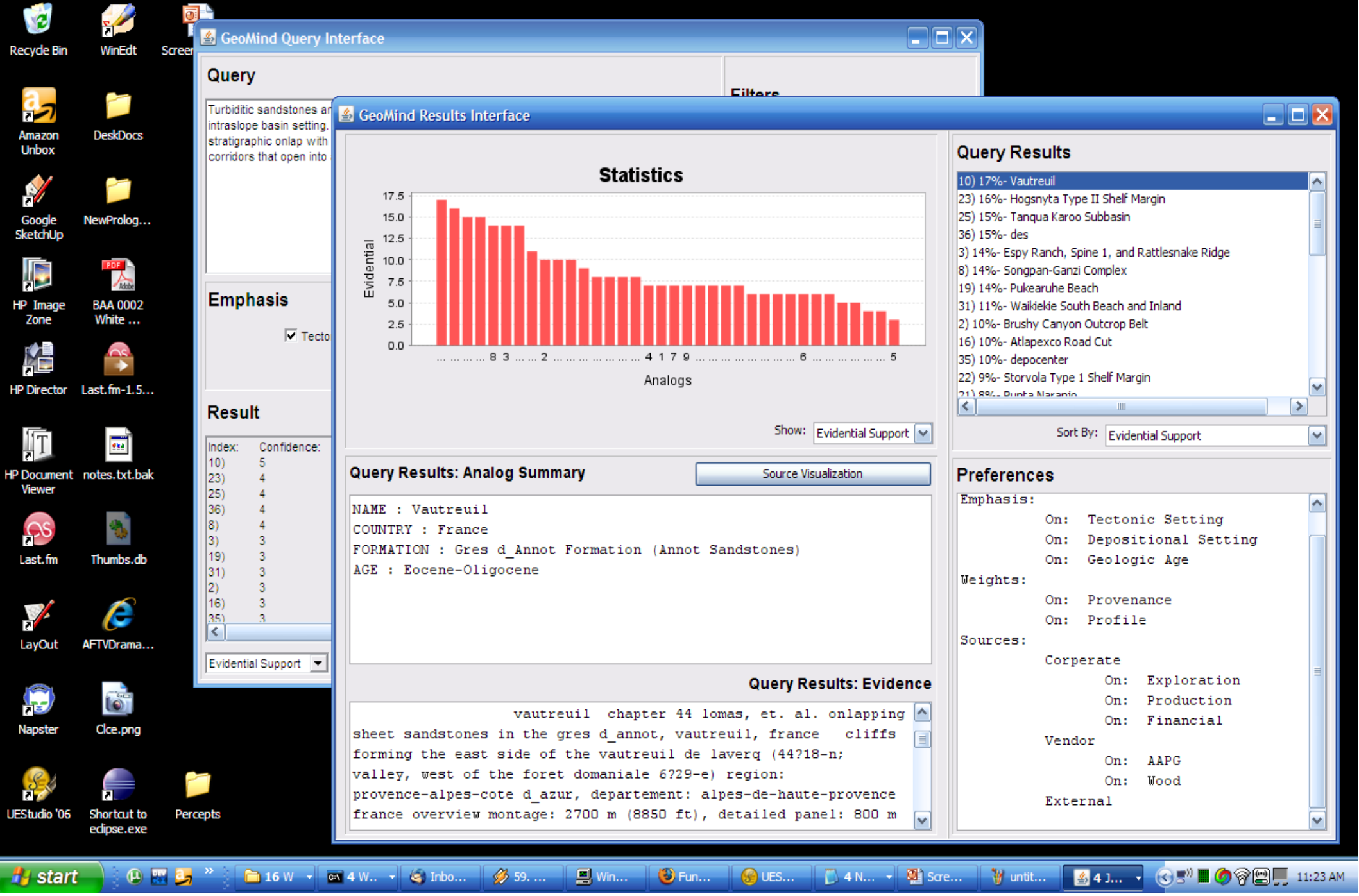
- 79 documents, ranging in length from 1 page to 50 pages.
- English, as written for human readers (no semantic tagging).
- Additional data from relational DBs and other structured sources.
- Lexical resources derived from WordNet, CoreLex, IBM-CSLI Verb Ontology, Roget's Thesaurus, and other sources.
- An ontology for the oil and gas domain developed in collaboration with EGI (Energy & Geoscience Institute at the University of Utah).

Queries:

- One or more sentences that describe a potential oil or gas field.
- Analogies compare the query to fields described in the documents.



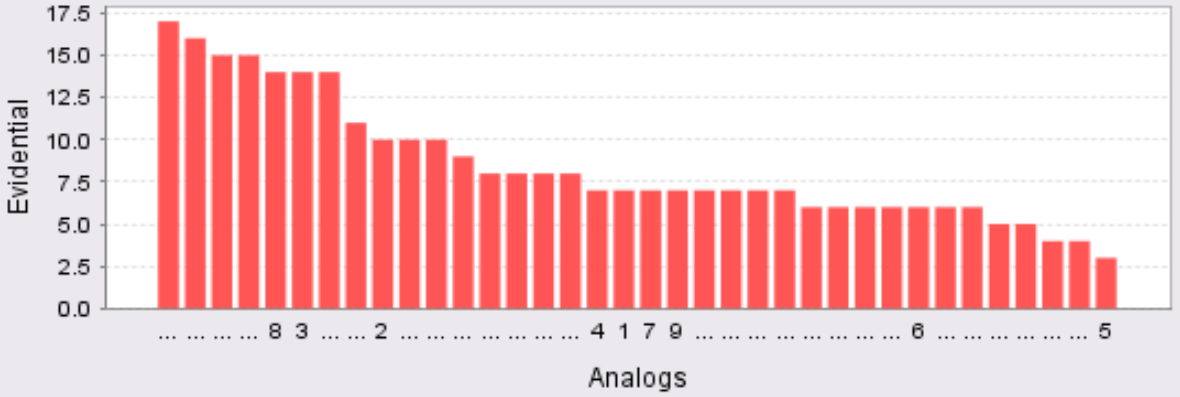
RESULTS, ranked by evidence (Dempster-Shafer) & confidence factors



After clicking the “Details” button on the previous window

GeoMind Results Interface

Statistics



Show: Evidential Support

Query Results

10) 17%- Vautreuil
23) 16%- Hogsnyta Type II Shelf Margin
25) 15%- Tanqua Karoo Subbasin
36) 15%- des
3) 14%- Espy Ranch, Spine 1, and Rattlesnake Ridge
8) 14%- Songpan-Ganzi Complex
19) 14%- Pukearuhe Beach
31) 11%- Waikiekie South Beach and Inland
2) 10%- Brushy Canyon Outcrop Belt
16) 10%- Atlapexco Road Cut
35) 10%- depocenter
22) 9%- Storvola Type 1 Shelf Margin
21) 8%- Punta Naranio

Sort By: Evidential Support

Query Results: Analog Summary

Source Visualization

```
NAME : Vautreuil
COUNTRY : France
FORMATION : Gres d_Annot Formation (Annot Sandstones)
AGE : Eocene-Oligocene
```

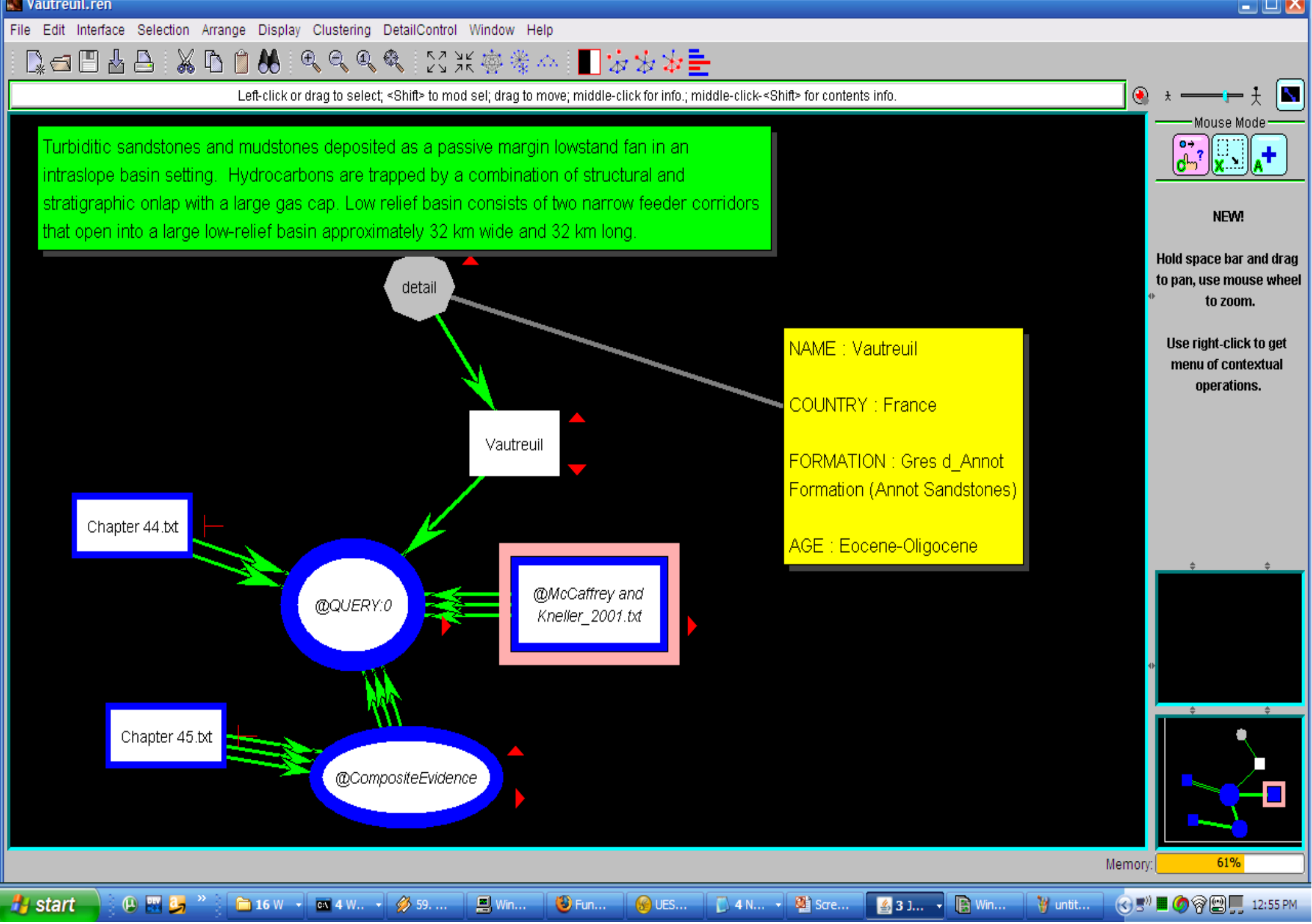
Query Results: Evidence

```
                  vautreuil  chapter 44 lomas, et. al. onlapping
sheet sandstones in the gres d_annot, vautreuil, france   cliffs
forming the east side of the vautreuil de laverq (44?18-n;
valley, west of the foret domaniale 6?29-e) region:
provence-alpes-cote d_azur, departement: alpes-de-haute-provence
france overview montage: 2700 m (8850 ft), detailed panel: 800 m
```

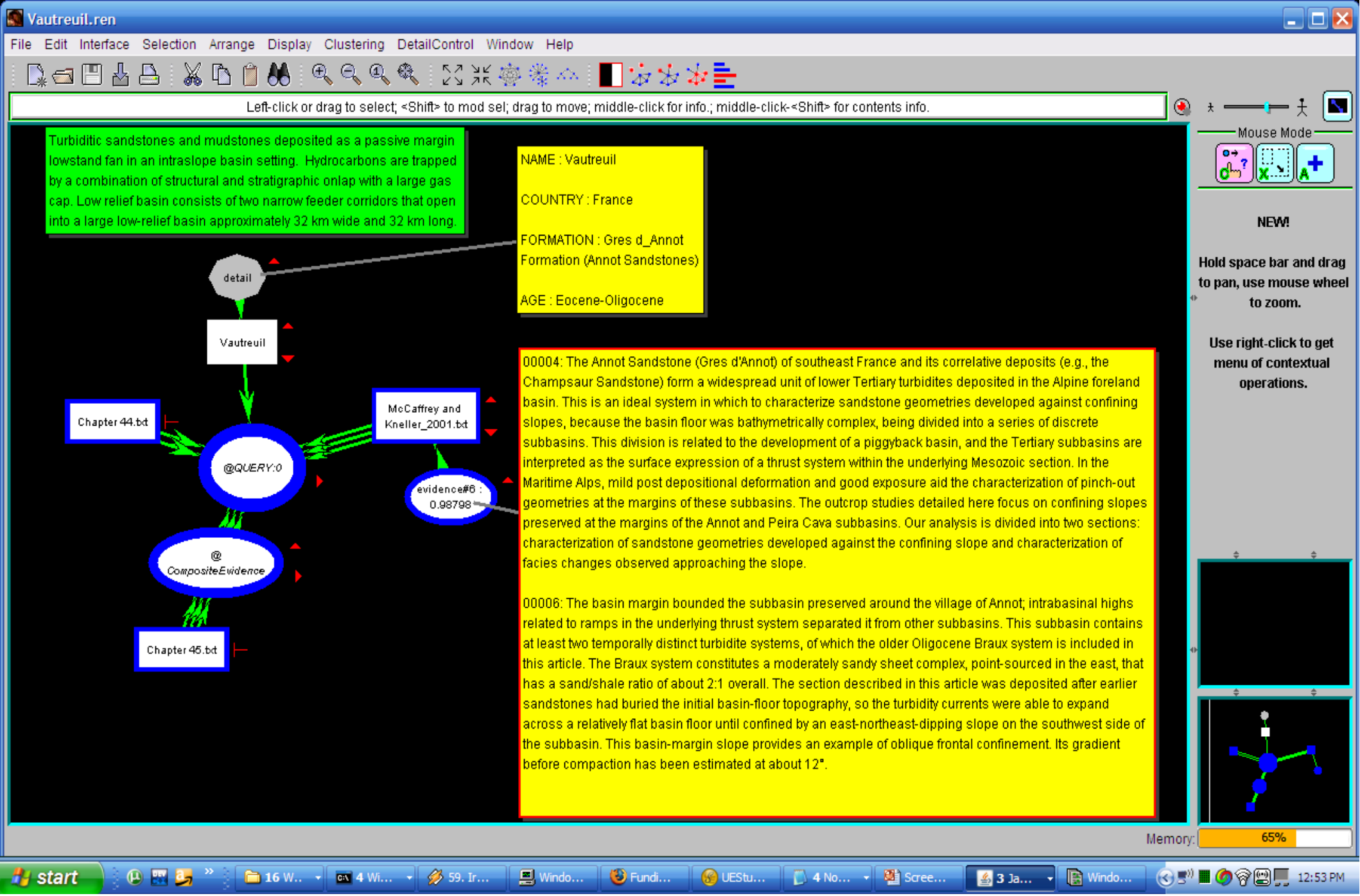
Preferences

```
Emphasis:
          On:  Tectonic Setting
          On:  Depositional Setting
          On:  Geologic Age
Weights:
          On:  Provenance
          On:  Profile
Sources:
          Corperate
                On:  Exploration
                On:  Production
                On:  Financial
          Vendor
                On:  AAPG
                On:  Wood
          External
```

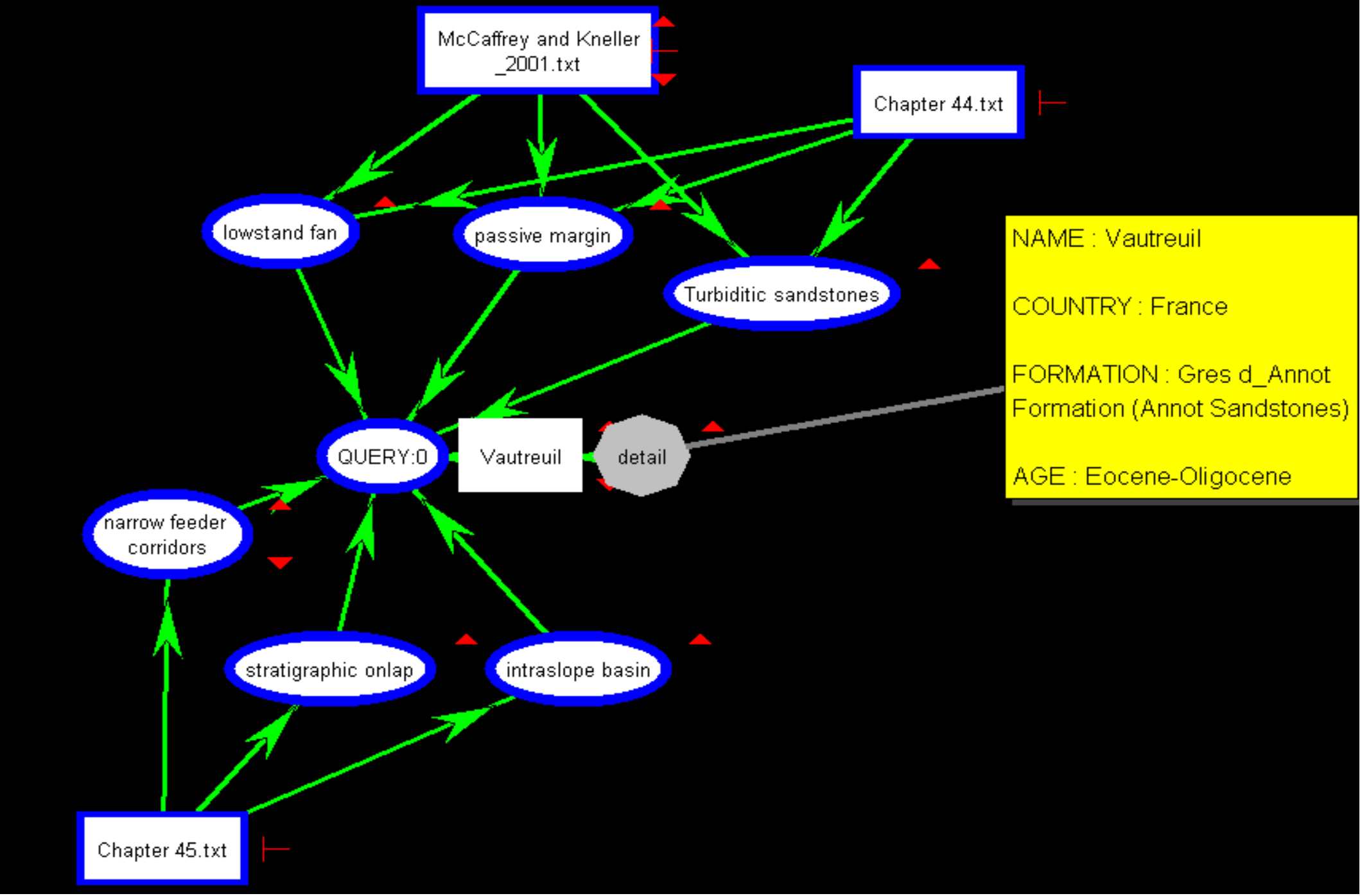
DETAILS – Next, click the “Source Visualization” button



Top-level source visualization and the highest-ranked result



Drill down to one of the documents for the human readers



Drill down into the query and its relationships to the source documents

Emergent Knowledge

When reading the 79 documents,

- **VLP translates the sentences and paragraphs to CGs.**
- **But it does not do any further analysis of the documents.**

When a geologist asks a question,

- **The VivoMind system may find related phrases in many sources.**
- **To connect those phrases, it may need to do further searches.**
- **The result is a conceptual graph that relates the question to multiple passages in multiple sources.**
- **Some of those sources might contribute information that does not have any words that came from the original question.**
- **That new CG can be used to answer further questions.**

By a "Socratic" dialog, the geologist can lead the system to explore novel paths and discover unexpected patterns.

Diagnosing Cancer Patients

The same technology can be applied to language about any topic.

As an example, the documents might describe cancer patients, and the query could describe another patient.

The analogies could highlight any aspect of interest: patient description, medical history, therapy, results, etc.

The source documents could include unstructured reports in any natural language and structured data from a database.

With appropriate parsers and translators, any of that data could be translated to conceptual graphs, indexed, and processed by the analogy engine.

Ontologies with detailed definitions would be useful, but not required.

A global alignment of the ontologies would be useful, but not required.

Operational decisions would be made by a physician, who could examine the source documents to evaluate any hypotheses generated by the system.

Conclusions

Analogy is the foundation for human reasoning.

Without analogy, language understanding is impossible.

Logic is a highly disciplined special case of analogical reasoning:

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

Conceptual graphs support both 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.**

Suggested Readings

The paper with the same title as this talk:
http://www.jfsowa.com/pubs/pursuing.pdf

A description of the VivoMind Analogy Engine and the Intellitex parser:
http://www.jfsowa.com/pubs/analog.htm

The agent architecture used for the VivoMind software:
http://www.jfsowa.com/pubs/arch.htm

The "Challenge of Knowledge Soup" for any approach to general AI:
http://www.jfsowa.com/pubs/challenge.pdf

An overview of conceptual graphs:
http://www.jfsowa.com/cg/cg_hbook.pdf

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

Papers on analogy finding by Forbus, Gentner, Falkenhainer, et al.:
http://www.qrg.northwestern.edu/papers/papers.html