THE CG MARS LANDER

Gil Fuchs\textsuperscript{1} and Robert Levinson\textsuperscript{1}

\textsuperscript{1} Department of Computer Science, University of California, Santa Cruz, CA 95060
\textsuperscript{2} in collaboration with TextWise, Syracuse, New York
E-mail: \{gil,levinson\}@cse.ucsc.edu
Phone: 408-459-2087

Abstract. The name of the system is “THE CG MARS LANDER”, which stands for \textbf{Type Hierarchy Enhanced CG Matching And Retrieval System Linear Associative N-tuple Deductive Embedded Representation}. The application is text retrieval. The technique involves translation of text fragments into an informationally ordered form (CGs) and retrieval on the induced subsumption ordering. This application is a linearized CG, NLP, query, matching and retrieving system. English discourse is converted into CGs which are stored in an associative database. The hierarchical database is guided by content morphology and associated ontological type hierarchy, to produce a partially ordered “more general than” hierarchy based on subgraph containment. Due to the nature of the ADB a query is answered by being inserted into the DB. The query’s successors in the DB are the answers to the query, as well as the (more specific) retrieved relevancies.

1 Introduction

1.1 Background and motivation

CGs have a great promise and many applications, natural language processing and understanding, knowledge acquisition, modeling, data mining to name a few. Development and research in this domain, however, has suffered from the belief that this promising technology will not scale, even on multi processor computers. The initial timings of this project gave comfort to the sceptics. The first attempt (early 1996) at converting 100,000 English sentences to CGs and matching the CG database against a small set of queries took about 6.5 hours on a Sun Sparc Ultra enterprise 4000. Using a variety of techniques, including the progressive application of the Levinson associative database methods III through VI \cite{8}, we managed to reduce this to 19 seconds (of which 16 seconds is the overhead ontology loading and 3 seconds is the actual processing time) on the same Sparc station (early 1997). Our goal is to further speed the process up so that this technology can actually be useful on real world size problems.

2 Architecture of the System

2.1 High level architecture
Figure 1 shows the overview of the CG tool. English discourse, coupled with ontology and queries are inputs to the system, and the result is an English answer with appropriate references to where in the English discourse the query gets answered.

2.2 Key features

Simple representation of CGs: By employing a linear tuple notation we provide for a simple and efficient implementation of CGs. A CG consists of $n$ tuples, each of which, has any number of arguments, which, in turn, can either be a linguistic construct or a CG.

Expressiveness of CGs:
item negation: Any CG, relation, or concept node arguments may be negated.
- argument values: Any concept node argument may have a value.
- ranges: Any place a single value is allowed, a range is also permissible.
- dates: These ranges, may be of date or time type.

Arbitrary nesting capability of CGs: An argument can, in turn, be a previously introduced CG tag. Hence, allowing nested contexts to any depth.

Methods III through VI of associative DB matching: Exploiting Levinson's algorithms for optimized graph retrieval.

Automatic maintenance of the type hierarchy: The type hierarchy is an isa partial order over some 100,000 English concepts.

Automatic cycle recognition for the type hierarchy: Should the above hierarchy contain cycles, a warning is issued.

DSPS codes: Each CG may contain an association to the document, section, paragraph and sentence from which it was derived, for bibliographic retrieval.

2.3 Levinson's UDS [8] design foundations slogans
- Every primitive data object, label or symbol should be stored only once with pointers used to denote the actual uses of the object.
- Every compound object should be stored with the minimum information required to represent the combination of its parts.
- Given no loss of accuracy, objects should be processed at the highest level of abstraction possible.
- If one were to implement a conceptual graph based on the diagrammatic representation, the costs associated with storage and matching would be much higher than they need to be.
- The same abstraction mechanism that goes from labels to graphs can be taken one step further to facilitate the storage and retrieval of nested context graphs.
- A graph is itself the best descriptor of its nodes.

3 Application of the System

3.1 Summary of results and timings
Our current program can accept a large set of CGs (tens of thousands), an ontology of some few hundred thousands words, and a set of queries. The program stores the CGs in a database which can be saved and restored, and answer queries, by returning relevant CGs from the previously saved DB. The timing statistics on a Sun Ultra Enterprise 4000 (with 4 UltraSPARC 167Mhz and 512KB External Cache CPU and 256BM of main memory) are as follows. Read, process, and store an 18,000 CG input file in 1 hour and 46 minutes. Reloading of above DB takes on the order of seconds. A 150,000 word ontology is processed in 16 seconds. Each query is handled in 5.5 seconds. For smaller databases (hundreds of CGs only), the time to handle a single query can be as low as 0.2
seconds. A typical CG consists of some ten tuples, each of which has two to five arguments. Some large CGs can, however, reach up to 30 tuples (with no effective limit in the program). CG processing includes the treatment of entity negation, item values and ranges (including dates), as well as nesting.

3.2 Sample session

Samples of elements in the package:

- An English sentence:

  The US government's privately managed Hungarian American Enterprise Fund has invested 1 million Dollars in Hungary's first business to establish automated teller machines nationwide.

- Tokenization and tagging using TextWise's parser and conversion to CGs. One sentence may give rise to several CGs, with embedding also possible.

- Standard CG notation:

  ```
  [[government]-->[AGNT] --> [be]] --> (AGNT) --> manage
  --> (OBJ)
  -->
  [[Hungarian_American_Enterprise_Fund] --> (AGNT)
  --> [invest] --> (OBJ) --> [dollars: 1000000]
  --> (IN) --> [First-Business]].
  ```

  Note that the sentence parsed above is not related to the question and answer which follow.

- Sample queries:

  ```
  /* Q1: When did Rupert Murdoch own the New York Post? */
  --> [New_York_Post]] --> (PTIM) --> [??].
  ```

  - Answer:

  ```
  [[Rupert Murdoch] -->[AGNT] --> [own]
  --> (OBJ) --> [New_York_Post]
  --> (PTIM) --> [year: (1976, 1988)].
  ```

3.3 Cost benefit analysis

Suppose one only plans to do a small number of queries $Q_s$ over a database of $N$ CGs. The question arises whether it is worth the overhead of creating a
Levinson Method III database, in which retrievals and insertions take approximately \(\log_{10}(N)\) comparisons for a database of size \(N\). The alternative is simply to compare each query to every CG.

Here we do the necessary math to aid in the decision: No precompilation of database; cost is \(N \cdot Q\) graph comparisons. (call this Method I)

Creating method III database: average cost per insertion is approximately \(\log_{10}(N^2)\), giving a cost to create the database of \(N \log_{10}(N^2)\) and a cost to answer the queries of \(Q \log_{10}(N)\).

So the question is when \(NQ < (N \log_{10}(N^2) + Q \log_{10}(N))\).

Here is a table of results:

<table>
<thead>
<tr>
<th>(N)</th>
<th>(Q)</th>
<th>Method I Cost</th>
<th>Method III Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>10</td>
<td>5.00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>100</td>
<td>14.90</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>10000</td>
<td>104.80</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>1000</td>
<td>296.60</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100000</td>
<td>328.64</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
<td>100000</td>
<td>688.60</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
<td>1000000</td>
<td>7,293.43</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>10000000</td>
<td>7,374.44</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>100000000</td>
<td>8,184.40</td>
</tr>
<tr>
<td>10000</td>
<td>1</td>
<td>1000000000</td>
<td>10,284.44</td>
</tr>
<tr>
<td>10000</td>
<td>10</td>
<td>10000000000</td>
<td>152,823.78</td>
</tr>
<tr>
<td>10000</td>
<td>100</td>
<td>100000000000</td>
<td>296,823.78</td>
</tr>
</tbody>
</table>

Table 1. Comparison table between costs of respective methods

From this it can clearly be seen, that Method III is most cost effective except for very small ratios of queries to database size, and for very high ratios the savings grow exponentially.

Levinson’s Methods IV-VI provide further order of magnitude improvements over these numbers.

4 Limitations of the System

4.1 Open questions and concerns

The translation from an English sentence to a CG is not, by all means, unique. One illustrative example might be the following sentence.

Rupert Murdoch owns Fox.

Which can be rendered either as a CG showing a characteristic (possession) or as a standard agent-verb-object relation.
Apart from the need for consistency, it isn’t clear which of the above alternatives is better and why. Do we need to add graph transformation rules to exploit these equivalencies (in addition to the type hierarchy)? Should we create a mechanism for inferencing? At this time, these are still open questions.

5 Acknowledgements

The work on this project was done in collaboration with the research staff of TextWise, Syracuse, New York. TextWise and Manning and Napier Information Systems have provided the entire financial support for the project. Some of the work is part of a larger SBIR commercialization project at TextWise administered by Rome Labs. All commercial rights are the property of TextWise and Manning and Napier Information Systems.

Relevant papers on which this work is based [1, 2, 3, 4, 5, 6, 7, 8] are included in the list of references that follows:

References


This article was processed using the LATEX macro package with LLNCS style