From Minimal Logical Forms for Answer Extraction to Logical Graphs for Question Answering

Diego Mollá

Department of Computing, Macquarie University, Sydney, Australia
diego.molla-aliod@mq.edu.au
http://web.science.mq.edu.au/~diego/

Abstract. Many exciting things happened since I left the ExtrAns project at the University of Zurich. In this paper I present a very brief journal of the research that derived from my work at Zurich. My work combined a study of several representations of questions and answer sentences with the development of procedures to find the answer. The culmination of this work was the definition of the Logical Graphs (LGs), which are graph representations derived from ExtrAns' original Minimal Logical Forms (MLFs), and the development of a method for the automatic learning of question-answer patterns based on LGs. I hope the reader will enjoy reading this journey of ideas.

Key words: answer extraction, question answering, minimal logical forms, logical graphs

1 Introduction

My post-doctoral research started with the ExtrAns answer extraction project at the University of Zurich. Work carried out there lead to the definition of Minimal Logical Forms (MLFs) as the core representation of text and questions. Life continued on and I left Zurich in pursue of new research paths. In the process my research has evolved from the original idea of the MLFs to a graph-based representation and procedures. In this paper I will give a quick review of the key research ideas that lead to my current work following a chronological order. Section 2 introduces the original MLFs; Section 3 describes the use of various similarity measures including MLFs; Section 4 gives some detail of the logical patterns I used for question answering; Section 5 shows the transformation of MLFs into graphs; Section 6 details some work on the format of graphs derived from the change to a new parser; and Section 7 wraps up everything.

2 Minimal Logical Forms

The Minimal Logical Forms (MLFs) were designed as a means to express the logical contents of sentences in a way that is practical to produce and to work with, yet still able to be generalised to express the full meaning of sentences. This is a compromise that worked well and that turned out to be the same sort of compromise used in the literature, notably by Jerry Hobbs [Hobbs, 1985]. Similar approaches were used in practical implementations of Question Answering (QA) systems such as Lasso [Moldovan et al., 1999], a system that obtained the best results in the Question Answering track of the TREC 1999 conference.

Let us recapitulate the key features of the Minimal Logical Forms used by ExtrAns:

- The format of the MLFs is in Prolog. The complete process of determining if a sentence is an answer uses Prolog unification.
- The logical forms are flat lists of predicates that use reification as a means to express information that otherwise would be presented in nested expressions.
- There is no attempt to express the full semantic interpretation of the sentence and many issues that are typical sources of ambiguity, such as determiner scope and tense, are left out. I have published some suggestions on how to handle these issues [Mollá, 2001] but it was never our goal to implement them.

The MLFs as used by ExtrAns used the output of Link Grammar (LG). LG is an open-source dependency-based parser and grammar system implemented in the C programming language and that is easily customisable [Sleator and Temperley, 1993]. We modified the original parser so that it could express dependency structures (called linkages in LG's terminology) with conjunctions as single trees, in contrast with the original form that expressed independent linkages for each conjunct. The resulting answer extraction system incorporating Link Grammar was a simple system that was easy to port to other machines.

One of the first changes that we did at Macquarie University was to switch to Conexor FDG [Tapanainen and Järvinen, 1997] (henceforth Conexor), a dependency-based parser that was faster and more accurate, as shown by our evaluations [Mollá and Hutchinson, 2003]. The aim was to simply replace the parser and keep the format of the original logical forms, and in the process we proposed a principled way to build the logical form from the output of a dependency-based parser such as Conexor using an introspection stage where each word is considered in isolation, and an extrospection stage where additional logical relations are incorporated [Mollá and Hutchinson, 2002]. Given the limited human resources we had there was no attempt to provide an optimised conversion of the parser output. For example, we did not attempt to trace long-distance dependencies and therefore a sentence like "Mary wants to marry a sailor" would not express the concept that the subject of marrying is Mary. This is a major difference from ExtrAns' logical forms, yet our comparative studies showed a better performance with the new parser. The new logical forms had less information but the available information was correct more often.

3 Similarity Measures

In the years that followed the incorporation of Conexor we performed more radical changes to the original system, this time on the way the answer was found. Instead of using Prolog unification and backtracking we experimented with various similarity measures between the question and the answer. We experimented with the following similarity measures [Mollá, 2003]:

– Word overlap: This is a simple method that retrieves the sentences with highest overlap of keywords with the question.
– Grammatical relation overlap: This method computes the overlap of grammatical relations [Carroll et al., 1998]. We used a subset of grammatical relations that could be extracted by our system.
– Logical form overlap: This method is a simplification of ExtrAns' answer extraction. It computes the number of MLFs that are in common between question and answer. This overlap used Prolog unification to handle the variables in the logical forms.

The framework was a question-answering system that returned exact answers. This task is more difficult than ExtrAns' answer extraction, which returned complete sentences containing the answer. The similarity measures listed above were used to select sentences and rank the named entities that were compatible with the expected answer type. So, if the question was "Who turned 60 years old on October 31st 2009", the expected answer type is a person. All person names identified in the preselected sentences were identified, and if a named entity appeared in several sentences its score would be the sum of scores of each sentence. With this simple framework we found that the simplest method, word overlap, was best than any other isolated methods, but a combination of all methods was best.

4 Answer Patterns

The question answering method described above relied on a good named entity (NE) recogniser but the NE recognisers we had available were unreliable. We tried ANNIE, GATE's [Gaizauskas et al., 1996] NE recogniser but the program had a tendency to crash. We tried to use ANNIE to compute all named entities off-line, and after over a month of computer processing and numerous crashes we managed to obtain the list of entities. Still, because of the crashes and the way we collected the entities there was an undetermined number of documents of which we did not have the named entities.

To reduce the dependency on the NE recogniser we used the logical forms to try to find the exact answers directly, without relying on named entities. In an initial study we examined questions and answers and after painstaking work we came with a few very generic rules such as the one of Figure 1 [Mollá and Gardiner, 2004].

Pattern:
object(what,_,[XW])
object(A,OA,[XW])
prop(of,_,[XW,XB])
object(B,OB,[XB])
Replacement:
object(B,OB,[XB])
evt(have,_,[OB,XW])
object(A,OA,[XW])
prop(of,_,[XW,XANSW])
dep(ANSWER,OX,[XANSW])

Fig. 1. Minimal Logical Form Rule

This rule addresses questions of the form "What is the X of Y?" where the candidate answer sentence is "X has a Y of ANSWER". The example shows clearly a shortcoming of this approach: it is very difficult to understand the logical patterns, and trying to find these patterns manually was a very labour intensive task. For this reason we moved on to try to learn the logical form patterns.

5 Using Graphs

To learn the logical form patterns we used a formalism that I wanted to try for quite a long time: graphs. A logical form can be expressed as a graph whose vertices represent predicate concepts and whose edges represent variable bindings between predicates. This way, for example, the pattern of Figure 1 can be expressed as in Figure 2.

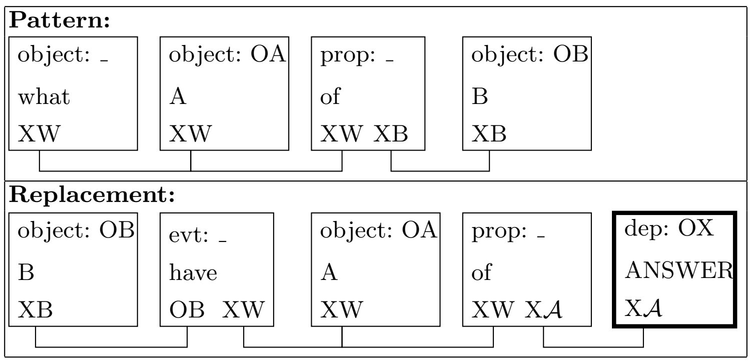

Fig. 2. Graph Representation of a Minimal Logical Form Rule; the vertex in thick lines represents the answer.

These graphs are sugar coating of the original notation but they are not graphs in the traditional Graph Theory sense because of the way the vertices are connected. We decided to simplify them so that it becomes easier to use traditional Graph Theory tools on them. Inspired on Sowa's Conceptual Graphs [Sowa, 1979], we defined directed bipartite graphs with two types of vertices named concepts and relations [Mollá and van Zaanen, 2005]:

Concepts are MLF's objects and eventualities;
Relations are MLF's properties and other relational predicates like lattices produced by a conjunction.

Figure 3 shows an example of a Minimal Logical Form and its corresponding Logical Graph.

A person is between a rock and a hard place
holds(e3), object(person,o2,[x2]), evt(be,e3,[x2]), prop(between,p4,[e3,x7]), x6≤x7, x10≤x7, object(rock,o6,[x6]), prop(hard,p9,[x10]), object(place,o10,[x10])

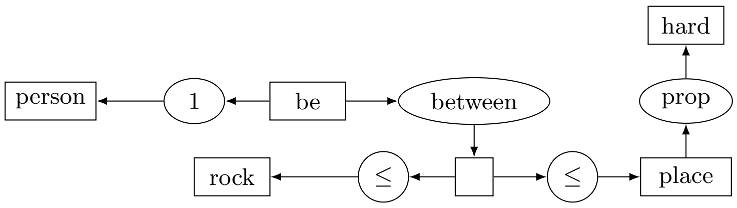

Fig. 3. A Logical Graph; concepts are represented as square boxes and relations are ellipses.

Though inspired on Sowa's Conceptual Graphs (CGs) they differ from them in that ours are traditional graphs that can use traditional Graph Theory operations. Sowa's CGs, in contrast, are associated with a semantic theory of inference.

The resulting graphs are named Logical Graphs (LGs). By using the LGs we substitute the concept of overlap with that of Minimum Common Subgraph (MCS) so that the size of the MCS between a question and a sentence is an indication of similarity and a hint that the answer might be there.

The good thing of graphs (and in general of any structure with relational information) is that now we can follow the connections between vertices to try to find the answer. So we devised a simple mechanism to learn question-answering patterns based on the MCS between question and answer sentence graphs [Mollá, 2006]. Basically, given a question-answer pair in the training corpus, the MCS between the question and the answer sentence can be used as the question pattern, and the path that connects this MCS with the actual answer in the sentence is the pattern to look for in the candidate answer sentence.

Our experiments showed that indeed we can obtain the answer this way, though our training set was very small and therefore the results were not impressive.

6 From Conexor to Enju

Recently the license of Conexor expired and we had to find another parser. This became an opportunity to experiment with different graph formats to decide what is the optimal graph for Question Answering. Our results in this section are sadly not comparable with those of the previous section so it is not known whether the modifications shown here are better than the original LGs as described above, but still we think the results are interesting enough to report here.

Enju is a parser developed by the University of Tokyo that uses a wide-coverage HPSG grammar [Sagae et al., 2007]. We chose this parser and grammar among other free tools because it produces a dependency-based structure that can be converted to graphs and because there is also a grammar that is fine-tuned for the biomedical domain, an area that I am interested in exploring in the future.

Being HPSG-based, the dependencies use conventions that differ from those of Conexor, though, and we had to modify some of them. In particular, we reversed the direction of dependency of all determiners and modifiers, since HPSG treats them as the head of the dependency. We also modified the dependencies concerning prepositions to make them similar to our original Logical Graphs. Figure 4 shows the original graph obtained from Enju, and after the modifications that were made.

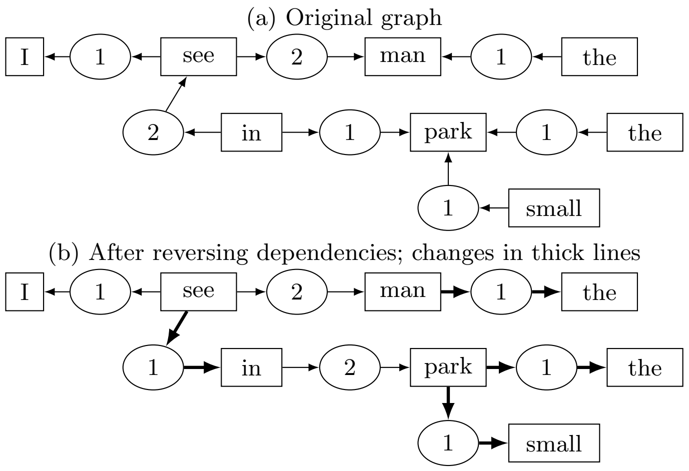

Fig. 4. Variations of the output of the Enju parser for the sentence *I saw the man in the park.*

Changes in the directions of dependency required changes in the relation labels. For example, the graph of Figure 4 (b) would seem to imply that "in" is the argument 1 of "see", but "see" already has "I" as its argument 1. We are currently experimenting with the level of detail that we want to encode in the labels: from simply stating the part of speech of the old head (so that, say, "in" is the argument IN1 of "see" and "I" is the argument VB1 of "see") to using a different label per different word. Our current experiments seem to indicate that, if we have a corpus of questions and answers large enough, using graphs with very specific information in the labels could be the best bet.

7 Conclusions and Further Work

To conclude, research initiated at the University of Zurich led to a very productive line of work. Engineering issues prevented our system from obtaining better results but, on the theoretical side, the methods developed were very exciting . . . and they are still my main area of research.

Further work includes expanding the learning method to incorporate different weighting systems of parts of the graphs according to some sort of semantic similarity between words and semantic importance of the particular word for question answering. An idea that I toyed with was the use of WordNet as in the work about MCS for Conceptual Graphs [Montes-y-Gómez et al., 2001]. Something that might be more exciting would be to use methods to identify word relations automatically [van der Plas and Bouma, 2005].

Furthering on the analysis of graphs for QA, an interesting area worth exploring is the study of the formal properties of graphs and the determination of the actual graph types that are expressive enough for what we need to represent of sentences and questions for QA, yet still allowing fast processing of the MCS. The original task of finding the MCS of two arbitrary graphs is NP-complete, but in theory it could be possible to define a subclass of graphs that allow a faster algorithm. Alternatively we could look at methods to compute an *approximate* MCS in polynomial time.

Moving to other areas, I am interested in exploring the use of graphs as a means to present abstracts of the original text source. This in a sense would be a step forward from the use of in-context highlighting of the original ExtrAns system, but rather than returning an extract, graphs can be used to detect the important bits of information, remove redundancies in complex answers spanning multiple documents, and generate new text. This would mean going beyond the boundary of sentences and enable graphs to express complete documents or even sets of documents.

And research continues on.

8 Thanks and Acknowledgements

Research described in this paper is originated by work carried out at the University of Zurich and led by Prof. Michael Hess. I am grateful for his original idea of using logical forms for question answering, and though I might have gone a bit away from the original form, I would like to think that I didn't go astray. The work presented here is in essence inspired from that original idea. Parts of the research carried out at Macquarie University have been funded with the Macquarie University New Staff Grant and with the Australian Research Council Discovery Grant number DP0450750.

References

[Carroll et al., 1998] Carroll, J., Briscoe, T., and Sanfilippo, A. (1998). Parser evaluation: a survey and a new proposal. In *Proc. LREC98.*

[Gaizauskas et al., 1996] Gaizauskas, R., Cunningham, H., Wilks, Y., Rodgers, P., and Humphreys, K. (1996). GATE: an environment to support research and development in natural language engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence*, Toulouse, France.

[Hobbs, 1985] Hobbs, J. R. (1985). Ontological promiscuity. In *Proc. ACL'85*, pages 61–69. University of Chicago, Association for Computational Linguistics.

[Moldovan et al., 1999] Moldovan, D., Harabagiu, S., Paşca, M., Mihalcea, R., Goodrum, R., Gîrju, R., and Rus, V. (1999). LASSO: A tool for surfing the answer net. In Voorhees, E. M. and Harman, D. K., editors, *Proc. TREC-8*, number 500-246 in NIST Special Publication. NIST.

[Mollá, 2001] Mollá, D. (2001). Ontologically promiscuous flat logical forms for NLP. In Bunt, H., van der Sluis, I., and Thijsse, E., editors, *Proceedings of IWCS-4*, pages 249–265. Tilburg University.

[Mollá, 2003] Mollá, D. (2003). Towards semantic-based overlap measures for question answering. In *Proc. ALTW03*, pages 130–137, Melbourne.

[Mollá, 2006] Mollá, D. (2006). Learning of graph-based question answering rules. In *Proc. HLT/NAACL 2006 Workshop on Graph Algorithms for Natural Language Processing*, pages 37–44.

[Mollá and Gardiner, 2004] Mollá, D. and Gardiner, M. (2004). Answerfinder - question answering by combining lexical, syntactic and semantic information. In Asudeh, A., Paris, C., and Wan, S., editors, *Proc. ALTW 2004*, pages 9–16, Sydney, Australia. Macquarie University.

[Mollá and Hutchinson, 2002] Mollá, D. and Hutchinson, B. (2002). Dependency-based semantic interpretation for answer extraction. In *Proc. 2002 Australasian NLP Workshop.*

[Mollá and Hutchinson, 2003] Mollá, D. and Hutchinson, B. (2003). Intrinsic versus extrinsic evaluations of parsing systems. In *Proc. European Association for Computational Linguistics (EACL), workshop on Evaluation Initiatives in Natural Language Processing*, pages 43–50, Budapest. Association for Computational Linguistics, ACL.

[Mollá and van Zaanen, 2005] Mollá, D. and van Zaanen, M. (2005). Learning of graph rules for question answering. In Baldwin, T., Curran, J. R., and van Zaanen, M., editors, *Proc. ALTW 2005*, Proceedings of the Australasian Language Technology Workshop. Australasian Language Technology Association.

[Montes-y-Gómez et al., 2001] Montes-y-Gómez, M., Gelbukh, A., and Baeza-Yates, R. (2001). Flexible comparison of conceptual graphs. In *Proc. DEXA-2001*, number 2113 in Lecture Notes in Computer Science, pages 102–111. Springer-Verlag.

[Sagae et al., 2007] Sagae, K., Miyao, Y., and Tsujii, J. (2007). Hpsg parsing with shallow dependency constraints. In *Proc. ACL 2007*, pages 624–631.

[Sleator and Temperley, 1993] Sleator, D. D. and Temperley, D. (1993). Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292.

[Sowa, 1979] Sowa, J. F. (1979). Semantics of conceptual graphs. In *Proc. ACL 1979*, pages 39–44.

[Tapanainen and Järvinen, 1997] Tapanainen, P. and Järvinen, T. (1997). A non-projective dependency parser. In *Proc. ANLP-97*. ACL.

[van der Plas and Bouma, 2005] van der Plas, L. and Bouma, G. (2005). Automatic acquisition of lexico-semantic knowledge for qa. In *Proceedings Ontolex 2005*, page 9 pages.