From Specialized Syntax to General Logic: The Case of Comparatives

Ruiting Lian[1,2], Rodas Solomon[3], Amen Belayneh[2,3], Ben Goertzel[4], Gino Yu[2], and Changle Zhou[1,5]

[1] Cognitive Science Department, Xiamen University [2] School of Design, Hong Kong Poly U
[3] iCog Labs, Addis Ababa[4] OpenCog Foundation [5] corresponding author

Abstract. General-purpose reasoning based on knowledge encoded in natural language, requires mapping this knowledge out of its syntax-dependent form into a more general representation that can be more flexibly applied and manipulated. We have created a system that accomplishes this in a variety of cases via mapping English syntactic expressions into predicate and term logic expressions, which can then be cognitively manipulated by tools such as a probabilistic logic engine, an information-theoretic pattern miner and others. Here we illustrate the functionality of this system in the particular case of comparative constructions.

1 Introduction

In order for an AI system to reason in a general-purpose way about knowledge that comes to it in natural language form, the system must somehow transform the knowledge into a more flexible representation that is not tied to the specific linguistic syntax in which it was originally expressed. There is no consensus in the AI or computational linguistics fields on the best way to do this; various approaches are being pursued in a spirit of experimental exploration [8]. We describe here the approach we have been exploring, in which a sequence of transformations maps syntactic expressions into abstract logic expressions, in a logical language mixing predicate and term logic as specified in Probabilistic Logic Networks [2] [3]. This language comprehension pipeline has been constructed as part of a broader project aimed at Artificial General Intelligence, the open-source OpenCog initiative [4] [5]; it has been described previously in a 2012 overview paper [11], but has advanced considerably in capabilities since that time.

To illustrate the properties of this comprehension pipeline, we focus here on the case of comparative sentences. We have chosen comparatives for this purpose because they are an important yet difficult case for any NLP system to deal with, and hence a more interesting illustration of our NLP concepts and system than a standard case like SVO constructs, which essentially any reasonably sensible language processing framework can deal with acceptably in most cases. Comparatives present a diversity of surface forms, which are yet ultimately mappable into relatively simple logical structures. They are somewhat confusing from the perspective of modern theoretical linguistics, and also tend to be handled poorly by existing statistical language processing systems.

The language comprehension pipeline reviewed here is broadly similar in concept to systems such as Fluid Construction Grammar [14] [13]and Cycorp's [1] proprietary NLP

[1] `http:\\cyc.com`

system. However, it differs from these in important aspects. The approach given here utilizes a dependency grammar (the link grammar [12]) rather than a phrase structure grammar, and at the other end involves a customized logic system combining aspects of term logic and predicate logic. As reviewed in [11], this combination of dependency grammar and term logic allows a large amount of ambiguity to be passed through from the surface level to the logic level, which is valuable if one has a powerful logic engine with a substantial knowledge base, able to resolve ambiguities based on context in a way that earlier-stage linguistic processes could not.

2 A Deep Linguistics and Logical Inference Oriented Comprehension Pipeline

We now briefly review the language comprehension pipeline utilized in the work presented here.

2.1 Link Grammar

The initial, syntactic phase of our pipeline consists of the link grammar [12]. The essential idea of link grammar is that each word comes with a feature structure consisting of a set of typed connectors . Parsing consists of matching up connectors from one word with connectors from another. Consider the sentence:

The cat chased a snake

The link grammar parse structure for this sentence is shown in Figure 1.

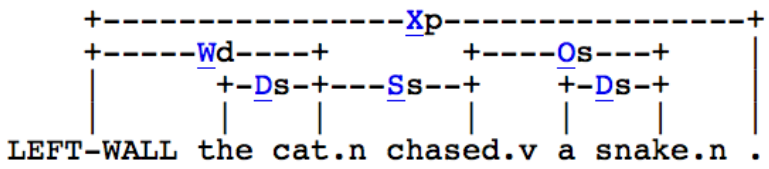

Fig. 1. Example link parse.

There is a database called the “link grammar dictionary” which contains connectors associated with all common English words. The notation used to describe feature structures in this dictionary is quite simple. Different kinds of connectors are denoted by letters or pairs of letters like S or SX. Then if a word W1 has the connector S+, this means that the word can have an S link coming out to the right side. If a word W2 has the connector S-, this means that the word can have an S link coming out to the left side. In this case, if W1 occurs to the left of W2 in a sentence, then the two words can be joined together with an S link.

The rules of link grammar impose additional constraints beyond the matching of connectors – e.g. the planarity and connectivity metarules.. Planarity means that links don't cross. Connectivity means that the links and words of a sentence must form a connected graph – all the words must be linked into the other words in the sentence via some path.

2.2 RelEx

The next phase in the pipeline under discussion is RelEx, an English-language semantic relationship extractor, designed to postprocess the output of the link parser [self-citation removed, to be inserted in the final version]. It can identify subject, object, indirect object and many other dependency relationships between words in a sentence; it generates dependency trees, resembling those of dependency grammars. The output of the current version of RelEx on the example sentence given above is:

```
singular(cat)
singular(snake)
_subj(chase, cat)
_obj(chase, snake)
past(chase)
```

Internally, RelEx works via creating a tree with a FeatureNode corresponding to each word in the sentence, and then applying a series of rules to update the entries in this FeatureNode. The rules transform combinations of link parser links into RelEx dependency relations, sometimes acting indirectly via dynamics wherein one rule changes a feature in a word's FeatureNode, and another rule then takes an action based on the changes the former rule made. Figure **??** gives a high level overview of RelEx's internal process.

The output of RelEx is not unlike that of the Stanford parser, and indeed RelEx has a Stanford parser mode that causes it to output relations in Stanford parser compatible format. However, in our tests RelEx + link parser proved around 4x as fast as the 2012 Stanford parser [9], and qualitatively appeared to give better performance on complex constructs such as conjunctions and comparatives (which makes sense as such constructs are probably not that diversely represented in the Stanford parser's training data).

2.3 OpenCog

The next phase of the pipeline, RelEx2Logic, has the purpose of translating the output of RelEx into a format compatible with the logical reasoning component of the OpenCog AI engine. OpenCog is a high level cognitive architecture aimed at exploration of ideas regarding human-level Artificial General Intelligence, in particular the CogPrime AGI design [4] [5]. OpenCog has been used for commercial applications in the area of natural language processing and data mining , and has also been used for research involving controlling virtual agents in virtual worlds, controlling humanoid robots, genomics data analysis, and many other areas.

The centerpiece of the OpenCog system is a weighted, labeled hypergraph knowledge store called the Atomspace, which represents information using a combination of predicate and term logic formalism with neural net like weightings. The NLP comprehension pipeline described here is centrally concerned with mapping English language text into logical representations within the Atomspace.

The primary component within OpenCog that acts on the output of RelEx2Logic is Probabilistic Logic Networks (PLN) [2], a framework for uncertain inference intended to enable the combination of probabilistic truth values with general logical reasoning rules. PLN involves a particular approach to estimating the confidence values with which these probability values are held (weight of evidence, or second-order uncertainty). The implementation of PLN in software requires important choices regarding the structural representation of inference rules, and also regarding "inference control" – the strategies required to decide what inferences to do in what order, in each particular practical situation.

PLN is divided into first-order and higher-order sub-theories (FOPLN and HOPLN). FOPLN is a term logic, involving terms and relationships (links) between terms. It is an uncertain logic, in the sense that both terms and relationships are associated with truth value objects, which may come in multiple varieties. "Core FOPLN" involves relationships drawn from the set: negation; Inheritance and probabilistic conjunction and disjunction; Member and fuzzy conjunction and disjunction. Higher-order PLN (HOPLN) is defined as the subset of PLN that applies to predicates (considered as functions mapping arguments into truth values). It includes mechanisms for dealing with variable-bearing expressions and higher-order functions. We will see some simple examples of the kinds of inference PLN draws below.

2.4 RelEx2Logic

OpenCog also contains a system called RelEx2Logic, that translates RelEx output into logical relationships, utilizing the mix of predicate and term logic codified in Probabilistic Logic Networks [2]. RelEx2Logic operates via a set of rules roughly illustrated by the following example:

```
_subj(y, x)
_obj(y, z)
==>
Evaluation y x z
```

which indicates, in OpenCog/PLN syntax, that y is mapped into a PredicateNode with argument list (x, z). The above rule format is highly simplified and for illustration purposes only; the actual rule used by the system is more complex and may be found along with the rest of the current rule-base at `https://github.com/opencog/opencog/tree/master/opencog/nlp/relex2logic`.

So for example, for the sentence "The pig ate the tofu", the RelEx relations

```
_subj(eat, pig)
_obj(eat, tofu)
```

would result (after some simple, automated cleanup operations) in output such as

```
InheritanceLink pig_55 pig
InheritanceLink tofu_1 tofu
EvaluationLink eat pig_55 tofu_1
```

where the subscripts indicate particular definite instances of the concepts involved. On the other hand, the sentence "Pigs eat tofu" would result (after some simple, automated cleanup operations) in simply

```
EvaluationLink eat pig tofu
```

3 Handling Comparatives

Comparatives provide more interesting examples of this sort of mapping from surface form into logical expressions. Theoretical linguistics is nowhere near a consensus regarding the proper handling of comparatives in English and other languages. Some theorists posit an ellipsis theory, suggesting that comparative syntax results from the surface structure of a sentence leaving out certain words that are present in the deep structure [10] [1]. Others posit a movement theory [6] [7], more inspired by traditional generative grammar, hypothesizing that comparative syntax involves a surface structure that rearranges the deep structure.

The link grammar framework essentially bypasses this sort of issue: either ellipsis or movement would be represented by certain symmetries in the link grammar dictionary, but these symmetries don't need to be explicitly recognized or utilized by the link parser itself, though they may guide the human being (or AI system) creating the link grammar dictionary. Currently, on an empirical basis, the link parser handles comparatives reasonably well, but the relevant dictionary entries are somewhat heterogeneous and not entirely symmetrical in nature. This suggests that either

1. the syntax of English comparatives is "messy" and heterogeneous, not fitting neatly into any of the available theories; and/or
2. the link grammar dictionary can be made significantly more elegant regarding comparatives

We suspect that the truth is "a little of both", but note that this issue need not be resolved in order to deploy the link grammar as part of a practical pipeline for comprehending complex sentences, including comparatives.

As an example of how our framework, described here, deals with comparatives, one of the RelEx2Logic rules for comparatives is in compact form

```
than(w1, w2)
_comparative(ad, w)
==>
TruthValueGreaterThanLink
    InheritanceLink w1 ad
    InheritanceLink w2 ad
```

A simple example using this rule would be:

```
Pumpkin is cuter than the white dog.
==>
_predadj(cute, Pumpkin)
than(Pumpkin, dog)
_comparative(cute, Pumpkin)
_amod(dog, white)
==>
AndLink
   InheritanceLink dog_11 white
   InheritanceLink dog_11 dog
   TruthValueGreaterThanLink
       InheritanceLink Pumpkin cute
       InheritanceLink dog_11 cute
```

On the other hand, to deal with a sentence like ”Amen is more intelligent than insane” we use a different rule, which in simplified form is

```
_predadj(adj1, W)
than(adj1, adj2)
_comparative(adj1, W)
==>
TruthValueGreaterThanLink
    InheritanceLink W adj1
    InheritanceLink W adj2
```

resulting in output

```
_predadj(intelligent, Amen)
than(intelligent, insane)
_comparative(intelligent, Amen)
==>
TruthValueGreaterThanLink
     InheritanceLink Amen intelligent
     InheritanceLink Amen insane
```

In cases where the link parser gives multiple parse options, the RelEx2Logic rules will provide a logic interpretation for each one. Statistical heuristics have been implemented to rank the multiple parses for plausibility based on a corpus, but these of course are not perfect. In some cases, multiple logical output options will be presented to OpenCog, and must be chosen between based on higher level contextual inference, which is a difficult topic and the subject of current research.

4 Reasoning About Comparatives

To illustrate the simplicity of reasoning about comparatives once the syntactic complexities are removed and a normalized logical form is achieved, we consider how our integrated system can take the inputs

- Bob likes Hendrix more than the Beatles
- Bob is American

– Menudo is liked less by Americans than the Beatles

and derive the conclusion that Bob likes Hendrix more than Menudo.

For the first sentence we obtain

```
_subj(like, Bob)
_obj(like, Hendrix)
than(Hendrix, Beatles)
_comparative(like, Hendrix)
==>
TruthValueGreaterThanLink
    EvaluationLink like Bob Hendrix
    EvaluationLink like Bob Beatles
```

and for the second, correspondingly

```
_subj(like, Americans)
_obj(like, Menudo)
than(Beatles, Menudo)
_comparative(like, Beatles)
==>
TruthValueGreaterThanLink
    EvaluationLink like Americans Beatles
    EvaluationLink like Americans Menudo
```

The logical format obtained from these sentences is quite transparent. Simply via deploying its knowledge that the TruthValueGreaterThan relationship is transitive, and that Bob is American, the PLN logic system can in two steps derive the conclusion that

```
TruthValueGreaterThanLink
    EvaluationLink like Bob Hendrix
    EvaluationLink like Bob Menudo
```

Now that we are dealing with knowledge in logical rather than syntactic form, all sorts of manipulations can be carried out. For instance, suppose we also know that Bob likes Sinatra more than Menudo,

```
TruthValueGreaterThanLink
    EvaluationLink like Bob Hendrix
    EvaluationLink like Bob Menudo
```

PLN's abduction rule then concludes that

```
SimilarityLink
    Hendrix
    Sinatra
```

This sort of reasoning is very simple in PLN, and that's as it should be – it is also commonsensically simple for humans. A major design objective of PLN was that inferences that are simple for humans, should be relatively compact and simple in PLN. The task of the language comprehension pipeline we have designed for OpenCog is to unravel the complexity of natural language syntax to unveil the logical simplicity of the semantics underneath, which can then oftentimes be reasoned on in a very simple, straightforward way.

5 Conclusion

We have summarized the operation of a natural language comprehension system that maps English sentences into sets of logical relationships, in the logic format utilized by a probabilistic inference engine implemented within a general purpose cognitive architecture. This comprehension system is being utilized within prototype applications in multiple areas including a non-player character in a video game, a humanoid robot operating in an indoor environment, and a chat system running on a smartphone interacting with a user regarding music and media consumption.

We have focused here on the processing of comparatives, as this is a nontrivial case that is currently confusing for linguistic theory and handled suboptimally by many parsing systems. For practical cases of comparatives, as for most other cases, our system qualitatively appears to give adequate performance.

However, significant work remains before we have a generally robust comprehension system capable for use in a wide variety fo dialogue systems. Handling of conjunctions and quantifiers is one of the primary subjects of our current work, along with the use of PLN to handle commonsense inferences more subtle than the simple inference case summarized here.

6 Beyond Hand-Coded Rules

The language comprehension architecture described here is, in its current implementation, largely founded on hand-coded linguistic rules: the link-grammar dictionary, the RelEx rule-based and the RelEx2Logic rule-base. However, this is not viewed as an integral aspect of the approach pursued. In fact, research is currently underway aimed at replacing these hand-coded rules with rules automatically learned via unsupervised corpus learning; this work is overviewed in [15].

The point of the hand-coded rule-bases used in the current work is not to serve as a lasting foundation for intelligent English language processing; our view is that this would be an infeasible approach in the end, as the number of rules required would likely be infeasible to encode by hand. Rather, the point of the hand-coded rule-bases is to dissociate the problem of *language processing architecture* from the problems of *language learning* and *linguistic content*. Using the hand-coded rule-bases as a "working prototype" of linguistic content regarding the English language, we are able dissociate the architecture problem from the learning problem, and present what we propose as a general and powerful architecture for language comprehension and generation. The problem of learning more broadly functional linguistic content to operate within this architecture, is then viewed as a separate problem, we believe addressable via OpenCog learning algorithms.

References

1. Bhatt, R., Takahashi, S.: Winfried lechner, ellipsis in comparatives. The Journal of Comparative Germanic Linguistics 14(2), 139–171 (2011), http://dx.doi.org/10.1007/s10828-011-9042-3

2. Goertzel, B., Ikle, M., Goertzel, I., Heljakka, A.: Probabilistic Logic Networks. Springer (2008)
3. Goertzel, B., Coelho, L., Geisweiller, N., Janicic, P., Pennachin, C.: Real World Reasoning. Atlantis Press (2011)
4. Goertzel, B., Pennachin, C., Geisweiller, N.: Engineering General Intelligence, Part 1: A Path to Advanced AGI via Embodied Learning and Cognitive Synergy. Springer: Atlantis Thinking Machines (2013)
5. Goertzel, B., Pennachin, C., Geisweiller, N.: Engineering General Intelligence, Part 2: The CogPrime Architecture for Integrative, Embodied AGI. Springer: Atlantis Thinking Machines (2013)
6. Grant, M.: The Parsing and Interpretation of Comparatives: More than Meets the Eye (2013), http://scholarworks.umass.edu/open_access_dissertations/689/
7. Izvorski, R.: A dp -shell for comparatives. Proceeding of CONSOLE III pp. 99–121 (1995)
8. Jurafsky, D., Martin, J.: Speech and Language Processing. Pearson Prentice Hall (2009)
9. Klein, D., Manning, C.: Accurate unlexicalized parsing. Proceedings of the 41st Meeting of the Association for Computational Linguistics pp. 423–430 (2003)
10. Lechner, W.: Ellipsis in Comparatives. Studies in generative grammar, Moulton de Gruyter (2004), http://books.google.com.hk/books?id=JsqUHHYSXCIC
11. Lian, R., Goertzel, B., Ke, S., OÕNeill, J., Sadeghi, K., Shiu, S., Wang, D., Watkins, O., Yu, G.: Syntax-semantic mapping for general intelligence: Language comprehension as hypergraph homomorphism, language generation as constraint satisfaction. In: Artificial General Intelligence: Lecture Notes in Computer Science Volume 7716. Springer (2012)
12. Sleator, D., Temperley, D.: Parsing english with a link grammar. Third International Workshop on Parsing Technologies. (1993)
13. Steels, L.: Design Patterns in Fluid Construction Grammar. John Benjamins (2011)
14. Steels, L.: Modeling The Formation of Language in Embodied Agents: Methods and Open Challenges, pp. 223–233. Springer Verlag (2010)
15. Vepstas, L., Goertzel, B.: Learning language from a large unannotated corpus: A deep learning approach. Technical Report (2013)