LOR-KBGEN, A Hybrid Approach To Generating from the KBGen Knowledge-Base

Bikash Gyawali
Université de Lorraine, LORIA, UMR 7503
Vandoeuvre-lès-Nancy, F-54500, France
bikash.gyawali@loria.fr

Claire Gardent
CNRS, LORIA, UMR 7503
Vandoeuvre-lès-Nancy, F-54500, France
claire.gardent@loria.fr

Abstract

This abstract describes a contribution to the 2013 KBGen Challenge from CNRS/LORIA and the University of Lorraine. Our contribution focuses on an attempt to automate the extraction of a Feature Based Tree Adjoining Grammar equipped with a unification based compositional semantics which can be used to generate from KBGen data.

Introduction Semantic grammars, i.e., grammars which link syntax and semantics, have been shown to be useful for generation and for semantic parsing. This abstract outlines an attempt to automatically extract from the KBGen data, a Feature Based Tree Adjoining Grammar which can be used for generation from the KBGen data.

Data The KBGen data consists of sets of triples extracted from the AURA knowledge base which encodes knowledge contained in a college-level biology textbook. Each set of triple was selected to be verbalisable as a simple, possibly complex sentence. For instance, the input shown in Figure 1 can be verbalised as[1]:

(1) The function of a gated channel is to release particles from the endoplasmic reticulum

Sketch of the Overall Grammar Extraction and Generation Procedure To generate from the KBGen data, we parsed each input sentence using the Stanford parser; we aligned the semantic input with a substring in the input sentence; we extracted a grammar from the parsed sentences provided with the input triples; and we generated using an existing surface realiser. In addition some of the input were preprocessed to produce a semantics more compatible with the assumption underlying the syntax/semantic interace of SemTAG;

[1]For space reasons, we slightly simplified the KBGen input and removed type information.

```
:TRIPLES (
(|Release-Of-Calcium646|
 |object| |Particle-In-Motion64582|)
(|Release-Of-Calcium646|
 |base| |Endoplasmic-Reticulum64603|)
(|Gated-Channel64605|
 |has-function||Release-Of-Calcium646|)
(|Release-Of-Calcium646|
 |agent| |Gated-Channel64605|))
:INSTANCE-TYPES
(|Particle-In-Motion64582|
 |instance-of| |Particle-In-Motion|)
(|Endoplasmic-Reticulum64603|
 |instance-of| |Endoplasmic-Reticulum|)
(|Gated-Channel64605|
 |instance-of| |Gated-Channel|)
(|Release-Of-Calcium646|
 |instance-of| |Release-Of-Calcium|))
```

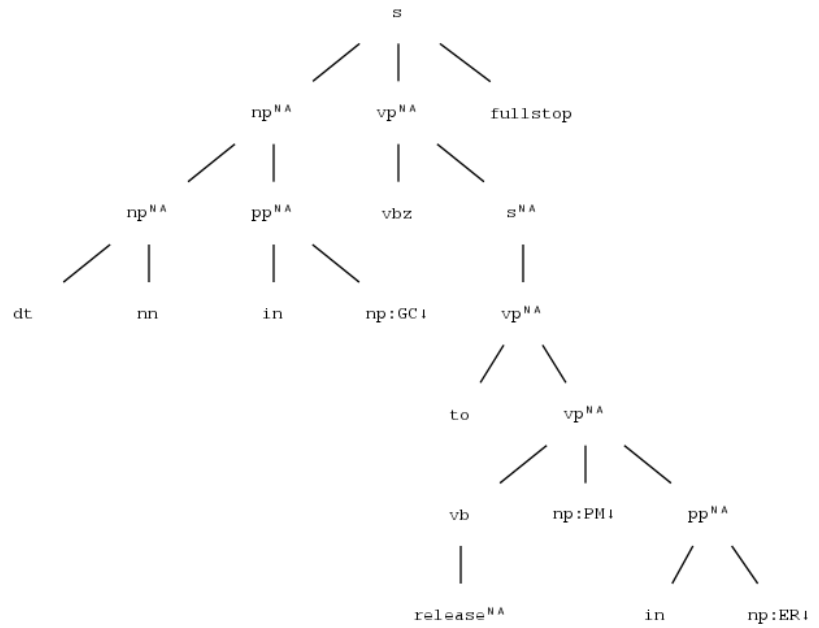
and a procedure was used to guess missing lexical entries.

Alignment and Index Projection Given a Sentence/Input pair (S, I) provided by the KBGen Challenge, we match each entity and event variable in I to a substring in S. Matching uses the variable name, the name of the unary predicate true of that variable and the word form assigned to that predicte in the KBGen lexicon. Digits occurring in the input are removed and the string in the input sentence which is closest to either of the used units is decorated with that variable. Index variables are then projected up the syntactic trees to reflect headedness. For instance, the variable indexed with a noun is projected to the NP level; and the index projected to the NP of a prepositional phrase is project to the PP level.

Grammar Extraction Grammar extraction proceeds in two steps as follows. First, the subtrees whose root node are indexed with an entity variable are extracted. This results in a set of NP and PP trees anchored with entity names and associated with the predication true of the indexing variable.

Second, the subtrees capturing relations between variables are extracted. To perform this ex-

Proceedings of the 14th European Workshop on Natural Language Generation, pages 204–205,
Sofia, Bulgaria, August 8-9 2013.




```
name: release:Event_3arg__arg3_arg21_arg1_arg2_1:TEvent_3arg__arg3_arg21_arg1_arg2_1-3
semantics: [agent(E GC) base(E ER) has-function(GC E) object(E PM) release-of-calcium(E)]
```

traction, each input variable X is associated with a set of dependent variables i.e., the set of variables Y such that X is related to Y ($R(X, Y)$). The minimal tree containing all and only the dependent variables $D(X)$ of a variable X is then extracted and associated with the set of literals Φ such that $\Phi = \{R(Y, Z) \mid (Y = X \wedge Z \in D(X)) \vee (Y, Z \in D(X))\}$. This procedure extracts the subtrees relating the argument variables of a semantics functors such as an event or a role.

The extracted grammar is a Feature-Based Tree Adjoining Grammar with a Unification-based compositional semantics as described in (Gardent, 2008). Each entry in the grammar associates a natural language expression with a syntactic tree and a semantic representation thereby allowing both for semantic parsing and for generation. Figure 1 shows the tree extracted for the *release* predicate in Example 1.

Pre-Processing The parse trees produced by the Stanford parser are pre-processed to better match TAG recursive modeling of modification. In particular, the flat structure assigned to relative clauses is modified into a recursive structure.

The input semantics provided by the KBGen task is also preprocessed to allow for aggregation and to better match the assumptions underlying the syntax/semantics interface of SemTAG.

For aggregation, we use a rewrite rule of the form shown below to support the production of e.g., *A cellulose-synthase which contains a polypeptide and* ***two glucose*** *synthesizes cellulose.*.

$$R(X, Y_1), \ldots, R(X, Y_n), P(Y1), \ldots, P(Y_n) \Rightarrow R(X, Y), P(Y), quantity(Y, n)$$

For relative clauses, we rewrite input of the form *plays(X Y),in-event(Y E), P(E), R(E X)* to *plays(X Y),in-event(Y E), P(E), R(E Y)*. This captures the fact that in sentences such as *A biomembrane is a barrier which blocks the hydrogen ion of a chemical.*, the entity variable bound by the relative clause is that associated with *barrier*, not that of the main clause subject *biomembrane*.

Guessing Missing Lexical Entries To handle unseen input, we start by partitioning the input semantics into sub-semantics corresponding to events, entities and role. We then search the lexicon for an entry with a matching or similar semantics. An entry with a similar semantics is an entry with the same number and same type of literals (literals with same arity and with identical relations). Similar entries are then adapted to create lexical entries for unseen data.

References

Claire Gardent. 2008. Integrating a unification-based semantics in a large scale lexicalised tree adjoining grammar for french. In *COLING'08*, Manchester, UK.