Modelblocks Development Repository

November 6, 2019

Contents

1 Overview

Modelblocks ('modelblocks-release') provides toolchains for prediction and evaluation of linguistic phenomena, like delimitation and classification of recursive signs (e.g. parsing), and psycholinguistic phenomena, like reading latencies and observations of neural activity or blood oxygenation, for the purpose of evaluating theories of sentence processing. The Modelblocks development repository ('modelblocks-repository') provides toolchains for developing annotated resources in support of these experiments. These toolchains are organized into recursive sub-recipes for generating files with structured filenames, called **make items**, using the *make* build manager. Make items whose filenames conform to the Modelblocks sub-recipe syntax can be automatically generated by a *make* command from any directory containing a recipe file that includes pointers to relevant Modelblocks sub-recipe files. The resulting make items may contain automatically-generated theoretically-motivated predictions or reports that evaluate the linguistic and psycholinguistic accuracy of these predictions, and thus provide evidence for or against the theories that motivate them.

2 Annotation Format

Modelblocks supports annotation of nested linguistic signs associated with scoped and inherited elementary predication structures, implemented as cued associations in a distributed associative memory (Rasmussen and Schuler, 2018), using a generalized categorial grammar notation based on that of Nguyen et al. (2012) and coreference and scope annotations based on those of Schuler and Wheeler (2014).

2.1 Syntactic Markup

Each annotation file contains a sequence of trees or delimiters between paragraphs or discourses:

$$\langle\text{file}\rangle \longrightarrow \left(\left\{ \begin{array}{l} \langle\text{tree}\rangle \\ \text{`}\texttt{!PARABREAK}\text{'} \\ \text{`}\texttt{!DISCBREAK}\text{'} \end{array} \right\} \right)^{*}$$

A **tree** may be recursive bracketing around a nonterminal category label, followed by zero or more semantic markups, followed by one or more trees; or simply a token in a sentence:

$$\langle\text{tree}\rangle \longrightarrow \text{`}\texttt{(}\text{'}\ \langle\text{category-label}\rangle \Big( \langle\text{semantic-markup}\rangle \Big)^{*} \Big( \text{`\ '}\ \langle\text{tree}\rangle \Big)^{+}\ \text{`}\texttt{)}\text{'}$$

$$\langle\text{tree}\rangle \longrightarrow \langle\text{[-,:;.!?\$\%A-Za-z0-9]+}\rangle$$

As described in Nguyen et al. (2012), a **category label** may consist of two other category labels connected by a type-combining operator, or may consist of a pair of brackets around another category label, or may consist of a primitive category:

$$\langle\text{category-label}\rangle \longrightarrow \langle\text{category-label}\rangle\ \text{`}\texttt{-}\text{'}\ \langle\text{[a-z]}\rangle\ \langle\text{category-label}\rangle$$

$$\langle\text{category-label}\rangle \longrightarrow \text{`}\texttt{\{}\text{'}\ \langle\text{category-label}\rangle\ \text{`}\texttt{\}}\text{'}$$

$$\langle\text{category-label}\rangle \longrightarrow \langle\text{[A-Z][a-z]+}\rangle$$

2.2 Semantic Markup

A **semantic markup** may specify an operation or application of a grammatical inference rule, or specify an antecedent for an anaphor, or specify a scope parent, or specify a unary lexical inference rule to define the lexical semantics of a token.

$\langle\text{semantic-markup}\rangle \longrightarrow \text{`}\texttt{-o}\text{'}\ \langle\text{[A-Z]}\rangle$	(specifies rule 'a' variant if annotated on left child, 'b' if on right)
$\langle\text{semantic-markup}\rangle \longrightarrow \text{`}\texttt{-n}\text{'}\ \langle\text{[0-9]+}\rangle$	(specifies anaphor association to nuclear scope of numbered token)
$\langle\text{semantic-markup}\rangle \longrightarrow \text{`}\texttt{-n}\text{'}\ \langle\text{[0-9]+}\rangle\ \text{`}\texttt{r}\text{'}$	(specifies anaphor association to restriction of numbered token)
$\langle\text{semantic-markup}\rangle \longrightarrow \text{`}\texttt{-w}\text{'}\ \langle\text{[0-9]+}\rangle$	(specifies weak association to nuclear scope of numbered token)
$\langle\text{semantic-markup}\rangle \longrightarrow \text{`}\texttt{-w}\text{'}\ \langle\text{[0-9]+}\rangle\ \text{`}\texttt{r}\text{'}$	(specifies weak association to restriction of numbered token)
$\langle\text{semantic-markup}\rangle \longrightarrow \text{`}\texttt{-s}\text{'}\ \langle\text{[0-9]+}\rangle$	(specifies scope association to nuclear scope of numbered token)
$\langle\text{semantic-markup}\rangle \longrightarrow \text{`}\texttt{-s}\text{'}\ \langle\text{[0-9]+}\rangle\ \text{`}\texttt{r}\text{'}$	(specifies scope association to restriction of numbered token)
$\langle\text{semantic-markup}\rangle \longrightarrow \text{`}\texttt{-x}\text{'}\ \langle\text{[A-Z]+}\rangle$	(specifies a lexical semantic association by name)

Semantic markup can also specify patterns for rewriting tokens into lexical semantic graph equations:

$$\langle\text{semantic-markup}\rangle \longrightarrow \text{‘}\texttt{-x}\text{’}\ \langle\text{str-0}\rangle \left(\text{‘}\texttt{\%}\text{’}\ \langle\text{str-}n\rangle\right)^{N}\ \text{‘}\texttt{|}\text{’}\ \langle\text{repl-0}\rangle \left(\text{‘}\texttt{\%}\text{’}\ \langle\text{repl-}m\rangle\right)^{M\leq N}$$

$$\langle\text{str-}n\rangle \longrightarrow \langle\text{[-,:;.!?\$\%\^{}=A-Za-z0-9]*}\rangle$$

$$\langle\text{repl-}n\rangle \longrightarrow \langle\text{[-,:;.!?\$\%\^{}=A-Za-z0-9]*}\rangle$$

which match a sequence of $\langle\text{str-}n\rangle$ and wildcard characters ‘%’ to an input string, then form an output string by replacing each wildcard-matched text and instance of $\langle\text{str-}n\rangle$ with the corresponding wildcard-matched text and instance of $\langle\text{repl-}n\rangle$ (if it exists), preserving their order. The initial input string for these rewrites has the form of a **predicate constant**, consisting of the category label of the token (the preterminal symbol in the tree), followed by a colon, followed by the token (the corresponding terminal symbol):

$$\langle\text{predicate-constant}\rangle \longrightarrow \langle\text{category-label}\rangle\ \text{‘}\texttt{:}\text{’}\ \langle\text{[A-Za-z0-9]+}\rangle$$

The final output string produced by these rewrite rules is either an empty string (specifying no lexical semantic constraints), or a **graph equation**, which may consist of a conjunction of other graph equations, a specification that one path of association labels (read from left to right) leads to the same referential state as another path of association labels, or a specification that one path of association labels leads to a predicate constant:

$$\langle\text{graph-eqn}\rangle \longrightarrow \langle\text{graph-eqn}\rangle\ \text{‘\^{}’}\ \langle\text{graph-eqn}\rangle$$

$$\langle\text{graph-eqn}\rangle \longrightarrow \left(\langle\text{assoc-label}\rangle\right)^{+}\ \text{‘='}\ \left(\langle\text{assoc-label}\rangle\right)^{*}$$

$$\langle\text{graph-eqn}\rangle \longrightarrow \left(\langle\text{assoc-label}\rangle\right)^{+}\ \text{‘='}\ \langle\text{predicate-constant}\rangle$$

Association labels in these graph equations consist of single characters – numerals for participant associations, ‘s’s for scope associations, other lower case letters for inheritance associations, and upper case letters for scaffolding associations (used to assemble structures not cued from a token’s signified referential state):

$$\langle\text{assoc-label}\rangle \longrightarrow \langle\text{[0-9a-zA-Z]}\rangle$$

3 Files for Annotation

Annotations of syntactic and semantic information use different indentation styles, to simplify annotation. Syntactic annotation files use lisp-style indentation and do not contain token numbers, to facilitate automatic indentation in text editors such as `vim`. Semantic annotation files use shortened indentation and contain token numbers, to facilitate antecedent and scope annotation. Filenames of annotation files each consist of a specification of an **annotated corpus** followed by a specification of an **indentation style**:

$$\overbrace{\left\{\begin{matrix}\texttt{srcmodel/wikisemC1.casp}\\ \vdots\end{matrix}\right\}}^{\text{annotated corpus}}\overbrace{\left\{\begin{matrix}\texttt{.annot-syn}\\ \texttt{.annot-sem}\end{matrix}\right\}}^{\text{indentation style}}$$

4 Release Files

Modelblocks produces the release version of annotated corpora by combining the syntactic markup from corresponding `.annot-syn` files with the semantic markup from corresponding `.annot-sem` files:

$$\overbrace{\left\{\begin{array}{c}\texttt{srcmodel/wikisemC1.casp}\\ \vdots\end{array}\right\}}^{\text{annotated corpus}}\texttt{.toktrees}$$

5 Update Files

Modelblocks can produce updates for `.annot-syn` and `.annot-sem` files from corresponding release files:

$$\overbrace{\left\{\begin{array}{c}\texttt{srcmodel/wikisemC1.casp}\\ \vdots\end{array}\right\}}^{\text{annotated corpus}}\texttt{.fromtoktrees}\overbrace{\left\{\begin{array}{c}\texttt{.annot-syn}\\ \texttt{.annot-sem}\end{array}\right\}}^{\text{indentation style}}$$

These updates can be incorporated to annotation by making these files and then renaming them (e.g. moving them with `mv`) to remove the `.fromtoktrees`.

6 Report Files

Modelblocks can also produce reports of formatting errors:

$$\overbrace{\left\{\begin{array}{c}\texttt{srcmodel/wikisemC1.casp}\\ \vdots\end{array}\right\}}^{\text{annotated corpus}}\texttt{.discgraphs.formerrors}$$

renderings of discourse graphs:

$$\overbrace{\left\{\begin{array}{c}\texttt{srcmodel/wikisemC1.casp}\\ \vdots\end{array}\right\}}^{\text{annotated corpus}}\texttt{.discgraphs.pdf}$$

and translations of discourse graphs into lambda-calculus expressions:

$$\overbrace{\left\{\begin{array}{c}\texttt{srcmodel/wikisemC1.casp}\\ \vdots\end{array}\right\}}^{\text{annotated corpus}}\texttt{.discexprs}$$

References

Nguyen, L., van Schijndel, M., and Schuler, W. (2012). Accurate unbounded dependency recovery using generalized categorial grammars. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING '12)*, pages 2125–2140, Mumbai, India.

Rasmussen, N. E. and Schuler, W. (2018). Left-corner parsing with distributed associative memory produces surprisal and locality effects. *Cognitive Science*, 42(S4):1009–1042.

Schuler, W. and Wheeler, A. (2014). Cognitive compositional semantics using continuation dependencies. In *Third Joint Conference on Lexical and Computational Semantics (*SEM'14)*.