

Grammar induction: User manual

James Cussens and Stephen Pulman

October 15, 2003

1 State

We have the following dynamic predicates which define the state of the system (in rough order of importance):

`cmp_synrule/3` current grammar

`cmp_synword/2` current lexicon

`rule/4` `rule(Id,CoveredSentences,CompiledRule,Info)` stores induced rules as follows:

`Id` An `Id` that matches the one found inside `CompiledRule`.

`CoveredSentences` Ids of sentences that would be parsed correctly were the rule `CompiledRule` added to the grammar.

`CompiledRule` The rule itself.

`Info` Information about the syntactic properties of the rule currently a list with gap status, head status and 'score'

These are rules that *might* be added to the grammar, they are not actually in the grammar.

`sentence/3` `sentence(SId,Sentence,QLF)` stores an indexed set of currently unparsable sentences, together with the QLF (quasi-logical form) they should render. QLF = sigma for unannotated sentences

`edge/8` `edge(SId,Identifier,Origin,LeftVertex,RightVertex,Category,Needed,Contents)` stores the edges for sentence `SId` generable using the current grammar

`last_edge/2` `last_edge(SId,LastEdge)` stores the id of the last edge stored in `edge/8` for the sentence with the id `SId`

`use_needs_for_rules/0` If asserted, needed edges, as well as actual edges are used to generate the RHS of induced rules.

`sentence_ctr/1` Contains the number of the last read-in sentence.

`rule_ctr/1` Contains the number of the last generated rule.

`verbose/0` If asserted, a commentary is sent to standard output when the other predicates are executing.

`strict_cover_testing/0` If asserted produced QLFs have to be syntactic variants of correct QLFs for a cover test to be deemed successful.

`rhs_length_limit/1` Limit on length of the RHSs of induced rules. Used in `make_rule/1`, since there it prevents infinite looping due to constituents of zero length, which are now allowed

2 Commands

Here are ‘commands’ which a user would invoke. The main ones used are `read_sentences/1`, `generate_rules_from_sentences/0`, `order_by_coverage/0` and `display_rules/0`.

2.1 Input/Output

`read_sentences/3` `read_sentences(InputFile,ParsableFile,UnparsableFile)` reads from either an input file of unannotated sentences stored as list, or annotated sentences stored with `parse/2` and indexes them, asserts them (using `sentence/3`) and divides them between parsable and unparsable, sending each to be output on the appropriate file. All current rules (if any) have their coversets updated

`read_sentences/1` Like `read_sentences/3` without the output

`read_rules/1` `read_rules(InputFile)` reads rules (saved as `cmp_synrule`) clauses from `InputFile` and asserts them and then cover tests them and computes score.

`write_sentences/1` Just writes them out unindexed.

`write_rules/1` Just writes them out unindexed

2.2 Rule generation

`generate_rules_from_sentences/0` Run `generate_rules_from_sentences/1` on each sentence in `sentence/3`.

`generate_rules_from_sentence/1` `generate_rules_from_sentence(SId)` uses sentence `SId` to generate rules, and then stores them with cover set and info (using `rule/4`) if not already found.

`lggify/0` Run `lggify/2` on all pairs of rules. Fails if no new rules can be generated.

`lggify/2 lggify(RuleId1,RuleId2)` produce a good rule from two rules or fail if not possible. Rulebase is updated with newly created rule

`[compactify/1] compactify(+RuleId)` produces a new rule of the form eg `vp -> vp mod` from a rule of the form `vp -> vp mod mod`

2.3 Modifying the grammar

`commit_to_rule/1 commit_to_rule(RuleId)`

1. Adds the named rule to the grammar
2. Removes the rule from the rule base
3. Removes all sentences that can now be parsed correctly from `sentence/2`
4. Reparses all rules in rule base to get new cover

2.4 Display

`display_rules/0` Runs `display_rule/1` on all rules

`display_rule/1` Displays human-readable information about the named rule in the rule base

`order_by_coverage/0` Reorders the `rule/4` predicate so that those with highest coverage are returned first by eg `display_rules/0`