Interactive pedagogical programs based on constraint grammar

Lene Antonsen, Saara Huhmarniemi, Trond Trosterud

http://oahpa.uit.no

Abstract

We here present a set of interactive parser-based CALL programs for North Sámi. The programs are based on a finite state morphological analyser and a constraint grammar parser which is used for syntactic analysis and navigating in the dialogues. The analysers provide effective and reliable handling of a wide variety of user input. In addition, relaxation of the grammatical analysis of the user input enables locating grammatical errors and reacting to the errors with appropriate feedback messages.

Background and pedagogical motivation

The pedagogical programs in OAHPA! are based upon three pre-existing language technology resources developed at the University of Tromsø: a morphological analyser/ generator, a CG parser for North Sámi and a number word generator compiled with the Xerox compiler xfst.

The main goal of the development of OAHPA! was to develop a language tutoring system going beyond simple multiple-choice questions or string matching algorithms, with free-form dialogues and sophisticated error analysis. Immediate error feedback and advice about morphology and grammar were seen as important requirements for the program.

Pedagogical lexicon

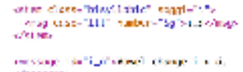
The OAHPA! programs share a set of common resources: a pedagogical lexicon and a morphological generator that is used for generating the different word forms that appear in the programs. The dialectal variation is taken into account in the lexicon as well as in the morphology. The lexical entry for *monni* "egg" is given below:

The morphological properties of words are used when making a detailed feedback on morphological errors. If the user does not inflect the lemma correctly, she can ask for hints about the inflection, and try once more, instead of getting the correct answer straight away.

Morphological feedback

The feedback messages are determined by the combination of morphological features in the lexicon and the inflection task at hand. The morphological specification below gives a rule stating that there is a vowel change in illative singular for bisyllabic nouns that end with the vowel *i*. The corresponding feedback message instructs the user to remember the vowel change.

The system-internal representation of *monni* states it is a bisyllabic i-stem, which triggers i > á change in illative.



The user types the errouneous *monnii*, and gets feedback from the machine. A correct answer gets green colour.

Handling dialectical variation

When generating sentences or providing the correct answers for the user, we allow only normative forms in the chosen dialect. On the other hand, the live analyser used for the analysis of the user input accepts all correct dialect variants of the same grammatical word. We compile one normative but variation-tolerant transducer for analysing the input, and one strict one for each dialect for sentence generation.

In the source code, forms are marked as missing in certain dialects (the default being that all forms occur in all dialects). Below is an example of dialectal variation in the comparative inflection. The resulting transducers give *stuorát* for the KJ dialect and *stuorit* for the GG one, of the adjective *stuoris* "big".

Sentence generation in the QA game Vasta

One of the main goals of the programs in OAHPA! is to practice language in natural settings with variation in the tasks. In order to provide variation in programs that involve sentential context we implemented a sentence generator. The sentence generator is used in the morphology in sentential context program (Morfa-C), and for generating questions to the QA drill (Vasta).

Conclusion

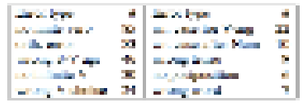
By using a sloppy version of the syntactical analyser for North Sámi, combined with a set of error-detection rules, we have been able to build a flexible CALL resource. The programs are modular, and the modules may be improved by adding more materials -- words, tasks, dialogues, levels, words from textbooks. The CG parser framework was originally chosen as parser framework for Sámi due to its extraordinary results for free-text parsing. The present project has shown that CG is well fit for making pedagogical dialogue systems as well.

Evaluation

The overall evaluation shows that the students answer correctly slightly half of the time. By far the most popular program is the basic morphological drill (but the interactve programs have been logged for a couple of days only).

Evaluating Sahka errors

The 322 logged Sahka errors are distributed along the following lines:

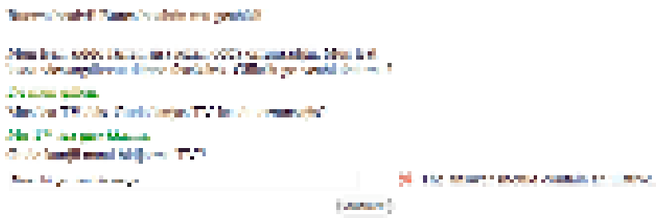


CG-parser in live analysis in the interactive programs Vasta and Sahka

We have chosen not to use multiple-choice, but rather let the student formulate her own answer. To a certain question one may give many kinds of acceptable answers. In Sámi one may change word order, and also add many kinds of particles.

We use a ruleset file which disambiguates the student's input only to a certain extent, because there will probably be grammatical and orthographic errors. The last part of the file consists of rules for giving feedback to the student's grammatical errors, and rules for navigating to the correct next question of in the dialogue, due to the student's answer.

The system question and student answer are analysed together, delimited by the boundary marker ^qst. They first get a morphological analysis (left) , and are then disambiguated, and, if possible, assigned an error tag or a navigation tag.



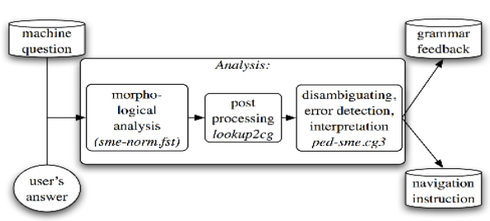
Grammar feedback

The system may give feedback, as shown here .....

Here comes an explanation to the example.

&grm-missing-Ill

Schematical view of the process



Navigation

Navigating inside the dialogue is implemented in CG-rules. The user input is tagged during analysis with information on whether the answer is interpreted as affirmative or negative. In addition, a special tag indicates whether the sentence contains some information that should be stored for the following questions or utterances. The program is thus able to store simple information such as the student's name, place where she lives and for example the type of her car and use this information in tailored utterances.