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.

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".

```
+A+Comp:i%>X4b BUStem ; ! NOT-KJ
+A+Comp:á%>X4b BUStem ; ! NOT-GG
```

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).

```
<q level="2" id="go_ikte">
  <qtype>PRT</qtype>
  <question>
    <text>MAINV go SUBJ ikte</text>
    <element id="MAINV">
  <grammar tag="V+Ind+Prt+Person-Number"/>
  <sem class="ACTIVITY"/>
    </element>
    <element id="SUBJ">
  <sem class="HUMAN"/>
  <grammar pos="N"/>
    </element>
  </question>
</q>
```

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

Program	Correct	Wrong	Total	%
Morfa-S	6920	6323	13243	52.3
Leksa	5659	4248	9907	57.1
Numra	3086	2512	5598	55.1
Morfa-C	1349	1613	2962	45.5
Sahka	322	322	644	50.0
Vasta	19	102	121	15.7
Total	17355	15120	32475	53,44

Error type	#	Error type	#
no finite verb	85	wr. case for V-arg	22
orth. error	83	wr. case after Num	10
wrong S-V agr	46	wrong lemse	9
no infinite V	30	no postposition	6
wrong V choice	24	wrong word	7

Evaluation the CG feedback for Sahka

Sahka feedback we test *precision* (correctly identified errors/all diagnostised errors) *recall* (correctly identified errors/all errors), and *accuracy* (correct judgements/cases).
Precision = 0.8; Recall = x.y; Accuracy = z.w (N=XXX)

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.

Leksa — training the vocabulary

The pedagogical lexicon is used for vocabulary training as well as for grammatical training. Each Sámi lemma is translated into Norwegian or Finnish. The user may chose direction (to or from Sámi), and restrict the vocabulary according to the vocabulary of the most commonly used Sámi textbooks, or according to topic (e.g. food/drink, family, school, nature, ...). The program accepts a wide range of synonyms, but

Numra — training numbers

The Numra game offers training on number expressions, either from number symbols to linguistic representation, or vice versa. As our only program, Numra offers training for four different Sámi languages, Kildin, Inari, North, Lule and South Sámi. We hope to extend our coverage to more langauges for the other programs as well.

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 to the right:

```
:entry>
  <lemma>monni</lemma>
  <pos class="N"/>
  <translations>
    <tr xml:lang="nob">egg</tr>
    <tr xml:lang="fin">muna</tr>
  </translations>
  <semantics>
    <sem class="FOOD-GROCERY"/>
  </semantics>
  <stem class="bisyllabic" diphthong="no"
      gradation="yes" soggi="i" rime="0"/>
  <dialect class="NOT-KJ"/>
  <sources>
    <book name="d1"/>
    <book name="sara"/>
    <book name="algu"/>
  </sources>
:/entry>
```

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.

```
<stem class="bisyllabic" soggi="i">
  <msg case="Ill" number="Sg">i_á</msg>
</stem>

<message id="i_á">Vowel change i > á.
</message>
```

monni
monnii ✕ help
"monni" has even-syllabic stem and shall have strong grade. Vowel change i > á. the suffix -i.
monni
monnái

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 as feedback.

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.

Buorre beaivi! Hures boahtin mu geahčči!

Mun lean aiddo fárren sisa iežan ođđa orrunsadjái. Mus leat lossa viessogálvvut dáppe fenskáris. Gillešit go veahkehit mu?

De mun gillen.

Mus lea TV dás. Guđe lanjas TV lea dii orrunsajis?

Mu TV lea gievkkanis.

Guđe latnjii moai bidje mu TV?

Moai bidje dan hivssega. ✕ The answer should contain an illative.

Answer

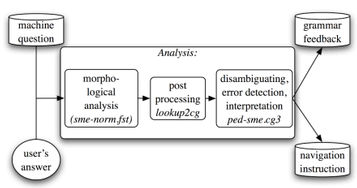
Grammar feedback

The system may give feedback, as shown here .....
Here comes an explanation to the example.
&grm-missing-Ill

```
MAP (&grm-missing-Ill) TARGET ("guhte") IF
(1 (N Ill) LINK *1 QDL LINK NOT *1 Ill OR
DOHKO OR Neg BARRIER S-BOUNDARY);

<message id="grm-missing-Ill">The answer
should contain an illative.</message>
```

Schematical view of the process



Navigating by regular expressions

The system asks "How old are you?"- The answer is analysed, and a regular expression is used to read the answer. Dependent upon the age span, the user is then directed to dfferent follow-up questions.

```
  # Picking the age
MAP (&dia-adult) TARGET Num (*-1 QDL LINK 0 (Man_boaris_dan_leat))
    (0 ("[2-9][0-9]"r)) ;
MAP (&dia-young) TARGET Num (*-1 QDL LINK 0 (Man_boaris_dan_leat))
    (0 ("[1][0-9]"r)) ;
MAP (&dia-child) TARGET Num (*-1 QDL LINK 0 (Man_boaris_dan_leat))
    (0 ("[1-9]"r)) ;

<utt type="question" name="How_old_are_you">
  <text>Man boaris dan leat?</text>
  <alt target="young" link="Do_you_go_to_school_young"/>
  <alt target="child" link="Have_you_started_at_school_child"/>
  <alt target="adult" link="Do_you_work_adult"/>
  <alt target="default" link="Do_you_work_adult"/>
</utt>
```

```
"<Gude>"
   "guhte" Pron Interr Sg Acc
   "guhte" Pron Rel Sg Gen
   "guhte" Pron Rel Sg Acc
   "guhte" Pron Interr Sg Gen
"<latnjii>"
   "latnja" N Sg Ill
"<moai>"
   "mun" Pron Pers Du1 Nom
"<bidje>"
   "bidjat" V TV Ind Prs Du1
   "bidjat" V TV Ind Prt Pl3
"<mu>"
   "mun" Pron Pers Sg1 Gen
   "mun" Pron Pers Sg1 Acc
"<TV>"
   "TV" N ACR Sg Acc
   "TV" N ACR Sg Nom
   "TV" N ACR Sg Gen
"<^qst>"
   "^sahka" QDL gosa_bidjat_TV
"<Moai>"
   "Mo" N Prop Plc Sg Ill
   "mun" Pron Pers Du1 Nom
   "Moa" N Prop Plc Sg Ill
"<bidje>"
   "bidjat" V TV Ind Prs Du1
   "bidjat" V TV Ind Prt Pl3
"<TV>"
   "TV" N ACR Sg Acc
   "TV" N ACR Sg Nom
   "TV" N ACR Sg Gen
"<hivssegis>"
   "hivsset" N Sg Nom PxSg3
   "hivsset" N Sg Loc
   "hivsset" N Sg Gen PxSg3
   "hivsset" N Sg Acc PxSg3
"<.>"
   "." CLB
```

The CG-rules disambiguate and add grammar-error-tag (&grm-missing-Ill) and navigation-tag (&dia-hivsset):

```
"<Gude>"
   "guhte" Pron Interr Sg Acc &grm-missing-Ill
   "guhte" Pron Interr Sg Gen &grm-missing-Ill
"<latnjii>"
   "latnja" N Sg Ill
"<moai>"
   "mun" Pron Pers Du1 Nom
"<bidje>"
   "bidjat" V TV Ind Prs Du1
"<mu>"
   "mun" Pron Pers Sg1 Gen
"<TV>"
   "TV" N ACR Sg Acc
"<^qst>"
   "^sahka" QDL gosa_bidjat_TV &dia-hivsset
"<Moai>"
   "mun" Pron Pers Du1 Nom
"<bidje>"
   "bidjat" V TV Ind Prs Du1
"<TV>"
   "TV" N ACR Sg Gen
"<hivssegis>"
   "hivsset" N Sg Loc
"<.>"
   "." CLB
```

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.

In the example to the left the question is "In which room do we put the TV?" One of the alternatives for the navigation is due to the target tag being assigned because of the lemma hivsset ("WC"). The answer will be "That is not a good idea. Make a new try." The CG-rule:

```
MAP (&dia-hivsset) TARGET QDL IF (0 (gosa_bidjat_TV))
(*1 ("hivsset") BARRIER ROOMS)  ;
```

There are alternative links in the dialogue, one of them is due to the &dia-hivsset tag:

```
<utt type="question" name="gosa_bidjat_TV">
  <text>Gude latnjii moai bidje mu TV?</text>
  <alt target="hivsset" link="gosa_bidjat_TV">
  <text>Dat gal ii heive! Geahčča! oddasit.</text>
        </alt>
  <alt target="default" link="gosa_bidjat_beavddi">
<text>Moai gudde dan ovttas dohko.</text>
</alt>
</utt>
```

Every question has its own unique id, which is used in navigating between questions. In addition, the CG-rules may be tailored for specific questions, like in the rule above.

Age-tags are assigned with help of regex-rules to the answer to the question "How old are you?". The tags function as links for moving to the next dialogue branch tailored to student's age.