The potential of a parser in a language teaching program

Lastow, Birgitta

1995

Lund University, Dept. of Linguistics
Working Papers 44 (1995), 111–128

The potential of a parser in a language teaching program

Birgitta Lastow

Introduction

This paper reports on results from studying what a parser can do when combined with Hypercard on a Macintosh computer to form a pedagogical tool for language teaching purposes. The system was applied to Japanese, but the exercises could easily be altered to suit other languages as well. There are exercises for learning vocabulary, hiragana (see *Writing system* below) and grammar. The focus is on syntax and the exercises have an increasing degree of complexity.

By using a computer for language teaching, and not a book, you can produce very flexible exercises, which are easy to change, more varied and more interesting to use. You can use sound and moving objects. In the program to be presented, a parser is used. This novel feature has many advantages and, for instance, makes it possible comment on spelling or grammatical errors.

The environment

The language teaching program has two major parts: a Japanese parser (described in detail in Lastow 1994), written in LPA MacProlog 4.5, and a user interface (Apple Hypercard stack, version 2.2). The communication between the two parts is made possible by using Apple System software's Open Scripting Architecture (OSA). Both MacProlog and Hypercard can send and receive OSA-defined messages called Apple Events. This communication is hidden from the user. The interface needed for Prolog to communicate with Hypercard was developed by Stephen Cooper at Uppsala University in co-operation with Nicky Johns at LPA Prolog, and is included in the parser. The Japanese characters have been obtained by using the software 'Japanese Language Kit' within the Swedish system version 7.1.

The parser

The Japanese parser is written in Prolog with a particular type of grammar formalism called 'Appending X-bar Grammar' or 'AXG' (cf. Sigurd & Lastow 1993, Sigurd 1994), which is based on X-bar syntax. The main idea of the AXG formalism is that larger (higher bar value) constituents are obtained by appending smaller constituents to each other, and in the end they will form a sentence. A typical AXG rule would be the following, where a single noun bar 1 is prefixed by an adjective bar 1, resulting in a complex noun bar 2:

```
jlexg(n,[B,B1],_,_,_,2,_,_,_) -->
        jlexg(a,B,_,_,_,1,_,_,_),
        jlexg(n,B1,_,_,_,1,_,_,_).
```

The second argument is the meaning representation, *[B, B1]*, of the noun bar 2, and it is the sum of the meaning representations of its parts, *B* and *B1*, respectively. This representation is useful when the parser is used in machine translation, but is not used here. The generative arrow in the grammar rule is part of the Definite Clause Grammar (DCG) definition in Prolog (cf. Gazdar and Mellish 1989).

The parser consists of three distinguishable parts – the grammar, the lexicon and a morphology module. The most complete section of the parser is the morphology module, which consists of rules that conjugate all Japanese verbs and a table of endings that are used for conjugation. The morphological rules are bi-directional, i.e. it is possible to either conjugate an infinitive form or to obtain the infinitive form from a conjugated form.

The parser can analyse core sentences containing subject, object, predicate (verbal or adjectival) and some different types of adverbs, but it still lacks essential syntactic patterns. The noun phrases may contain a demonstrative adjective, an adjective and a noun or a relative clause preceding a noun. The verbal predicate contains the main verb and a number of auxiliaries showing durative aspect, perfective, politeness, negation and past tense. The adjectival predicate only contains the last three auxiliaries. Only one type of subclause has been included. The lexicon is limited and is a test-tool, rather than a well-developed part of the parser. The reference book for Japanese grammar was mainly Martin 1987. The Japanese parser was developed as part of the Swetra project (cf. Sigurd et al. 1990) at the Department of Linguistics at Lund University, Lund, Sweden. It was not developed specifically for the purpose of Japanese language teaching, but its division into several separate (bar) levels, turned out to be useful when teaching parts of sentences. Better adaptation of

the parser for the teaching purpose may increase the processing speed when used together with the Hypercard user interface.

The parser can be used to check grammaticality. But it cannot tell if a sentence is semantically correct or not. It cannot distinguish between natural and less natural sentences.

In comparison with the parser presented in Lastow 1994, some new routines have been added, giving comments on errors. These routines are used in the third sentence exercise and are described in the next section.

Comments on errors

In sentence exercise number three the parser is used for giving comments on the errors that the user makes, when typing a complete sentence (see also figures in *Sentence exercises* below). The number of possible comments are rather limited at present. The parser starts with checking the spelling and determines if the combination of letters are possible in Japanese. This results in the comments ‘Spelling is OK’ or ‘Spelling is wrong’. It is necessary to check the spelling first (preparsing), as the grammar rules will judge a sentence to be ungrammatical when the parser fails to parse the sentence, and there is no difference between failure due to misspelt words and failure due to ungrammaticality. The spell-checker judges the words to be spelt correctly if the given word is in the lexicon, if it is a particle or if it is a conjugated verb or inflected adjective. Thus, a correctly spelt word that is not in the lexicon will be judged to be misspelt. Although an incorrect conclusion, this has the desired effect that the processing will be interrupted, since the parsing would fail anyway.

The comments on grammatical errors that can be given are ‘Subject, Topic or Agent particle missing’, ‘Missing or wrong particle before the noun’, ‘Wrong form of adjective’, ‘Wrong form of adjective in the last part.’, ‘Object particle missing or verb phrase incomplete.’, ‘Wrong predicative construction’, ‘Object particle or predicate missing.’ (for examples, see *Sentence exercises* below). The last ones are implemented in an inefficient way. When trying to parse an ungrammatical sentence, the parser backtracks through all possible solutions, before saying whether the sentence is grammatical or not. This amount of backtracking needs to be limited, otherwise it will be very time consuming to get a comment.

The programming technique which has been used for making comments, involves making assertions (additions of new clauses during execution) to the Prolog internal database, created when the Prolog code is interpreted on start-

up. The assertions are made on strategic points in the grammar, so that it is possible to determine what went wrong by looking at how far the parsing succeeded. An example of a grammar rule with assertions is shown below:

```
jlexg(n,[B,B1],_,_,_,2,_,_,_) -->
        jlexg(na,B,_,fin,_,1,_,_,_),
        {one((retract(na(_)), assert(na(1))))},
        [na],
        {one((retract(na_part(_)),
        assert(na_part(1))))},
        jlexg(n,B1,_,_,_,1,_,_,_).
        /* Nouns may be prefixed by na-type Adjectives
        resulting in a Noun bar 2 */
```

If the user tries to combine a *na*-type adjective with a noun, but leaves out the particle *na*, the parsing will fail after the first assertion of ‘na(1)’ and ‘na_part(1)’ will never be asserted. If the user does put in the particle between the adjective and the noun, on the other hand, both assertions will be made and the processing continues at a higher (bar) level.

Writing system

The Japanese writing system includes two syllabaries and a set of Chinese characters, called *kanji*. The two syllabaries contain 48 characters each and are called *hiragana* and *katakana*, respectively. Each character represents a syllable or a mora. Kanji characters represent lexical morphemes and are used for words or word stems. All three kinds of characters are used, but for different purposes. Hiragana characters are used for writing endings and some shorter words. Katakana characters is used for loan-words only (cf. Svantesson 1991). The Japanese characters can be transcribed with the Roman alphabet using, for instance, the Hepburn system.

What is Hypercard?

The basic concept in Hypercard is the ‘stack’. A stack is a Hypercard document and it consists of an unlimited number of ‘cards’. The cards may have a ‘background’ that is shared by all cards in the stack and a ‘foreground’ that is unique for each card. The cards are similar to cards in an ordinary card file. You can type on them, add pictures to them, add cards to the card file and remove them etc. You can record and play back sound. You can also add instructions to a card (script). An instruction could for example be ‘Go to next card’. Instructions are often linked to a button. When the user presses the button, the instruction is executed and as a result, in this case, the

next card will be visible. The buttons enable the user to navigate through the cards in the stack(s). If you want to display a text or if the user should be able to enter text on a card – for example if the stack is an address register – you create fields that either contain text or are empty rectangles where text can be entered. By adding buttons, fields, pictures and instruction to your cards you create a Hypercard program.

The user interface

The user interface consists of a series of exercises (cards). The cards contain a number of different objects – pictures, text fields and buttons. Operations on objects are performed by executing scripts, which are attached to buttons, fields or cards. The scripts can send messages to the environment outside Hypercard (see description above) and to the Hypercard environment itself. For example, when a button is pressed, the script attached to that particular button is executed. The script performs a number of operations, among other things it may send a message to Prolog consisting of a Prolog goal, to be evaluated by Prolog. The result of the evaluation is returned to Hypercard and displayed in a suitable way. The exercises are described in detail in the following sections.

Lexical items

Lexical items, words or phrases, can be taught by using pictures. The learner looks at the picture, reads the corresponding word, written in kana, Roman letters or kanji. By using the menu ‘Preferences’ he can turn on and off either representation of the word independently. He can also listen to the word by clicking on the button ‘Say ...’. He can also learn new words by conclusion – if he sees a picture of one big horse and one small horse and the phrases underneath each picture consist of two words in each case and the latter is the same, he might be able to conclude that the last word signifies the horse and the first word a description of the size of the horse by contrasting the two. The conclusion process makes the learner think and stay active, which ought to have a positive effect on his learning. He can also check his conclusion by clicking on the text. This will cause the translation of the phrase to appear in a rectangle (see figure 1). The rectangle will disappear if he clicks on it. It would also be possible to make a direct link to a dictionary, so that the learner can look up the word he does not know in the dictionary by selecting that word. All this would not have been possible if the learner had been reading a book instead of using a computer. If one starts to turn pages and go and get a

Figure 1. Hiragana exercise.

dictionary to look something up, concentration is lost. When working with a computer it is easy to go to a dictionary module and it is just as quick to go back to practising.

Hiragana exercise

At present only a hiragana exercise is included, but a katakana exercise of exactly the same type is of course possible. The learner is given randomly selected hiraganas, written in Roman letters, when pressing the button 'New Exercise'. He is then supposed to identify the correct hiragana in the hiragana table and click on it. The computer will then tell him whether it was right or wrong. If he needs help he can retrieve a hiragana chart by clicking on the button 'Show table' (see figure 2). The chart can then be hidden again with the same button, now called 'Hide table'.

This exercise only improves the learner's ability to recognise hiragana, but it is of course necessary for the learner to practice writing them, too. However, the ability to recognise hiragana may shorten the time it takes to learn how to write them. To give the exercise more game-like appearance one could add a counter of correct and incorrect answers.

Verb morphology exercises

The two verb exercises are of a type often found in language learning programs. The learner is supposed to fill in a conjugated form of the given verb – in the first exercise polite style imperfect tense, called '*masu*-form', and in the second, plain style perfect tense, called '*ta*-form'. After completing the exercise, the learner can obtain the results by pressing the button 'Check'. The verb-forms written by the user are sent to the morphology part of the parser, which checks if they are correct or wrong and the result is displayed to their right (see figure 3).

It is also possible to get help when working with these exercises. If the user presses the 'Table' button, he will see a conjugation table that gives the conjugation rules for all possible stems or forms of all groups of Japanese verbs. The table provides a good overview of the Japanese verb system in a systematic way. To understand the table it is necessary to have some prior knowledge of how to use it. This explanation should be included in the grammar explanations (see *Further developments* below). By having this table

Figure 3: Verb exercise, *masu*-form.

Figure 4. Adjective+noun exercise.

easily accessible when doing the exercises, the learner may bother to check the rules, instead of guessing, when he has forgotten a pattern – in contrast to the situation when he has to go and find his book and look it up.

This type of exercise is a stereotype and not very thrilling, and the learner will probably get bored easily. It is already decided which verbs you are supposed to practise conjugating. At present it is possible to edit the given verbs, but if you enter a new verb, it is probable that you will select one that is not included in the lexicon and thus the parser would not be able to conjugate it. This problem will be discussed more below along with the adjective+noun and sentence exercises.

Adjective+noun exercise

The adjective+noun exercise is an improvement of the verb exercises. The learner is supposed to form a noun phrase consisting of an adjective (either an adjectival, called '*na*-type', or a true adjective, called '*i*-type') and a noun. If the adjective is an adjectival noun, the insertion of a particle *na* between the adjective and the noun is obligatory (see Martin 1987). The selection of words is limited by the words given in pull-down menus. In this way the learner can choose freely between the words in the lexicon and combine them as he likes. The number of possible combinations increases very rapidly as the lexicon increases. The noun phrases can be checked by the parser as explained above (see figure 4).

If the learner is not aware of the rules of noun phrase formation from the beginning it is possible for him to draw conclusions about them by looking at the results of his attempts. It would also be nice to be able to go to an explanation of the rules involved (perhaps to check the conclusion), just by clicking on a button and after reading the explanation go back to the exercise again. By limiting the choice of structures (in this case the choice is between the two types of adjectives), it is possible to use the same exercise for finding out the rule governing the formation of the possible structures as for practising.

Figure 6. Sentence exercise 2.

Sentence exercises

In the three sentence exercises, the freedom of word selection is increased, compared to the adjective+noun exercise. Again, the words available are limited by the number of words in the lexicon, but now the learner can work with full sentences.

The first exercise is a translation exercise dealing with predicative sentences, where the learner can investigate and compare the use of demonstrative pronouns, negation and past time in English and Japanese. It could also serve as an example of word order differences (see *Word order* below). Since the number of words in the 'Dem. pron', 'Copula', and 'Neg.' menus (see figure 5) is very limited, the learner can also use the exercise to learn new words by finding out what the Japanese correlate of the English sentence is, after he has understood the rules. Later, he can also use it to check the vocabulary he has already learned.

The parser cannot translate all basic sentence types, but predicative sentences are one of the possible types and was therefore selected to illustrate this exercise type. The translation is performed by using a common representation of the sentence content as an interlingua. The English sentence is parsed by the English grammar and thus a representation of it is produced. This representation is then fed into the Japanese module, which generates the Japanese target sentence. If the generated sentence is identical to the one given by the learner, the result 'Correct' will appear in the window (see figure 5).

The second sentence exercise deals with sentences with subject, object and verb (see figure 6). Again, the number of words is limited by the size of the lexicon. Sentences without subject are not allowed at present, even though the parser can handle them, as it would interfere with the error commenting (see below).

The third sentence exercise type (figure 7) is the most advanced. Here the learner will get comments on the mistakes he makes and not only the short and non-informative answer 'Wrong'. It is also possible to type any sentence the learner would like to test. At present, this means that the user needs to know how a sentence should be typed in, otherwise he may get a wrong answer. He would, for example, need to know that he must use the Roman alphabet (the Hepburn system) and not kana. This information should of

Figure 8. Word order, example 1.

course be included in the online documentation or help file (see *Further developments* below).

Comments on errors are possible by using the parser's knowledge of what sentences are grammatical. This would never be possible if all correct answers are stored in a long list. In that case the user may get comments on common errors that almost every learner would do. To correctly judge all sentences the parser would have to cover all possible sentences of Japanese, which it does not. (For a more detailed discussion of the commenting abilities of the parser, see *Comments on errors* above).

Word order

There are two cards explaining the difference in word order between English and Japanese – SVO and SOV respectively. These cards are not exercises, but examples of how motion can be used in explanations. On the first card (figure 8) there are two buttons, 'In English' and 'In Japanese'. When the learner presses the first button, rectangles representing 'Subject', 'Verb' and 'Object' will appear. When the rectangle named 'Verb' appears, it will flash three times and thereby catch the learner's attention and make him aware of the placement of the verb in relation to the other parts. When pressing the 'In Japanese'-button, the same thing will happen, but now the 'Verb' rectangle will appear last, according to the word order rules of Japanese.

The possibility of using moving objects in a computerised version of a language teaching material, may serve as an important tool for helping the student to focus on the most important parts of an explanation. The method can of course be applied to many other situations, e.g. to show how suffixes change when conjugating verbs. The idea of using moving objects in grammar teaching has been used to demonstrate relative clause formation in a program called 'Animated Grammar', developed at Ohio University (cf. Soemarmo 1986) and the importance of perceptual cues for learning relativisation has been documented in a study by C. Doughty 1992.

The second word order card (figure 9) tries to explain the same phenomenon, but in a different way. In the beginning the learner can see three rectangles containing the words 'The boy', 'is reading' and 'a book', respectively. When he presses the button 'In Japanese', the rectangles start to move and the two last rectangles (representing verb and object) change places.

Figure 9. Word order, example 2.

When the rectangles have found their correct order, their content is translated into Japanese. The original English sentence is also shown for comparison.

The line of thought here – besides the fact that motion will draw the learner's attention – is that by making the learner watch the rectangles change order, he might find it easier to understand that this is what he should do himself. It is also easier to remember a moving illustration like the one discussed here, than if reading a text giving the same piece of information.

Further developments

Grammar explanations

The addition of grammar explanations, at any time accessible through menus and links from cards, for example by using button or maybe hypertext (underlined text that is linked to another place in the text, to which you will come by clicking on the underlined text). In the same way, the reference to the conjugation table could be obtained. It should also be possible to go in the opposite direction – from grammar explanations to exercises. The grammar explanations would be a kind of reference grammar, and the menu (and its submenus) would be like a list of contents of the grammar book, which takes you to the section you would like to read.

Help

On-line help on, for instance, how to navigate through the cards, how to enter text, should of course also be available to the user who has no prior experience of the Hypercard environment. The help file should only contain the information that the user needs in order to be able to use the program in general, and nothing about Japanese. There should be a clear-cut distinction between the grammar explanations and the help file.

Faster parser

The parser is not optimised for the combined use of the parser together with Hypercard. The processing of the sentence exercises takes a long time at present, but this can be remedied by tailored solutions and faster computers.

Text and glossary

It would be helpful and elegant if the words appearing in the texts were linked to a Kanji dictionary, e.g. MacJDic by Jim Breen (Department of Robotics & Digital Technology, Monash University, Victoria, Australia, available by anonymous ftp to 133.39.16.66, National Institute of Genetics, Japan) and if a glossary to the texts were linked to the text and vice versa.

Kanji-kana-Roman letters conversion

It would of course be convenient if the texts in fields could easily be converted to and from kanji or kana and Roman letters and not as an on/off-option like the one available now. The matter of conversion of text is more or less a research area in itself. There are a lot of ambiguities. The word *sake* 'Ç≥ÇØ' could either be written like 'é' or 'ç¯', where the first kanji symbolises Japanese rice wine and the latter a fish. The conversion of the Roman transcription to kana and vice versa is unambiguous if the text is written with spaces between words, so you can tell the difference between the case when a word ends with *-na*, and the case when one word ends with *-n* and the next begins with *a-*.

Different levels of difficulty
If one could easily change between different levels of difficulty, e.g. by using a different number of kanji characters of varying difficulty, the program would be useful for learners of a wider range of Japanese language skills and for a longer period of time for one individual learner.

Conclusions

Using a parser in conjunction with the Hypercard user interface to form a language teaching program adds new possibilities to language teaching programs, but there are some disadvantages. At present, the most apparent disadvantage is the amount of time it takes to check those exercises that could have been checked without the parser, e.g. the first sentence exercise. In contrast, other exercises, e.g. the third sentence exercise, would not have been possible at all, without the parser. One big advantage with using a parser is that – especially for languages with a rich morphological system (if it is regular) – the amount of space needed to store the information you need to be able to tell which sentence is correct and which is wrong is very small, since instead of storing all possible correct sentences, you store rules that can generate all correct sentences possible, given the words in the lexicon. But again, there is a trade-off between speed and space. If the lexicon is relatively small, the number of correct sentences you can generate using those words are comparatively small and up to some point it might be faster to search through a list of all correct sentences instead of using rules.

Using a parser also gives other advantages. The exercises can be built in a more flexible way. There is no need to change the Hypercard part when e.g. the lexicon is expanded, since the exercises rely on the information in the parser part. It also enables the user to choose between all the words in the lexicon in every exercise, if that is what the programmer wants. The user can then choose what he himself would like to practice and not what some other person decided would be best for him to do. I think this makes the program more interesting to use and furthermore interesting to use during a longer period of time.

I think that the most important thing about a language teaching program (or any other program) is that it is easy and fun to use. It should be challenging, catch the user's interest somehow and let the user explore and investigate the features of the language in a way similar to the way of the child who out of curiosity explores the environment around him.

Using computers more and more is an overall trend in society that must not be underestimated, and I think computers definitely have a place in the

language teaching area too. The world university is now a fact – students and teachers need no longer be in the same place. Computers can never replace the teachers, but they can help, provided that suitable programs are developed. It is important to remember that the use of computers only adds an extra dimension to learning if you give the program features that books or classroom teaching cannot provide. By using interactive multimedia presentations, the learner is given an opportunity to influence what, when and at what speed he is going to learn. Some phenomena might be better illustrated in this way than when explained verbally. One example, as pointed out above, is the use of moving objects to make the user focus on what is important, e.g. when explaining various grammar rules.

I think that Hypercard provides an opportunity to create multimedia programs, even for non-programmers, although it has some drawbacks. The use of colour, for example, is not supported very well, but is said to be in the upcoming version, Hypercard 2.3. Hypercard can produce stand-alone applications (independent programs), which can be given free of charge to students.

I hope what I have discussed here has shown that it is fairly easy to develop small teaching programs in Hypercard and I also hope that it can inspire language acquisition researchers to start developing their own programs.

References

Doughty, Catherine. 1992. ‘Computer applications in second language acquisition research: Design, description, and discovery’. In Martha C. Pennington (ed.), *Computers in Applied Linguistics*, 127-54. Cleveland: Multilingual Matters.

Gazdar, Gerald & Christopher Mellish. 1989. *Natural language processing in PROLOG*. Reading, Mass.: Addison-Wesley.

Lastow, Birgitta. 1994. ‘Appending X-bar grammar (AXG) for a fragment of Japanese’. In Bengt Sigurd (ed.), *Computerized grammars for analysis and machine translation*, 99-117. Lund: Lund University Press.

Martin, Samuel E. 1987. *A reference grammar of Japanese*. Rutland, Vermont: Tuttle.

Sigurd, Bengt. 1994. ‘Fragments of a Swedish appending X-bar grammar (AXG)’. In Nils Jörgensen, Christer Platzack & Jan Svensson (eds.), *Språkbruk, grammatik och språkförändring. Festskrift till Ulf Teleman*, 219-28. Lund: Dept. of Scandinavian Languages.

Sigurd, Bengt, Mats Eeg-Olofsson, Barbara Gawroƒska & Per Warter. 1990. *SWETRA – A multilanguage translation system.* Praktisk Lingvistik 14. Dept. of Linguistics, Lund University.

Sigurd, Bengt & Birgitta Lastow. 1993. 'Fragments of a Japanese appending X-bar grammar (AXG)'. *Proceedings NLPRS '93*, 353-357. Fukuoka: Organizing Committee of Natural Language Processing Pacific Rim Symposium.

Soemarmo, Marmo. 1986. *Animated grammar*. Athens, OH: The Ohio University.

Svantesson, Jan-Olof. 1991. *Språk och skrift i Öst- och Sydöstasien.* Lund: Studentlitteratur.