A GENERALIZED LR PARSER FOR TEXT-TO-SPEECH SYNTHESIS

Per Olav Heggtveit

Telenor R&D
N-2007 Kjeller, Norway
per.heggtveit@fou.telenor.no

ABSTRACT

The development of a parser for a Norwegian text-to-speech system is reported. The Generalized Left Right (GLR) algorithm [1] is applied, which is a generalization of the well known LR algorithm for parsing computer languages. This paper describes briefly the GLR algorithm, the integration of a probabilistic scoring model, our implementation of the parser in C++, attribute structures, lexical interface, and the application of the parser to part-of-speech (POS) tagging for Norwegian.

Applied to a small test set of about 4 000 words this method correctly tags 96 % of the known words, which is close to the performance of other POS-taggers trained on large text databases [2] [3]. 85 % of the unknown words are tagged correctly, and the probability of choosing the wrong pronunciation of a word from lexicon is less than 0.1 %.

1. INTRODUCTION

A parser in a text-to-speech (TTS) system should be fast and robust enough to make an analysis, no matter how ill-formed the input may be. It should also be able to handle different attribute structures, relevant to syntax, morphology, prosody etc., depending on the exact application in a TTS system. If the same parser is applicable for several different tasks, a compact and elegant TTS system could be the result.

Several fast and efficient algorithms for natural language parsing have been proposed [4]. Chart parsers are fast and efficient, and frequently applied in TTS [5]. They are all based on a chart technique to keep a record of the successfully parsed substructures during the analysis. This eliminates the time consuming reinvestigation of structures. Well known chart parsers are Earley's algorithm [6], the CKY algorithm [4], and the Left Corner (LC) algorithm [7]. In an article by Shann [8], these algorithms, with some recent improvements, are compared to the GLR algorithm with respect to parsing time. In all tests the GLR algorithm performed better, particularly in cases of high ambiguity, where the GLR parser was about 1.9 times faster than the best chart parser (LC with top-down filtering).

These and some practical experiments with a public domain GLR parser were the reasons for choosing this parsing algorithm for our TTS system.

One possible application of the GLR parser is in POS-tagging. Several methods for POS-tagging or homograph disambiguation have been proposed the last years. Two of the most popular and well known are the Xerox stochastic POS-tagger [2], and E. Brills rule-based POS-tagger [3]. Both taggers apply manually POS-tagged text corpora to induce a lexicon, and for testing. Today there exists no large tagged Norwegian corpus, and therefore we propose to use general lexical and grammatical information available in a lexicon and a grammar to disambiguate the words.

2. THE GLR ALGORITHM

The easiest way to explain the GLR algorithm, without going into details is to look at an example. (For detailed description of the algorithm, see [1]).

```
s      =   np    vp    1.0;
vp     =   V     np    0.6;
       |   vp    pp    0.4;
np     =   DET   N     0.8
       |   np    pp    0.2;
pp     =   P     np    1.0;

DET:   the 0.6,   a 0.4;
N:     saw 0.1,  boy 0.4,  girl 0.3,  telescope 0.2;
V:     saw 1.0;
P:     in  0.7,   with 0.3;
```

Figure 1: A demo grammar (grammar1) with rule probabilities

Figure 1 shows a small and well known demo grammar with one ambiguous word (saw) and an ambiguous prepositional phrase attachment. A probability is associated with each rule and lexical word. We will return to the application of the probabilities in section 3. The grammar is precompiled into a LR parsing table as shown in figure 2.

The parsing table shows the states, the actions to be taken in each state according to the next input symbol, and where to go next. The action conflicts in states 4, 10 and 11 between shift and reduce, make this grammar unacceptable to an ordinary LR parser. These action conflicts reflect the ambiguity in the prepositional phrase attachment. The GLR parsing algorithm handles action conflicts or non-determinism by generalizing the parser stack into a *graph-structured stack* and searching this graph in a breadth first manner. This makes the GLR-parser able to parse any context-free grammar (CFG).

To show a snapshot of the graph-structured stack we parse the the following sentence:

STATE	ACTION DET N V P	GOTO s vp np pp DET N V P
0	sh	1 2 3
1	accept	
2	sh sh	4 5 6 7
3	sh	8
4	sh red(s = np vp)	9 7
5	red(np =np pp)	
6	sh	10 3
7	sh	11 3
8	red(np=DETN)	
9	red(vp = vp pp)	
10	sh red(vp = V np)	5 7
11	sh red(pp = P np)	5 7

Figure 2: A LR(0) parsing table for grammar1

The boy saw a girl with a telescope

After all the words have been read, and just before the final prepositional phrase is going to be reduced together with the noun phrase (*a girl with a telescope*) or the verb phrase (*saw (a girl) with a telescope*), the graph-structured stack is as shown in figure 3.

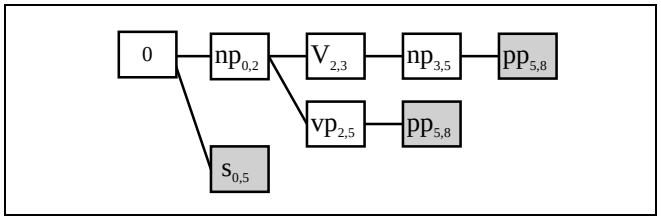

Figure 3: The graph-structured stack

The three dark nodes are the tops of the stack and the numbers are the sentence positions. The stack will reduce further until all sentence parses are found according to the grammar, or until no further reduction is possible. In this way the stack elegantly handles the ambiguity in the sentence.

The other structure that makes the GLR parser efficient is the *packed shared parse forest.* This is a method for sharing partial analyses between the nodes in the parse tree. The parse forest for the complete analysis of the sentence is shown in figure 4. The underlined parse node $vp_{2,8}$ is built in two different ways: Both $vp_{2,5} + pp_{5,8}$ and $V_{2,3} + np_{3,8}$ reduce to $vp_{2,8}$. This means that the ambiguity is locally packed and the nodes further up in the tree does only see the $vp_{2,8}$ node as one single node.

After this brief outline of the GLR algorithm we will now look closer at the computation of probabilities in our GLR-parser.

3. PARSING WITH PROBABILITY

Probabilistic parsing with a context free grammar require a probability for all lexical words or homographs and a probability for each rule in the grammar as in the demo grammar in figure 1.

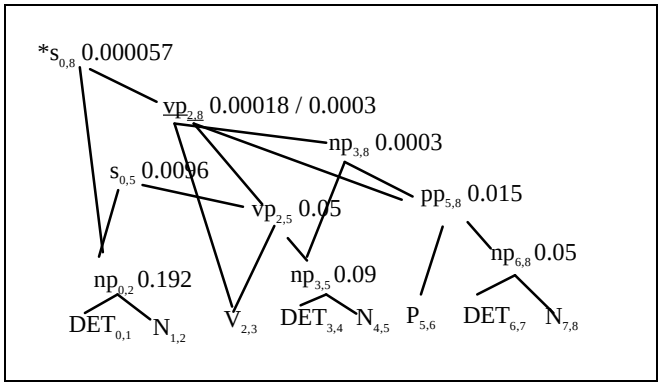

Figure 4: The packed parse forest with probabilities

The word probability is here the probability of the word given the lexical class (part-of-speech), *P(word | POS).* The probability of an edge (a non-terminal symbol spanning a certain word string) is computed according to the Inside algorithm[9]:

$$\beta(e) = P(rule(e)) \prod_{e_r \in rhs(e)} \beta(e_r)$$

P(rule(e)) is the rule probability of each rule as in figure 1, e is an edge as in figure 4, and $\beta(e_r)$ is the probability of each of the edges on the right hand side of a rule. This is a recursive algorithm which is terminated by the lexical word probabilities. To show how this work, we will apply the demo grammar on our demo sentence. We will start by calculating the probability of the first noun phrase (*The boy*) :

$$P(np_{0,2}) = P(np_{0,2}=DET_{0,1}\ N_{1,2}) * P(DET_{0,1}=the) * P(N_{1,2}=boy) = 0.8 * 0.6 * 0.4 = 0.192$$

The parser proceeds in the same way for all other possible edges and we get a probability for all edges as shown in figure 4. Edges that can be built in more than one way, will have a probability associated with each way. This makes it possible to select the constituents with highest probability in a sentence. The underlined edge $vp_{2,8}$ in figure 4 is built in two different ways: Both $vp_{2,5} + pp_{5,8}$ and $V_{2,3} + np_{3,8}$ reduce to $vp_{2,8}$. The first alternative gets the highest probability and is therefore preferred.

In this way we can select the parse tree with the highest probability and the part-of-speech tags that correspond to this parse. This requires that the grammar is capable of treating different POS-tags as different terminal symbols in the grammar.

When parsing free text with a high coverage grammar, there will always be word strings with no complete sentence parse. Even if the grammar allows sentences to consist of a string of unrestricted phrases, there will still be word strings with no

analysis. For these word strings a fallback solution is provided. The lexical probability of each homograph is compared to decide which to select. To do that we need the overall probability of each homograph, and not the conditional probability as in the demo grammar in figure 1. The estimated first order probability of a homograph is:

$$P(word) * P(POS \mid word) = P(word \mid POS) * P(POS)$$

This makes it possible to compare the probability of homographs directly. The same lexical probability could also be applied generally in the computation of the probability of higher order constituents. This will give us the sentence probability assuming that the words not necessarily have a certain POS-tag. All the rule probabilities of the "real" grammar described in section 4 are set by hand. This subjective "guess" is guided by the statistical formalism, which says that sum of the rule probabilities of a non-terminal node is 1. This restriction makes these pseudo-probabilities easier to tune than other penalty or score functions.

3.1 Lexical probabilities

Lexical probabilities for all words in our lexicon are determined by counting word frequencies for words in a large text corpus. Our corpus consists of 6.54 million words from news articles (4.9 mill), fiction (640 000) and non-fiction (1 mill). Our pronunciation lexicon which contains about 60 000 words, covers 85% of the words in the corpus.

The probability of each word in the lexicon is computed as the number of occurrences of the word divided by the total number of words in the corpus. Most words obtain a very low probability. To have the same information available in a more compact form, a logarithmic function is applied:

$$P = -10 \ln(n_w/N)$$

Here, n_w is the number of occurrences of each word, and N is the total number of words. This function gives a nice range of numbers from about 40 for the most common words to about 160 for the least common words.

A problem with this statistics is word ambiguity itself. When automatically counting occurrences in an untagged corpus, the occurrence number will represent the sum of all homographs of a word. This is a serious problem if we want to use this statistics to disambiguate words. But it is still important to compute the probability of each word because this will be the input for computing sentence probability.

In our pronunciation lexicon of 60 000 words there are about 16 000 ambiguous words, but in text-to-speech synthesis only the homographs with different pronunciations are of critical importance. Therefore we have chosen to manually adjust only the occurrence count of these words (about 2 600 words). This adjustment is a subjective selection of the degree of commonness of each homograph. For about 80 % of the ambiguous words this is easy. But for the difficult ones, we assume that they also would have similar occurrence counts in an objective count. When it is hard choose, the homographs are all common, or they are all uncommon.

The words in our lexicon which never occur in the text corpus get a small probability corresponding to an occurrence count of 1. The probability of an unknown word is 0.15 due to the coverage of the lexicon, which is 85 %.

4. THE PARSER

A handy starting point in the development of the parser was a public domain GLR parser found at FTP site ftp://*csd4.csd.uwm.edu/pub/regex/tomita.* This software contained a parser generator and the core parsing algorithm.

But the parser could not handle attribute structures and it had no interface to a large lexicon, which are both important for application in a TTS system. The parser generator and the parser were re-programmed in C++ to get this additional functionality.

4.1. Grammar specification

The parser generator takes a grammar specification as input and generates the LR parsing table like the one shown in figure 2. The grammar is specified in a DCG (Definite Clause Grammar) like style, and the attributes are written as C++ statements in brackets after each DCG rule. An example is shown in figure 5.

```
np      =  n              { lhs =new N(rhs[0],0.5); }
        |  det n          { lhs =new N(*rhs[0] | rhs[1],0.4); }
        |  PRN_POSS n     { lhs =new N(*rhs[0] | rhs[1],0.1);};
```

Figure 5: Example of grammar specification. Symbols: "|" means "or", {...} surrounds an attribute statement (in C++), where "lhs" is the left hand side attribute, and "rhs[0]" is the right hand side attribute number 0.

4.2 Attributes

The parser generator can be used with or without the attribute specifications. The left-hand side attribute in each rule is named *lhs* and the right-hand side attributes are named *rhs[n],* where n is the symbol number on the right-hand side of a rule (starting at 0). Manipulation of attributes is then simply done by using these attribute names in C++ functions in brackets after the rules. The attribute statement in the following rule:

*np = det n { lhs =new N(*rhs[0] | rhs[1],0.4);}*

says that the left-hand side attribute is constructed by initializing a new noun attribute structure with the unification (|) of the attribute values of the determiner and the noun on the right-hand side. The last term in the attribute statement is the rule probability.

Using ordinary C++ classes and functions to handle attributes, makes the parser very flexible in defining different attribute structures and functions according to the application.

4.3 Lexical interface

The interface between the lexicon and the grammar is shown in the following rule;

N:(s.m.eu) {lhs=new N(NOM,SG,MA,INDEF);};*

where a part-of-speech (POS) tag is written inside the parenthesis. A star (*) is used to handle wildcards. When a word is found in the external lexicon with this POS tag, it comes into the anlysis by this rule with the specified attribute values, and with the lexical probability.

4.4 A Norwegian grammar

A Norwegian grammar is developed for the GLR parser. Most of the rules are based on classical descriptions of Norwegian syntax found in [10]. The ambition of the work was not to get as high coverage as possible, but in general to cover the most common syntactic structures. The grammar consists of about 300 rules. Phrases are in general much better covered than sentential constructions. If the parser can't find a complete sentence, it will try to find the substructures with the highest probability.

5. PART-OF-SPEECH TAGGING

After grammar construction and the initial guess of rule probabilities, the rules and their corresponding pseudo-probabilities are tuned by hand from the analysis of a very small part of the text corpus described in section 3.1, only about 3 000 words.

As a test set, a text from a Norwegian prosody database [11] is applied (4 200 words). 425 words (10 %) in the test set are ambiguous, and 197 words (4.7 %) are unknown to the lexicon. This means that 85.3 % of the words are correctly tagged without any tagging algorithm. If 40 % of the homographs were randomly picked right, the rate of correctly tagged words would be 92.7 %. Our results are (percentage of correctly tagged words):

- All words in the test set: 95 % correct
- Lexically known words: 96 % correct
- Unknown words: 85 % correct (classified as nouns, adjectives, verbs or adverbs)
- 0.1% of the known words get wrong pronunciation from lexicon

A tag set of 150 different tags is applied, but there are only 40 terminal symbols in the grammar. The higher class resolution is achieved through the application of attributes for each terminal symbol.

The test results given here are not directly comparable to other results achieved by taggers for other languages. Our test set is much smaller due to the lack of a hand tagged Norwegian corpus. The Xerox POS-tagger applied on the American English Brown corpus gives 96% correct classification [2], and the same tagger ported to Swedish and applied on the Teleman corpus gives 95 % correct classification [12].

6. CONCLUSION

In this paper we have presented a new Generalized Left Right parser for a Norwegian text-to-speech system. The parser combines a fast and efficient parsing algorithm (GLR) with a probabilistic context free parsing model to be able to score different parses. So far we have applied the parser in part-of-speech tagging. Our results indicate that the parser is capable of tagging texts with about the same degree of correctness as other well known taggers. Manual tuning or automatic training of the rule probabilities can improve the results further.

The parser together with the Norwegian grammar also gives us the grammatical analysis which is important input data for the prosodic analysis in a text-to-speech system. Grapheme-to-phoneme conversion is another possible application of the parser.

7. REFERENCES

1. Tomita, M. and Ng, See-Kiong, "The Generalized LR Parsing Algorithm", In *Generalized LR Parsing*, pages 1-16, Kluwer Academic Publishers, Boston, 1991.
2. Cutting, D. et. al., "A practical part-of-speech tagger.", *Proc. 3. Conf. .on Applied NLP*, Trento, Italy, 1992.
3. Brill, E., "A simple Rule-Based Part-of-Speech Tagger." *Proc. 3. Conf. on Applied NLP*, Trento, Italy, 1992.
4. Winograd, J., *Language as a cognitive process, Syntax*, Addison-Wesley, 1983.
5. Russi, T., "A framework for morphological and syntactic analysis and its application in a text-to-speech system for German", In *Talking Machines: Theories, Models, and Designs, pages 163-182*, Elsevier Science Publishers, Amsterdam, 1992.
6. Earley, J., "An efficient context-free parsing algorithm", In *Comm. of the ACM*, 13(2), pages 94-102, 1970.
7. Slocum, J., "A practical comparison of parsing strategies", *Proc. 19th ACL*, Stanford, 1981.
8. Shann, P., "Experiments with GLR and Chart Parsing", In *Generalized LR Parsing*, pages 17-34, Kluwer Academic Publishers, Boston, 1991.
9. Charniak, E., *Statistical language learning*, The MIT Press, Cambridge, 1993.
10. Næs, O., *Norsk Grammatikk*, Fabritius, Oslo, 1979.
11. Ottesen, G., "Prosodidatabase for norsk". SINTEF DELAB report STF40 F94109 (in Norwegian), Trondheim, 1994.
12. Cutting, D. "Porting a Stochastic Part- of-Speech Tagger to Swedish", In *Proc. of 9. NODALIDA*, Stockholm, 1993.