Robust Parsing toward Spontaneous Speech Dialogues

Hiroaki SAITO
Department of Mathematics, Keio University
3-14-1 Hiyoshi Kouhoku-ku, Yokohama, 223, Japan
e-mail: hxs@nak.math.keio.ac.jp

1. An Error-recoverable LR Parser

An error-recoverable LR parser has been developed which can handle various errors and noise phenomena in spontaneous speech. A context-free grammar for the parser is phoneme-based to deal with phonemic and word information uniformly. The parser corrects such minor errors as phoneme insertion, deletion and substitution in the noisy phoneme sequence of an utterance by looking ahead the action part of the generalized LR table. Furthermore, the parser puts a dummy nonterminal symbol against a lengthy noisy portion by looking ahead the goto table. We call this technique "gap-filling."

Here we re-describe four techniques above in a formal way.

Situation: Suppose S is the top state of a parsing stack. Suppose X is the current input symbol. Action function 'action(S,X)' returns possible actions (shift, reduce, accept or error) by looking up the action table of a grammar. Multiple actions might be returned because of the generalized LR parsing. Suppose action(Si,Xi) returns 'error' in parsing an input sequence X1, X2, ..., Xn.

[symbol substitution]

Substitute a terminal symbol Yi for Xi on the condition that action(Si,Yi) does not return error.

[symbol deletion]

Ignore Xi on the condition that action(Si,Xi+1) does not return error.

[symbol insertion]

Insert a terminal symbol Yi just before Xi on the condition that action(Si,Yi) does not return error.

[gap filling (putting a dummy nonterminal)]

When state Si is a shift-able entry, consult the goto table to see if nonterminal D can be reduced at state Si and the new state is Sj. If that is the case, put a dummy nonterminal D onto the stack and go to state Sj.

The first three techniques are described for a single symbol substitution / deletion / insertion. Naturally, these techniques can be applied to manipulation of two or more symbols. In practice, however, that may expand search too much. Applying the gap filling technique loosely may also explode search. Thus a heuristics like "Two consecutive dummy nonterminals are not created" should be adopted.

The gap-filling technique enhances robustness of the parser in handling noisiness in the following points.

(1) A dummy nonterminal fills big missing constituents of the input which would yield no hypotheses without the gap-filling function.

(2) The gap-filling function enables an LR parser to perform reduce actions only when the action creates a definite high-score nonterminal. A dummy nonterminal is likely to be either an insertion noise or an unknown word.

2. Error-recovery Parsing against Speech Utterances

We evaluated effectiveness of our error-recoverable LR parsing against spoken utterances. We used a device which recognizes a Japanese utterance and produces its phoneme sequence. The hypothesized sequence contains recognition errors. Our parser explores all possibilities of substituted/inserted/deleted phonemes by looking up the confusion matrix, which was constructed from large vocabulary data. The confusion matrix is also used to assign a score to each explored phoneme, because the recognition device gives neither the alternative phoneme candidates nor the likelihood of hypothesized phonemes. The gap-filling function is incorporated into the parser in the following experiments. Parsing a phoneme sequence might sound less popular than parsing a word lattice in speech recognition. Because the parser builds a lattice dynamically in parsing the sequence from left to right using a context-free grammar which contains the dictionary, no static lattice is necessary.

125 sentences (five speakers pronounced 25 sentences) were tested in the domain called "conversation between doctors and patients." 111 sentences were parsed correctly [88.8 obtained as the top-scored hypothesis). 14 failed sentences can be classified into three groups:

(i) 4 sentences were parsed as the top-scored hypotheses with dummy nonterminals.

(ii) 6 sentences were parsed incorrectly in that the correct sentence did not get the highest score mainly because an incorrect nonterminal had a slightly higher score than the correct one.

(iii) 4 sentences were pronounced very badly.

5 sentences with unknown words were also tested. In all cases, the unknown word was detected.

These results show that our parser is very effective both for error-recovery and for detection of unknown words.

3. Interactive Speech Understanding

In making use of a two-way speech system, the parser interacts with the speaker against unidentified portions instead of conjecturing them. The user does not have to speak the whole sentence again; only the unidentified portion suffices. The reutterance of the unidentified portion is parsed efficiently by using the parse record of the first utterance. When the unidentified portion is found to be an unknown word after interaction, the word is incorporated into the system incrementally.

In the experiments of the previous section, the system asks the user in the 14 unsuccessful cases in the following ways.

(i) The parser asks the user to speak the unidentified portion again.

(ii) The parser asks the user to speak only that unclear portion in the same way as in (i) instead of producing a barely top-scored hypothesis. In the current implementation the parser asks the user which word is the correct one.

(iii) The user has to speak the whole sentence again.

Thus, we can say that interactive partial re-utterance is effective to resolve noisiness in the original utterance.

4. Caseframe Instantiation and Context-free Grammar

It is hard to write context-free grammar rules to cover free word order phenomena in spontaneous speech. Even the recognition performance gets improved, the utterance itself might be ungrammatical. Thus, it is not practical to pursue precise and specified syntactic rules.

An approach which combines a context-free grammar principle and caseframe instantiation has been claimed to be robust for ungrammaticality. This approach, however, is not effective in speech applications, because its strategy heavily leans on a word which specifies case and such word is often short and pronounced unstressed, especially in English. Caseframe instantiation also relies on a verb. Depending on such particular words is risky in speech applications. For instance, it would be easy to parse a written sentence 'I send a mail to Smith.' When that sentence is pronounced, the word 'to' is often hard to be recognized. This causes great difficulty for the system to understand the sentence, because that tiny word plays an important role semantically. If the task is small, we can effortlessly build knowledge where 'send' customarily puts 'to' before destination. As the task gets bigger, however, building such knowledge gets hard and time-consuming. Thus knowledge should be extracted from a corpus automatically.

5. Syntactic Co-occurrence and Context-free Grammar

Here we propose a method which handles syntax loosely and extracts the meaning of an utterance with the help of co-occurrence of words as an important semantic information. Co-occurrence information is obtained by parsing corpus sentences using a generalized LR parser.

We describe how to extract co-occurrence word relations from a corpus. A semantic function which detects word connectivity is attached to each rule. The notation used in the semantic function is similar to LFG(Lexical Functional Grammar)'s. 'X0' refers to the feature of the left-hand-side nonterminal symbol. 'X1' refers to the feature of the first [non]terminal in the right-hand-side sequence. 'X2' refers to the feature of the second [non]terminal in the right-hand-side sequence. '=' refers to assignment (unification, to be exact); for instance, '((x0 head) = (x1 form))' in rule (3) means that the actual word taken as noun is assigned to the head feature of NP. Function 'cow' recognizes co-occurrence relation between the parameters.

(1) S –> NP VP
    ((x0 head) = (x1 head)
    (cow (x1 head) (x2 head)))

(2) S –> S PP
    ((x0 head) = (x1 head)
    (cow (x1 head) (x2 head))
    (cow (x1 head) (x2 prep)))

(3) NP –> noun
    ((x0 head) = (x1 form))

(4) NP –> det noun
((x0 head) = (x2 form))

(5) NP –> NP PP
((x0 head) = (x1 head)
(cow (x1 head) (x2 prep))
(cow (x1 head) (x2 prep) (x2 head)))

(6) PP –> prep NP
((x0 head) = (x2 head)
(x0 prep) = (x1 form))

(7) VP –> verb NP
((x0 head) = (x1 form)
(cow (x1 form) (x2 head)))

In parsing 'I send a mail to Smith' using the grammar above, we can extract the following co-occurrence relations.

Rule (1) detects 'I, send' as <subject - verb> relation.

Rule (2) detects 'send, Smith' as <verb - object of prep> relation.

Rule (2) also detects 'send, to' as <verb - prep as modifier> relation.

Rule (5) detects 'mail, to' as <noun - prep> relation.

Rule (5) also detects 'mail, to, Smith' as <noun - prep - noun> relation.

Rule (7) detects 'send, mail' as <verb - noun as object> relation.

Note that this method can extract syntactic connectivity between words, while such conventional statistical measures as bigram or trigram simply scan surface connection.

6. Preliminary Experiments in Brown Corpus

It is practically impossible to write context-free grammar rules to parse all the corpus sentences to find co-occurrence relations. Here we consider a dilemma between the grammar size and ambiguity growth. If the grammar is written so as to cover unusual word orders as well, the number of ambiguity grows a lot as a side effect. As a result, important co-occurrence relations would be buried. Therefore, the grammar should be slim enough to extract basic English structures.

Preliminary experiments were performed against 1693 sentences in Brown Corpus. The grammar contains 292 rules, in which 95 terminal symbols which are described in the corpus as a tag and 38 nonterminals are used. A standard LR parser successfully parsed 43.1sentences. Many extracted relations with high occurrences contain such function words as pronouns. Some <preposition - noun> relations with regular nouns are 'for example' 'under direction' and 'in manner.' Note that 'for example' relation does not necessarily mean that

the particular phrase 'for example' is extracted. The actual phrase in the corpus might be 'for a good example' or 'for this example' or 'for example.' It simply means that 'for' is often used as a preceding preposition for 'example' to form a prepositional phrase.

Three-word co-occurrence relations are also extracted. For example, 'plays role of' and 'make most of' as <verb - noun as obj - prep> relation.

The error-recoverable LR parser, which skips one input symbol ([symbol deletion] technique) when parsing can not proceed further, successfully parses 67.8

7. Using Co-occurrence Relations in Parsing

Checking word co-occurrence in the semantic action in each rule enhances robustness of our error-recoverable generalized LR parser especially for ungrammaticalities and ellipses in spontaneous speech. Now we see how the extracted relations are utilized in parsing an erroneous sentence "I send a mail Smith." Parsing proceeds with no trouble against "I send a mail." In parsing the next word 'Smith,' the action function returns error. The same four error recovery mechanisms described in Section 1 are used in this parsing phase. Although actual manipulation depends on the grammar, [word insertion] technique inserts a preposition. If the terminal symbols of a grammar are lexicons, the actual prepositions are chosen. If terminals are parts of speech, a preterminal like *preposition is chosen. In either case, if the co-occurrence data show that 'to' is frequently used after 'send' in <verb - prep as modifier> relation, 'to' can be inserted with high probability.

Another example: 'He bought some cakes the shop' is syntactically grammatical. The error-recoverable parser also hypothesizes 'He bought some cakes at the shop,' which sounds much more natural. Co-occurrence relations can help the parser to infer which preposition is more appropriate.

Although co-occurrence information is not almighty, of course, such relation should be incorporated into the context-free grammar paradigm toward robust parsing.

Appendix: NLyacc Distribution

As a related research, a parser generator called NLyacc has been announced as a free software. NLyacc accepts an arbitrary context-free grammar (cyclic rules like A –> A are excluded) written in the yacc format and produces its generalized LR parser. Produced parsers are not currently equipped with the error-correction mechanism mentioned in Section 1. NLyacc, unlike yacc, is able to accept multiple values from a lexical analyzer, which is useful for handling ambiguous parts of speech of a word in natural language applications. Please contact nlyacc@nak.math.keio.ac.jp for distribution.

References:

Ishii, M., Ohta, K. and Saito, H. An Efficient Parser Generator for Natural Language. In Proceedings of COLING94, pp.417–420, 1994.

Keywords: speech understanding, spontaneous speech, parsing, interactive systems, word co-occurrence