Parsing Spoken Dialogues

Yue-Shi Lee
Department of Information Management
Ming Chuan University
Taipei, TAIWAN.CHINA
leeys@mcu.edu.tw

Hsin-Hsi Chen
Dept. of Computer Science and Information Engineering
Taiwan University
Taipei, TAIWAN, CHINA
hh_chen@csie.ntu.edu.tw

Abstract
This paper presents a spoken language processing system for parsing spoken dialogues. The differences between spoken data and written data are clarified. At first, we employ acoustic and prosodic cues to remove noises and identify the linguistic boundaries. Then a fast multi-level chunking-and-raising parser is used to analyze the more "clean" spoken data. The experimental results in parsing a Chinese spoken corpus show that the labeled precision and recall rates are 94.14% and 90.97%, respectively.

1. Introduction
Spoken dialogues different from conventional written texts in several aspects. First, more than one speaker may involve in conversations. One speaker may interrupt another speaker before s/he finishes her/his utterances. Second, speakers may repeat, add, replace, or even abandon some constructions in the utterances for some mental reasons. The high flexibility in conversation makes parsing spoken dialogues more difficult than parsing written texts. Consider the following example.

...nǐ yě %--
2.SG also (You also)
...nǐ yě bang ta [tiaojie].-
2.SG also help 3.SG adjust (You also help it to adjust.)

In this example, (nǐ yě) (you also) is repeated. The repeating constituent introduces noises in the context and makes training a spoken parser harder.

Several papers (Joakim N., et al., 1996; Lee and Chen 1999b; Schwartz, et al., 1994) use written training data for spoken language modeling. This paper adopts the similar concept. We first train a parser from a written corpus, and then adapt the parser to analyze the spoken dialogue.

2. System Architecture
Our architecture for parsing spoken dialogues consists of two components, namely Speech Component and Text Component. In the first component, the Syllable Recognition Module takes continuous speech inputs from users and then generates the corresponding syllable-strings (SYL). Similarly, the Prosodic Extraction Module analyzes continuous speech inputs and produces some useful prosodic information (PRO). The Repair Processing Module then detects and corrects speech repairs from SYL and PRO, and outputs the more fluent context (REP) to the Clause Boundary Identification Module. Instead of using acoustic segmentations as the basic analysis units, the Clause Boundary Identification Module segments REP into the more meaningful units such as clauses (CLA) based on PRO. The Homophone Disambiguation Module converts CLA into the corresponding character-strings (CHA). The CHA is the input of the Text Component.

The Text Component consists of three modules, i.e., Word Segmentation Module, Part-of-Speech Tagging Module, and Sentence Parsing Module. The Word Segmenta-
tion Module segments CHA into the word-strings (WRD). The Part-of-Speech Tagging Module assigns a lexical tag to each word of WRD and produces word-part-of-speech-strings (WRD-POS). The Sentence Parsing Module then generates the corresponding parsing trees (PAR) from WRD-POS. These three modules are the conventional text processing modules, but the features of spoken languages are considered. That is, the differences between the written languages and the spoken languages must be investigated in these three modules.

In summary, the speech component employs acoustic and prosodic cues to eliminate the noises embedded in the spoken data, and produces more “clean” data for the text component. Because these modules are pipelined, each module must be effective. For the limitation of paper space, the following sections only discuss the Sentence Parsing Module in more detail. The other modules can refer to Lee and Chen (1997a) for identification and correction of speech repairs, Lee and Chen (1999a) for clause boundary identification, Lee and Chen (1997b) for homophone disambiguation, and Lee and Chen (1999b) for tagging spoken corpus.

3. A Multi-Level Chunking-and-Raising Parser

The basic idea follows from our previous work (Chen and Lee, 1995). An input sentence, which is in terms of a part-of-speech sequence, is inputted to the chunking-and-raising parser. The chunking action of the parser groups some tags into chunks. The raising action then assigns a (syntactic) tag to each chunk and then generates a new tag sequence. Subsequently, this tag sequence is sent to the next chunking-and-raising cycle. The chunking and raising actions are repeated until no new tag sequence is generated.

A special type of grammar, called Constrained Grammar, guides each level of chunking-and-raising. It is learned from a treebank. Figure 1 shows an example.

**Figure 1. A Parsing Tree**

For the first level of chunking-and-raising, six constrained rules that appear on the lowest level of the parsing tree are extracted. They are shown below.

\[
\begin{align*}
( * ) \ np \ ( np ) \rightarrow & \ N' \\
( np ) \ np \ ( vadv ) \rightarrow & \ N' \\
( np ) \ vadv \ ( vn ) \rightarrow & \ RP \\
( vn ) \ np \ ( vi ) \rightarrow & \ N' \\
( np ) \ vi \ ( p ) \rightarrow & \ V' \\
( p ) \ nc \ ( * ) \rightarrow & \ N'
\end{align*}
\]

Two constraints enclosed in parentheses, i.e., the left and the right constraints, are added into each constrained rule. For example, the constrained rule, \(( np ) \ vi \ ( p ) \rightarrow V'\), has the left constraint “np” and the right constraint “p”. It means that chunk \{ vi \} can be raised to V’ when its left tag is “np” and its right tag is “p”. The other five rules have the similar interpretations. The asterisk marks the beginning or the ending of a sentence. These six rules are added into Level 1 Constrained Grammar (abbreviated as Constrained Grammar 1). Similarly, other constrained rules are also learned from this example. They belong to Constrained Grammars 2, 3, 4, 5, 6, 7 and 8, respectively.

After extraction, the same constrained rules in a con-
strained grammar are grouped. While the rules are conflicted, the rules having lower frequencies are removed from the grammar. A decision tree is used to model the remaining rules for efficiency consideration. The basic parsing algorithm is composed of several levels of chunking-and-raising, which are based on different constrained grammars. For simplicity, only one level of actions is demonstrated in Algorithm 1.

Algorithm 1. Chunking-and-Raising (PS)

\[
\begin{align*}
CP & = 1; \\
\text{While } CP & \leq N \text{ Do} \\
& \text{Found=false;} \\
& \text{For } CL=\text{Max \_Length} \text{ To } 1 \text{ Do} \\
& \quad \text{If } (CP+CL-1) < N \text{ Then} \\
& \qquad \text{If Search the Decision Tree for} \\
& \qquad \text{PS[CP-1] as Left Constraint,} \\
& \qquad \text{PS[CP-(CP+CL-1)] as Chunk and} \\
& \qquad \text{PS[CP+CL] as Right Constraint} \\
& \qquad \text{Is Success Then} \\
& \qquad \text{Output "["; Output Raised Tag;} \\
& \qquad \text{For Position=CP To } (CP+CL-1) \\
& \qquad \text{Output PS[Position];} \\
& \qquad \text{Next Position} \\
& \qquad \text{Output Raised Tag; Output "}"; Output Raised Tag;} \\
& \qquad \text{Found=true; Goto OK;} \\
& \quad \text{End If} \\
& \text{End If} \\
\text{OK:If } \text{Found Then} \\
& \quad CP=CP+CL-1; \\
\text{Else} \\
& \quad \text{Output PS[CP];} \\
& \text{End If} \\
& \quad CP=CP+1; \\
\text{End While} \\
\end{align*}
\]

In this algorithm, variable PS denotes the part-of-speech sequence, CP denotes the current position, Found tells out if a chunk is found in the decision tree, and CL denotes the chunk length. Assume that the length of the tag sequence is N. The symbol * is added to the beginning position (0) and the ending position (N+1) of the sequence. Because the larger chunks are preferred in this algorithm, it checks the plausible chunks from maximal length to 1. In this algorithm, a constrained rule can be applied when its left constraint, chunk, and its right constraint are satisfied. That is, a path is found in the decision tree.

4. An Error-Correction Mechanism

Because the errors on the lower levels will be propagated to the upper levels, how to get better performance on lower levels becomes an important issue. After examining the constrained rules, we find that some are over-generated. To avoid such overgeneration, an error-correction mechanism is employed to find those rules that should be modified from the Constrained Grammars. This mechanism counts how many times a constrained rule succeeds (fails). This mechanism tries to correct the error results and to keep the correct results. According to the success times and failure times, the constrained rules are partitioned into three types.

1. Failure Times = 0
2. Success Times ≥ Failure Times
3. Success Times < Failure Times

We only keep the type (1) and type (2) constrained rules. Type (3) rules are removed from the constrained grammar.

5. Experimental Results for Parsing Spoken Corpus

In the experiments, we first train a parser from a written corpus (NTU Treebank). A Chinese spoken corpus is partitioned into a training part and a testing part. We apply the error-correction mechanism again in training part to capture some special linguistic phenomena occur in the spoken data. The rules, which do not occur in the original grammars, are appended. Besides, the rules, which are not appropriate for parsing the spoken data, are also removed from the original grammars.

Table 1 shows the experimental results of the testing part. The evaluation criteria, i.e., precision, recall and
crossing brackets, based on Black et al. (1991), are adopted. Because the precision and the recall measures do not consider the constituent labels, the measures of labeled precision and labeled recall are also considered. In the experiments, the speech repairs are removed from the test sentences and these sentences are also segmented linguistically.

<table>
<thead>
<tr>
<th>Table 1. Experimental Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Items</strong></td>
</tr>
<tr>
<td>Average Sentence Length</td>
</tr>
<tr>
<td>Average Time (Sec.)/Sentence</td>
</tr>
<tr>
<td>Precision</td>
</tr>
<tr>
<td>Recall</td>
</tr>
<tr>
<td>Labeled Precision</td>
</tr>
<tr>
<td>Labeled Recall</td>
</tr>
<tr>
<td>Crossings Per Sentence</td>
</tr>
<tr>
<td>Sentence with 0 Crossing</td>
</tr>
<tr>
<td>Sentence with 1 Crossing</td>
</tr>
<tr>
<td>Sentence with 2 Crossing</td>
</tr>
</tbody>
</table>

The experiments show that the parser is very efficient and effective. Based on a PC/Pentium--III-500 computer with 8 megabytes of RAM, the parser achieves the parsing speed under 0.018 seconds per sentence. At the same time, a 96.52% (94.14%) of (labeled) precision rate and a 93.27% (90.97%) of (labeled) recall rate can be achieved.

6. Concluding Remarks

Spoken data is more complicated than written data. In addition, unavailability of large-scale spoken treebank for training a spoken parser is also a challenge. In this paper, we use a written corpus to train a multi-level chunking-and-raising written parser, and use a small spoken corpus to adapt the written parser. Parsing is composed of multi-level chunking-and-raising actions, which can be done in linear time. The constraints in each rule, which have the function of two-way lookahead, guide the rule selection and also decrease the local ambiguity. Besides, an error-correction mechanism is also proposed to deal with the inappropriate constrained rules, and to learn the new grammar rules for parsing spoken dialogue. Because the chunking-and-raising model can also serve as a partial parser, the integration of correcting speech repair, identifying linguistic boundaries, and parsing in an interleaved (rather than pipelined) style may be studied further.

Acknowledgements

Research on this paper was partially supported by National Science Council grant NSC87-2213-E002-030 and NSC88-2213-E002-052.

References


