Disambiguation of Hypotheses in LCFLEX

Taro Watanabe

August 3, 1999

1 Introduction

Analyzing spontaneous spoken language is one of the difficult task for a parser due to disfluencies in input sentence. A parser should be robust enough so that it can analyze and generate candidates as much as possible by combining partial parses or by introducing additional non-terminals to recover the original sentence. This kind of parsing strategy tends to yield numbers of analyses for one input, thus required knowledge to distinguish the intended meanings of the sentence from other meanings.

This project investigate this problem by combining various knowledge source available through parsing stage, then score each of the analyzed result. The scores can be used to pick up a plausible analysis.

Currently, the knowledge source for disambiguation come from the LCFLEX internal registers, for example, skipped words, inserted words and statistical score. These parameters are scored by penalty functions for each parameter and linearly combined. To allow certain degree of control of those parameters, hill-climbing search is used to train the coefficients for the penalty score combination. The penalty functions are combined with coefficients to form one score for each analysis, and this score is used for disambiguation.

At first, one of the flexible parsers, LCFLEX is introduced, then available knowledge source from the LCFLEDX is discussed. In addition, penalty functions are formalized and some methods for hill-climbing search is presented. Finally, experimental results and its analysis is discussed later.

2 Flexibility of LCFLEX

LCFLEX is a flexible left-corner parser, developed by Rose and Lavie, and is specifically designed to parse spontaneous spoken language [2]. The parser is able to analyze extragrammatical utterances, either transcribed by hand or recognized by speech recognizer, including repetition, false starts, filled pause and idiosyncratic constructions that is not covered by a system's parsing grammar.

For the flexibility, the LCFLEX implements four features: one is the inter-analysis skipping, which allows parsing in chunks. However, this analysis will sometimes result in numbers of chunks and not combined together if a sentence contain uncovered words in the middle of a sentence. Thus, intra-analysis skipping can be used to skip some words inside a chunk. This intra-analysis skipping can be controlled by the maximum number words words that can be skipped.

Thirdly, insertion allowed particular nonterminals or terminals to be inserted in system's grammar. This is one of the grammar dependent features, though, this enhance the ability of the LCFLEX. For example, if a sentence missed a subject, the parser can insert one if such insertion is allowed.

Finally, the parser can relax a particular conditions for unification. LCFLEX is a unification based parser, and if one unification failed, such non-terminal will not be allowed. However, in some case, such a constrain, like agreement is too strong for parsing spontaneous spoken language, it is plausible to relax condition in order to allow unification.

With all the features mentioned above, the LCFLEX is capable of parsing ill-formed sentences, but resulted in numbers of possible analyses. Those hypotheses should be disambiguated at later stage.

3 Disambiguation of Hypotheses

For the disambiguation, less plausible parse results should be ranked lower, and at the same time, maximum covering analyses should be ranked higher. In order to implement this, a set of penalty functions are implemented according to both of the linguistics consideration and statistical analysis.

3.1 Penalty Function

Penalty functions ranks hypotheses according to the available criteria in LCFLEX, skipping, insertion, unification, counter and statistics. Those functions independently compute the function value according to each criteria.

The skipping penalty assigns penalty value, ranging from 0.95 to 1.05 for each skipped words. This is applied from the GLR* parser, and slightly higher penalty is assigned to the words that appear later in the input utterances, in order to take into account the phenomena of false starts.

The penalty for insertion and unification is just the number of inserted words and the number of occurrence of flexible unification respectively.

The counter value is the value implemented in parsing grammar to show the preference of a certain attachment of non-terminals. It is assumed to have higher value for less preferred hypotheses, thus this value can be used as a penalty function.

The statistical penalty is computed from a bigram probability of rules applied in a sequence of parsing. This bigram model is trained through a set of training corpus that are disambiguated correctly without flexibility in LCFLEX. The penalty function is implemented to assign higher score for lower confident hypothesis as following: [1]

$$0.1 \times (-log_{10}P)$$

3.2 Combination of the Penalty Functions

The penalty functions are independently computed based on the hypotheses and corresponding internal register in LCFLEX, and should be accumulated as one parse analysis should be selected properly. For the combination, linear combination of functions is assumed, but combined with weights. These weights depend of the behavior of analysis grammar and the target sentences, thus should be carefully set up. In this project, hill-climbing search is used to train those weights.

Each function is assigned coefficients and those coefficients are trained through searching. The evaluation function is thus realized as following.

$$C_s N_s f_s + C_i N_i f_i + C_u N_u f_u + C_c N_c f_c + C_{stat} N_{stat} f_{stat}$$

The C_* represents coefficients for each penalty function, and N_* stands for normalization value for each function. The normalization factors are used to normalize the function scores to range from 0 to 1 and computed from training corpus. The coefficients takes the value from 0 to 1.

The possibly best combination of coefficients is calculated through the hill-climbing search. The next section describe the hill-climbing search used in this project.

4 Hill-Climbing Search

The hill-climbing search is one of the technique used to explore the vast search space and is trying to find out the local optimum solution. Although this search method is not assured to obtain the best solution, still useful if a good heuristic is not available.

In this project, two kind of methods are tested: One is usual training, selecting coefficients combination that will improve disambiguation performance, and varying only one parameter in one stage. The algorithm is shown as following:

1. Select one coefficient randomly.
2. Increment or decrement the coefficient.
3. Parse and disambiguate sentences in training set.
4. Compute the disambiguation quality score.
5. If the disambiguation quality score improved, use this as a combination of coefficients and go to 1; otherwise, choose different coefficient then go to 2.

The other method change only one parameter, others remain fixed at one stage as following:

1. Select one coefficient randomly.
2. Among the possible value of the coefficient (ranging from 0 to 1), select one value.
3. parse and disambiguate then, compute the disambiguation quality score.
4. If all the possible value is exhausted, pick up a combination of coefficients that shows better score, then go to 1; otherwise, continue.

The score for the disambiguated hypothesis is a measurement of the closeness of a disambiguated hypothesis to its target feature-value structure, and a sum of the number of miss matches and the number of missed features. This score gives the closeness, but not exact matching, of the disambiguated hypothesis and lower score means higher performance in disambiguation.

The following sections describes Experimental results of hill-climbing search for parameter adjustment with the detail analysis of the effects of insertion and flexible unification.

5 Experiment

To test the penalty functions, hill climbing search is conducted on the Scheduling Task. The parsing grammar in Scheduling Task is mainly developed for GLR*, therefore, the grammar is well developed with skipping, but no insertion or flexible unification in mind. The application of the Scheduling Task has two reasons: One reason is that the task has training data and test data with annotation in ILT, a c-structure used for this task, at the same time, hence easier to find out one of the best optimum setting for flexibility.

Another reason for choosing the Scheduling Task grammar is to clarify the flexibility of LCFLEX with insertion and flexible unification, by using the existing parsing grammars that is not developed for such new features. Grammar development is one of the long effort for a spoken language system, thus re-use of existing grammar rules with enhancement by parser's sophisticated flexibility will shorten the time for grammar development, and, at the same time, will allow the parser to analyze further extragrammatical phenomena in utterances.

The corpus has 635 sentences and randomly split into 509 and 126 sentences for training and testing respectively.

5.1 LCFLEX setting

With a magnitude of flexibility in LCFLEX, the tuning flexibility parameter in LCFLEX for a specific grammar is a critical issue. However, the flexibility parameter is solely dependent on a grammar specification, there exist no uniform or convenient way to do this. In the experiment, inter-analysis skipping is enabled and intra-skipping was set to 3. This will allow skipping for the words not covered by the grammar up to 3, like adverb or false starts. The beam search parameter was set to 3. The increment of this parameter will allow more sentences to be parsed, but, at the same time, increase the speed of parsing.

The insertion parameter is set to `((<aux1> 1) (<subject> 1))` in order to insert missed subject and/or auxiliary, like "do". The flexible unification parameter is assigned `((sentence-type sentence-type2))`, thus when a certain unification occurred that will check for the feature of `sentence-type`, this unification will succeed.

The ambiguity packing was set to false, as it seems to cause the analyses with insertion and flexible unification applied and ones without those flexibility get packed, thus yield unexpected analyses.

Finally, one parse results contain multiple feature structures as illustrated in figure 1. This is due to the multiple path allowed for unification, and there seem to be no way to pick up the right one among them. Thus, the multiple feature structures are treated as a list of feature structures, and the disambiguation quality is computed for each structures. Among the disambiguation quality values, the lowest value is treated as the quality value for the analysis.

The following table shows the number of parsable sentences with or without insertion and flexible unification.

Table 1: Pre-experimental Result of Parsing with flexibility

	# of sentences	Parsable 1	Parsable 2
Training set	509	343	265
Test set	126	86	69
All set	635	429	334

note: Parsable 1: skipping
Parsable 2: skipping + insertion + flexible unification

The number of parsable sentences dropped in a parser with insertion and flexible unification. One of the reason of this drop is the limitation in skip beam setting, and because of this limitation, some partial parse are stripped off during the course of parsing.

```
(*OR*
((COUNTER 13.200001)
 (UTTERANCES
   (*MULTIPLE*
     ((WORDS-PARSED-TO-THIS-POINT
         ((14 MORNING) (13 THE) (12 IN) (11 ELEVEN) (10 AT) (9 OVER)
           (8 GETS) (7 THAT) (6 MEETING) (5 A) (4 HAVE) (3 FIRST) (2 THE)
           (1 FRIDAY)
      )  )
       (FRAME *ELLIPSIS) (COUNTER 2.1) (A-SPEECH-ACT *NEGATE)
       (NEGATIVE +) (SENTENCE-TYPE *QUERY-IF) (ASPECT PERFECT)
       (PAST *DUMMY*) (ADVERB *DUMMY*)
       (WHEN ((FRAME *SIMPLE-TIME) (TIME-OF-DAY MORNING)))
     )
     ((COUNTER 1)
       (WORDS-PARSED-TO-THIS-POINT
         ((14 MORNING) (13 THE) (12 IN) (11 ELEVEN) (10 AT) (9 OVER)
           (8 GETS) (7 THAT) (6 MEETING) (5 A) (4 HAVE) (3 FIRST) (2 THE)
           (1 FRIDAY)
       ) )
       (FRAME *FREE) (SENTENCE-TYPE *DUMMY*)
       (WHEN ((FRAME *SIMPLE-TIME) (HOUR 11))) (ADVERB *DUMMY*)
       (NEGATIVE *DUMMY*)
     )
    .....
```

Figure 1: multiple feature structures in one analysis

5.2 Result

In the hill-climbing search, only a set of parsable sentences was selected from the training set. The step value for hill-climbing was set from -16 to 16, then normalized from -1 to 1. The normalization factors were trained from the training corpus, by looking into the range of values each coefficients might take.

Table 2: Normalization value

N_s	N_c	N_{stat}	N_i	N_u
0.0316	0.0709	0.0232	0.500	0.333

Two methods of hill-climbing were tested, but resulted in the same parameter value, given in table . This result indicated that skipping, insertion and unification should be weighed highly

Table 3: Hill-climbing result

C_s	C_c	C_{stat}	C_i	C_u
1.0	0.125	0.125	1.0	1.0

for disambiguation. That means that a hypothesis with inserted words or skipped words should be ranked lower, and one with flexible unification, if it occurs, also should be neglected. One of the reasons for the lower weight for counter penalty might be the reliability of the counter value assignment, thus even if its penalty function indicates higher value, the function value should be subtracted. This is also true for statistical penalty function.

Using this parameter combination, the sentences in the test set were parsed and disambiguated, resulted in successful disambiguation of 38 sentences.

However one sentence, that could be parsed only using insertion and flexible unification was not correctly disambiguated (see figure 2 for the sentence and corresponding feature structure). For

```
BUT THAT WOULD BE UNCOOL

((SENTENCE-TYPE *FIXED-EXPRESSION)
(CONJUNCTION BUT)
(SPEECH-ACT *REJECT)
(DEGREE WEAK)
(TYPE NEGATIVE)
(FRAME *RESPOND))
```

Figure 2: A sentence that can be parsed only with insertion and flexible unification enabled

this sentence, the same hill-climbing was applied to check for the combination of the parameter, and result is shown in table 4. For this particular experiment, all the hypotheses from the parser

Table 4: Hill-climbing result

C_s	C_c	C_{stat}	C_i	C_u
1.0	*	0.0	*	0.0

note: * means any value ranging from 0.0 to 1.0

were filtered out to pick a set of best covered sentences, in other word, a set of hypotheses with less number of skipped words. Without this filter, the correct hypothesis was not ranked as the best one.

The result indicates that skipping is highly penalized, and the counter and insertion penalty can be any degree, but statistical and unification penalty should not be considered in any combinations.

It was expected that skipping penalty should be highly considered as the analysis grammar is highly tuned for such a situation. However, statistical penalty is derived from parsable sentences without any flexibility, thus the statistical model did not contribute for disambiguation. The statistical model with all the flexibility enabled might help for its disambiguation when a training is involved in parsing of all the sentences in training set.

The sentence is parsable only by flexible unification, thus it should be reasonable not to consider the unification penalty.

6 Conclusion

The disambiguation for the LCFLEX was introduced with a combination of penalty functions. The penalty functions were mainly taken from the GLR*, with modification for the LCFLEX. Those functions were combined through coefficients and the coefficients were trained through hill-climbing search under parsable sentences. At the same time, a sentence parsable only by insertion and flexible unification enables is also tested with hill-climbing serarch to obtain a penalty function combination. Those two kind of hill-climbing search resulted in a different type of combination of penalty functions, both of them were plausible combination under an analysis grammar tested in this project.

References

[1] Alon Lavie. Glr*: A robust parser for spontaneously spoken languge. In *Proceedings of ACL-94*, 1994.

[2] Carolyn Penstein Rose and Alon Lavie. Lcflxe: An efficient robust left-corner parser. *LCFLEX manual.*