A Probabilistic Chunker

Kuang-hua Chen and Hsin-Hsi Chen

Department of Computer Science and Information Engineering

National Taiwan University

Taipei, Taiwan, R.O.C.

Abstract

This paper proposes a probabilistic partial parser, which we call chunker. The chunker partitions the input sentence into segments. This idea is motivated by the fact that when we read a sentence, we read it chunk by chunk. We train the chunker from Susanne Corpus, which is a modified but shrinked version of Brown Corpus, underlying bi-gram language model. The experiment is evaluated by outside test and inside test. The preliminary results show the chunker has more than 98% chunk correct rate and 94% sentence correct rate in outside test, and 99% chunk correct rate and 97% sentence correct rate in inside test. The simple but effective chunker design has shown to be promising and can be extended to complete parsing and many applications.

1. Introduction

A probabilistic approach to natural language processing is not new [1]. Recently, many parsers based on this line have been proposed [2-9]. Garside and Leech [2] apply the constituent-likehood grammar of Atwell [10] to probabilistic parsing. Magerman and Marcus [3] adopt the chart-based probabilistic parsing. Zuijlen [4] tells out three probabilistic applications in parsing task. He also claims the probabilistic method should be controlled, otherwise it is not useful to us. Some papers [5-9] employ probabilistic context-free grammar to parsing task. The probabilistic context-free grammar is a modified version context-free grammar, which associates each grammar

rule with a probability. The fact that these papers [11, 12, 13] use probabilistic approach to process speech also shows this approach has wide applications. Although these parsers apply different approaches, they all try to completely parse an input sentence into an annotated tree.

Abney [14] proposes a two-level architecture to tackle with the parsing task. The first level is a chunker, which is responsible for segmenting the input sentence into chunks. The second is an attacher, which is accountable for uniting the chunks to a parsing tree. This idea is motivated by the intuition:

(1) When we are about to read a sentence, we usually read it chunk by chunk.

We examplify the intuition by (2).

(2) [When we] [are about to] [read a sentence,] [we usually read it] [chunk by chunk].

The words between the left square bracket and the right square bracket form a chunk. Between chunks, we pause a while, when we read it. Abney further applies the context-free grammar to forming the backbone of chunker and attacher. Therefore, Abney's chunker and attacher are special LR-style parsers.

In this paper, we will propose a probabilistic chunker underlying bi-gram language model as a partial parser. The reason to call it partial parser is the fact that the chunker only segments the sentence into chunks. Instead of producing the hierarchical annotated tree, the chunker only produces the linear chunk sequence. The parameters of underlying bi-gram language model are trained from Susanne corpus [15, 16], which contains one tenth of Brown Corpus [17] and adopts the LOB corpus [18] tagging style. The Susanne corpus has more syntactic information and semantic information than Brown corpus, including parsing trees and trace marks.

This kind of partial parsers has many applications [19-22]. Church [19] applies the idea of partially parsing to designing a probabilistic NP detector. Church et al. [20] use Fidditch parser to extract typical arguments of verbs. Hindle [21] also employs Fidditch parser to extract arguments of verb for noun classification. Smadja [22] applies partial parser to collocation extraction. Our partial parser, chunker, not only provides the linear chunk sequence, but also the head of each

chunk. This information can be applied to extracting the argument structure of verb and collocation. In addition, the chunker may be extended to a complete parser.

Section 2 will give a brief introduction to Susanne Corpus. Section 3 will describe the task and the language model. We will present the experiment procedure in Section 4 and show the preliminary results of the experiment in Section 5. In Section 6, we will describe the applications of chunker and future developments. Finally, we will give a brief conclusion.

2. Susanne Corpus

The Susanne Corpus is the modified and the condensed version of Brown Corpus. It only contains the 1/10 of Brown Corpus, but involves more information than Brown Corpus. The Corpus consists of four kinds of texts: 1) A: press reportage; 2) G: belles letters, biography, memoirs; 3) J: learned writing; and 4) N: adventure and Western fiction. The Categories of A, G, J, and N are named from each of the Brown Corpus. Each Category consists of 16 files and each file contains about 2000 words.

The following shows a snapshot of Susanne Corpus.

(3)
```
A01:0010a     -    YB      <minbrk>          -        [Oh.Oh]
A01:0010b     -    AT      The       the     [O[S[Nns:s.
A01:0010c     -    NP1s    Fulton    Fulton  [Nns.
A01:0010d     -    NNL1cb  County    county  .Nns]
A01:0010e     -    JJ      Grand     grand   .
A01:0010f     -    NN1c    Jury      jury    .Nns:s]
A01:0010g     -    VVDv    said      say     [Vd.Vd]
A01:0010h     -    NPD1    Friday    Friday  [Nns:t.Nns:t]
A01:0010i     -    AT1     an        an      [Fn:o[Ns:s.
A01:0010j     -    NN1n    investigation     investigation    .
A01:0020a     -    IO      of        of      [Po.
A01:0020b     -    NP1t    Atlanta   Atlanta [Ns[G[Nns.Nns]
A01:0020c     -    GG      +<apos>s          -        .G]
A01:0020d     -    JJ      recent    recent  .
A01:0020e     -    JJ      primary   primary .
A01:0020f     -    NN1n    election          election          .Ns]Po]Ns:s]
A01:0020g     -    VVDv    produced          produce [Vd.Vd]
A01:0020h     -    YIL     <ldquo>   -       .
A01:0020i     -    ATn     +no       no      [Ns:o.
A01:0020j     -    NN1u    evidence          evidence          .
A01:0020k     -    YIR     +<rdquo>          -        .
A01:0020m     -    CST     that      that    [Fn.
A01:0030a     -    DDy     any       any     [Np:s.
```

```
A01:0030b       -    NN2      irregularities  irregularity     .Np:s]
A01:0030c       -    VVDv     took     take    [Vd.Vd]
A01:0030d       -    NNL1c    place    place   [Ns:o.Ns:o]Fn]Ns:o]Fn:o]S]
A01:0030e       -    YF       +.       -       .O]
```

The snapshot shows each line of the corpus includes six fields: 1) reference; 2) status; 3) wordtag; 4) word; 5) lemma; and 6) parse. Reference field shows the information of file name, the original line number in the Brown Corpus and word index in the Corpus (indexed with lower-case letter). Status field denotes the "abbreviation" or "symbol" information. Wordtag field points out what part of speech of the word should be. The tagging set, which is an extension and a modification of the tagging set of LOB Corpus, consists of 358 tags. Lemma field shows the base form of the word. Parse field is the core of the corpus, which shows the grammatical structure of the text and the current word is represented by "." symbol. Table 1 gives an overview of the Susanne Corpus. The details can refer to [15, 16].

Table 1. The Overview of Susanne Corpus

Categories	Files	Paragraphs	Sentences	Words
A	16	767	1445	37180
G	16	280	1554	37583
J	16	197	1353	36554
N	16	723	2568	38736
Total	64	1967	6920	150053

3. Task Description and Language Model

Parsing can be viewed as optimizing. Suppose a n-word sentence, w_1, w_2, ..., w_n (including pucntuation marks) , the parsing task is to find a parsing tree T, such that $P(T|w_1, w_2, ..., w_n)$ has the maximal probability. The annotated form of parsing tree T is changeable freely according to the task demand. We define T here to be a sequence of chunks, c_1, c_2, ..., c_m, and each c_k $(0 < k \leq m)$ contains one or more words $w_j (0 < j \leq n)$. For example, the sentence "parsing can be viewed as optimizing ." consists of 7 words. Its one possible parsing result under our guideline is:

(4) [Parsing] [can be viewed] [as optimization] [.]
c_1 c_2 c_3 c_4

Now, the parsing task is to find the best chunk sequence, C^*, such that

(5) $C^* = \arg\max_{C_i} P(C_i|w_1^n)$

The C_i is one possible chunk sequence, $c_1, c_2, \ldots, c_{m_i}$, where the m_i is the number of chunks of the possible chunk sequence. To resolve the optimization problem, we may adopt various language models. Here bi-gram language model is applied. Therefore, we further reduce $P(C_i|w_1^n)$ as (6),

(6)

$$P(C_i|w_1^n) = P_i(c_1^{m_i}|w_1^n)$$
$$\cong \prod_{k=1}^{m_i} P_i(c_k|c_{k-1}, w_1^n) \times P_i(c_k|w_1^n)$$
$$\cong \prod_{k=1}^{m_i} P_i(c_k|c_{k-1}) \times P_i(c_k)$$

where $P_i(\cdot)$[1] denotes the probability for the i'th chunk sequence. Once a probability $P_i(\cdot)$ is zero, the formula (6) will be zero. We then transform (5) to (7). In addition, when $P_i(\cdot)$ is zero, we define $\log(P_i(\cdot))$ to be zero.

(7)

$$\arg\max_{C_i} P(C_i|w_1^n)$$
$$\cong \arg\max_{C_i} \prod_{k=1}^{m_i} P_i(c_k|c_{k-1}) \times P_i(c_k)$$
$$= \arg\max_{C_i} \sum_{k=1}^{m_i} [\log(P_i(c_k|c_{k-1})) + \log(P_i(c_k))]$$

In order to make the expression (7) match the intuition of human being, namely, 1) the scoring metrics are all positive, 2) large value means high score, and 3) the scores are between 0 and 1, we define a score function $S(\cdot)$ shown as (8).

(8) $S(P(\cdot)) = 0$ when $P(\cdot) = 0$;

$S(P(\cdot)) = 1.0/(1.0+\mathrm{ABS}(\log(P(\cdot))))$ otherwise.

We then rewrite (7) as (9).

---

1 In general, $P(\cdot)$ repesents the probabilities of some events.

(9)

$$\arg\max_{C_i} P(C_i|w_1^n)$$

$$\cong \arg\max_{C_i} \prod_{k=1}^{m_i} P_i(c_k|c_{k-1}) \times P_i(c_k)$$

$$= \arg\max_{C_i} \sum_{k=1}^{m_i} [\log(P_i(c_k|c_{k-1})) + \log(P_i(c_k))]$$

$$= \arg\max_{C_i} \sum_{k=1}^{m_i} [S(P_i(c_k|c_{k-1})) + S(P_i(c_k))]$$

The final language model is to find a chunk sequence C^*, which satisfies the expression (9).

4. Experiment Procedure

There are three parts in the experiment: the first part is training; the second is testing; the third is evaluating. Training process is to extract bi-gram data from Susanne corpus; testing process is 1) to tag the input raw data from the Susanne corpus, and then output tagged data; 2) to chunk the input data and produce the chunked data. Evaluating process is to compare the chunked data to Susanne corpus, and reports the correct rate. These are shown in the Figure 1.

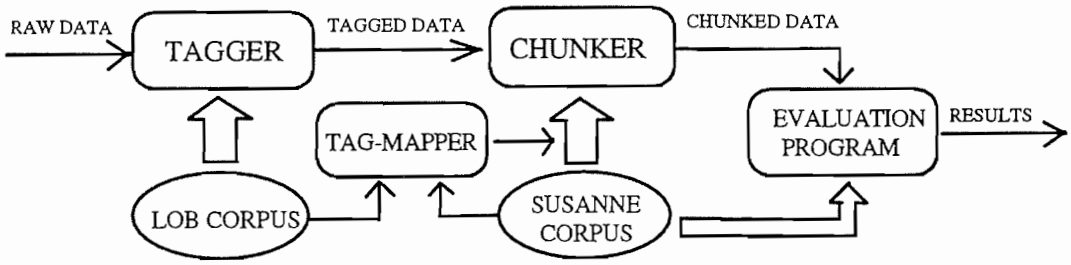


Figure 1. Experiment Procedure

The tagger is trained from LOB corpus [18]. This corpus contains 1 million words of English texts. Since the tag set of LOB corpus is different from that of the Susanne corpus, we first write a mapping program, TAG-MAPPER, to recover the LOB tags from the Susanne tags. The program maps 358 tags which Susanne corpus defines to 134 tags LOB corpus defines[2]. Then,

[2] Susanne corpus tags genitive case noun as [John_NP 's_GG], but LOB corpus tags it as [John's_PN$]. Two tags of Susanne corpus may be mapped to one tag of LOB corpus.

according to the criteria of (10), we extract the bi-gram chunk data from 3/4 of Susanne corpus (the rest is for outside test).

(10) a. The chunk is similiar to the phrase with content word as its head.

b. The considered content words are noun, verb, adjective, and preposition.

c. When a considered phrase is complex, a chunk contains at most two level sub-tree.

When we extract the bi-gram chunk data, we map them to the LOB tags and store them in datafile. Then, we sort this chunk data and build the "chunk grammar". As the results, the number of chunk grammar rules is 8675.

The second part is to test the Susanne corpus. The original 3/4 of Susanne corpus is used for inside testing; the rest of it for outside testing. The chunker runs on Sun SPARC-1 workstation. The processing time is shown in Table 2. In Table 2, Time/W means the time taken to process a word; Time/C means the time taken to process a chunk; and Time/S means the time taken to process a sentence.

Table 2. The Processing Time

	OUTSIDE TEST			INSIDE TEST		
	Time/W	Time/C	Time/S	Time/W	Time/C	Time/S
A	0.00944	0.0182	0.2268	0.01006	0.0264	0.2653
G	0.00889	0.0172	0.2174	0.00933	0.0252	0.2249
J	0.00902	0.0181	0.2738	0.00888	0.0263	0.2316
N	0.00988	0.0180	0.1634	0.00972	0.0220	0.1426
Average	0.00931	0.0179	0.2204	0.00950	0.0250	0.2161

According to Table 2, to process a word needs 0.00931 seconds for outside test, 0.00950 seconds for inside test, and 0.00941 on average. To process all Susanne corpus needs about 1412 seconds, or 23.6 minutes. Figure 2 depicts this results.

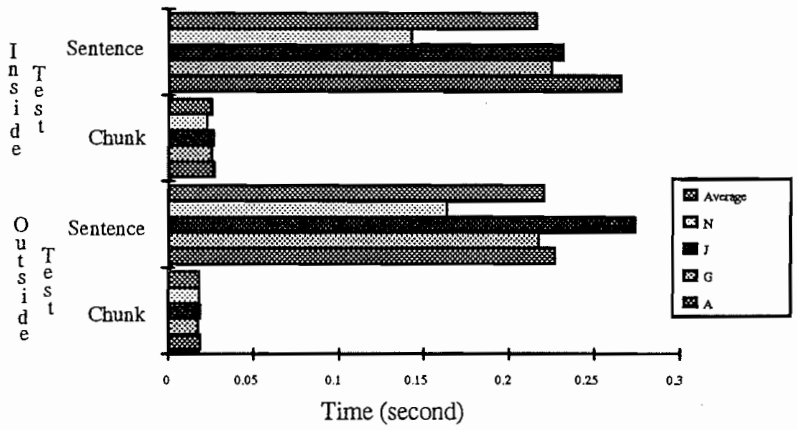
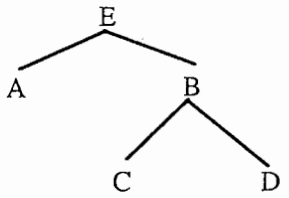


Figure 2. The Processing Time for Sentence and Chunk

The evaluating part is to compare the parsing results of our chunker with the denotation made by the Susanne corpus. The criterion is that the content of each chunk should be dominated by one non-terminal node in Susanne parse field.



[A C D]	right
[A] [C D]	right
[A C] [D]	wrong
[A] [C] [D]	right

Figure 3. The Evaluation Criterion

Figure 3 further explains this criterion. For a parsing tree [E [A] [B [C D]]], as shown in the left part of Figure 3, there are four possible chunk sequences. The third chunk sequence violates the criterion, since the contents of the first chunk are dominated by the different non-terminal nodes.

5. Preliminary Results

As the Section 4 points out, we begin the inside test by using the 3/4 of Susanne corpus and outside test by using the rest of the corpus. Evaluating the results by the criterion mentioned previously, we have the preliminary results shown in Table 3.

Table 3. Experimental Results

Category	TEST	OUTSIDE TEST		INSIDE TEST	
		Chunks	Sentences	Chunks	Sentences
A	# of correct	4866	380	10480	1022
	# of incorrect	40	14	84	29
	#	4906	394	10564	1051
	correct rate	0.99	0.96	0.99	0.97
G	# of correct	4748	355	10293	1130
	# of incorrect	153	32	133	37
	#	4901	387	10426	1167
	correct rate	0.97	0.92	0.99	0.97
J	# of correct	4335	283	9193	1032
	# of incorrect	170	15	88	23
	#	4505	298	9281	1055
	correct rate	0.96	0.95	0.99	0.98
N	# of correct	5163	536	12717	1906
	# of incorrect	79	42	172	84
	#	5242	578	12889	1990
	correct rate	0.98	0.93	0.99	0.96
Average	# of correct	19112	1554	42683	5090
	# of incorrect	442	103	477	173
	#	19554	1657	43160	5263
	correct rate	0.98	0.94	0.99	0.97

There are two kinds of correct rates. The first is chunk correct rate, which is measured by the correct segmented chunks over the total segmented chunks. The second is sentence correct rate, which is measured by the correct segmented sentences over the total sentences. A wrong segmented chunk means the whole sentence is not chunked properly. From Table 3, we know the overall sentence correct rate is over 94% and the chunk correct rate is over 98%. The difference between the inside test and outside test is not trivial. We compare the training data extracted from all Susanne corpus and the 3/4 of corpus, and find that the data from the latter cover the 80% of data from the former. The rest 20% data capture the gap of correct rate between inside test and outside test. But the 94% chunk correct rate have shown the work is promising. Figure 4 shows the correct rates of these experiments and gives an overview of these experiments.

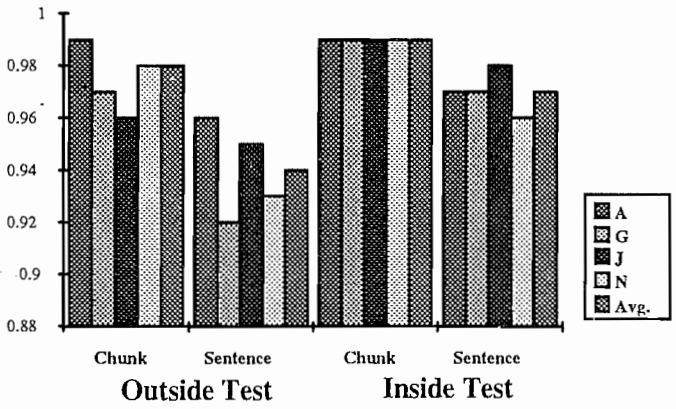


Figure 4. The Correct Rate of Experiments

For further analyzing the experiment, we define the chunk length.

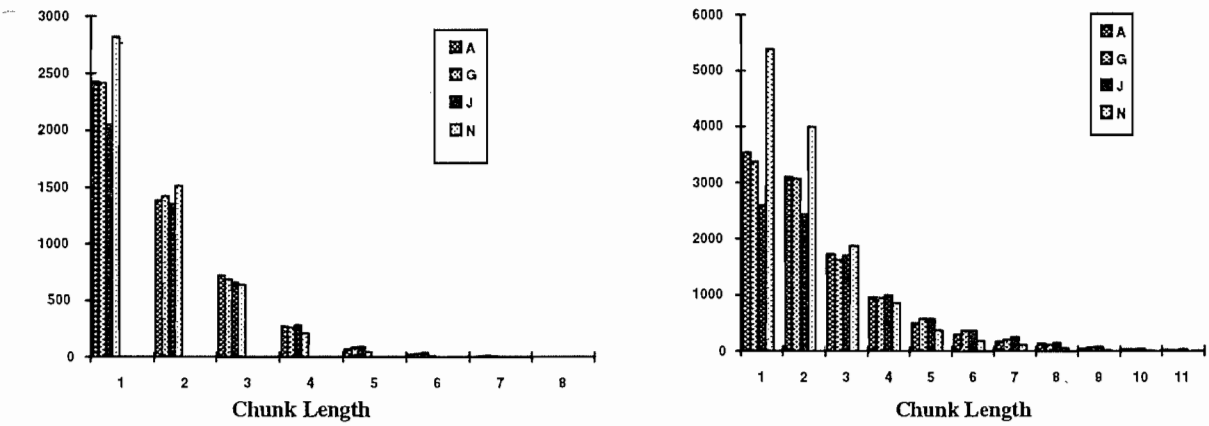
(11) Chunk length is the number of the words in a chunk.

We analyze the distribution of chunk length and list it in Table 4.

Table 4. The Distribution of Chunk Length

Chunk Length	OUTSIDE TEST				INSIDE TEST			
	A	G	J	N	A	G	J	N
1	2427	2411	2054	2823	3540	3380	2602	5390
2	1385	1420	1355	1511	3109	3070	2439	3999
3	721	688	659	635	1730	1630	1711	1873
4	276	260	283	208	959	952	997	854
5	67	83	95	46	509	590	587	378
6	24	31	43	11	302	363	368	186
7	3	7	13	7	169	210	253	117
8	3	1	3	1	143	115	151	55
9					52	74	85	20
10					28	28	52	13
11					23	14	36	4

The number of one-word chunks covers 43% of all kinds of chunks. This can be viewed in Figure 5. At the first glance, this result seems to challenge our probabilistic chunker. We further analyze what grammatic component constitutes the one-word chunks. The analysis is listed in Table 5.



(a) Outside Test (b) Inside Test

Figure 5. The Distribution of Chunk Length

In Table 5, WH-PN means wh-pronoun. OTHERS includes interjection, punctuation marks, letters, formulas, and foreign words. Ql/Qn represents the qualifiers and quantifiers. The rest types of one-word chunk are easy to understand.

Table 5. The Types of One-Word Chunks

Chunk Type	OUTSIDE TEST				INSIDE TEST			
	A	G	J	N	A	G	J	N
Noun	851	698	481	934	1399	1082	746	2224
Verb	672	674	549	957	1532	1639	1314	2390
Conj.	172	167	162	151	98	135	62	99
Prep.	145	169	227	109	106	92	91	64
Adjective	113	169	164	95	125	158	145	174
Adverb	143	145	117	288	90	81	88	274
Ql/Qt	96	94	87	70	43	62	64	41
WH-PN	46	46	18	24	76	59	2	43
OTHERS	189	249	249	195	69	72	89	76

We then scrutinize the table and know the most of the one-word chunks consist of noun, verb, and verbial adjective. This is because pronoun and proper name form the bare subject or object; verb is presented in the form of third person and singular, past tense, or base form; adjective forms the verbial adjective phrase, like beautiful in the sentence "Mary is beautiful". Figure 6

gives a clear view on the distribution. Noun and verb consist of 72% of one-word chunks. This shows our approach is useful to segment the sentence into the suitable chunks.

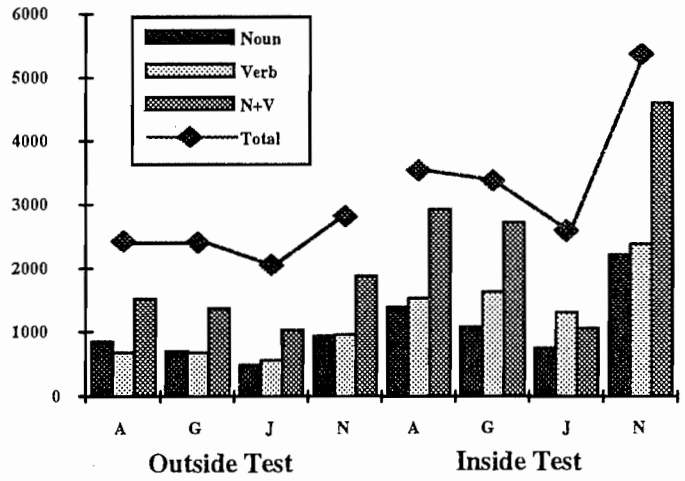


Figure 6. The Distribution of Noun and Verb Chunk

In Appendix, we list a sample output of the partial parsing.

6. Applications

Recently, the partial parsers have been applied to many problems as a preprocessor [19-22]. The applications include extracting argument structure of verbs [19, 20], grouping words [21], gathering collocations [22], and so on. Our probabilistic chunker is also capable of resolving these problems. We may modify the current version of chunker. The modified chunker not only partitions the input text, but also associates each chunk with a phrase mark (or a chunk mark). If it is a one-word chunk, the word itself is the chunk mark. For other chunks, the chunker finds the most manifest word in this chunk as the chunk mark. Generally speaking, the word is the head of this chunk. (12) is a possible chunked sentence.

(12) [We_PP1AS] [saw_VBD] [NN(2): a_AT woman_NN] [IN(1): with_IN a_AT telescope_NN] [._.]

In (12), every chunk is associated with a mark and its position in the chunk (it is unnecessary to associate one-word chunk with this information). According to the information, we may extract

argument structure of verb with SVO and other heuristic rules. Furthermore, we can group noun or verb according to the extracted argument structure.

In addition to these applications, we may construct a recursive probabilistic chunker to be a complete parser. We may reorganize the parsing task as a sequence of actions, chunking and raising interleavingly. The parsing task is finished, when no more chunking is needed. This idea is shown in Figure 7.

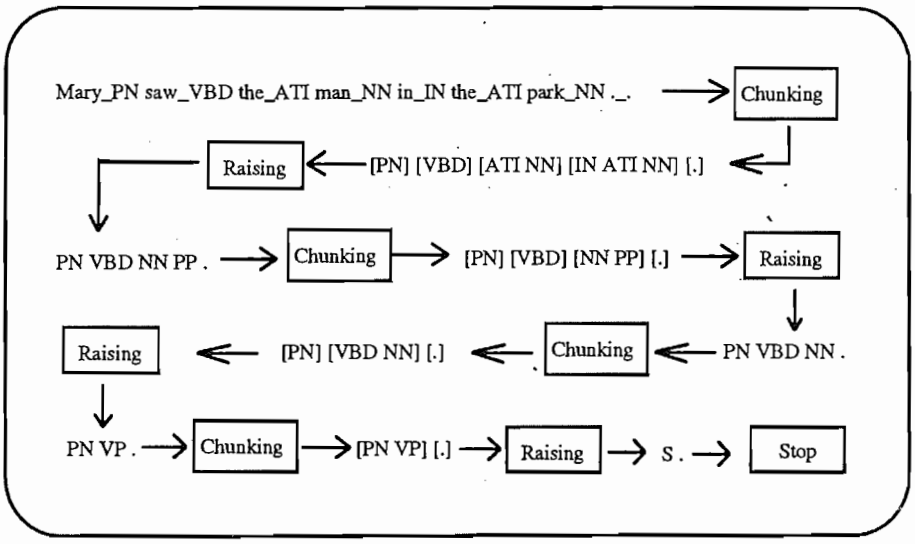


Figure 7. A Recursive Chunker as a Parser

We formally define parsing as (13)-(16) based on the idea.

(13) Parsing is a sequence of actions consisting of chunking and raising interleavingly.

(14) Chunking is an action of segmenting input components into a sequence of chunks.

(15) Raising is an action of lifting the head from input chunks.

(16) Parsing is finished, when no chunking can be operated on.

7. Concluding Remarks

To process real text is indispensable for a practical natural language system. Probabilistic method provides a robust way to tackle with the unrestricted text. This is why probabilistic method

dominates the recent research directions of natural language processing. In the field of parsing techniques, many parsers based on this line are proposed. Some of them are LR-style [5-9]; some of them are chart-based [3]; some adopt constituent-likehood grammar [2]. These approaches are more complexive. For example, it is necessary for the probabilistic LR parsing to extract hierarchical context-free grammar rules from corpus and to calculate the probability associated with each rule. Once there are left-recursive rules, we must transform them or use equations to solve these intermixing probabilities [7]. In this paper, we report a probabilistic chunker to execute the partial parsing. Comparing to these approaches mentioned above, ours is simple and easy to extend to construct a complete parser. In training process, the mere work we do is to extract bi-gram (according to the language model; maybe tri-gram) linear data from a parsed corpus. Through the evaluation procedure, the correct rate is promising. The preliminary experimental results show the chunker has the 98% correct rate for chunk and 94% for sentence in outside test. It depicts our finding is worthy looking forward to. In addition, we also provide the future development and the possible applications of the finding.

Acknowledgements

We are grateful to Dr. Geoffrey Sampson for his kindly providing Susanne Corpus and the details of tag set to us. Research on this paper was partially supported by National Science Council grant NSC82-0408-E002-029.

References

[1] P. Suppew, "Probabilistic Grammars for Natural Languages," *Synthese* 22, 1970, pp. 95-116.

[2] R. Garside. and F. Leech, "A Probabilistic Parser," *Proceedings of Second Conference of the European Chapter of the ACL*, 1985, pp. 166-170.

[3] D.M. Magerman and M.P. Marcus, "Pearl: A Probabilistic Chart Parser," *Proceedings of Fifth Conference of the European Chapter of the ACL*, 1991, pp. 15-20.

[4] J.M.V. Zuijlen, "Probabilistic Methods in Dependency Grammar Parsing," *Proceedings of International Workshop on Parsing Technologies*, 1989, pp. 142-151.

[5] T. Fujisaki, "A Stochastic Approach to Sentence Parsing," *Proceedings of 22th Annual Meeting of the ACL*, 1984, pp. 16-19.

[6] T. Fujisaki, *et al.*, "Probabilistic Parsing Method for Sentence Disambiguation," *Proceedings of International Workshop on Parsing Technologies*, 1989, pp. 85-94.

[7] S.K. Ng and M. Tomita, "Probabilistic LR Parsing for General Context-Free Grammars," *Proceedings of International Workshop on Parsing Technologies, 1991, pp. 154-163.*

[8] A. Corazza, *et al.*, "Stochastic Context-Free Grammars for Island-Driven Probabilistic Parsing," *Proceedings of International Workshop on Parsing Technologies*, 1991, pp. 210-217.

[9] J. Wright and E.N. Wrigley, "Adaptive Probabilistic Generalized LR Parsing," *Proceedings of International Workshop on Parsing Technologies*, 1991, pp. 100-109.

[10] E.S. Atwell, "Constituent-Likelihood Grammar," *(ICAME News)*, No. 7, 1983, pp. 34-66.

[11] K. Kita, *et al.*, "Parsing Continuous Speech by HMM-LR Method," *Proceedings of 27th Annual Meeting of the ACL*, 1989, pp. 126-131.

[12] J.H. Wright and E.N. Wigley, "Probabilistic LR Parsing for Speech Recognition," *Proceedings of International Workshop on Parsing Technologies*, 1989, pp. 105-114.

[13] S. Seneff, "Probabilistic Parsing for Spoken Language Applications," *Proceedings of International Workshop on Parsing Technologies*, 1989, pp. 209-218.

[14] S. Abney, "Parsing by Chunks," in *Principle-Based Parsing,* Berwick, Abney and Tenny (Eds.), Kluwer Academic Publishers, 1991, pp. 257-278.

[15] G. Sampson, "The Susanne Corpus," *ICAME Journal*, No. 17, 1993, pp. 125-127.

[16] G. Sampson, *English for the Computer*, Oxford University Press (Forthcoming).

[17] N. Francis and H. Kucera, *Manual of Information to Accompany a Standard Sample of Present-day Edited American English, for Use with Digital Computers*, Department of Linguistics, Brown University, Providence, R. I., U.S.A., original ed. 1964, revised 1971, revcised and augmented 1979.

[18] S. Johansson, *The Tagged LOB Corpus: Users' Manual*, Bergen: Norwegian Computing Centre for the Humanities, 1986.

[19] K.W. Church, "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," *Proceedings of Second Conference on Applied Natural Language Processing*, 1988, pp. 136-143.

[20] K.W. Church, *et al.*, "Parsing, Word Associations and Typical Predicate-Argument Relations," *Proceedings of International Workshop on Parsing Technologies*, 1989, pp. 389-398.

[21] D. Hindle, "Noun Classification from Predicate-Argument Structures," *Proceedings of 28th Annual Meeting of ACL*, 1990, pp. 268-275.

[22] F. Smadja, *Extracting Collocations from Text, An Application: Language Generation* Ph.D. Dissertation, Columbia University, 1991.

Appendix

Five examples are demonstrated in the appendix. In each example, the first part is the sentence extracted from Susanne Corpus but tagged with the LOB tagging set; the second part is the corresponding chunked results by our Chunker.

1. The_ATI Fulton_NP County_NPL Grand_JJ Jury_NN said_VBD Friday_NR an_AT investigation_NN of_IN Atlanta_NP recent_JJ primary_JJ election_NN produced_VBD <ldquo>_*' +no_ATI evidence_NN +<rdquo>_**' that_CS any_DTI irregularities_NNS took_VBD place_NPL +._.

[The_ATI Fulton_NP County_NPL]

[Grand_JJ Jury_NN]

[said_VBD]

[Friday_NR]

[an_AT investigation_NN]

[of_IN Atlanta_NP]

[recent_JJ primary_JJ election_NN]

[produced_VBD]

[<ldquo>_*' +no_ATI evidence_NN +<rdquo>_**']

[that_CS any_DTI irregularities_NNS]

[took_VBD]

[place_NPL]

[.]

2. The_ATI jury_NN further_RBR said_VBD in_IN term_NR +<hyphen>_*- +end_NN presentments_NNS that_CS the_ATI City_NPL Executive_JJB Committee_NN +,_, which_WDTR had_HVD over<hyphen>all_JJB charge_NN of_IN the_ATI election_NN +,_, <ldquo>_*' +deserves_VBZ the_ATI praise_NN and_CC thanks_NNS of_IN the_ATI City_NPL of_IN Atlanta_NP +<rdquo>_**' for_IN the_ATI manner_NN in_IN which_WDTR the_ATI election_NN was_BEDZ conducted_VBN +._.

[The_ATI jury_NN]

[further_RBR said_VBD]

[in_IN term_NR +<hyphen>_*- +end_NN]

[presentments_NNS]

[that_CS the_ATI City_NPL Executive_JJB Committee_NN +,_,]

[which_WDTR had_HVD]

[over<hyphen>all_JJB charge_NN of_IN the_ATI election_NN +,_,]

[<ldquo>_*' +deserves_VBZ]

[the_ATI praise_NN and_CC thanks_NNS]

[of_IN the_ATI City_NPL of_IN Atlanta_NP +<rdquo>_**']

[for_IN the_ATI manner_NN]

[in_IN which_WDTR]

[the_ATI election_NN]

[was_BEDZ conducted_VBN]

[.]

3. The_ATI September_NR +<hyphen>_*- +October_NR term_NR jury_NN had_HVD been_BEN charged_VBN by_IN Fulton_NP Superior_JJ Court_NN Judge_NPT Durwood_NP Pye_NP to_TO investigate_VB reports_NNS of_IN possible_JJ <ldquo>_*' +irregularities_NNS +<rdquo>_**' in_IN the_ATI hard_RB +<hyphen>_*- +fought_VBN primary_NN which_WDTR was_BEDZ won_VBN by_IN Mayor_NPT +<hyphen>_*- +nominate_RB Ivan_NP Allen_NP Jr_NPT +._.

[The_ATI September_NR +<hyphen>_*- +October_NR]

[term_NR]

[jury_NN]

[had_HVD been_BEN charged_VBN]

[by_IN Fulton_NP Superior_JJ Court_NN]

[Judge_NPT Durwood_NP Pye_NP]

[to_TO investigate_VB]

[reports_NNS of_IN possible_JJ]

[<ldquo>_*' +irregularities_NNS +<rdquo>_**']

[in_IN the_ATI]

[hard_RB +<hyphen>_*- +fought_VBN]

[primary_NN]

[which_WDTR was_BEDZ won_VBN]

[by_IN Mayor_NPT +<hyphen>_*- +nominate_RB]

[Ivan_NP Allen_NP Jr_NPT]

[.]

4. <ldquo>_*' +Only_RB a_AT relative_JJ handful_NN of_IN such_ABL reports_NNS was_BEDZ received_VBN +<rdquo>_**' +,_, the_ATI jury_NN said_VBD +,_, <ldquo>_*' +considering_IN the_ATI widespread_JJ interest_NN in_IN the_ATI election_NN +,_, the_ATI number_NN of_IN voters_NNS and_CC the_ATI size_NN of_IN this_DT city_NPL +<rdquo>_**' +._.

[<ldquo>_*' +Only_RB a_AT relative_JJ handful_NN of_IN such_ABL reports_NNS]

[was_BEDZ received_VBN +<rdquo>_**' +,_,]

[the_ATI jury_NN]

[said_VBD +,_,]

[<ldquo>_*' +considering_IN the_ATI widespread_JJ interest_NN in_IN the_ATI election_NN +,_,]

[the_ATI number_NN of_IN voters_NNS]

[and_CC the_ATI size_NN of_IN this_DT city_NPL +<rdquo>_**']

[.]

5. The_ATI jury_NN said_VBD it_PP3 did_DOD find_VB that_CS many_AP of_IN Georgia_NP registration_NN and_CC election_NN laws_NNS <ldquo>_*' +are_BER outmoded_JJ or_CC inadequate_JJ and_CC often_RB ambiguous_JJ +<rdquo>_**' +._.

[The_ATI jury_NN]

[said_VBD]

[it_PP3]

[did_DOD find_VB]

[that_CS]

[many_AP]

[of_IN Georgia_NP]

[registration_NN]

[and_CC election_NN]

[laws_NNS]

[<ldquo>_*' +are_BER]

[outmoded_JJ or_CC inadequate_JJ]

[and_CC often_RB ambiguous_JJ +<rdquo>_**']

[.]