A Controlled Skip Parser

Kenji Yamada
kyamada@isi.edu
USC/Information Sciences Institute

Abstract

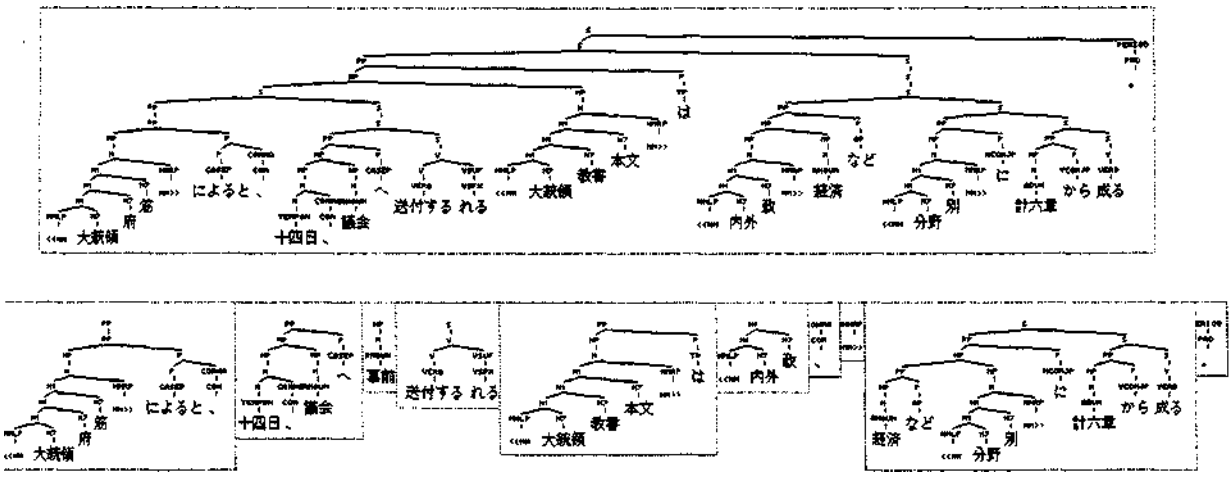
Real-world natural language sentences are long and complex, and always contain unexpected grammatical constructions. It even includes noise and ungrammaticality. This paper describes the Controlled Skip Parser, a program that parses such real-world sentences by skipping some of the words in the sentence. The new feature of this parser is that it can control its behavior to find out which words to skip, without using domain-specific knowledge. Statistical information (N-grams), which is a generalized approximation of the grammar learned from past successful experiences, is used for the controlled skip. Experiments on real newspaper articles are shown, and our experience with this parser in a machine translation system is described.

1 Introduction

Parsing real-world natural language text is a difficult task. The difficulty comes from several sources. One is that a real-world sentences are long and complex, especially in newspaper articles. The average length of a sentence is 20 to 30 words, and it often exceeds 100 words. The sentence structure is complex and syntactically ambiguous due to the frequent usage of relative clauses, appositives, disjunctions, and other constructions. Also, there are always unexpected syntactic phenomena in real-world texts, which are beyond the formal grammar's coverage. Another difficulty conies from noise and ill-formedness of the input, typically in spoken language. Interjected words *(ah, well),* false starts *(I'd like to do you have any flights from ...*), or errors in speech recognition devices make parsing with a rigid grammar impossible. This kind of noise and ill-formedness is also seen in written text, for example, misspelled words, unbalanced parentheses, incorrect punctuation, symbol characters, and special formatting characters which fail to be filtered by text preprocessing modules.

One extreme way to deal with these difficulties is to abandon syntactic analysis completely, instead relying purely on statistics for machine translation [Brown, et al. 1990], or using template matching for information extraction [Jackson, et al. 1991]. With these methods, no grammar rules are used, and no analysis of syntactic and semantic structure of sentences is performed. Target translation or extracted information is directly obtained from the surface input, using context sensitive word-by-word translation statistics or target-dependent database templates. These approaches, however, cannot fully utilize the semantic and pragmatic knowledge even when the syntactic structure is easy to parse.

When domain-specific semantic and pragmatic knowledge is available, a certain level of loose syntactic analysis has been shown to be useful, such as in [McDonald 1992], [Hobbs, et al. 1992], [Seneff 1992], and [Stallard and Bobrow 1993]. In these systems, a regular parser first attempts to parse a sentence, and when the parser fails, the parse fragments, which represent syntactic structure of parts of the sentence, are combined together using domain-specific semantics and pragmatics.



The upper tree is the result of parsing with two words skipped. The lower trees are from a non-skipping parser, producing 10 fragments. The meaning of those fragments (from left) are; 1. *According to a source near the president,* 2. *on 14th, to the parliament,* 3. *in advance* (skipped), 4. *being sent,* 5. *the presidential message is,* 6. *domestic and foreign policy,* 7. comma (skipped), 8. inserted-marker, 9. *consists of six chapters by category such as economics,* 10, period.

Figure 1: Skip Parsing versus Non-skip Parsing

Further, if domain-specific lexical and syntactic knowledge is available, grammar rule can be designed to create parse fragments which can be easily combined and used, as in [Ayuso, et al. 1994].

However, these methods, which rely heavily on domain-specific knowledge, are useful only when such knowledge is available. When such knowledge is not available, or in systems which need to handle texts in any domain, domain independent syntactic methods are more useful. For example, [Mellish 1989] shows how to enhance chart-based techniques for parsing ill-formed sentences. In his method, when bottom-up chart parsing fails, top-down predictions, derived from grammar rules, make hypotheses to add/delete/substitute words to parse the sentence. [Lavie and Tomita 1993} show an extension to the GLR (Tomita) parser, which skips words by allowing shift operations at inactive states.

Skipping words in the input sentence to obtain a complete parse is effective in processing spoken language with noise. As mentioned above, noise is also seen in written text. Moreover, some unexpected grammatical constructions can be parsed by skipping words. Figure 1 is one example of skip-parsing a Japanese sentence in a newspaper article. In this example, a complete parse (shown as an upper tree in the figure) is obtained by skipping an adverb and a comma. Without skipping these two words, the parser is left with six major parse fragments (shown as lower trees in the figure).

This example is taken from the parser described in this paper. This parser is used in a broad-coverage Japanese-to-English machine translation system [Knight, et al. 1995]. Because this system operates across domains, it is impossible to rely on domain-specific semantic and pragmatic knowledge to complete a parse. Syntax-only domain-independent mechanism for unparsable sentence is the most effective method for a broad-coverage system.

There is another reason for using a skip parser. In the early stage of the development, we wanted to have a simple general fallback mechanism which absorbs the inadequacy of text preprocessing module and low coverage of grammar rules. Text formatting of the input can be arbitrary. It is

difficult to have a preprocessing module which perfectly filters out symbol characters and special formatting characters, especially in the early stage of development.[1] It takes several months or even years to build sufficiently many grammar rules.[2] Skip parsing is a convenient way to absorb the immaturity of text preprocessing and grammar rules.

The basic idea of a skip parser is very simple; just allow the parser to create a syntactic constituent with a gap. But the naive implementation would make it practically unusable. Consider the English noun phrases noun + "of"+ noun and noun + noun . These word sequences often appear in English texts. Notice that most of the noun + "of"+ noun sequences would still be grammatically correct even if the word "of" were skipped. The parser wastes time and space in processing all the possible skips. In other examples, skipping a certain word is locally grammatical but will not lead to a complete successful parse.

Due to undesirable or unsuccessful skipping, a practical implementation of a skip parser has to deal with time and space limitations. In other words, the exhaustive search must be avoided by using efficient technique for reducing the search space.

[Lavie and Tomita 1993] reduce the search space by using beam-search which limits the number of active nodes. Although this is a heuristic and is not guaranteed to find a optimal skipping, it is reported to be useful. Following this idea, we investigated more powerful heuristics for controlling the search space: using statistics of past successful parses. The statistics are a form of knowledge generalized from past experiences.

The heuristics and mechanism of the parser are described in the next section. Our experimental results are shown in Section 3. Discussion and conclusions follow.

2 Mechanism

2.1 Skip Control using N-gram

To avoid grammatically correct but undesirable skipping, we use N-gram information obtained from past successful parsing. As we saw in the last section, it is generally fruitless to try skipping the preposition *of* between two nouns. To get this kind of knowledge, we feed our parser thousands of unannotated sentences, select sentences which are successfully parsed, and record which sequences of words can be parsed. The frequency statistics of the sequences of words are called N-gram. Note that we select parsable sentences using our own grammar and parser (without skipping), because we need knowledge about which word sequences can be parsed under the current grammar, to find out which words should be skipped for unparsable sentences.

The N-gram information obtained from past successful parsing is a handy measure to guess which words are good candidates to be skipped. Normally N-gram frequency information records how often two or more words appear adjoining. We use a variant of N-gram information. As is seen in the previous example, it is enough to know the sequence noun + "of"+ noun is likely to be parsed successfully. Therefore, for content words such as nouns, adjectives, and verbs, the

[1] Also, word skipping can work well with markers inserted by a preprocessing module. We use a separate pre-parsing module which inserts phrase boundaries and noun compound markers for efficient parsing (examples are shown in Figure 1). Since this module is not always accurate, the markers must be treated only as hints. The mechanism of skip parsing works as a hint handling facility. It skips the marker if the parse fails and does the parsing without the marks.

[2] Word skipping can give good feedback to the grammar writer. By knowing which words are skipped, a grammar writer can easily identify an error or a missing rule in the grammar.

grammatical category name is used as an N-gram index. Then, the preposition "of" in noun + "of"+ noun is guessed not to be skipped, estimated by the N-gram value of noun + "of"+ noun . The N-gram can be also thought as an approximation of the grammar rules. This approximation can be seen as generalized knowledge learned from past successful experiences. Although long-distance syntactic relations and global phrase structures cannot be captured by the N-grams, intra-phrase local constraints and specific patterns at phrase boundaries can be approximated.

2.2 Extension to Chart Parser

We applied this idea of controlled skip parsing on our bottom-up chart parser. A bottom-up chart parser ([Kay 1980]) builds up larger constituents by concatenating adjoining constituents as prescribed by grammar rules. By recording created constituents in a chart, a parse fragment (constituent) is shared by multiple different parse trees, avoiding repetition of the analysis of the same part of the input sentence.

Two extensions are needed to the standard chart parser to be able to skip words using the ranking information provided by an N-gram model. One is controlling the order of parsing by the cost (ranking) information, and another is extending lexical constituents to skip words.

Each constituent in the chart is given a cost value. If the cost is lower, the parser gives precedence to it, and delays the processing if the cost is higher. The N-gram information is one of the costs used for controlling skip parser. Other factors such as the total number of words skipped, and the *isolation factor* (described later), are also considered and single cost as an integer value is assigned to each constituent in the chart.

The cost is cumulative. When a new constituent is created from several children, the sum of each child's cost (with an adjustment function) is assigned to the new constituent.[3] If it is a normal (non-skipping) one, the cost is assigned zero. Thus, if the parser can parse the input without skipping a word, the final parse tree has a cost of zero.

If the normal parsing fails, skipped lexical constituents are created, by extending normal lexical constituents. A normal lexical constituent corresponds to one word in the input sentence. By extending it, it covers more than a two-word span, then words under the extended span are effectively skipped.[4]

Skipped lexical constituents are assigned non-zero costs. A value from the N-gram database and other factors decide the cost. The cost for extended lexical constituents are decided only when the such constituents are created, and the cost of non-lexical constituents is just the sum of the cost of its children.

2.3 Cost assignment

The following two subsections describe the details of how the cost is assigned. There are two kinds of cost assignments; one is the initial cost assignment, and another is the compositional

[3] It is possible to use a complicated function rather than summing the childrens' cost. In our implementation, we only adjust the value when skipping a pair of brackets in the input sentence to treat it as if only one word is skipped. Generally, just a simple summing of childrens' cost is enough, because recalculating the cost of each constituent is costly.

[4] A normal lexical constituent can be extended leftward or rightward, but to skip a word, only either left or right word should be extended. It may be possible to extend non-lexical constituents for efficiency, though we have not done tests. See Discussion.

Initial cost :=
    if normal lexical constituent then 0
    else /* skipped lexical constituent */
number-of-words-skipped * 100 + *N-gram-value* (0,1,2,5,8,or 10) +
part-of-bigger-constit? * 30 + *non-skipping-words?* * 99999
Compositional cost :=
    if the skip broke newly balanced brackets then 9999
    else *sum-of-its-children- number-of-balanced-pair-of-new-brackets* * 100

Figure 2: Cost Assignment

cost assignment. The initial cost assignment is for a newly created lexical (normal and skipped) constituent, and the compositional cost assignment is for a newly created non-lexical constituent. These costs assignment changes the total behavior of the skip parser.

Our particular implementation is shown in Figure 2. This implementation enforces parsing without skipping first, then if it fails, trying to skip one word.[5] If it still fails, it tries skipping two words, and it tries successively until maximum cost (determined by the length of the sentence) is reached.[6] A total cost of zero means no skipping has been attempted. A total cost of 1 to 100 means one word was skipped, a total cost of 101 to 200 means two words were skipped, and so on.[7]

We use an N-gram value which is normalized, quantized, and applied an transformation function. The detail is described in the next subsection.

The *chart isolation factor* is given by the variable *part-of-bigger-constit?*. It takes a value of 0 or 1. If there is a bigger constituent whose span covers the word, it becomes 1. This roughly means that the word successfully becomes a part of a bigger constituent, so that it seems bad idea to skip the word. The variable becomes 0 when a word cannot attach to other constituents due to a grammar problem, and becomes an isolated constituent in the chart.

The variable *non-skipping-words?* prevents the skipping of some important words. Currently, dummy words START-OF-TEXT and END-OF-TEXT are set not to be skipped.

2.4 N-gram construction

We next describe assigning costs derived from N-grams, which is a part of initial cost assignment. The N-gram information is important for spotting which words to skip. Effective and efficient construction of N-gram information is one of the key points for the controlled skip parser.

First, we mentioned that the grammatical category name is enough for indexing open-class words in the N-grams, as seen in the example noun + "of"+ noun . However, using just the category name is sometimes too coarse. For example, our grammar is an unification-based feature grammar. Following the trend of contemporary linguistics, fewer grammatical categories are used and complicated variations are expressed using a feature representation. In this setting, N-gram information from the sequence of grammatical categories would not be useful enough to control

[5] A pair of balanced brackets is considered as one word, and skipping only one of a pair is prohibited by a high cost. These brackets are dummy words inserted by the pre-parsing bracketing module, and are not parentheses or quotation marks in the original input sentence. As mentioned before, these brackets are inserted as hints for the parser, and are sometimes wrong. The skip parser effectively removes those wrong hint marks.

[6] It also has a limit of total CPU time and a limit of the number of constituents in the chart.

[7] This is not strictly true. For example, if there are four *part-of-bigger-constit* words skipped in the constituent, the cost will be greater than 130 * 4 = 520, which exceeds lowest five-word skipped one. But we usually do not allow five words to be skipped, and practically we do not need strict cost calculation.

skipping. Therefore, in our implementation, some of the syntactic features are used to index open-class words in addition to the category name.[8] For closed-class words such as particles and affixes, we used the surface form and the grammatical category.[9]

Second, the collected N-gram counts (how many such sequences of words, grammatical categories, or features appear in parsable sentences), should be normalized and quantized. The purpose of normalization is to make the N-gram count independent of the sampling size. To do this, the raw N-gram count is divided by the maximum count, so that the range of normalized count is zero to one. Quantization is done for efficiency at run time. Floating point data is converted to integer data for speed. A transformation function is applied at the same time. Since the N-gram count does not linearly correspond to the parsing cost, some transformation is necessary. The function is arbitrary. In our implementation, we use a step function with five thresholds 0, 1/1000, 1/100, 1/20, and 1/10. The range values are 0, 1, 2, 5, 8, and 10.

Finally, but most importantly, we describe how to mimic N-grams by just bigrams (N=2). Since obtaining arbitrary length of N-gram is costly in terms of computation and preparing sample data, we only use bigrams. A bigram value is obtained for an adjacent pair of words. This value is assigned to the word boundary of the two words. Note that if the words have multiple part-of-speech tags, there are multiple possible bigram values at the boundary. If two words have N tags and M tags, the possible number of bigram value is N * M. We select the maximum of these possible values and assign it as the bigram value at the boundary. This value shows how frequently those two words appeared in the past successful parsing. Therefore, the existence of an unseen tagging sequence, which has low value, is irrelevant if there is a more frequently seen tagging sequence. This justifies the use of the maximum of possible bigram values.

When assigning cost to a skipped lexical constituent, the parser must decide the cost for the skipped word, using the word boundary bigram value. The minimum of right adjacent boundary value and left adjacent boundary value is used as a cost for skipping the word. A low bigram value at a boundary means that the words either the left or right of the boundary should be skipped. To select one of the two, each word's opposite boundary value can be used. If the word fits well in the sentence, the opposite side's value are likely to be high, which justifies the use of minimum value of the right and left adjacent boundary value.

3 Experiments

3.1 Coverage

We have gathered performance data during the development of a large-scale Japanese grammar. The grammar and the parser was built as a part of JAPANGLOSS Japanese-to-English machine

[8] For Japanese nouns, we have eight syntactic subcategories, which are represented in a subcategory feature. For Japanese verbs, we put the inflectional form as a feature. These feature values together with its grammatical category are used for an index to the N-gram database.

[9] Multiple part-of-speech tags (grammatical category) and multiple parse trees must be handled properly. We use JUMAN ([Matsumoto, et al. 1993]) part-of-speech tagger before parsing. Unlike other English part-of-speech taggers, it gives multiple possible tags. Recall that we collect N-gram information from sentences which can be parsed by the current grammar. When a sentence is parsed with multiple different parse trees due to multiple part-of-speech tagging of a word, the N-gram value must be discounted by the number of trees. For example, a specific word in a sentence has three possible part-of-speech tags and each tag assignment leads to two parse trees, (thus obtaining six parse trees), each word sequence involving such tag must be counted 1/6 times. The same treatment has to be done for a disjunctive feature if such a feature is used for indexing N-grams.

	sentences	parsed without skip	parsed with skip	Total parsed sentences
Test2	458	189 (41.3%)	255 (49.1%)	414 (90.4%)
Test3	482	253 (52.5%)	198 (41.1%)	451 (93.6%)
Test4	500	295 (59.0%)	186 (37.2%)	481 (96.2%)
Test4n	500	295 (59.0%)	147 (29.4%)	442 (88.4%)

Table 1: Coverage of Skip Parsing

translation system [Knight, et al. 1995].

Table 1 is the summary of the performance as the raw grammar coverage expanded from around 40% to 60%, as seen in the third column of the table, which shows the number of sentences parsable without skipping. The fourth column shows sentences which were parsed *only* by skipping, and the rightmost column shows the total number of parsed sentences. In other words, the second column indicates the raw grammar coverage, and the rightmost column is the coverage of the skip parser.

Five hundred test sentences were randomly[10] taken from actual Japanese newspapers.[11] Bi-grams were collected from four thousand parsable sentences from the same corpus (not overlapping with the test sentences). The average sentence length was 22.4 words (86.4 bytes). The longest sentence was 199 words (696 bytes) long.[12] On Test4n,[13] the average number of skipped words among the sentences which were parsed only by skipping was 1.67 words. The maximum number of words skipped was limited to five words.

As the raw grammar coverage grew from 41.3% to 59.0%, the number of sentences parsed by skipping declined from 49.1% to 29.4%, while the total parsed sentences still grew from 90.0% to 96.2%. This shows that some of the sentences parsed by skipping became covered as the raw grammar coverage expanded. Also some sentences which were not parsed even by skipping became parsable as those sentences became easy to parse by skipping. Since the skip parser skips practically only two or three words at most, if the original sentence had three or more problems, it would not be parsed even by skipping until the problems uncovered by the raw grammar had become one or two.

3.2 Correctness

Although the coverage of the skip parser is satisfactorily high, the practical question is how good the output of skipped parse is. Skipping a word such as adverb or symbol character is almost harmless for the whole translation, but skipping an important noun or verb would be disastrous. We investigated what kind of words were skipped in the skipped parse in Test4n. The results is shown in Table 2.

There were 147 sentences which were parsed by skipping one or more words. The total number of skipped words were 246 words.[14] The first column of the table shows the category of skipped

[10] It even included sentences from an article which only consisted of listings of upcoming events.

[11] We used the same 500 test sentences for all the tests. Test2 and Test3 failed to test some of the sentences due to some system trouble. The percentages for those two tests are calculated by dividing by the number of sentences which didn't have system trouble (458 for Test2, and 482 for Test3).

[12] This number ignores markers inserted by the preprocessing chunker module. If the markers were included, the average was 35.3 words and the longest was 248 words.

[13] Test4 and Test4n used the same grammar, but Test4n had a stricter limit on the number of constituents. The detailed analysis in the following sections was done on Test4n.

[14] A pair of brackets and chunker marks are counted as one word.

Category	correct	almost	wrong	Total
noun	1	6	13	20
verb	0	1	5	6
particle	15	11	15	41
other categories	3	1	7	11
symbol character	17	0	8	25
parenthesis	23	1	12	36
chunker marker	30	0	0	30
segment error	0	0	11	11
Total	89	20	71	180
(in percentage)	(49.4%)	(11.1%)	(39.4%)	(100.0%)

Table 2: Correctness of word skipping

words. A human reader rated the skip correctness subjectively (correct, almost, or wrong).These are shown in the second to the fourth column of the table.

As expected, if the skipped word was a noun or a verb, most of the sentences were broken. However, for particles, skipping those words worked well in some cases. It included particle "no" used as subject marker, which was relatively a rare case. The next row "other categories" includes adverbs, pronouns, and affixes.

Symbol character and parenthesis were usually safe to be skipped. Those symbol characters included a dot or a small circle at the beginning of a sentence, or a pair of parentheses enclosing an entire sentence. Chunker markers inserted by a pre-parsing module were always safe to be skipped, because these were just optional markers for the parser. On the other hand, if the front-end word segmentation module, which broke a sentence into a sequence of words[15], made an error, it was almost always no help skipping some of the words from a erroneously segmented sequence.

The overall correctness is shown in the bottom row, indicating how percentage of skipped words were marked as correct, almost, or wrong, in total. It shows 49.4% of the skipping was correct, and 39.4% was wrong. This means that about half of the skipping did not harm seriously in carrying out the syntactic analysis of the sentence.

Even though more than half of the correct skipped words were symbol characters, parentheses, and chunker markers, this does not mean that skip parsing mainly works as a compensation of bad preprocessing effects. Symbol characters and parentheses are vital part of our grammar even in the early stage of the development. Newspaper articles make extensive use of commas for lexical, phrasal, and sentential conjunctions, and small dots and other symbol characters often appear in compound nouns or phrasal itemizations. It is very difficult to exhaustively enumerate these non-alphabetic usage as grammar rules, since these are not well studied in linguistic theory, and the variations are diverse. Fortunately, some of the non-alphabetic parts of a sentence are optional, and the skip parser works with them very well. However, these are not totally optional. Human readers benefit from comma and other symbol characters when they read long texts. Similar thing applies to chunker markers. These markers are completely optional in theory, but without them, the CPU and memory usage would be enormous and many newspaper sentences, which tend to be long, are practically unparsable. Therefore, skip parsing not only works as a compensation of wrong but indispensable preprocessing (as in chunker marking), but also works really well for the gaps that the grammar rules could not fill (as in commas and symbol characters, as well as adverbs and particles).

[15] There are no overtly marked boundaries between words in Japanese.

4 Discussion

The results observed in our experiment are satisfactory. As expected, we experienced the good effect of skip parser when the grammar was being developed, and it boosted the grammar coverage from about 60% to 90%. And about half of the extra coverage, the skipping was reasonable.

The heuristics presented here are more powerful than beam-search as in [Lavie and Tomita 1993] or than top-down prediction as in [Mellish 1989], yet our method is still simple, efficient and domain-independent. Using the N-gram as the primary heuristic is a convenient way to approximate grammar rules, generalized from past successful experiences. Note that using N-grams is not a method dependent on the parsing mechanism. This method can be applied to virtually all parsers.[16]

It is further possible to enhance it by creating constituents with skipped holes from non-lexical larger constituents. In our implementation, we only extend lexical constituents and re-build larger constituents with skipped words. This will save the cost of creating larger constituents from scratch, although bookkeeping of constituents might become complicated.

Another possible extension is to use more complicated N-grams, such as collecting data for N=3 or more, or using finer-grained grammatical categories and features for indexing open-class words. Using variable length N-grams and more generalized N-grams as in [Pereira, et al. 1995] would be an interesting extension.

Skipping words in a sentence means some kind of loss of information unless the skipped words are noise. However, since human language has some redundancy and all the information in a sentence is not equally important, parsing by skipping words gives reasonably good results if the skipped words are not particularly important. An important direction of further research is to find a way to identify an unreasonable skipping after parsing. One possible strategy is to reject a parse which skipped words with rich content information in the dictionary. Ultimately, it would be best handled by semantic and pragmatic level. [Lavie 1994] uses semantic coherence for ranking N-best results of his skip parser.

5 Conclusion

We presented a controlled skip parser, which selectively skips words to parse an unparsable sentence. The control information comes from heuristics obtained by statistical information. The statistical information (N-grams) can be seen as generalized knowledge learned from past successful parses. We presented a parsing algorithm based on extending a standard bottom-up chart parser. Experiments show that it significantly improves the grammar coverage, and the result of skipping is satisfactory. This method is very useful for a system which cannot utilize domain-specific knowledge, such as a broad-coverage machine translation system.

6 Acknowledgments

We would like to thank Eduard Hovy and Kevin Knight for their support of this work. We would also thank all the JAPANGLOSS machine translation group.

[16] In fact, we first experimented the first version of the controlled skip parser by writing a program surrounding our original non-skipping parser.

References

[Ayuso, et al. 1994] D. M. Ayuso and the PLUM Research Group, "Pattern Matching in a Linguistically-Motivated Text Understanding System", In *Proceedings of the Human Language Technology Workshop,* pp. 182-186, 1994.

[Brown, et al. 1990] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A Statistical Approach to Machine Translation", *Computational Linguistics,* Vol.16, No.2, pp. 79-85, 1990.

[Hobbs, et al. 1992] J. R. Hobbs, D. E. Appelt, J. Bear, and M. Tyson, "Robust Processing of Real-World Natural Language Texts", In *Proceedings Applied Natural Language Processing,* pp. 186-192, 1992.

[Jackson, et al. 1991] E. Jackson, D. Appelt, J. Bear, R. Moore, and A. Podlozny, "A Template Matcher for Robust NL Interpretation", In *Proceedings DARPA Speech and Natural Language Workshop,* pp. 190-194, 1991.

[Kay 1980] M. Kay, "Algorithm Schemata and Data Structures in Syntactic Processing", CSL-80-12, Xerox Palo Alto Research Center, 1980.

[Knight, et al. 1995] K. Knight, I. Chander, M. Haines, V. Hatzivassiloglou, E. Hovy, M. lida, S. K. Luk, R. Whitney, and K. Yamada, "Filling Knowledge Gaps in a Broad-Coverage Machine Translation System", In *Proceedings of the International Joint Conference on Artificial Intelligence,* pp. 1390-1396, 1995.

[Lavie 1994] A. Lavie, "An Integrated Heuristic Scheme for Partial Parse Evaluation", In *Proceedings of the 32rd Annual Meeting of the Association for Computational Linguistics,* pp. 316-318, 1994.

[Lavie and Tomita 1993] A. Lavie and M. Tomita, "GLR* - An Efficient Noise-skipping Parsing Algorithm For Context Free Grammars", In *Proceedings of Third International Workshop on Parsing Technology,* pp. 123-134, 1993.

[Matsumoto, et al. 1993] Y. Matsumoto, S. Kurohashi, T. Utsuro, Y. Myoki, and M. Nagao, "Japanese Morphological Analysis System JUMAN Manual", Kyoto University, 1993.

[McDonald 1992] D. D. McDonald, "An Efficient Chart-based Algorithm for Partial-Parsing of Unrestricted Texts", In *Proceedings Applied Natural Language Processing,* pp. 193-200, 1992.

[Mellish 1989] C. S. Mellish, "Some Chart-based Techniques for Parsing Ill-formed Input", In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics,* pp. 102-109, 1989.

[Pereira, et al. 1995] F. C. Pereira, Y. Singer, and N. Tishby, "Beyond Word N-grams", In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics,* pp. 95-106, 1995.

[Seneff 1992] S. Seneff, "A Relaxation Method for Understanding Spontaneous Speech Utterances", In *Proceedings of the Human Language Technology Workshop,* pp. 299-304, 1992.

[Stallard and Bobrow 1993] D. Stallard and R. Bobrow, "The Semantic Linker - A New Fragment Combining Method", In *Proceedings of the Human Language Technology Workshop,* pp. 37-42, 1993.