Robust Segmentation of Japanese Text into a Lattice for Parsing

Gary Kacmarcik, Chris Brockett, Hisami Suzuki
Microsoft Research
One Microsoft Way
Redmond WA, 98052 USA
{garykac,chrisbkt,hisamis}@microsoft.com

Abstract

We describe a segmentation component that utilizes minimal syntactic knowledge to produce a lattice of word candidates for a broad coverage Japanese NL parser. The segmenter is a finite state morphological analyzer and text normalizer designed to handle the orthographic variations characteristic of written Japanese, including alternate spellings, script variation, vowel extensions and word-internal parenthetical material. This architecture differs from conventional Japanese wordbreakers in that it does not attempt to simultaneously attack the problems of identifying segmentation candidates and choosing the most probable analysis. To minimize duplication of effort between components and to give the segmenter greater freedom to address orthography issues, the task of choosing the best analysis is handled by the parser, which has access to a much richer set of linguistic information. By maximizing recall in the segmenter and allowing a precision of 34.7%, our parser currently achieves a breaking accuracy of ~97% over a wide variety of corpora.

Introduction

The task of segmenting Japanese text into word units (or other units such as *bunsetsu* (≈phrases)) has been discussed at great length in Japanese NL literature ([Kurohashi98], [Fuchi98], [Nagata94], et al.). Japanese does not typically have spaces between words, which means that a parser must first have the input string broken into usable units before it can analyze a sentence. Moreover, a variety of issues complicate this operation, most notably that potential word candidate records may overlap (causing ambiguities for the parser) or there may be gaps where no suitable record is found (causing a broken span).

These difficulties are commonly addressed using either heuristics or statistical methods to create a model for identifying the best (or *n*-best) sequence of records for a given input string. This is typically done using a connective-cost model ([Hisamitsu90]), which is either maintained laboriously by hand, or trained on large corpora.

Both of these approaches suffer from problems. Handcrafted heuristics may become a maintenance quagmire, and as [Kurohashi98] suggests in his discussion of the JUMAN segmenter, statistical models may become increasingly fragile as the system grows and eventually reach a point where side effects rule out further improvements. The sparse data problem commonly encountered in statistical methods is exacerbated in Japanese by widespread orthographic variation (see §3).

Our system addresses these pitfalls by assigning completely separate roles to the segmenter and the parser to allow each to delve deeper into the complexities inherent in its tasks.

Other NL systems ([Kitani93], [Kurohashi98]) have separated the segmentation and parsing components. However, these dual-level systems are prone to duplication of effort since many segmentation ambiguities cannot be resolved without invoking higher-level syntactic or semantic knowledge. Our system avoids this duplication by relaxing the requirement that the segmenter identify the best path (or even *n*-best paths) through the lattice of possible records. The segmenter is responsible only for ensuring that a correct set of records is present in its output. It is the function of the parsing component to select the best analysis from this lattice. With this model, our system achieves roughly 97% recall/precision (see [Suzuki00] for more details).

1 System Overview

Figure 1 shows a simple block diagram of our Natural Language Understanding system for Japanese, the goal of which is to robustly produce syntactic and logical forms that allow automatic

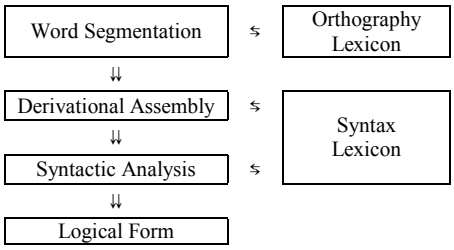

Figure 1: Block diagram of Japanese NL system

extraction of semantic relationships (see [Richardson98]) and support other linguistic projects like information retrieval, NL interfaces and dialog systems, auto-summarization and machine translation.

The segmenter is the first level of processing. This is a finite-state morphological analyzer responsible for generating all possible word candidates into a word lattice. It has a custom lexicon (automatically derived from the main lexicon to ensure consistency) that is designed to facilitate the identification of orthographic variants.

Records representing words and morphemes are handed off by the segmenter to the derivational assembly component, which uses syntax-like rules to generate additional derived forms that are then used by the parser to create syntax trees and logical forms. Many of the techniques here are similar to what we use in our Chinese NL system (see [Wu98] for more details).

The parser (described extensively in [Jensen93]) generates syntactic representations and logical forms. This is a bottom-up chart parser with binary rules within the Augmented Phrase Structure Grammar formalism. The grammar rules are language-specific while the core engine is shared among 7 languages (Chinese, Japanese, Korean, English, French, German, Spanish). The Japanese parser is described in [Suzuki00].

2 Recall vs. Precision

In this architecture, data is fed forward from one component to the next; hence, it is crucial that the base components (like the segmenter) generate a minimal number of omission errors.

Since segmentation errors may affect subsequent components, it is convenient to divide these errors into two types: recoverable and non-recoverable. A *non-recoverable* error is one that prevents the syntax (or any downstream) component from arriving at a correct analysis (e.g., a missing record). A *recoverable* error is one that does not interfere with the operation of following components. An example of the latter is the inclusion of an extra record. This extra record does not (theoretically) prevent the parser from doing its job (although in practice it may since it consumes resources).

Using standard definitions of *recall* (*R*) and *precision* (*P*):

$$R = \frac{Seg_{correct}}{Tag_{total}} \qquad P = \frac{Seg_{correct}}{Seg_{total}}$$

where $Seg_{correct}$ *and* Seg_{total} *are the number of "correct" and total number of segments returned by the segmenter, and* Tag_{total} *is the total number of "correct" segments from a tagged corpus,*

we can see that recall measures non-recoverable errors and precision measures recoverable errors. Since our goal is to create a robust NL system, it behooves us to maximize recall (i.e., make very few non-recoverable errors) in open text while keeping precision high enough that the extra records (recoverable errors) do not interfere with the parsing component.

Achieving near-100% recall might initially seem to be a relatively straightforward task given a sufficiently large lexicon – simply return every possible record that is found in the input string. In practice, the mixture of scripts and flexible orthography rules of Japanese (in addition to the inevitable non-lexicalized words) make the task of identifying potential lexical boundaries an interesting problem in its own right.

3 Japanese Orthographic Variation

Over the centuries, Japanese has evolved a complex writing system that gives the writer a great deal of flexibility when composing text. Four scripts are in common use (kanji, hiragana, katakana and roman), and can co-occur within lexical entries (as shown in Table 1).

Some mixed-script entries could be handled as syntactic compounds, for example, ID カード *[ai dii kaado="ID card"]* could be derived from ID_{NOUN} + カード $_{NOUN}$. However, many such items are preferably treated as lexical entries because

Kanji-Hiragana	新しい *[atarashii = "new"]* ほ乳類 *[honyuurui = "mammal"]*
Kanji-Katakana	歯ブラシ *[haburashi = "toothbrush"]* ヒ素 *[hiso = "arsenic"]*
Kanji-Alpha	CGS 単位 *[CGS tan'i = "CGS system"]*
Kanji-Symbol	12 月 *[juunigatsu = "December"]* γ 線 *[ganma sen = "gamma rays"]*
Mixed kana	おトイレ *[otoire = "toilet"]* ダブる *[daburu = "to double"]*
Kana-Alpha	ID カード *[aidii kaado = "ID card"]* メッセンジャーRNA *[messenjaa RNA = "messenger RNA"]*
Kana-Symbol	ストロンチウム 90 *[sutoronchiumu 90 = "Strontium 90"]* ほえる 40° *[hoeru yonjuu do = "roaring forties"]*
Other mixed	消しゴム *[keshigomu = "eraser"]* α ケンタウリ星 *[arufa kentauri sei = "Alpha Centauri"]* ト書き *[togaki = "stage directions"]*

Table 1: Mixed-script lexical entries

they have non-compositional syntactic or semantic attributes.

In addition, many Japanese verbs and adjectives (and words derived from them) have a variety of accepted spellings associated with *okurigana*, optional characters representing inflectional endings. For example, the present tense of 切り落とす (*kiriotosu = "to prune"*) can be written as any of: 切落す, 切り落す, 切落とす, 切りおとす, きりおとす or even きり落とす, きり落す.

Matters become even more complex when one script is substituted for another at the word or sub-word level. This can occur for a variety of reasons: to replace a rare or difficult kanji (ら致 *[rachi="kidnap"]* instead of 拉致); to highlight a word in a sentence (ヘンなかっこう *[henna kakkou = "strange appearance"]*); or to indicate a particular, often technical, sense (ワタって *[watatte ="crossing over"]* instead of 渡って, to emphasize the domain-specific sense of "connecting 2 groups" in Go literature).

More colloquial writing allows for a variety of contracted forms like オレ達ゃ ≈ オレ達 + は *[ore-tacha ≈ ore-tachi + wa = "we" + TOPIC]* and phonological mutations as in でェーす ≈ です *[dee–su ≈ desu = "is"]*.

This is only a sampling of the orthographic issues present in Japanese. Many of these variations pose serious sparse-data problems, and lexicalization of all variants is clearly out of the question.

repeat characters	時々刻々 ➜ 時時刻刻 *[jijikokkoku = "every moment"]* 甲斐々々しい ➜ 甲斐甲斐しい *[kaigaishii = "diligent"]*
distribution of voicing marks	ヒ゛デ゛オ ➜ ビデオ *[bideo = "video"]*
halfwidth & fullwidth	ＦＭ放送 ➜ FM 放送 *[FM housou = "FM broadcast"]* ﾀﾞｲﾔｸﾞﾗﾑ ➜ ダイヤグラム *[daiyaguramu = "diagram"]*
composite symbols	ﾊﾟｰｾﾝﾄ ➜ パーセント *[paasento = "percent"]* ㍿ ➜ 株式会社 *[kabushiki gaisha = "incorporated"]* ㉘日 ➜ ２８日 *[nijuuhachi nichi = "28th day of the month"]*

Table 2: Character type normalizations

4 Segmenter Design

Given the broad long-term goals for the overall system, we address the issues of recall/precision and orthographic variation by narrowly defining the responsibilities of the segmenter as:

(1) Maximize recall

(2) Normalize word variants

4.1 Maximize Recall

Maximal recall is imperative. Any recall mistake made in the segmenter prevents the parser from reaching a successful analysis. Since the parser in our NL system is designed to handle ambiguous input in the form of a word lattice of potentially overlapping records, we can accept lower precision if that is what is necessary to achieve high recall.

Conversely, high precision is specifically *not* a goal for the segmenter. While desirable, high precision may be at odds with the primary goal of maximizing recall. Note that the lower bound for precision is constrained by the lexicon.

4.2 Normalize word variants

Given the extensive amount of orthographic variability present in Japanese, some form of normalization into a canonical form is a pre-requisite for any higher-order linguistic processing. The segmenter performs two basic kinds of normalization: Lemmatization of inflected forms and Orthographic Normalization.

okurigana	詫状 ➜ 詫び状 *[wabijou = "apology letter"]* 吹ぬき ➜ 吹き抜き *[fukinuki = "drafty"]* ➜ 吹き貫き *[fukinuki = "a well-hole"]*
non-standard script	女のコ ➜ 女の子 *[onnanoko = "girl"]* でいすこ ➜ ディスコ *[disuko = "disco"]*
ヶカ variants	一か月 ➜ 一ヶ月 *[ikkagetsu = "one month"]* 霞が関 ➜ 霞ヶ関 *[kasumigaseki = "Kasumigaseki"]*
numerals for kanji	５輪 ➜ 五輪 *[gorin = "Olympics"]* １人 ➜ 一人 *[hitori = "one person"]*
vowel extensions	おにーさぁん ➜ おにいさん *[oniisan = "older brother"]* ファイトォ ➜ ファイト *[faito = "fight"]*
katakana variants	ヴァイオリン ➜ バイオリン *[baiorin = "violin"]* アラモード ➜ ア・ラ・モード *[aramoode = "à la mode"]*
inline yomi	映(は)える ➜ 映える *[haeru = "to shine"]* 爬虫(はちゅう)類 ➜爬虫類 *[hachuurui = "reptile"]*
inline kanji	げっ(齧)歯類 ➜ 齧歯類 *[gesshirui = "rodent"]*

Table 3: Script normalizations

4.2.1 Lemmatization

LEMMATIZATION in Japanese is the same as that for any language with inflected forms – a lemma, or dictionary form, is returned along with the inflection attributes. So, a form like 食べた *[tabeta ="ate"]* would return a lemma of 食べる *[taberu = "eat"]* along with a PAST attribute.

Contracted forms are expanded and lemmatized individually, so that 食べてっちゃった *[tabe-tecchatta = "has eaten and gone"]* is returned as: 食べる GERUND + いく GERUND + しまう PAST *[taberu="eat" + iku="go" + shimau=ASPECT]*.

4.2.2 Orthographic Normalization

ORTHOGRAPHIC NORMALIZATION smoothes out orthographic variations so that words are returned in a standardized form. This facilitates lexical lookup and allows the system to map the variant representations to a single lexicon entry.

We distinguish two classes of orthographic normalization: character type normalization and script normalization.

CHARACTER TYPE NORMALIZATION takes the various representations allowed by the Unicode specification and converts them into a single consistent form. Table 2 summarizes this class of normalization.

SCRIPT NORMALIZATION rewrites the word so that it conforms to the script and spelling used in the lexical entry. Examples are given in Table 3. Two cases of special interest are *okurigana* and inline *yomi*/*kanji* normalizations. The *okurigana* normalization expands shortened forms into fully specified forms (i.e., forms with all optional characters present). The *yomi*/*kanji* handling takes infixed parenthetical material and normalizes it out (after using the parenthetical information to verify segmentation accuracy).

たべる	[食:た]べる	*taberu = "to eat"*
かみあわせる	[噛:か,み][合:あ,わ]せる	*kamiawaseru = "to engage (gears)"*
みつもり	[見:み][積:つ,も,り]	*mitsumori = "an estimate"*
ID カード	[I:アイ][D:ディー]カード	*aidii kaado = "ID card"*
かぎタバコ	[嗅:か,ぎ][煙草:タバコ]	*kagi tabako = "snuff tobacco"*
ながい	[長:なが][!居:い]	*nagai = "a long visit"*

Table 4: Orthography lattices

5 Lexicon Structures

Several special lexicon structures were developed to support these features. The most significant is an orthography lattice* that concisely encapsulates all orthographic variants for each lexicon entry and implicitly specifies the normalized form. This has the advantage of compactness and facilitates lexicon maintenance since lexicographic information is stored in one location.

The orthography lattice stores *kana* information about each *kanji* or group of *kanji* in a word. For example, the lattice for the verb 食べる *[taberu = "eat"]* is [食:た]べる, because the first character (*ta*) can be written as either *kanji* 食 or *kana* た. A richer lattice is needed for entries with *okurigana* variants, like 切り落とす *[kiriotosu = "to prune"]* cited earlier: commas separate each *okurigana* grouping. The lattice for *kiriotosu* is [切:き,り][落:お,と]す. Table 4 contains more lattice examples.

Enabling all possible variants can proliferate records and confuse the analyzer (see [Kurohashi 94]). We therefore suppress pathological variants that cause confusion with more common words and constructions. For example, 長居 *[nagai = "a long visit"]* never occurs as 長い since this is ambiguous with the highly frequent adjective 長い *[nagai = "long"]*. Likewise, a word like 日本

* Not to be confused with the word lattice, which is the set of records passed from the segmenter to the parser.

[nihon = "Japan"] is constrained to inhibit invalid variants like に本, which cause confusion with: に $_{POSP}$ + 本 $_{NOUN}$ *[ni=PARTICLE + hon ="book"]*.

We default to enabling all possible orthographies for each entry and disable only those that are required. This saves us from having to update the lexicon whenever we encounter a novel orthographic variant since the lattice anticipates all possible variants.

6 Unknown Words

Unknown words pose a significant recall problem in languages that don't place spaces between words. The inability to identify a word in the input stream of characters can cause neighboring words to be misidentified.

We have divided this problem space into six categories: variants of lexical entries (e.g., *okurigana* variations, vowel extensions, et al.); non-lexicalized proper nouns; derived forms; foreign loanwords; mimetics; and typographical errors. This allows us to devise focused heuristics to attack each class of unfound words.

The first category, variants of lexical entries, has been addressed through the script normalizations discussed earlier.

Non-lexicalized proper nouns and derived words, which account for the vast majority of unfound words, are handled in the derivational assembly component. This is where compounds like フランス語 *[furansugo = "French (language)"]* are assembled from their base components フランス *[furansu = "France"]* and 語 *[go = "language"]*.

Unknown foreign loanwords are identified by a simple maximal-katakana heuristic that returns the longest run of katakana characters. Despite its simplicity, this algorithm appears to work quite reliably when used in conjunction with the other mechanisms in our system.

Mimetic words in Japanese tend to follow simple ABAB or ABCABC patterns in hiragana or katakana, so we look for these patterns and propose them as adverb records.

The last category, typographical errors, remains mostly the subject for future work. Currently, we only address basic 二 (*kanji*) ↔ ニ (*katakana*) and へ (*hiragana*) ↔ ヘ (*katakana*) substitutions.

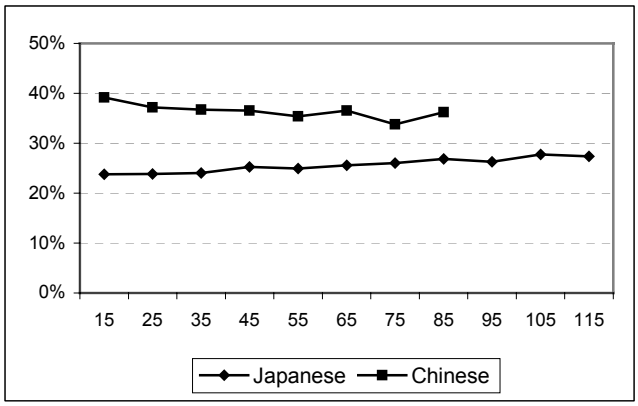

Figure 2: Worst-case segmenter precision (y-axis) versus sentence length (x-axis - in characters)

7 Evaluation

Our goal is to improve the parser coverage by improving the recall in the segmenter. Evaluation of this component is appropriately conducted in the context of its impact on the entire system.

7.1 Parser Evaluation

Running on top of our segmenter, our current parsing system reports ~71% coverage[†] (i.e., input strings for which a complete and acceptable sentential parse is obtained), and ~97% accuracy for POS labeled breaking accuracy. A full description of these results is given in [Suzuki00].

7.2 Segmenter Evaluation

Three criteria are relevant to segmenter performance: recall, precision and speed.

7.2.1 Recall

Analysis of a randomly chosen set of tagged sentences gives a recall of 99.91%. This result is not surprising since maximizing recall was a primary focus of our efforts.

The breakdown of the recall errors is as follows: missing proper nouns = 47%, missing nouns = 15%, missing verbs/adjs = 15%, orthographic idiosyncrasies = 15%, archaic inflections = 8%.

It is worth noting that for derived forms (those that

[†] Tested on a 15,000 sentence blind, balanced corpus. See [Suzuki00] for details.

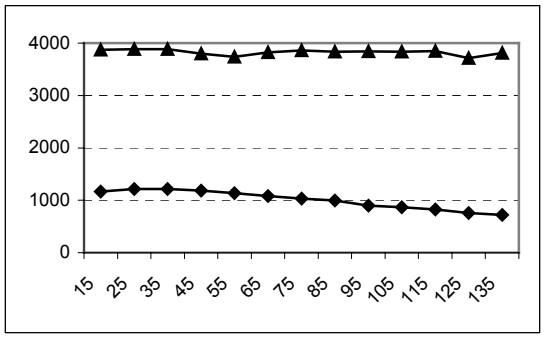

Figure 3: Characters/second (y-axis) vs. sentence length (x-axis) for segmenter alone (upper curve) and our NL system as a whole (lower curve)

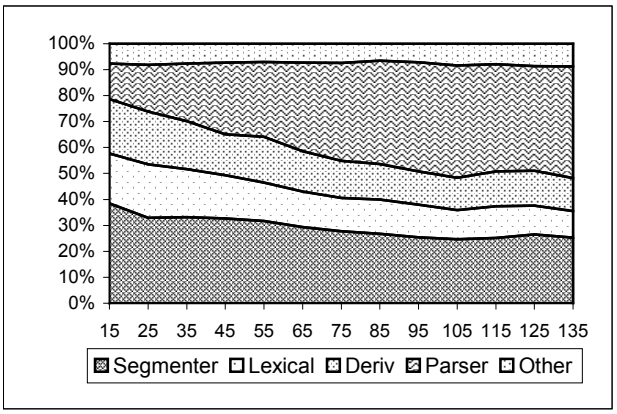

Figure 4: Percentage of time spent in each component (y-axis) vs. sentence length (x-axis)

are handled in the derivational assembly component), the segmenter is considered correct as long as it produces the necessary base records needed to build the derived form.

7.2.2 Precision

Since we focused our efforts on maximizing recall, a valid concern is the impact of the extra records on the parser, that is, the effect of lower segmenter precision on the system as a whole.

Figure 2 shows the baseline segmenter precision plotted against sentence length using the 3888 tagged sentences[‡]. For comparison, data for Chinese[§] is included. These are baseline values in the sense they represent the number of records looked up in the lexicon without application of any heuristics to suppress invalid records. Thus, these numbers represent worst-case segmenter precision.

The baseline precision for the Japanese segmenter averages 24.8%, which means that a parser would need to discard 3 records for each record it used in the final parse. This value stays fairly constant as the sentence length increases. The baseline precision for Chinese averages 37.1%. The disparity between the Japanese and Chinese worst-case scenario is believed to reflect the greater ambiguity inherent in the Japanese writing system, owing to orthographic variation and the use of a syllabic script.

[‡] The tagging was obtained by using the results of the parser on untagged sentences.

[§] 3982 sentences tagged in a similar fashion using our Chinese NLP system.

Using conservative pruning heuristics, we are able to bring the precision up to 34.7% without affecting parser recall. Primarily, these heuristics work by suppressing the hiragana form of short, ambiguous words (like き *[ki="tree, air, spirit, season, record, yellow,…"]*, which is normally written using kanji to identify the intended sense).

7.2.3 Speed

Another concern with lower precision values has to do with performance measured in terms of speed.

Figure 3 summarizes characters-per-second performance of the segmentation component and our NL system as a whole (including the segmentation component). As expected, the system takes more time for longer sentences. Crucially, however, the system slowdown is shown to be roughly linear.

Figure 4 shows how much time is spent in each component during sentence analysis. As the sentence length increases, lexical lookup, derivational morphology and "other" stay approximately constant while the percentage of time spent in the parsing component increases.

Table 5 compares parse time performance for tagged and untagged sentences. This table[**] quantifies the potential speed improvement that the parser could realize if the segmenter precision was improved. Column A provides baseline lexical lookup and parsing times based on untagged input.

[**] Note that segmenter time is not given this table because it would not be comparable to the hypothetical segmenters devised for columns B and C.

		A	B	C
Lexical processing		7.661 s	2.510 s	2.324 s
Parsing		13.480 s	8.865 s	7.179 s
Other		4.195 s	3.620 s	3.519 s
Total		25.336 s	14.995 s	13.022 s
Percent Improvement	Overall	-	40.82%	48.60%
	Lexical	-	67.24%	69.66%
	Parsing	-	34.24%	46.74%
	Other	-	13.71%	16.11%

Table 5: Summary of performance (speed) experiment where untagged input (A) is compared with space-broken input (B) and space-broken input with POS tags (C).

Columns B and C give timings based on a (hypothetical) segmenter that correctly identifies all word boundaries (B) and one that identifies all word boundaries and POS (C)[††]. C represents the best-case parser performance since it assumes perfect precision and recall in the segmenter. The bottom portion of Table 5 restates these improvements as percentages.

This table suggests that adding conservative pruning to enhance segmenter precision may improve overall system performance. It also provides a metric for evaluating the impact of heuristic rule candidates. The parse-time improvements from a rule candidate can be weighed against the cost of implementing this additional code to determine the overall benefit to the entire system.

8 Future

Planned near-term enhancements include adding context-sensitive heuristic rules to the segmenter as appropriate. In addition to the speed gains quantified in Table 5, these heuristics can also be expected to improve parser coverage by reducing resource requirements.

Other areas for improvement are unfound word models, particularly typographical error detection, and addressing the issue of probabilities as they apply to orthographic variants. Additionally, we are experimenting with various lexicon formats to more efficiently support Japanese.

[††] For the hypothetical segmenters, our segmenter was modified to return only the records consistent with a tagged input set.

9 Conclusion

The complexities involved in segmenting Japanese text make it beneficial to treat this task independently from parsing. These separate tasks are each simplified, facilitating the processing of a wider range of phenomenon specific to their respective domains. The gains in robustness greatly outweigh the impact on parser performance caused by the additional records. Our parsing results demonstrate that this compartmentalized approach works well, with overall parse times increasing linearly with sentence length.

10 References

[Fuchi98] Fuchi,T., Takagi,S., "Japanese Morphological Analyzer using Word Co-occurrence", ACL/COLING 98, pp409-413, 1998.

[Hisamitsu90] Hisamitsu,T., Nitta,Y., Morphological Analyis by Minimum Connective-Cost Method", SIGNLC 90-8, IEICE pp17-24, 1990 (in Japanese).

[Jensen93] Jensen,K., Heidorn,G., Richardson,S., (eds.) "Natural Language Processing: The PLNLP Approach", Kluwer, Boston, 1993.

[Kitani93] Kitani,T., Mitamura,T., "A Japanese Preprocessor for Syntactic and Semantic Parsing", 9th Conference on AI in Applications, pp86-92, 1993.

[Kurohashi94] Kurohashi,S., Nakamura,T., Matsumoto,Y., Nagao,M., "Improvements of Japanese Morphological Analyzer JUMAN", SNLR, pp22-28, 1994.

[Kurohashi98] Kurohashi,S., Nagao,M., "Building a Japanese Parsed Corpus while Improving the Parsing System", First LREC Proceedings, pp719-724, 1998.

[Nagata94] Nagata,M., "A Stochastic Japanese Morphological Analyzer Using a Forward-DP Backward-A* N-Best Search Algorithm", COLING, pp201-207, 1994.

[Richardson98] Richardson,S.D., Dolan,W.B., Vanderwende,L., "MindNet: Acquiring and Structuring Semantic Information from Text", COLING/ACL 98, pp1098-1102, 1998.

[Suzuki00] Suzuki,H., Brockett,C., Kacmarcik,G., "Using a broad-coverage parser for word-breaking in Japanese", COLING 2000.

[Wu98] Wu,A., Zixin,J., "Word Segmentation in Sentence Analysis", Microsoft Technical Report MSR-TR-99-10, 1999.