AN EFFICIENT TREATMENT OF JAPANESE VERB INFLECTION FOR MORPHOLOGICAL ANALYSIS

Toru Hisamitsu and Yoshihiko Nitta
Advanced Research Laboratory, Hitachi, Ltd.
Hatoyama, Saitama 350-03, JAPAN
{hisamitu, nitta}@harl.hitachi.co.jp

ABSTRACT

Because of its simple appearance, Japanese verb inflection has never been treated seriously. In this paper we reconsider traditional lexical treatments of Japanese verb inflection, and propose a new treatment of verb inflection which uses newly-devised segmenting units. We show that our proposed treatment minimizes the number of lexical entries and avoids useless segmentation. It requires 20 to 40% less chart parsing computation and it is also suitable for error correction in optical character readers.

Introduction

In this paper we focus on lexical entries for coping with Japanese verb inflection. The problem of treating verb inflection comes from the nature of written Japanese, in which word boundaries are not usually indicated explicitly. The morphological analyzer must therefore check for the existence of a verb and its inflection at each position in an input character string.

As a consequence, an awkward treatment of verb inflection may result unacceptably low computational efficiency.

Japanese verb inflection seems to be quite simple. Therefore, it has never been a central subject of natural language processing (NLP) studies. It is also because, in the early stages of Japanese NLP, the most time-consuming process of the Japanese morphological analysis (JMA) was found to be accessing the dictionary stored in a secondary memory. Therefore greater effort was put into designing the dictionary data structure and methods for quick access.

The situation, however, has changed. Highly efficient data structures based on the TRIE structure seem to have finally solved the data structure problems (for instance, Morimoto and Aoe, 1993), and the access problem is also being resolved by the emergence of cheap main memory on which the dictionary can be stored directly, and a dictionary-accessing chip that can access the dictionary thousands of times faster (Fukushima, 1991). As a result, problem of treating Japanese verb inflection is becoming more important.

Although phonological description of Japanese verb inflection is highly simple, it cannot be applied to JMA directly. Because each Japanese *hiragana* phonogram basically corresponds to a consonant-vowel pair, *not* to a phoneme. On the other hand, traditional *school grammar* gives a description based on the ordinary Japanese writing system, and has thus been widely used in JMA. However it is neither as rational as the phonological description nor is it the most efficient from a computational viewpoint.

We reconsider lexical entries for verb inflection and propose a new method for segmenting verbal complexes. Though our method is based on the ordinary Japanese writing system, it has various advantages over existing ones: 1) it minimizes the number of lexical entries together with avoiding useless segmentation; 2) it requires 20 to 40% less chart parsing computation, where the parser is based on dynamic programming and suitable for robust analysis; 3) it is also suitable for error correction in OCRs; 4) it requires a smaller incident matrix than other treatments, making the morphological analyzer easier to construct and maintain.

Section 1 overviews descriptions of Japanese verb inflection in terms of phonology and in terms of traditional *school grammar*. **Section 2** reviews three different treatments of verb inflection in NLP, which are based on the two descriptions in **section 1**. **Section 3** introduces our proposed treatment, and **section 4** shows the advantages of our treatment from several aspects, including a quantitative comparison of the computational efficiency of a chart parser.

1 Descriptions of Japanese Verb Inflection

Japanese verbs can be roughly classified into three groups as shown in **Table 1**. The number of *regular verbs* amounts to several thousand (our dictionary for JMA has about 3000 *regular verbs*). *Regular verbs* are classified into two groups: *consonant-stem verbs* whose stems end with consonants, and vowel-stem verbs whose stems end with vowels. *Sahen-verbs* are also classified into two groups: *verbal nouns*, whose stems can be used as nouns, and the others. This is the largest of the three groups (our dictionary has about 6000 verbs in class II). The number of *irregular verbs* is negligibly small.

group		Examples
regular verbs	consonant-stem verbs	*tob-u* (to fly), *kak-u* (to write), *kes-u* (to extinguish), ...
	vowel-stem verbs	*mi-ru* (to see), *ki-ru* (to wear), *sake-ru* (to avoid), ...
sahen-verbs	verbal nouns	*kenkyuu-suru* (to study), *kopii-suru* (to copy), ...
	others	*yuttari-suru* (to relax), *guttari-suru* (to be exhausted),..
irregular verbs		kuru (to come), suru (to do)

Table 1 Classification of Verbs

In terms of inflection processing, *Sahen-verbs* are the easiest of the three: their stems precede the special verb "*s-uru*" (to do), and inflectional affixes are attached to its stem '*s*'. Thus their inflection can be reduced to the inflection of "*s-uru*" and we can treat them by registering all inflectional forms in the dictionary. From the same reason *irregular verbs* are also easy to treat. Thus the central problem is treating the inflection of *regular verbs*. In the following, we focus on the treatment of these verbs.

First of all, we give two descriptions of the inflection of Japanese *regular verbs*. One is based on phonology, the other on the traditional *school grammar.*

1.1 Phonological Description

In Japanese, morphemes which correspond to "Past / Non-past", "Causative", "Passive", and so on directly follow a verbal stem as inflectional affixes. The first study of phonological analysis of Japanese verb inflection was done by an American linguist B. Bloch (Bloch, 1946). Unlike traditional *school grammar*, phonological description is based purely on phonemes, not on Japanese phonograms. A standard phonological description of Japanese *regular verbs* is shown in **Table 2**.

		consonant-stem verb	vowel-stem verb
Example		*kes-u* (to extinguish)	*mi-ru* (to see)
Indicative	Past	*-ita*	*-ta*
	Non-past	*-u*	*-ru*
Presumptive	Past	*-itaroo*	*-taroo*
	Non-past	*-oo*	*-roo*
Imperative		*-e*	*-ro* *-yo*
Hypothetical	Provisional	*-eba*	*-reba*
	Conditional	*-itara*	*-tara*
Participal	Infinitive	*-i*	- ϕ
	Gerund	*-ite*	*-ite*
	Alternative	*-itari*	*-itari*
Negative		*-ana*	*-na*
Causative		*-ase*	*-sase*

Table 2
Verb Inflection (Phonological Description)

For example, the inflection of a verb "消*s-u*" (*kes-u*: to extinguish) is as follows:

kes / ana / i, 消さない (*kesanai*: Negative);
kes / ase / ru, 消させる (*kesaseru*: Causative);

....

kes / u, 消す (*kesu*: Non-past);
kes / eba, 消せば (*keseba*: Provisional);
kes / e, 消せ (*kese*: Imperative).

Consonant-stem verbs have nine consonants {*b, g, k, m, n, r, s, t, w*} as their stem endings. According to phonological transformation, they are classified into six groups {*b, m, n*}, {*k*}, {*g*}, {*r, t*}, {*w*} and {*s*}. For instance, if x ∈ {*b, m, n*}, then the following transformation occurs:

$[[\_x]_{VS}\ ita]_V$ -----> $[\_nda]_V$,

where '_x' stands for a verbal stem whose ending is 'x', '$_{VS}$' for the boundary of the verbal stem and '$_V$' for the boundary of the inflected verb respectively. This transformation is called *Onbin*. For example,

yom- + *-ita* ----> *yonda*.
(to read) (Past)

1.2 Traditional *School Grammar*

As stated in the **introduction**, the phonological analysis of the previous subsection cannot be directly applied to JMA. Because each *hiragana* corresponds to a consonant-vowel pair, some phonological morphemes, such as '*ana*' and '*ase*' do not appear in character strings. For example, in the character string "消さない" (*kesanai:* not to extinguish), the stem '*kes*' and the negative affix 'ana' are glued together to form "消さな(*kesana*)". This is why the *school grammar* description is a little bit complex. The *school grammar* considers the indicative non-past form of a verb to be the "basic form". Verbs are "transformed" when inflectional affixes are attached. This transformation is called *Katsuyou*, and is illustrated in **Table 3.**

	Godan (consonant-stem verb)				*Kami-ichidan*	*Shimo-ichidan*
	ka-gyou	sa-gyou		wa-gyou	(vowel-stem verb)	
Example	書・く (*kak-u:* to write)	消・す (*kes-u:* to extinguish)		会・う (*aw-u:* to meet)	見・る (*mi-ru* : to see)	着・る (*ki-ru* : to wear)
Mizen (Irrealis)	ーか (*-ka*) ーこ (*-ko*)	ーさ (*-sa*) ーそ (*-so*)		ーわ (*-wa*) ーお (*-o*)	—	—
Renyou (Adverbial)	ーき (*-ki*) ーい (*-i*)	ーし (*-si*)	...	ーい (*-i*) ーっ (*-t*)	—	—
Renntai (Attributive)	ーく (*-ku*)	ーす (*-su*)		ーう (*-u*)	ーる (*-ru*)	ーる (*-ru*)
Shuushi (Conclusive)	ーく (*-ku*)	ーす (*-su*)		ーう (*-u*)	ーる (*-ru*)	ーる (*-ru*)
Katei (Hypothetical)	ーけ (*-ke*)	ーせ (*-se*)		ーえ (*-e*)	ーれ (*-re*)	ーれ (*-re*)
Meirei (Imperative)	ーけ (*-ke*)	ーせ (*-se*)		ーえ (*-e*)	ーれ　ーよ (*-re*), (*-yo*)	ーれ　ーよ (*-re*), (*-yo*)

Table 3
Verbal Inflection (*School Grammar*)

This time the *Katsuyou* of "消す" is described as follows:

消す＋な＋い ---> 消さない (*kesanai*),
to extinguish + Neg. + Non-past,
transformation 消す ---> 消さ;
消す＋せる ---> 消させる (*kesaseru*),
to extinguish + Caus. +Non-past,
transformation 消す ---> 消さ;

....

消す＋ば ---> 消せば (*keseba*),
to extinguish + Prov.,
transformation 消す ---> 消せ;
消す＋せ ---> 消せ (*kese*),
to extinguish + Imp.,
transformation 消す -> 消せ.

The underlined *hiragana* above are called *Katsuyougobi* (inflectional endings), and the inflected forms are called *Katsuyoukei.* Corresponding to the *Onbin* transformation stated in subsection **1.1**, an additional transformation is needed. For example,

読む＋た ----> 読んだ,
to read + Past,
transformation 読む ----> 読ん，　た ----> だ.

Although the description above lacks uniformity and seems to be far more complicated than phonological description, traditional JMAs have followed this description.

2 Existing Approaches

In this section, we sketch some methods of inflection analysis based on the two descriptions stated in the previous section.

2.1 Phonological Method

To use phonological description for verb inflection analysis, one first needs to convert the *hiragana* in an input character string into a string of Roman characters (*romaji*) corresponding to the Japanese phonemes. In this way, morphemes such as '*ana*' and '*ase*' become observable in the character string. Lexical entries for the inflection analysis of *regular verbs* are shown in **Table 4.**

entry	comments
消*s*	stem
:	
ana	Negative
ase	Causative
are	Passive
ita	Past
:	

Table 4
Examples of Lexical Entries (Phonological Method)

For example, "消さなかった" (*kesanakatta*: did not extinguish) is analyzed as follows:

消さなかった ---> 消*sanakatta*
---> 消*s* / *ana* / *katta*
kes: to extinguish / *ana*: Neg. / *katta*: Past

We will refer to this method with the abbreviation **PM** in the rest of this paper.

In the case of our dictionary, which includes 2807 *regular verbs*, an extra 1598 allomorphs (morpheme that are transformed from their *basic form*) are registered to cope with *Onbin* transformations of *regular verbs*.

The disadvantage of **PM** is that the target character strings must be lengthened as they are analyzed. In particular, character sequences including no *kanji*, which must be treated in *kana-kanji* conversion, are doubly lengthened. To make matters worse, for all the vowels *a, i, u,* *e*, and *o*, there are lexical entries with a single vowel. These facts deteriorate the computational efficiency. Thus this framework is suitable for generation (Kamioka, Tsuchiya and Anzai, 1989) but not for JMA.

2.2 *School Grammar* Method

Almost all existing systems employ inflectional analysis based on the school grammar. In this framework *kana*-to-*romaji* conversion is not necessary. There are two different lexical treatments for allomorphs.

2.2.1 Allomorph Expansion

The simplest method is to register all *Katsuyoukeis* as lexical entries (see **Table 5**).

For example, allomorphs of "消す(*kes-u*: to extinguish)", {消さ, 消し, 消す, 消せ1, 消せ2, 消そ} are all registered. Using these lexical entries, the example in subsection **2.1** is analyzed as follows:

消さなかった (*kesanakatta*)
---> 消さ / なかっ / た
kesa : to extinguish / *nakat* : Neg. / *ta* : Past

This method is referred to as **SG-I** in the rest of this paper.

entry	comments
消さ	*Mizenkei 1* (Irrealis Form)
消し	*Renyookei* (Adverbial Form)
消す	*Rentaikei* (Attributive Form)
消す	*Shuushikei* (Basic Form)
消せ	*Kateikei* (Hypothetical Form)
消せ	*Meireikei* (Imperative Form)
消そ	*Mizenkei 2* (Cohortative Form)
⋮	
な	Negative
せ	Causative
れ	Passive
た	Past
⋮	

Table 5
Examples of Lexical Entries (SG-I)

If **SG-I** is employed, an additional 11652 allomorphs requires to be registered in our dictionary to cope with *Katsuyou* transformation of *regular verbs*. This number of allomorphs is far larger than the true number of verbs: and explains why this method is not usually used in existing systems, especially those developed when memories were much more expensive.

2.2.2 Separating Inflectional Endings

The most popular treatment of *Katsuyou* involves separating inflectional endings and registering them as lexical entries (see **Table 6**).

Since the number of inflectional endings of *regular verbs* is 76, the number of lexical entries is far smaller than in **PM** or **SG-I**. For this reason, this method has been considered to be the best one. This time the same example is analyzed as follows:

消さなかった ----> 消 / さ / なかっ / た
kesanakatta *ke*[*s*]: to extinguish / *sa:* ϕ /
[*a*]*nakat*: Neg. / *ta*: Past

We will refer this method as **SG-II** in the rest of this paper.

entry	comments
消	stem
⋮	
さ	*Mizenkei* inflectional ending1
し	*Renyookei* inflectional ending
す	*Rentaikei* inflectional ending
す	*Shuushikei* inflectional ending
せ	*Kateikei* inflectional ending
せ	*Meireikei* inflectional ending
そ	*Mizenkei* inflectional ending2
⋮	
な	Negative
せ	Causative
れ	Passive
た	Past
⋮	

Table 6
Examples of Lexical Entries (SG-II)

However, analysis by **SG-II** requires one more segmentation than **PM** and **SG-I**. Worse still, the segment / さ/ has no meaning, thus this segmentation is useless. Since memories have become much lower in price, this problem cannot be disregarded.

3 Proposed Lexical Treatment of Japanese Verb Inflection

In the previous section we described three different lexical treatments. Here we summarize their advantages and problems:

1) **PM** is the simplest but is not directly applicable to ordinary written character strings.

2) **SG-I** realizes the minimum segmentation but requires a large number of allomorphs amounting to several times the original number of *regular verbs*.

3) **SG-II** requires the smallest number of lexical entries, but causes useless segmentations.

Only our proposed lexical treatment can solve these problems.

Let us explain our approach using the same example. In **PM**, the character string "消さなかった (*kesanakatta*)" is analyzed as "消*s*/*ana*/*katta*", where the ending consonant '*s*' of the stem "消*s*" and the head vowel '*a*' of the affix '*ana*' come from the phonogram 'さ(*sa*)'. Here recall that neither '*s*' nor '*a*' itself has a corresponding phonogram in the original character string. The school grammar description gives an observable lexical entry '消さ' by concatenating the head vowel '*a*' of '*ana*' to the tail of '消*s*'. It may be linguistically appropriate, but computationally not; there can be an alternative.

We attach the consonant '*s*' to the head of '*ana*' and generate an entry 'さな (*sana* =*s*+*ana*)' as a kind of an allomorph of '*ana*'. At the same time, the stem '消' is marked as a morpheme which can only be followed by "*s*-attached inflectional affixes", that is, {させ (*s*+*ase*: Causative), され (*s*+*are*: Passive), さな (*s*+*ana*: Negative,....}. Other lexical entries are generated in the same manner (see **Table 7**).

entry	comments
消	stem
:	
さな	Negative (s + *ana*)
させ	Causative (s + *ase*)
され	Passive (s + *are*)
した	Past (s + *ita*)
:	

Table 7
Examples of Lexical Entries (Proposed Method)

This time the previous example is analyzed as follows:

消さなかった (*kesanakatta*)
---> 消 / さな / かった
ke(*s*): to extinguish / *sana*: Neg. / *katta*: Past

It is obvious that this segmentation gives exactly the same semantic information as the other methods. This time the number of "allomorphs" is only 125, which is comparable to one of **SG-II**. On the other hand, the number of segments is as same as that of **SG-I** in this example. In the next section we discuss the advantages of our proposed method.

4 Advantages of Proposed Lexical Treatment

4.1 The Number of Allomorphs

As stated in the previous sections, **SG-I** and our proposed method require almost the same number of allomorphs, which is far smaller than that of the other methods.

4.2 Quantitative Comparison of Parsing Efficiency

In order to compare the computational efficiency of each method, we used a chart parsing algorithm (Hisamitsu and Nitta, 1991) and three dictionaries based on **SG-I**, **SG-II**, and the proposed method. Here we only sketch the outline of the algorithm (See **Fig.1**).

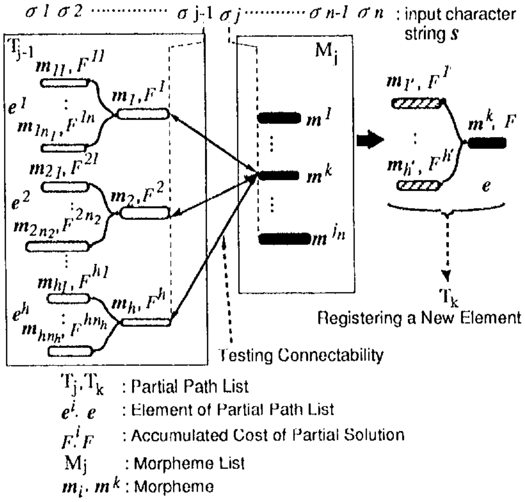
σ1 σ2 ··············· σj-1 σj ··········· σn-1 σn : input character string s
Tj-1
m11, F11
e1
m1, F1
m1n1, F1n1
m21, F21
e2
m2, F2
m2n2, F2n2
mh1, Fh1
eh
mh, Fh
mhnh, Fhnh
Mj
m1
mk
min
m1', F1'
mk, F
mh', Fh'
e
Tk
Registering a New Element e
Testing Connectability
Tj, Tk : Partial Path List
ei, e : Element of Partial Path List
Fi, F : Accumulated Cost of Partial Solution
Mj : Morpheme List
mi, mk : Morpheme

Figure 1 Illustration of Chart Parsing

Here s denotes an input string $\sigma_1 \cdots \sigma_n$. A candidate-word lattice $\{M_1, \cdots, M_n\}$ is used for recording candidate morphemes, where M_j records the morphemes extracted at position j. Partial path lists $\{T_1, \cdots, T_n\}$ are used for recording the fragments of partial solutions, where T_j contains fragments of partial solutions which reach the j-th position in s. An element in T_k ($1 \leqq k \leqq n$) has the form $\{\boldsymbol{m}, C, \{<\boldsymbol{m}_1, C_1>, \cdots <\boldsymbol{m}_k, C_k>\}\}$where $\boldsymbol{m}$ is the last morpheme of partial solutions $\boldsymbol{a}_1, \cdots \boldsymbol{a}_k$, C is their common cost, and $<\boldsymbol{m}_j, C_j>$ is the preceding morpheme of $\boldsymbol{m}$ at $\boldsymbol{a}_j$; $\{<\boldsymbol{m}_1, C_1>, \cdots, <\boldsymbol{m}_k, C_k>\}$ is regarded as a "pointer" for tracing solutions backward. The elements of T_k are calculated using T_j and M_{j+1}, where $1 \leqq j \leqq n-1$, and $j \leqq k \leqq n$. Once the Partial path lists $\{T_1, \cdots, T_n\}$ is constructed, the solutions are extracted by depthfirst backtracking to trace pointers backward.

To enable a quantitative comparison, we use the following three measures, which reflect the efficiency of chart parsing and are independent of implementing variations:

A) Total number of morphemes contained in morpheme lists $\{M_1,..., M_n\}$.

B) Total number of tests which check for the connectability between partial-solution fragments in T_j and morphemes in M_{j+1}.

C) Total number of elements contained in partial path lists $\{T_1,..., T_n\}$.

Figure 2 compares the three methods. The comparison was made using *100* sentences taken from *Nikkei Shinbun*, which contain a total of 5286 characters. The dictionary contained about 60000 words. Our proposed method is far more efficient than the most popular method **SG-II**, and its efficiency is comparable to that of **SG-I**.

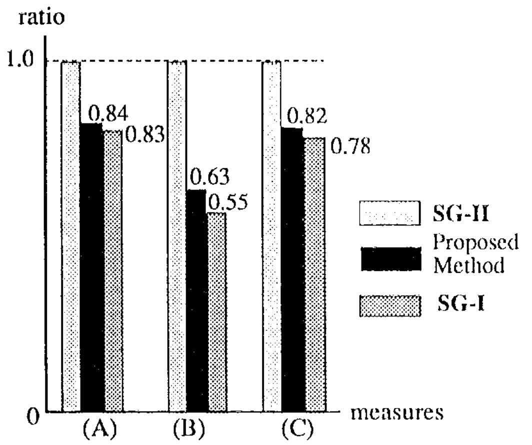

Figure 2 Comparison of Three Methods

4.3 Application to OCR Error Correction

Recently, morphological analysis has been applied more extensively to various systems, especially to error correction in OCRs (optical character readers). In general, a character recognition module outputs a sequence of lists which include candidate characters at each position in the pattern sequence. Each candidate character is given a positive confidence ratio (see **Fig.3**). We call the sequence of lists "candidate character lattice". Note that the character string of the top candidate characters, which is the final output of a bare character recognizer, is not necessarily the correct sentence.

To correct the errors, we use two main processes: 1) constructing a candidate words lattice by using the candidate character lattice and a dictionary; 2) extracting plausible word sequences from the candidate words lattice.

Generally process 1) is time-consuming, because we need to find potential word candidates from the combination of candidate characters at each position. To avoid combinatorial explosion, a skillful method has been widely used in error correction (Takao and Nishio, 1989): at each position, first extract all words whose first character matches the top candidate character at the position, secondly compare those words with the candidate character lattice.

input: "消さなかった"

Position	Candidate Characters
1	((消, 0.79) (朋, 0.72) (明, 0.64) ...)
2	((ざ, 0.81) (さ, 0.79) (き, 0.58) ...)
3	((な, 0.72) (だ, 0.69) (た, 0.65) ...)
4	((か, 0.89) (が, 0.86) (力, 0.75) ...)
5	((っ, 0.82) (つ, 0.81) (て, 0.77) ...)
6	((た, 0.84) (だ, 0.82) (な, 0.76) ...)

Figure 3

Example of Candidate Character Lattice

For this method to be effective, the lexical entries should be as long as possible, because a longer entry is easier to recover when one or two characters are mis-recognized. There should also be as few entries as possible whose first characters coincide.

In terms of the former requirement, our proposed method is obviously better than **SG-II**. Although **SG-I** results in the longest lexical entries, it is the worst in terms of the latter requirement because each verb has basically six allomorphs in the dictionary, and the first characters of these words are inevitably the same. For this reason, our experiments have shown that error correction based on the **SG-I** dictionary is 3.6 times more time-consuming than error correction based on the proposed dictionary. Thus our proposed method is the most suitable for this purpose.

4.4 Other Advantages

4.4.1 Incident Matrix Size

Compared with **SG-I** and **SG-II**, our proposed method reduces the size of the incident matrix, because, using our lexical entries makes it unnecessary to check for connection between a *Katsuyougobi* and various inflectional affixes.

4.4.2 The Number of Free Parameters in Statistical Heuristics

In obtaining a (simple) Markov model, one may notice a major difference between the proposed method and **SG-I. Figure 4 (a)** illustrates the linguistically possible incidence between our lexical entries including a verbal stem v. To construct a probabilistic likelihood function, one needs to estimate all of the free parameters p_{wv}, where p_{wv} denotes the transition probability from word w to v. Since a verbal stem can succeed almost all grammatical categories, the number of parameters $\{p_{wv}\}$ (= N(v)) is almost equal to the number of all categories.

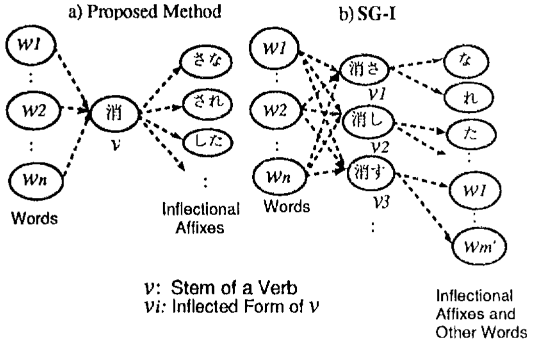

Figure 4
Difference Between Two Methods
in Constructing Probabilistic Models

With **SG-I**, the number of parameters $\{p_{wv1}, p_{wv2},...\}$ is about seven times as large as N(v), where 'v_i' denotes a *Katsuyoukei* of the verb v **(Fig.4 (b))**. In other words, the number of free parameters is inevitably increased by using **S G-I**.

5 Further Study

In subsection 4.2, we used a standard chart parser based on dynamic programming for the comparison. While the parser itself is robust and efficient, there are several kinds of parsing methods. For example, the *longest matching method* is popular. Actually, our lexical treatment is also effective for such a parsing strategy. We will also make an experimental comparison based on various parsing methods.

6 Conclusion

In this paper we reconsidered lexical entries for verb inflection and proposed a new way of segmenting verbal complexes that has various advantages over existing methods: 1) it minimizes the number of lexical entries and avoids useless segmentation; 2) it requires 20 to 40% less computation than standard chart parsing; 3) it is suitable for error correction in OCRs; 4) it requires a smaller incident matrix than other treatments, thus making it easier to construct and maintain the morphological analyzer; 5) it is the most suitable for obtaining statistical heuristics because it can intrinsically reduce the number of free parameters.

References

Bloch, B. (1946), "Studies in Colloquial Japanese, Part I, Inflection", *Journal of the American Oriental Society 66.*

Fukushima, T., (1991), "A Morpheme Extraction Hardware Algorithm and Its Implementation", *Transaction of IPSJ*, Vol. 32, No. 10, pp. 1259-1268.

Hisamitsu, T. and Nitta, Y. (1991), "A Uniform Treatment of Heuristic Methods for Morphological Analysis of Written Japanese", in *Proc. of the 2nd Japan-Australia Joint Symposium on Natural Language Processing*, pp. 46-57.

Kamioka, T., Tsuchiya, T. and Anzai, Y. (1989), "Generation and Representation of Predicate Complex", *Trans. of IPSJ*, Vol. 30, No. 4, pp. 457-466.

Morimoto, K. and Aoe, J. (1993), "Two Trie Structures for Natural Language Dictionaries", in *Proc. of Natural Language Processing Pacific Rim Symposium (NLPRS '93)*, pp. 302-311.

Takao, T. and Nishio, F. (1989), "Implementation and Evaluation of Post-Processing for Japanese Document Readers", *Trans. of IPSJ*, Vol. 30, No. 11, pp. 1394-1401.