IMPLEMENTING SCRAMBLING IN KOREAN: A PRINCIPLES AND PARAMETERS APPROACH

Franklin S. Cho
MIT Artificial Intelligence Laboratory
NE43-802
MIT, Department of Electrical Engineering and Computer Science
545 Technology Square, Cambridge, MA, 02139, USA
fscho@ai.mit.edu

3 July 1995

Summary

This paper describes how the most complete recent linguistic results on Korean scrambling (switching of word order) can be readily incorporated into an existing principles-and-parameters parser with minimal additional machinery. Out of all 29 sets of examples in chapters 2.2 and 3.2 of perhaps the most advanced linguistic analysis of Korean scrambling, (Lee, 1994), 26 sets of examples can be correctly parsed, greatly extending the variety of scrambling handled by any current parser. This approach is compared to other current approaches to scrambling, such as PRINCIPAR and Tree Adjoining Grammar.

Subject Areas: parsing, scrambling, Korean

Word Count:

1. INTRODUCTION

Scrambling is a complex yet common phenomenon in Korean that allows the apparent movement of a noun phrase over both short and long distances:

- Scrambling of more than one noun phrase that belongs to the same verb's argument structure, or "multiple scrambling". For example, in (1)(ii), "chayk" (book) moves in front of "Youlee", or even to the front of the sentence, as in (1)(iv).

(1)	(i)	Sunhee-ka	Youlee-eykey	[chayk hankwen]-ul	senmwulhayssta
		Sunhee-nom	Youlee-dat	[book one-volume]-acc	gave-a-present
		"Sunhee gave Youlee a book as a present."			
	(ii)	sunhee-ka	[chayk hankwen]-ul	youlee-eykey	senmwulhayssta
		Sunhee-nom	[book one-volume]-acc	Youlee-dat	gave-a-present
	(iii)	youlee-eykey sunhee-ka [chayk hankwen]-ul senmwulhayssta			
	(iv)	youlee-eykey [chayk hankwen]-ul sunhee-ka senmwulhayssta			

(v) [chayk hankwen]-ul sunhee-ka youlee-eykey senmwulhayssta

(vi) [chayk hankwen]-ul youlee-eykey sunhee-ka senmwulhayssta[1]

- No limit to the number of clauses that a scrambled element can cross, or "unbounded dependency," For example, in (2)(ii), "chayk" (book) can be arbitrarily far from its canonical argument position:[2]

(2) (i) John-i [Mary-ka [Sally-ka Bill-eykey chayk-ul cwuessta-ko] malhayssta-ko]
John-nom [Mary-nom [Sally-nom Bill-dat book-acc gave-compl] said-compl]
sayngkakhanta
think
"John thinks that Mary said that Sally gave Bill a book."

(ii) chayk-ul$_i$ [John-i [Mary-ka [Sally-ka Bill-eykey t_i cwuessta-ko] malhayssta-ko]
book-acc [John-nom [Mary-nom [Sally-nom Bill-dat gave-compl] said-compl]
sayngkakhanta]
think]

Handling scrambling correctly is very difficult for a parser. The reason is that adding a permutation component to generate all possible word orders independently of other grammatical constraints is easy. What is more difficult is to make scrambling interact correctly with all the *other* components of the grammar, for instance those that establish the interaction of scrambling with coreference. Consider these examples:

(3) (i) *Younghee-ka **ku**-eykey [**Minswu**-uy sacin]-ul poyecwuessta
Younghee-nom him-dat Minswu-gen picture-acc showed
'Younghee showed **him Minswu's** picture'

(ii) Younghee-ka [**Minswu**-uy sacin]$_i$-ul **ku**-eykey t_i poyecwuessta[3]

(iii) [**Minswu**-uy sacin]$_i$-ul Younghee-ka **ku**-eykey t_i poyecwuessta[4]

The coreference relation is indicated by bold face, and the filler-gap relation (or, equivalently, antecedent-trace relation) by coindexation. (3) shows that scrambling interacts with coreference: the scrambled versions (3)(ii) and (3)(iii) are acceptable, but the canonical version (3)(i) is not. So, scrambling "saves" a sentence in (3). Now, consider these examples:

(4) (i) [**Minswu**-uy tongsayng]-i **ku**-eykey sacin-ul poyecwuessta
Minswu-gen brother-nom him-dat picture-acc showed
'**Minswu's** brother showed **him** a picture.'

(ii) ***ku**-eykey [**Minswu**-uy tongsayng]-i sacin-ul poyecwuessta[5]

Conversely, the canonical version (4)(i) is acceptable but the scrambled version (4)(ii) is not. So, scrambling "destroys" a sentence in (4). Therefore, scrambling is much more complicated than simply generating correct word orders. In both (3) and (4), scrambling interacts with a

[1](Lee, 1994), example 7. See figure 1 in the text.
[2](Lee, 1994), p. 5
[3]Here, "t_i" denotes a link between the canonical argument position for "chayk"(book) and its actual position in the sentence. At this point, I remain theory-neutral as to the exact nature of "t_i". It could be implemented in several ways.
[4](Lee, 1994), example 81.
[5](Lee, 1994), example 82.

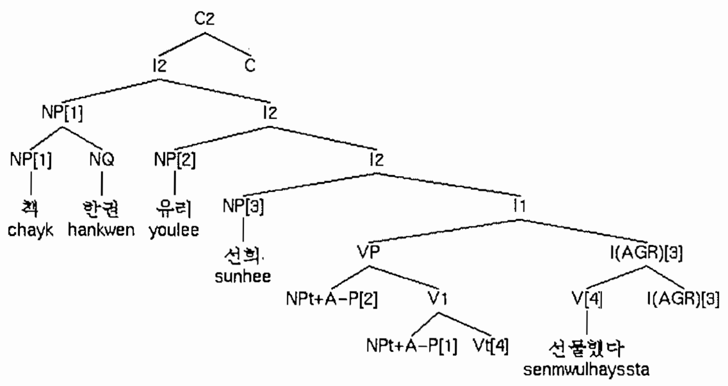


Figure 1: A Parsed Korean Example Sentence 1(vi)

coreference constraint referred to as "Condition C" in the linguistic literature. Roughly, Condition C states that a referring expression e.g., "John," "the house," must not be bound anywhere in a sentence.[6] (4)(ii) is ruled out by Condition C, since "Minswu," a referring expression, is bound by "ku." Such an interaction must be taken into account when parsing these examples. As far as it is known, the system presented here is the first that can correctly parse such a wide range of scrambling examples. Although no good quantitative measures are known to us, scrambling seems quite common in Korean, and is therefore important for parsing. Figure 1 shows an example of the parser's actual output on a scrambled example sentence.

2. IMPLEMENTATION

2.1. A Simple Scrambling Mechanism

Let us see how to parse scrambled sentences using a constraint-based parser, **Pappi** (Fong, 1991). As a first approximation, here is a simplified description of the scrambling mechanism (Figure 2), showing only the modules relevant to the scrambling-coreference relations. In subsequent sections, I will refine this analysis to accomodate new examples. (There are many more filters and generators in **Pappi** than the figure shows.)

The key idea behind constraint-based parsing is to reproduce complex surface sentence patterns by the interaction of separable but linked "modules," each handling a different kind of constraint. For instance, our examples (1)-(4) motivate four modules:

1. One to move NP's from their canonical locations.
2. One to coindex filler NP's with their gaps and other NP's.
3. One to check whether this indexing meets Condition C.
4. One to move variables into proper scope (assuming a typed first-order predicate calculus (FOPC) representation.)

[6]A node A binds another node B iff A and B are coindexed, and A c-commands B. A c-commands B iff the lowest branching node which dominates A also dominates B. For example, in $[_\alpha [...[\beta\ \theta]...]]$, α c-commands β and θ. This is the canonical definition of binding, and this definition will be modified later, as shown in Figure 4.

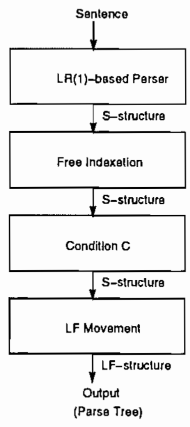


Figure 2: A Simple Scrambling Mechanism

First, a LR(1)-based bottom-up shift-reduce parser is used to recover the phrase structure. Note that this parser allows an NP to freely attach to the beginning of a sentence (or, equivalently, adjoin to an Inflection Phrase(IP)) or to the beginning of a verb phrase (or, equivalently, adjoin to a verb phrase(VP)), to account for scrambling.[7]

A mechanism such as the "Free Indexation" mechanism is called a "generator." A generator generates new structures based on the old structure that was passed to it. For example, the "Free Indexation" generator takes a parse tree without indices, and generates parse trees with indices assigned to each NP and trace (or, equivalently, gap). This generator generates all possible coindexations between the NP's and their gaps, and among the NP's.[9]

A mechanism such as the "Condition C" mechanism is called a "filter." A filter eliminates the wrong structures that enter it, and pass the correct structures to the next filter or generator. For example, the "Condition C" filter filters out every parse tree that violates the Condition C.

The "Logical Form (LF) Movement" mechanism performs two operations: first, it raises each quantifier (e.g., "every boy", "somebody") and attaches it to the beginning of the innermost sentence (or, equivalently, adjoins it to the nearest IP.) Then, it raises each wh-word (e.g., "who", "where") to the specifier position of the nearest Complementizer Phrase(CP).[10]

To summarize, after the phrase structure is recovered by the LR parser, these structures are passed through a series of filters and generators, until only the correct parses remain. The implementations are taken care of by following a generate and test paradigm (but in a more sophisticated way).

2.2. Subject Binding Generalization

However, this simple first order approximation does not suffice to cover all examples in Korean. Consider these examples:

(5)	(i)	*ku-ka	[**Minswu**-uy emma]-lul	coahanta
		he-nom	Minswu-gen mother-acc	like

[7]Also, the mechanism used to avoid "string vacuous scrambling" in Japanese, as described in (Fong, 1994) is used for Korean as well.[8] A "non-vacuous" or "visible" scrambling is a scrambling that "passes over" one or more overt elements (Fong, 1994).

[9]Please refer to (Fong, 1991) for the details on how the free indexation is implemented.

[10]Under the Government and Binding framework, the Complementizer Phrase (CP) immediately dominates the Inflection Phrase (or, equivalently, the Sentence), and wh-words move to the specifier position of a CP at Logical Form (LF) level. The specifier position is immediately dominated by the CP.

'**He** likes **Minswu's** mother.'

(ii) *[**Minswu**-uy emma]$_i$-lul **ku**-ka t_i coahanta[11]

(5)(i) is ruled out by Condition C, since "ku" binds "Minswu", a referring expression. (Please refer to Figure 3 for the definition of "binding."[12]) However, Condition C alone cannot explain why (5)(ii) is unacceptable, since nothing seems to bind "Minswu" (it is therefore free, unbound, and satisfies Condition C.) We can repair this problem by introducing a new definition of binding (defined in (Frank et al., 1992).) According to Lee, (5)(ii) is ruled out by the Subject Binding Generalization:

(6) **Subject Binding Generalization:** If X in subject position binds Y at Deep Structure (D-structure or, equivalently, canonical predicate-argument structure)[13], then X binds Y at all levels of representation (i.e., FOPC level and the surface level).

In (5)(ii), "ku" binds "Minswu" in D-structure, and therefore "ku" still binds "Minswu" in surface structure (S-structure.) Therefore, (5)(ii) is ruled out by Condition C, since "Minswu" is bound.

To implement this, I revised the definitions of binding in the parser. The original and the new definition of binding are shown below as flowcharts.

Here is the original definition of binding (Figure 3):

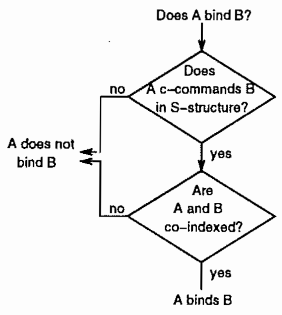


Figure 3: The Original Definition of Binding

Here is the new definition of binding (Figure 4):

[11](Lee, 1994), example 86

[12]A node A c-commands another node B iff the lowest branching node which dominates A also dominates B. For example, in [α [...[β θ]...]], α c-commands β and θ.

[13]Each scrambled element moves back to its canonical position at D-structure.

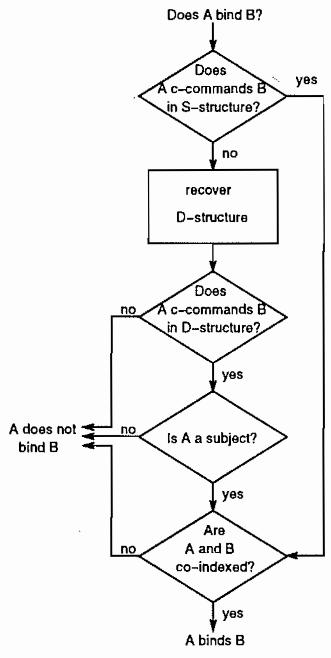


Figure 4: The New Definition of Binding

Notice that a change in the definition of binding automatically changes the definition of Condition C (since the code implementing Condition C calls the code that implements binding.) Also notice that the "subject" is defined as an NP with an "agent" thematic role (θ-role), following Lee's analysis.[14]

Here is how the D-structure is recovered from the S-structure: (1) recurse down the parse tree and replace each element by its D-structure element. For example, a head of a chain (or, equivalently, a filler) would be replaced by an empty element, and a trace (gap) would be replace by its antecedent (filler). (2) Delete any empty elements introduced by step (1).

2.3. Scrambling and Scope

Korean scrambling (as well as scrambling in other languages) is complicated further by examples such as these:

(7) (i) ne-nun [Minswu-ka nwukwu-lul coaha-nunci] a-ni
ne-nun Minswu-nom who-acc like-qm know-qm
'Do you know who Minswu likes?'

(ii) nwukwu$_i$-lul ne-nun minswu-ka t_i coaha-nunci a-ni
'Who do you know Minswu likes?' / 'Do you know who Minswu likes'[15]

In (7), (ii) has a different interpretation than (i). This is because scrambling interacts with scope interpretation. In (Lee, 1994), Lee claims that a scrambled wh-element (like "who" or "what") optionally reconstructs for scope interpretation. Reconstruction means that a scrambled NP optionally moves back to its unscrambled position.

[14]Please refer to (Fong, 1991) for the details on how θ-roles are assigned.
[15](Lee, 1994), example 160

Remember that at Logical Form (LF) interpretation (or, equivalently, scope interpretation at the FOPC level), the wh-word raises to the specifier position of the *nearest* CP (or Sentence.) The sentence is interpreted as a wh-question if a wh-word occupies the specifier position of the matrix (outer) CP. If a wh-word occupies the specifier position of the embedded (inner) CP, and the specifier position of the matrix (outer) CP is empty, the sentence is interpreted as a yes/no question.

In (7)(i), the wh-word in the embedded (inner) clause "nwukwu" raises to the nearest CP, and the whole sentence is interpreted as a yes/no question, as shown in (8):

(8) (i) [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ ne-nun [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ Minswu-ka *nwukwu* coaha-nunci]] a-ni]]
'The sentence before wh-raising'

(ii) [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ ne-nun [$_{\mathrm{CP}}$ *nwukwu*$_i$ [$_{\mathrm{IP}}$ Minswu-ka t_i coaha-nunci]] a-ni]]
'*nwukwu* is raised to the specifier position of the nearest CP'

In (8)(ii), wh-word occupies the specifier position of the embedded (inner) CP, and the specifier position of the matrix (outer) CP is empty, i.e., the sentence is interpreted as a yes/no question.

In (7)(ii), the whole sentence can be interpreted as a wh-question (as shown in (10)) as well as a yes/no question (as shown in (9).) For the yes/no interpretation (9), the scrambled wh-word reconstructs to its base position and then raises to the nearest CP.

(9) (i) [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ *nwukwu*$_i$ ne-nun [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ Minswu-ka t_i coaha-nunci]] a-ni]]
'The sentence before wh-raising (*nwukwu* is scrambled to the front of the sentence.)'

(ii) [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ ne-nun [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ Minswu-ka *nwukwu* coaha-nunci]] a-ni]]
'Scrambled *nwukwu* reconstructed.'

(iii) [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ ne-nun [$_{\mathrm{CP}}$ *nwukwu*$_i$ [$_{\mathrm{IP}}$ Minswu-ka t_i coaha-nunci]] a-ni]]
'*nwukwu* raised to the nearest CP.'

In (9)(iii), a wh-word occupies the specifier position of the embedded (inner) CP, and the specifier position of the matrix (outer) CP is empty, i.e., the sentence is interpreted as a yes-no question.

For the wh-interpretation of (7)(ii), the scrambled wh-word raises to the nearest CP, without undergoing reconstruction.

(10) (i) [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ *nwukwu* ne-nun [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ Minswu-ka coaha-nunci]] a-ni]]
'The sentence before wh-raising (*nwukwu* is scrambled to the front of the sentence.)'

(ii) [$_{\mathrm{CP}}$ *nwukwu*$_i$ [$_{\mathrm{IP}}$ t_i ne-nun [$_{\mathrm{CP}}$ [$_{\mathrm{IP}}$ Minswu-ka coaha-nunci]] a-ni]]
'Scrambled *nwukwu* is raised to the nearest CP without undergoing reconstruction.'

In (10)(ii), since a wh-word occupies the specifier position of the matrix (outer) CP, the sentence is interpreted as a wh-question. It is crucial to parse these examples correctly for a Korean Q/A system.

Here is how reconstruction is implemented: (1) Recurse down the parse tree and replace a scrambled wh-word with an empty element, and replace a trace (gap) of a wh-word with its antecedent (filler.) (2) Delete any empty elements introduced by step (1).

Reconstruction is incorporated into the "LF Movement" generator described in Figure 2. The original implementation of the "LF Movement" generator can be understood as two smaller generators serially linked (Figure 5):

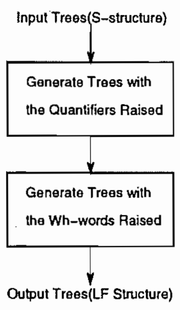

Input Trees(S-structure)
Generate Trees with the Quantifiers Raised
Generate Trees with the Wh-words Raised
Output Trees(LF Structure)

Figure 5: The Original Implementation of the LF Movement Generator

A new definition of the "LF Movement" generator is shown in Figure 6. (This definition will be revised in the following section).

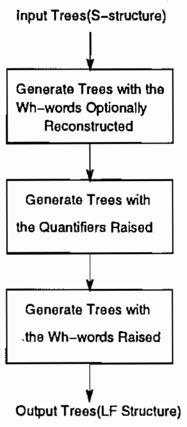

Input Trees(S-structure)
Generate Trees with the Wh-words Optionally Reconstructed
Generate Trees with the Quantifiers Raised
Generate Trees with the Wh-words Raised
Output Trees(LF Structure)

Figure 6: A New Implementation of the LF Movement Generator

Vacuous Wh-Chain Reconstruction The implemented reconstruction algorithm must be more sophisticated if it is to avoid any redundant parses.[16] Here, redundant parses mean that two parses have the same scope interpretations at LF (typed first order predicate calculus) level. Consider this example:

(11) (i)	**nwukwu**$_i$-lul	[**pro**	chinkwu]-ka	t_i	paypanhayss-ni
	who-acc	pro-gen	friend-nom		betrayed-Q

"**Who** did **his** friend betray"[17]

One possibility is to:

1. Reconstruct the chain (*nwukwu*$_i$,t_i), then
2. Raise "nwukwu" to the nearest CP at LF (or, equivalently, FOPC level.), as shown in (12)

[16]The author would like to acknowledge Dr. Sandiway Fong's help with implementing the algorithms outlined in this subsection (Vacuous Wh-Chain Reconstruction) and the next subsection (Reconstruction for Subject Binding.)
[17](Lee, 1994), example 78b.

(12) (i) $[_{CP}$ $[_{IP}$ **nwukwu**$_i$ [**pro** chinkwu]-ka t_i paypanhayss-ni]]
'The sentence before wh-raising (**nwukwu** is scrambled to the front of the sentence.)'
(ii) $[_{CP}$ $[_{IP}$ [**pro** chinkwu]-ka **nwukwu** paypanhayss-ni]]
'**nwukwu** is reconstructed.'
(iii) $[_{CP}$ **nwukwu**$_i$ $[_{IP}$ [**pro** chinkwu]-ka t_i paypanhayss-ni]]
'**nwukwu** is raised to the nearest CP after reconstruction.'

Compare this with:

1. No reconstruction, then
2. Raise "nwukwu" from the original scrambled position to the nearest CP at LF, as shown in (13)

(13) (i) $[_{CP}$ $[_{IP}$ **nwukwu**$_i$ [**pro** chinkwu]-ka t_i paypanhayss-ni]]
'The sentence before wh-raising (**nwukwu** is scrambled to the front of the sentence.)'
(ii) $[_{CP}$ **nwukwu**$_i$ $[_{IP}$ t_i [**pro** chinkwu]-ka paypanhayss-ni]]
'Scrambled **nwukwu** is raised to the nearest CP without undergoing reconstruction.'

These two parses make no scope distinction, and are therefore redundant. The solution to this "vacuous wh-chain reconstruction" problem is to reconstruct only long distance wh-chains, i.e. only if there is some scope distinction to be derived. This eliminates the first possibility above.
The revised reconstruction algorithm is implemented as follows:

1. Mark each element of a long distance wh-chain, then
2. Replace the trace (gap) with the antecedent (filler), and replace the antecedent (filler) with a null element, for each member of the chain. Mark any null element so introduced for deletion.
3. Delete all elements marked for deletion.

Recall that all reconstruction is optional, so even if the scrambling is long distance, reconstruction may not occur.

Reconstruction for Subject Binding Consider these examples:

(14) (i)

[**caki** chinkwu]$_i$-eykey	**nwukwuna**-ka t_i	komin-ul	thelenohnunta
self's friend-dat	everyone-nom	problem-acc	tell.

"**Everyone** tells **his/her** friend problems"[18]

(ii)

[**caki** uymwu]$_i$-lul	**nwukwuna**-ka t_i	chwungsilhi	ihaynghayssta
self's duty-acc	everyone-nom	faithfully	carried-out

"**Everyone** carried out **his/her** duty faithfully"[19]

When the quantifier "nwukwuna" is raised at LF, it produces a weak cross-over (WCO) violation.[20] Therefore, we need to avoid WCO violation by reconstructing the scrambled element which is bound by the subject (through the Subject Binding Generalization) before raising the quantifier.

[18](Lee, 1994), example 79b.
[19](Lee, 1994), example 80b.
[20]Weak crossover involves the coindexing of an empty category and a genitive inside an NP, as in **Who$_i$ does his$_i$ mother love e$_i$?* A WCO violation occurs when a wh-word or a quantifier raises from its D-structure position to the [spec, CP] and it "crosses over" a coindexed genitive inside an NP. The presence of a weak crossover makes the sentence unacceptable.

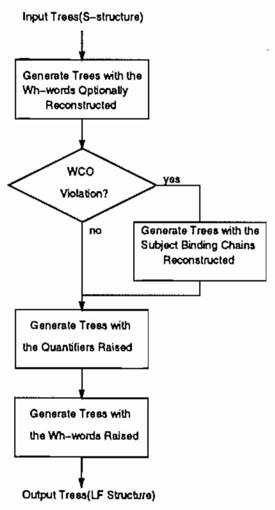

Input Trees(S-structure)
Generate Trees with the Wh-words Optionally Reconstructed
WCO Violation?
yes
no
Generate Trees with the Subject Binding Chains Reconstructed
Generate Trees with the Quantifiers Raised
Generate Trees with the Wh-words Raised
Output Trees(LF Structure)

Figure 7: The Final Implementation of the LF Movement Generator

Implementation:

1. Mark each element of a Subject Binding chain, then
2. Replace the trace (gap) with the antecedent (filler), and replace the antecedent (filler) with a null element, for each member of the chain. Mark any null element so introduced for deletion, then
3. Delete all elements marked for deletion.
4. Check that the reconstructed tree satisfies conditions A[21], B[22], and C. If the tree violates any of these conditions, then do not reconstruct (The generator outputs the input tree unchanged.)

The tentative solution is to only allow Subject Binding reconstruction as a last resort for WCO violations.[23]

Implementation Here is the algorithm that implements the ideas outlined above (Figure 7):

[21]Condition A states that an anaphor (e.g., "myself", "herself") must be bound within its governing category. The governing category of an NP is the smallest NP or Inflection Phrase (or, in standard notation, Sentence Phrase) containing that NP, its governor, and an "accessible" subject.

[22]Condition B states that a pronominal (e.g., "he", "they") must not be bound in its governing category.

[23]There is some overlap between the coverage of Subject Binding Generalization and reconstruction of the Subject Binding chain, but they have distinct functions, and both are needed. In the current implementation, Subject Binding Generalization is relevant when analyzing whether the sentence satisfies conditions A, B, and C in the S-structure. Subject Binding reconstruction is applied when the LF-structure is derived from the S-structure. So, without Subject Binding Generalization at the S-structure, reconstruction of the Subject Binding chain may never occur, since the parse tree may have been eliminated before reaching the LF-movement stage. Also, Subject Binding Generalization is relevant for both long and short distance scrambling, but reconstruction of the Subject Binding chain is only (optionally) applicable for long distance chains, as shown above.

3. COMPARISONS WITH OTHER SYSTEMS

To implement the scrambling mechanism described above, less than 150 lines of Prolog code need to be added to the standard **Pappi** framework. Why is scrambling relatively easy to implement in this way? Essentially, **Pappi** can easily handle reconstruction since the code that encodes principles directly deals with sentence structures. In order to handle reconstruction, **Pappi** optionally moves the scrambled wh-elements back to their base position, and then raises them to the nearest CP at logical form interpretation. This is difficult in other systems proposed to handle Korean. First, consider Lin's PRINCIPAR ((Dorr et al., 1995)). Since Lin's PRINCIPAR deals with the *description* of structures, it has difficulty dealing with reconstruction (unless the current design is drastically changed), since the messages it uses only pass *up* parse trees. In order for PRINCIPAR to handle reconstruction, the node representing a trace would have to know whether its antecedent is a wh-word, and decide if it should reconstruct. This would be difficult, since the trace cannot know its antecedent until a message from the trace and the message from the antecedent meet at a node which dominates both the trace and the antecedent. If the scrambled element is going to reconstruct, a message has to travel "down" the tree to the trace, which is not allowed in the current implementation of PRINCIPAR.[24]

Currently, neither PRINCIPAR nor the V-TAG formalism proposed to Korean in (Rambow and Lee, 1994) and (Park, 1994) handle Binding Theory, e.g., Condition C, or Scope Interpretation. Both systems produce the different possible word orders, for both short and long distance scrambling, but neither systems capture the interaction between scrambling and binding, or the interaction between scrambling and scope interpretation.

4. PARSING TIME ANALYSIS

Not surprisingly, scrambling does introduce additional computational complexity into parsing. A sample excerpt from our analysis is given below, where times are normalized to the unscrambled base time. It appears as though multiple scrambling beyond one clause results in the same nonlinear increases observed by Fong (Fong, 1991) for indexing.

	sentence	time, s.	ratio	comment
local	1(a)	1.32	1	(no scrambling)
multiple scrambling	1(b)	2.11	1.60	(one elem scrambled)
	1(c)	1.93	1.46	(one elem scrambled)
	1(d)	3.20	2.42	(two elem scrambled)
	1(e)	2.46	1.86	(one elem scrambled)
	1(f)	3.32	2.52	(two elem scrambled)
long distance	3(a)	6.61	1	(no scrambling)
multiple scrambling	3(b)	8.90	1.35	(one elem scrambled)
	3(c)	15.92	2.41	(two elem scrambled)

5. CONCLUSIONS

This paper describes how one of the most current linguistic analyses of Korean scrambling can be readily incorporated into an existing parser. The parser can correctly handle 26 out of all 29

[24](Lin, p.c.) proposes implementing a filter after the structure has been built, to handle reconstruction, therefore abandoning the structure description idea for this part of the parse.

sets of examples in chapters 2.2 (excluding 2.2.6 on parasitic gaps) and 3.2 of (Lee, 1994). The interaction between scrambling and other components of the grammar is easily accommodated, just as described by Lee. The approach outlined in this paper is compared with other approaches to scrambling, and surpasses them in coverage. The directness with which Lee's linguistic theory can be modeled demonstrates the value of using a "transparent" principles and parameters approach. We can simply use the theoretical assumptions that Lee makes and then test her results using the wide range of scrambling data she exhibits.

ACKNOWLEDGEMENTS

This report describes research done within the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences. This research is supported by NSF grant 9217041-ASC and ARPA under the HPCC program.

REFERENCES

Bonnie J. Dorr, Dekang Lin, Jye hoon Lee, and Sungki Suh. 1995. Efficient parsing for korean and english: A parameterized message passing approach. *Computational Linguistics.*

Sandiway Fong. 1991. *Computational Properties of Principle-Based Grammatical Theories.* Ph.D. thesis, MIT, Cambridge, MA 02139.

Sandiway Fong. 1994. Towards a proper linguistic and computational treatment of scrambling: An analysis of japanese. In *Proceedings of COLING-94.*

Robert Frank, Young-Suk Lee, and Owen Rambow. 1992. Scrambling as non-operator movement and the special status of subjects. In *Proceedings of the Third Leiden Conference for Junior Linguists.*

Young-Suk Lee. 1994. *Scrambling as Case-Driven Obligatory Movement.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA 19104-6228.

Hyun Seok Park. 1994. Korean grammar using tags. Master's thesis, University of Pennsylvania, Philadelphia, PA.

Owen Rambow and Young-Suk Lee. 1994. Word order variation and tree-adjoining grammar. *Computational Intelligence*, 10(4):386–400.

Owen Rambow. 1994. *Formal and Computational Aspects of Natural Language Syntax.* Ph.D. thesis, University of Pennsylvania, 3401 Walnut Street, Suite 400C, Philadelphia, PA 19104-6228.

Optimising Tools for the French Letter-to-Phone Grammar TOPH With a View to Phonographic Spelling Correction

Nada GHNEIM & Véronique AUBERGÉ
Institut de la Communication Parlée
INPG/Université Stendhal, URA CNRS n° 386
BP 25, 38040 Grenoble Cedex 9, FRANCE
Phone: (+33) 76 82 43 38 Fax: (+33) 76 82 43 35
e-mail: ghneim@icp.grenet.fr, auberge@icp.grenet.fr

Abstract
The goal ,in the long term, of this work is to give a formal and linguistic description of the text-to-phone processing, and to use the description in a view to phonographic spelling correction errors. This description can be formulated, by TOPH language, in a determinist grammar.

The French letter-to-phone exhaustive grammar *TOPH* is the base of our work in phonographic spelling correction: the inverse grammar *PHOT* gives the phone-to-letter correspondences. However, as it was constructed by human expert, *TOPH* has many redundant and incoherent rules which affect correction results. By eliminating these rules we got an optimal grammar to be used in the spelling correction.

We have developed an environment around a French letter-to-phone system, which enables to filter the redundancies and the incoherences in a lexical grammar written in *TOPH* language. The result is an optimal grammar in its logic and linguistic description. Such an environment had never been developed for a French letter-to-phone system. We use this exhaustive phonetic description to establish the duality between letter-to-phone processing and phone-to-letter one, with a view to phonographic spelling correction.

I. Introduction

The right use of spelling is a crucial problem in French language: on the one hand, the spelling presents real difficulties [Catach 89], and on the other hand, a good skill of spelling is an important social criterion. This is why automatic error correction of French text was the object of many studies [Pérennou 86, Laporte 89, Strube de Lima 90, Véronis 93].

In this work we are interested in the phonographic spelling correction errors, in which the writer substitutes a phonetically "close" but orthographically incorrect sequence of letters for the intented words.

A minimal French letter-to-phone grammar is the core of an actual grammar, extended following a systematic methodology in exploring a French representative dictionary: the ICP's[1] dictionary *"Le 60000"*, and in referring to the Petit Robert1[90][2] for the phonetic entries [Belrhali 92].

Rules order is an essential parameter in *TOPH* grammar description: expert expresses his knowledge naturally with the logic of exception followed by a general case, which means: "If a grapheme (*i.e.* a sequence of letters correspondent to an unit of pronunciation) is in a singular context, then it follows a singular transcription; Else, it follows the usual transcription"

[1]Institut de la Communication Parlée

[2]Commun French dictionary with 80.000 entries

When adding a new rule, the expert must take into account the meaning of this rule induced by the rules order, and by their syntagmatic concatenation on the word. As the grammar complexity increases, the control of the implicit logic becomes more difficult. Then it is necessary to treat non optimal grammars, and overall to detect prospective rules redundancies and incoherences, which do not affect, or not much, the text-to-phone global results: grammar determinism is ensured by rules writing order. On the other hand, word treatment time, and necessary memory size are uselessly increased, and the linguistic description becomes non optimal. Moreover, if we consider the PHOT phone-to-letter grammars, which is the result of the inversion of *TOPH* grammars, parasite phones introduction, by redundant or incoherent rules, makes invalid the linguistic formality of *PHOT* grammars.

In the next two paragraphs we introduce *TOPH* and *PHOT* systems, after that we present different tools to verify the coherence and the redundancy of language *TOPH* rules.

2. *TOPH* system

TOPH (Transcription from Orthograph to PHonetics) is a multilanguage text-to-phone system, which allows to transliterate a graphic string to a phonetic one.

A *TOPH* language grammar is constituted of two modules:

1. The first module contains the declaration of the sets, which allows to describe in the same syntax:
   - Phonetic class, like the set of nasal consonants: "Nasal Consonants" = (n, m, ...)
   - Lexicons of words which correspond to non regular etymological families, for example the lexicon of words in which the letter "s" is pronounced in final, which is not generally the case in French where a mute "s" is a plural mark: "Lexicon s final" = (bu, consensu,...), which corresponds to the set (bus, consensus,...)
2. In the second module transposition rules (partitioned into classes) are described. Rules class is determined by the first character in the transliterated string. Rule syntax has the following form:

(CtxtG) +Ch_graph+ (CtxtD) {Ens_Ctg} —> [Ch_phon]

where CtxtG, CtxtD are respectively left and right contexts which could be empty, Ch_graph is the string to transliterate, Ens_Ctg is the set of lexical categories with which this rule should be applied (could be empty in general when the rule does not treat a morpho-phonologic or morpho-syntactic ambiguity), Ch_phon is the correspondent phonetic string.

a. Organisation of a class: vertical order

Rules in the same class are not defined independently one of others; rules complete contexts are not explicitly given, but they are described by a local order relation, which is defined as "vertical", on the class of rules. This order is the one given sequentially by the expert, for example:

```
R1:    ("Vowel") +s+ ("Vowel") —> [z]
R2:              +s+           —> [s]
```

If R1 is a rule followed by R2 (R1, R2 means that the grapheme "s" in a vocalic context is voiced and pronounced [z] instead of [s] in the general case),then the explicit description of R2 is R3:

R3: (α) +s+ (β) —> [s]

where $\alpha, \beta = V_T$\("Vowel"), where V_T is the terminal vocabulary of the grammar.

The expert develops rules in each class with the logic "If R1 is applicable then apply R1; else if R2 is applicable then apply R2; else ...". So, in each class, the grammar is always determinist, even if the complexity of a class makes the administration of the vertical order (by the expert) very difficult.

b. Organisation of the grammar: horizontal order

We have seen that the grammar determinism is ensured, inside the same class, by rules vertical order. The determinism of the grammar all over the classes is ensured by another mechanism of order in rules application, which will be defined as an "horizontal order" on the grammar. To define this order, let the string "gu" be the input string,:

Extract of "gu" class

R4: +gu+ —> [g]

R5: +g+ —> [g]

If we explicit R5 according to intra-classes vertical order, R4 and R5 will be independent and stated as:

R4: +gu+ —> [g]

R5: +g+ (V_T\u) —> [g]

The two rules R4 and R6 describe the same transcription window "gu":

R6: (g) +u+ —> [u]

R4 transliterates all "gu", R6 transliterates "u" in "gu". So, there is an ambiguity. However, input order is imposed from left to right; which implies that R4 will be applied, and R6 will never be taken into account (R6 is inaccessible).

Then, input order (from left to right) implies, implicitly, an horizontal order of inter-classes rules application.

Horizontal order sets the expert the fundamental problem of the choice of graphemes: it is difficult to maintain the coherence of this choice between classes, because rules are developed class by class according to the vertical order.

3. *PHOT* System

PHOT (Transcription from PHonetics to Orthograph) is a description language for spelling grammars. *PHOT* grammar rules are obtained by the inversion of *TOPH* grammar, grouped in phonetic classes. A *PHOT* grammar is composed of two modules:

1. The first one contains the declaration of the sets (the same of the initial grammar *TOPH*)

2. The second contains transcription rules which are partitioned in classes of phones. A rule syntax has the following form:

(Ch_graph) —> (CtxtG) +[Ch_phon]+ (CtxtD) {Ens_Ctg}

Example:

R7:	[s] —>	("#"+tourne)	+s+	(ol+"#")
R8:	[z] —>	("Vowel")	+s+	("Vowel")

Then, "s" in a vocalic context, in the word "tournesol" (turnsole) [tuʀnəsɔl], is not pronounced [z] but [s] to give the oral information of the morphologic structure, which is the agglutination of the two lexemes "tourne" (to turn) and "sol" (sun).

This grammar, which describes correspondences between a sequence of phones and the set of contextual strings which could produce it, will be the base of our model for the phonographic spelling correction.

4. Preliminary definitions

- A rule is called *redundant* if there are other rules in the grammar, which treat all (or a part) of the same graphic string (with correspondent contexts), and generate the same phonetic transcription.
- A rule is called *incoherent* if there are other preceding rules in the grammar, which treat all (or a part) of the same graphic string (with correspondent contexts), and generate the same phonetic transcription.
- *The local (or global) study of the redundancy* of a given rule consist in looking for the first (or all) redundant rule(s) in the grammar.
- *The local (or global) study of the incoherence* of a given rule consist in looking for the first (or all) incoherent rule(s) in the grammar.

5. Utility of the research and treatment of redundancy and incoherence

The treatment of redundant and incoherent rules is very important to optimise memory and computation time, and also from a rules linguistic description validity point of view :

5.1. Redundancy and incoherence effect on phonetic transcription execution time

The necessary time to transliterate a word is equal to $\sum_{gr \in word} T(gr)$, where $T(gr)$ is the necessary time to find the applicable rule on the graphic string gr in the correspondent class.

The average time necessary to find the rule is: $T_{average}(gr) = \sum_{i \in [1..n]} T_i(gr) \times \Pr op_i$

where n is the number of rules in the class correspondent to the string "gr", $T_i(gr)$ is the necessary time to reach the ith rule (equal to i operations) and $\Pr op_i$ is the probability of using the ith rule.

In *TOPH*, the last rule of a class is always the general one (supposed to be the most frequently used). So, in this case:

$$T_{average}(gr) = \sum_{i\in[1..n]} i * \Pr op_i \geq \frac{n+1}{2} operations$$

Therefore, the time necessary to transliterate a word depends on the number of grammar and on the position of each rule inside the class.

5.2. Redundancy and incoherence effect on the size of memory

The necessary memory to store rules is:

$$Memory_{total} = nb_{rules} * Memory_{rule}$$

where nb_{rules} is the number of rules in the grammar, and $Memory_{rule}$ is the constant memory size necessary to store one rule.

5.3. Redundancy and incoherence effect in determining the set of phones in *PHOT*

We have seen that it is difficult, for the expert, to determine, in a single way, the graphic string which he must choose as string to transliterate in *TOPH* rules. While *TOPH* grammar determinism is ensured by the horizontal and vertical order, this choice could modify the set of phones, which implies considerable modifications in *PHOT*.

Example

Considering the following rules in *TOPH* grammar:

R9 :	("#"+qu) +ia+ ("#") —> [ija]	("quia" —>[kija] (*quia*))
R10 :	("#"+qu) +i+ (é,a) —> [ij]	("quiétude" —> [kijetyd] (*quietude*))
R11 :	+a+ —> [a]	

When inversing this grammar, we will have the strings "ija", "ij" and "a" (correspondent to the graphemic strings "ia", "i" and "a") in the phonetic input group. Then, in eliminating R9, which is redundant comparing to R10 and R11 in *TOPH*, the phonetic string "ija" will be, consequently, eliminated from the set of phones concerning *PHOT*.

6. Local study of the redundancy and the incoherence

The local check of redundancy and incoherence is applied on a new inserted rule, in an optimal grammar, in which there is neither redundancy nor incoherence. In this case the research is stopped when finding the first rule which can be applied on the string to transliterate of the new rule.

6. 1. general method

To test redundancies and incoherences of rules in the grammar, we proceed as follows:

- Remove the studied rule from the grammar to get the new grammar.
- Transliterate the grapheme of the studied rule, considering right and left contexts, using the new grammar.
- If there was no applicable rule in the new grammar then the studied rule is not redundant neither incoherent.
- Else, compare the result of the transcription (the phone produced by the applicable rule) with the phone produced by the studied rule:
  – if they were equal then the studied rule is redundant to the applicable one.
  – else the studied rule is incoherent to the applicable one.

6.2. Computational time

If we explicit the jth rule under the form:

$$\left(Set_{0j} + Set_{1j} + \ldots + Set_{ij} + \ldots\right) + Ch\_graph + \left(\ldots + Set_{ij} + \ldots + Set_{NbConcat_j j}\right)\{Ens\_Ctg\} \to [Ch\_phon]$$

where Set_{ij} is the set of strings limited by two plus signs, in the jth rule. Then, total execution time is of the order: $\sum_{j=1}^{NbRules} \prod_{i=1}^{NbConcat_j} \left|Set_{ij}\right|$, where $\left|Set_{ij}\right|$ is the cardinal of the set i, in the jth rule.

6.3. Memory size

Memory size is proportional to the number of rules in the grammar:

$$Memory_{total} = nb_{rules} * Memory_{rule}$$

6.4. Examples of local redundancy

Rules are redundant if :

- They are identical in their expression.
- They can produce the same string

R12 : ("#"+"EXCEP:OO") +oo+ —> [u] (*e.g.* "booling" —> [buliŋ] (*booling*))

R13 : ("#"+igl) +oo+ —> [u]

where the set "EXCEP:OO" contains the element "igl".

- One string produced by a rule is a sub-string of a string which is produced by another one. For example (the following rules treat etymologically Greek words):

R14 : (spiro, sti, sto, stœ, syn, sporotri) +ch+ (a,è,é,i,o) —> [k]

R15 : (tri) +ch+ (i) —> [k]

where "trichi" $\subset$ "sporotrichi", and "trichi" is a string produced by R15 and "sporotrichi" is an element of the set of strings produced by R14.

• One rule can be replaced by a set of other general rules, for example with the following rules we can have the same phonetic transcription obtained by the rule R16, in applying successively R17 and R18:

R16 : +iu+ (m+"#") —> [jo] (Latin final "-ium", *e.g.* "atrium" —> [atrijɔm])

R17 : +i+ ("Vowel") —> [j] (general rule, like in "sioux" —> [sju] (*Sioux*))

R18 : +u+ (m+"#") —> [o] (Latin final "-um", *e.g.* "quantum" —> [kwɑ̃tɔm])

6.5. Examples of local incoherence

Incoherences occur when:

• The expert puts a rule treating a particular case after the general rule, so the system stops always before reaching the second, for example :

R19 : ("#"+st) +ea+ (k+"#") —> [ɛ]

R20 : ("#"+st,str,sw) +ea+ —> [i] (like in "steamer" —> [stimœʀ] (*steamer*))

• The expert treats the same case using graphemes with different length, for example:

R21 : ("#"+séqu) +o+ (ia) —> [ɔ] ("séquoia" —> [sekɔja] (*sequoia*))

R22 : ("#"+séqu) +oia+ ("#") —> [ɔja]

• The expert treats the same case in two different ways :

R23 : ("#"+m) +oe+ (re) —> [] ("moere" —> [mwɛʀ] (*polder*))

R24 : (m) +oe+ (re) —> [wɛ]

7. Global study of the redundancy and the incoherence

In this case, the rules of the grammar are written by the expert without local treatment of the redundancy or the incoherence. To optimise this grammar, the local research method explained before is incomplete: the research is stopped at the first applicable rule, and this will possibly hides other redundant or incoherent rules.

A global research method allows to identify all redundant and incoherent rules in the grammar, in extending the local research algorithm to find all the applicable rules on the string to transliterate.

7.1. Examples of global redundancy

• Redundancy global research of the studied rule detects one (or a set) of redundant rule(s), for example the rule:

R25 : ("#"+tr) +ou+ (ée+"#") —>[u] ("trouée" —> [tʀue] (*breach*))

has the following set of redundant rules:

R26 : (t,f,c,b,p+"Liquid Cons.") +ou+ —> [u]

R27 : ("#"+ encr,tr) +ou+ (é, er) —> [u]

R28 : ("Cons."+"Liquid Cons.") +ou+ ("Vowel") —> [u]

In this case, we could have rules which treat the same case (with contexts of different length), and then we could unify these rules, if possible, in one general rule.

• Detected global redundant rules are parasite: this case is the result of the exhaustive research of applicable rules, and it happens when a succession of letters implies the application of a general rule concerning this succession, but not in words in which another (more general) rule must be applied. For example, the general rule R29 to transliterate the grapheme "a" in the succession ay"Vowel" is:

R29 : +a+ (y+"Vowel") —> [ɛ] (like in "payer" —> [pɛje] (*to pay*))

On the other hand, there are certain words like "pagaye" (disorder) which do not follow this rule but the general rule R30 of the class " a" :

R30 : +a+ —> [a]

but, because rules in each class are treated by vertical order, the expert must put the rule R31 which treats this word before R29:

R31 : ("#"+pag) +a+ (ye) —> [a] ("pagaye" —> [pagaj] (*disorder*))

This is why, the application of the research global of R31 redundancy will give R30 as a redundant rule (parasite redundancy). This result allows to find exception rules treating exception words, to put them in a set of exceptions "EXCEP:AY", and to replace R29 by R32:

R32: +a+ (y+"Vowel") \ ("EXCEP: AY") —> [ɛ]

where '\' means the "except" operation.

7.2. Examples of global incoherence

• Global research detect for the studied rule a set of incoherent rules, for example the rule:

R33 : +o+ (in+"Non Nas.Cons.") —> [w] (like in "lointaine" —> [lwɛ̃tɛn] (*distant*))

has the following set of global incoherent rules:

R34 : +oin+ ("#","Cons.") —> [wɛ̃]

R35 : +oi+ —> [wa]

• Detected global incoherent rules are parasite: this happens when a graphic string follows a certain rule, except in words (which must be given in rules situated before this rule, "vertical order" of applying the rules). Then the global research will detect the exception rules as incoherent to the studied one. For example, the general rule applied on the grapheme "u" followed by a vowel is R36:

R36 : +u+ ("Vowel") —> [ɥ] (like in "quidam" —> [kɥidam] (*person*))

but there are exceptions where this rule is not applicable, as in following rules:

R37 : +u+ (e+"#") —> [y] (like in "rue" —>[ʀy] (*street*))

R38 : +u+ (y) —> [ɥi] (like in "tuyauter" —> [tɥijote] (*to quill*))

which must be met before R36, and so detected as incoherent rule.

8. Conclusion

Redundant and incoherent rules, which are detected by the algorithm are not automatically filtered: in treating a *TOPH* French grammar of 1200 rules, we have established the fact that they reveal more much deep linguistic incoherence problems (usually in relation with the horizontal order, from which maximum graphemes are defined).

The list of rules is proposed to the expert, who has the choice to handle the grammar.

After obtaining the new optimal grammar *TOPH*, we inversed it to produce the correspondent grammar *PHOT*. In the *PHOT* system, the graphemic strings are generated under the strong constraint of right and left phonetic contexts (calculated like the phonetic correspondences of the orthographic contexts in the *TOPH* grammar) surrounding the processed phonetic string.

This is not the case of all the phone-to-letter systems developed for French spelling errors environments, which use correspondences between graphemes and phonetic strings without any contextual constraints. This is why the number of orthographic solutions produced for one phonetic string in such systems is very numerous in comparison with that produced using *PHOT*.

7. References

[Aubergé 91] V. Aubergé, "La synthèse de la parole: des règles aux lexiques", Thèse de l'université Pierre Mendès France, Grenoble2, 1991.

[Belrhali 92] R. Belrhali, L. Libert, V. Aubergé, L.J. Boë, "Élaboration des lexiques d'une grammaire de phonétisation du français", 19es JEP-SFA, Bruxelles, 1992.

[Catach 89] N. Catach, "Les délires de l'orthographe", Plon, 1989.

[Laporte 89] E. Laporte, M. Silberztein, "Vérification et correction orthographique assistées par ordinateur", Actes de la 1ère conférence européenne sur les techniques et les applications de l'Intelligence Artificielle en milieu industriel et de service, Hermès, 1989.

[Pérennou 86] G. Pérennou, P. Daubeze et F. Lahens, "Vérification et correction automatique de textes, prise en compte de fautes orthographiques et typographiques, Un modèle VORTEX", TSI, vol. 5, n°4, juillet-août, 1986.

[Strube de Lima 90] V. L. Strube de Lima, "Contribution à l'étude du traitement des erreurs au niveau lexico-syntaxique dans un texte écrit en français", Thèse de l'Université Joseph Fourier, Grenoble I, 1990.

[Véronis 93] J. Véronis, "Distance entre chaînes : extension aux erreurs phono-graphiques", Travaux de l'Institut de Phonétique d'Aix, vol. 15, pp.217-234, 1993.

ERROR CORRECTION OF SPEECH RECOGNITION OUTPUTS USING GENERALIZED LR PARSING AND CONFUSION MATRIX

Tatsuya Iwasa and Kenji Kita
Faculty of Engineering
Tokushima University
Minami-josanjima, Tokushima 770, JAPAN

e-mail. {iwasa and kita}@is.tokushima-u.ac.jp

Abstract

In this paper, we describe a method for correcting misrecognition results derived from a speech recognizer. Our method is based on generalized LR parsing and uses a confusion matrix in order to select the best sentence out of multiple candidates. The proposed method was applied to the actual speech recognition system developed in our laboratory.

1 Introduction

Recently, remarkable progress has been made in acoustic modeling research, with hidden Markov models (HMMs) and a neural network-based approach. However, speech recognition based on acoustic information alone does not result in good performance for large vocabulary tasks. Successful speech recognition/understanding requires the use of linguistic information. There are two approaches using linguistic information:

(1) To directly use linguistic information at recognition time.

(2) To correct recognition outputs using linguistic information.

This paper is concerned with the second approach. In this paper, we describe a method for correcting misrecognized utterances using lexical and syntactic information. More precisely, we describe an error-correcting generalized LR parser using candidate scoring based on a confusion matrix. The use of the confusion matrix provides a good cost calculation scheme and contributes to select the most likely sentence out of multiple candidates. Saito et al. [1] already proposed an error-correcting method using generalized LR parsing and a confusion matrix. Their method, however, assumes that consecutive errors do not occur in the misrecognized utterances. Our method is based on error-correcting LR parsing introduced in [2], which can handle consecutive errors, and is enhanced with scoring by the confusion matrix.

2 Speech Recognition System: An Overview

This section describes a speech recognition system developed in our laboratory.

2.1 Acoustic Models

As acoustic models, we adopt *hidden Markov models* (HMMs for short) [3], which have been successfully used in recent state-of-the-art speech recognition systems. HMMs are stochastic models suitable for handling the uncertainty that arises in speech, such as contextual effects and speaker variabilities.

In our speech recognition system, Japanese syllables are used as the basic HMM unit because the whole word-based approch is difficult to meet the real-time requirements in case of the large vocabulary size. There are about 100 phonetically different spoken syllables in all. Each syllable is represented by a continuous mixture HMM, in which an output probability density function is characterized by a 39-component diagonal covariance Gaussian mixture. See Table 1 for speech analysis conditions.

2.2 Recognition Method

As stated above, our system uses hidden Markov models of syllables as the basis for speech modeling. Word models are built by concatenating syllable models. The speech recognition module performs a time-synchronous Viterbi beam search, matching syllable models against the speech input. That is, it maintains a beam of the best scoring candidates and extends these one frame at a time. Recognition candidates with a low likelihood score are pruned.

Table 1: Speech analysis conditions

Sampling frequency and precision	16 kHz, 16 bit
Pre-emphasis	$1 - 0.97z^{-1}$
Hamming window	25 ms
Frame period	10 ms
Acoustic parameters	12 MFCC (mel-frequency cepstral coefficients) + 12 Δ MFCC + 12 $\Delta\,\Delta$ MFCC + power + Δ power + $\Delta\,\Delta$ power (39 dimensions in all)

All candidates cover the utterance from the beginning to the most recently processed frame. As a recognition path reaches the end of a syllable model, the search transits to the beginning of syllable models that can follow the current syllable which ends the path. Currently, our system uses no restrictions concerning syllable connections (i.e. any syllable can follow any other syllables). In the speaker-dependent condition, the syllable recognition rate is around 90%.

2.3 Examples of Recognition Outputs

Since our speech recognition system does not use any syntactic or semantic knowledge, it sometimes produces a syllable sequence which include misrecognized syllables. Table 2 shows some examples of recognition outputs.

Table 2: Speech recognition examples

	bf Real sequence	bf Recogniged sequence
1	ha chi ga tsu yo Q ka no yo ru ka ra de su	ha chi ga tsu yo Q ka no yo [ro] ka ra de su
2	i tsu ka ra o to ma ri ni na ri ma su ka	i tsu ka ra [ ] to ma ri ni na ri ma su ka
3	wa ka ri ma shi ta	wa ka ri [i] ma shi ta

There are three types of errors as follows:

1. Substitution error

   A syllable is incorrectly recognized as another syllable. For example, the tenth syllable /ru/ in example (1) is recognized as /ro/.

2. Deletion error

   A syllable which is actually spoken is not recognized. For example, the fifth syllable /o/ in example (2) is a deleted syllable.

3. Insertion error

   A syllable which is not actually spoken is recognized. For example, the fourth syllable /i/ in example (3) is an inserted syllable.

3 Error-Correcting Method: An Example

S	→	NP	V	
S	→	V		
NP	→	N		
NP	→	N	V	
N	→	ko	re	
P	→	o		
V	→	ku	re	
V	→	o	ku	re

Figure 1: Example of Grammar.

	action					goto				
	o	ko	ku	re	$	S	N	V	P	NP
0	sh 5	sh 1	sh 4			7	2	6		3
1				re 8						
2	sh 9, re 3		re 3						10	
3	sh 5		sh 4					11		
4				sh 12						
5			sh 13							
6					re 2					
7					acc					
8	re 5		re 5							
9	re 6		re 6							
10	re 4		re 4							
11					re 1					
12					re 7					
13				sh 14						
14					re 8					

Figure 2: LR parsing table for Figure 1

In this section, we present a sample trace of the error-correcting parser. An error-correcting method is essentially based on [2]. Nodes indicated by ●, ◎ and ○ are state vertexes. State vertex ◎ shows a Vca(current active vertex). Parsing actions are performed against all the current vertexes. State vertex ○ shows a Vna(next active vertex), that is, it will become an active vertex in the next step. Our error correcting parser has four kinds of parses. The first one is a deletion-correcting parse, whose results are included in Vca. The second one is a matching-parse, whose results are included in Vna. The third one is a substitution-correcting parse, whose results are included in Vna. The fourth one is an insertion-correcting parse, whose results are included in Vna.

In our example below, syllable sequence "o re ku re" will be corrected using the grammar in Figure 1 and the LR parsing table in Figure 2. For simplicity, the following assumptions are made:

(1) Syllable /a/ may be substituted by /o/.

(2) Syllable /o/ may be deleted.

(3) Syllable /re/ may be inserted.

(4) Two cosecutive syllables cannot be deleted.

Initially, state 0 is created.

Parsing the first syllable /o/

The assumption (2) creates a deletion-correcting parse in Figure 3. The parser looks for deleted syllable candidates before the first syllable /o/ in "o re ku re". By referring to the LR table, at state 0, /o/ expects the action "shift 5". The parser creates a new grammar vertex labelled by "o" containing "1, del". Here, "1, del" means that we correct the first syllable as a deleted syllable, and creates a new state vertex 5 which is included in Vca.

For the matching-parse in Figure 4, the parser considers all the state vertexes which is included in Vca. At state vertex 0, by referring to the LR table, "shift 5" is indicated. The parser creates a new grammar vertex labelled by "o" and a new state vertex 5 which is included in Vna. At state vertex 5, no action is attached to syllable /o/.

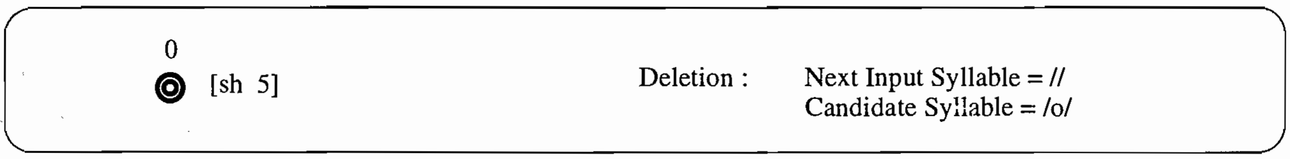


Figure 3: Trace of Error Correcting (1)

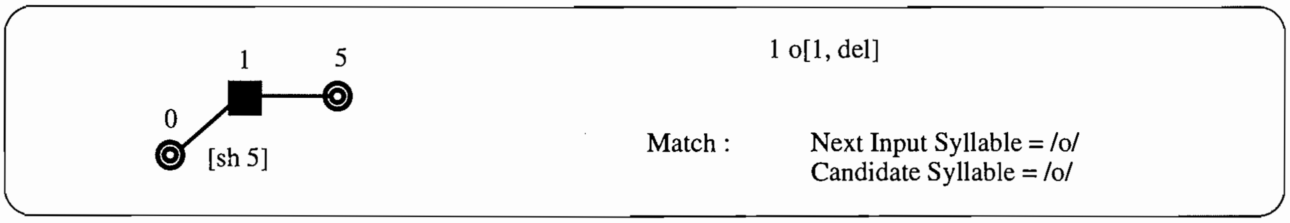


Figure 4: Trace of Error Correcting (2)

The assumption (1) creates the substitution-correcting parse in Figure 5, At state vertex 0, by referring to the LR table, "shift 1" is indicated. The parser creates a new grammar vertex labelled by "a" and a new state vertex 1 which is included in Vna. At sate vertex 5, no action is attached to syllable /a/.

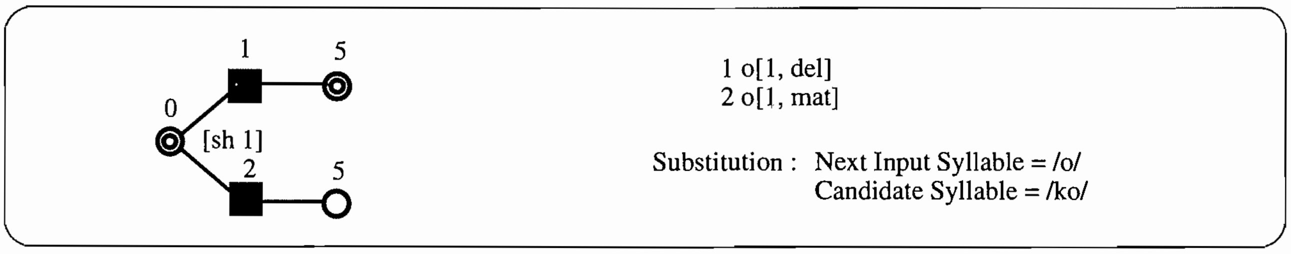


Figure 5: Trace of Error Correcting (3)

Our assumption tells that /o/ is not inserted.

Parsing the second syllable /re/

The matching-parse in Figure 6 has a current active vertex 1, which expects "shift 8" by syllable /re/. The parser creates a new grammar vertex labelled by "re" and a new state vertex which is included in Vna.

The assumption (3) tells that /re/ is possibly inserted. Thus, we obtain the insertion-correcting parse in Figure 7. The parser creates a new grammar vertex from each Vca and a new state vertex which is included in Vna.

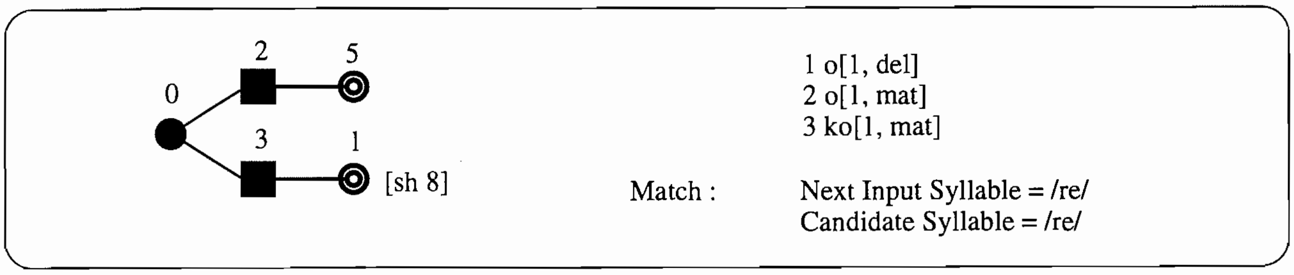


Figure 6: Trace of Error Correcting (4)

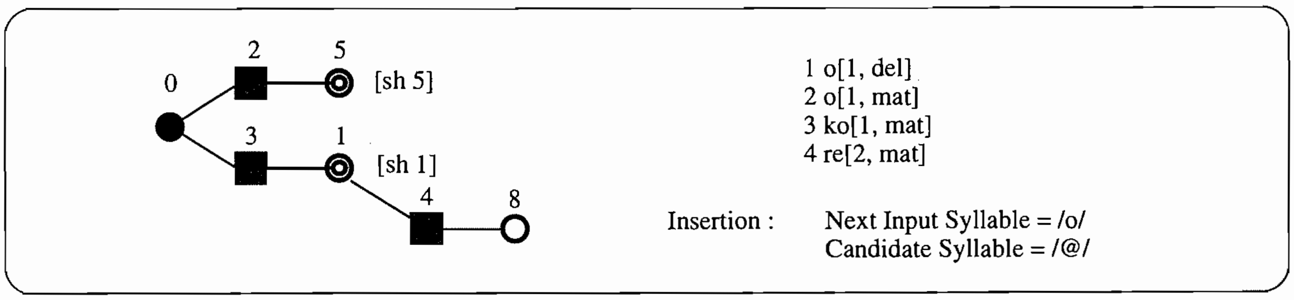


Figure 7: Trace of Error Correcting (5)

Parsing the third syllable /o/

The assumption (2) tells that /o/ is possibly deleted. Thus, we obtain the deletion-correcting parse in Figure ??. By referring to the LR table, state 8 has action "reduce 5". A reduce action will be performed in the same manner as in generalized LR parsing. The parser creates a new grammar vertex labelled by "N" containg "3 4". The parser creates a new state vertex 2.

Parsing the end-symbol /$/

State vertex 14 expects "reduce 8" in Figure 8. In reduce action, if the path includes the insertion-symbol("@"), our algorithm executes one more pop.

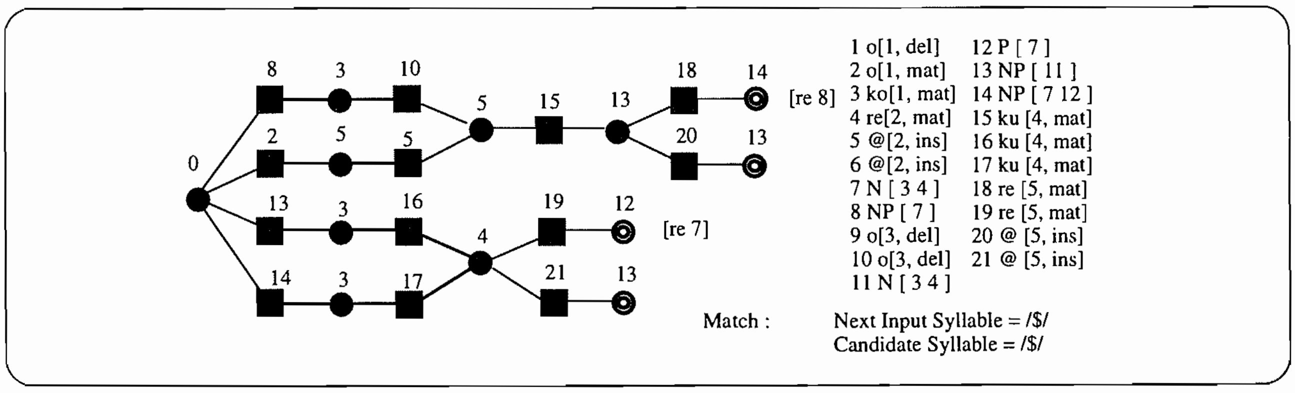
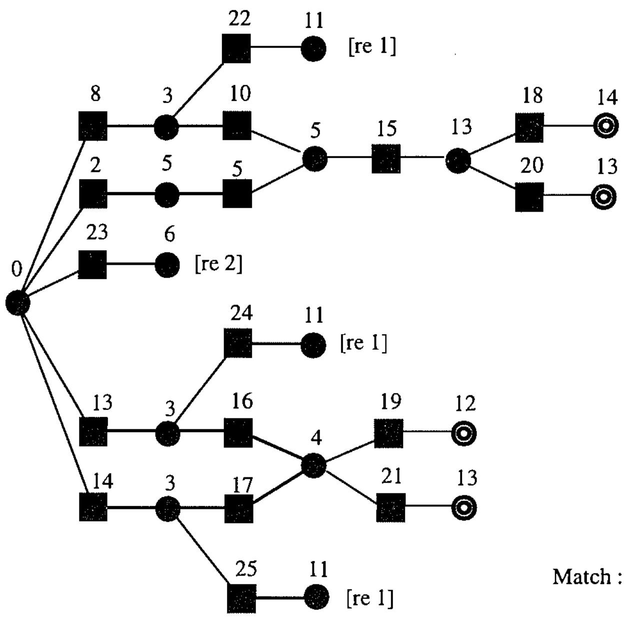


Figure 8: Trace of Error Correcting (6)

After this action, the stack is as in Figure 9.



1 o[1, del] 22 V [10 15 18]
2 o[1, mat] 23 V [2 5 15 18]
3 ko[1, mat] 24 V [16 19]
4 re[2, mat] 25 V [17 19]
5 @[2, ins]
6 @[2, ins]
7 N [3 4]
8 NP [7]
9 o[3, del]
10 o[3, del]
11 N [3 4]
12 P [7]
13 NP [11]
14 NP [7 12]
15 ku [4, mat]
16 ku [4, mat]
17 ku [4, mat]
18 re [5, mat]
19 re [5, mat]
20 @ [5, ins]
21 @ [5, ins]

Match : Next Input Syllable = /$/
Candidate Syllable = /$/

Figure 9: Trace of Error Correcting (7)

4 Confusion Matrix

A confusion matrix shows the probability of recognition (syllable by syllable). Figure 10 shows an example of a confision matrix. The first line shows that if the syllable /a/ is spoken, then it is recognized correctly 69.5% of the time; misrecognizes it as /i/ 2.4% of the time, as /ka/ 2.4% of the time, and so on. The last column shows the deletion probability. Therefore the probability that syllable /a/ was a deleted syllable is 8.5%. The last line shows insertion probability. Therefore the probability that syllable /a/ was a inserted syllable is 3.8%.

4.1 Cost Calculation Using Confusion Matrix

We calcurate the cost for the incorrect syllables as following,

1. Deleted syllables

   According to a confision matrix, the syllable /o/ is deleted more frequently than any other syllables. Thus, we parse the syllable /o/ with low cost than any other syllables as deleted syllable.

	a	i	u	e	o	ka	...	@
a	69.5	2.4	0.0	0.0	0.0	2.4		8.5
i	0.2	86.6	0.2	0.7	0.2	0.0		4.5
u	0.0	0.9	82.1	0.9	0.0	0.0		7.1
e	0.0	1.9	0.0	84.0	0.0	0.0		2.8
o	0.0	0.0	0.0	0.0	69.3	0.0		11.6
ka	0.0	0.0	0.0	0.0	0.0	97.7		0.3
⋮								⋮
@	3.8	26.5	3.8	1.7	3.8	0.4	...	

Figure 10: An Example of Confusion Matrix

2. Substituted syllables

   According to the confusion matrix, not all syllables can be substituted to any other syllable. If the syllable /a/ is generated from the device, there is a slight chance that original input was /i/ or /ka/,respectively, but no chance that the original input was /u/,/e/ or /o/. Thus,we pase the substituted syllable /i/ and /ka/ with lower cost than any other syllables.

3. Inserted syllables

   According to the confision matrix, the probability of the syllable /i/ being extra is higer than the probability of any other syllables. Thus, we paese the syllable /i/ with lower cost than any other syllables as inserted syllable.

4.2 An Error-Correcting Example

We include an error-correcting example in Table 3. The correct sentence is "ka i gi ni sa N ka shi ta i no de su ga", which means "I would like to register for the conference" in English. This sentence was misrecognized as "i Q ka i gi ni sa N ka shi ta i no de su ga", but successfully corrected.

5 Conclusion

This paper described a method for correcting misrecognition results derived from a speech recognizer. Our method is based on generalized LR parsing and uses a confusion matrix in order to select the best sentence out of multiple candidates.

Table 3: An Error-Correcting Example

	sentence	cost
correct	ka i gi ni sa N ka shi ta i no de su ga	
misrecognized	i Q ka i gi ni sa N ka shi ta i no de su ga	
1	@ @ ka i gi ni sa N ka shi ta i no de su ga	0.24
2	@ @ ka i gi ni sa N ka o shi ta i no de su ga	0.29
3	o o i @ ka i gi ni sa N ka shi ta i no de su ga	0.33
4	@ @ ka i gi ni sa N ka o shi Q ta @ no de su ga	0.36
5	o o i @ ka i gi ni sa N ka o shi ta i no de su ga	0.38
6	@ @ ka i gi ni o o sa @ ka o shi ta i no de su ga	0.40
7	@ @ ka i gi ni sa N ka o shi ta o o o i no de su ga	0.42
8	o o i @ ka i gi ni sa N ka o shi Q ta @ no de su ga	0.45
9	@ @ ka i gi ni o o sa @ ka o shi Q ta @ no de su ga	0.47
10	@ @ ka i gi ni sa N ka o a shi ta o o i no de su ga	0.49

References

[1] H. Saito and M. Tomita: "GLR Parsing for Noisy Input", *Generalized LR Parsing*, Kluwer Academic Publishers (1991).

[2] T. Nagao and E. Tanaka: "An Error-correcting Algorithm for Context-free Languages Based on a Generalized LR Parser", IEICE Technical Report, COMP94-66 (1994).

[3] X. D. Huang, Y. Ariki and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University Press (1990).

[4] A. V. Aho, R. Sethi and J. D. Ullman: "Compilers, Principles, Techniques, and Tools", Addison-Wesley (1986).

[5] M. Tomita (Ed.): *Generalized LR Parsing*, Kluwer Academic Publishers (1991).

THE PHONETIC VARIANTS OF TAIWANESE "VOICED" STOPS: AN AIRFLOW STUDY

Ho-hsien Pan
pan@ling.ohio-state.edu
Department of Linguistics
The Ohio State University, Columbus, OH, 43220

Abstract: As in other languages with "voiced" stops, the phonetic realization of Taiwanese /b/ and /g/ is highly variable from one context to another. Zhang (1983) even claims that they are replaced by homorganic nasals [m] and [ŋ] when the syllable is closed by a nasal. The positional variants were examined using oral and nasal airflow recordings. Three types of syllable structures were chosen with initial voiced stops in CV, CVN, and CVC. The target syllables are placed in utterance initial position and in medial positions after nasals, vowels, and voiceless stops. Utterance initially, there is a voiceless prenasalization, although its occurrence is strongly speaker-dependent. When preceded by nasals, the prenasalized voiced stops alternate with post-stopped nasals. When preceded by vowels, the prenasalization was found only in nine out of 630 tokens.

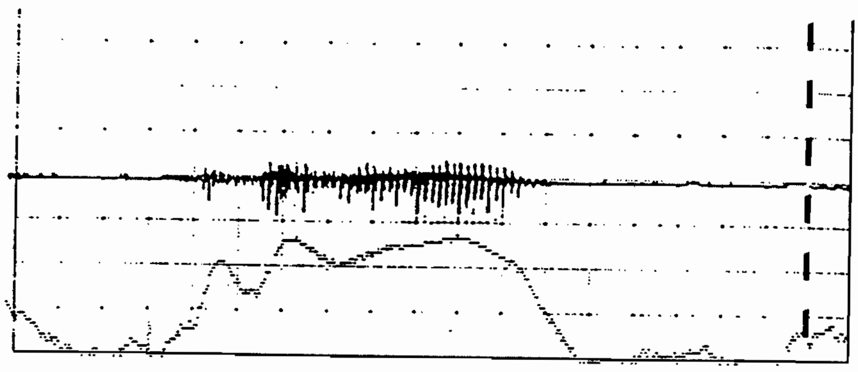
Introduction

As in other languages with "voiced" stops, (e.g. Spanish where the voiced stops show up as spirants) the phonetic realization of Taiwanese /b/ and /g/ is highly variable from one context to another. For example, in utterance initial positions, they are said to be "prenasalized" as well as prevoiced, and the prenasalization is voiceless (Chan, 1987; Zhang, 1983). That is /b/ here is $[^{m}b]$ (or $[^{\mathring{m}}b]$) and /g/ is $[^{ŋ}g]$ (or $[^{ŋ}g]$). /l/ will be included here, because /l/ fill the systematic position where one would expect and /d/. However, the prenasalization is not easily observed in the acoustic data collected here (e.g. Figure 1), nor could it be noted in the fiber-optical study done by Iwata et al.(1979), since they looked at glottal width not velo-pharyngeal opening width. In order to analyze the prenasalization, an air flow study is necessary. In an airflow study any air flowing out of nose and mouth are recorded independently, regardless of whether the air causes vocal folds to vibrate or not. Therefore, an air flow study is most efficient in picking up voiceless nasal articulation with low amplitude, such as the prenasal segments here.

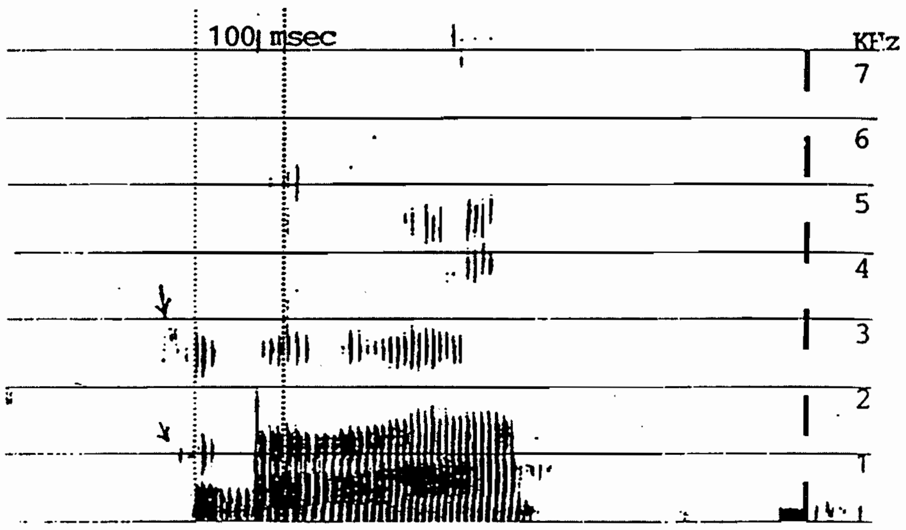
The airflow data are used to address two issues: first , the precise distribution of initial (prenasalized) voiced stops and initial nasals with regards to following segmental environments. The relationship between Taiwanese initial voiced stops and initial nasals is a controversial one. Historically, Taiwanese initial voiced stops came from initial nasals. Many studies influenced by the diachronic relationship between the two series, proposed an allophonic relationship between the two. According to Zhang, Taiwanese voiced stops change into homorganic nasals when the syllables are closed by nasals.

/b, l, g/ -----> [m, n, ŋ] / ________ ṽ, VN.

However a counterexamples such as /gɔŋ˧˥/ [gɔŋ˧˥] 'dizzy' was found by Pan (1994), as shown in Figure 2. This counterexample raise doubts to the rule. What challenges the validity of the rule even more is that the rule does not mention anything about /bŋ ˧˥/ [məŋ˧˥] 'door' (Figure 3), If Taiwanese initial voiced stops and nasals are indeed in an allophonic relationship, then CN syllable structure should definitely be mentioned in Zhang's rule. Therefore air flow data is brought in to test the phonological rule Zhang proposed.

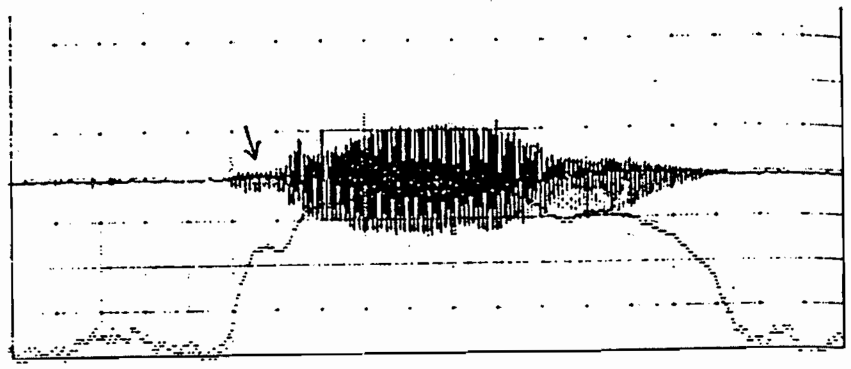


(a) waveform

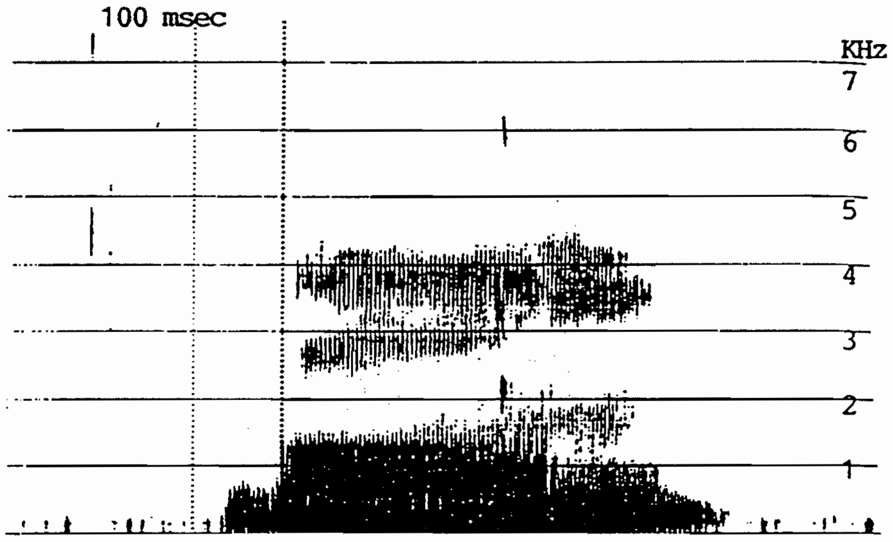


(b) Spectrogram

Figure 1 (a) waveform, (b) spectrogram, Taiwanese /bo˧/ [^{m}bo˧] 'hat'

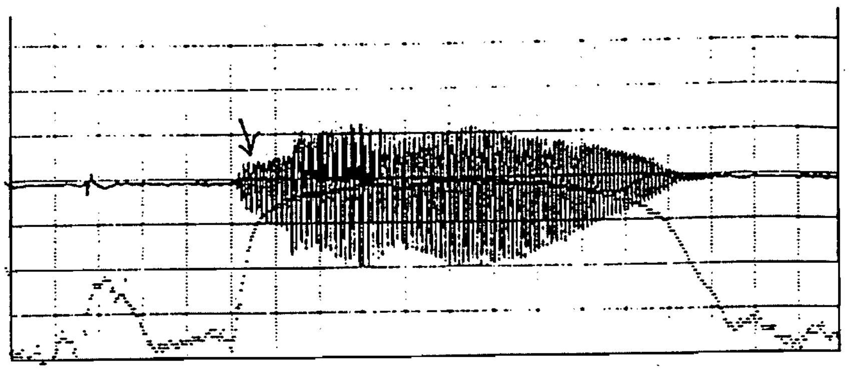


(a) waveforms

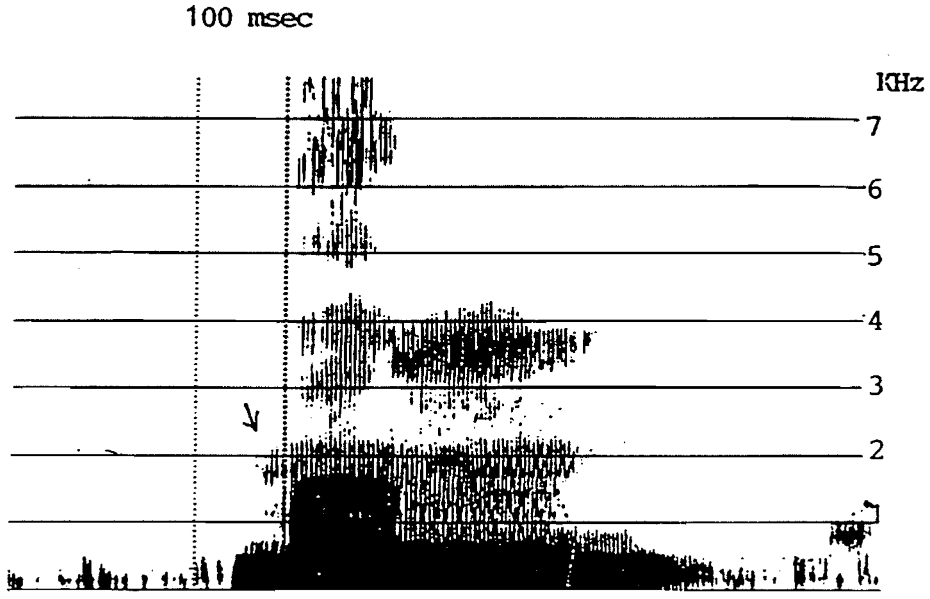


(b.) Spectrogram

Figure 2 (a) waveform, (b) spectrogram /gɔŋ˧˩/ [gɔŋ˧˩] 'dizzy'



(a) Waveform



(b.) Spectrogram

Figure 3 (a) waveform, (b) spectrogram /bəŋ˧˥/ [məŋ˧˥] 'door'

The second issue addressed is to find out the actual phonetic variants of Taiwanese voiced stops in utterance initial position and other positions not usually elicited in field studies such as Zhang's (1983).

Method of Air Flow Study

Instrumentation

An oral and nasal dual channel Rothenberg air flow mask was used to collect airflow. The airflow signals were recorded by digitizing directly to the hard drive of a 80486 PC. The signals were digitized at 10 kHz using Cspeech version 4.0. There is a 10 ms delay in the nasal air flow due to the 30 Hz filter used. The response time and delay of this filter are both about 10 ms.

Subjects

Seven male native Taiwanese speakers from the graduate school of The Ohio State University participated voluntarily in the experiment. They are all tri-lingual speakers of Taiwanese, Mandarin, and English. CF, YF, and BH all grew up in Tainan (Southern Taiwan). HJ is the only speaker from Taichung (Central Taiwan). LK is from Pingtong (Southern Taiwan). WS and CC are from Taipei (North Taiwan). The dialectal differences of Taiwanese spoken in South, Central, and North Taiwan are not relevant to the study here.

Corpora

Morphemes with target initial consonants were recorded in three separate contexts --- citation form, phrase initial position, and phrase medial positions following vowels, voiceless consonants, and nasals.

Table 1 shows the fifteen Taiwanese morphemes with three different syllable structures, CV, CVN, and CV+[voiceless stops], in the citation forms. Taiwanese initial /l/ was also included, since it shows the same distributional pattern relative to

Table 1 The fifteen citation form morphemes in the three different syllable structures.

CV:		CVN		CV+[voiceless stops]	
labial					
/ba˨˦ /	'numb'	/baĩ˥˧/	'mosquito'	/baʔ˨˩/	'meat'
		/ban˧/	'slow'	/bak˥˧/	'eye'
dental					
/la˧/	'stir'	/laĩ˨˦/	'human'	/lak˥˧/	'six'
		/lan˥˧/	'we, us'	/lap˥˧/	'pay'
velar					
/gia˨˦/	'centipede'	/giam˨˦/	'strict'	/get˥˧/	'evil'
		/gen˥˧ /	'study'	/giap˨˩/	'press'

the nasals as /b/ and /g/ do. Table 2 shows the same fifteen morphemes embedded in phrase initial position. The 45 phrases with morphemes embedded in phrase-medial position are shown in Table 3. The segments preceding the chosen morphemes included vowels, voiceless stops, and nasals. All tokens were randomized. There are two repetitions for most items in the corpus. Due to copying mistakes in making the lists, some items were repeated only once, while others were repeated three times. There are all together six such list-making mistakes. However, this should not affect the interpretation of the results, since there is large number of samples (154 tokens)

Table 2 The phrases with the target morphemes embedded in initial position.

CV		CVN	
/ba˧bi˨˩/	'paralysis'	/baŋ˥aʔ˥˧/	'mosquito'
/la˨˩təŋ˥/	'stir soup'	/ban˧fun˥/	'(train, airplane) delay'
/gia˧kaĩ˥/	'centipede'	/lan˥ta˧e˨˦/	'we'
		/laŋ˧baʔ˨˩/	'human flesh'
		/giam˧gek˥˧/	'strict'
		/gen˥kiu˨˩/	'research'
CV+voiceless stops			
/baʔ˥˧wan˨˦/	'meat ball'		
/bak˨˩tçiu/	'eye'		
/lak˨˩siaĩ˥/	'six pairs'		
/lap˨˩s^{W}ey˨˩/	'pay tax'		
/giap˥˧aʔ˥˧/	'tong'		
/giat˨˩tsu˥˧/	'disobedient child'		

Table 3 Examples of phrases with target morphemes embedded in the phrase medial position after vowels, voiceless stops, or nasals (The preceding context segment and target voiced stops are in bold face.)

A. Preceded by vowels	
<u>-V + CV</u>	
/sio˥dz**i**˧˥**ba**˧pi˧˩/	'polio'
<u>-V + CVN</u>	
/**ɔ**˧**ba**ŋ˥la˥˧/	'black mosquito'
<u>-V + CV[voiceless stops]</u>	
/l**ɔ**˥**ba**ʔ˥˧pəŋ˧/	'pork stew on rice'
B. Preceded by nasals:	
<u>-N + CV</u>	
/la**ŋ**˧**ba**˧pi˧˩/	'paralyzed person'
<u>-N + CVN</u>	
/hu**n**˧**ba**ŋ˥la˥˧/	'kill mosquito with smoke'
<u>-N + CV[voiceless stops]</u>	
/ku**n**˧**ba**ʔ˥˧so˧˩/	'simmer ground meat'
C. Preceded by voiceless stops	
<u>-[voiceless stops]+CV</u>	
/tau˧k^{h}a**k**˧˩**ba**˧pi˧˩/	'brain paralysis'
<u>-[voiceless stops] + CVN</u>	
/p^{h}a**ʔ**˥˧**ba**ŋ˥la˥˧/	'kill mosquito'
<u>-[voiceless stops]+CV[voiceless stops]</u>	
/tɕia**ʔ**˥˧**ba**ʔ˥˧piã˥˧/	'eat meat patty'

for each subject, and the results are discussed in terms of percentages, instead of absolute numbers.

Procedure

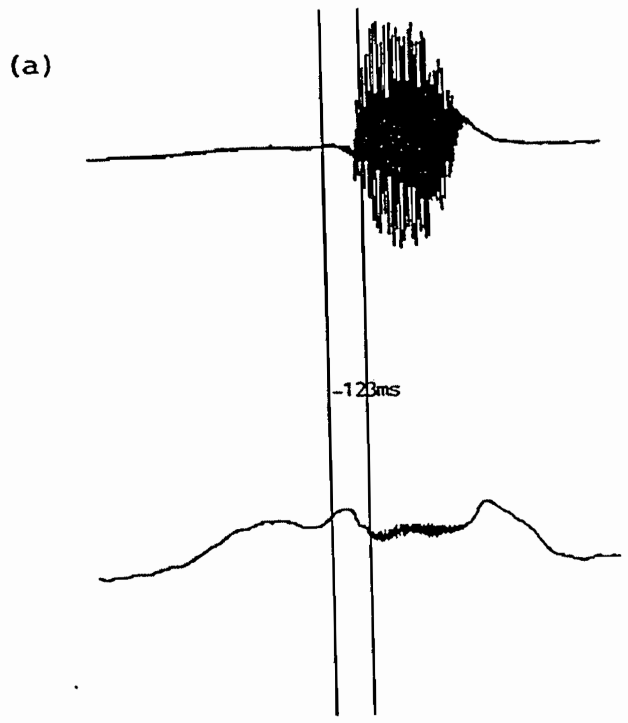
Each subject was recorded in a sound booth holding an air flow mask against his or her own face while reading the corpus. The experimenter was outside of the booth controlling Cspeech to record the airflow data.

The recording consisted of three sessions. In the first session, subject were instructed to produce phrases with chosen morphemes embedded in medial position.

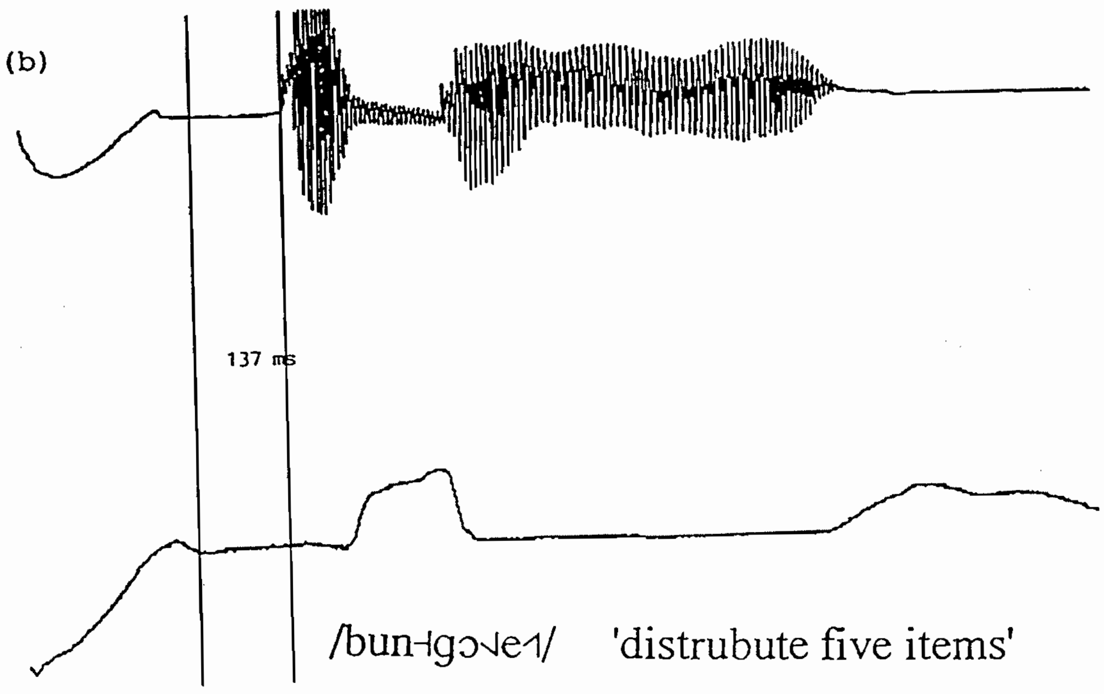
In the second session, phrases with chosen morphemes in initial position were produced. In the third session, subjects were instructed to produce individual morphemes in citation form.

Data analysis

Data were displayed and analyzed with Cspeech 4.0. See sample displayed in Figure 4. The oral and nasal airflow were compared to determine the presence of prenasalization, and the two positions were marked, as shown in the figure. Cursor A was placed at the point where the nasal air flow ceases. Cursor B was placed at the point where the oral air flow of vowel starts, as shown in Figure 4. In utterance initial position, to avoid including subjects' breathing as the prenasalization of voiced stops, 50 ms is used as the cut-off point. If the duration between A and B is less than 50 ms, then the token is judged to be prenasalized. Any trace of nasal airflow that are more than 50 ms prior to the onset of voicing is judged to be breathing. When in phrase medial position, if the duration between point A and B is less than 10 ms, then the voiced stop is judged to be fully nasalized. The reason why 10 ms was used here as a cut off point is because there is a 10 ms delay in the nasal air flow due to the 30 Hz filter used.



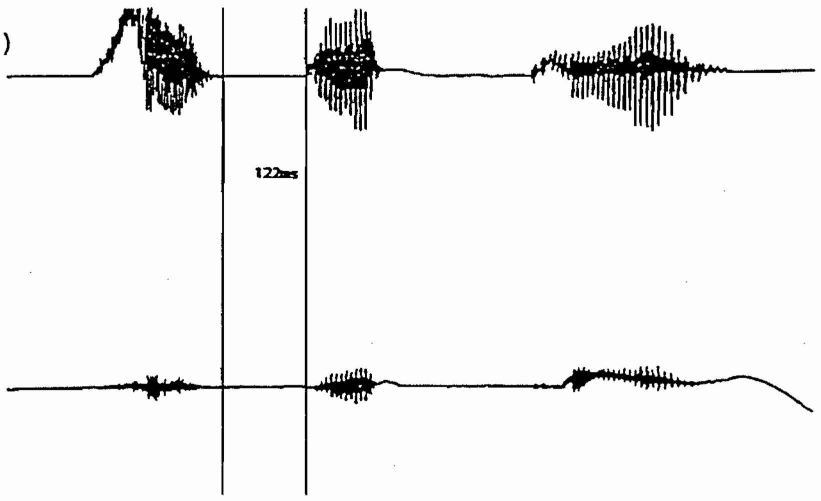
/ba˥/ 'numb'



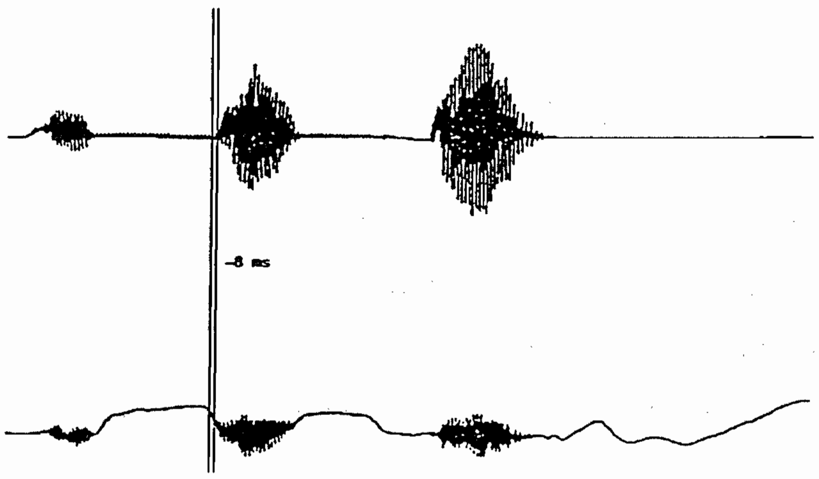
/bun˧gɔ˨e˥/ 'distrubute five items'

Figure 4. Example of air flow analysis. (a) /ba˥/ 'numb' (b) /bun˧gɔ˨e˥/ 'distribute five items'. The upper channel is oral airflow. The lower channel is nasal airflow.

(c)



(d)



(e)

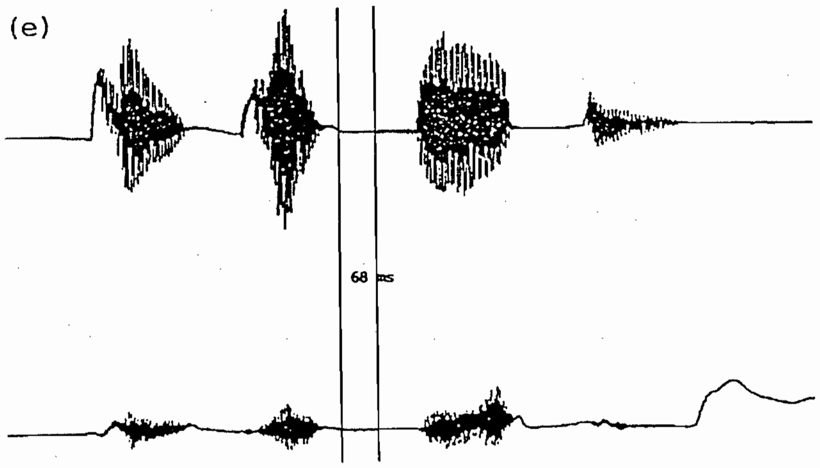


Figure 4 (Continued) (c) /ɔ˧gia˧kaŋ˥/ 'black centipede' (d) /tɕin˧giam˧ke˧/ 'very strict' (e) /tau˧k^{h}ak˩ba˧pi˧˩/ 'brain paralysis'

Results of Adult Air Flow Study

The results of airflow data will be discussed according to the three types of contexts: (1) citation form (2) phrase-initial position, and (3) phrase-medial position after (3) vowels, (4) nasals, and (5) voiceless stops.

Citation form

Table 4 and 5 shows the percentage of prenasalized voiced stops in citation form and phrase-initial position. Although tokens in citation form and phrase-initial position were recorded in different sessions, they are essentially in the same utterance initial position. Therefore the results are discussed together. For voiced stops in citation form, there does not seem to be any uniform pattern across speakers for the presence of prenasalization. The occurrence of prenasalization was very speaker-dependent. While subject CF, WS, and BH had low percentages of prenasalization for voiced stops in citation forms, YF, LK, HJ, and CC had high percentages of prenasalization. There was also no consistent pattern of differences across the syllable types, either.

The results from morphemes placed in phrase-initial position are also shown in Table 5. Again, there was no uniform pattern for the presence of prenasalization. Whether a token shows nasal air flow during the early part of the closure or not seems to be arbitrary. While some subjects, like YF, LK and HJ, had high percentage of prenasalization for the voiced stops, others like CF and WS had low percentage of prenasalization. Again, the occurrence of prenasalization is very speaker-dependent

Phrase medial position

The results of voiced stops embedded in phrase-medial position are shown in Table 6. For those voiced stops preceded by vowels, prenasalization was found only in nine tokens and all these cases were due to the existence of alternate

Table 4 Percentage of prenasalized voiced stops produced in citation form:

Subjects	Syllable Structures		
	[voiced stops]+V	[voice stops]+V+[stops]	[voiced stops] + V+ N
CF	25.00% (3/12)	16.67% (2/12)	10.00% (1/10)
YF	100.00% (12/12)	91.67% (11/12)	100.00% (10/10)
LK	100.00% (12/12)	100.00% (12/12)	100.00% (10/10)
HJ	75.00% (9/12)	75.00% (9/12)	70.00% (7/10)
WS	58.33% (7/12)	33.33% (4/12)	40.00% (4/10)
BH	33.33% (4/12)	75.00% (9/12)	70.00% (7/10)
CC	100.00% (12/12)	83.33% (10/12)	60.00% (6/10)

Table 5 Percentage of prenasalized voiced stops produced in phrase-initial position:

Subjects	Syllable Structure		
	[voiced stops]+V	[voiced stops]+V+[stops]	[voiced stops] + V+ N
CF	37.5%	76.15%	66.67%
	(3/8)	(10/13)	(6/9)
YF	100.0%	92.31%	100.0%
	(8/8)	(12/13)	(9/9)
LK	87.5%	92.31%	100.0%
	(7/8)	(12/13)	(9/9)
HJ	100.0%	100.0%	88.89%
	(8/8)	(13/13)	(8/9)
WS	62.5%	61.54%	33.33%
	(5/8)	(8/13)	(3/9)
BH	87.5%	92.31%	77.78%
	(7/8)	(12/13)	(7/9)
CC	87.5%	84.62%	77.78%
	(7/8)	(11/13)	(7/9)

pronunciations for the preceding or target words. For example, the word 'hair' can be /t^hau˧mɔ˧/ or /t^hau˧məŋ˧/. The prenasalization occurred when the subjects produced nasals instead of vowels to precede the voiced stops. For example instead of /t^hau˧m ɔ˧**g**iap˨˩a˥˧/ CF, YF, and WS produced four tokens of /t^hau˧mə**ŋ**˧**g**iap˨˩a˥˧/ 'hair pin'. In another case, CF and HJ produced three tokens of /sio˥dzi˧˥**m**ã˧pi˨˩/ instead of /sio ˥dzi˧˥**b**a˧pi˨˩/ 'polio'. That is , they used an alternate form of the target morpheme with a nasalized vowel and hence produced a nasal /m/ rather than the targeted /b/ for the initial. CF and YF also produced two tokens of /hue˥t**ɕia**˥**m**an˨˩hun˥/ instead of /hue˥t**ɕia**˥**b**an˨˩hun˥/ 'train delay'. In all of these cases, the subjects did not produce a vowel + voiced stop sequence as directed. After excluding these cases, no other prenasalized voiced stops preceded by vowels were found.

When the voiced stops are preceded by voiceless stops, the nasalization was found in only three cases, as shown in Table 6. In HJ's case, because of alternative pronunciation in the same dialect, he produced /tau˧k^ha**k**˥˧**b**a˧pi˨˩/ as /tau˧k^ha**k**˥˧**m**a˧ pi˨˩/ . YF, on the other hand, produced a homorganic nasal instead of the target stop, e. g. /tɕia**ʔ**˧**m**aʔ˥˧pia˥˧/, instead of /tɕia**ʔ**˧**b**aʔ˥˧pia˥˧/ 'eat meat patty'. However, in the second repetition of the same item, CF and YF produced voiced stops as instructed, and the voiced stop was not prenasalized. Apparently when preceded by voiceless stops, the voiced stops are not prenasalized.

For the voiced stops preceded by nasal consonants, almost 100% of the tokens were fully nasalized, as shown in Table 6. According to the airflow data, the voiced stops changed into nasals when preceded by nasals. However, the spectrogram of these voiced stops indicated that the plosive quality was somehow still preserved.

When the segment preceding the voiced stops is a nasal, the nasalization of the preceding nasal extends all the way through the voiced stop closure, as shown in the air flow data of Figure 5.

From the first glance of the air flow data it seems that the voiced stop is assimilated to become a nasal when preceded by a nasal. If this is indeed the case, then the only distinction between phrases such as, /saŋ˥˧bi˥˧/ 'send rice' (Figure 6), which has nasal airflow extending all the way into the voiced stops, and /saŋ˥˧mĩ˧/ 'send noodles' (Figure 7) is the nasalization or lack of it on following vowel. However, acoustic spectrographic data revealed acoustic differences in the closure portion, as shown in Figure 6.

When comparing /saŋ˥˧bi˥˧/ ' send rice' (Figure 6) with /saŋ˥˧mĩ˧/ 'send noodles' (Figure 7) three acoustic characteristics differentiate /b/ from /m/. First is the gap in low-frequency energy at the end of the /b/, immediate preceding the vowel; second is a strong burst release found at the onset of vowel following /b/; third is the strong attack and immediate rise in energy in the oral flow at the onset of vowel following /b/ suggested for a post-stopped nasal. This strong burst of energy is similar to the oral release of the Zhongshan post-stopped nasal (Chan & Ren, 1987). Post-stopped nasals are composed of a nasal segment with a strong burst release at the end. Zhongshan is one of several other Chinese dialects that has been reported to have nasal+stop segments, but instead of prenasalized stops, Zhongshan is said to have post-stopped nasals. Of the Chinese dialects with complex nasal+stop segments, Taiwanese represents an unique case of a prenasalized stop in which the stop is the more dominant element than the nasal (Chan & Ren, 1987). Most of other nasal + stop segments of Chinese dialects are post-stopped nasals, in which the nasals are more prominent than the stop components. However, judging from the result of acoustic and air flow data, it is likely that Taiwanese prenasalized stops change into post-stopped nasals when preceded by nasals.

Table 6. Percentage of prenasalized voiced stops produced in phrase-medial position:

Subjects	Preceding Segments		
	V	/p/,/t/,/k/,/ʔ/	N
CF	9.375% (3/32)	0.0% (0/27)	100.0% (31/31)
YF	9.375% (3/32)	3.70% (1/27)	100.0% (31/31)
LK	0.0% (0/32)	0.0% (0/27)	96.78% (30/31)
HJ	6.25% (2/32)	3.70% (1/27)	100.0% (31/31)
WS	3.125% (1/32)	0.0% (0/27)	96.78% (30/31)
BH	0.0% (0/32)	0.0% (0/27)	100.0% (31/31)
CC	0.0% (0/32)	0.0% (0/27)	100.0% (31/31)

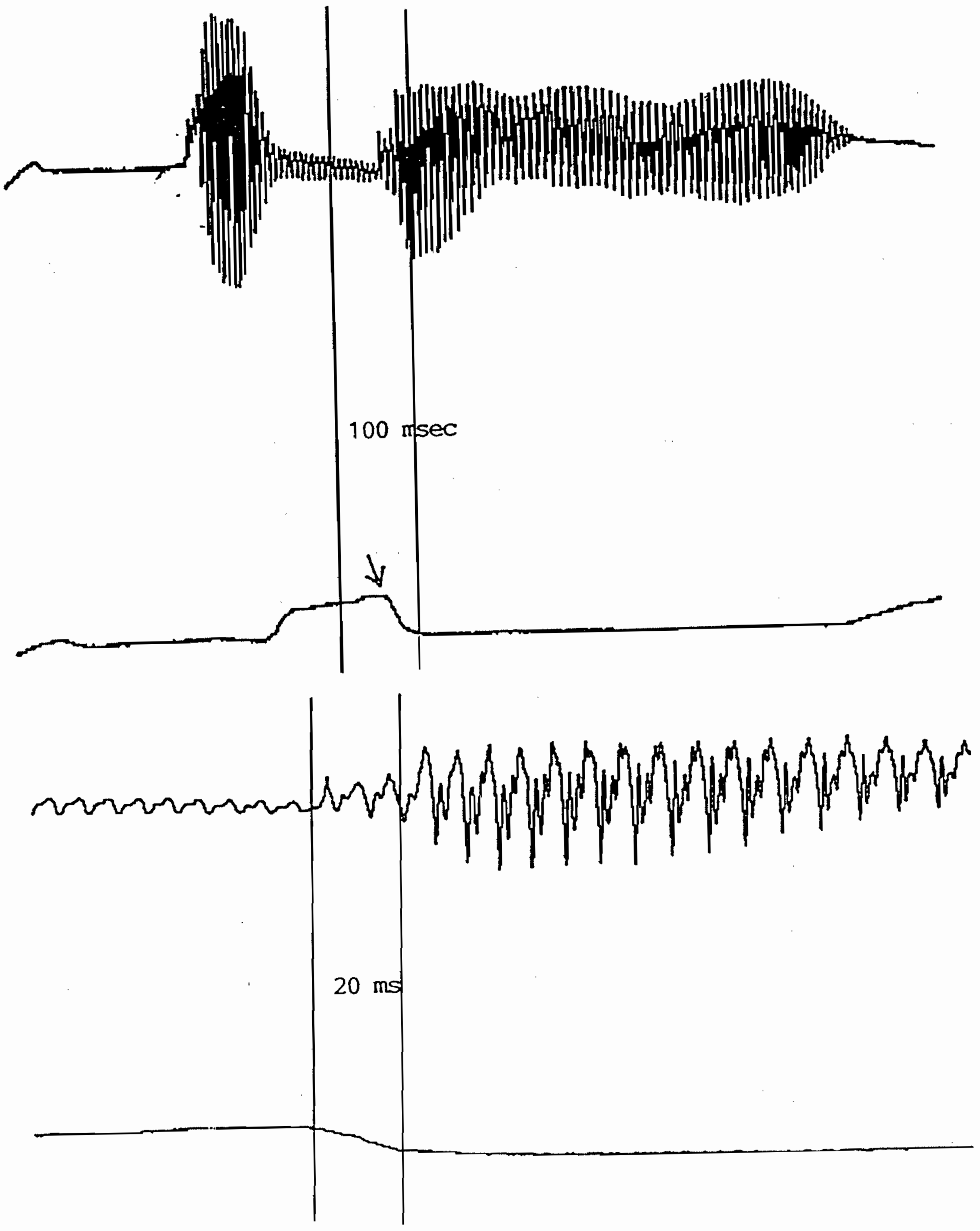
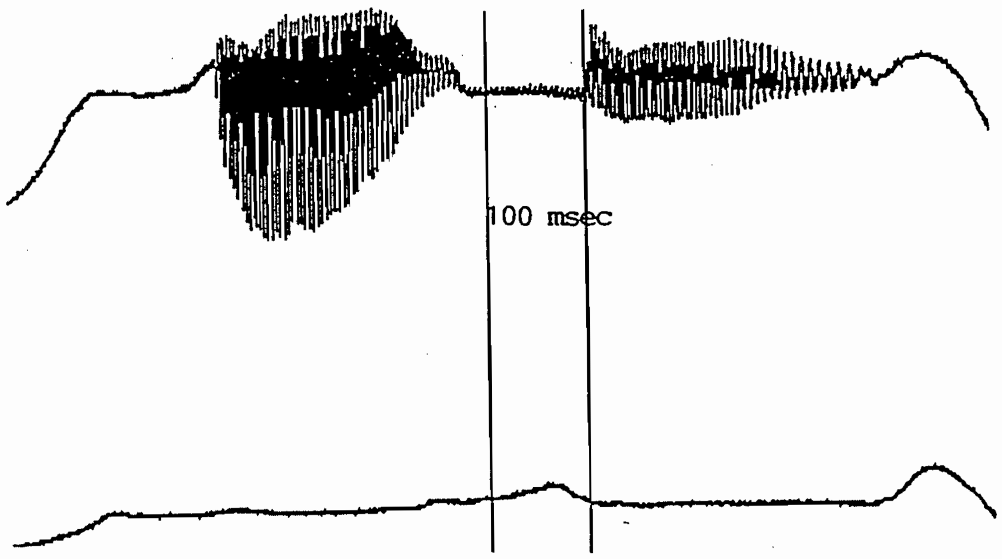
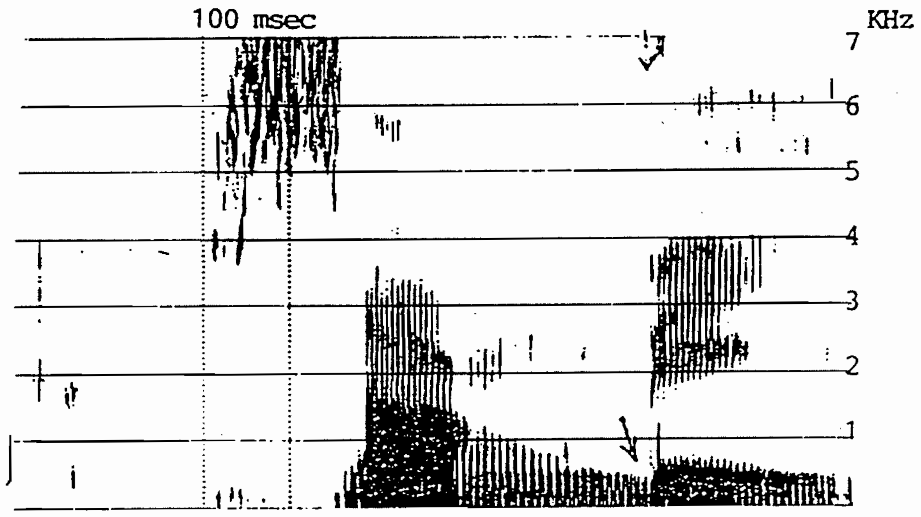


Figure 5. Air flow data of Taiwanese voiced stop preceded by a nasal. (a) /bun˧gɔ˧˩e˧˥/ 'distribute five items' (b) zoom in of the 100 ms in (a) The upper channel is the oral air flow. The lower channel is the nasal air flow.

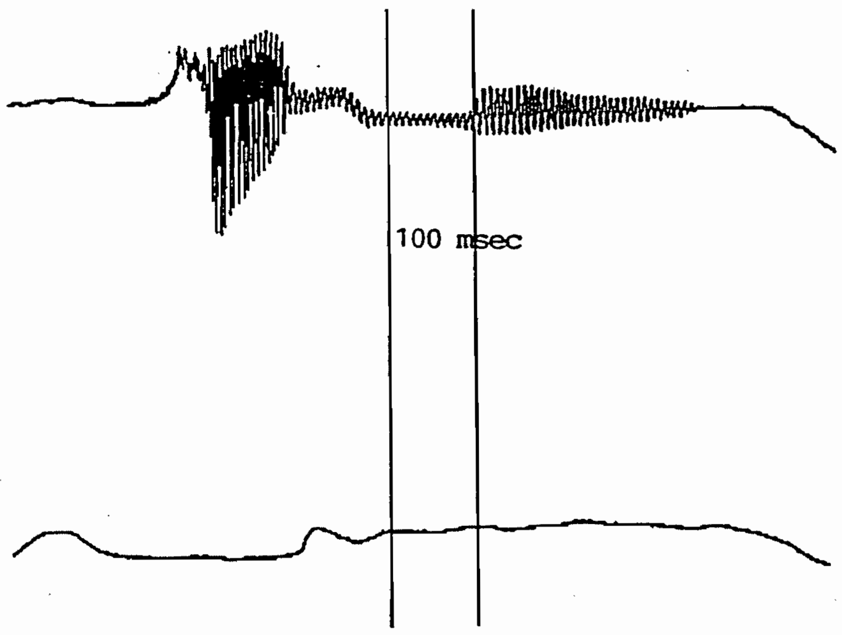


(a) Air flow: the upper channel is oral air flow, the lower channel is nasal air flow

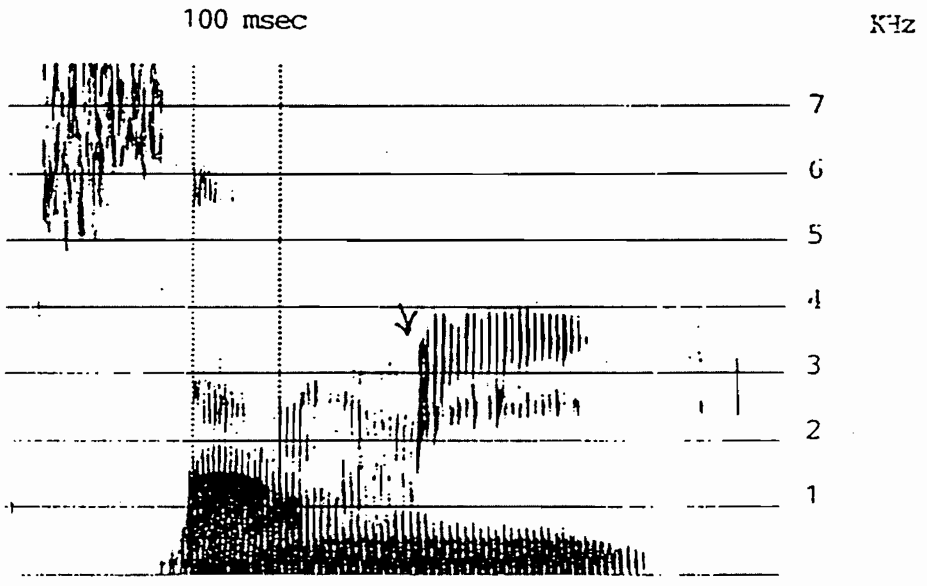


(b.) Acoustic data: spectrogram

Figure 6. (a) Air-flow and (b) acoustic data of a Taiwanese voiced stop preceded by a nasal /saŋ˥biɁ˥/ 'send rice'. The tokens in (a) and (b) are not the same, but the speaker is the same. The release gap in low frequency energy by the lower arrow, and the strong burst by the higher arrow.



(a) Air flow: the upper channel is oral air flow, the lower channel is nasal air flow



(b.) Acoustic data: spectrogram

Figure 7 (a) Air-flow and (b) acoustic data of a Taiwanese nasal preceded by a nasal /saŋ˥˧mĩ˧/ 'send noodles'. The speaker is the same as for Figure 3.11. The tokens in (a) and (b) are not the same, but the speaker is the same. Note the lack of a gap or strong burst at the arrow.

Summary

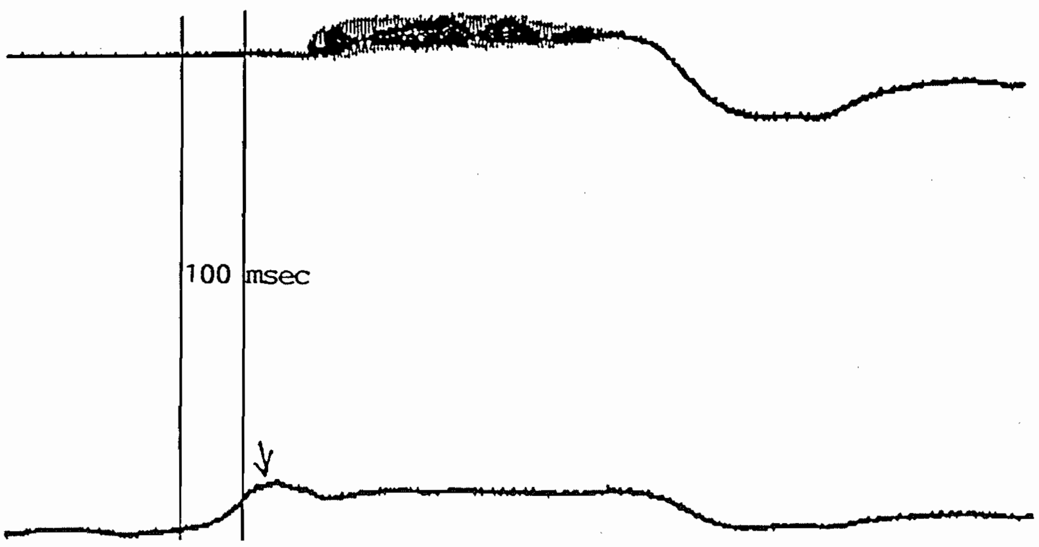
The phonetic variants of Taiwanese prenasalized voiced stops are not only influenced by the following segments, as proposed by Zhang's (1983) allophonic rule. They are also influenced by preceding segments. The prenasalization is lost when preceded by vowels, or voiceless stops. When preceded by nasals the prenasalized voiced stops change into post-stopped nasals. Only in utterance-initial positions do prenasalized voiced stops occur, but their occurrence varies in an speaker-dependent manner.

CONCLUSION

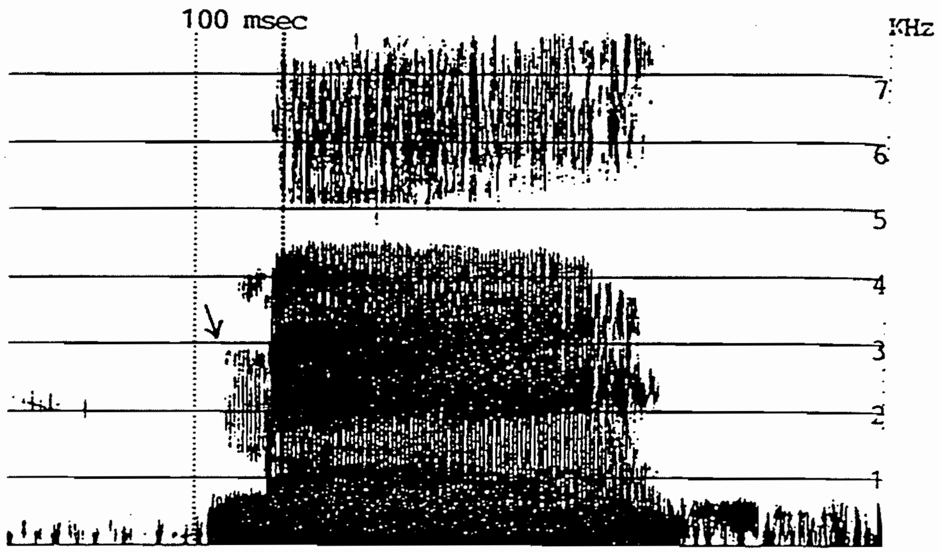
Relationship Between Initial Voiced Stops and Nasal

Zhang (1983) proposed a phonological rule relating Taiwanese initial voiced stops in forms such as /be˥˧/ ' horse' with initial nasals in forms such as /mẽ˧/ ' scold' (Figure 8). That is, noting a complementary distribution between voiced stops and nasals in initial position and citation form, he categorized the two types of sounds together as voiced stop phonemes claiming that Taiwanese initial voiced stops change into homorganic nasals when the syllable is closed by a final nasal or a nasalized vowel. In other words, he claims initial [m] is an allophone of /b/ and initial [ŋ] an allophone of /g/ before nasalized vowels and rhymes closed with nasals. However, the subjects in this study produced /b/ in /ban/ 'slow' as [b], not [m] as Zhang (1983) predicted (Figure 9). There are no extra nasal formants above 100 kHz in the initial consonants portion. The segmental environment proposed in Zhang's (1983) allophonic rule does not describe what happen here.

It is unclear whether the two series are indeed in an allophonic relationship, or whether there is a natural gap such that initial voiced stops occur before unnasalized vowel, while nasals occur before nasalized vowels, and nasals. Since there are no

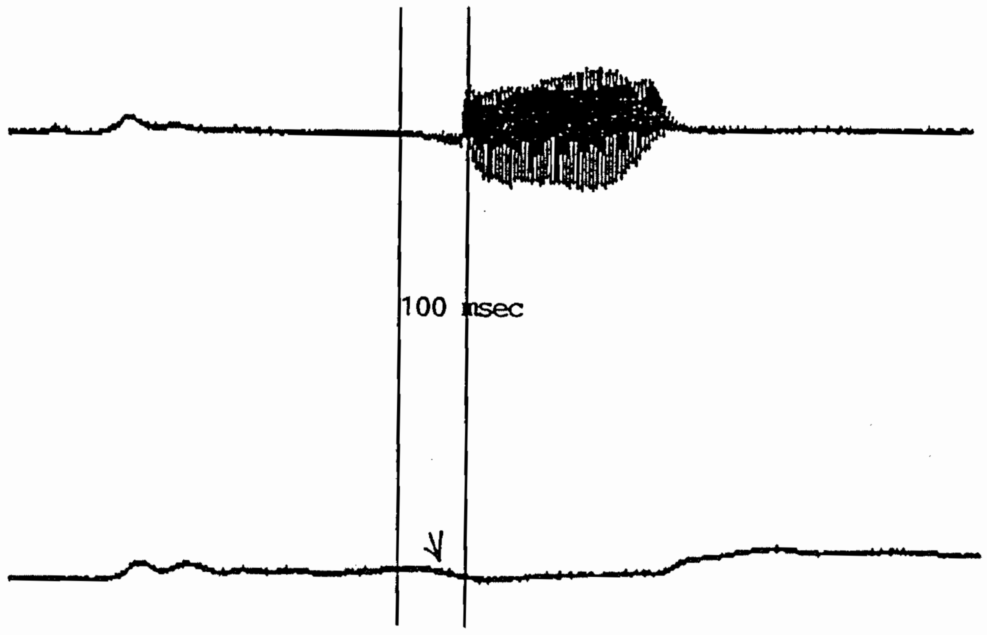


(a) Air flow: the upper channel is oral air flow, the lower channel is nasal air flow

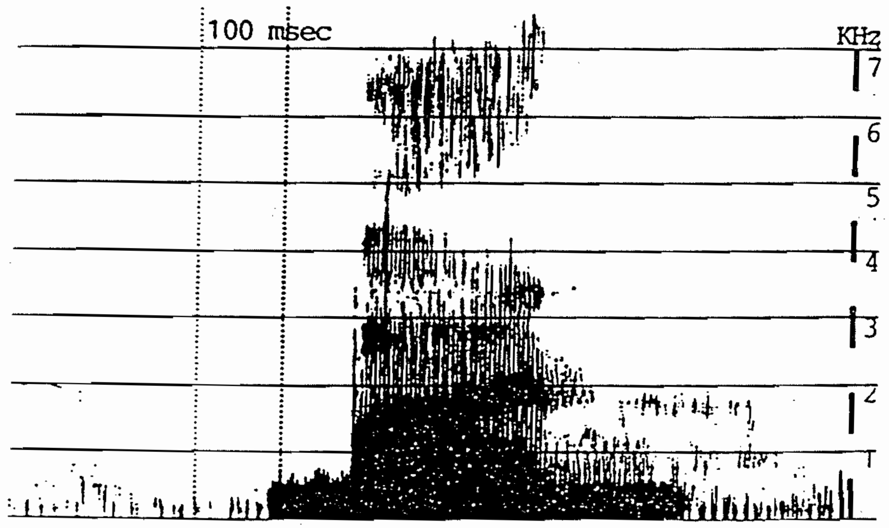


(b.) Acoustic data: spectrogram

Figure 8. (a) air flow and (b) acoustic data of Taiwanese voiced stops followed by nasalized vowels /mẽ˧/ [mẽ˧] 'scold'. The time scales of (a) and (b) are different. The tokens in (a) and (b) are not the same, but the speaker is the same.



(a) Air flow: the upper channel is oral air flow, the lower channel is nasal air flow



(b.) Acoustic data: spectrogram

Figure 9. (a)Air-flow and (b) acoustic data of Taiwanese voiced stops followed by a vowel and a final nasal /ban˧/ [ᵐbban˧] 'slow'. The time scales of (a) and (b) are different. The tokens in (a) and (b) are not the same, but the speaker is the same.

morphological alternations supporting the identification of the two sets of initials as coming from a common underlying form. Further psychological studies, such as a perception test, are necessary to investigate native speakers' intuitions. complementary distribution alone is not merely enough to support for an allophonic relationship. Therefore, before any further psychological studies are done, no further claim will be made as to the relationship between the two series.

Phonetic Variants of Taiwanese Voiced Stops.

The results of the previous airflow study showed that when the voiced stops are in citation forms or other phrase-initial position, the presence of prenasalization was highly speaker-dependent. Some speakers tend to prenasalize utterance initial voiced stops more than others.

The only consistent phonetic patterns observed across subjects were in the three phrase-medial positions, when the voiced stops were preceded by segments such as vowels, nasals, and voiceless stops. It was discovered that when preceded by vowels or voiceless stops, no prenasalization occurred and the sounds are simply voiced stops. When preceded by nasals, the voiced stops change into post-stopped nasals.

Summarizing from the results of the airflow and acoustic measurement we can conclude that the so-called Taiwanese prenasalized voiced stop only appears in utterance-initial position, but its appearance is very speaker-dependent. This prenasalization is lost when the stop is preceded by vowels or voiceless stops. When preceded by a nasal, the prenasalized voiced stop changes into a post-stopped nasals, phonetically.

REFERENCES

Chan, M., & Ren, H. (1987) "Post-stopped nasals in Chinese: An areal study," *UCLA Working Papers in Phonetics,* **68**, 73-120.

Chan, M. (1987) "Post-stopped nasals in Chinese: An acoustical investigation," *UCLA Working Papers in phonetics,* **68**, 121-131.

Iwata, R. et al., (1979) "Laryngeal adjustments of Fukienese stops, initial plosives and final applosives,". *Annual. Bulletin. Research Institute Logopedics Phoniatircs* **13**, 61-81.

Pan, H. (1994) "The voicing contrasts of Taiwanese (Amoy) initial stops: Data from adults and children," The Ohio State University, Ph. D. dissertation.

Zhang, Z. (1983). *Taiwan Minnan Fangyan Jilue,* [A brief description of the Southern Min dialects in Taiwan]. Taipei: Wen Shi Tse Chubanshe.

Maintenance of Machine-Readable Dictionary

Yasuhito TANAKA
Hyogo University

2301 Shinzaike Hiraoka, Kakogawa, Hyogo
675-01 JAPAN
TEL : +81−794−27−5111 FAX : +81−794−27−5112

Abstract
This article discusses various problems related to machine-readable dictionaries. The author focused our attention on the corpus as a means of finding entries. Furthermore, we discusses these problems from the standpoint of users and also from the standpoint of system development. At the same time, the questions of verification of conversion accuracy and overall evaluation of developed dictionaries were studied.

0) Introduction

Machine-readable dictionaries have been compiled by companies, institutes, software houses, and others. No machine-readable dictionary maintains its usefulness without revision. Unless it is continuously renewed or maintained, it quickly becomes outdated. Here we will cite various problems related to the maintenance of dictionaries and introduce the results of tests regarding the solution of the above-mentioned problems.

1) Problems related to maintenance of dictionaries

i) Dictionaries become dated year by year.

Although a machine-readable dictionary may be up-to-date when it is completed, it becomes progressively more outdated year by year. For instance, one can easily recognize that sentences written 10, 20 or 50 years ago are old in content, concepts, diction, etc. Similarly, dictionaries get old.

ii) Assurance and maintenance of personnel for renewal

Personnel must be secured for the maintenance of a machine-readable dictionary. Such personnel must be experienced in the processing of natural language and have an adequate knowledge of linguistics. Furthermore, they must continuously be able to complete a fixed amount of work within a given period. This is a very difficult task.

iii) How to obtain data for maintenance

If a product using a machine-readable dictionary is offered on the market, it is possible to obtain data for renewal just by gathering complaints from users.

Another way is to extract words from the corpus, and refer such words to a machine-readable dictionary. If there are words that are not found in the dictionary, new words that appear with high frequency may be added to the dictionary. This positive method is important.

2) How can data for maintenance of a machine-readable dictionary be obtained, and how should unknown words be handled?

In developing a system for processing natural language, it is necessary to perform the following three types of maintenance for a machine-readable dictionary.

(1) Maintenance for solving problems

If a new word (unknown word) is found, its meaning should be determined based on its outward characteristics. It is impossible to deal with new words in the course of sentence processing.

(2) Maintenance according to users' demands

New words are added to a machine-readable dictionary according to users' demands.

It is possible to have users make proposals. For instance, an electronic bulletin board can be provided in a communication network system to collect data. The degree of success of this method depends greatly on whether the information receive rewards or some kind of privilege.

If the consent of users can be obtained, words, compound nouns, idiomatic expressions, technical terms, coincidental relations, etc., that the users use and add to their machine-readable dictionaries, may be collected so that words, etc., of common use can be selected and incorporated in a dictionary for a natural language processing system.

(3) Preventive maintenance

There is also a method of continuously extracting new words from a large corpus and registering them in a machine-readable dictionary. By this method, it is possible to analyze a number of words according to the frequency with which they appear. It is also possible to use corpuses devoted to different areas. This is a positive and important task in the maintenance of machine-readable dictionaries.

It is necessary to find and eliminate as many detects as possible before a product is put on the market. The correction of defects after shipment of the product requires tremendous costs.

Here we will focus on (3) Preventive maintenance.

3) Extraction of words from a corpus

3-1)

In extracting words from a corpus, it is possible to analyze sentences and extract words. However, this would require a great deal of time and effort for processing, and a completed machine-readable dictionary would be needed. Unfortunately, a dictionary will never be complete.

Another way is to mechanically separate text into words and then to select words. Words can be extracted incorrectly when this method is used, so care should be taken in incorporating them into a dictionary. It is possible to refer words extracted from a corpus to those already registered in a dictionary and to consider adding unmatched words as candidate words to be newly incorporated in a dictionary.

This method is advantageous in that it depends to a considerable extent on mechanical processing, and a large number of candidate words can be found in this way.

Thus, we extracted three-Chinese-character words from a data file containing the full text of the Asahi Shimbun over a period of a year.

Code	№ of different words	Total number of words	Code
10	35,623	406,827	10…common nouns
80	10,427	96,946	80…proper nouns (names of persons)
90	3,312	24,536	90…proper nouns (place names)
95	763	3,292	95…numerals
99	17,941	112,711	99…others
Total	78,066	644,312	

3-2) Checking of words (examples)

One way to check words is to check words extracted by a mechanical method. Here we will introduce another method. A large number of four-Chinese-character words are divisible into combinations of two two-character conceptual words. These two-character words should be included in a dictionary. We examined whether such two-character words are included in a machine-readable dictionary.

Machine-readable dictionary

	№ of different 2-character words	Total number of different of 2-character words
Yes	10,512	794,649
No	3,336	17,740
Total	13,848	812,389

The above table shows that since two-character words are basic words, the machine-readable dictionary covered 97.8 percent of the total number of different two-character words.

We conducted a similar analysis of the data provided by the Japan Scientific and Technical Information Center, and obtained the following results.

Machine-readable dictionary

	No. of different 2-character words	Total number of different of 2-character words
Yes	8,3711	519,820
No	9,3821	11,889
Total	17,753	1,631,709

It is clear that there are more unmatched words for the data obtained from the Japan Scientific and Technical Information Center.

This means that the evaluation of machine-readable dictionaries differs depending on the main areas for which they are developed.

Five-character words were extracted and divided into two-character and three-character words. Of these, three-character words were checked with a machine-readable dictionary.

Machine-readable dictionary (3-character words contained within strings of five characters)

	No. of different 3-character words	Total number of of 3-character words
Yes	11,034	109,051
No	2,492	8,311
Total	13,526	117,362

The above figure shows that the machine-readable dictionary covered 92.9 percent of the total data.

Similarly, two-character words contained within strings of five characters found in the Asahi Shimbun file were checked with a machine-readable dictionary.

A very good result was obtained, as the machine-readable dictionary covered 96.1 percent of the total data.

Machine-readable dictionary (2-character words contained within strings of five characters)

	No. of different 2-character words	Total number of of 2-character words
Yes	5,573	113,947
No	843	3,415
Total	6,416	117,362

Machine-readable dictionaries are maintaine d by analyzing data obtained in this way.

4) Four-character and five-character words with furigana

An attempt was made to determine how many of the four-character words that can be divided into two two-character words can have furigana (phonetic transcriptions in kana written at the side). The following table shows the number of four-character words which can have furigana entirely.

Four-character words with furigana

	No. of different four-character words	Total number of four-character words
Yes	68,465	377,062
No	11,212	29,133
Total	79,677	406,195

A similar attempt was made for five-character words that can be divided into two-character and three-character words, or vice versa, and the number that can have furigana was determined.

Five-character words (2•3)

	No. of different five-character words with furigana	Total number of five-character words with furigana
Yes	14,385	49,390
No	3,320	7,881
Total	17,705	57,271

Five-character words with furigana (3•2)

	No. of different five-character words with furigana	Total number of five-character words with furigana
Yes	18,181	52,043
No	3,895	8,050
Total	22,076	60,093

When providing furigana to Chinese characters it should be remembered that the first consonants in the latter parts of four-character and five-character words tend to be voiced by liaison after the first parts. Ex. Kabushiki Kaisha (Gaisha) Data obtained in this way can be excellent material for addition to machine-readable dictionaries.

3-3) Checking with a new file

Three-character words were extracted from the file containing one year of the full text of the Yomiuri Shimbun and were added to those extracted from the Asahi Shimbun file.

Three-character words from the one-year Yomiuri Shimbun file

Code	No. of different 3-character words	Total No. of 3-character words
10	19,960	432,426
80	3,571	91,261
90	1,837	31,221
95	443	6,948
99	6,969	57,219
None	66,121	234,208
Total	98,901	853,283

This table shows that about 66,000 different three-character words out of the total of about 99,000 are codeless. This means that about 67 percent of the total different words and 27.4 percent of the total number of words are unmatched data. This increase in unmat ched data is probably due to the mechanical method of extraction and also to the fact that a large number of the words contain numerals. It is also a fact, however, that there are more words that are required to be registered. So far as common nouns (Code 10) are concerned, about 20,000 words are matching words, and the total number of such words is about 432,000. Furthermore, 66,000 different words are code-less, and they appear a total of about 234,000 times, or an average of 3.5 times per each words. It will be necessary in the future to analyze these words one by one and register them in the dictionary. About 66,000 different word s are now being analyzed.

3-4) Method of analysis

Checking with existing files is the first step in analysis. In this analysis, only about 30 percent are matching words. Therefore, the following empirical method should better be followed.

However, it should be remembered that there are exceptions to any rule.

(1) Three-character words containing one of the following characters are place names (90).

県、市、区、郡、町、村、字、省 and three-character words of which the last character is following 浦、駅、沖、河、海、街、岳、橋、郷、局、圏、原、湖、江、港、崎、山、坂、寺、州、沼、川、線、台、沢、谷、地、島、峠、灘、岬、野、路、湾

(2) Three-character words containing one of the following characters are numerical expressions (95).

一、二、三、四、五、六、七、八、九、十、拾、百、千、万、億、兆

(3) Three-character words containing one of the following characters as the last character are names of persons or proper nouns (80).

～氏、～子

(4) Words beginning with the following code are "other nouns" (99).

案外、以下、以外、以上、以後、以前、以来、依然、一応、一括、一見、一言、一向、一時、一瞬、一生、一切、一層、一体、一度、一回、一般、一晩、一番、一部、一歩、一面、一目、一律、何人、何度、過去、回、極力、月末、現在、現代、今後、今回、今後、今月、今春、今週、今度、今日、今年、再三、再度、最近、最終、最大、最低、最高、昨年、氏

(Characters at the beginning of words)

時折、時代、若干、最終、終日、週間、週末、十分、従来、順次、将来、場合、常時、随分、数日、数年、世紀、世代、絶対、先月、先週、先日、戦後、戦前、前回、前後、前日、前年、前夜、全国、全身、全然、全部、全面、早速、相当、即刻、多少、多数、対応、断然、直接、通常、程度、当時、当初、当然、当分、当日、当面、到底、同年、突然

There are also other words with various characteristics, and we are now analyzing them. When the two files are integrated, we obtain the following table.

Three-character words in the files of Asahi Shimbun and Yomiuri Shimbun Files

Code	№ of different 3-character words	Total № of 3-character words
10	35,623	839,253
80	10,427	188,207
90	3,312	55,757
95	763	10,240
99	27,940	169,930
None	66,121	234,208
Total	144,186	1,497,595

"None" denotes that code numbers are now being added.

When the analysis of 66,000 different words has been completed, a better file of analysis will be obtained. Furthermore, 60-70 percent of the different three-character words can be automatically coded.

We have discussed various problems related to dictionary maintenance from the standpoint of compilers and managers of machine-readable dictionaries. However, it is also necessary to view these problems from the standpoint of users.

4) Satisfaction of users

Experiences of people engaged directly or indirectly in the development of machine-readable dictionaries show that the demands of their users are satisfied in different degrees.

(1) When no machine-readable dictionary was available and different organizations had to compile their dictionaries, the existence of a machine-readable dictionary itself was highly ⁄ appreciated. Even a small-scale dictionary was welcomed.

A dictionary with 500,000-100,000 entries was appraised by users.

However, as the number of users increased and as the utilization areas of machine-readable dictionaries expanded, users became more demanding.

(2) They wanted an increase in the number of entries. Furthermore, they wanted to have conditions for their usage clarified so that they could select suitable words more easily. Thus, they wanted a dictionary with 200,000-300,000 entries.

We are now at this stage of development of machine-readable dictionaries.

(3) In addition to the demand for a largerscale dictionary, there will also be a demand for more detailed descriptions of each entry, semantic contents, origins and related information on words, as well as for clarification of usage of words. Machine-readable dictionary will come to be questioned as to their scale and c ontents, and a dictionary with about 1 million entries will be desired.

5) Viewpoint of system development

(1) Before there was a machine-readable dictionary, or when there was one of only a limited scale, system development was processed only by rules, and exceptions were also dealt with by rules. It soon became clear, however, that there were many problems in this method, and systems development got nowhere.

(2) Expansion of scale of dictionary and grouping of entries

The method of expanding a machine-readable dictionary to a certain extent for word-processing proved to be more efficient, and it became clear that there was no need to deal with exceptional entries by rules one by one.

However, there was concern about the extent to which dictionaries could be expanded. Against this background, attempts to group similar entries began.

Active studies were made on markers and their expansion, and also on a thesaurus.

(3) Development of a large-scale dictionary

In order to develop a large-scale dictionary, it is necessary to collect entries for it from a corpus. It has become possible to obtain a large corpus in the form of newspaper files thanks to the electronic photocomposing systems which have been extensively adopted by newspaper publishers. Furthermore, systems for electronically processing manuscripts have been developed for book and magazine publishing houses, and as a result, electronic books and magazines have made their debuts. However, these developments also copyright problems.

Furthermore, the method of dictionary development is changing from a method based on personal work to machine compilation, or a combination of machine compilation and review by personnel. Development of a large-scale dictionary depends to a large extent on computers, and for this purpose, methods of data extraction by statistical processing, the n gram system, and other methods have been developed. However, since the extraction of entries and contents for a dictionary by these methods is mechanical, it should ultimately depend to a great extent on human judgment.

It has become possible to compile a large-scale dictionary thanks to the following three factors.

(1) Research and development efforts have been made to resolve ambiguities, and medium-scale machine-readable dictionaries have been developed. These developments have helped to automate the compilation of machine-readable dictionaries.

(2) Large-scale corpuses have been made available.

(3) Various statistical methods have been developed.

6) Verification of contents of dictionaries

While it is important to compile a large-scale dictionary, it is also necessary to take note of the fact that errors are accumulated. In this context, it is important to consider how to verify the content of each entry in the dictionary.

(1) Verification by personnel

This requires a many people, and the ability of verifiers should be examined.

(2) Method of mechanical inspection

Various dictionaries are compared partially, do to find errors.

(3) To evolve an experimental system for verification

It takes time to evolve such a system, and a lot of time is needed before a dictionary is actually tested.

(4) A large-scale dictionary is partially completed, and entries in the partially completed part are classified and verified.

(5) Others

It is necessary to study the systems to inspect machine-readable dictionaries.

7) Overall evaluation of dictionaries

According to the author's experience in a project carried out around 1972 to convert kana characters to Chinese characters for the names of persons, the development costs required for conversions with accuracy rates of 80, 90, 95 and 99 percent rose from are to two, three and for times, respectively. Although these differences in conversion rates were not so large relative to the costs, they were of great significance. In connection with this, we made the following conversion calculations. The overall evaluation of 90 and 95 percent appears to be a good evaluation result. However, by multiplying the overall evaluation rates by large negative numbers, we can change evaluation points substantially.

(1) $70+(30\times(-5))=-80$

(2) $80+(20\times(1-5))=-20$

(3) $90+(10\times(-5))=\ 40$

(4) $95+(5\times(-5))=\ 70$

(5) $99+(1\times(-5))=\ 94$

The above shows that tremendous costs (amount of work) are needed to correct wrong conversion results. Therefore, the above evaluation equation appears to assume reality.

In actual fact, development costs doubled when the conversion accuracy rate was raised from 90 to 99 percent. This is what this writer actually experienced around 1972 in developing a kana-Chinese character conversion system for names of persons.

The evaluation of a machine-readable dictionary differs according to whether a conversion accuracy rate is based on the examination of discrete words or of compound words, and also according to whether a discrete system (only form element analysis) or a system as a whole is evaluated.

An adequate evaluation method of a natural language processing system has yet to be evolved.

We should be strict in evaluating the present state of affairs, but optimistic about the future.

Furthermore, it will be necessary not only to turn our attention to the number of entries and coverage, but also to continue our efforts to upgrade the quality of entries and their contents.

8) Future tasks

In the above we have made a proposal concerning a policy on data to be added to dictionaries.

Together with an increase in the number of entries in a dictionary, the improvement of the quality of entries (expansion of contents) is desired. It is also desired to clarify the usage of words and word associations, and further expand contents concerning upper-ranking and lower-ranking words, antonyms, associated words, technical terms, idiomatic expressions, etc.

9) References

Yasuhito TANAKA : Maintenance of Machine-Readable Dictionary, 48th (for the first half of 1994) National Conference of the Information Processing Society, 3Q-1, March 23, 1994

Yasuhito TANAKA : Maintenance of Machine-Readable Dictionary (Part 2), 49th (for the first half of 1995) National Conference of the Information Processing Society, March 15, 1995·