

Recognizing Japanese numeral-classifier combinations

KEIICHIRO SUZUKI

.NET Speech Group, Microsoft Corporation

keisu@microsoft.com

1. Introduction

The main goal of this paper is to present a result of the study conducted to improve the performance of Large-Vocabulary Continuous Speech Recognition (LVCSR) by modeling context-dependent pronunciation variation (i.e. morphophonemic alternation) and context-independent pronunciation variation (i.e. free variation). In particular, I report the results of performance tests ran on numeral (数詞)-classifier (助数詞) combinations in Japanese (e.g. *ni-hon* ‘two stick-type object’, *san-bon* ‘three stick-type objects’), showing how the accuracy of our Japanese LVCSR engine was improved through modeling the context-dependent pronunciation variation and context-independent pronunciation variation. On one hand, these numeral-classifier combinations can be a typical subject of phonological/morphological study, displaying linguistically significant, “regular” morphophonemic voicing alternation patterns. On the other hand, the same set of data shows linguistically insignificant free variation involving voicing. I demonstrate that these two types of pronunciation variation are indeed captured by the same process of statistical adjustment in our LVCSR engine.

The secondary goal of this paper is to introduce a glimpse of research in the area of Automatic Speech Recognition (ASR) to the audience (i.e. phonologists) and to contribute to the knowledge transfer between the two disciplines. While much attention has been paid to the interdisciplinary study between phonology and cognitive science, not much attention has been generated between phonology and speech engineering. The practice of computational phonology (Bird 199x) does exist; however, it studies implementations of theoretical phonology, which is not the same as the research aimed at improving ASR systems.

Note that it is not the goal of this paper to offer some particular linguistic insight. Rather, this paper presents an alternative look at the Japanese numeral-classifier combinations, a typical subject for a phonological analysis, from the viewpoint of a commercial LVCSR.

This paper is organized as follows. In the remainder of this section, I provide a brief overview of our LVCSR. In Section 2, I describe what the problem was that we were trying to resolve. Next, I discuss the solution we took to resolve the problem. Section 4 gives our test results to verify that the solution we took actually worked. Finally, Section 5 concludes the paper.

1.1 Brief overview of LVCSR

LVCSR takes incoming acoustic stream as its input and outputs some text string that matches the input. Thus, the goal of ASR systems is to maximize the probability of a string that best matches an input acoustic stream. What is the most likely sentence out of all sentences in the language L given some acoustic input X ? This can be expressed as:

- (1)
 - a. Acoustic input = sequence of observations: $X = x_1, x_2, \dots x_t$
 - b. Output sentence = sequence of words: $W = w_1, w_2, \dots w_n$
 - c. The goal of ASR: $W = \mathbf{argmax} P(W | X)$

The formula in (1c) straightforwardly expresses the goal of ASR: to find the string of words W that maximizes $P(W | X)$. By applying Bayes' rule to (1c):

$$(2) \quad W = \operatorname{argmax} P(X | W) P(W) / P(X)$$

In (2), the divider, $P(X)$, is a non-factor, since the probability for the given acoustic input does not change for each potential output sentence. Removing the divider, the final formula is $P(X | W) * P(W)$, and this is THE golden formula for any ASR systems.

$$(3) \quad W = \operatorname{argmax} P(X | W) P(W)$$

The first part $P(X | W)$ is called Acoustic Model (AM) and the second part is called Language Model (LM).

AM, the $P(X | W)$ part of (3), is a collection of probabilistic sound sequences for a given word. In our LVCSR, AM is based on Hidden Markov Model (HMM) which gives transitions of observation sequence when there is no deterministic information about observed input is given (that is why the name Hidden). Most, if not all, state-of-the-art ASR systems employs some form of HMMs. Our AM treats each state in HMMs as the basic subphonetic unit called senone (Hwang and Huang 1993). Senones are the unit composing a triphone (context dependent phones), consisting of the left context, the phone, and the right context (e.g. /to/ consists of two triphones, <sil>-t+o and t-o+<sil> where <sil> is silence). The parameters (probability values) for our AM can be automatically estimated by going through hours of acoustic data. Once the AM is trained, the spectral features extracted from the acoustic input get computed and matched against the probable phone sequence for the candidate word.

LM, the $P(W)$ part of (3), is the model for determining the probability of a word sequence w_1, w_2, \dots, w_n , namely $P(w_1, w_2, \dots, w_n)$. This probability gets broken down into its component probabilities by Chain Rule:

$$(4) \quad P(w_1, w_2, \dots, w_i) = P(w_1) * P(w_2 | w_1) * \dots * P(w_i | w_1, w_2, \dots, w_{i-1}) \\ = \prod_{i=1}^n P(w_i | w_1^{i-1})$$

Since it may be difficult to compute a probability of the from $P(w_i | w_1, w_2, \dots, w_{i-1})$ even for moderate values of i , we typically assume that the probability of a word depends only on the previous N word(s). This leads to an n -gram language model.

$$(5) \quad P(w_1^n) \approx \prod_{i=1}^n P(w_i | w_{i-N+1}, w_{i-N+2}, \dots, w_{i-1})$$

If the probability of the word depends on the previous two words ($N=3$), we have a *trigram* (6a). Similarly, it is called *unigram* when $N=1$ (6c), *bigram* when $N=2$ (6c). The trigram language model is widely used in most commercial LVCSR systems today.

$$(6) \quad \text{a. Trigram: } P(w_1^n) \approx \prod_{i=1}^n P(w_i | w_{i-2} w_{i-1}) \\ \text{b. Unigram: } P(w_1^n) \approx \prod_{i=1}^n P(w_i)$$

$$\text{c. Bigram: } P(w_1^n) \approx \prod_{i=1}^n P(w_i | w_{i-1})$$

Applying this to Japanese, consider the sequence “学校に行く”. Since there is no white space to delimit words in Japanese texts, we take what we consider a morpheme as the base unit, such as 学校, に, 行, く. Using bigram, the probability for this sentence is calculated as:

$$(7) \quad P(\text{学校, に, 行, く}) = P(\text{学校} | \langle s \rangle) * P(\text{に} | \text{学校}) * P(\text{行} | \text{に}) * P(\text{く} | \text{行}) * P(\langle /s \rangle | \text{く})$$

where $\langle s \rangle$ and $\langle /s \rangle$ are the placeholders for “sentence initial” and “sentence final”.

The current practice is that we use large text corpora to calculate the n -gram probabilities. It follows that the larger the size of the text corpora, the better the n -gram coverage. Even with the large corpora, there will always be many word sequences, especially for trigrams, that get zero probability. There are various discounting and smoothing techniques to circumvent this data sparseness problem, but I will not discuss them here (See Huang, Acero, and Hon 2001 for more details).

The above is a quick introduction to ASR. Besides AM and LM, there are two more important pieces to the system: Front End and Decoder. I will not cover these topics, since these are not relevant to the main discussion of this paper. However, there are some good introductory books on ASR that I recommend to the interested readers (Jurafsky and Martin 2000, Huang, Acero, and Hon 2001, Shikano, et al. 2001).

2. The Problem

2.1 Japanese numeral-classifier combinations

Classifiers (助数詞) in Japanese attach to numerals (数詞) to express the type of object being counted, e.g. *ni-mai* (二枚) ‘two thin paper-like objects’, *san-mai* (三枚) ‘three thin paper-like objects’. Many numeral-classifier combinations are regular, in the sense that the pronunciation for the particular combination is just a concatenation of the pronunciations of individual parts. For example, we get *ni-mai* (二枚) ‘two thin paper-like objects’ by adding the pronunciation of the classifier part *mai* to the numeral part *ni*. However, majority of numeral-classifier combinations are irregular, in the sense that the pronunciation of the whole was not just an addition of the parts.

There are three notable characteristics about these irregular numeral-classifier combinations. First, some classifiers have multiple pronunciations, and the pronunciation of the particular classifier is determined by the preceding numeral. For example, the same classifier /hon/ (本) ‘stick-shape object’ is pronounced as *pon* in *iQ-pon* (一本) ‘one stick-shape object’, *hon* in *ni-hon* (二本) ‘two stick-shape objects’, and *bon* in *san-bon* (三本) ‘three stick-shape objects’. Second, some numerals themselves have multiple pronunciations, and the particular numeral’s pronunciation is dependent on the following classifier. For example, the same numeral /ichi/ (一) ‘one’ is pronounced as *ichi* in *ichi-ban* (一番) ‘the first’, *iQ* in *iQ-pon* (一本) ‘one stick-shape object’, and *hito* in *hito-tsuki* (一月) ‘one month’. Finally, some numeral-classifier combinations have multiple pronunciations that are in free variation. For example, for the term ‘one stock’ (一株) it is equally plausible to pronounce it as *ichi-kabu*, *iQ-kabu*, or *hito-kabu*.

Similarly for ‘eight cups (of)’ (八杯), it can be pronounced as *hachi-hai* or *haQ-pai*. The three characteristics of irregular combinations are summarized below.

- (8) **The characteristics about the irregular numeral-classifier combinations**
- a. For some classifiers the pronunciation varies depending on what the preceding numeral is.
 - b. For some numerals the pronunciation varies depending on what the following classifier is.
 - c. Some numeral-classifier combinations are in free variation.

2.2 The description of the problem

Our Japanese LVCSR engine was having trouble dealing with numeral-classifier combinations. The accuracy of our Japanese engine was noticeably lower when the dictated sentence included some numeral-classifier combination(s). This was initially my subjective judgment. However, the same judgment was shared by a few of my colleagues, and the case was strong enough for me to conduct a large scale study to tackle the issue.

The essence of the problem was the following. When we train our LM, billions of raw text data from various corpus are processed to 1) produce a list of words appear in the data to be in the master lexicon, and 2) produce *n*-gram counts by calculating the frequencies of each word sequences in the lexicon. We had a problem in each of the two points in the LM creation process. For 1), no checking was done on the lexicon to make sure that all pronunciation variants are listed in the lexicon. Thus, even though our lexicon contained the item /hon/ (本) with the pronunciation *hon*, it might have lacked its pronunciation variants such as *pon* and *bon*. For 2), regular variant may be incorrectly modeled even for the irregular cases. If the combination was an irregular one, the dependencies in (8a,b) must be resolved to get the correct pronunciation of the combination. This step was simply ignored. Moreover, even if we resolved the dependencies, the correct pronunciation may have its free variant. No mechanism existed to assign appropriate probabilities to these free variants. The three problem areas that we identified at two points during the LM training process were summarized below.

- (9) **The problem areas**
- a. Lexicon creation stage: no checking was done on the lexicon to make sure that all pronunciation variants are listed in the lexicon.
 - b. N-gram count file creation stage: no attention was paid to the pronunciation variation for numeral-classifier combinations.
 - c. N-gram count file creation stage: no mechanism existed to assign appropriate probabilities to the free variants.

In the next section, I discuss how we dealt with each of the above problem areas.

3. The Fix

3.1 Lexicon check

In order to make sure that all pronunciation variations are covered in the lexicon (9a), I went through the lexicon and checked to see if all pronunciation variants for a given numeral or a

given classifier are listed in the lexicon. This process was necessary, since the pronunciation for a numeral-classifier combination is not always *regular*. The actual method I used is the following.

(10) The method used for lexicon check

- a. Identified 65 representative (frequently used) classifiers.
- b. Produced a table that lists all pronunciation variations of the above 65 classifiers.
- c. Produced a table that lists all pronunciation variations of the numerals through 0-10.
- d. Went through the lexicon and added the entries if any of the pronunciation variants in the two tables were not listed in the original lexicon.

The 65 representative classifiers were selected from the base list created to cover most frequent classifiers. The complete phonological description for the numeral-classifier combinations in the base list is provided in the Appendix.

Note that there is an alternative solution in which we add individual numeral-classifier combinations as a lexical entry. Such a brute-force solution was avoided since we did not wish to include all sorts of numeral-classifier combinations to the probability distribution. The size of n -grams will unnecessarily increase if we added individual numeral-classifier combinations like *iQ-pon* (一本) ‘one stick-shape object’ *ni-hon* (二本) ‘two stick-shape objects’, *san-bon* (三本) ‘two stick-shape objects’, ... to the lexicon. Thus, we decided to leave the choice of the correct pronunciation variant for a given combination for n -gram.

Another possibility was to introduce an intermediate level rather than the golden formula in (3) (Cremelie and Martens 1995, 1997, 1999, Fukada et. al 1998, 1999)

(11) Intermediate level

$$W = \operatorname{argmax} P(X | V) * P(V | W) * P(W)$$

The goal now is to find the string of words W and the corresponding string of variants V that give the highest probability for the acoustic input. $P(V | W)$ determines the probability of the variants given the words, while $P(W)$ describes the probabilities of sequences of words. However, we did not take this route, since in this model the context-dependence of pronunciation variants is not modeled directly in the LM. As we saw in (8a,b), except for ones in free variation (8c), majority of irregular numeral-classifier combinations are context-dependent, and thus, it is better to model the variation directly in n -gram.

Exhaustively listing the pronunciation variants in the lexicon (9a) was a prerequisite to the next step – adjustment of probability for numeral-classifier combinations.

3.1 Explicit n -gram extrapolation

Adding the pronunciation variants to the lexicon does not provide “context” information about when the variant pronunciation would occur. It is necessary to calculate the n -grams with the variants, so that the specific numeral-classifier combination yields a particular pronunciation of the whole (9b).

In order to directly model the numeral-classifier combinations in LM, we used *explicit n -gram extrapolation*. That is, we manually increased the n -gram count to cover unseen data in the corpus. Since some numeral-classifier combinations were not seen from our corpus, the n -gram would get under-trained. The sparsity of data is a common problem during the training of LMs,

and individual combinations of numerals and classifiers would have a probability that is too low to factor in LMs. Thus, we first divided the 65 classifiers into three *tiers* based on their frequency of occurrence in our corpora: Tier 1 classifiers included /hon/ 本 ‘stick-shape object’, /en/ 円 ‘yen’, etc., Tier 2 classifiers included /hyoo/ 票 ‘number of votes’, /shoo/ 勝 ‘number of win’, etc., and Tier 3 classifiers included /seki/ 隻 ‘number of ships’, /kumi/ 組 ‘group of’ etc. Then, during the training, we explicitly added count for the numeral-classifier sequences that have relatively lower frequency count within the tier. This resulted in the equal distribution for the numeral-classifier combinations within the same tier. For example, if there are 1,000,000 occurrences of the numeral-yen sequence (i.e. (n 円) ‘ n yen’ where n is 0-10), all the tier 1 classifiers will have the count of 1,000,000 for its numeral-classifier counts.

In addition to the probability adjustment for the numeral-classifier combinations within a tier, we also smoothed the probability distribution for the different numerals (0-10) for a given classifier. Most, if not all, classifiers have significantly larger count for its combination with /ichi/ (一) ‘one’ compared to the other numerals. This would cause the pronunciation variant for one-classifier (e.g. *pai* in *iQ-pai* (一杯) ‘one cup of’) to be too strong to win out for the other numeral-classifier combinations (e.g. **ni-pai* (二杯) ‘two cups of’ instead of the correct *ni-hai*). Thus, by taking the count for one-classifier combination for the particular classifier to be the base for the rest of numbers (0-10 except 1), we explicitly added *ni-hai* (二杯) ‘two cups of’, *san-bai* (三杯) ‘three cups of’, ... for each occurrence of *iQ-pai* (一杯) ‘one cup of’ in the data.

Some classifiers do not take zero as its numeral (**zero-choome* (0 丁目) ‘zero street address (?)’) or take it with extremely low probability (*?zero-hai* (0 杯) ‘zero cup (?)’), so zero was discounted from the count of numeral-classifier combinations for these particular classifiers.

Incorporating the explicit n -gram extrapolation, it was possible to directly model the pronunciation variants for both numerals and classifiers in the n -gram, thereby resolving the problem in (9b).

3.1 Explicit n -gram extrapolation for free variation

One final issue to resolve was (9c), modeling of free variants. Here again, we used explicit n -gram extrapolation. The assumption here is that the probability between the free variants is unpredictable. Based on this assumption, we assigned equal frequency count to all the free variants of a particular numeral for a given classifier. For example, we gave the equal distribution to each of the free variants, *ichi-kabu*, *iQ-kabu*, and *hito-kabu* for (一株) ‘one stock’. This makes possible to model the free variation directly in n -gram, making these variants of a numeral equally probable for a given classifier. After n -gram counts are manually adjusted, we applied smoothing and adjusted the backoff weighting to minimize the side effects (see Huang, Acero, and Hon 2001 for more details on smoothing and backoff).

Having rebuilt our LM using the explicit n -gram extrapolation, we tested the performance of our Japanese LVCSR to measure the improvement. In the next section, I discuss the test procedure and the results.

4. The Test

4.1 Test Procedure

The test was conducted with data consisting of sentences with 65 representative classifiers and was run against our daily produced builds of the Japanese LVCSR engine. The 65 representative

classifiers were divided into two groups: 12 *h*-initial classifiers and 53 non *h*-initial ones. The reason was that *h*-initial classifiers show most variability and the performance against the *h*-initial classifiers was identified as particularly critical to the overall performance. For example, *h*-initial set would contain sentences like ‘I had three cups of coffee this morning’ that contains the numeral-classifier combination *san-bai* ‘three cups of’ where the otherwise *h*-initial classifier *hai* appears as *bai* after *san*. The test sets contained the total of 65 base sentences, and the numbers from 1-10 are substituted to produce the total of 650 (65*10) sentences: 120 for *h*-initial test set (12*10) and 530 for non *h*-initial test set (53*10).

Then, 16kHz recordings of all 650 sentences were collected for 6 speakers (3 male and 3 female). The two wave files for each speaker were created: one for *h*-initial set, the other for non *h*-initial set. These wave files were fed to our accuracy test tool which spits out the recognition accuracy results for the given build of the engine. Both the *h*-initial and the non *h*-initial sets were tested with SI (Speaker Independent) mode.

4.2 Results

The results of the accuracy tests from builds 1 through 8 (the build numbers do not represent the actual build numbers we used) were recorded. The result consists of the following two numbers per build: Word Error Rate (WER) and Numeral-Classifier Combination Error Rate (NCCER).

WER is based on how much the output string returned by the recognizer differs from the correct string for a given test set. WER is calculated as $100 * (\#Insertions + \#Deletions + \#Substitutions) / (\#Words)$. For example, for the correct string “I had three cups of coffee this morning” consisting of 8 words, if the hypothetical output of the recognizer was “I hid three cups of cold feet morning” (“hid” is substituted for “had”, “cold” is inserted, “feet” is substituted for “coffee”, “this” is deleted) then the WER is $100 * (1+1+2)/8 = 50\%$.

NCCER is based on how much the output string returned by the recognizer differs from the correct string only for the numeral-classifier combinations for a given test set. For each numeral-classifier combination, I gave the value “1” if the output contained the correct numeral-classifier combination; otherwise, I gave “0”. Note that the insertion errors were not counted for NCCER. As long as the output contained the correct string, it was counted as correct. Thus, the formula for NCCER is $100 * (\#correct\ numeral\ classifier\ combinations) / (\#numeral\ classifier\ combinations)$. For example, the previous hypothetical output “I hid three cups of cold feet morning” would yield the NCCER of $100 * 1/1 = 100\%$. The test sets contain the total of 65 (12 for *h*-initial, 53 for non *h*-initial), so the maximum number of correct numeral-classifier combinations is 65 for the two sets.

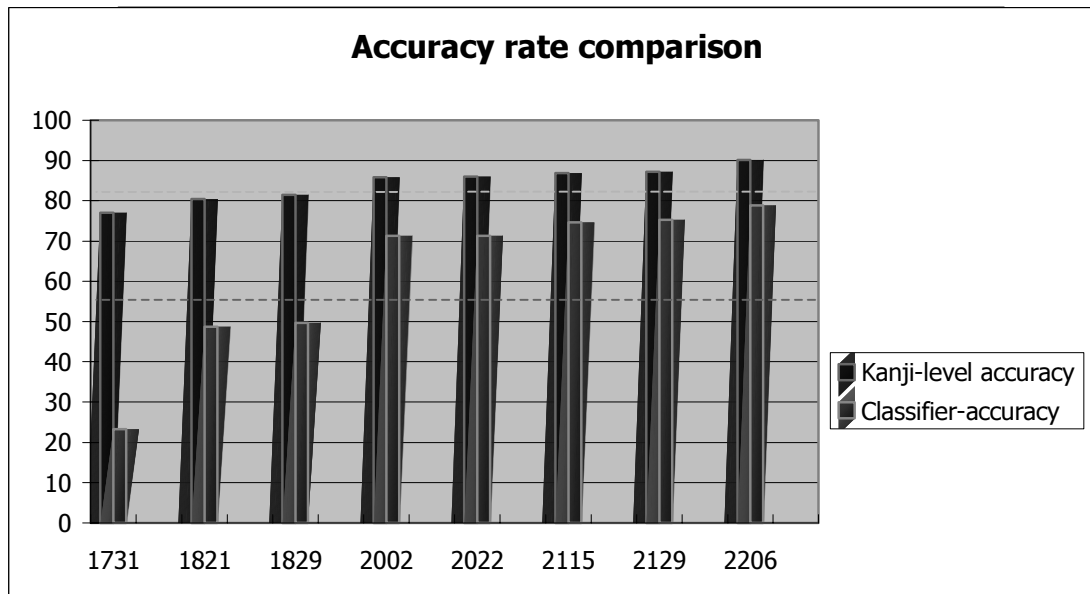
The table of the overall test results is shown below in (12). The numbers for the accuracy rate for each test set (*h*-initial and non *h*-initial) against each build (1-4) are included (rounded for readability). The average of *h*-initial and non *h*-initial numbers are given in the bottom two rows (bolded). Our Build 1 engine did not have any fix for the numeral-classifier problem. Build 2 and 3 engines incorporated partial fixes for the problem with extending the coverage of numeral-classifier combinations. Our Build 4 engine implements the LM that had the full coverage of the targeted 65 classifiers and their pronunciation variants along with the pronunciation variants for the numerals. The baseline data here was obtained by running the identical test sets with a commercial 3rd party Japanese LVCSR engine.

(12) Overall test results

	Build 1		Build 2		Build 3		Build 4		Baseline	
	<i>h</i> -ini	non	<i>h</i> -ini	non	<i>h</i> -ini	non	<i>h</i> -ini	non	<i>h</i> -ini	rest
WER	77	77	80	81	82	90	88	92	80	84
NCCER	23	24	59	38	67	76	76	82	47	64
	Avr.		Avr.		Avr.		Avr.		Avr.	
WER	77.0		80.4		85.8		90.2		81.8	
NCCER	23.3		48.7		71.3		78.9		55.2	

The graph in (13) below shows the improvement more clearly. The upper dotted line represents the WER while the lower dotted line indicates the NCCER of the baseline. As obvious from the graph, our Build 1 engine was performing very poorly, getting lower accuracy rates for both WER and NCCER than the baseline. As we incorporate the fix progressively build by build, gradually completing the adjustment of the frequency counts of numeral-classifier combinations, it is evident that the performance of our engine for both the WER and the NCCER improved dramatically. By Build 3, our engine outperformed the baseline engine for both WER and NCCER. At Build 4, as our implementation of the probability adjustment for the numeral-classifier combinations was completed, we obtained the best results. Note that the improvement of NCCER did not hinder the WER but helped the WER improvements.

(13) Accuracy rate progression against the baseline



4.3 Summary

The test results proved that it was possible to significantly improve the performance of Japanese LVCSR engine against the pronunciation variability of numeral-classifier combinations by making probability adjustment using the explicit *n*-gram extrapolation. Three problem areas identified earlier in (9) were resolved by 1) exhaustive listing of pronunciation variants for numerals as well as for classifiers in the lexicon, and by 2) manually adjusting the counts of numeral-classifier combinations to model in the *n*-grams. Not only did the explicit *n*-gram extrapolation resolve the context-dependent pronunciation variation, but it also resolved the issue

with free variation as well. The test result in (13) shows that as we increase the coverage of numeral-classifier combination

One of the things we did not cover with this study was the testing for zero-classifier cases. As I stated earlier, not all classifier can take zero as its numeral. We need to test if we modeled these exceptional cases appropriately. Another possibility is to test with numerals that are larger than 10 and see if there were any side effects. Increasing the frequency counts of numeral-classifier combinations for the cases where the numeral is 0-10 may have affected the other cases. It will be necessary to increase the test coverage before the engine actually gets productized. Expanding the test cases as well as seeking other ways of improving the performance of our Japanese LVCSR engine are the subjects for my further study.

5. Conclusion

In this paper, I have presented the result of the study, describing how the performance of our Japanese LVCSR engine against numeral-classifier combinations was improved. I have demonstrated that the context-dependent pronunciation variation and the context-independent pronunciation variation are handled by the same mechanism, explicit n -gram extrapolation. These two are not different species from the perspective of an LVCSR. This is very different from linguistic point of view where morphophonemic alternations are considered linguistically significant while free variations are considered insignificant.

I have also introduced, at least minimally, the domain of ASR to the primarily linguistic audience. The research on modeling pronunciation variation for ASR systems has increased lately, and some of the new ideas were actually inspired from theoretical phonology (see Strik and Cucchiarini 1999). I believe opportunities are there for phonologists to make contributions to the ASR research.

As the ASR researchers get ideas from linguistic studies, I also believe that phonologists will benefit from studying the research in ASR systems. There have been some attentions to stochastic models of phonology (Anttila 1995, Frisch 1996, Boersma 1997, Coleman and Pierrehumbert 1997, etc.). The fully stochastic nature of the techniques used in the current ASR systems may be worthy for a phonologist to explore.

References

- Anttila, A. (1995). *Deriving variation from grammar: A study of Finnish Genitives*, ROA.
- Boersma, P. (1997). *How we learn variation, optionality, and probability*. ROA.
- Coleman, J. and J. Pierrehumbert (1997). "Stochastic phonological grammars and acceptability", In *Computational Phonology: Third meeting of the ACL special interest group in computational phonology*, pages 49-56. Association for Computational Linguistics, Somerset.
- Frisch, S. (1996). *Similarity and frequency in phonology*. Ph.D. thesis, Northwestern University.
- Huang, X.D., A. Acero, and H.W. Hon (2001) *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall PTR, Upper Saddle River, NJ.
- Hwang, M.Y. and X.D. Huang (1993) "Shared-Distribution Hidden Markov Models for Speech Recognition", *IEEE Trans. on Speech and Audio Processing*, 1993, 1 (4): 414-420.

- Jurafsky, D. and J.H. Martin (2000) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, Upper Saddle River, NJ.
- Cremelie, N. and J.P. Martens (1995) "On the use of pronunciation rules for improved word recognition", In *Proceedings Eurospeech '95*, 1747-1750.
- Cremelie, N. and J.P. Martens (1997) "Automatic rule-based generation of word pronunciation networks", In *Proceedings Eurospeech '97*, 2459-2462.
- Cremelie, N. and J.P. Martens (1999) "In search of better pronunciation models for speech recognition", *Speech Communication* 29: 225-246.
- Fukada, T., T. Yoshimura, and Y. Sagisaka (1998) "Automatic generation of multiple pronunciations based on neural networks and language statistics", In Strik, H., J.M. Kessens, and M. Wester (Eds) *Proceedings of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, Rolduc, Kerkrade: 41-46.
- Fukada, T., T. Yoshimura, and Y. Sagisaka (1999) "Automatic generation of multiple pronunciations based on neural networks", *Speech Communication* 27 (1): 63-73.
- Strik, H. and C. Cucchiariini (1999) "Modeling pronunciation variation for ASR: A survey of the literature", *Speech Communication* 29: 225-246.

Appendix: Phonological description of numeral-classifier combinations

Below I provide a table that provide phonological description of numeral-classifier combinations for the most frequently used classifiers. Numerals include 1, 2, 3, 4, 6, 8, and 10, since the other numerals are either have one form or the familiar variants (e.g. *nana* or *shichi* for ‘seven’) are in free variation. A “*” in *R* column indicates the item is selected as representative and used for the test. I also used symbols listed below to indicate particular phonological pattern when the numeral and the classifier concatenates. Finally, I gave each pattern a name to categorize based on the patterning. For example, if the classifier is *k*-initial, I gave the name *k1*, *k2*, and so on. I labeled as “irregular” for classifiers with mostly irregular patterning.

Symbols describing the phonological pattern

- Q:** causes the last consonant of the number word change to a glottal stop
N: causes no change
-: free variation
vl: causes the initial consonant of the classifier to be voiceless (e.g. [h] to [p])
vd: causes the initial consonant of the classifier to be voiced (e.g. [h] to [b])
HITO: causes “1” be pronounced as [hito], instead of [ichi]
HUTA: causes “2” be pronounced as [huta], instead of [ni]
MI: causes “3” be pronounced as [mi], instead of [san]
YO: causes “4” be pronounced as [yo], instead of [yon]
SHI: causes “4” be pronounced as [shi], instead of [yon]
MU: causes “6” be pronounced as [mu], instead of [roku]

1. *k*-initial (か行) classifiers

Done	Writing	Reading	1	2	3	4	6	8	10	Pattern
	金	kin	N	N	N	N	N	N	Q	0
*	口	kuchi	HITO	HUTA	N MI	N YO	Q	Q	Q	Irregular
*	組	kumi	N HITO	N HUTA	N	N	Q	N Q	Q	Irregular
*	桁	keta	N HITO	N HUTA	N MI	N	Q	N Q	Q	Irregular
*	回	kai	Q	N	N	N	Q	N Q	Q	k1
*	階	kai	Q	N	N	N	Q	N Q	Q	k1
*	海里 湮	kairi	N	N	N	N	Q	N Q	Q	k1
	画	kaku	Q	N	N	N	Q	N Q	Q	k1
*	箇月 個月 箇月間	Kagetsu(kan)	Q	N	N	N	Q	N Q	Q	k1
*	箇所 個所	kasho	Q	N	N	N	Q	N Q	Q	k1
	箇条 個条	kajoo	Q	N	N	N	Q	N Q	Q	k1
	刊	kan	Q	N	N	N	Q	N Q	Q	k1
*	卷	kan	Q	N	N	N	Q	N Q	Q	k1

	貫	kan	Q	N	N	N	Q	N	Q	Q	k1		
	間	kan	Q	N	N	N	Q	N	Q	Q	k1		
	管区	kanku	Q	N	N	N	Q	N	Q	Q	k1		
*	期	ki	Q	N	N	N	Q	N	Q	Q	k1		
*	機	ki	Q	N	N	N	Q	N	Q	Q	k1		
*	基	ki	Q	N	N	N	Q	N	Q	Q	k1		
	級	kyuu	Q	N	N	N	Q	N	Q	Q	k1		
	球	kyuu	Q	N	N	N	Q	N	Q	Q	k1		
	強	kyoo	Q	N	N	N	Q	N	Q	Q	k1		
	教科	kyooka	Q	N	N	N	Q	N	Q	Q	k1		
	局	kyoku	Q	N	N	N	Q	N	Q	Q	k1		
	極	kyoku	Q	N	N	N	Q	N	Q	Q	k1		
*	曲	kyoku	Q	N	N	N	Q	N	Q	Q	k1		
	斤	kin	Q	N	N	N	Q	N	Q	Q	k1		
	区	ku	Q	N	N	N	Q	N	Q	Q	k1		
	句	ku	Q	N	N	N	Q	N	Q	Q	k1		
*	軒	ken	Q	N	N	vd	N	Q	N	Q	Q	k1	
	件	ken	Q	N	N	N	Q	N	Q	Q	k1		
*	個	ko	Q	N	N	N	Q	N	Q	Q	k1		
	戸	ko	Q	N	N	N	Q	N	Q	Q	k1		
	校	koo	Q	N	N	N	Q	N	Q	Q	k1		
	項	koo	Q	N	N	N	Q	N	Q	Q	k1		
	光年	koonen	Q	N	N	N	Q	N	Q	Q	k1		
	国	koku	Q	N	N	vd	N	Q	N	Q	Q	k1	
	角形	kaQkee	N/A	N/A	N	SHI	Q	N	Q	Q	k1 & SHI		
	課	ka	N	Q	N	N	N	Q	N	Q	Q	k2	
	科	ka	N	Q	N	N	N	Q	N	Q	Q	k2	
*	株	kabu	N	Q	HITO	N	N	N	Q	N	Q	Q	k2
*	科目	kamoku	N	Q	N	N	N	Q	N	Q	Q	k2	
*	Cal C a l c a l 架	karorii	N	N	N	N	Q	N	Q	Q	k3		
*	K K キ k k	kiro	N	N	N	N	Q	N	Q	Q	k3		
*	k g kg kg	kiroguramu	N	N	N	N	Q	N	Q	Q	k3		
	KB	kirobaito	N	N	N	N	Q	N	Q	Q	k3		

*	km km k m	kiromeetoru	N	N	N	N	Q	N	Q	Q	k3
	kW kw k w k W	kirowatto	N	N	N	N	Q	N	Q	Q	k3

2. *s*-initial (さ行) classifiers

Done	Writing	Reading	1	2	3	4	6	8	10	Pattern
*	才	sai	Q	N	N	N	N	Q	Q	s1
*	歳	sai	Q	N	N	N	N	Q	Q	s1
*	作	saku	Q	N	N	N	N	Q	Q	s1
*	冊	satsu	Q	N	N	N	N	Q	Q	s1
	誌	shi	Q	N	N	N	N	Q	Q	s1
	紙	shi	Q	N	N	N	N	Q	Q	s1
	式	shiki	Q	N	N	N	N	Q	Q	s1
	室	shitsu	Q	N	N	N	N	Q	Q	s1
	品	shina	Q	N	N	N	N	Q	Q	s1
*	社	sha	Q	N	N	N	N	Q	Q	s1
	者	sha	Q	N	N	N	N	Q	Q	s1
	種	shu	Q	N	N	N	N	Q	Q	s1
*	周	shuu	Q	N	N	N	N	Q	Q	s1
*	週	shuu	Q	N	N	N	N	Q	Q	s1
*	勝	shoo	Q	N	N	N	N	Q	Q	s1
*	章	shoo	Q	N	N	N	N	Q	Q	s1
	進	shin	Q	N	N	N	N	Q	Q	s1
	寸	sun	Q	N	N	N	N	Q	Q	s1
	世	see	Q	N	N	N	N	Q	Q	s1
*	世紀	seeki	Q	N	N	N	N	Q	Q	s1
*	隻	seki	Q	N	N	N	N	Q	Q	s1
	石	seki	Q	N	N	N	N	Q	Q	s1
	世代	sedai	Q	N	N	N	N	Q	Q	s1
	節	setsu	Q	N	N	N	N	Q	Q	s1
	錢	sen	Q	N	N	N	N	Q	Q	s1
*	戦	sen	Q	N	N	N	N	Q	Q	s1
	千石	sengoku	Q	N	N	N	N	Q	Q	s1
*	センチ cm c m cm	senchi	Q	N	N	N	N	Q	Q	s1

	層	soo	Q	N	N	N	N	Q	Q	s1
	艘	soo	Q	N	N	N	N	Q	Q	s1
	足	soku	Q	N	vd	N	N	Q	Q	s1
	速	soku	Q	N	N	N	N	Q	Q	s1
*	試合	shiai	N	Q	N	N	N	Q	Q	s2

3. *t*-initial (た行) classifiers

Done	Writing	Reading	1	2	3	4	6	8	10	Pattern		
	T T TB	Tera(baito)	N	N	N	N	N	N	N	0		
*	束	taba	HITO	HUTA	N	N	YO	N	N	Q	Irregular	
*	月	tsuki	HITO	HUTA	MI	YO	MU	Q	Q	Irregular		
*	坪	tsubo	HITO	Q	N	HUTA	N	N	Q	Q	Irregular	
*	通り	toori	HITO	Q	N	HUTA	N	MI	N	Q	Q	Irregular
*	着	chaku	Q	N	N	N	N	Q	Q	t1		
	昼夜	chuuya	Q	N	N	N	N	Q	Q	t1		
*	丁 丁目	choo(me)	Q	N	N	N	N	Q	Q	t1		
	対	tsui	Q	N	N	N	N	Q	Q	t1		
*	通	tsuu	Q	N	N	N	N	Q	Q	t1		
	手	te	Q	N	N	N	N	Q	Q	t1		
*	滴	teki	Q	N	N	N	N	Q	Q	t1		
*	店	ten	Q	N	N	N	N	Q	Q	t1		
*	点	ten	Q	N	N	N	N	Q	Q	t1		
	棟	too	Q	N	N	N	N	Q	Q	t1		
	頭	too	Q	N	N	N	N	Q	Q	t1		
	党	too	Q	N	N	N	N	Q	Q	t1		
*	等	too	Q	N	N	N	N	Q	Q	t1		
	体	tai	N	Q	N	N	N	N	Q	Q	t2	
*	対	tai	N	Q	N	N	N	N	Q	Q	t2	
	反	tan	N	Q	N	N	N	N	Q	Q	t2	
*	t t ト	ton	Q	N	N	N	N	N	Q	Q	t2	

4. *n,m,r,w*-initial (なまらわ行) classifiers

Done	Writing	Reading	1	2	3	4	6	8	10	Pattern
*	年	nen	N	N	N	YO	N	N	N	<i>Irregular</i>
*	人	nin	HITO	N HUTA	N	N	N	N	N	<i>Irregular</i>
*	坪	tsubo	HITO	Q N HUTA	N	N	N	Q	Q	<i>Irregular</i>
*	通り	toori	HITO	Q N HUTA	N MI	N	N	N Q	Q	<i>Irregular</i>
*	枚	mai	N	N	N	N	N	N	N	<i>m1</i>
*	万	man	N	N	N	N	N	N	N	<i>m1</i>
*	面	men	N	N	N	N	N	N	N	<i>m1</i>
*	列	retsu	N	N	N	N	N	N	N	<i>r1</i>
*	厘	rin	N	N	N	N	N	N	N	<i>r1</i>
*	話	wa	N	N	N	N	N	N	N	<i>w1</i>
*	羽	wa	N	N	vl vd	N	N Q	N Q	Q	<i>Irregular</i>

5. *h*-initial (は行) classifiers

Done	Writing	Reading	1	2	3	4	6	8	10	Pattern
*	敗	hai	Q-vl	N	vl	N	Q-vl N	Q-vl N	Q-vl	<i>h1</i>
*	泊	haku	Q-vl	N	vl	N	Q-vl N	Q-vl N	Q-vl	<i>h1</i>
*	発	hatsu	Q-vl	N	vl	N	Q-vl	Q-vl	Q-vl	<i>h2</i>
*	班	han	Q-vl	N	vl	N	Q-vl	Q-vl	Q-vl	<i>h2</i>
*	品	hin	Q-vl	N	vl	N	Q-vl	Q-vl N	Q-vl	<i>Irregular</i>
*	分	hun	Q-vl	N	vl	N vl	Q-vl	Q-vl N	Q-vl	<i>Irregular</i>
	編	hen	Q-vl	N	vl	N	Q-vl N	Q-vl N	Q-vl	<i>h1</i>
*	篇	hen	Q-vl	N	vl	N	Q-vl N	Q-vl N	Q-vl	<i>h1</i>
	hPa	hektopasukaru	N	N	N	N	N	N	N	<i>0</i>
	H z	herutsu	N	N	N	N	N	N	N	<i>0</i>
*	箱	hako	HITO	Q N HUTA	vl N	N	Q-vl N	Q-vl	Q-vl	<i>Irregular</i>
*	杯	hai	Q-vl	N	vd	N	Q-vl N	Q-vl N	Q-vl	<i>vd</i>
*	匹	hiki	Q-vl	N	vd	N	Q-vl	Q-vl	Q-vl	<i>vd</i>
*	本	hon	Q-vl	N	vd	N	Q-vl	Q-vl N	Q-vl	<i>vd</i>
	判	han	Q-vl	N	vl vd	N vd	Q-vl	Q-vl	Q-vl	<i>vl/vd</i>
*	版	han	Q-vl	N	vl vd	N vd	Q-vl	Q-vl	Q-vl	<i>vl/vd</i>
*	歩	ho	Q-vl	N	vl vd	N vl	Q-vl	Q-vl N	Q-vl	<i>vl/vd</i>
*	票	hyoo	Q-vl	N	vl vd N	N vl	Q-vl N	Q-vl N	Q-vl	<i>vl/vd/N</i>