3-6-2007

Statistical Machine Translation of Japanese

Erik A. Chapla

STATISTICAL MACHINE TRANSLATION OF JAPANESE

THESIS

Erik A. Chapla

AFIT/GE/ENG/07-06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/07-06

STATISTICAL MACHINE TRANSLATION OF JAPANESE

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Erik A. Chapla, B.S.E.E.

March 2007

AFIT/GE/ENG/07-06

STATISTICAL MACHINE TRANSLATION
OF JAPANESE

Erik A. Chapla

Approved:

___/Signed/___	6 March 2007
Dr. Steven Gustafson (Chairman)	date
___/Signed/___	6 March 2007
Dr. Timothy Anderson (Member)	date
___/Signed/___	6 March 2007
Dr. Raymond Slyh (Member)	date
___/Signed/___	6 March 2007
Dr. Richard Martin (Member)	date

AFIT/GE/ENG/07-06

Abstract

Statistical machine translation (SMT) uses large amounts of language training data to statistically build a knowledge base for translating from one language to another. Before introducing this language data, usually in the form of a parallel set of sentences from both languages, the SMT system has no other linguistic information available to it. With supervised SMT, however, additional linguistic knowledge is allowed in addition to the training data. When translating between languages with little or no common linguistic backgrounds, like English and Japanese, using supervised SMT is extremely useful. By giving the system linguistic rules before training on the parallel corpus, the SMT system can build better alignments between words in both languages.

This thesis investigates different ways of augmenting the training data to find the best possible alignments between Japanese and English texts from a large travel domain corpus to yield the highest numeric scores for accurate translation from Japanese into English. Results show that altering the topic and locative particles and altering tense, politeness levels, and verb endings in the training data result in the best score improvement.

In addition to SMT experiments, automatic speech recognition (ASR) and text segmentation experiments are performed. The ASR experiments yield promising results, but they did not improve on the error rates of the default experiments. The text segmentation experiments show that by using a combination of a 2-gram and 3-gram windows, segments in Japanese text are correctly placed 94.4% of the time while adding incorrect segments to 3.2% of possible locations.

Acknowledgments

I would like to thank Drs. Anderson and Slyh for their guidance in choosing a topic of research, helpful comments, and their encouragement and support when needed. I would also like to express my appreciation to Dr. Gustafson for acting as my thesis advisor and for his assistance in focusing my research and help in compiling my thesis. I wish to also thank Dr. Martin for his time in reviewing my thesis and acting as my academic advisor while at AFIT.

In addition, I would like to thank Mr. Hoeferlin for always providing help with computers and software when performing experiments in the lab.

Erik A. Chapla

Table of Contents

List of Figures

List of Tables

Table Page

List of Abbreviations

STATISTICAL MACHINE TRANSLATION OF JAPANESE

I. Introduction

One of the more difficult tasks in natural language processing is machine translation (MT). Despite considerable research over the years, MT systems have yet to reach human levels of performance. Before modern computers, lack of computational power meant that computers were limited in their ability to handle large amounts of linguistic data. Thus early efforts at MT focused on less computational methods of translation, such as simple word-for-word or phrase-for-phrase substitutions between source and target languages. Often the basic idea of a text could be found by using this method when languages were from similar language families. However, in the case of languages that have no common linguistic background, like Japanese and English, simple word-for-word or phrase-for-phrase substitution often does not work well. As computational power grew, researchers built rule-based systems such as SYSTRAN [28]. However, these rule-based systems are difficult to build and expand, leading to long development times and large cost. In addition, they tend not to be robust.

Today, considerable research effort in MT is focused on statistical machine translation (SMT) goes beyond these early methods by creating translations using statistical methods and parallel text corpora. The main idea behind SMT is to train a system by aligning large amounts of source language text data with target language text data, thereby educating an otherwise ignorant system as to what words in the target language are most likely to occur given particular words in the source text. A new set of source text is then input to the trained system, and, applying statistical rules learned during training, target text translation is output by the system. The quality of the translation is directly affected by the size of the original training corpus. The more often the system encounters a particular word during training, the more likely it is to understand the context in which the corresponding target word or words belong. In addition, if very large amounts of training data are used, very sparse words and even unknown words during testing become less of a problem.

In this thesis, research efforts are made at improving Japanese-to-English SMT using linguistic techniques. It is found that the performance improved most when topic and locative particles in Japanese are removed from the 2005 IWSLT (International Workshop on Spoken Language Translation) training and test data. Other improvements come from altering verbs in terms of their honorific levels and inflections. The results here are competitive with the top 2005 IWSLT participants, including Japanese participants ATR and University of Tokyo. Word segmentation techniques are also examined. These techniques are important since word segmentation can be useful in building a Japanese language model for Japanese automatic speech recognition (ASR) and in English-to-Japanese MT. These experiments show that by applying a 2-gram followed by 3-gram n-gram analysis to unsegmented Japanese text, spaces are correctly placed between words 94.4% of the time. In addition, ASR experiments are performed on Japanese speech. The most successful experiment involve replacing the fricative /hu/ phoneme sound with /f/.

This thesis is organized as follows. The next chapter provides background on SMT. Chapter 3 provides background on the Japanese language, while Chapter 4 discusses Japanese segmentation experiments and their effects on SMT performance.

Chapter 5 discusses Japanese speech recognition and the results of some experiments designed to improve performance of the ASR system. Finally, Chapter 6 presents the conclusions and future work.

II. Statistical Machine Translation Background

This chapter addresses some of the basic issues behind SMT, including language models, translation models, smoothing techniques, and decoding. Many of the examples found here are based on translation from Japanese text to English text. For more information on the Japanese written language and grammar, see Chapter 3.

2.1 Statistical Machine Translation

Statistical machine translation requires large amounts of language data from both source and target languages to properly train the translation system. In the early nineties large bilingual corpora became available for research, thus making SMT a viable research field.

All SMT is based on Bayes' rule,

$$P(E \mid J) = \frac{P(J \mid E) \cdot P(E)}{P(J)}, \qquad (1)$$

where J (Japanese) and E (English) are translation pairs of source and target languages, respectively. P(J) is the probability of encountering the Japanese word sequence J, and P(E) is the a priori probability of a given English word sequence, E. P(E) is known as the language model probability. The translation model, P(J|E), is the probability that the Japanese word sequence J is the translation of the English work sequence E.

The goal of an SMT system is to find the best estimation of the translation of a Japanese sentence into English, P(E|J), that maximizes the product of the two probabilities of the language model and the translation model. Given a Japanese word sequence, J, P(J) is constant for all English translations under consideration, so the estimate of the translated English string simplifies to:

$$E_{est} = \arg\max_E \frac{P(J \mid E) \cdot P(E)}{P(J)} = \arg\max_E P(J \mid E) \cdot P(E). \qquad (2)$$

A basic model of an SMT system is shown in Figure 2.1. A typical SMT system must have a language model, translation model, and some decoding algorithm.

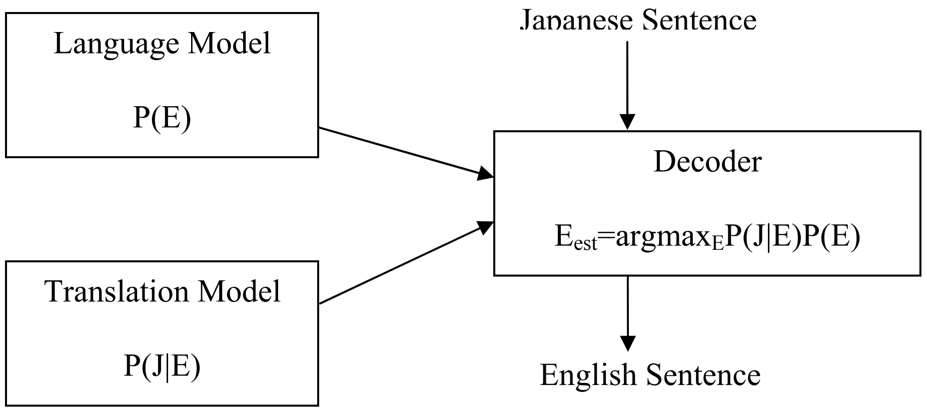

Figure 2.1 Model for a typical statistical machine translation system

2.2 Language Model

The language model is the probability that a phrase will occur in the target language. In our case, P(E) is the probability that an English string E has occurred in the training sentences. If the sentence “I have reservations” occurs 20 times in a database of 20000 sentences, P(E) = P(“I have reservations”) = 0.001. On the other hand, the sentence “Reservations have I” would rarely occur and would have a significantly lower probability. Since P(E|J) is the product of P(J|E) and P(E), a language model with very low probability like this second example would result in a very low P(E|J) probability. Therefore, when computing P(E|J), P(E) acts to filter out poorly constructed strings from the translation model by multiplying grammatically poor word sequences by a low probability and rewarding well-constructed sequences with a higher probability.

Finding the probability of some English string E is not a simple task. Since E is just a string of words, what is needed is the probability of that string occurring in English. In addition, since the probability of each word being in the string depends on preceding words, the probability of a string of words E is a product of multiple conditional probabilities:

$$P(E)=P(w_1\ w_2\ w_3\ \ldots w_{n-1}w_n)=P(w_1)P(w_2|\ w_1)P(w_3|w_1w_2)\ldots P(w_n|\ w_1w_2\ w_3\ldots\ w_{n-1}).\ (3)$$

One major problem in SMT is finding the language model probability when a word in a phrase does not occur in the training data. When absence happens, P(E) is assigned a zero probability, and no matter how well aligned the English and Japanese phrases are, P(E|J) is also be zero. For example, in Figure 2.2 the first sentence is found in the training data, but the second, while perfectly reasonable from a language standpoint, is not.

Accomplishing such a task seems impossible since, regardless of the size of the corpus, all possible strings will certainly not appear. Since this task is fruitless, the next best thing is to approximate the probability of such a string using the various probabilities in the corpus. This approximation is done using n-grams.

P(E) = 0.0002 0.05 0.01 0.03 0.01 0.1 > 0

イタリア の 食べ物 が 好き です。

Italy (possessive) food (subject) like(d) is.

"Food of Italy is liked."(lit) or "I like Italian food."

P(E) = **0** 0.05 0.01 0.03 0.01 0.1 = 0 = P(E|J)

スペイン の 食べ物 が 好き です。

Spain (poss.) food (sub.) like(d) is.

"Food of Spain is liked."(lit) or "I like Spanish food."

Figure 2.2. Example of how a single word with zero probability (since it never appears in the training sentences) can affect P(E|J) despite being an otherwise grammatically well-constructed sentence. Here, since Spain is never seen in the training data, P(Spain) = 0.

An n-gram is a sequence of n words. In the phrase "I would like to make reservations," unigrams (n=1) are each single word in the phrase, bigrams (n=2) are strings like "I would," "would like," "like to," etc., and trigrams (n=3) have longer strings "I would like," "would like to," "like to make," and "to make reservations." The probability of a bigram is the probability of some word given the word directly preceding it, or $P(w_n|w_{n-1})$, where the subscript is the location of a word within the sentence. In the phrase just mentioned, one bigram probability would be $P(w_3|w_2)$, or P("like"|"would"). The probability of a trigram is $P(w_n|w_{n-2}w_{n-1})$, the probability of a 4-gram is $P(w_n|w_{n-3}w_{n-2}w_{n-1})$, and so forth.

In order to calculate the probability of a given n-gram string, all that is needed is to find number of times the string occurs in the corpus divided by the number of times the preceding word occurs. Thus for a bigram the probability of a given two-word string, such as "would like" is

$$P(w_n \mid w_{n-1}) = \frac{\sum(w_{n-1}w_n)}{\sum(w_{n-1})}. \qquad (4)$$

To find a long string of words like those in Equation 3, all that needs to be done is to multiply bigram probabilities. To find the probability of the phrase "I would like to make reservations," for example, seven bigram probabilities are multiplied:

P("I would like to make reservations") ≈ P("I"|Beginning of string)P("would"|"I")…

P("reservations"|"make")P(End of string|"reservations") (5)

If each conditional probability in this example, found individually by Equation 4, has a non-zero probability, then the probability of the entire string can be approximated regardless of whether it exists in the corpus. However, similar to the illustration of zero-probabilities for individual words in Figure 2.2, if any of the bigram probability elements in Equation 5 does not occur in the entire corpus and thus is given a probability of zero, the problem of a zero-probability language model, P(E), reappears. Some n-gram strings have zero probabilities even though they simply have not occurred in the corpus and instead should have a small non-zero probability. To overcome this obstacle, smoothing techniques can be used.

Smoothing is a method for allocating portions of the overall probability to strings with zero probability. The simplest smoothing technique is simply adding one to the count of all strings before computing probabilities. Thus, if a given string occurs two times in the overall corpus, the total count using this method is three. Likewise, all previously unseen strings would have a count of one. This method for computing the probability of any given bigram is

$$P(w_n \mid w_{n-1}) = \frac{\sum(w_{n-1}w_n)+1}{\sum(w_{n-1})+V}, \tag{6}$$

where V is the number of unique words encountered in the corpus.

While this method eliminates the possibility of zero-probability strings, it, like other smoothing methods, also introduces the problem of a skewed language model. Since the probability of all bigrams must total to one, assigning non-zero probabilities to these zero-probability bigrams must come at the expense of probabilities of non-zero bigram strings. Likewise, some strings are meant to have zero probability. Using the example of Equation 5, it not likely for a bigram to be something like “I I” or “like like.” These examples should indeed rarely have a count.

To account for this problem, a more common method of smoothing is used. Witten-Bell discounting computes the probability of bigrams unseen in the training data based on bigrams with similar first words [5]. Unlike the above method, this smoothing technique does not add one to every count, thereby causing increases in probabilities that give an incorrect view of the overall distribution, but instead bases the increase in count for any particular string on how often the first part of the string occurs. The probability of all such unobserved bigrams is

$$\sum_{count(w_{n-1}w_n)=0} P(w_n \mid w_{n-1}) = \frac{T(w_{n-1})}{N(w_{n-1})+T(w_{n-1})}, \tag{7}$$

where $T(w_{n-1})$ is the observed number of different bigrams with w_{n-1} as the first word and $N(w_{n-1})$ is the total number of times the first word in the bigram appears in the training corpus. If there are Z different strings, where Z is the vocabulary size minus $T(w_{n-1})$, starting with w_{n-1} that have no count, the probability in Equation 7 is

$$P(w_n \mid w_{n-1}) = \frac{T(w_{n-1})}{Z(N(w_{n-1}) + T(w_{n-1}))} \tag{8}$$

when the bigram count is zero. When the bigram count is non-zero the result is

$$P(w_n \mid w_{n-1}) = \frac{N(w_{n-1} w_n)}{N(w_{n-1}) + T(w_{n-1})} . \tag{9}$$

The reduction in probability of encountered sequences occurs because the probability mass shifts slightly toward unencountered strings in Equation 8, and Equation 9 must compensate for this shift.

2.3 Translation Model

The translation model is the probability that, given some English string, a native speaker of both English and Japanese will translate it as a specific Japanese string, or P(J|E). The translation model's goal is to align words in the source Japanese string with words in the target English string, where order is not important. Instead, the idea is to match words with legitimate partners regardless of order. For example, in Figure 2.3 the mapping of English to Japanese is correct both in alignment and in order, but if the sentence in English is instead "My monkey friend's is this," the probability score for P(J|E) is similar to that of "This is my friend's monkey." Informally, the task of the translation model is to match words between languages and give a score based on whether or not each word or group of words in English has a partner or partners in Japanese.

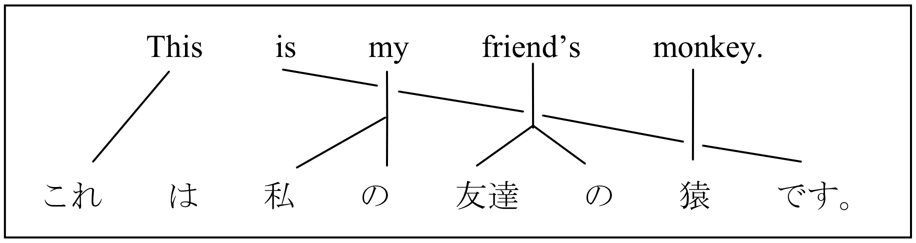

Figure 2.3. Word alignment of English and Japanese strings.

Since the training data is a parallel corpus, it is easy to say that a given Japanese string in the training data is the translation of the English sentence to which it is paired. The translation is accomplished by matching each sentence in Japanese to the similarly numbered sentence in English. However, knowing this process reveals little about how individual words within each sentence are aligned between the two languages. One English word can be aligned with more that one word in Japanese and vice versa. In addition, word order in Japanese is different from that of English. In order to get meaningful results in word alignment, expectation-maximization (EM) is used.

The EM algorithm finds the best approximation of P(J|E), the probability of a source sentence J given a target sentence E, by first setting all probabilities to an equal value, and then, through an iterative process, refining the individual probabilities to obtain a better idea of estimate P(J|E). Five translation models using the EM algorithm are examined here. These translation models are part of the GIZA++ [20] training software used in this research.

2.4 Word Based Translation Models

This section describes a number of translation models as proposed in [5]. All translation models described here were created at IBM labs and are commonly referred to as IBM Models 1-5. IBM Model 1 is the simplest of the five translation models. By examining it in detail, it is possible to understand the basic idea of how all five models work. Models 2 through 5 each have different parameters that allow them to function in particular ways during word alignment. Each of these models is discussed briefly after the analysis of Model 1.

The overall goal of the translation model is to maximize P(J|E) for the training. First, the number of times each word in English aligns with each word in Japanese is counted. In order to find the translation model P(J|E), it is necessary to sum all alignments for where sentence J can be translated as sentence E, or

$$P(J \mid E) = \sum_{a \in \psi} P(J, a \mid E), \tag{10}$$

where J is a single Japanese sentence from the training corpus, E is the corresponding sentence in English, ψ is the set of all alignments, and a is a specific word alignment from the entire set of alignments of J and E. However odd a particular matching of words may seem to a human translator, to the SMT system all alignments start with identical probabilities.

The probability of a particular alignment *a* between some sentences J and E is

$$P(a \mid J, E) = \frac{P(J, a \mid E)}{P(J \mid E)}. \tag{11}$$

Substituting Equation 10 into Equation 11 results in

$$P(a \mid J, E) = \frac{P(J, a \mid E)}{\sum_{a \in \psi} P(J, a \mid E)}. \tag{12}$$

Since $P(J, a \mid E)$ is found in both Equations 10 and 12, it can be written in terms of individual word translation probabilities. The rewritten version of $P(J, a \mid E)$ is

$$P(J, a \mid E) = \underbrace{P(m \mid E)}_{1} \prod_{i=1}^{m} \underbrace{P(a_i \mid a_{1,i-1}, w_{1,i-1}^{J}, m, E)}_{2} \cdot \underbrace{P(w_i^{J} \mid a_{1,i}, w_{1,i-1}^{J}, m, E)}_{3}, \tag{13}$$

where m is the length of the Japanese sentence J and w_i^J is ith word in J. Here, the right hand side is divided into three parts. Part 1 is the probability that concerns the length of the Japanese sentence given the English sentence. Part 2 allows the choice of the position of the words in J. Finally, Part 3 corresponds to the choice of the word w in J given the position, the length of J, and the English sentence.

In IBM model 1 $P(m \mid E)$ is constant and is assumed to be independent of E and m. This model also assumes that any given word in J has the same alignment probability regardless of position. The part 2 position probability is

$$P(a_i \mid a_{1,i-1}, w_{1,i-1}^J, m, E) = \frac{1}{l+1}, \tag{14}$$

where l is the length of the English sentence.

The third assumption is that any Japanese word chosen depends only on its aligned English word. Therefore, a simplified version of Part 3 in Equation 13, where $t(w_i^J \mid w_{a_i}^E)$ is the translation probability of the Japanese word w_i^J given its alignment with the English word $w_{a_i}^E$, is

$$P(w_i^J \mid a_{1,i}, w_{1,i-1}^J, m, E) = t(w_i^J \mid w_{a_i}^E)\,. \tag{15}$$

After combining the three assumptions made by Model 1, Equation 13 is

$$P(J, a \mid E) = \frac{\varepsilon}{(l+1)^m} \cdot \prod_{i=1}^{m} t(w_i^J \mid w_{a_i}^E)\,, \tag{16}$$

where $(l+1)^m$ is the number of possible alignments and ε is the value of P(m|E). This result follows since $(l+1)$ positions exist in sentence E (length + 1) and there are m possible words in J to align with each of these positions. Here $(l+1)^m$ possible alignments have the potential to be an unmanageable number. A few remedies for this possible problem is discussed later in this chapter. Summing over all possible alignments a_1 through a_m, Equation 16 is

$$P(J \mid E) = \frac{\varepsilon}{(l+1)^m} \cdot \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} \prod_{i=1}^{m} t(w_i^J \mid w_{a_i}^E). \tag{17}$$

In order to maximize $P(J \mid E)$, it is necessary to adjust the translation probabilities in Equation 17 subject to the constraints that for each E

$$\sum_J t(w^J \mid w^E) = 1. \tag{18}$$

To solve the constrained maximization problem, Lagrange multipliers are used. The auxiliary function to maximize is given as

$$h(t, \lambda) = \frac{\varepsilon}{(l+1)^m} \cdot \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} \prod_{i=1}^{m} t(w_i^J \mid w_{a_i}^E) - \sum_{w^e} \lambda_{w^E} \left(\sum_J t(w^J \mid w^E) - 1\right), \tag{19}$$

which is maximized by taking the partial derivative of h with respect to $t(w^J \mid w^E)$. The partial derivative is zero for

$$t(w^J \mid w^E) = \lambda_{W^E}{}^{-1} \frac{\varepsilon}{(l+1)^m} \cdot \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} \sum_{i=1}^{m} \delta(w^J, w_i^J) \delta(w^E, w_{a_i}^E) \prod_{k=1}^{m} t(w_i^J \mid w_{a_i}^E). \tag{20}$$

The product of the Kronecker delta functions in Equation 20 is equal to one when the arguments are the equal and zero when not equal.

Altering Equation 16 so that

$$\prod_{i=1}^{m} t(w_i^J \mid w_{a_i}^E) = P(J, a \mid E) \frac{(l+1)^m}{\varepsilon} \tag{21}$$

and substituting this product into Equation 20 yields

$$t(w^J \mid w^E) = \lambda_{W^E}{}^{-1} \underline{\sum_{a \in \psi} \sum_{i=1}^{m} \delta(w^J, w_i^J) \delta(w^E, w_{a_i}^E)} P(J, a \mid E), \tag{22}$$

where the underlined portion is the count of all word alignments between J and E.

The number of times a word in English aligns with a word in Japanese in the sentence translation (J|E) is estimated from

$$count(w^J \mid w^E; J,E) = \sum_{a \in \psi} P(a \mid J,E) \sum_{i=1}^{m} \delta(w^J, w_i^J)\delta(w^E, w_{a_i}^E) . \tag{23}$$

In addition, from Equation 11, $P(a \mid J,E)P(J \mid E)$ is substituted for $P(J,a \mid E)$ in Equation 2.22. Next, λ_{w^E} is replaced by $\lambda_{w^E} \cdot P(J \mid E)$, thereby canceling the two instances of $P(J \mid E)$ and leaving

$$t(w^J \mid w^E) = \lambda_{W^E}{}^{-1} count(w^J \mid w^E; J,E) , \tag{24}$$

where $\lambda_{w^E}{}^{-1}$ is a normalization constant. Since the corpus has a large number of Japanese-English sentence pairs (here, 20,000 pairs for training), Equation 24 is now

$$t(w^J \mid w^E) = \lambda_{W^E}{}^{-1} \sum_{sent=1}^{N} count(w^J \mid w^E; J_{sent}, E_{sent}) , \tag{25}$$

where J_{sent} and E_{sent} are the sentence numbers corresponding to a given pair in the parallel corpus, and N is the number of sentence pairs in the training corpus.

As mentioned earlier, there is now a potentially enormous number of alignments. If the Japanese and English sentences are only 8 words in length each (m = l = 8), the number of possible alignments is more than 43 million. The current number of alignments, $(l+1)^m$, may be simplified using

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{i=1}^{m} t(w_i^J \mid w_{a_i}^E) = \prod_{i=1}^{m} \sum_{k=0}^{l} t(w_i^J \mid w_k^E) . \tag{26}$$

Substituting this equation into Equation 17 and then taking the partial derivative yields

$$count(w^J \mid w^E; J,E) = \frac{t(w^J \mid w^E)}{t(w^J \mid w_0^E) + \ldots + t(w^J \mid w_l^E)} \sum_{i=1}^{m} \delta(w^J \mid w_i^J) \sum_{k=0}^{l} \delta(w^E \mid w_k^E) , \tag{27}$$

which reduces the overall number to approximately $(l+m)$, or 16 alignments for the previous example.

The individual translation probabilities are now estimated using the Japanese-English parallel sentences and Equations 25 and 27. Brown et al [5] summarizes this estimation process with the four following four steps:

1) Choose initial non-zero values for $t(w^J \mid w^E)$.
2) Determine the word translation counts using Equation 27
3) Find a new estimate for $t(w^J \mid w^E)$ by taking counts from step 2 and using Equation 25.
4) Repeat starting at step 2 until $t(w^J \mid w^E)$ converges to a maximum.

IBM Model 2 is similar to Model 1 except that Model 2 adds a distortion parameter. Thus, when attempting to align words between Japanese and English, Model 2 takes into account distance between words in both languages and penalizes words that are in distant positions.

IBM Model 3 goes a step further and adds fertility, which gives the probability of a single word in English being the best alignment for multiple words in Japanese. In Model 1 it is possible for a word in English to align with all words in Japanese. One problem with Model 3 is that it uses all alignment possibilities, which greatly increases complexity. However, note that the use of any of these models is not exclusive. It is reasonable to take the values of $t(w^J \mid w^E)$ found with Model 1 and use them as inputs to Model 2, for example. Thus the aforementioned processing problem associated with Model 3 can be greatly reduced if Model 1 is used first. By refining results with Model 1 and then using the best alignment as the input to Model 3, the total number of likely alignments is reduced.

IBM Model 4 improves on the distortion parameter from Model 2 by dividing it into two different sets of parameters. The first is a separate distortion probability for head words, or those that are the first words in each Japanese sentence. The second is a distortion probability for all non-head words, or those occurring later in the string of Japanese words.

Finally, IBM Model 5 adds a deficiency parameter to eliminate the wasting of probability mass on strings that cannot occur. In other models it is possible to generate words at the same position in a string.

2.5 Phrase Based Translation Models

Phrase based translation models overcome the limitations of word based translation models by allowing words or sequences of words in the source language to align with sequences of words in the target language. Using such models means that the two sequences can have differing lengths. Pharaoh is the phrase-based decoder used with this research [12]. Once a translation model and language model have been created, the decoder allows for the source language test data to be translated into the target language.

2.6 Problems with SMT

There are number of problems with statistical machine translation that must be addressed to produce quality translations. These problems are worse for some languages, and they include issues with morphology, syntax, word order, and compounds. Many of these issues are discussed in the following chapter and relate to Japanese.

Another common problem in SMT is dealing with out-of-vocabulary (OOV) words in the test data. Since the training data is finite, it is likely that words will appear in the test data that have never been seen before. One benefit to working with Japanese is that, as discussed in the next chapter, Japanese word meaning, or at least an approximation of meaning, can often be deduced from individual characters making up a particular word. In addition, since Japanese uses different stems to denote different parts of speech, reducing the meaning of the OOV word in question to a particular part of speech is possible. That is, Japanese attaches characters to the end of words to give the reader the idea that although a given word may not be known, it is certain that it is a verb, adverb, adjective, etc. In addition to other grammatical points in Japanese, these aspects make it possible for OOV words to be estimated by the SMT system. There are also other non-linguistic approaches for solving this problem. Bazzi and Glass [2] have shown that OOV words can be modeled with success by dividing them into multiple classes and using pattern recognition techniques to reduce word error rates to reasonable values.

III. Japanese Background

This chapter addresses basic issues concerning the written Japanese language, including the writing system, morphology, and grammar.

3.1 Japanese Writing System

East Asian languages such as Chinese, Japanese, and Korean (CJK languages) pose a number of different problems for those working on machine translation systems that are not encountered in Romance languages. With supervised translation, some of these problems can be addressed before processing language data. Among the most noticeable qualities of CJK languages is the use of writing systems based on Chinese characters (*hanzi*). In the case of Japanese, the written language has developed to include not only Chinese characters (*kanji*), but also a pair of *kana* syllabaries (seen in Figure 3.1), *hiragana* and *katakana* (each containing 102 possible characters). *Kanji* is mainly used for nouns and for verb, adjective, and adverb roots, while *hiragana* is used for inflecting verbs, adjectives, and adverbs and as particles, which are used to mark various grammar points within a sentence. Finally, *katakana* is used only for foreign loan words and for onomatopoeia sounds.

The example sentence in Figure 3.2 shows how each of these kinds of characters is incorporated into a typical Japanese sentence. Refer to Figure 3.1 to fully appreciate the use of *katakana* as a foreign word syllabary.

Hiragana

Vowels alone

A	I	U	E	O
あ	い	う	え	お

Consonant + vowel

	A	I	U	E	O
K	か	き	く	け	こ
G	が	ぎ	ぐ	げ	ご
S	さ	し *	す	せ	そ
Z	ざ	じ *	ず	ぜ	ぞ
T	た	ち *	つ *	て	と
D	だ	ぢ	づ	で	ど
N	な	に	ぬ	ね	の
H	は	ひ	ふ	へ	ほ
B	ば	び	ぶ	べ	ぼ
P	ぱ	ぴ	ぷ	ぺ	ぽ
M	ま	み	む	め	も
Y	や		ゆ		よ
R	ら	り	る	れ	ろ
W	わ				を *

Misc. Syllables

N	ん
*	っ

Consonant + y + vowel

	YA	YU	YO
K	きゃ	きゅ	きょ
G	ぎゃ	ぎゅ	ぎょ
S	しゃ	しゅ	しょ
CH	ちゃ	ちゅ	ちょ
N	にゃ	にゅ	にょ
H	ひゃ	ひゅ	ひょ
B	びゃ	びゅ	びょ
P	ぴゃ	ぴゅ	ぴょ
M	みゃ	みゅ	みょ
R	りゃ	りゅ	りょ

Katakana

Vowels alone

A	I	U	E	O
ア	イ	ウ	エ	オ

Consonant + vowel

	A	I	U	E	O
K	カ	キ	ク	ケ	コ
G	ガ	ギ	グ	ゲ	ゴ
S	サ	シ*	ス	セ	ソ
Z	ザ	ジ*	ズ	ゼ	ゾ
T	タ	チ*	ツ*	テ	ト
D	ダ	ヂ	ヅ	デ	ド
N	ナ	ニ	ヌ	ネ	ノ
H	ハ	ヒ	フ	ヘ	ホ
B	バ	ビ	ブ	ベ	ボ
P	パ	ピ	プ	ペ	ポ
M	マ	ミ	ム	メ	モ
Y	ヤ		ユ		ヨ
R	ラ	リ	ル	レ	ロ
W	ワ				ヲ*

Misc. Syllables

N	ン
*	ッ

Consonant + y + vowel

	ya	yu	yo
K	キャ	キュ	キョ
G	ギャ	ギュ	ギョ
S	シャ	シュ	ショ
CH	チャ	チ	チョ
N	ニャ	ニュ	ニョ
H	ヒャ	ヒュ	ヒョ
B	ビャ	ビュ	ビョ
P	ピャ	ピュ	ピョ
M	ミャ	ミュ	ミョ
R	リャ	リュ	リョ

Figure 3.1. Japanese kana syllabaries, hiragana for native Japanese words, word endings, and particles, and katakana for foreign loan words. The pronunciations are the same for each. A few exceptions, marked with the asterisk, exist: SI→SHI, ZI→JI, TI→CHI, TU→TSU, WO→O, and small TSU is used for plosives, doubling the consonant that follows.

kanji	*hira*	*kanji*	*hira*	*kata*	*kata*	*hira*
私	の	名前	は	エリック	チャプラ	です。
I	(possessive)	name	(topic)	Erik	Chapla	is

"My name is Erik Chapla."

Figure 3.2. Simple Japanese sentence showing the use of kanji, hiragana, and katakana. Kanji is used for nouns and verb, adjective, and adverb roots. Hiragana is used mostly for word endings, particles, markers. Katakana is used almost exclusively for foreign names and words.

Kanji used in this research are all among the *jouyou kanji* list of government approved characters for basic fluency. This list included 1945 characters, most of which have been greatly simplified from the Chinese versions of the same characters. All Japanese sentences used in training and testing were encoded with UTF-8 character encoding.

3.2 Morphology

Morphology deals with word formation in terms of inflection, derivation, and compounding. It can be helpful for statistical machine translation in that not all words need to be recognized by the system to make a prediction of word meaning. In other words, if a verb stem, 食べ (*tabe* or "eat"), for example, has already been encountered by the system, but 食べられません (*taberaremasen* or "not able to eat") has not been seen, by using the base characters, it is possible to conclude that the verb "eat" is involved. In addition, if verb inflections are known (in this case, られる(*rareru*; potential suffix) and ません (*masen*; negative suffix)), then a better approximation of the true meaning of the previously unobserved verb can be made.

First Character	Meaning	Second Character	Meaning	Combined	Meaning
文	Writing	学	Study	文学	Literature
芸	Art	人	Person	芸人	Artist
風	Wind	上	Up	風上	Windward

Table 3.3. Examples of derivation

First Character	Meaning	Second Character	Meaning	Combined	Meaning
日	Sun	本	Origin	日本	Japan
手	Hand	首	Neck	手首	Wrist
花	Flower	火	Fire	花火	Fireworks

Table 3.4. Examples of compounding

Derivation is the formation of a word from another word, as in "computer" from the word "compute," where both the root word and the desired word have similar meanings. In Japanese, this kind of word formation is accomplished in much the same way as inflection described previously. For example, the word 日本語 (*nihongo*; Japanese language) is a result of adding the character 語 (go; language) to 日本 (*nihon*; Japan). Other examples appear in the Table 3.3.

Compounding, on the other hand, takes a word and with the attachment of another word results in a different meaning. While compounding results in creative word meanings, sometimes to the point of poetry, the meanings are never directly based on any of the words involved. Examples are found in the Table 3.4.

As can be seen from these examples, derivation in Japanese is very closely related to compounding, so when this process is described later, both will be lumped under the general category of compounding.

3.3 Japanese Grammar

In addition to the fact that Japanese and English have entirely different scripts, there are a number of other problems that must be examined when attempting translation of Japanese text. These include Japanese word order, formality, topic and subject markers, and sentence-final particles, among others.

3.3.1 Word Order

Word order can affect alignment of Japanese and English sentences. English is known as an SVO language, meaning that typical word order within a sentence is first a subject, followed by a verb, and ending with an object. A simple example of a sentence with SVO form is “John kicked the ball.” Japanese, however, is a SOV language. The same sentence here would be something like “John the ball kicked.” Word order and alignment is not much of a problem with small sentences, but as sentence length grows and grammar becomes more complicated, word order becomes increasingly more important.

For MT, word order is important in that it may greatly affect language model probabilities. If an English sentence is misaligned with a Japanese sentence, it may take much more training for the system to correctly match words between the two languages. If, however, the user can implement prior knowledge about the language to artificially alter the word order in Japanese to change it into something close to SVO form before any training takes place, the final translation can be greatly affected in terms of quality and accuracy.

3.3.2 Formality and Politeness Levels

Formality and politeness levels, known as *keigo*, within the Japanese language are the subject of numerous books and is an academic sub-discipline of Japanese language studies in its own right. Because of the difficulty, *keigo* is an annoyance to most foreign students studying Japanese. *Keigo* has multiple levels of politeness, each with differing words and word endings depending on the relationship between the speaker and listener. If, for example, a company employee is talking to the company president, a higher level of formality is used than if both people are friends. Different levels include speech toward a low inferior (such as a pet), inferiors (employees working under you or children), those on equal status level (friends), those of equal position in a business environment, moderate superior level (when talking to one's boss, for example), and high superior level (only used for those approaching the emperor).

Here, the problem of formality levels is addressed by making all Japanese training and testing sentences all on equal *keigo* levels. This solution comes from analyzing the Japanese sentences and the corresponding English sentences. It is evident that rarely do the many different levels of formality found in the Japanese sentences appear in their English counterparts. In the Japanese example of Figure 3.5, two words give a sense of formality that is not conveyed in the English translation. Because there are different politeness levels used throughout the Japanese training and test data, and therefore often occurrences where multiple Japanese words are assigned to the same English word without any hint of the politeness from the original Japanese, the solution was to reduce all politeness in the Japanese texts to neutral formality.

何時　に　伺え　ば　$\underbrace{\text{よろしい}}_{\textbf{\textit{good, alright (formal)}}}$　$\underbrace{\text{でございます}}_{\textbf{\textit{is, exist (formal)}}}$　か。

何時　に　伺え　ば　$\underbrace{\text{いい}}_{\textbf{\textit{good, alright (neutral)}}}$　$\underbrace{\text{です}}_{\textbf{\textit{is, exist (neutral)}}}$　か。

When is it alright for me to visit?

Figure 3.5. Two Japanese sentences from the test data. The first is the original sentence with two formal words that give the reader the idea that the speaker is being very polite with his audience. The second is the neutral form of the same sentence with both words reduced to their more common forms. The English translation stays the same in both instances.

Another part of speech examined here is the use of -*san* after names and professions. Examples, of this are Smith-*san* or Yamamoto-*san* appearing in the Japanese text. The meaning in these cases is simply Mr./Ms./Mrs. Smith or Mr./Ms./Mrs. Yamamoto, respectively, in English. In addition, the –*san* suffix is also applied in the Japanese text to professions such as *sakanaya-san* or *kachô-san*, or Mr./Ms./Mrs. fish store owner or Mr./Ms./Mrs. section chief, respectively, in English. In both cases, the *san* suffix tell nothing about the gender of the person in question since all sentences in the training and test data are single sentences without any supporting sentences to give context. It is unreasonable to eliminate the suffix, however, since it allows the system to conclude that the attached word is a name or profession.

3.3.3 Markers

Another feature of Japanese is the use of particles. These individual *hiragana* characters are not usually considered words, but instead act as grammatical markers. Because there are rarely direct translations for particles, they are one of the more troubling aspects of SMT for Japanese. Take, for example, the simple sentence in Figure 3.6.

本田さん	は	何	で	大阪	へ	行く	ん	です	か。
Mr. Honda	(p)	why	(p)	Osaka	(p)	go	(p)	is	(p).

Why is Mr. Honda going to Osaka?

Figure 3.6. Example of segmented Japanese sentence with English word meanings and particles marked with (p).

Looking at the Japanese sentence, it is obvious that many of the words translate directly word-for-word into English. However, three of the five particles found in this example do not have direct meaning in English. In fact, は, ん, and か are usually left untranslated when human translation is performed. Respectively, these particles mark the preceding word as the sentence topic, nominalize the preceding verb, and act as a question mark. Here, で and へ have multiple meanings, but usually act to show place of a given action.

Here only the seven most common particles in the Japanese language are employed. These particles are described briefly in Table 3.7. Along with common meanings for each, also included are the probabilities of each occurring in the training and test data in terms of occurrences of each versus total word count.

Particle	Meaning	Token Probability (%)
は	Marks the topic of sentence	4.7
が	Indicates subject of sentence	1.7
に	Marks location of action/Indirect object/Passive/Causative	3.6
へ	Marks location of action	0.2
で	Indicates location/use of something/time of action	1.3
の	Indicates possession/Indefinite pronoun/Nominalization	4.2
か	Marks question	5.5
を	Indicates direct object	2.5

Table 3.7. Seven most common particles in Japanese along with meaning and approximate percentage of occurrences within the training and test data.

One of the experiments involved substituting and eliminating various particles from the text data based on knowledge of Japanese grammar. For example, one attempt at improving the overall translation assumes that は (topic marker) and が(subject marker) perform the same basic function. Since the concept of topic and subject are so closely related in Japanese, it may be possible to only use one marker instead of both, thus eliminating some confusion by the system. Another example of a hypothesis tested here concerned replacing all occurrences of the locative particle へ with the locative particle に. It is generally accepted that へ always performs the same function as に in its locative meaning. This correspondence does not, however, work in the opposite direction, replacing に with へ, since に has various meanings that go beyond that of へ.

3.3.4 Context

As indicated earlier, Japanese is a highly contextual language. This fact alone poses many problems for the MT system. Entire portions of sentences can be left out for the convenience of the speaker. For example, if the topic has already been referred to in a previous sentence, or even if the topic is generally understood without being previously mentioned, the topic can be omitted from there on. If two people see someone sleeping, one of them may just say “sleeping” (寝ている or *neteiru*) about the sleeping person. There is no need to say in such a sentence who the sleeping person is or what the person’s gender is. Since it is understood by everyone involve, only this single verb is necessary.

A one word sentence like this would be matched in the English data as something like “He/She/It is sleeping.” The SMT system can only assume, therefore, that the single Japanese word is three words.

One way to deal with this problem is to artificially insert “missing” words into the Japanese to make the corresponding sentences match well. Another way is to eliminate words from the English training sentences to match those in the Japanese. Both methods would require supervised human intervention and take, for even a few thousand sentences of language data, many hours to correct. In addition, the test data must be changed to match the format of the training data. For these reasons, the problems of context are not addressed here.

3.3.5 Other Grammar Points

Numbers pose another problem to performing SMT on the Japanese data available. The way all numeric information (times, dates, money, bus numbers, etc.) is displayed in the training and test data is with kanji characters representing numbers from zero to nine, with no tens, hundreds, thousands, etc. Unlike normal written Japanese, where large numbers have separate characters for marking tens, hundreds, and thousands places, the data used here has none of these place markers. This absence is inconvenient because each individual number character in the sentences in the texts used here is separated from its neighbors by a space, making them appear as individual words and therefore making them more susceptible to errors in alignment. For example, consider the string of characters in Figure 3.8, taken from Japanese data.

一二三四
(1,234 in this data)

一千二百三十四
(1,234 in normal Japanese texts)

一千　二百　三十　四
(possible version of correctly spaced 1,234)

Figure 3.8. Example of numeric information from Japanese data used here and what this string of numbers looks like in other Japanese texts (newspaper, books, etc.). In addition, the third line shows the way the corrected string would appear in research here, with spaces included.

Since the Japanese sentences throughout the text data include spaces between individual words, it impossible without context to know if the sequence in Figure 3.8 means, "one-two-three-four" (for a bus, train, or hotel room number) or "one thousand two hundred and thirty-four." In a typical Japanese sentence, when the former (and more uncommon) case is meant, 一二三四 would be written. If on the other hand the latter meaning is being conveyed, 一千二百三十四 is likely to be written, where the characters 千 ("thousand"), 百 ("hundred"), and 十("ten") are included to tell the reader that this is one complete number instead of four distinct ones. These cues are not available here, so much meaning is immediately lost. There is no place marker for the ones place in Japanese. It is also possible to use 万 ("ten thousand") and 億 ("one hundred million") as place markers, but these do not seem to appear in any of the data used in this research. Lack of context again plays a detrimental role in translation.

One way to remedy this problem is to artificially insert place marker characters (千, 百, and 十) into number strings when appropriate. It is also be necessary to eliminate spaces between each of these new characters and the preceding number that they modify to avoid possible alignment errors. There are a number of problems with this solution. First, it is extremely time consuming, since it would require the user to make corrections to the number strings. There is no way for the researcher to quickly make changes to the numeric examples, since each example must be considered separately. This inability leads to a second problem. Most of the Japanese sentences in the training data have no supporting sentences from which to draw context. There is no way of knowing which of the two above cases are meant. Third, correcting each number and leaving no spacing between parts of the numeric strings results in numbers that are too specific. In order to build a good language model, it is necessary to find multiple occurrences of words, but in this case there are very limited examples of each case. In the previous example, 一千二百三十四 is very specific, and in all likelihood will never occur again, even in extremely large corpora.

IV. Japanese Word Segmentation

The first part of this chapter concerns methods for applying word segmentation to raw Japanese text to prepare it for processing. The text used in the SMT experiments is from the 2005 IWSLT campaign, and was already word segmented by others. Many times, however, text must first have spaces inserted between words before any SMT work can be performed. The first part of this chapter explains the methods for dealing with text that has not been segmented. The second part of this chapter details other experiments performed on Japanese sentences, after they are word segmented, to alter the performance of the SMT system.

4.1 N-grams and Segmentation of Japanese Text

Segmentation creates spaces between words in a string of text and is a necessary step for SMT of Japanese text. In the data used here, all sentences are already segmented, but if new sentences are mined from the internet, books, newspapers, etc., segmentation would have to be performed on each sentence. Segmentation can be done by native Japanese speakers or by using an algorithm that applies spaces where needed to unsegmented text. The former is best if time is not a factor, since a native speaker is always more skilled at properly segmenting text than the most well trained SMT system.

Segmentation is an important issue with Japanese text because, unlike English sentences, where each word is clearly defined with a space between it and its neighbors, a normal Japanese sentence is a long string of characters with no spaces. Figure 4.1 shows an example of an unsegmented English sentence and its Japanese equivalent. Unlike native English speakers who have little problem reading the string "Itlookslikeitwillraintoday" and assigning the correct meaning to it, the SMT system has little chance of extracting any meaning from either the unsegmented English or Japanese sentence and will instead see both as a single word composed of a string of characters, not a group of unsegmented individual words. A first step, therefore, is to artificially add spaces between words.

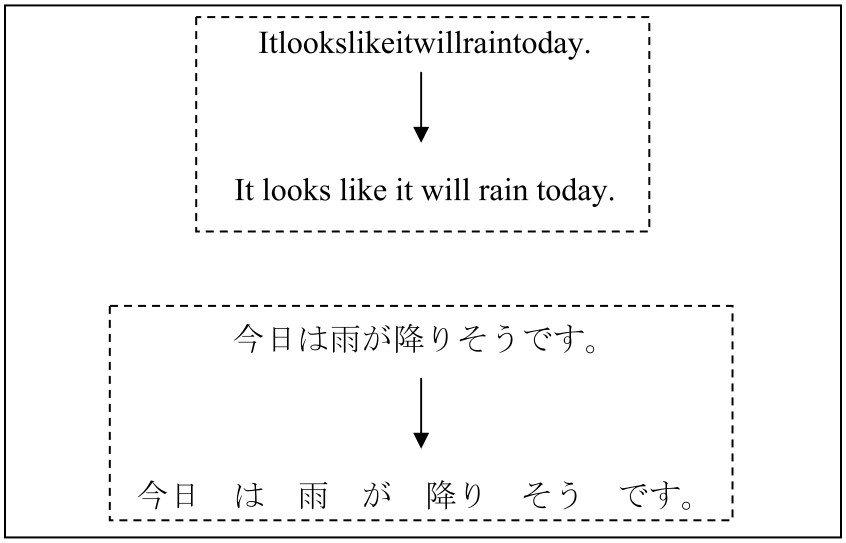
Itlookslikeitwillraintoday.
It looks like it will rain today.
今日は雨が降りそうです。
今日　は　雨　が　降り　そう　です。

Figure 4.1. Unsegmented and segmented versions of English sentence and translated Japanese sentence.

Unlike the unrealistic unsegmented English sentence in Figure 4.1, the unsegmented Japanese equivalent sentence is completely normal. If fact, other than elementary children's texts, text with spaces between words is rarely seen in Japanese. Therefore, for SMT a typical Japanese sentence must be changed into an atypical, unrealistic one. To change Japanese text into something that the system can process with each word clearly separated by space, an algorithm must be used to segment each sentence in both the training and test data before continuing on with other machine translation tasks.

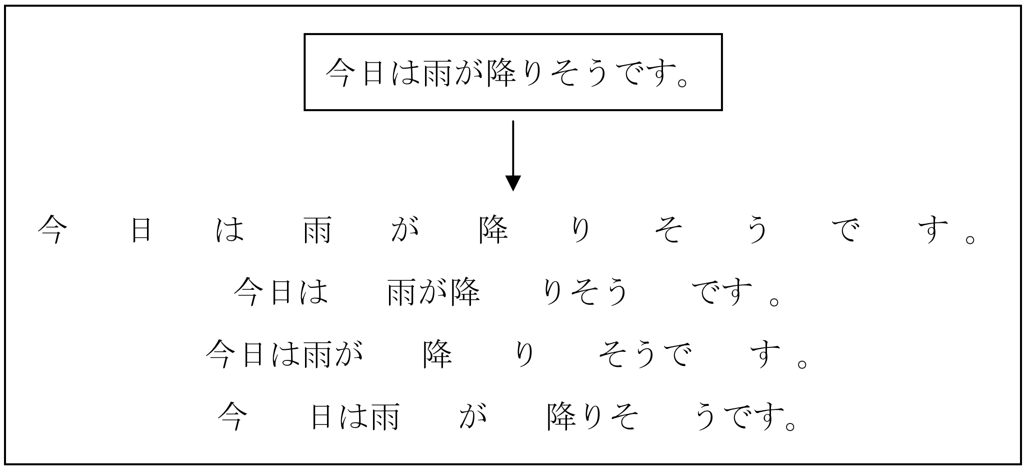
今日は雨が降りそうです。
今 日 は 雨 が 降 り そ う で す 。
今日は 雨が降 りそう です 。
今日は雨が 降 り そうで す 。
今 日は雨 が 降りそ うです。

Figure 4.2. Original unsegmented Japanese sentence in the above box and four possible ways the algorithm may break the sentence into component words. The spacing here does not imply that these are correct word boundaries, but instead indicates where a poorly trained system may place spaces.

The method employed here uses n-grams, or windows of length n, to find and compare probabilities of a potential word to those of its neighbors. This method is a variation on the method of Ando and Lee [1]. Their research is primarily focused on segmentation of character strings composed entirely of *kanji*, as opposed to strings of more colloquial, less official Japanese used here that also incorporates *hiragana* and *katakana* characters. In more difficult texts such as technical papers, business journals, and legal documents, there are often long strings of eight or more *kanji* with no *kana*. These strings of characters therefore often look like Chinese. In the simpler travel domain, strings of more than three consecutive kanji characters are rarely seen and the long *kanji* string phenomenon poses no problems. Figure 4.2 shows some of the possible (and incorrect) ways that a sentence can be segmented.

The main idea behind the segmentation method used here is to examine groups of characters around a given point in a string of characters and decide, by using word probabilities, whether or not a space should be inserted at that point.

This method uses more than one n-gram window to make the decision. For example, move a 2-gram window (or window holding only two characters at a time) over a given sentence, followed by a 3-gram window over that same text, then do so with each possible combination of windows (i.e. a 2-gram window with a 3-gram window, a 2-gram with a 4-gram, etc.). It is found that using a combination of 2-gram and 3-gram windows gives the best results, while any other combination results in more segmentation errors. Charts showing the numeric results of all segmentation experiments attempted are found in Section 4.4.

The main problem with unsupervised segmentation versus human-led segmentation is that a computer addressing this problem has no knowledge of how a Japanese sentence should be properly segmented. The task at this stage is basically like giving a Japanese sentence to an English speaker, who has never seen Japanese text, and asking him to find the individual words. Here n-gram analysis is useful. The system takes a sample window of n consecutive characters in a given sentence and examines the probability of that sample in relation to the probabilities of adjacent samples. The number of consecutive characters of the sample and of the comparison samples is based on the number of the n-gram window length. For example, if one uses 5-gram analysis, the window used is five characters in length and all adjacent samples used for comparison are also five characters long.

4.2 Data Used

The text segmentation technique must take into account that the word segmentation system has a large amount of data to work with, both in terms of reference words and in test data sentences. Without segmentation, n-gram analysis would be useless since all word occurrences would be equally likely. Training allows for individual word probabilities to be compiled so that reliable statistics can be referenced during testing.

The first part of data collection requires a corpus of Japanese words that would allow the system to have accurate probabilities for word occurrences. In order to have accurate probabilities, and because of the sheer number of possible words in the Japanese language, the corpus must be extremely large, most likely with a size of at least a few million characters. The optimal corpus should contain the two 102 character *kana* syllabaries and the 1945 *kanji* characters required for literacy. These *kanji* characters, known as *jouyou kanji*, are from a list compiled by the Japanese Ministry of Education in 1981 that consists of 1,006 elementary school *kanji* and 939 secondary school *kanji*.   The list was created to give Japanese knowledge of all *kanji* needed to read a typical newspaper. In addition to the 1945 characters, the corpus should also contain all combinations of *kanji* and *kana* characters that would form intelligible words, since words are often composed of a combination of individual characters (*kanji* + *kanji*, *kanji* + *kana*, and two or more *kana* characters together).

An existing body of data is used because creation of such a large corpus is not a feasible task. Girardi and Kelly [10] collected and put into the public domain a large amount of Japanese text data from the Mainichi Shimbun, one of Japan's main newspapers. Their work collected four years of daily newspaper text and resulted in nearly 70,000 different words. In addition, they totaled the number of occurrences of each word over the four year period, with the most common word occurring over three million times and the least common word occurring only a handful of times. This list is important in that it allows the creation of a sorted lookup table with probabilities for nearly every Japanese word that could appear in the testing stage. In fact, at no time during testing did the system find a word for which it had no previous reference.

The next step in the data collection process finds hundreds of new sentences from Japanese newspapers for testing. It is necessary for this sentence data to be from newspapers since the original vocabulary data was mined from newspapers. It is important to collect data that would have similar style, format, and vocabulary so that the testing data is of the same sort as the training data. Otherwise, there can be no real way of knowing that word occurrences matched correctly.

In order to analyze collected data, it must be in a numeric form and not in normal Japanese text. To accomplish this task, the entire set of words from the training and test data is translated into UTF-8 format so that a numeric code could be used to represent each Japanese character. Each character has its own unique five-digit UTF-8 identifier, so every word can be examined as a group of multiple five-digit numbers instead of a group of Japanese characters, which greatly simplifies the process, for now the entire lookup table of trained data and the testing data is in a numeric form like the following (instead of Japanese text): 10453 10433 16785 12340 12819 10983 19893 10912 …

The training data table therefore has the format shown in Table 4.3 after all characters where converted into UTF-8. The top part the table shows a few sample lines from the thousands of words and occurrences from the raw data that could be referenced, while the bottom table is the same sample after the Unicode conversion is created.

Word	# of occurrences	P(occurrence)	Rank
の	3104746	0.010796	1
はばたく	40153	0.000156	5601
漢字	3693	0.000035	12007

$\downarrow$

UTF-8 Code	# of occurrences	P(occurrence)	Rank
12403	3104746	0.010796	1
10516 15652 16199 16546	40153	0.000156	5601
16150 12601	3693	0.000035	12007

Table 4.3. Sample lines for the lookup table for Japanese word probabilities. The top table is from the large lookup table before any numeric alteration. The bottom is the same table after characters were changed into Unicode for easy numeric reference and sorting. There is one Unicode number for each character in a word, therefore, since the second word has four characters, it is necessary to keep track of for Unicode numbers in exact order each time this word is referenced in a sentence. Otherwise, the same four characters can be referenced, but character order can not be guaranteed.

A few other decisions are made about the text data used for testing. The first decision is whether or not to leave or delete all punctuation and symbols. Again, this decision is taken care of by humans in the text information provided for SMT research explained later, but for testing the segmentation algorithm the newspaper data has punctuation and symbols left in. Obviously, punctuation can provide useful information to a person reading a text. However, since punctuation provides the system with information that makes analysis difficult, since no punctuation is included in the word lookup table, and since punctuation is not very useful in Japanese, it is decided to uniformly clean all sentences of punctuation.

Second, a decision as to whether or not to analyze the first and last group of characters of a sentence is needed. This decision is important since n-gram analysis takes into account the score of boundaries before and after the current boundary being analyzed. If the current candidate boundary for a space occurs at the beginning of a sentence (directly after the first character of the sentence), it means that the boundary directly before that space occurs before the start of the sentence, which is illogical. In order to rectify this problem and still allow for the possibility of individual characters at the beginning and end of sentences, special rules for such cases are created. For example, if the boundary in question occurs between the first and second characters of the sentence, only the next possible boundary is considered for comparison, and a possible boundary before the first character is disregarded. In addition, when the boundary being proposed for a space occurs directly before the last character of the sentence, only the previous boundary is considered for comparison. These exceptions only requires about twenty extra lines of code.

The final problem concerns the size of the corpus needed for training. Since the corpus contains many thousands of words, the size of the lookup table of sorted words needed for reference by the test data greatly slows overall processing. In addition, since the first fifty percent of word occurrences are less than 0.1% of the total number of words (since the most common words have a large number of occurrences), the size of the table is due mostly to words that will probably never be seen during testing. In other words, it is very likely that each time a test sentence is analyzed with the lookup table, the system needs to sort through much computational dead weight to find the statistics of the word. Other than sorting the data, not much can be done about this problem, since there is no guarantee that any of the characters in the table will be unused. Therefore, the entire table is retained for the entire testing portion of this work.

4.3 Japanese N-gram Analysis

To illustrate n-gram analysis, consider a 3-gram window, where each sample window contains three consecutive characters. For clarity, instead of Japanese characters, consider a sentence “ABCDEFGH,” where each English letter represents a single Japanese character. The first step in n-gram analysis is to propose a place in the sentence where a space may occur. This proposal occurs on a rolling basis through the sentence. That is, for 3-grams, the first possible space is after the first character of the sentence. Once a decision is made on whether or not to put a space at that location, the length-3 window moves one character to the right and a possible space is proposed after the second character. This process continues until the end of the string of characters. This process of using n-grams is somewhat different from the process described in Section 2.2, where Japanese n-grams refer to sequences of n characters.

In this example, assume that the likelihood of a space between the third character C and the fourth character D must be found. A temporary boundary is placed at this location, and the proposal is made that ABC is a word, followed by a space (located at the temporary boundary just proposed), then the start of a new word at character D and continuing for some unknown length. To test this space proposal, the next step examines the probabilities of the three-character string before the proposed boundary, ABC, and the three-character string directly following the proposed boundary, DEF.

Next, the probabilities of the two possible three character sets that straddle the boundary between C and D are examined. These character strings are BCD and CDE. The probability of the left three characters, ABC, is compared to BCD and CDE. This comparison is done by finding these probabilities in the look-up table of thousands of word occurrences, which is previously created during training. The same comparison is then done with the right side characters, DEF, comparing its probability of occurrence with the probabilities of the boundary-straddling characters, BCD and CDE. This process is illustrated in Figure 4.4.

A score is given to the proposed boundary based on the number of times the characters on the left and right sides have larger probabilities than the characters that span the boundary. Thus when the character strings on the right and left sides occur more times than character strings that span the boundary, it is likely that the boundary is the location at which a space should naturally occur. In other words, if the boundary is proposed between characters C and D, and the words ABC and DEF occur much more often than BCD or CDE, the boundary is most likely correct. However, before creating a space at this location, the score for the boundary is compared in a final step to scores of boundaries directly before and directly after the proposed space.

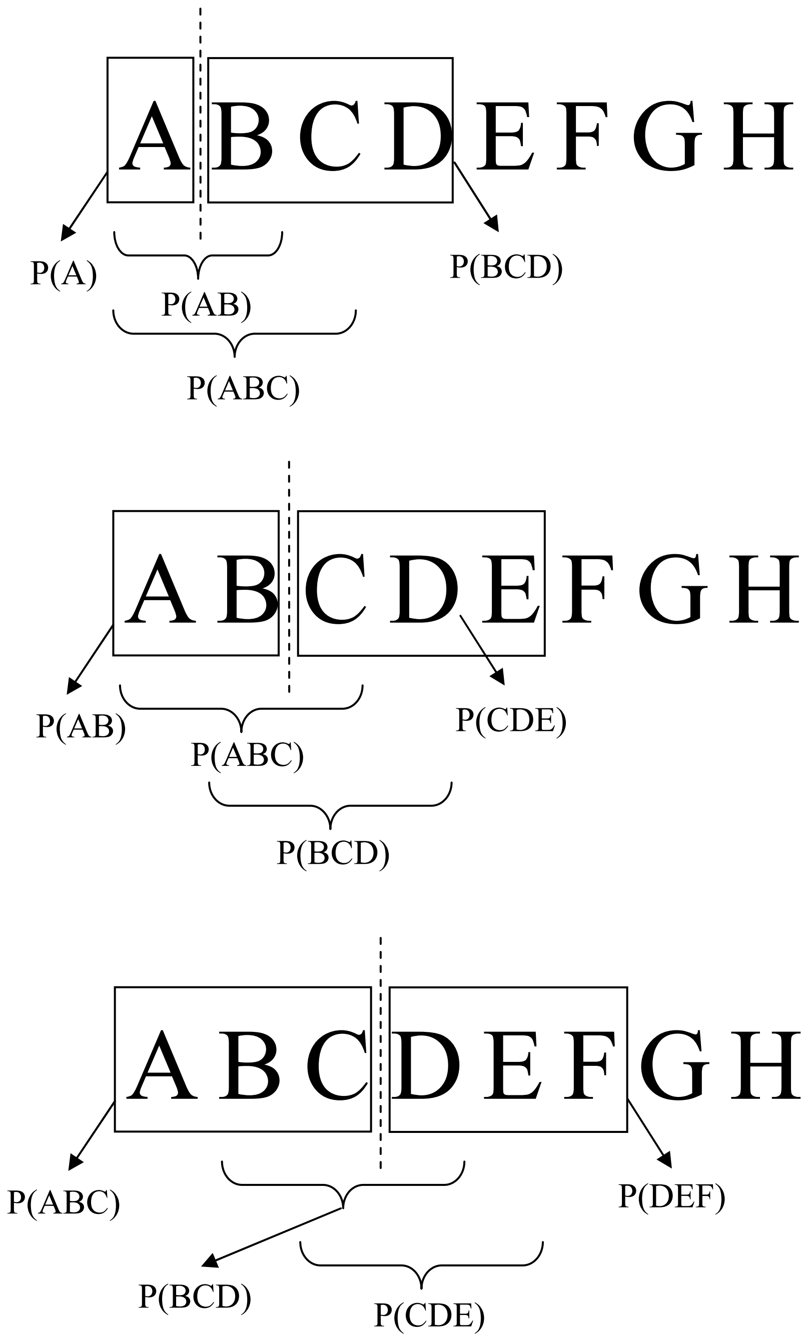

Figure 4.4. Example of n-gram analysis of Japanese character strings showing the first three steps.

This final step slides the length-3 window to the right by one character and examines the next possible boundary position, between D and E. A score is given to this boundary as before, and if the score for the boundary between C and D is higher than the score for the preceding boundary (between B and C) and also higher than the score for the following boundary (between D and E), then a space is placed into the Japanese sentence between characters C and D. If not, no space is placed there. This process is repeated for each potential space location.

A decision is needed on whether or not to give higher scores to more common occurrences or simply to work on a point by point basis. Ando and Lee provide a solution to this problem [1]. Using the TANGO (*T*hreshold *A*nd maximum *N*-*G*rams that *O*verlap) algorithm, they total the number of times the left and right n-sized windows have word occurrences larger than any windows that overlap the proposed word boundary. If in the previous example, ABC occurs more than BCD and more than CDE, the proposed boundary is given two points. If the right side word occurrence is greater than BCD but not greater than the occurrence of CDE, another point is added to the proposed boundary score. The equation used is

$$v_n(k) = \frac{1}{2(n-1)} \sum_{d \in \{L,R\}} \sum_{j=1}^{n-1} I > (\#(s_d^n), \#(t_j^n)), \tag{28}$$

where $v_n(k)$ is the preliminary boundary score, n is the order of the n-gram, I is a binary indicator (1 when inequality is true and 0 when false), s is the word on the left (or right) of the boundary, and t is the occurrence of the straddling words.

The score from 2- to 8-gram as n-gram length increases has the potential to increase dramatically because there are more comparisons to more straddling words in larger n-grams. Therefore, a weight is added to normalize scores from one n-gram type to another type. The TANGO algorithm takes the score found above and averages it with all n-gram types used for that particular boundary. In addition, a random component is added to the equation to account for stray words that appear as valid components of other valid words, but that are not meant to be taken in that manner in the particular context of the sentence. Thus, a Gaussian is added to account for uncertainty in the area of the sentence near the boundary. The equation used is

$$v_N(k) = \frac{1}{N}\sum_{n \in N} v_n(k) + N(0,1)\,, \tag{29}$$

where N is total n-gram order and N(0,1) is the standard normal distribution with zero mean and unit variance.

After finding the score for a particular proposed space location, a boundary is placed at the particular location, L if it is true that

$$v_N(L) > v_N(L-1) \quad \text{AND} \quad v_N(L) > v_N(L+1). \tag{30}$$

In this case L-1 and L+1 are locations one place to the left and right, which means that if the score of the boundary in question is larger than the score of both immediate neighbor boundaries, then a space is inserted at the current proposed boundary.

It is possible to perform word segmentation on a particular group of sentences then immediately perform word segmentation on the same sentences using a different length window. By doing this, larger or smaller words and tokens that are left unsegmented in the first pass, are more likely to be found on the second pass. The reason for this is that if a word is passed over and not segmented from its neighbors, it is because it fell outside of the size of the n-gram window. If a second pass using a larger window is used, the probability of a particular word being missed a second time must decrease [11].

The research here is unique because it deals with more realist Japanese text than other researchers have used. The closest research to this topic performed n-gram analysis on strings of *kanji* characters [1], which is not a common case. In colloquial Japanese found in newspapers, books, and magazines, strings are more likely to be composed of *kanji*, *hiragana*, and *katakana* and not just *kanji* alone. This more realistic type of data is what is used in experiments found here. Since this kind of data appeals to a much wider audience, the research found here is useful and unique.

4.4 Word Segmentation Results

Figures 4.5 and 4.6 show the results of the word segmentation experiments using one n-gram window and two n-gram windows, respectively. When using one n-gram window, the most successful method is to word segment using a 2-gram analysis. The result shows that correctly placed spaces between words occur 83.2% of the time, which is significantly better than the other n-gram lengths. When using two n-gram windows, doing n-gram analysis with a 2-gram followed by a 3-gram window performs even better, correctly placing spaces between words 94.4% of the time.

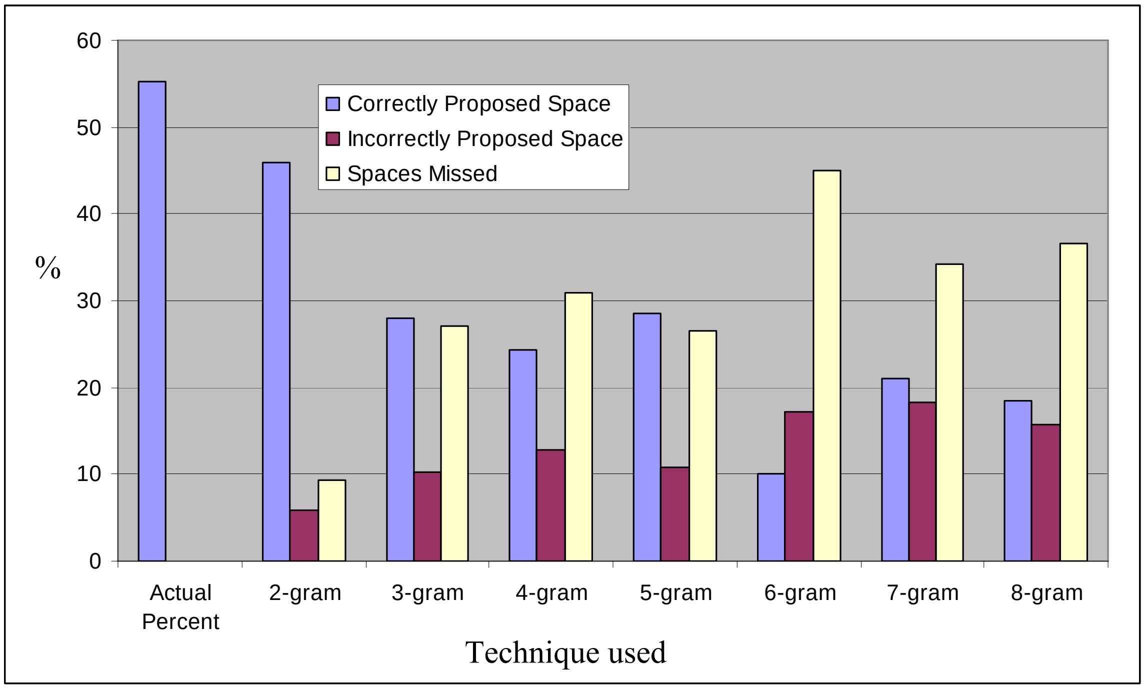

Figure 4.5. Probabilities of occurrences for 2-gram through 8-gram techniques compared to actual space probability. The occurrences are for correct space placement, incorrect placement, and missing spaces.

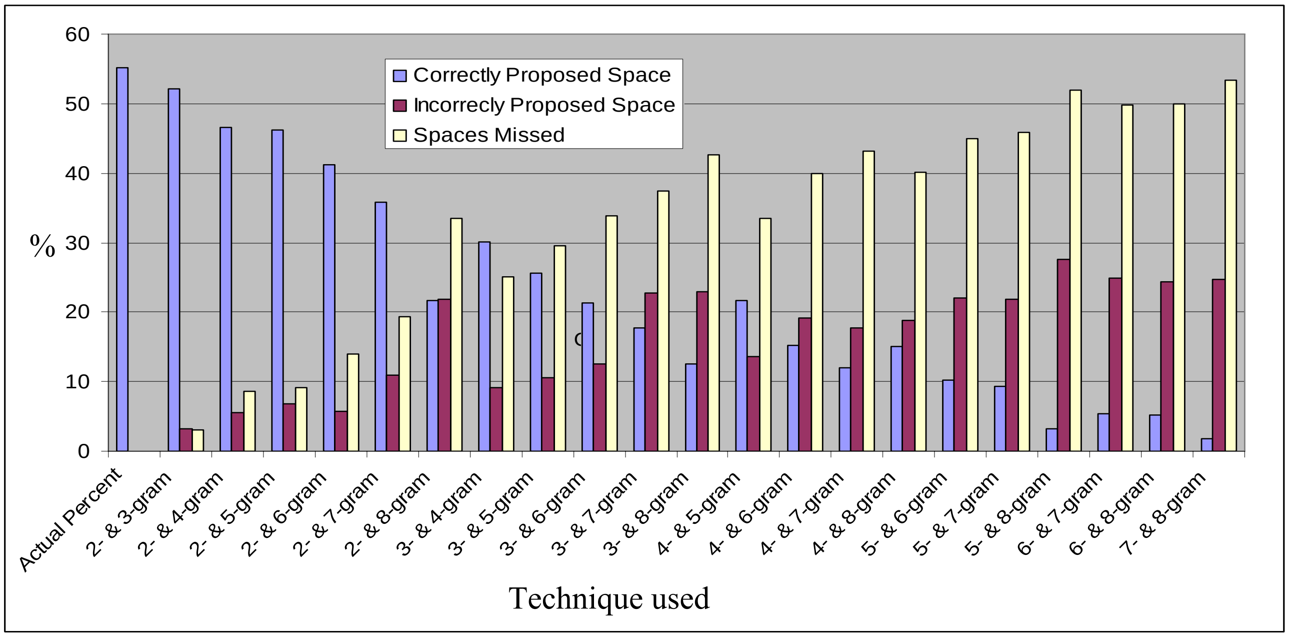

Figure 4.6. Probabilities of occurrences for combination pairs of 2-gram through 8-gram techniques compared to actual space probability. The occurrences, as before, are for correct space placement, incorrect placement, and missing spaces altogether.

V. Japanese Statistical Machine Translation

The previous chapter described methods for taking raw Japanese sentences without any word segmentation and adding segments between words. This addition is of course not necessary if the data used is properly word segmented in advance. Experiments described take place on segmented Japanese text where human translators have gone through the entire corpus and placed segments where they are required. Included in this section is information about the corpus used, steps performed to train the data, scoring of results, and information about the various experiments performed. Results from these experiments appear after a discussion of each experiment.

5.1 Corpus and Software

A parallel English-Japanese corpus is used. The language information is collected from the 2005 IWSLT test set. The training data consists of 20,000 sentences in both languages. Also, GIZA++ and Pharaoh, are used for alignment and decoding.

5.2 Training

The first step in training finds word alignments using GIZA++, which is a program that trains translation models from the English-Japanese parallel training data by implementing the IBM models 1-5 described in Section 2.4. This algorithm uses the Japanese and English training texts to align words and phrases in both an English-to-Japanese and a Japanese-to-English direction.

The next step extracts phrases from the word-aligned sentences created by GIZA++ and then Pharaoh decoder. Pharaoh takes the phrases from a phrase table after GIZA++ has aligned all sentences, and then decodes using this phrase table (translation model) and the language model to provide a translation.

5.3 Scoring of results

In order to evaluate the quality of English translations created by the system, it is customary to use a numeric method for scoring results. This was accomplished by comparing the artificial machine translations created by the SMT system to human translations in terms of n-gram matches using two such scoring methods, BLEU and NIST.

BLEU (Bilingual Evaluation Understudy) [22] scores range from 0 (worst) to 1 (best). compares a set of candidates (machine translations) to a set of references (human translations) by counting the number of matches between n-grams in both. As the number of matching n-grams increases, the BLEU score also increases. The default number of n-grams used in BLEU scoring is 4, so that the total score is based on the combined results of unigrams, bigrams, trigrams, and 4-grams.

BLEU scores are based on n-gram counts and are correlated with the human judgments of accuracy and fluency. In addition, a brevity penalty is given to translations to better judge the quality of each translated sentence Accuracy measures how often identical words are used in both the candidate and the reference sentences. If all of the words in the candidate have a matching word in the reference, the candidate is considered highly accurate. Fluency measures how well strings of words in the candidate match with strings of words in the reference. Finally, brevity gives a measure of the length of the candidate sentence. Examples of these three qualities are shown in Figure 5.1.

Reference
The hospital is down the street to the right.

Candidate 1
A clinic is down this street to the right.

Candidate 2
Street is hospital the to down the right.

Candidate 3
Hospital is down.

Figure 5.1 Example sentences showing accuracy, fluency, and brevity.

Candidate 1 is less accurate than the candidates 2 and 3, since candidate 1 has six of nine possible words that match those in the reference, while all of the words in both candidates 2 and 3 have matches in the reference. Since a human will most likely select candidate 1 as the best translation when compared to the reference, accuracy does not mean much in isolation. In terms of fluency, candidate 1 is scored highest since it has four matching bigrams ("is down," "street to," "to the," and "the right") versus one bigram match for candidate 2 and two matches for candidate 3. In addition, candidate 1 has two matching trigrams ("street to the" and "to the right") versus zero for candidate 2 and one for candidate 3 and one matching 4-gram ("street to the right") versus zero matches for the other two candidates. Finally, candidate 3 is the briefest of the three sentences and therefore has the worst brevity score. Again, none of these three qualities works well by itself, and they should instead be considered together to obtain a more accurate score.

The BLEU methodology accounts for these three attributes by first finding a precision score p_n which contains an accuracy and a fluency score, then augmenting it with a penalty for brevity, where p_n is

$$p_n = \frac{\sum_{C \in (Candidates)} \sum_{n-gram \in C} count_{clip}(n-gram)}{\sum_{C \in (Candidates)} \sum_{n-gram \in C} count(n-gram)} . \qquad (31)$$

Thus, p_n is found by totaling all n-gram matches between a given candidate sentence or candidate string and a reference divided by the total number of n-grams in the reference.

Once the precision score is found, the BLEU score is calculated by modifying p_n with a brevity penalty, BP. This penalty accounts for small sentences that may have high precision despite being poor translations. Examining Figure 5.1, it is immediately obvious that candidate sentence 3 is a poor translation since it conveys little of the meaning of the reference sentence. In terms of fluency, all three words appear in the reference translation, and in terms of accuracy both bigrams ("hospital is" and "is down") and the entire trigram string ("hospital is down") appear in the reference translation. Thus, despite being a poor translation, without some penalty for small sentence length, candidate 3 receives good precision and accuracy scores. The brevity penalty is

$$\text{BP} = \begin{cases} 1 & \text{if } c>r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}, \qquad (32)$$

where c is the length of the candidate sentence or string in words and r is the reference sentence length. The BLEU score is computed by summing all weighted precision scores and multiplying the result by the brevity penalty so that

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n \right), \qquad (33)$$

where N is the number of n-grams used (e.g., N=4 for unigram to 4-gram, N=3 for unigram, bigram, and trigram), and w_n is a weight equal to 1/N.

The NIST (National Institute of Standards and Technology) [19] score is similar to BLEU score, but instead uses the arithmetic mean of weighted n-gram values. NIST scores start at 0 as the worst score and increase as precision increases. The NIST scoring formula is

$$NIST = \sum_{n=1}^{N} \left(\frac{\sum Info(w_1 ... w_n)}{\sum (1)} \right) \cdot \exp\left(\beta \log^2 \left(\min\left(1, \frac{L_{sys}}{L_{ref}} \right) \right) \right), \tag{34}$$

where N is the n-gram size (default is N=5), β acts as a brevity penalty similar to that of the BLEU score, L_{sys} is number of scored words in the SMT translation, and L_{ref} is the number of reference translation words averaged over all reference translations.

5.4 Experiments

Here, multiple hypotheses are examined, and their BLEU and NIST scores are compared. In order to obtain baseline scores, 20,000 unaltered sentences are used for training. Each candidate score can be compared to this common set of scores to find whether changes result in improvements or reduced scores. The baseline BLEU score is 0.3967 and the NIST score is 6.7814.

The first experiment is based on creating uniform politeness levels (see Section 3.3.2) throughout the training and test data. The Japanese verbs *arimasu* (あります) and *gozaimasu* (ございます) have identical meanings but different formality levels. *Arimasu* is a plain form verb and is the English equivalent of the verb "to be," while *gozaimasu* is a much more formal version of this verb. After reviewing the training English translations, despite *arimasu* in some sentences and *gozaimasu* in others, there appears to be no difference in the formality of the corresponding English translations. Thus, the system is using two very common Japanese words to describe the same English word or words. It appears that reducing the formality of all sentences to normal plain form by replacing all occurrences of *gozaimasu* with *arimasu* will improve translation. The BLEU score for this experiment is 0.3973 and NIST score is 6.7494, both nearly identical to the baseline scores.

The second experiment eliminated the topic marker *wa* (は) in all sentences (see Section 3.3.3 for more on grammatical markers). There are approximately four thousand occurrences of this marker that are eliminated from the text data. Since Japanese is a SOV language, the subject (here topic) of the sentence almost always appears at the beginning of the sentence. The addition of a topic marker may be unnecessary in most sentences, and in fact is seen as an additional word, therefore adding more variability and confusion to a sentence that may otherwise be clear. The elimination may not work for every case because it is grammatically acceptable to delay using the topic, and therefore *wa*, until the end of a sentence, especially when the speaker wants to emphasize the topic. However, cases where the topic appears early in a sentence greatly outnumber cases where the topic is displaced. Therefore, removing the topic marker is advisable. The BLEU score is 0.4154, 2 points higher than the baseline, and NIST score is 7.1092, which is the best of all experiments.

The third experiment concerned the use of the particle *he* (へ). The particle always serves the same purpose as the locative component of the particle *ni* (に), although *ni* itself has many other grammatical uses than *he*. Thus, merely replacing all occurrences of *he* with *ni* in the training and test data may simplify many sentences that contain these particles, thereby increasing the overall quality of translation. There seems to be no need to have two particles for identical tasks. Slightly more than one thousand occurrences of the particle *he* are replaced with this experiment. One possible problem with this experiment is that, since *ni* has other grammatical uses, these uses can be attributed to *he*. The experiment's BLEU score is 0.4047, nearly a one point improvement over the baseline, and the NIST score is 6.8177.

The fourth experiment concerns the subject particle *ga* (が). As mentioned earlier, *wa* and *ga* are extremely close in meaning. The idea of topic and subject is difficult to specify for human translators and would most likely cause problems in SMT. Thus it may be desirable to eliminate as much confusion as possible between the two markers by eliminating occurrences of *ga*. If deleting all occurrences of *wa* from the training and test data improves translation quality, then it is reasonable to assume that eliminating the subject marker *ga* may also improve performance. Nearly five thousand occurrences of *ga* are deleted from the text for this experiment. The BLEU score is 0.4018, and the NIST score is 6.8455, with no significant improvements over the baseline scores.

The fifth experiment involves correcting problems with numbers in the Japanese data set. The numbers 1 through 9 appear many times in the data, each represented by a single *kanji* character. Although it can also be written as a single character (零), "zero" is instead written using two *katakana* loan characters, *ze* and *ro* (ゼ and ロ) throughout all of the Japanese sentences. Thus, to describe the string of ten numbers from zero to nine, eleven different characters are used. Many of the translated sentences output by the system which had numbers (in terms of time, money, etc.) have either poor alignment between the two languages or missing numbers in the English translation. For this reason, all occurrences of zero as two characters (ゼロ) in the Japanese training and test sets are replaced with the single character (零) for zero. The BLEU score is 0.3966 and the NIST score is 6.9759. The scores for this experiment show that no significant improvement is made by making the change from two characters to one.

The sixth experiment altered all verbs in the Japanese data sets to better deal with segmentation errors in the original text, make a uniform formality, and remove extraneous verb endings, among other aspects. This task is formidable, but it seems worthwhile because it has potential for a large impact on scores. An example of major changes made is shown in Figure 5.2. The first change separates all *–te* form verbs into their verb root and the *te* (て) particle. The *te* character is added on to a verb to allow for affixation of another verb. By separating verb roots from the *te* particle, the assumption is that the individual word parts are seen as "go and buy" versus "go buy" in English, for example.

Also, there is a segmentation problem with the verb-*mashou* form, where *mashou* (ましょう) adds volition to the preceding verb, as in "let's eat" or "let's go." The problem here is that in the text data *mashou* usually appears as *masho+ u* (ましょ and う) with a misplaced space between the two mora, and also as *mashou* (ましょう) with no space. To correct this problem, all occurrences of *masho+ u* are changed to *mashou*.

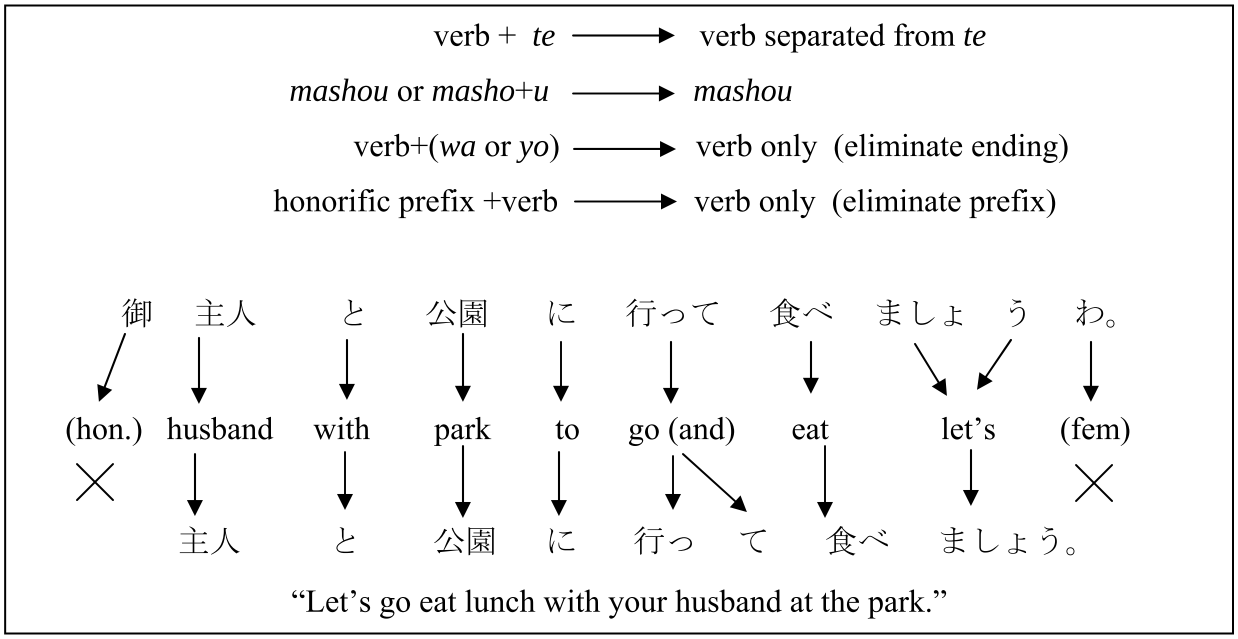
verb + *te* → verb separated from *te*
mashou or *masho*+*u* → *mashou*
verb+(*wa* or *yo*) → verb only (eliminate ending)
honorific prefix +verb → verb only (eliminate prefix)
御 主人 と 公園 に 行って 食べ ましょ う わ。
(hon.) husband with park to go (and) eat let’s (fem)
主人 と 公園 に 行っ て 食べ ましょう。
“Let’s go eat lunch with your husband at the park.”

Figure 5.2. Example sentence showing the multiple changes made in the sixth experiment. These include the elimination of feminine and conviction endings and honorific prefixes as well as dividing verbs from their *te* (and) forms and combining any of the incorrectly segmented imperative verb endings *mashou*.

The original text in both training and testing has honorific prefixes attached, seemingly at random, to verbs. These prefixes, as before, did not seem to carry over into the English sentences, so they are eliminated throughout.

Finally, stripping of two sentence final particles, *wa* (わ) and *yo* (よ), which are attached to verbs, is performed on all sentences. In both cases neither the feminine particle *wa* (not to be confused with the topic marker *wa*) nor the particle *yo* (marking strong assertion by the speaker) seem to have significant influence on the English translation. Note that *wa* may also be a sentence final particle spoken by men in Kansai (Osaka-Kyoto-Kobe) dialect. Regardless of the source, none of the corresponding English sentences gain meaning or change in dialect by having this particle attached. Thus, in Japanese sentences where *wa* is used, it quite obvious that a woman is speaking, but in the corresponding English there is never an indication that the speaker is a woman. In addition, when *yo* is used to show that the speaker is certain of what is said in a particular Japanese sentence, this strong conviction rarely translates to the corresponding English sentence. For these reasons, both particles are eliminated from the Japanese for this experiment. The BLEU score here is 0.4216, which was about 2.5 points higher than the baseline score. Unfortunately, the NIST score is 6.3126, lower than the baseline.

In addition to the various individual experiments attempted, combinations of each are tried where they make sense grammatically. For example, experiment 8 combines experiments 2 and 3 and determines if removing both the topic marker *wa* and replacing the locative marker *he* gives favorable results.

Experiments 9 and 10 show that despite their individual experiments making improvements on the baseline score, combining two experiments together does not mean that the resulting experiment will work well. The tenth experiment, for example, combines two experiments that performed well individually, and in the case of experiment 2, result in a very good NIST score, but perform poorly when combined.

Finally, to show how character segmentation performs compared to the word segmented baseline experiment, an eleventh experiment is performed. NIST and BLEU scores drop, but not considerably, in any of the character segmentation experiments. Results of all experiments performed can be found in Table 5.3.

Experiment	BLEU Score	NIST Score
Baseline w/20000 sentences of training data	0.3967	6.7814
1) Change *Arimasu*/*gozaimasu*	0.3973	6.7494
2) Deleted *wa*	0.4154	7.1092
3) Remove *he*	0.4047	6.8177
4) Remove *ga*	0.4018	6.8455
5) Alter zeros	0.3966	6.9759
6) Clean verbs and prefixes/suffixes	0.4216	6.3126
7) Remove *wa*, *ga*, and *he*	0.4024	6.9293
8) Remove *wa* and *he*	0.4436	6.5185
9) Remove *ga* and *he*	0.3809	6.9309
10) Remove *wa* and *ga*	0.3952	6.7384
11) Baseline with character segmentation	0.3940	6.7201
12) Experiment 1 with character segmentation	0.3822	6.7031
12) Experiment 2 with character segmentation	0.4057	6.7213
13) Experiment 3 with character segmentation	0.4020	6.7948

Table 5.3. SMT experiments performed with BLEU and NIST scores. Included are less successful experiments that fall below the baseline scores. Also included are results from character segmentation experiments.

VI. Japanese Speech Recognition

In translating from Japanese spoken data into English, a speech recognition component must be added to the overall system. Automatic speech recognition (ASR) systems allow for speech data to be output as text. In other words, the ASR component takes Japanese speech and outputs Japanese text to be used as the input to the SMT system already described.

This chapter describes the basics of Japanese ASR, including different methods for setting up the pronunciation dictionary and an explanation of sampling audio from Japanese speech. Conclusions are also made as to which method is best depending on word, syllable, and phoneme error rates. The SONIC program [23] performs speech recognition on audio samples from Japanese news broadcasts after training on pairs of audio samples and corresponding text transcriptions.

6.1 Automatic Speech Recognition

ASR systems rely only on training data for a priori knowledge. In additional, knowledge useful in training comes from applying known information about Japanese pronunciation and basic syllabary used as a written representation of sounds.

The speech recognition experiments conducted in this thesis examine how changing the speech recognition system pronunciation dictionary affects the error rates of output text. The data consist of pairs of Japanese text sentences mined from Japanese news and spoken versions of each sentence in audio format. The audio components are taken from native Japanese speakers who simply read each sentence out loud, as illustrated in Figure 6.1.

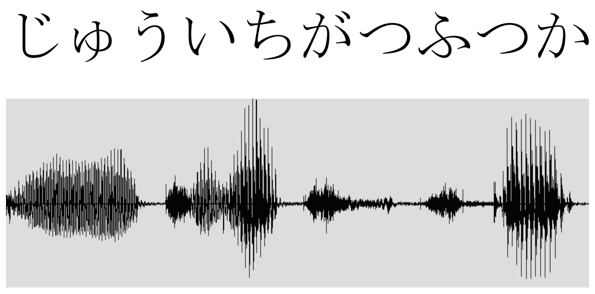

Figure 6.1. Sample from a Japanese text sentence ("November 2nd" in English) and the audio waveform of the sample spoken by a native Japanese speaker.

There are roughly 10,000 sentences of Japanese text in the GlobalPhone database [27] used in the ASR experiments conducted here, and the total time of all sound clips is about 28 hours. About 25 hours are used for training, and the remaining 3 hours are used for testing. The data are as follows: Training data are a total of 90,028 seconds of speech with 116 speakers (31 female and 85 male), and testing data are a total of 11,316 seconds of speech with 11 speakers (3 female and 8 male).

The speech and text data used here is from the GlobalPhone database, originally created to especially accommodate researchers working on large-vocabulary multilingual speech recognition [27]. In practice, acoustic models need at least 100,000 spoken words and language models need at least 200,000 spoken words to achieve reliable probabilities. There is an average of 10 spoken words per sentence in the GlobalPhone database. In addition, there are 200 sentences per speaker, and 127 speakers, for a total of approximately 254,000 spoken words. The number of Japanese syllables per sentence is 52.1 and the total number of syllables (or individual Japanese kana characters) is more than 1.3 million.

GlobalPhone has native Japanese speakers reading excerpts from the Nippon Keizai newspaper, which consists of mostly similar-article reporting styles and, therefore, has a uniform grammar and linguistic style, which results in a large speech database with corresponding transcripts.

An overall view of the speech recognition system is found Figure 6.2. In the following sections each of the components is described in detail.

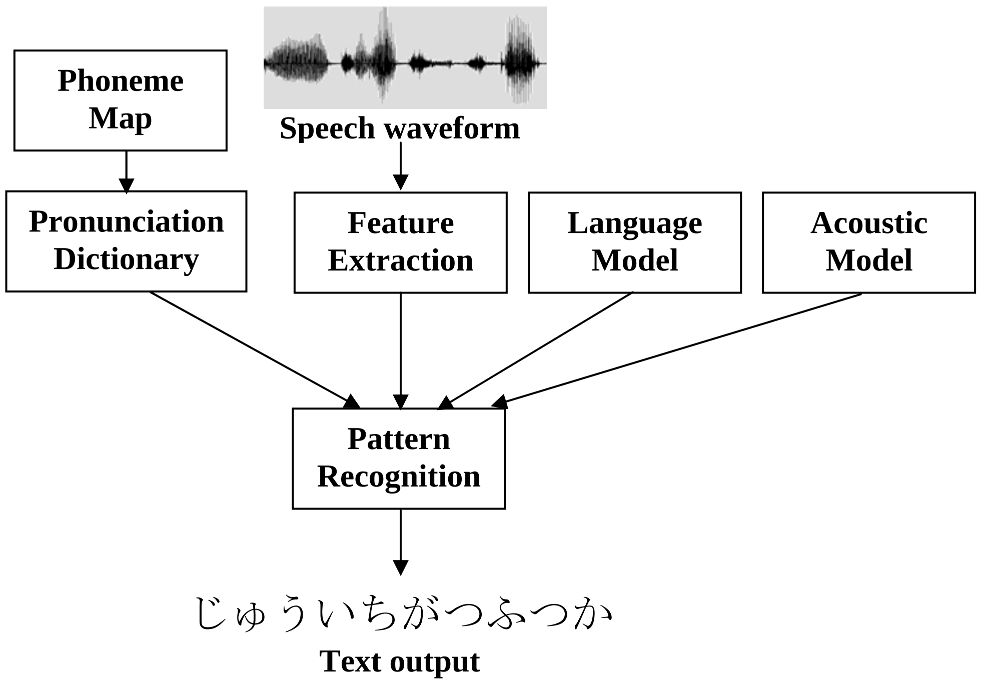

Figure 6.2. Diagram of the main features of the trained speech recognition system

6.2 Phoneme Map

The system must be trained with all possible phonemes, or basic word sounds. For example, if the Japanese word *watashi* is spoken, it may be phonologically displayed as /W AA T AA SH IY/, where each of the six elements between the slashes is a written representation of the phonemes that make up the word.

Figure 6.3 shows how the individual waveform that represents the sound of the word being spoken can be considered as a collection of smaller waveforms, each of which represents a phoneme. For the system to be able to recognize the first block in Figure 6.2 as /JH/ instead of one of the other phonemes, it is necessary to train the system with audio that has a large number of each of the phoneme waveforms. Thus, when a waveform is encountered that has features close to those of the segment on the far left, the system approximates this sound as /JH/.

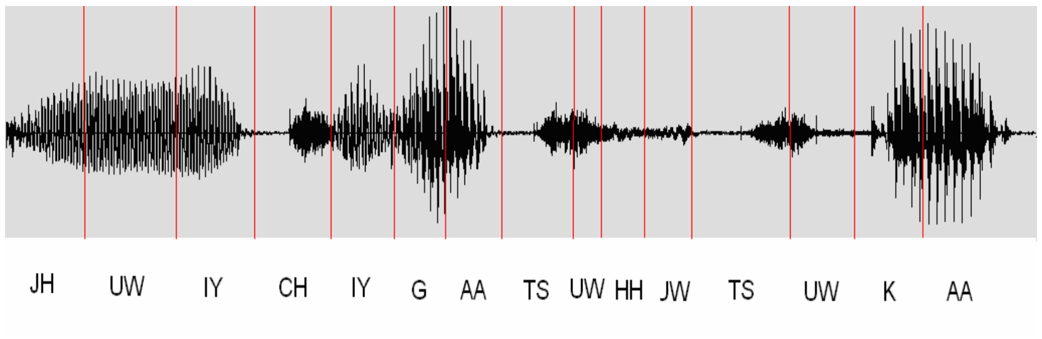

Figure 6.3. Japanese audio waveform broken down into smaller waveforms, each corresponding to individual phonemes.

A second pre-training step maps Japanese text to the different phonemes shown in Table 6.4. Since Japanese is composed of an entirely different alphabet than English, it is necessary to define how each Japanese *kana* "letter" is assigned to the phonemes in Table 6.4. One *kana* can represent one or two phonemes. A chart showing the possible assignment of *kana* to phonemes is in Figure 3.1.

The complete phonetic symbol set in Table 6.4 is the set of sound representations that the SONIC software uses to define each sound.

Phone	**As in**	**Phone**	**As in**	**Phone**	**As in**
AA	f***a***ther	EY	m***a***te	OY	b***oy***
AE	h***a***d	F	***f***ate	P	***p***in
AH	h***u***t	G	***g***ale	R	***r***eal
AO	f***or***	HH	***h***ead	S	***s***orry
AW	d***ow***n	IH	***i***s	TS	gu***ts***
AX	***a***lone	IX	p***e***n	GD	ha***g***
AXR	bett***er***	IY	w***ee***	SH	***sh***e
AY	fire	BD	la***b***	T	***t***ell
B	***b***ear	DD	ma***d***	TH	***th***row
CH	***ch***ill	KD	wal***k***	UH	g***oo***d
D	***d***ine	JH	***j***ust	UW	c***oo***l
PD	to***p***	K	***k***ill	V	***v***iolin
TD	go***t***	L	***l***ong	W	***w***ell
DX	le***tt***er	M	***m***an	Y	***y***es
DH	***th***ose	N	***n***o	Z	***z***ip
EH	h***ea***d	NG	si***ng***	ZH	mea***s***ure
ER	b***ir***d	OW	m***o***le	SIL	silence

Table 6.4. Symbols used by SONIC to represent 51 possible phoneme sounds

Suppose that the system is to correctly align the string of characters わたし with its audio waveform. The first step determines how these three characters sound if spoken. In this case, わ is a written representation of the phoneme sounds /W/ and /AA/, た is /T/ and /AA/, and し is /SH/ and /IY/. The system is then trained with all English approximations of the 51 phonemes and their approximate phoneme features, and thus the word わたし in Japanese text is broken down into a string of six individual audio waveforms. The exceptions to the pronunciations of the syllables found in Figure 3.1 are that し is pronounced /SH IY/, じ is /JH IY/, ち is /CH IY/, つis /TS UW/, and を is /OW/.

6.3 Pronunciation Dictionary

The pronunciation dictionary holds the training data words found in the total body of written texts and their corresponding pronunciation and is based on the phoneme map. It is created from text words not yet encountered. For example, if the word わたし is encountered for the first time, it is placed in the dictionary as is. Then the individual characters are referenced in the English phone set in Table 6.4, so that わたし is paired with the pronunciation /W AA T AA SH IY/. Thus, each time the system encounters the word わたし it knows the basic pronunciation and the overall audio waveform for the word based merely on the six phonemes that make up the text word. An excerpt from the pronunciation dictionary is shown in Figure 6.5.

.	
.	
.	
さびしい	**/S AA B IY SH IY/**
さみしい	**/S AA M IY SH IY/**
さむい	**/S AA M UW IY/**
しずかな	**/SH IY Z K AA N AA/**
.	
.	

Figure 6.5. Example of possible entries in the pronunciation dictionary. The left side is the Japanese word as it is found in the text. Only one instance is necessary for the dictionary, so even if the word is extremely common it only appears once. The right side is the pronunciation of the word based on the way the phone map was originally configured. If the configuration of the phone map is altered between one character and one phoneme sound, the pronunciations in this dictionary changes for thousands of words that contain that sound.

6.4 Feature Extraction

This component forms feature representations of each audio waveform. Figure 6.6 is an overview of the feature extraction process.

The first step in processing the audio signal applies a low-pass filter with a cutoff frequency of 8 kHz. An 8 kHz bandwidth is sufficient to carry all human speech information. Next, the signal is digitally sampled at 16 kHz. A pre-emphasizing filter is then applied to the audio signal to remove glottal source effects. A first-order FIR filter,

$$H(z) = 1 - \alpha \cdot z^{-1} \tag{35}$$

is used, where α is close to 1 (SONIC uses α = 0.97 as a default value) and z is a one sample delay.

Next, the spectrum magnitude of the audio signal is found from the DFT of the signal. A scale change is then made on the frequency axis to the Mel scale frequency by

$$f_M = \frac{1000}{\log 2}(1 + \frac{f}{1000}), \tag{36}$$

where f is the real frequency (Hz) and f_M is the Mel frequency (mels). The Mel (short for melody) scale represents the human auditory frequency perception better than the linear frequency scale of the Fourier transform.

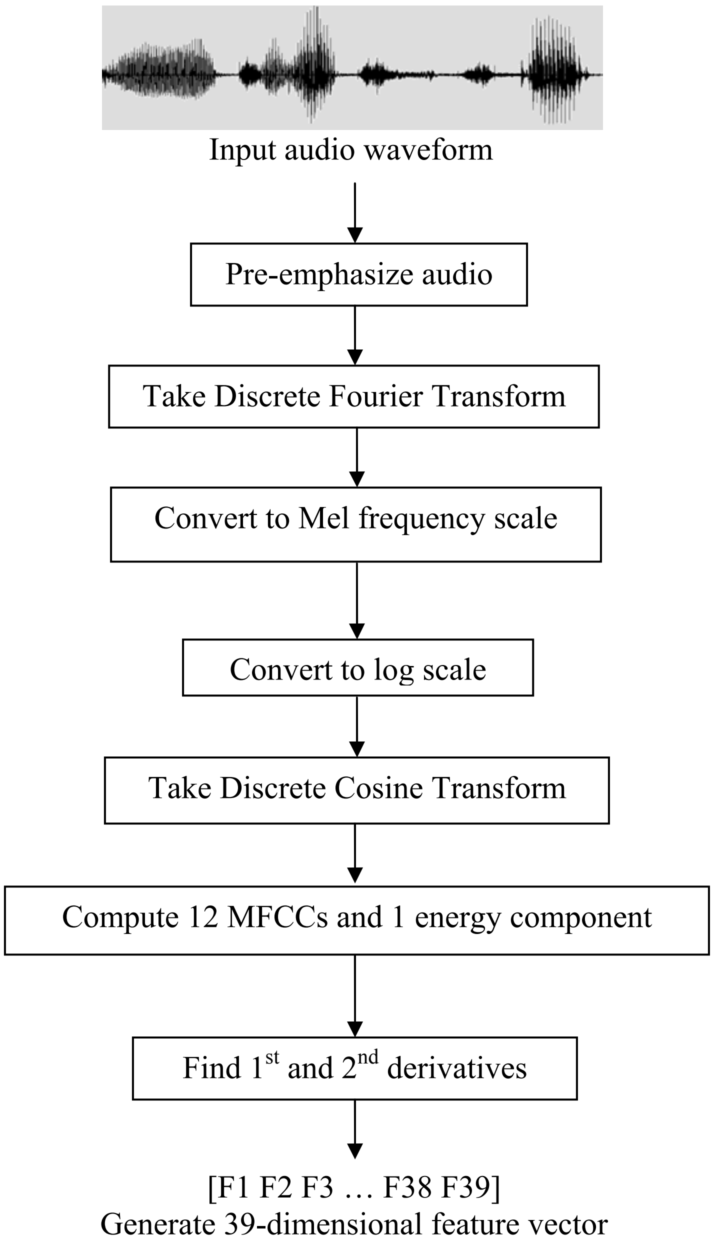
Input audio waveform
Pre-emphasize audio
Take Discrete Fourier Transform
Convert to Mel frequency scale
Convert to log scale
Take Discrete Cosine Transform
Compute 12 MFCCs and 1 energy component
Find 1st and 2nd derivatives
[F1 F2 F3 … F38 F39]
Generate 39-dimensional feature vector

Figure 6.6. Overview of technique to find the Mel Frequency Cepstral Coefficients as features of the speech recognition system

This new spectrum is passed into a set of triangle filters that divide the Mel frequency scale into a number of regions, e.g., a default value of 12 regions in SONIC. Filtering yields a set of 12 log-energy terms, e[1] through e[12],

$$e[j] = \log_2(\textstyle\sum_{k=0}^{N-1} w_j[k] \cdot |S_{MEL}[k]|), \tag{37}$$

where j is the number of terms from 1 to 12 , N is the value of an N-point DFT, $w_j[k]$ is the weight of the j^{th} filter to the k^{th} frequency of the sampled signal s(n), and $S_{MEL}[k]$ is the magnitude of the same signal in the Mel frequency scale.

The mel frequency cepstral coefficients (MFCCs) result from taking the Fourier transform of the log of the Fourier transform. The discrete cosine transform (DCT) is taken instead of the FT, but the classification as cepstral coefficients still applies. Each of the 12 Mel Frequency Cepstral Coefficients (i = 1 to 12) is found using the DCT of the energy terms

$$c^*[i] = \sqrt{2/P}\sum_{j=1}^{P}[e[j] \cdot \cos(\frac{\pi \cdot i}{P}[j - \frac{1}{2}])], \tag{38}$$

where P is always assumed to be 12. After the DCT is found, the mean is eliminated from each of the features to find the final values of each of the coefficients, i.e.,

$$c[i] = \frac{1}{T}\textstyle\sum_{t=1}^{T} c^*[i]. \tag{39}$$

The result is a 12 dimensional feature vector, and to each vector a thirteenth element is added to hold the log energy term of the sampled signal. This term is found by taking the log of all summed squared samples from 1 to the number of samples in the frame N of audio in question, i.e.,

$$c[13] = \log_2(\textstyle\sum_{n=1}^{N} s^2(n)). \tag{40}$$

The final step finds 13 first and 13 second order derivatives of the 13 elements of the vector to generate a 39 dimensional feature vector.

6.5 Language Model

The language model creates a reference for probabilities of each individual word. Similar to SMT, ASR uses Bayes' rule,

$$P(W \mid O) = \frac{P(O \mid W) \cdot P(W)}{P(O)}, \tag{41}$$

where O is the observed sequence of feature vectors extracted from sampled speech and W is the proposed sequence of speech that corresponds to the string of observed feature vectors. Also, P(W) is the language model, and P(O|W) is the acoustic model. Speech recognition finds the best sequence of words given some observed feature vectors. The estimate for a word sequence that maximizes P(W|O) is

$$W_{est} = \arg\max_W \frac{P(O \mid W) \cdot P(W)}{P(O)} = \arg\max_W P(O \mid W) \cdot P(W). \tag{42}$$

The language model gives the probability that a phrase occurs in the target language. Similar to its use in SMT, the language model serves to decrease the likelihood of low probability word sequences output by the recognizer.

6.6 Acoustic Model

The acoustic model, P(O|S) from Bayes' rule, maps acoustic features to distinct phonetic representations by modeling each phoneme sound with a set of Hidden Markov Model (HMM) states. As stated before, a 39-dimensional feature vector is created for each sample. The acoustic model uses these vectors to assign a mixture of Gaussian distributions to represent each HMM state. The likelihood of any particular emission from a given state is

$$P(O \mid S) = \sum_{m=1}^{M} \frac{w_m}{(2\pi)^{\frac{D}{2}} \sqrt{\prod_{d=1}^{D} \sigma^2{}_m}} \cdot \exp(-\frac{1}{2}\sum_{d=1}^{D}\frac{(O_t - \mu_m)^2}{\sigma^2{}_m}) \tag{43}$$

where D = 39 dimensions, M is the number of Gaussians distributions, w_m are weights that allow all Gaussians to sum to one, and $\sigma^2{}_m$ is the diagonal covariance matrix found during the feature extraction phase. Since the underlying Gaussian distributions are continuous and since the HMMs representing encountered phonemes consist of a mixture of many of these distributions, the overall HMM structure used by the acoustic model is said to be continuous.

A string of phonemes is represented by HMMs as shown in Figure 6.7. Each phoneme HMM contains three states, one for the start, middle, and end of each phoneme.

The pattern recognition portion of the system brings together the pronunciation dictionary, feature extraction, language model, and acoustic model modules of the system and generates an optimal text sequence. Thus it attempts to find the string of words and characters that best matches the probabilities found by the other parts of the system. This task is accomplished using a Viterbi decoding algorithm to search through every possible recognition choice.

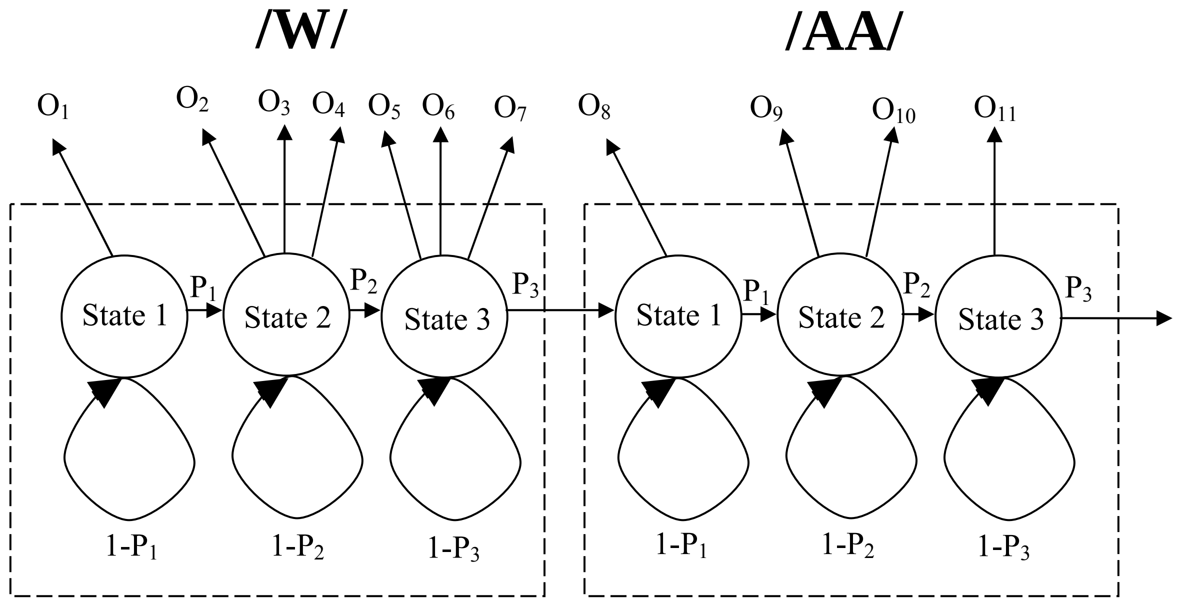

Figure 6.7. Example of phonemes with underlying Hidden Markov Models. The observations (in this case O1 through O11) vary depending on HMM state transitions, use of monophone, diphone, or triphone models, and string of sounds in question. This example is for the first two phonemes of the spoken word *watashi*. If encountered again, /W/ and /AA/ may have completely different observations.

6.7 Automatic Speech Recognition Experiments

The following hypotheses on actions that may affect speech recognition routines are based on knowledge of the Japanese language:

- Clean all speech of fragments
- Use all speech including fragments
- Change postalveolar flap /r/ in all occurrences to alveolar lateral approximant /l/
- Change postalveolar flap /r/ in all occurrences to voiced alveolar plosive /d/
- Replace palatal fricative /h/ from /hu/ with labio-dental fricative /f/ sound
- Replace velar stop /g/ in /ga/ with nasal /n/

The postalveolar flap /r/ sound found in Japanese speech is a mixture of both the alveolar lateral approximant sound /l/ found as the first letter of the word "letter" and the voiced alveolar plosive sound /d/ found as the first letter in the word "dad." For this reason, substituting /r/ for /l/ and /r/ for /d/ seem to be suitable choices. In addition, the syllable /hu/ in Japanese is often approximated as an /f/ sound as in the word “far,” and /ga/ is also approximated by a nasal sound /n/ similar to the sound /ng/ in the word “song.” These approximations explain the final two hypotheses.

6.8 Default training data

Fragments are parts of sentences that contain mistakes by the speakers. For example, if the sentence “I like to take long walks” is misspoken as “I love…,” it is considered a fragment. Japanese speech data fragments and their corresponding texts are kept along with the correctly spoken sentences. Thus, each time an incorrectly spoken sentence is kept, the amount of acoustic training data increases.

Although the uncleaned acoustic training data improves word error rates, the hypothesis is that for syllable and phoneme error rates such data is actually detrimental because many of the mistakes made when reading sentences change the statistics on individual syllables. For example, by halting in the middle of a word, a single syllable can be introduced into the total dictionary that is not meant to appear. In addition, if the goal of the overall system is to output text to an SMT, the language model statistics can also be altered significantly by broken words and incorrectly spoken syllables, thus changing the efficiency of the SMT system. Therefore, using the unfragmented default training data for all other hypotheses may be advantageous.

Table 6.8 shows the numeric results from the six ASR experiments. The first experiment where fragmented speech is left in the language data yields nearly identical results for each error as the second experiment with deleted speech fragments. In terms of word error rates (WER) and their components, the default error rates are best, with /hu/→/fu/ WER the best of all alternatives. Replacing /r/ with /l/ and /r/ with /d/ performs poorly because five different sounds are replaced for each hypothesis, which allows for deletion, insertion, and substitution errors at more possible points. The hypotheses that substitutes only one sound for one other sound has better performance in every type of error rate, but still underperforms when compared to the two default hypotheses. None of the experiments is successful in improving Japanese ASR performance.

	#1 (Fragment)	#2 (No fragment)	#3 (L)
Deletion	4.0 ± 1.0	4.1 ± 1.0	5.3 ± 1.2
Insertion	3.7 ± 1.7	3.9 ± 1.8	8.2 ± 2.2
Substitution	20.1 ± 6.3	20.4 ± 6.2	33.2 ± 6.0
Total	27.8 ± 8.2	28.3 ± 8.0	46.7 ± 8.0

	#4 (D)	#5 (HU)	#6 (Nasal)
Deletion	5.8 ± 1.5	3.0 ± 0.4	3.9 ± 0.8
Insertion	6.6 ± 1.4	4.4 ± 0.7	4.6 ± 1.2
Substitution	41.4 ± 2.3	21.2 ± 1.9	26.7 ± 4.5
Total	53.8 ± 5.1	28.6 ± 2.8	35.2 ± 6.4

Table 6.8. Mean and standard deviation of deletion, insertion, substitution, and totoal word error rates (in percent) for the test speakers

VII. Results and Conclusions

This chapter summarizes the results from experiments in the word segmentation, SMT, and ASR parts of this research and discusses future work and possible ways to improve results found here.

7.1 Word Segmentation Results

Results from Tables 4.5 and 4.6 show that smaller n-gram windows work best to segment Japanese newspaper text data. When used alone, 2-gram analysis results in spaces at 45.9% of all possible locations, whereas the actual space probability from hundreds of sentences is 55.2%. In addition, 2-grams yields for incorrect placement of spaces 5.8% of the time. The result for 2-grams space probability is about twice as favorable as any of the other n-gram results. Also, in using a 2-gram window along with a 3-gram, the space probability increases to 52.1%, and incorrect space classification decreases to 3.1%. Thus, 94.4% of all segments are found by the system using this method.

The reason for the smaller n-gram windows working best may be that the average Japanese word is only a few characters long. Larger n-gram windows, therefore, tend to skip over entire groups of words and particles and only find longer individual words, while smaller n-gram windows are more likely to catch each occurring word larger or small.

7.2 Results from SMT experiments

Table 7.1 shows the BLEU scores from the baseline and three best SMT experiments compared to highest scoring experiments performed by participants in the 2005 IWSLT campaign.

Experiment	BLEU Score
Baseline w/20000 sentences of training data	0.3967
2) Deleted *wa*	0.4154
6) Clean verbs and prefixes/suffixes	0.4216
8) Remove *wa* and *he*	0.4436
ATR-C3	0.4770
Microsoft	0.4060
ATR-SLR	0.3880
University of Tokyo	0.3720

Table 7.1. Comparison of best SMT experiment results to best 2005 IWSLT results

The three best BLEU scores in order are experiments 8, 6, and 2. In comparison only the ATR-C3 experiment resulted in better BLEU scores. Each of these three experiments score higher than the Microsoft, ATR-SLR, and University of Tokyo experiments. This appears to be good considering that aside from Microsoft, the groups conducting SMT research are all Japanese groups.

Experiment 8 scores over two points higher than 6 and nearly five points higher than the baseline experiment. Because of this result and since the experiment is relatively easy to perform, Experiment 8 seems to be the best choice of the eight experiments. In terms of NIST scores, Experiments 6 and 8 do not perform as well. Instead Experiment 1 is best followed by Experiment 5.

The ratio of time to performance must be taken into account when performing these experiments. An experiment like 2, where a single particle is removed from the data, takes little time and can be accomplished by means of a simple algorithm. Experiment 6, on the other hand, takes considerable time to set up, and since its performance in terms of BLEU score was close to that of 2, it may not be worth the effort to implement a complex algorithm to clean the training and testing data sets.

Some of the worst performing experiments under both scoring methods are Experiments 9 and 10, both failing to do better than the baseline scores. The reason may be grammatical. As mentioned earlier, merely deleting or replacing a particle, as in these cases, may lead to confusion by the system. In the case of the third experiment, it is possible that the system is aligning some of the other grammatical meanings for the particle *ni* with the replaced locative particle *he*. In other words, it appears that a meaning like "when" or "toward" may be used when "to" is the only possible meaning of *he*. This result is, however, rather confusing since the final experiment performs well and has experiment 3 as a component. In addition, removing both the topic and subject particle proves unproductive. This is probably due to the fact that both topic and subject can appear in the same sentence, and without a marker telling which is which, it is likely that they can be switched. Therefore, as expected Experiment 10 performs poorly.

7.3 Results from ASR experiments

Comparing the first and second experiments shows that there is no significant difference in any of the error rates when using either fragmented or unfragmented speech. Since the language data yields nearly identical results, the remaining four experiments use fragmented speech since it allows for the use of slightly more training data. When examining the experiments in terms of word error rates (WER), the /hu/$\rightarrow$/fu/ experiment performs the best, nearly matching both of the default experiments. All of the other experiments perform much worse. One of the reasons for this may be the relatively small test size. With only nine speakers, there may not be enough acoustic speech data to get accurate results. Another problem is that dialect is neglected in the ASR experiments. The speakers used for testing may be of a different dialect subset of speakers, and because of this may very well have dialectical differences from the training speakers than can affect the performance of the system. This can be examined in the future experiments.

It is interesting to note that the experiments do not perform poorly in all aspects. The major problem seen in Table 6.8 comes from the fact that substitution errors are quite high when compared to the two default experiments. What this might mean is that the system is not properly trained and that incorrectly substituted phonemes occur more often than they should because of this poor training. This may be another area for improvement in the future.

The graphs found in the Appendix show the results for the six ASR experiments. The graph for each experiment shows error rates for deletion, insertion, and substitution errors as well as total word error rate. The continuous densities found in these graphs represent the data points from each of the experiments. Each probability density is found by using unimodal Gaussian Parzen windows [7], where a Gaussian is located at each of the nine data points, the variance is increased until only one peak appears in the overall distribution, and the curve is standardized so that it has unit area. Using these graphs is useful for giving an idea of where each kind of error falls in relation to other errors.

7.4 Future Research

It may be of interest to compare results found here with a complete speech recognition/machine translation system that takes Japanese speech in audio format, changes it into Japanese text transcriptions, and finally outputs English text (or speech). Since the text data into the machine translation component of this new system is based on imperfect speech recognition outputs, scores are likely to decrease significantly. However, in the interest of speech-to-text technology and the ability to interpret speech into another language without human help, this comparison could be a good next step.

In the ASR experiments, it may be of interest to look at the effects of dialect on results. For example, substituting a nasal /g/ sound for a regular /g/ sound only appears in the Japanese language for some dialects. Since the speakers are randomly chosen without regard to dialect, this substitution does not affect all cases. The same is true for the substitution of /fu/ for /hu/. In both cases, if speaker dialects are considered in the training and test data, both of these hypotheses may show improvement over either of the default cases, which may be a subject for future research.

Finally, comparing the BLEU scores from the SMT experiments with scores from the 2005 IWSLT evaluation campaign shows that the experiments here work quite well. The eighth experiment has a score of 0.4436, which is higher than all scores in the standard evaluation and all but one score in the evaluation using supplied language data plus tools[8]. Adapting the experiments used here with the techniques used in the IWSLT campaign may be a good next step in further improving scores.

Appendix

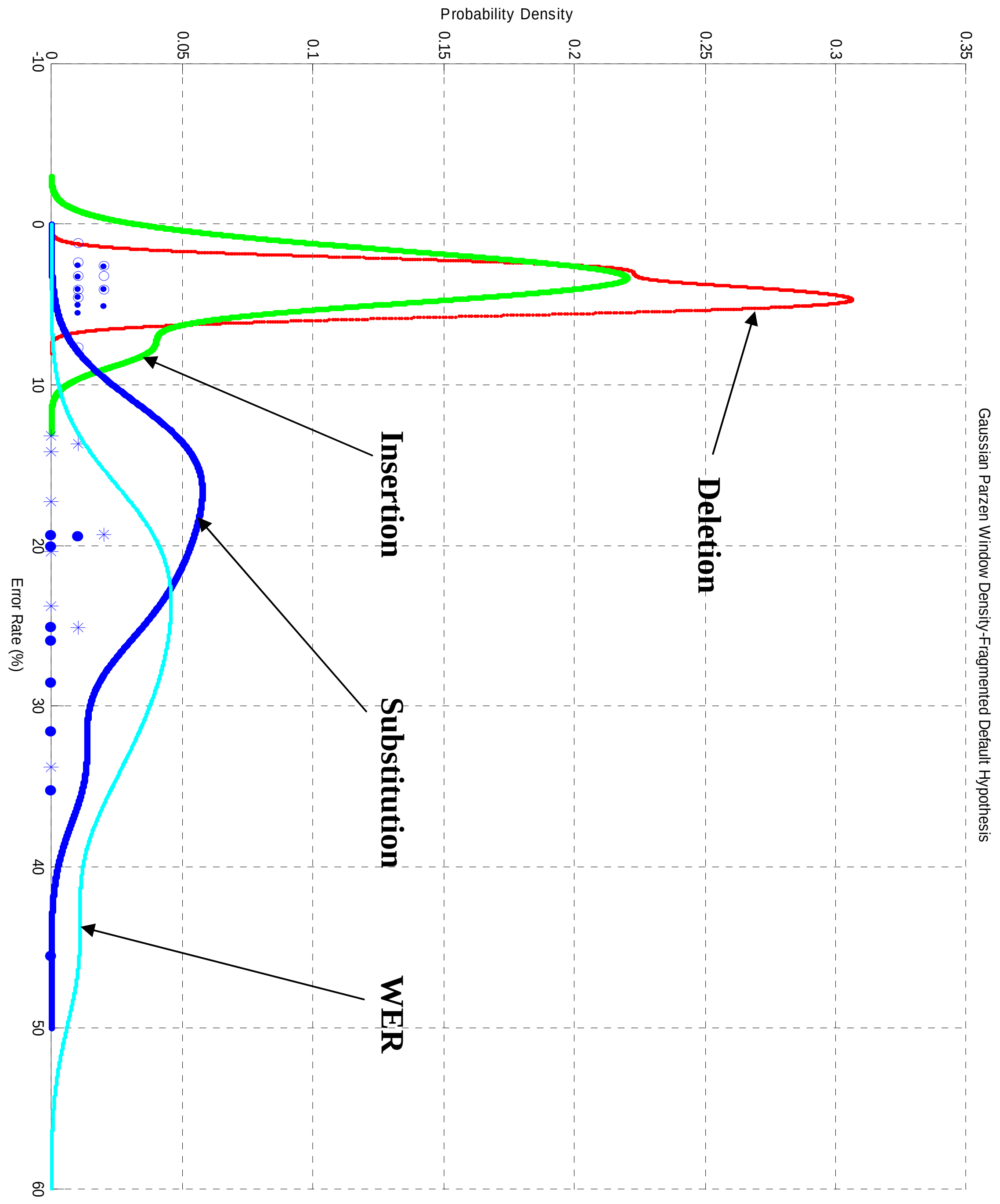

Figure A1. Deletion, insertion, substitution, and total word error rate for the default hypothesis with fragments

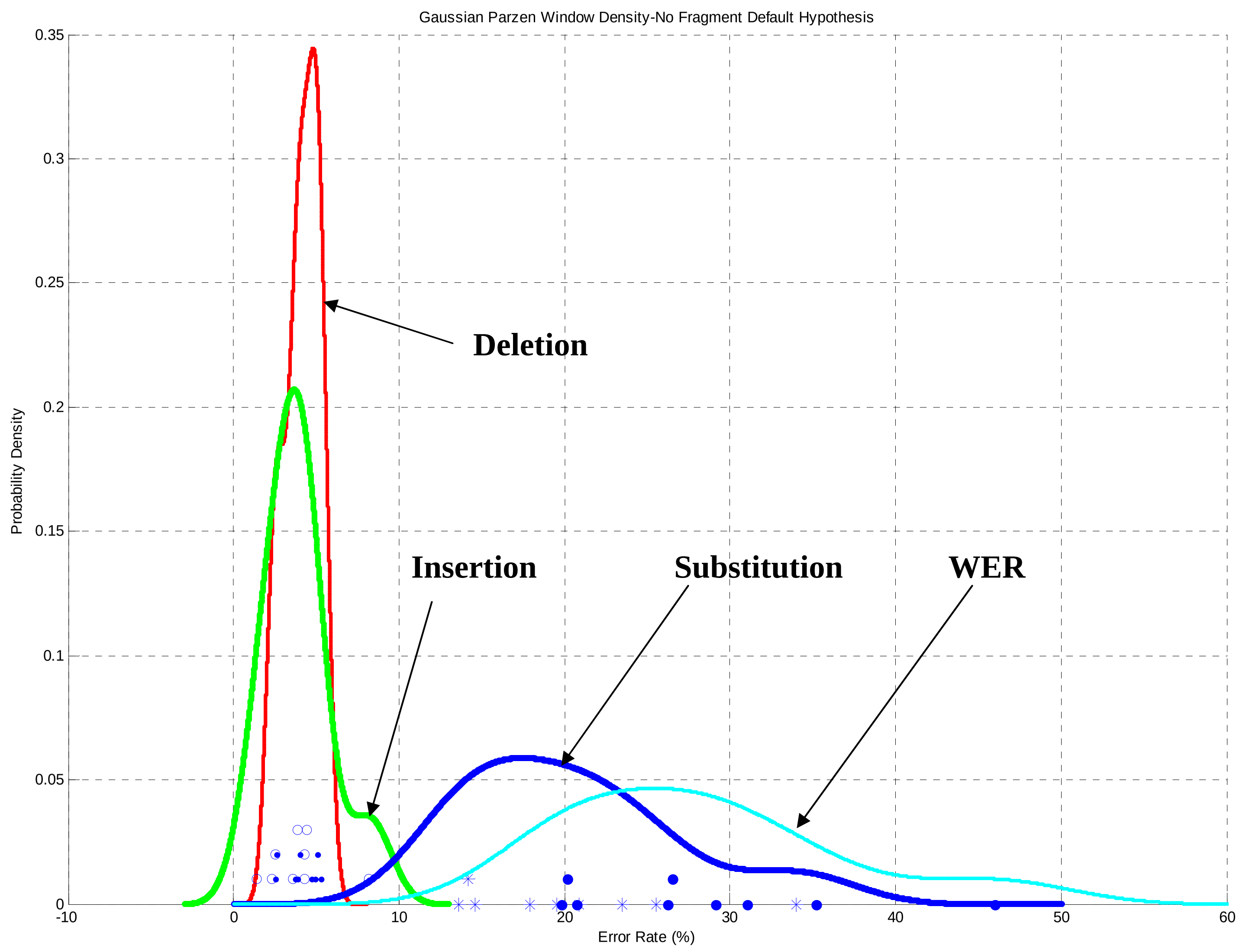

Figure A2. Deletion, insertion, substitution, and total word error rate for the default hypothesis with no fragments

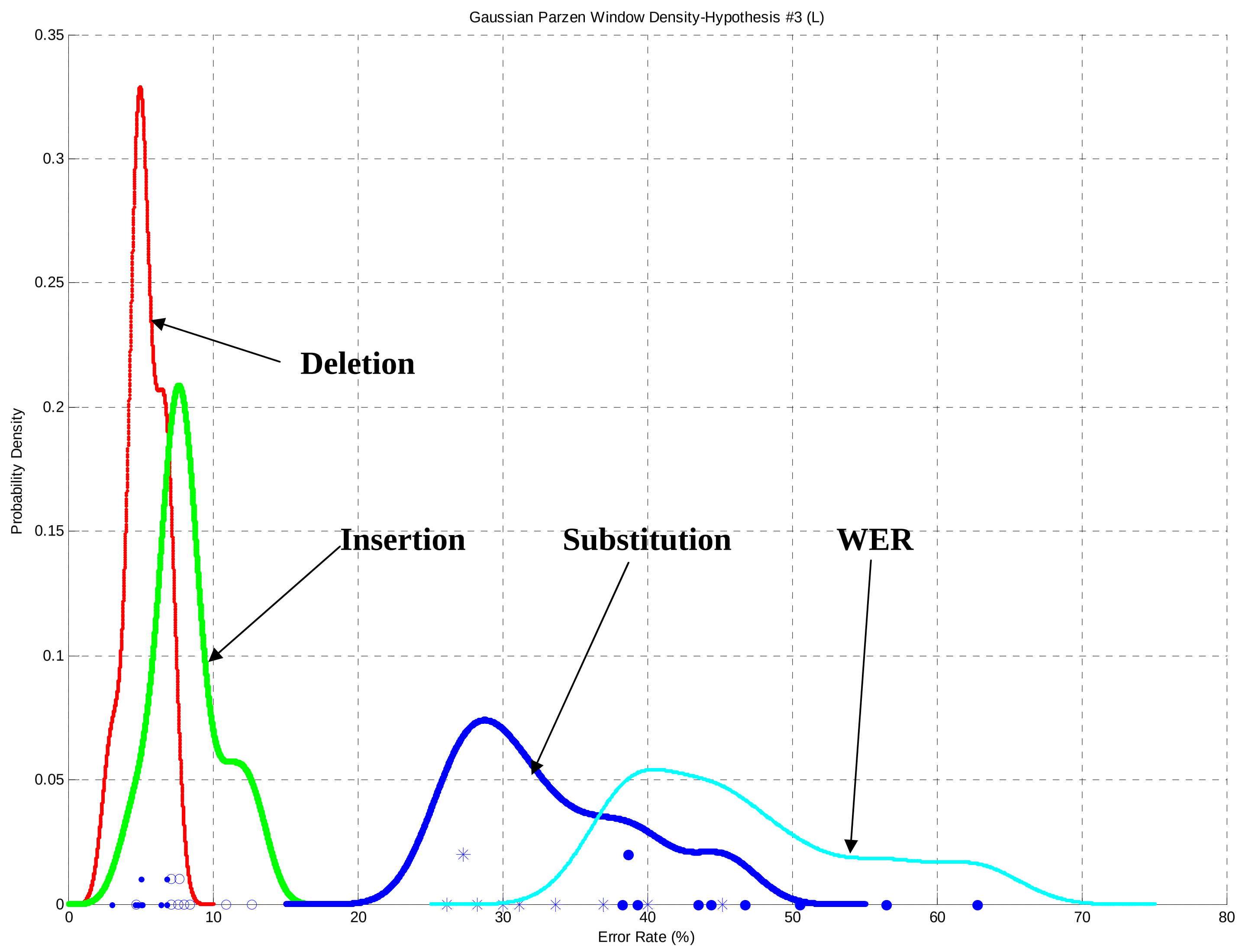

Figure A3. Error rates for the hypothesis replacing all /r/ sounds with /l/ sounds. The deletion error rate is about 25 percent higher than the default rate, the insertion error rate is doubled compared to the default, and the substitution error rate is approximately the same as the default.

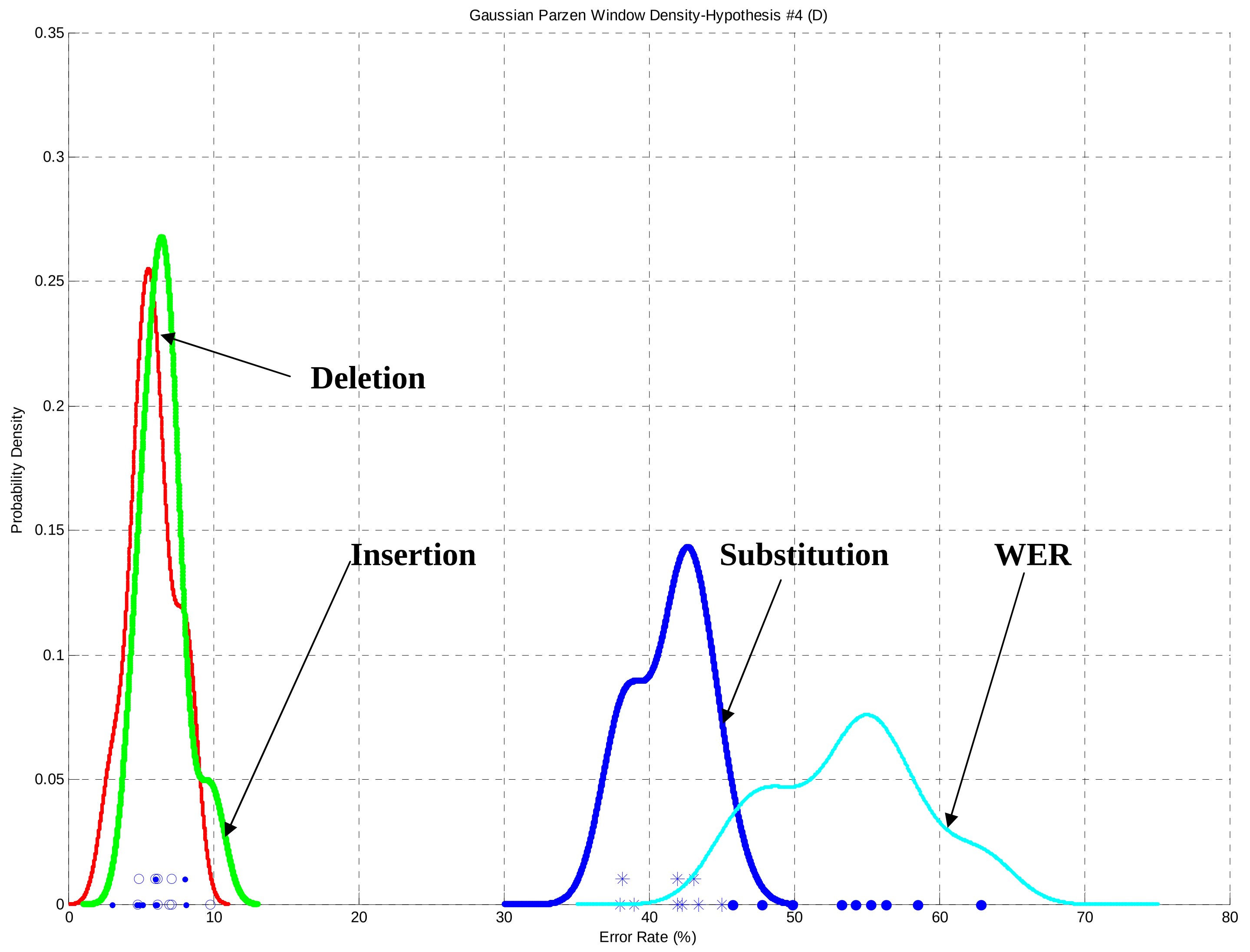

Figure A4. Error rates for the /r/ $\rightarrow$ /d/ hypothesis. The deletion error rate is worse than the /r/ $\rightarrow$ /l/ hypothesis, and the substitution errors are the highest of all experiments. Replacing with /d/ may be a problem since /d/ already exists for many words. The WER is very high because of large substitution errors.

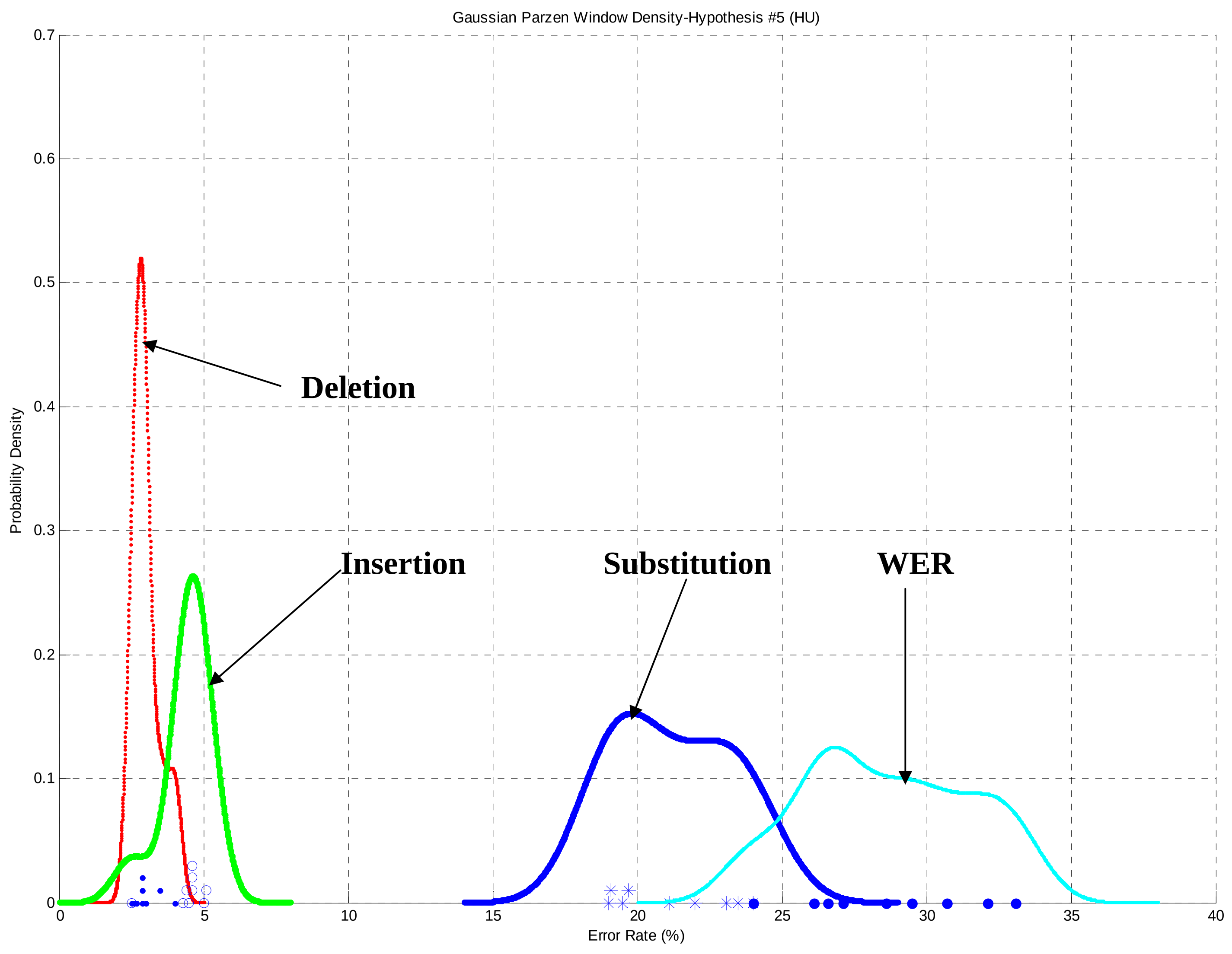

Figure A5. Error rates for the /hu/→/fu/ experiment. The deletion error rate is about half that of the default hypothesis, and the insertion error rate is close to that of the default. The substitution error rate is slightly higher than the default substitution rate. The total WER for the /hu/→/fu/ hypothesis is about 15% lower than the default WER.

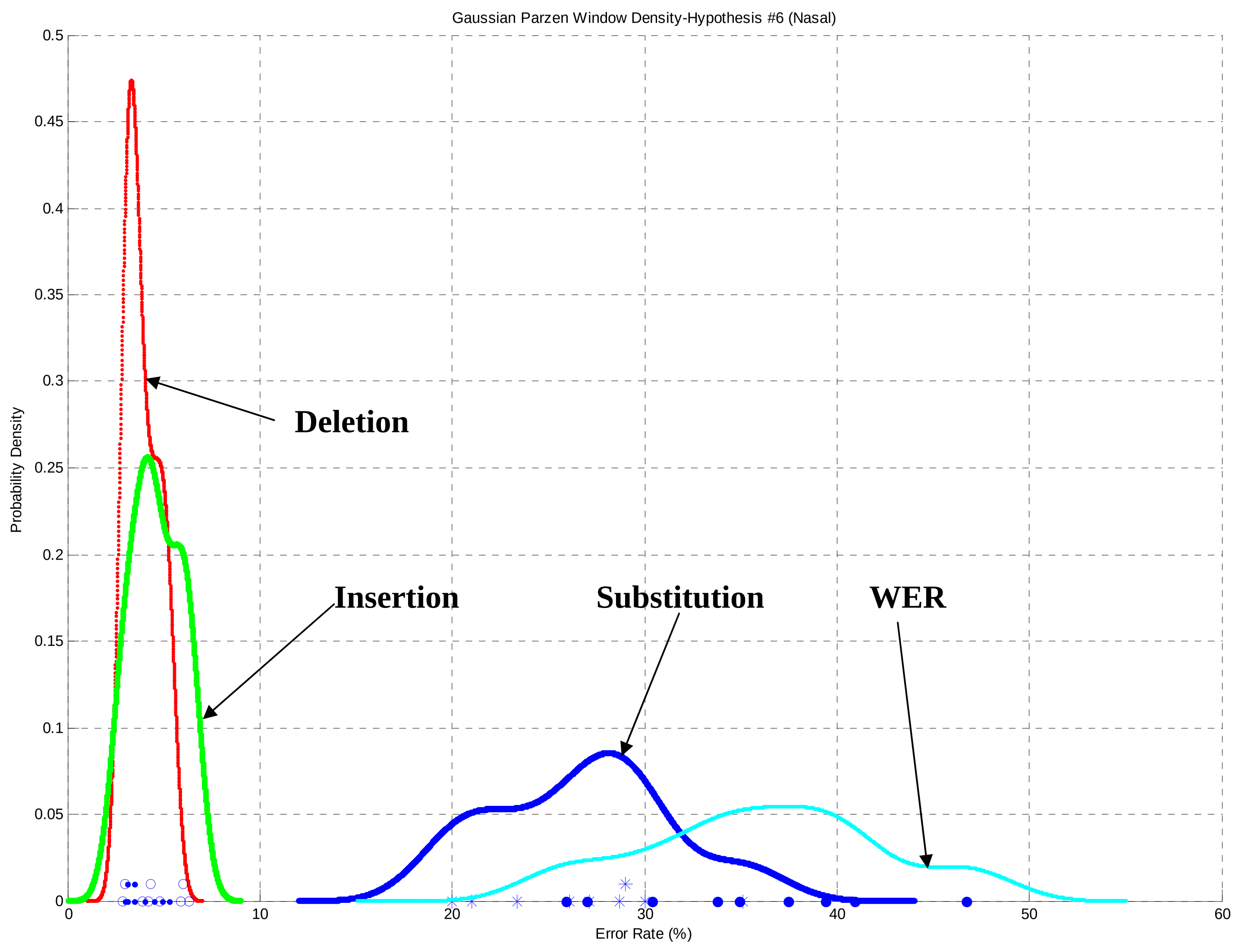

Figure A6. Error rates for the nasal hypothesis (replacing /g/ with nasal version of /g/ sound). As with the /hu/→/fu/ hypothesis, the deletion errors are significantly lower than the default. The insertion error rate is higher than the default, but not significantly, and the substitution error rate is much higher than the default. The WER is higher than the default, so overall this hypothesis is not used over the default or the /hu/→/fu/ hypothesis.

Bibliography

1. R. Ando and L. Lee. Mostly-unsupervised statistical segmentation of Japanese: applications to kanji. In *First Conference of the North American Chapter of the Association for Computational Linguistics*, 2000.

2. I. Bazzi and J. Glass. A multi-class approach for modeling out-of-vocabulary words. In *Proceedings of the International Conference on Spoken Language Processing*, 2002.

3. C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.

4. G. Bristow, *Electronic Speech Recognition: Techniques, Technology, and Applications*, McGraw Hill, New York, 1986.

5. P. Brown, S. Pietra, V. Pietra, and R. Mercer. The mathematics of statistical machine translation: Parameter estimation. In *Computational Lingustics*, 1993.

6. S. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. In *Computer Speech and Language*, 1999.

7. R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley and Sons, Inc. New York, NY, 2001.

8. M. Eck and C. Hori. Overview of the IWSLT 2005 Evaluation Campaign. In *Proceedings of IWSLT*, 2005.

9. S. Furui, *Digital Speech Processing, Synthesis, and Recognition*, Tokyo Institute of Technology, 2001.

10. A. Girardi and C. Kelly. Mainichi Shimbun newspaper study, 2006. Available at http://ftp.monash.edu.au/pub/nihongo/00INDEX.html

11. A. Ito and M. Kohda. Language modeling by string pattern n-gram for Japanese speech recognition. In *Proceedings of the International Conference on Spoken Language Processing*, 1996.

12. P. Koehn. Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. User Manual. *Technical report, USC Information Sciences Institute*, 2004

13. P. Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2004.

14. K. Lee, *Automatic Speech Recognition*, Kluwer, 1989.

15. Y. Lee. IBM Statistical Machine Translation for Spoken Languages. In *Proceedings of International Workshop on Spoken Language Translation*, 2005.

16. S. Makino and M. Tsutsui. *A dictionary of basic Japanese grammar*. The Japan Times Press, Tokyo, 1999.

17. C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2003.

18. M. Nagao and S. Mori. A new method of n-gram statistics for large number of n and automatic extraction of words and phrases from large text data of Japanese. In *Proceedings of the International Conference on Computational Linguistics*, 1994.

19. NIST. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. Available at http://www.nist.gov/speech/tests/mt/doc/ngram-study.pdf.

20. F. J. Och. GIZA++: Training of statistical translation models. Available at http://www.fjoch.com/GIZA++.html

21. C. Papageorgiou. Japanese word segmentation by hidden Markov model. In *Proceedings of the Human Language Technology Workshop*, 1994.

22. K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. *Technical report, IBM T.J. Watson Research Center*, 2001.

23. B. Pellom and K. Hacioglu, SONIC: The University of Colorado Continuous Speech Recognizer," *Technical Report*, *Center for Spoken Language Research*, University of Colorado, 2005.

24. B. Plannerer. An Introduction to Speech Recognition. *SMG Workshop*, 2000.

25. L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of IEEE*, Vol. 77, No. 2, 1989.

26. W. Shen, B. Delaney, and T. Anderson. The MIT-LL/AFRL MT System. In *Proceedings of International Workshop on Spoken Language Translation*, 2005.

27. T. Shultz, The GlobalPhone Project, 2006. Available at www.cs.cmu.edu/~tanja/ GlobalPhone

28. Systran Translation Software. Available at http:// http://www.systransoft.com/

29. L. Tomokiyo and K Ries. An automatic method for learning a Japanese lexicon for recognition of spontaneous speech. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1998.

30. Y. Zhang and S. Vogel. Measuring confidence intervals for the machine translation evaluation metrics. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, 2004.

REPORT DOCUMENTATION PAGE					*Form Approved OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 22-03-2007	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* Jun 2006 – Mar 2007
4. TITLE AND SUBTITLE Statistical Machine Translation of Japanese		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Chapla, Erik, A., GS-07		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Bldg 640 WPAFB OH 45433-7765		**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/07-06
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The purpose of this research was to find ways to improve the performance of a statistical machine translation system that translates text from Japanese to English. Methods included altering the training and test data by adding a prior linguistic knowledge, altering sentence structures, and looking for better ways to statistically alter the way words align between the two languages. In addition, methods for properly segmenting words in Japanese text through statistical methods were examined. Finally, experiments were conducted on Japanese speech to produce the best text transcription of the speech.

The best statistical machine translation methods implemented resulted in improvements that rivaled the best evaluations from the 2005 International Workshop on Spoken Language Translation from which training and test data was used. Recommendations, including how the methods presented may be altered for further improvements for future research, are also discussed.

15. SUBJECT TERMS
Statistical Machine Translation, Automatic Speech Recognition, Japanese Language, International Workshop on Spoken Language Translation, Statistical Word Segmentation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT** UU	**18. NUMBER OF PAGES** 96	**19a. NAME OF RESPONSIBLE PERSON** Dr. Stephen Gustafson (ENG)
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U			**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636; e-mail: Stephen.Gustafson@afit.edu

Standard Form 298 (Rev: 898)
Prescribed by ANSI Std. Z39-18