ON IMPROVING NATURAL LANGUAGE PROCESSING THROUGH PHRASE-BASED AND ONE-TO-ONE SYNTACTIC ALGORITHMS

by

CHRISTOPHER HENRY MEYER

B.S., Kansas State University, 2004

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2008

Approved by:

Major Professor
Dr. William H. Hsu

Copyright

CHRISTOPHER HENRY MEYER

2008

Abstract

Machine Translation (**MT**) is the practice of using computational methods to convert words from one natural language to another. Several approaches have been created since MT's inception in the 1950s and, with the vast increase in computational resources since then, have continued to evolve and improve. In this thesis I summarize several branches of MT theory and introduce several newly developed software applications, several parsing techniques to improve Japanese-to-English text translation, and a new key algorithm to correct translation errors when converting from Japanese kanji to English. The overall translation improvement is measured using the BLEU metric (an objective, numerical standard in Machine Translation quality analysis). The baseline translation system was built by combining Giza++, the Thot Phrase-Based SMT toolkit, the SRILM toolkit, and the Pharaoh decoder. The input and output parsing applications were created as intermediary to improve the baseline MT system as to eliminate artificially high improvement metrics. This baseline was measured with and without the additional parsing provided by the thesis software applications, and also with and without the thesis kanji correction utility.

The new algorithm corrected for many contextual definition mistakes that are common when converting from Japanese to English text. By training the new kanji correction utility on an existing dictionary, identifying source text in Japanese with a high number of possible translations, and checking the baseline translation against other translation possibilities; I was able to increase the translation performance of the baseline system from minimum normalized BKEU scores of .0273 to maximum normalized scores of .081.

The preliminary phase of making improvements to Japanese-to-English translation focused on correcting segmentation mistakes that occur when attempting to parse Japanese text into meaningful tokens. The initial increase is not indicative of future potential and is artificially high as the baseline score was so low to begin with, but was needed to create a reasonable baseline score.

The final results of the tests confirmed that a significant, measurable improvement had been achieved through improving the initial segmentation of the Japanese text through parsing the input corpora and through correcting kanji translations after the Pharaoh decoding process had completed.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would first like to thank Dr. William Hsu for guiding me with infinite patience down the path of Artificial Intelligence, Natural Language Processing, and Machine Translation. I became interested in these topics during the early college years because of his teachings and due in no small part to his excitement, passion, and skill in the subjects.

I would also like to thank Dr. Masaaki Mizuno and Dr. Scott DeLoach for agreeing to serve on my committee. It has been a long road to make it here, and they have never faltered in their personal effort, attention, or interest in seeing this thesis complete.

To the Graduate Studies Committee and to the professors of Kansas State University – thank you for your flexibility, allowing me time to finish my work, and for your personal effort to teach material to prepare your students for the outside world. It is a big playground out here, and the lessons learned during the college years are serving well.

A special thanks goes to Teja Pydimarri. His advice on the subject of Natural Language Processing and the implementation of his thesis on Part-of-Speech Tagging proved to be invaluable in surmounting similar challenges encountered along the way.

Finally; heartfelt thanks to my wife, parents, sister, and family for supporting me, encouraging me, and believing in me every day. Your support has meant everything to me through missed dates, absent evenings, and sleepless nights at the computer. I promise not to be a ghost from now on.

CHAPTER 1 - Introduction

Machine Translation (MT) provides a computational aid to human communication and a test bed for subsidiary and related human language technologies such as language-specific parsing tools, word-level semantics, and evaluation metrics based on reference translations. This thesis explores the problem of improving the phrase-based approach to statistical machine translation (SMT) by incorporating sources of semantic information, such as dictionaries, and also by developing improved parsing tools for a specific translation task. The dictionary approach helps us to resolve contextual ambiguity in phrase-based statistical MT, while the parsing tools improve the baseline performance of an end-to-end Japanese-to-English MT system developed as part of this thesis research. Building an entire MT system capable of translating input from a source language text to a target language text contains several components, each of which is a small MT system in-and-of itself. These components, assuming an input of text instead of speech or sign language, are generally broken down into 3 groups by current research standards.

The first part of a standard MT system involves reading in several source/target language sentence pairs and aligning matching text in both languages. This process is called “Text Alignment” and results in word-to-word and phrase-to-phrase mappings stored in alignment files. Using the same reference sentence pairs; it is also possible to build a language model in the form of a table of probabilities of words occurring in a certain order. The table produced by this process is called a “Language Model” and is the second essential component in any MT system. The final element in our MT system is the process of combing both the text alignment mapping and the language model, and applying the combined learning from both input to a sentence in the source language. The actual process of using a Text Alignment file and a Language Model to convert words from a source-to-target language is called “Decoding” and produces translated text that is the best-possible match the MT system being utilized can provide.

After these three sub-components are explained further, I then explain the BLEU metric and how it can be used to measure the translation quality of any MT system. With an

objective scoring metric used to judge the quality of the MT system output and through comparing the MT system's BLEU score to other MT systems (such as Google Translate or Yahoo Babelfish), I am able to establish reasonable baselines for both lowest and highest measures of performance. The principal claims of the thesis will then be explained in detail to build the foundation of understanding for the thesis methodology and results.

1.1 Text Alignment

Text alignment is the process of aligning corresponding words in parallel sentences written in two different languages. Several Statistical Machine Translation (**SMT**) toolkits are available that contain word alignment applications (Egypt, Moses, GenPar, and MTTK to name some), but Egypt was chosen due to the writer's familiarity with the application and the proven accuracy of the SMT system. Contained within the SMT Toolkit Egypt is an application called Giza++ which is designed specifically for aligning parallel text. The following subsections describe the text alignment process and the Giza++ approach to creating alignment files that serve as an input to the SMT system.

1.1.1 Parallel Corpora

Text alignment applications rely on having pre-aligned corpora in two languages (one language provides text in the source language, and the other provides text in the target translation language). The definition of pre-aligned corpora in this context is illustrated by this excerpt from the Tanaka Corpus which shows a sentence ID on one line, followed by a sentence written in Japanese, followed by the English translation. This pattern continues resulting in over 200,000 parallel sentence pairs (Breen, 2008):

Figure 1.1 An Excerpt from the Tanaka Corpus

```
¥1¥100
¥2¥「ニューヨークへ行ったことがありますか。」「ええ２・３度行ったことがあります」
¥3¥"Have you ever been to New York? " "Yes, I've been there a couple of
times."
¥1¥101
¥2¥「終わったの」「それどころかまだ始めてもいないよ」
¥3¥"Have you finished?" "On the contrary I have not even begun yet."
¥1¥102
¥2¥「終わったの」「それどころかまだ始めていないよ」
¥3¥"Have you finished?" "On the contrary, I have not even begun yet?"
```

As seen in Figure 1.1, the Tanaka Corpus is not a perfect compilation of translations. Sentence ID 102 contains an incorrect question mark at the end of the English translation; errors such as these are common in parallel corpora and must be accounted for. Aside from minor grammatical mistakes; these sentences are pre-aligned in a one-to-one format and thus are easily converted to be an input for most text alignment applications. Some other corpora consist of large blocks of text with no one-to-one sentence relationship (for instance, the French-English translation of the Constitution), and these texts must be analyzed by a human and reordered in a fashion that a text alignment application can understand for accurate alignment results.

The second parallel corpus that will be used is the Kanji Dictionary corpus. An excerpt from the Kanji dictionary is shown below:

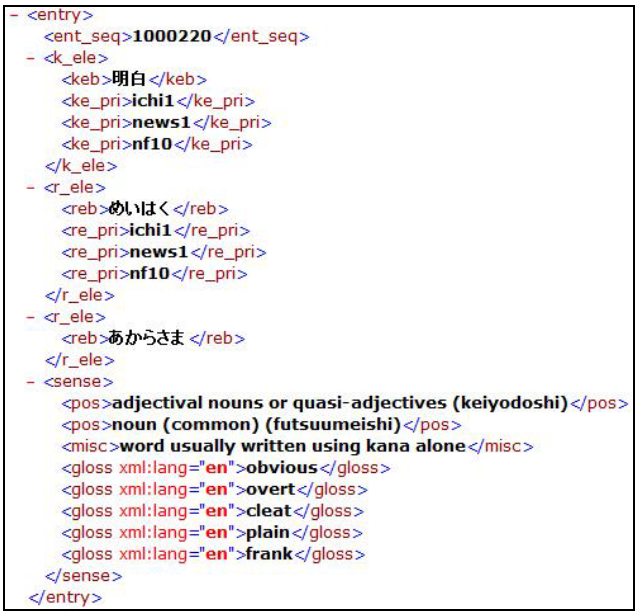
Figure 1.2 An Excerpt from the Kanji Dictionary

```
阿 3024 U963f B170 G9 S8 XN5008 F1126 N4985 V6435 H346 DK256 L1295 K1515 O589 MN41599
MP11.0798 IN2258 P1-3-5 I2d5.6 Q7122.0 Yal Ye1 Ya5 Ya2 Ya4 Wa Wog ア オ おもね.る T1 く
ま ほとり あず あわ おか きた な {Africa} {flatter} {fawn upon} {corner} {nook}
{recess}
哀 3025 U54c0 B8 C30 G8 S9 F1715 N304 V791 H2068 DK1310 L401 K1670 DO1249 MN3580
MP2.0997 E998 IN1675 DJ1804 DG327 P2-2-7 I2j7.4 QD073.2 DR465 Yail Wae アイ あわ.れ あ
わ.れむ かな.しい {pathetic} {grief} {sorrow} {pathos} {pity} {sympathize}
愛 3026 U611b B87 C61 G4 S13 F640 N2829 V1927 H2492 DK1606 L737 K436 O2018 DO456
MN10947 MP4.1123 E417 IN259 DS339 DF545 DH441 DT602 DC268 DJ1079 DG790 P2-4-9 I4i10.1
Q2024.7 DR2067 ZPP2-1-12 ZPP2-6-7 Yai4 Wae アイ いと.しい T1 あ あし え かな なる まな
めぐ めぐみ よし {love} {affection} {favourite}
```

As shown in Figure 1.2, the list of reference codes is followed by several pronunciations in Katakana and Hiragana (the Japanese alphabets, in addition to the Kanji alphabet), and finally by the English translations. Over 12,000 individual Kanji characters and their definitions are contained within the Kanji dictionary corpus.

A final parallel corpus that was used to develop the kanji correction portion of the thesis was Jim Breen's JM Dictionary file. Jim Breen and his students at Monash University created a compilation of Kanji compounds (two or more Kanji placed together to form a new word) in .xml format. Breen's corpus contains not only Kanji pairs and English translations, but also reference values and Part-of-Speech tags to aid in the SMT process. An excerpt from Jim Breen's JM Dictionary is show below:

Figure 1.3 An Excerpt from Jim Breen's JM Dictionary Corpus



As shown in Figure 1.3, a single Japanese Kanji compound can have multiple English translations. This fact contributes to many of the existing SMT mistakes when translating from one language to another. I revisit this in Chapter 3 as the thesis methodology is further explained. Over 120,000 Kanji compound definitions have been defined in Jim Breen's JM Dictionary corpus.

The availability and organization of parallel corpora has increased dramatically over the past decade, and this has contributed to the overall increase in quality in end-to-end MT systems. Having pre-aligned corpora on a sentence-to-sentence basis that are correct translations **and** that provide multiple examples of word usage, phrase order, parts of speech, and context improves any SMT system able to process the information in the first place. The above three referenced

corpora (Tanaka, Kanji Dictionary, and Jim Breen's JM Dictionary corpora (Breen, 2008)) are used as the parallel text building blocks of this thesis' baseline SMT system.

1.1.2 Text Alignment Process

As mentioned in Section *1.1.1*; the text alignment process requires an input of at least one aligned corpus containing corresponding sentences in both the source and target language for the SMT system. Once a parallel corpus is obtained, text alignment tables must be built in order to determine how words in the source and target sentence map to one another.

The main objective of constructing a text alignment table is to represent an accurate mapping from a word or string in a source language to a corresponding word or string in a target language. The actual decomposition of the text alignment tables differs, however, depending on the alignment algorithm used. Franz Josef Och and Hermann Ney provide the implementation of six different types of algorithms in their Giza++ toolkit that are further explained in (Och & Ney, 2000). One common point among all six text alignment model algorithms is a statistical probability map in the following form shown in Figure 1.4,

Figure 1.4 A Common ArgMax Formula for Alignment Tables (Och & Ney, 2000)

$$\hat{a}_1^J = \arg\max_{a_1^J} Pr(f_1^J, a_1^J | e_1^I)$$

where $\hat{a}_1^J$ is our resulting text alignment mapping, f_1^J is a string contained in the source language, a_1^J is a "hidden" alignment describing a mapping from source to target word, and finally e_1^I which is our target language string (Och & Ney, 2000).

The second commonality between all six text alignment algorithms is the training of all alignment models based on the Expectation-Maximization (**EM**) algorithm using parallel corpora (Och & Ney, 2000). As shown below in Figures 1.5 and 1.6 (which are correspondingly the E-

step and M-step), each sentence pair is processed with each word in the source language being assigned a target language word with a certain probability.

Figure 1.5 Lexicon Parameters for E-step of EM Text Alignment Algorithm (Och & Ney, 2000)

$$c(f|e;\mathbf{f},\mathbf{e}) = \sum_{\mathbf{a}} Pr(\mathbf{a}|\mathbf{f},\mathbf{e}) \sum_{i,j} \delta(f,f_j)\delta(e,e_{a_j})$$

Figure 1.6 Lexicon Parameters for the M-step of EM Text Alignment Algorithm (Och & Ney, 2000)

$$p(f|e) \propto \sum_{s} c(f|e;\mathbf{f}^{(s)},\mathbf{e}^{(s)})$$

The summation of these probabilities is then compared against all other sentence alignment outcomes - the result of which is the alignment file that maps words from source-to-target language with probability weights assigned to them. The accuracy of these pairings can be increased by modifying the base algorithm's alignment model (zero- or first-order), adding a fertility model, and accounting for table deficiencies (Och & Ney, 2000). More information on the EM algorithm can be found in (Dellaert, 2002).

By iterating through the hundreds of thousands of parallel sentences; the Giza++ text aligner creates a one-way mapping that relates a single word in the source language to another single word in the target language. In the first and most basic of Giza++'s text alignment algorithms, the probability of each word mapping is given equal weight. In the more complex text alignment algorithms, the designers of Giza++ bring in more advanced concepts such as zero-order, first-order, (inverted) first-order, and fertility model algorithms to output higher-accuracy text alignment files. For the purposes of this thesis, Och and Ney's third iteration of text alignment improvement utilizing a Hidden Markov Model (**HMM**) was deemed sufficient for building a baseline translation system. The HMM will be described in detail in Chapter 3.

The alignment table created with the Giza++ application results in one of two key inputs for the Pharaoh Decoder system. Pharaoh is explained in detail below in Section 1.3, but the second of the two key inputs, Language Modeling, is explained first.

1.2 Language Modeling

In addition to an alignment file produced by a text alignment application, the Pharaoh Decoder system also requires a language model in order to more-accurately translate a source sentence. A language model is built from the exact same inputs used by text alignment applications such as Giza++ explained in Section 1.1. While a text alignment tool is used to map a source language word to a target language word; a language modeling kit is used to create a map for estimating sentence structure based on the prior probabilities of word strings (Stolcke, 2002). Many language modeling kits, such as the CMU-Cambridge Statistical Language Modeling Toolkit or the SRILM Language Modeling Toolkit (LMT), are publicly available. The SRILM toolkit was chosen due to its accurate and customizable interfaces capable of tailoring phrase processing to the needs of this thesis. In the following subsections, the fundamental mathematics behind building an accurate Language Model are discussed along with two different methods used to improve model accuracy.

1.2.2 Probability Tables

The key output produced by a Language Modeling Toolkit is a probability map that can be used to match strings of n-words long (called "N-grams") to a target language text. The construction of these maps begins by applying a variant of Bayes' Rule to available corpora (also known as the product rule of probabilities) defined below in Figure 1.7:

Figure 1.7 Bayes' Rule

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

where 'A' and 'B' are observable, discrete events in any given system. In this case, 'A' and 'B' are replaced by both single words and N-grams where 'N' is the maximum word length of any sentence within the source and target corpora. As will be shown in later chapters – a measurable drop occurs in probability as longer word combinations are compared when modeling Japanese to English N-grams.

As illustrated in another presentation on French-to-English corpora analyzed in (Goodman, 2008); the probability of the following formula:

P ("And nothing but the truth")

is very difficult to compute realistically on a normal corpus and nearly useless to a machine translation system. But, by decomposing the sentence fragment into the following form,

P ("And nothing but the truth") =

P("And") ×P("nothing|and") × P("but|and nothing") × P("the|and nothing but") × P("truth|and nothing but the")

we can compute the entire probability of any given source-to-target language sentence much easier. This writer has found that a 3-gram approximation was the best-fit choice for building the Tanaka Corpus language models when weighing computational time to probability map accuracy; a choice that will be justified in later chapters.

1.2.2 Language Model Improvement – Smoothing

Several techniques exist for improving the quality of the probability table matching N-grams in the source language to the target language. Most of these methods will be described in Chapter 2, but one improvement to the base algorithm listed above that was necessary for this thesis to complete successfully was a technique called "Smoothing."

Smoothing a language model becomes necessary when a common problem called "over-fitting" occurs when analyzing a very small subset (200K sentence pairs) of a near-infinite problem set (all possible language translations from Japanese to English). When analyzing a certain word or kanji with multiple definitions and seeing 1,000 cases of one possible translation vs. one case of a rare translation, most language modelers such as SRILM or IRSTLM will assign a near-zero probability (<.000001) to the rare translation. Once an alignment table and a language model combine and begin to assess the correct translation for that certain hypothetical

kanji, it is near impossible to ever see the rare translations again no matter the context of the rest of the sentence. To fix this problem, the smoothing technique essentially "steals" probability from the higher-probability translations and redistributes it evenly between all extremely-low probability translations. This thesis used a simple smoothing technique (linear distribution), but more advanced smoothing techniques for natural language processing are explored and presented in (Ney & Essen, 1991).

Now that both text alignment and language model table creation have been explained, the methods behind combining these two artifacts and inputting them into the beam search decoder translation system called Pharaoh will be explored.

1.3 Beam-Search Decoder: Pharaoh

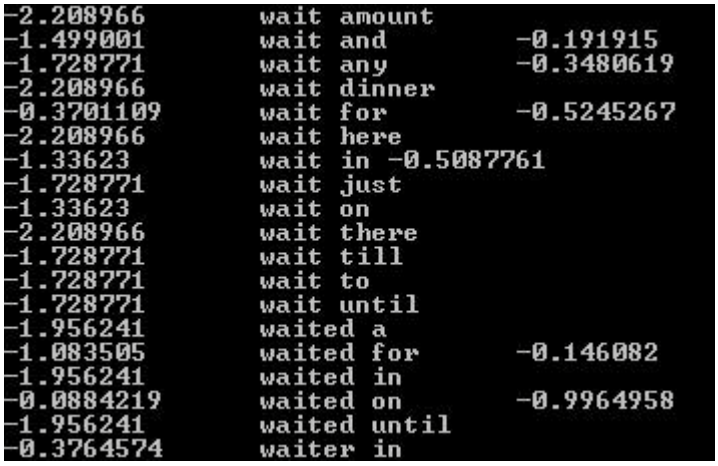
Pharaoh is a beam search decoder system that requires a full translation model consisting of both a translation table (as produced by Giza++ explained in Section 1.1) and a language model (produced by SRILM explained in Section 1.2) (Koehn, 2004). Pharaoh first uses a language model to determine probability matches of certain phrases in the target language. This includes single-word matches (or 1-grams), so one translation model input to the Pharaoh application takes on the form shown in the following figure:

Figure 1.8 Sample Language Model Generated by SRILM

```
-2.208966       wait amount
-1.499001       wait and        -0.191915
-1.728771       wait any        -0.3480619
-2.208966       wait dinner
-0.3701109      wait for        -0.5245267
-2.208966       wait here
-1.33623        wait in -0.5087761
-1.728771       wait just
-1.33623        wait on
-2.208966       wait there
-1.728771       wait till
-1.728771       wait to
-1.728771       wait until
-1.956241       waited a
-1.083505       waited for      -0.146082
-1.956241       waited in
-0.0884219      waited on       -0.9964958
-1.956241       waited until
-0.3764574      waiter in
```

As shown above in Figure 1.8, weights for each of the 2-grams (Bi-grams) are listed, followed by the actual text of the Bi-gram (in this case, the probability of the second word given the first) and followed finally by the total, normalized probability total for the Bi-gram (Stolcke, 2002). The reader will notice that some Bi-grams do not actually have a chance of being a correct translation, and this is due to an adjustable cutoff value weeding out unlikely translations based on the input corpora. Even though these Bi-grams do not contain a probability for actual translations – the Bi-grams are still recorded as part of the experimental set if the decoder (Pharaoh) cannot find any other likely translations. Also, the reader should note (for future understanding of the thesis methodology) that larger N-grams depend upon previous N-gram probabilities to speed up the calculation of the probability of one word sequence given a previous sequence. For instance – in Figure 8 the probability of the Bi-gram 'wait and' is -.191915; a fairly low score. And, as shown below in Figure 1.9:

Figure 1.9 Second Sample Language Model Generated by SRILM



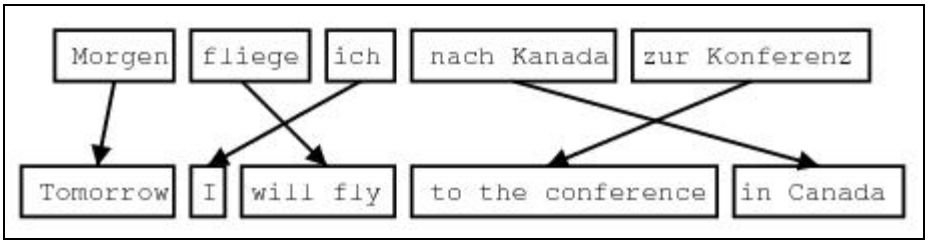
The Tri-gram 'wait and see' has no chance of being selected if another Tri-gram with any probability exists that would fit a given scenario. This is due in part to the low score of the Bi-gram 'wait and' as shown in Figure 1.8.

In addition to the language model produced by running SRILM (or other similar language modeling toolkit such as CMU's Toolkit), Pharaoh also requires a phrase table to be generated. This phrase table is a heavily altered form of the Giza++ output to be discussed in Chapter 2: Background and Related Work. But, to briefly describe the necessary process, Giza++ produces

a ***.A3.final file which defines how each word in each sentence is matched up in the parallel corpora. Each full text alignment training cycle (where '***' represents the direction of text alignment; for instance: English_Japanese.A3.final) produces such a file. This process must be run both in the source-to-target and target-to-source directions to create a sum and intersection of likely N-gram matches. Once a ***.A3.final file is created for each direction of translation, more alterations using both files as inputs and the Thot toolkit take place to create the eventual phrase table that Pharaoh depends on. Please reference Section 3.1 and Section 3.2 for algorithm details for each of the above processes.

With both a language model generated from the original corpus in the target language and a phrase table constructed with a variety of applications beginning with personally developed Java applications and ending with the Thot toolkit (Ortiz-Martinez, Garcia-Varea, & Casacuberta, 2005), Pharaoh is able to begin decoding and reordering text strings in a source language to the target language (German to English in the below example) as shown in the figure below:

Figure 1.10 Decoding and Reordering Phrases in Pharaoh (Koehn, 2008)



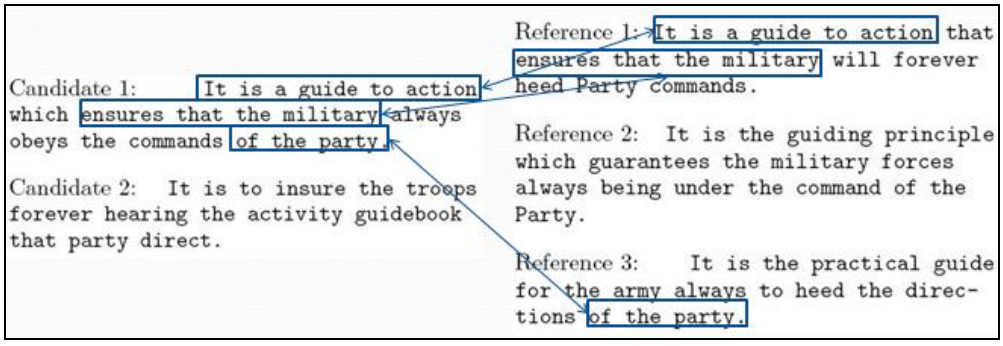
As the reader can see in Figure 1.10; Pharaoh first locates the largest N-gram possible to translate (Bi-grams being the largest N-gram in the example), and proceeds to reorder the phrases as dictated by the supplied language model. Words are dropped from the Pharaoh translation if a match cannot be found in the phrase table, and phrase order is left as provided in the original sentence if a phrase-reordering cannot be found in the supplied language model (Koehn, 2008). The Pharaoh Beam-Search Algorithm is revisited in Chapter 2 as areas for improvement and adjacent research topics are analyzed.

Once all inputs are provided to the Pharaoh application; a user may begin to gauge the effectiveness of their overall system. Many measurements are available to qualitatively and quantitatively judge how a SMT translation compares to a human translation (and therefore set at "100%" accurate), but the BLEU metric algorithm is perhaps the most widely-accepted and automatic algorithmic method of determining translation quality in the SMT community. The facts behind why this is the case and the methodology of the BLEU metric are explained in the following sub-section.

1.4 BLEU Metric

BLEU is an abbreviation for "**B**i**L**ingual **E**valuation **U**nderstudy" and is one available metric to evaluate the performance of SMT systems (Papineni, Roukos, Ward, & Zhu, 2001). The BLEU metric was introduced to the Machine Translation community due to a need for a tool to automatically evaluate the hundreds of thousands of sentences being produced by Machine Translators near the end of the 20th century. The BLEU metric allows a user to compare vast numbers of translated sentences against multiple references of a single sentence in a source language in order to compute maximal BLEU scores. An example of this concept is shown in the figure below:

Figure 1.11 Phrase Matching Between Multiple Candidate and Reference Sentences; Figure Edited from (Papineni, Roukos, Ward, & Zhu, 2001)



As shown above in Figure 1.11; Candidate 1 contains several phrases that occur in multiple reference sentences. If only a single reference sentence was provided and the candidate

sentence did not contain the exact phrases within it – the BLEU score will be artificially low as the algorithm does not provide leniency for synonyms or near-matches. Thus, when utilizing the BLEU algorithm as an evaluation metric for a MT system, it is imperative to have as many reference sentences as possible. Further research on the relationship between number of reference sentences, quality of machine versus human translations, and raw BLEU score is presented in Chapter 2.

BLEU metric scores range from a value of zero to one and analyze a variety of translation aspects to determine a score for a candidate sentence when compared to its reference sentence counterparts. The BLEU metric requires two inputs - a brevity score penalty and a geometric mean of the test corpora modified precision scores (Papineni, Roukos, Ward, & Zhu, 2001). The brevity penalty occurs when a candidate sentence is shorter than the maximum length of the references sentences used as an input as demonstrated by the equation below in Figure 1.12,

Figure 1.12 BLEU Metric Brevity Penalty (Papineni, Roukos, Ward, & Zhu, 2001)

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

where c is the length of the candidate sentence and r is the effective corpus reference length (Papineni, Roukos, Ward, & Zhu, 2001). The geometric mean of the test corpus modified precision scores is obtained via the following formula:

Figure 1.13 - BLEU Metric Modified Precision Score Algorithm (Papineni, Roukos, Ward, & Zhu, 2001)

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count(n\text{-}gram)}$$

After computing the brevity penalty and precision score for a candidate translated sentence, the raw BLEU score is calculated via either of the following two algorithms in Figure 14 and Figure 15 below.

Figure 1.14 - BLEU Metric Geometric Averages Using Modified N-Gram Precisions (Papineni, Roukos, Ward, & Zhu, 2001)

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

With some minor modifications, the algorithm in Figure 13 can be altered to be a logarithmic score with a range of zero to one as shown in Figure 14 below,

Figure 1.15 - BLEU Metric Geometric Averages Using Modified N-Gram Precisions in the Log Domain (Papineni, Roukos, Ward, & Zhu, 2001)

$$\log \text{BLEU} = \min(1 - \frac{r}{c}, 0) + \sum_{n=1}^{N} w_n \log p_n$$

where N is the length of the longest N-gram being analyzed, w_n is a positive weight summing to 1 (1/N in this case), and p_n being the geometric average of the modified N-gram precisions (Papineni, Roukos, Ward, & Zhu, 2001). The result of this algorithm on any given set of candidate and reference sentences is a value from zero to one for all N-gram lengths of phrases as defined by the user. For reference; even human translators do not usually score a '1' as measured using the BLEU metric. The only way a '1' can be achieved is if every single candidate sentence matched at least one reference sentence word-for-word (in which case the test cases must be analyzed for validity). Most machine translation systems today fall within the .05 - .25 BLEU range, with Google Translate being the commercial-best at an average of roughly .30 depending on the test corpus and language pair in question. Most bilingual human translations fall within the .30 - .50 range as many human translators also have multiple sentence structures to express similar ideas (Papineni, Roukos, Ward, & Zhu, 2001).

The BLEU metric, although widely accepted as an impartial judge of sentence quality, is not without its pitfalls and shortcomings. The BLEU algorithms are vulnerable to generating artificially low scores due to synonym word use, sentence length, and will not accept different phrase orderings to express the same idea. While the NIST algorithm corrects some of these

deficiencies and falls prey to others; it was deemed necessary to choose only one method to measure the quality of this thesis' MT system, and therefore the BLEU metric was chosen.

1.5 Introduction Summary and Thesis Claims

The concepts of Parallel Corpora, Text Alignment, Language Models, Beam-Search Decoding, and the BLEU metric form the building blocks of this thesis' overarching goal; to improve the quality of Japanese-to-English text translation. While any single one of these foundations of MT could have been targeted for improvement; this thesis proved the following claims:

1) By analyzing the context of kanji compounds in the Japanese language that have greater-than or equal-to three English translations and correcting these errors in a translated sentence; a measurable increase in BLEU score and human understandability was achieved.

2) By segmenting Japanese text more correctly in the pre-text alignment phase of MT; a measurable increase in BLEU score and human understandability was achieved.

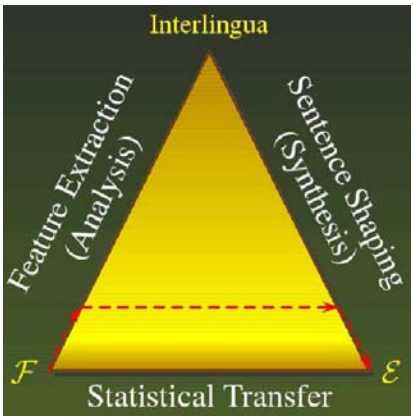
As the writer of this thesis has discovered; these improvements are a very small subsection of possible improvements to MT systems in general. Being familiar with Japanese and fluent in English, however, has allowed this experiment to improve this small scope of MT for this pair of languages. And, by combining these unique results with other Natural Language Processing research in the fields, the Statistical Machine Translation knowledge base can continue to grow.

The next chapter focuses on adjacent areas of research where parallel progress can and has been made, and also introduces several other interesting discoveries from other researchers that have helped this thesis experiment be built upon a rational basis.

CHAPTER 2 - Background and Related Work

In Chapter 1, the topics of Text Alignment, Language Models, Decoders, the Measuring Statistical Machine Translation Quality, and Thesis Claims were explained. This chapter builds upon those areas of research and explains prevalent strengths and weaknesses. The weaknesses were then analyzed to determine which categories were the most value-added areas for unique contributions. As noted in Section 1.5 above; the concept of higher-quality segmenting and correcting definitions of incorrectly-translated kanji compounds were deemed the two areas most likely to result in highest overall translation improvement. Section 2.1 begins by analyzing promising methods for improving the word-to-word levels of translation, which can be thought of as the base of the MT pyramid as shown in Figure 2.1 (courtesy of Papineni, et. al.):

Figure 2.1 **The Machine Translation Pyramid (courtesy of Papineni, et. al.)**



The red dotted-line shown in Figure 16 is very similar to the path through MT that this thesis took. That is to say; this thesis focused on improving the base levels of MT such as text segmentation and single-word corrections for contextual definitions combined with basic language modeling and decoding techniques rather than improving higher-level MT concepts such as language modeling and reordering of phrases. The reader may think of the lower-levels of the MT pyramid above as gravitating towards total unthinking, dictionary-based syntax matching, and of the higher-levels as gravitating towards fine-tuning sentence structure, understanding parts of speech, grammar, and hidden meanings.

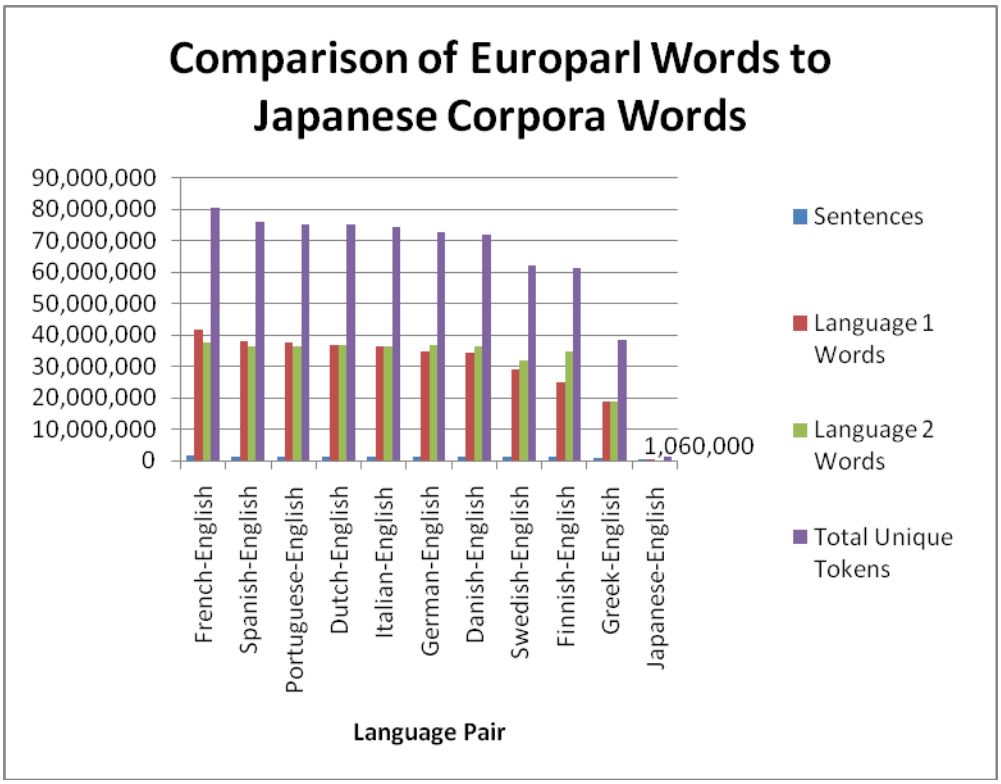
The chapter will now continue explaining adjacent areas of research that could be targeted for future improvement, and conclude by detailing the research that led up to the development of this thesis' claims.

2.1 Text Alignment Methodologies and Areas for Improvement

2.1.1 Parallel Corpora Background and Potential Improvements

One of the first steps in building a MT system is the identification of test corpora to use in the initial text alignment process. One problem with developing a Japanese-to-English text translation system is the comparatively small number of corpora readily available. The NHK, Nikkei, Tanaka, and JMDict corpora are four of the largest, parallel-sentence files for Japanese-to-English comparisons, and their size is dwarfed by other language pairs when compared side-by-side (Nightingale & Tanaka, 2003). The following graph illustrates this point by comparing Japanese-to-English corpora against other language pairs' resources:

Figure 2.2 **Numerical Comparison of Europarl Corpus Language Totals against 4 Japanese-English Corpora Totals**



As shown in Figure 2.2 (where L1 is the source language, and L2 is the target language) - the sheer amount of data present in the Europarl corpus alone for the smallest subset of language pairs (Greek-English) is nearly 40x the amount of data present in the four combined Japanese-English corpora listed at the beginning of this sub-section (Koehn, 2005). The total number of Japanese-to-English sentences and tokens is near 1,060,000, and the total number of Greek-to-English sentences 38,324,089. An exact measure of the Japanese-to-English sentences is difficult to estimate due to the duplication of some sentences in the Tanaka Corpus, but the total number of unique sentences and tokens could still not be above the number listed above which is the important item of note. The Europarl corpus is extracted from the European Parliament proceedings and translated to 11 European languages on an ongoing basis, and is a common resource for SMT developers when translating those specific languages. Nightingale and Tanaka have presented methods to increase the quality of alignment through literal word translations in (Nightingale & Tanaka, 2003), and this is a promising area of research due to the need to make

existing Japanese-English parallel corpora as efficient as possible. Due to the negative impact to the quality of overall SMT systems that is caused by the relatively small number of Japanese-English corpora; a future area for improvement will be increasing both the quantity and quality of available translations for SMT processing. Unique methods to improve the quality of available Japanese-English corpora are presented in Chapter 3.

Another area of potential improvement is the potential creation of a system able to analyze a translated sentence and identify incorrect translations based on contextual situations. Eric Brill proposed a similar method of learning regular expressions to lower target-language word ambiguity based off of surrounding string context in (Brill, 2000) which is further explored in Chapter 4. A rudimentary implementation was created in this thesis that built off of the same hypothesis with vastly different implementation schemes, but the end goal is the same: to improve the quality of a translation through disambiguation of translated words based on sentence context.

2.1.2 Segmenting Japanese Text in Parallel Corpora

As written in the previous section – the quantity of available Japanese-English corpora (and other East Asian languages such as Chinese or Korean) is very low when compared to other language pairs in general. To compensate for this deficiency, the available corpora must be made as accurate and as representative of the language pairing as possible. A key difference between Japanese and English is the distinct lack of spacing between any character, kanji, part-of-speech tag, or punctuation mark in the Japanese language. The lack of whitespace in Japanese makes using standard MT tools incredibly difficult (requiring several other modifications such as character conversion and C++ database interface updates that are described in Chapter 3), and therefore some preliminary edits to the initial parallel corpora were needed. An example Japanese sentence before and after parsing through one of the thesis intermediary applications is shown in the below figure:

Figure 2.3 **Text Alignment Pre-Processing Specific to Japanese**

```
4

今日は月 18 日、ムーリエルの誕生日です！

Today is June 18th and it is Muiriel's birthday!"

\u4eca\u65e5 \u306f \u670818\u65e5\u3001 \u30e0\u30fc\u30ea\u30a8\u30eb
\u306e \u8a95\u751f\u65e5 \u3067\u3059 !
```

One important item of note in Figure 2.3 is the conversion of the raw Japanese character set (the second line of text) to Unicode values (the fourth line of text) that are understood in string-form by all SMT alignment tools used to complete this thesis experiment. Another important fact is that whitespace has been added in to the Unicode values, creating a "tokenized" form of the Japanese language that has improved overall translation quality. The reasons and methodology behind these two additions in functionality are fully explained in Chapter 3 and Chapter 4.

While many other researchers have described tactics to segment text correctly, very few have released any usable toolkits into the SMT open source domain. Only one other commercial or open-source tool, ChaSen, was found that is supposedly capable of segmenting Japanese text correctly. ChaSen suffers from a multitude of compilation errors and, as the instructions are completely in Japanese and the software was abandoned in March of 2007, the writer of this thesis deemed it prudent to develop an in-house solution to the segmentation problem that produces results as shown in Figure 2.3. A link to the ChaSen homepage is available here:

http://chasen.naist.jp/hiki/ChaSen/

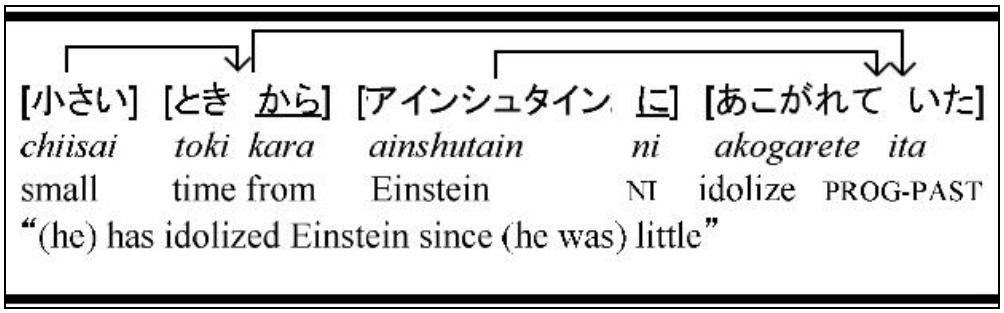
Improving the segmentation quality of Japanese textual input vastly improved overall translation quality and is explained in subsequent chapters.

2.1.3 Predicting Japanese Case and Grammatical Markers

Quantity and accuracy of Japanese-English parallel corpora have a large correlation with the quality of an end-to-end MT system, as does being able to tokenize Japanese into logical character sets for the text alignment tools to use. In addition to these translation improvements is the concept of case and grammatical markers being able to further refine sentence tokenization as presented in (Suzuki & Toutanova, 2006).

An example of case markers is shown in the below figure as referenced from Suzuki and Toutanova's paper:

Figure 2.4 **Example of Japanese Case Markers (Suzuki & Toutanova, 2006)**

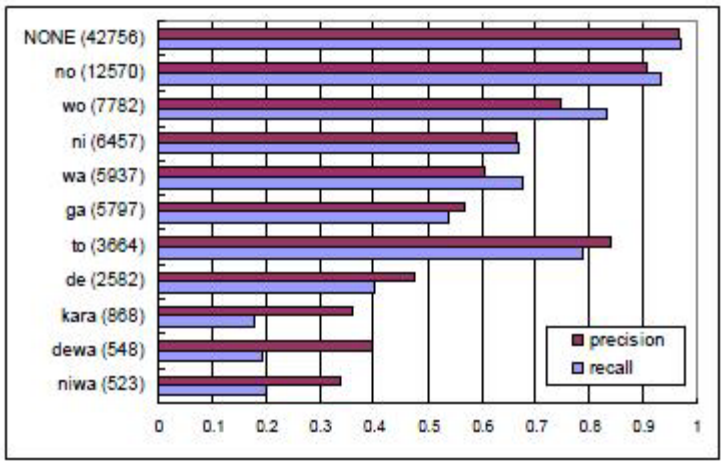


The underlined characters in the Japanese text in Figure 19 are examples of case markers; in this case 'kara' and 'ni'. Dependencies between bracketed phrases are shown with drawn arrows.

While this thesis experiment's scope was defined short of fully implementing Japanese grammatical and case marker recognition, the research does deserve mention due to inspiring some of the programming behind the tokenizing of Japanese text. The reader should note the difference between precision and recall defined in the table: Precision refers to the correctness of a translated word, and Recall refers to the ability of the MT system to find a translation for a word or phrase at all. Scores range from zero to one when measuring precision and recall for a MT system.

Results of the (Suzuki & Toutanova, 2006) research are promising as shown by the table below:

Figure 2.5 **Precision and Recall Measures from Kyoto Corpus (Suzuki & Toutanova, 2006)**



As the reader can see (and understand, if they happen to be familiar with Japanese), the Japanese markers ‘no’, ‘wo’, and ‘to’ are the markers most often identified correctly. Other Japanese markers trail behind since they are more difficult to identify when compared to the rest of the source sentence.

The reader should now have a good idea as to the areas of research describing the MT process on the lowest levels of the MT pyramid (Figure 2.1). The chapter will now continue further up the pyramid, exploring higher-level concepts such as language modeling techniques and decoding algorithms.

2.2 Building and Improving Language Models

Constructing a high-quality language model is an imperative part of current-day MT systems. Unlike the input for the text alignment process, however, the input to the language model applications can be in the target language only (English, in this case). The models learned from building 1-through-N-gram sentence structures are used in conjunction with the translation tables to determine what order is most natural after translation for source words is complete. The following sub-sections explain the three different types of language modeling software used by this thesis experiment, and how the methodology behind their processes.

2.2.1 IRSTLM Language Modeling Kit

The first of the three language modeling toolkits is specifically tailored for use with the Moses application (an alternative to Pharaoh that, after an in-depth trade study, was decided against). IRSTLM does, however, offer a smoothing methodology that was found useful in tailoring other developed language models to this thesis' needs. IRSTLM performs four actions to create a complete language model; 1) Collecting N-grams and caching the collections to memory, 2) Performing Frequency Smoothing of N-grams based on their predecessor (N-1)-grams, 3) Compilation to binary form, and finally 4) Compression to a bin format (Federico, Bertoldi, & Cettolo, 2008). While each of these steps in the IRSTLM process is worth studying, only parts of the process behind Smoothing (#2) were carried forward into the final thesis artifacts. IRSTLM is capable of producing a language model compatible with the Pharaoh decoder, but IRSTLM was meant for very large corpora numbering in the millions of corpora sentences and gigabytes of memory needs. IRSTLM's main discriminator from other language modeling toolkits is the capability to separate corpus analysis into parallel thread processes and merge the results back together. The runtime differences between IRSTLM and SRILM are illustrated in Figure 2.6.

Figure 2.6 **Run Time Comparisons between IRSTLM and SRILM Language Modeling Toolkits (Federico, Bertoldi, & Cettolo, 2008)**

LM(s)	kit	file size (Gb)	proc. size (Gb)	RTR	WER %
wb-3gr-news	sri	7.6	6.7	.65	14.91
wb-3gr-news	irst	2.0	2.4	.76	14.96
wb-3gr-q-news	irst	1.2	1.6	.75	14.94
kn-3gr-news	sri	7.6	6.7	.68	14.76
kn-3gr-news	irst	2.0	2.4	.79	14.73
kn-3gr-q-news	irst	1.2	1.6	.79	14.77
skn-3gr-news	sri	7.6	6.7	.65	14.89
skn-3gr-news	irst	2.0	2.4	.76	14.95
skn-3gr-q-news	irst	1.2	1.6	.76	15.00
wb-4gr-q-(nws+itwc)	irst	3.6+7.3	11.3	.82	12.78

Figure 2.6 shows the Language Models (LM), type of modeling toolkit used (SRILM vs. IRSTLM), real-time-ratios (RTR) for the language model processing compared to actual speech duration (this IRSTLM data was taken from language model built off of speech recordings), and finally word-error-rate (WER) percentage. As the reader can see, IRSTLM outperforms SRILM in nearly every category except for minor increases in WER% for two of the corpora. IRSTLM compresses the language model data by a factor of three to four, and processes the information roughly 15% faster than SRILM in most cases. The preliminary runs of IRSTLM on the Tanaka Corpus looked promising, but also resulted in several compilation challenges that did not lie within the scope of this thesis to solve. IRSTLM was truly meant to process huge corpora efficiently; but when processing small corpora (< one million sentences), the benefits of using IRSTLM vs. SRILM dropped dramatically in terms of language model build time. Due to the near-equal performance time of both language modeling kits on the Tanaka Corpus and other implementation difficulties with IRSTLM, IRSTLM was not chosen as the language modeling toolkit of choice. More information on Federico, Bertoldi, and Cettolo's IRSTLM Toolkit is available through the (Federico, Bertoldi, & Cettolo, 2008) reference.

Since IRSTLM was not chosen as the final language modeling toolkit, no further analysis of the methods behind it will be written in this thesis. This brings us to the next language modeling toolkit considered for use; the CMU-Cambridge Statistical Language Modeling Toolkit v2.

2.2.2 CMU-Cambridge Statistical Language Modeling Toolkit v2

The CMU v1 Toolkit was originally written by Roni Rosenfeld of Carnegie Mellon University in 1994. After three years in the public domain, the toolkit was updated by both Philip Clarkson and the original author. This collaboration led to the current language modeling toolkit being developed and release to the public under the CMU-Cambridge Statistical Language Modeling Toolkit v2 label.

The CMU-Cambridge toolkit offers a unique advantage over other language modeling kits in the fact that each phase of language modeling is decoupled from the rest of the process, and each of these phases has a high-degree of customizability. The figure below will help to illustrate this fact by showing the usual order of input/output to the toolkit processes, and how these steps combine to form the entire language model construction and verification.

Figure 2.7 **CMU-Cambridge Statistical Language Modeling Toolkit v2 Process Flow (Clarkson & Rosenfeld, 1999)**

LM(s)	kit	file size (Gb)	proc. size (Gb)	RTR	WER %
wb-3gr-news	sri	7.6	6.7	.65	14.91
wb-3gr-news	irst	2.0	2.4	.76	14.96
wb-3gr-q-news	irst	1.2	1.6	.75	14.94
kn-3gr-news	sri	7.6	6.7	.68	14.76
kn-3gr-news	irst	2.0	2.4	.79	14.73
kn-3gr-q-news	irst	1.2	1.6	.79	14.77
skn-3gr-news	sri	7.6	6.7	.65	14.89
skn-3gr-news	irst	2.0	2.4	.76	14.95
skn-3gr-q-news	irst	1.2	1.6	.76	15.00
wb-4gr-q-(nws+itwc)	irst	3.6+7.3	11.3	.82	12.78

Every rectangle in 2.7 represents a process provided by the CMU-Cambridge toolkit, and every oval represents a file that either serves as an input, output, or both to the language modeling system (Clarkson & Rosenfeld, 1999).

It should be noted that the only inputs into building a language model for a SMT system are corpora from the target language (English, in this case). Beginning with the same corpus used as text alignment input, the CMU-Cambridge toolkit must first parse every unique token that occurs within the target sentences. The output from this process is used to create a standard vocabulary file that contains all unique tokens and the count of their occurrence in the target language. This capability was also provided as part of this thesis' deliverables; the thesis artifact of which contained extra formatting functionality to account for Japanese text and the CMU-Cambridge toolkit containing certain token-excluding capabilities to account for non-words (i.e.

apostrophes, periods, quotes, etc. that count as tokens but not part of spoken language). The input corpus and vocabulary file are then used as inputs to the process that produces an integer-map of word positions in target sentences. This file, combined with the original vocabulary file, is then converted to a language model for use by a decoder such as Pharaoh.

While the highly-decoupled process for creating a language model might seem to warrant a bit more streamlining; the CMU-Cambridge toolkit has allowed for improvements at any phase of the language model creation to be created by the Natural Language Processing (**NLP**) community. Various potential avenues for further research are presented in (Clarkson & Rosenfeld, 1999), along with further definition of each phase of the process.

While the CMU-Cambridge toolkit offered an interesting alternative way to build a language model, eventually compliance issues with both front- and back-ends of the application forced the use of SRILM as the language toolkit of choice.

2.2.3 SRILM Language Modeling Toolkit

SRILM was by far the most difficult application to bring to full functionality, but also proved to be the most extensible when compared to the IRSTLM and CMU-Cambridge language modeling toolkits. SRILM was built based on many discussions between its authors and Rosenfeld, the author of CMU v1. Therefore, it not only contains all of the listed functionality within the CMU-Cambridge language modeling toolkit, but extends the functionality in key areas (Stolcke, 2002). For example; the most current version of SRILM offers additional word graph rescoring, test suites, and support for Windows-Cygwin distributions.

The SRILM toolkit contains 14 executable tools with which to create and / or modify language models, all written in C++ (Stolcke, 2002). The most important of these tools is *ngram-count*, as this executable is necessary to create the language model. As all basic language model toolkit functionality is described above in Sections *2.2.1* and *2.2.2*, the additional capability that SRILM contributes will now be explained:

1) Customizable N-gram Orders: SRILM is not constrained by a maximum N-gram length for language model creation. However, as Chapter 5 shows, diminishing returns make N-grams with a length greater than three nearly useless.

2) Choice of Discounting Algorithm: Most language modeling toolkits utilize a Good-Turing or Witten-Bell discounting algorithm by default, but SRILM offers the choice between these, absolute, or modified Kneser-Ney methods.

3) Supports Pre-Defined Vocabulary Lists: Although this feature was not utilized, it could help to improve the quality of the training data output.

4) Unknown Word Filters: Similar to the <s> tags in the CMU-Cambridge toolkit; unknown words can be specially handled with SRILM to improve training data quality.

5) Text Formatting: SRILM is capable of performing many formatting actions to reduce text ambiguity (for example, changing the word “San Francisco” to “san francisco,” and adding it to the word class <city>) (Stolcke, 2002). This helps to reduce unnecessary vocabulary words due to factors such as capitalization, and is similar to Giza++’s mkcls executable which classified words based on alignment potential.

SRILM offers several other capabilities that proved interesting, but the limitation for using most of the features was corpus size. Without large samples of target sentences with good examples of word use (> one million sentences corpus size with a lower percentage distribution of vocabulary than the Tanaka Corpus), SRILM’s capabilities to alter language models, segment N-best lists, create language lattices, and others proved to be ineffective at further improving translation quality.

The basic functionality of SRILM along with its compatibility with both the corpora inputs and needed outputs for Pharaoh was enough to justify its use for this thesis. Chapter 6:

Conclusions and Future Work explores some potential areas where the additional SRILM capabilities that were not fully explored in this thesis could be further studied as a key improvement area for SMT.

With SRILM chosen as the language modeling toolkit; the beam-search decoder applications Moses and Pharaoh are briefly described before moving on to Chapter 3.

2.3 Beam-Search Decoders and Machine Translation Output

Brief mention must be given to two of the most prevalent, openly-available translator applications; Moses and Pharaoh. Both toolkits are phrase-based, beam-search, statistical machine translation decoders. The toolkits are described in Sections *2.3.1* and *2.3.2*, but as little unique improvements were made to either system will not be described in as great of detail as the previous parts of the MT pyramid.

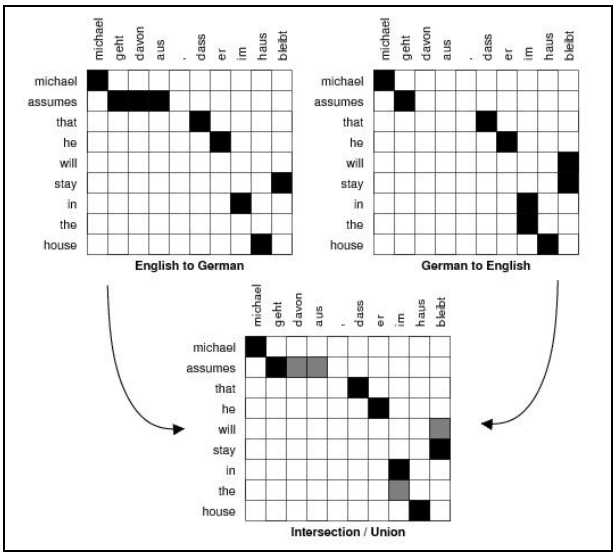
2.3.1 Moses Decoder

The Moses decoder (last updated in February of 2008) was created to be a replacement for Pharaoh (coded in 2004). The Moses decoder offers several advantages over the Pharaoh decoder, but also brought several challenges along with the extra functionality. Moses is packaged as a complete SMT system as it includes interfaces to Giza++, Giza++ training, IRSTLM, SRILM, and several helper applications for processing parallel corpora. The Moses homepage can be found here:

http://www.statmt.org/moses/?n=Development.GetStarted

Moses begins its operation by requiring both an intersection and a sum of text alignments be created through bi-directional runs of Giza++ training. An illustration of this concept is shown in the below figure, courtesy of the Moses staff (Koehn, et al., 2008):

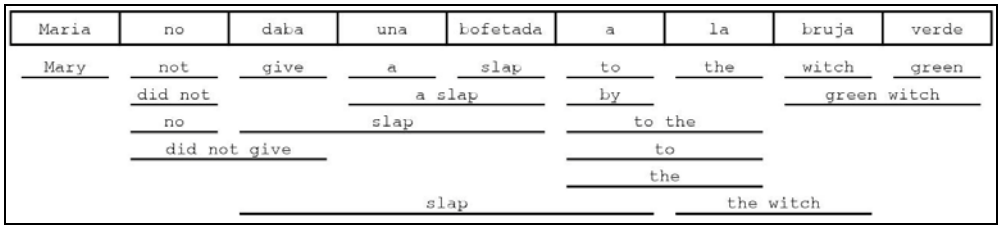
Figure 2.8 **Text Alignment Intersection and Union (German-English) (Koehn, et al., 2008)**



This concept is important to note for the thesis experiment described in Chapter 3 because only a union of text alignments was used to create the phrase translation table needed by Moses or Pharaoh. The union of bi-directional alignment runs results in a translation table with a high recall but low accuracy (aka a "quantity over quality" approach to building a MT system). This approach was necessary due to the relatively small size of Japanese-English parallel corpora and the uniqueness of the vocabulary tokens encountered in the Japanese language corpus.

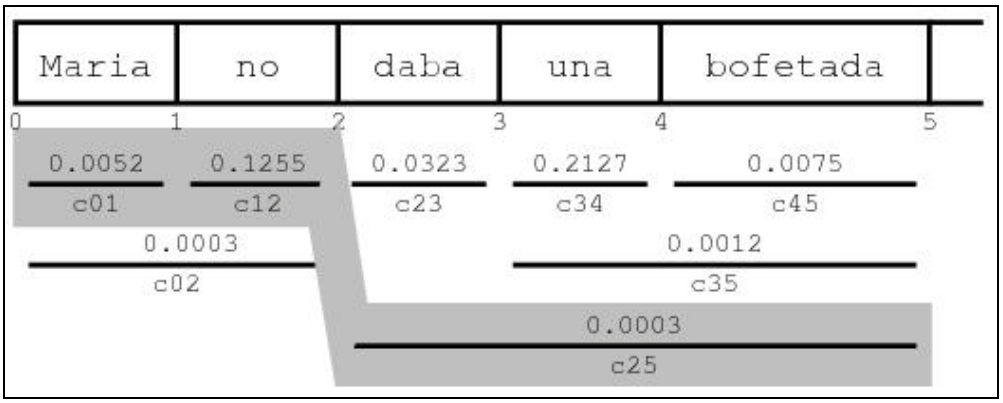
Moses combines the translation / phrase table (as shown above in Figure 2.8) with a number of translation options for each sentence in the source language. A table similar to the one shown below provides a visualization of this activity:

Figure 2.9 **Translation Options for a Spanish-English Sentence Pair (Koehn, et al., 2008)**



Each word in the source sentence is analyzed first as a 1-gram, and the beam-search decoder algorithm within Moses proceeds to analyze N-grams up to the entire sentence length (of which a match is found for only if a source sentence was seen word-for-word several times in the training data). A "cost" of translation is then assigned to each of the possibilities, with the lowest-cost path resulting in the final translation (Koehn, et al., 2008). This final phase of the Moses decoding process is illustrated in Figure 2.10 below:

Figure 2.10 **Lowest-Cost Translation of a Spanish-to-English Sentence (Koehn, et al., 2008)**



The Moses and Pharaoh decoders both compute the probabilities of every translation of all found N-grams in a source sentence to compute several hypotheses of potential target language translations (Koehn, 2008). The hypothesis with the lowest cost (and therefore highest probability) of accuracy is chosen, unless certain advanced features of Moses are activated prior to translation. While these advanced features are outside of the scope of this thesis, topics such as lexical reordering, binary phrase tables, additional language model improvements, n-best list

generation, and additional word-to-word alignment capabilities did offer inspiration for future work.

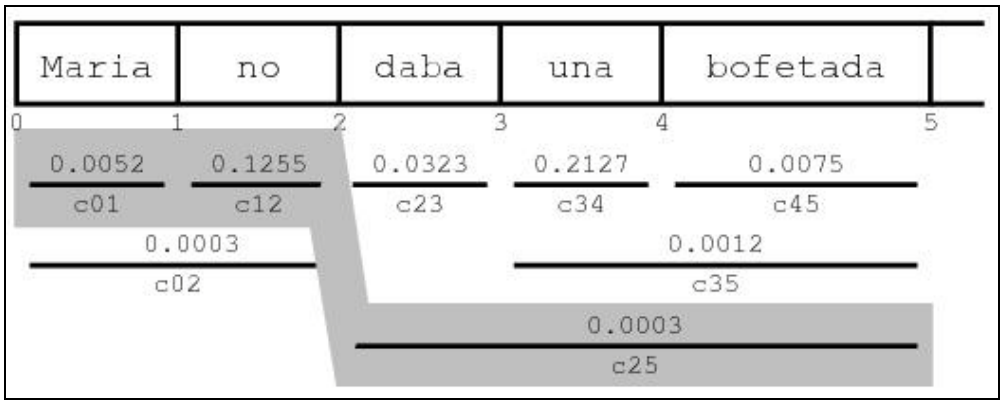
Now that the basic functionality of the Moses decoder has been explained, the chapter concludes by analyzing an application that served as a subset of Moses; namely, the Pharaoh beam-search decoder.

2.3.2 Pharaoh Decoder

As the Pharaoh decoder was written in 2004 (and last updated functionally in 2006) by one of the authors of Moses, the Pharaoh capability is mostly subsumed by the information contained within Section *2.3.1*. The Pharaoh decoder, however, is much more decoupled from the rest of the MT process allowing for many types of text alignment and language modeling applications to be used instead of Giza++ or SRILM. While Giza++ and SRILM were chosen as the text alignment and language modeling tools for this thesis anyway – as Chapter 3 will show it was important to allow for additional input/output parsing to occur between processes that would have been more difficult if the highly-coupled Moses SMT toolkit had been used.

The one capability of Pharaoh that was tailored to this thesis was the ability to set a "no limit" option on sentence reordering. This capability proved to be of interest due to the large shift in alignment when translating from Japanese to English sentences. A small example of this technique is shown in the below figure:

Figure 2.11 **Example of Phrase Reordering in Pharaoh (Koehn, 2008)**



In Figure 2.11 - the concept of reordering phrases to a target language is shown by switching phrases two and three in the source language. This switch is made by Pharaoh depending on the weight the language model of the target language is given when compared to the N-gram word translations in the phrase table (Koehn, 2008).

This concludes Chapter 2. While this is still only a small portion of the available research concerning Statistical Machine Translation, the basis that this research formed was sound enough to further refine the claims mentioned in the conclusion of Chapter 1, and transform those claims into verifiable thesis statements:

1) Japanese to English Machine Translation quality, as measured using the BLEU metric, is improved by locating and correcting kanji in a source sentence that Pharaoh was unable to find a match for.

2) Segmenting text as part of the Giza++ text-alignment preprocessing results in a baseline increase in translator effectiveness as measured using the BLEU metric.

The methodology behind proving the above thesis statements is described in the next chapter, Chapter 3.

CHAPTER 3 - Methodology

Chapter 1 and Chapter 2 together have detailed both the buildings blocks of building an end-to-end MT system, as well as explained the current state of many different areas of parallel research occurring in the SMT community. This chapter will now explain the implementation behind building all of the above systems, the modifications necessary to get the decoupled processes to work together, the intermediary processing necessary to turn one applications output into another's input, and concludes by explaining the unique kanji correction utility.

3.1 Corpora and Parsing Techniques

The Tanaka Corpus, consisting of roughly 212,000 original sentence pairs (Breen, 2008), was selected to train the initial text alignment machine learning. The Kanji Dictionary was also evaluated for final use, but due to a lack of standardization with its formatting and being largely subsumed by Jim Breene's JM Dictionary corpus was discarded for the final test runs.

The Tanaka Corpus required that 1.5% of its sentences (about 3,180) be removed due to syntactical errors such as incorrectly numbered lines, missing Japanese or English translations, or because of containing characters outside both the English and Japanese Unicode subset. The Tanaka Corpus also required that roughly 2% (about 4,000) of its sentences be removed because of exact duplications. Jim Breen warns of even more errors in the Tanaka Corpus on his website, but the Tanaka Corpus was the most readily available and correct corpus available on the open source market to use (Breen, 2008), and therefore chosen for this thesis experiment. After all modifications were hand-made to the Tanaka Corpus; 203,226 Japanese-to-English sentence pairs remained to serve as the 100% mark for text alignment training and phrase table construction.

A parsing application was also developed as a result of this thesis out of a need for methods to quickly convert from one application's output to another's input. The parsing interfaces were developed in Java, and greatly reduced the amount of formatting time required when dealing with large corpora. The interfaces requiring newly-developed software were as follows:

1) Tanaka Corpus (Breen, 2008): Initial parsing tools were created to break the consolidated, parallel corpora into sentence ids and sentence into two separate English-sentence-only and Japanese-sentence-only files. Corrections were also made to numbering and sentence syntax errors. Parsing tools to add in Japanese sentence tokenization were also developed for use during this stage of experimentation. Conversion tools to change all text present in the Tanaka Corpus to UTF-8 format (8-bit Unicode, which subsumes the entire Japanese and English languages) were also developed and used on the separate files at this point.
   Subsets of the Tanaka Corpus were used in all experiments divided by percentage of sentence coverage. The 10% version contained the first 10% of the corpus sentence pairs, the 25% version contained the previous 10% plus an additional 15% of the remaining sentence pairs, the 50% version contained the first 25% plus an additional 25% new sentence pairs, and the 100% version contained the entire Tanaka Corpus set.

2) Vocabulary Files: Giza++ requires vocabulary files for each language file which contain all unique tokens and the number of times they appear within a corpus. A simple vocabulary counter application was created and included as part of the thesis MT system that will work for any corpus in any language.

3) Tatoeba Test Set (Trang & Public_Domain_Test_Sentences, 2008): As will be discussed in Section 4.6; the Tatoeba corpus was used as an impartial test bed of sentences after errors and duplication with the Tanaka Corpus were removed. Parsing the Tatoeba corpus required heavy modifications to the Tanaka Corpus parser, but both systems were created, tested, and delivered with the thesis application.

4) JM Dictionary (Breen, 2008): The JM Dictionary corpus required a completely new parsing system, and also resulted in the construction of a many-to-many Hashtable key-pairing system to record all possible kanji Unicode values and link them to all possible English translations; and vice-versa. This Hashtable pairing scheme was needed as a reference by the kanji correction utility explained in Section 4.6.

5) BLEU Metric: The BLEU metric auto-evaluation tool required a specific .xml file format for source, reference, and test files being analyzed for BLEU score. Another file processing application was created to convert both Pharaoh output and kanji correction system output into a format that BLEU could evaluate automatically.

6) Kanji Correction System: The final parsing application developed in the interface subset of thesis artifacts is a tool to apply the kanji correction system to Pharaoh output. Each incorrect translation in the Pharaoh output is fed into the kanji correction analysis, and then output in a format very similar to the original Pharaoh sentence structure (with word corrections).

This concludes the explanation of the parsing artifacts created as a result of this thesis. Every separate application described in previous chapters has been integrated with one another through the above thesis tools which will serve the next person to tackle building a complete end-to-end MT system well.

3.2 Pre-Processing and Tailoring Giza++ Input

Before Giza++ can run basic text alignment functions on parallel corpora, the corpora must be in a specific format and contain as many meaningful tokens as possible. This preprocessing was the first unique contribution to the MT community as Japanese text (along with many East Asian languages such as Chinese and Korean) contain no whitespace between words or sentences. A lack of whitespace (and therefore word tokenization) prevented Giza++

from matching words in the Japanese source language to the target-language English words for any sentence where less than an exact match was found. The Japanese language is also outside the normal ASCII range of character values, and this resulted in several application failures using output produced by Giza++. The solution to this problem was to bring the Japanese language into the Unicode range (which all toolkits used in this thesis understand) and convert back to standard Japanese for final comparisons. Examples of how parallel corpora are uniquely parsed into correct Giza++ inputs are shown in the below figures.

Figure 3.1 **Parallel Corpora Example from the Tatoeba Project**

```
5
お誕生日おめでとうムーリエル！
Happy birthday Muiriel
6
ムーリエルは 20 歳になりました。
Muiriel is 20 now.
7
パスワードは「Muiriel」です。
The password is "Muiriel."
8
すぐに戻ります。
I will be back soon.
9
知らない。
I don't know.
```

The process began with input from the Tanaka Corpus in roughly the same format as the test file above in Figure 3.1. That input was then parsed to remove extraneous data by removing extra lines, extra whitespace, changing words to all lower-case for English sentences, and correcting syntactical mistakes such as Japanese in an English sentence. Further parsing of the corpora produces files with the format demonstrated below in Figure 3.2 and Figure 3.3 (Trang & Public_Domain_Test_Sentences, 2008):

Figure 3.2 **English Extraction from Tatoeba Project**

```
happy birthday muiriel
muiriel is 20 now .
the password is  " muiriel . "
i will be back soon .
i don't know .
```

Figure 3.3 **Japanese/Unicode Extraction from Tatoeba Project**

```
\u304a \u8a95\u751f\u65e5 \u304a\u3081\u3067\u3068\u3046
\u30e0\u30fc\u30ea\u30a8\u30eb !
\u30e0\u30fc\u30ea\u30a8\u30eb \u306f20\u6b73
\u306b\u306a\u308a\u307e\u3057\u305f \u3002
\u30d1\u30b9\u30ef\u30fc\u30c9 \u306f \u300c muiriel\u300d \u3067\u3059
\u3002
\u3059\u3050\u306b \u623b \u308a\u307e\u3059 \u3002
\u77e5 \u3089\u306a\u3044 \u3002
```

As shown in Figure 3.2, English sentences and words required very little additional formatting. Characters that were actual letters were separated from extraneous quotations, grammatical symbols were given surrounding whitespace characters, contractions were left as a single word, and all letters were converted to lowercase.

As shown in Figure 3.3, however, the preprocessing necessary for the Japanese sentences resulted in a file that is unreadable by any human who has not memorized the more than 65,000 total Unicode values. The Japanese text has been separated through several rules isolating Katakana (characters in the Japanese language usually representing words borrowed from foreign languages, or used as emphasis), kanji compounds, and certain grammatical symbols such as the Japanese version of quotations or punctuation. This type of formatting, while complicated for a human being to read, resulted in maximum compatibility with all applications utilized or created in the end-to-end MT system.

After much trial and error, Giza++ eventually produced correct output in the form of translation tables. Giza++ was run bi-directionally for both Japanese-to-English and English-to-Japanese text alignments which resulted in maximum recall for the fairly limited number of tokens in the Tanaka Corpus. A sample of Giza++ generated outputs is shown below in Figure 3.4:

Figure 3.4 **Giza++ Final A3 File Segment Generated from the Tanaka Corpus Bi-Directional Text Alignment Process**

```
# Sentence pair (41) source length 8 target length 9 alignment score :
3.40789e-20
\u300c\u623b \u3063\u3066\u3053\u3044 \u300d \u3068 \u5f7c \u306f \u53eb
\u3093\u3060 \u3002
NULL ({ 4 6 }) " ({ }) come ({ }) back ({ }) ! ({ }) " ({ 3 }) he ({ 5 })
shouted ({ 1 2 7 8 }) . ({ 9 })
# Sentence pair (42) source length 10 target length 5 alignment score :
9.31573e-13
\u964d \u308a\u3066\u304d\u306a\u3055\u3044 \u3002\u6669\u5fa1\u98ef
\u3067\u3059\u3088 \u3002
NULL ({ }) " ({ }) come ({ }) downstairs ({ 1 2 3 4 }) . ({ 5 }) the
({ }) dinner ({ }) is ({ }) ready ({ }) . ({ }) " ({ })
# Sentence pair (43) source length 15 target length 9 alignment score :
3.87545e-22
\u300c \u304a\u3044\u3067\u304a\u3044\u3067 \u300d\u5f7c\u5973 \u306f
\u53eb \u3073\u307e\u3057\u305f \u3002\u300c
\u3053\u3063\u3061\u3067\u3042\u305d\u307c \u300d
NULL ({ 4 }) " ({ 1 }) come ({ }) , ({ }) boy ({ }) , ({ }) " ({ }) she
({ }) whispered ({ 2 3 5 6 7 8 }) , ({ }) " ({ }) come ({ }) and ({ })
play ({ }) . ({ }) " ({ 9 })
```

The table format above represents the alignment of Japanese word tokens to English sentences. As the reader may notice – several English words contain no Japanese token equivalent which introduces one of the major challenges when translating from Japanese to English. The number of tokens in a standard Japanese sentence is usually far lower than the number of English tokens. This results in alignment problems as shown above. Taking a look at the last sentence; only three English tokens of fifteen contain a Japanese text equivalent. Misalignment results in large phrase gaps when performing a beam-search decoding that will be explained in Chapter 5. After the A3 final files are generated for both directions of text alignment, the output needed to be converted a final time for use by the Pharaoh application. The formatting of the Giza++ A3 final files was done by the Thot toolkit as presented in (Ortiz-Martinez, Garcia-Varea, & Casacuberta, 2005) and is explained in the next subsection.

3.3 Thot – Processing Giza++ Output and Constructing Phrase Tables

Thot is a toolkit for creating phrase tables specifically from a format like that produced by the Giza++ text alignment process. Capabilities such as performing operations on word matrices, inverting translation tables, applying various filters to the phrase table being created, and many counting and measuring scripts are included as part of the Thot toolkit (Ortiz-

Martinez, Garcia-Varea, & Casacuberta, 2005). For the purpose of this thesis, only the capability to perform operations between the bi-directional text alignments were used, along with the capability to estimate a complete phrase model based on unions and sums of the Giza++ text alignment output.

The basic Thot operation to create a phrase table based on bi-directional text alignment processes is described in the following equation:

Figure 3.5 **Ortiz-Martinez, Och Formula to Estimate a Phrase Table Based on Bi-Directional Text Alignment Output (Ortiz-Martinez, Garcia-Varea, & Casacuberta, 2005)**

$$\mathcal{BP}(f_1^J, e_1^I, A) = \{(f_j^{j+m}, e_i^{i+n} : \forall (i', j') \in A : j \leq j' \leq j+m \iff i \leq i' \leq i+n\}$$

where *f* is the source word or words, *e* is the target language word or words, and *A* is the word alignment matrix set produced by aligning the text of the parallel corpora. i, *j*, *m*, and *n* are arbitrary iterating variables that search through the source and target sentence to find the maximum products (how many is customizable) of all phrase possibilities.

The second Thot operation utilized for this thesis was the capability to Union, Intersect, Sum, or perform a Symmetrization of the source translation tables. As will be shown in Chapter 5; the Union operation resulted in maximum BLEU score due to the relatively small size of the Tanaka Corpus. When users are performing text alignment and phrase table construction on larger corpora (> one million sentences), the authors of Thot recommend summing several iterations of Thot on chunks of the input as opposed to attempting to process 100% of the input due to memory restrictions and computational time explosions (Ortiz-Martinez, Garcia-Varea, & Casacuberta, 2005).

After Thot had been configured to various settings, a .ttable formatted file was produced for direct input into the Pharaoh decoder. A sample of the Thot output is shown below:

Figure 3.6 **Thot Output Sample from Giza++ Text Alignment, Tanaka Corpus Input**

```
\u300c \u30de\u30a4\u30af \u3001\u98db\u884c\u6a5f \u3063\u3066 \u666e\u901a
\u306f\u3053\u3093\u306a\u3075\u3046\u306b \u63fa
\u308c\u308b\u3082\u306e\u306a\u306e\u304b\u3044 \u300d ||| " mike , do the
planes usually rock like this ? " ||| 1.00000000
\u300c \u30de\u30a4\u30af \u3001\u98db\u884c\u6a5f \u3063\u3066 \u666e\u901a
\u306f\u3053\u3093\u306a\u3075\u3046\u306b \u63fa
\u308c\u308b\u3082\u306e\u306a\u306e\u304b\u3044 ||| " mike , do the planes
usually rock like this ||| 1.00000000
\u300c \u30de\u30a4\u30af ||| " mike ||| 1.00000000
\u201d\u767e\u4e07\u5339 \u306e\u306d\u3053 \u201d ||| " millions of cats . "
||| 1.00000000
```

The format of the output in Figure 3.6 takes the form of:

Source Language Phrase (Japanese / Unicode Tokens) |||
Target Language Phrase (English Tokens) |||
Probability (0.00000000 – 1.00000000)

With the Thot output in hand and waiting on a language model to dictate how the target language tokens should be arranged once translated, the SRILM toolkit was then utilized to create this partner input for the MT system which is now described.

3.4 SRILM vs. CMU-Cambridge Language Modeling Toolkit

As mentioned in Sections 1.2 and 2.2, the main challenge with language models was getting the applications to work in the first place. The actual process of building a language model is fairly streamlined once all of the individual building blocks are in place. The CMU-Cambridge language modeling toolkit did provide some interesting ways to modify language model construction, but the scope of this thesis did not allow for many useful and unique improvements to be made. Therefore, even though the CMU-Cambridge toolkit offers unique future work opportunity as discussed in Chapter 6, only SRILM language models were used in the formal thesis experiments.

As explained in Section *2.2.3*, no further modifications were made to the SRILM source code other than fixing compilation errors. Using the same parallel corpora input to SRILM that was used in Giza++ (the two parallel text files in Japanese and English, and the accompanying vocabulary count files created by another of the thesis applications), SRILM produces output as shown below in Figure 3.7 . The output generated contains only Unigrams through Trigrams probabilities. Any phrase 4-gram and higher resulted in no additional phrase probability increase higher than the cutoff threshold. The format for the SRILM output is:

Probability of N-gram <whitespace> *N-gram* <whitespace> *Back-off Weight*

All probabilities are in logarithm (base 10) format (Stolcke, 2002).

Figure 3.7 Sample SRILM Output Generated from the Tanaka Corpus

```
-0.2850669  imagine yourself to
-0.2850669  introduce yourself ?
-0.1615009  introduced yourself to
-0.1962947  it yourself .
-1.02543    it yourself ?
-1.277574   it yourself by
-0.225869   know yourself .
-0.2886207  make yourself at
-1.098486   make yourself comfortable
-2.07503    make yourself heard
-1.523229   make yourself understand
-0.4866666  make yourself understood
-1.612426   of yourself !
-0.03732432 of yourself .
-0.2361813  out yourself .
-0.2361813  prepare yourself for
-0.1601281  present yourself of
-0.1601281  pull yourself together
-0.05115252 put yourself in
```

With the SRILM output in ARPA (aka Doug Paul) format and the phrase tables built through Giza++ and Thot; both inputs into Pharaoh are completed and the end-to-end MT system has become fully enabled and testable.

3.5 Combining Outputs and Utilizing Pharaoh

As with SRILM – Pharaoh was used largely as-compiled. While several settings are customizable, the settings used are detailed in Chapter 4 as there was no change to the core methodology behind Pharaoh. Pharaoh relies on a configuration file detailing the location of all program inputs and containing any changes to default settings. The language model input to Pharaoh must be provided in the ARPA format, and the phrase table must be in the format producible by Thot as described in Section 3.2. The configuration files for this thesis are included as supplementary documents, but contain settings for the various experiments described in later chapters. A sample Pharaoh run is shown in the figure below:

Figure 3.8 **Sample Pharaoh Execution Based on Europarl Toy Example (Koehn, 2008)**

```
% echo 'das ist ein kleines haus' | bin/pharaoh -f model/pharaoh.ini > out
Pharaoh v1.2.3, written by Philipp Koehn
a beam search decoder for phrase-based statistical machine translation models
(c) 2002-2003 University of Southern California
(c) 2004 Massachusetts Institute of Technology
loading language model from europarl.srilm
loading phrase translation table from phrase-table, \
   stored 21, pruned 0, kept 21
loaded data structures in 2 seconds
reading input sentences
translating 1 sentences.translated 1 sentences in 0 seconds

% cat out
this is a small house
```

As the reader can see – input sentences are fed directly into Pharaoh either through the command line or through a strict file format for multiple sentences. The output from Pharaoh can be displayed directly to the console, or written to another file for further processing and evaluation.

The most important features of Pharaoh utilized to analyze the translation results are the capabilities to both trace an end-translation through from source word to target word, and to execute Pharaoh in a 'verbose' mode which counts and analyzes different translation hypothesis

and final sentence structure decisions. A sample of these operations is shown below in Figure 3.9:

Figure 3.9 Verbose Output Produced by Pharaoh for Europarl Toy Example (Koehn, 2008)

```
   echo 'das ist ein kleines haus' | bin/pharaoh -f pharaoh.ini -v 2
   ...
   collected 12 translation options
   HYP: 114 added, 284 discarded below threshold, 0 pruned, 58 merged.     BEST:
this is a small house -28.9234
```

Results for formal tests on Japanese-to-English translations are contained within Chapter 5. The output produced by Pharaoh is then analyzed via both the BLEU / NIST evaluation metric scripts for a baseline score and by the final part of the thesis artifact – a module to correct contextual errors for kanji identified in the source Japanese sentence as having greater than three possible translations. As the BLEU / NIST scores show in later chapters; comparing the Giza++/SRILM/Pharaoh system to that system **plus** the end-to-end SMT thesis module leads to a very favorable translation quality increase. Human understandability also plays into the judgment, but those subjective observations are reserved for Chapter 6.

Combining the above unique thesis pre-processing, Giza++, Thot, SRILM, and Pharaoh systems results in a baseline translation metric that is detailed in Chapter 4 and Chapter 5. This leads to the post-translation Kanji Correction processing as described in the next subsection that resulted in a unique net gain of BLEU / NIST score.

3.6 Kanji Correction Techniques

The baseline system as described in Section 3.1 through Section 3.4 produced sentences with a large amount (>70% on average) of untouched Japanese text within the final English form. The Japanese text (or Unicode representation of such) remaining in the translated sentence was a result of the low recall ability of the baseline system. Even with 200K+ sentence pairs to work with – the tokenization of the Japanese sentences was not enough to allow Giza++ to assign good correlations between actual Japanese words to their English counterparts or to have a

reasonable vocabulary for the entire Japanese language. Results of the different alignments for different percentages of the training corpus are shown in Chapter 5. With 70% of the sentences having remaining Japanese characters (as Pharaoh did not try to translate words in a source language for which it had no reference), the thesis kanji correction utility had many opportunities to be utilized.

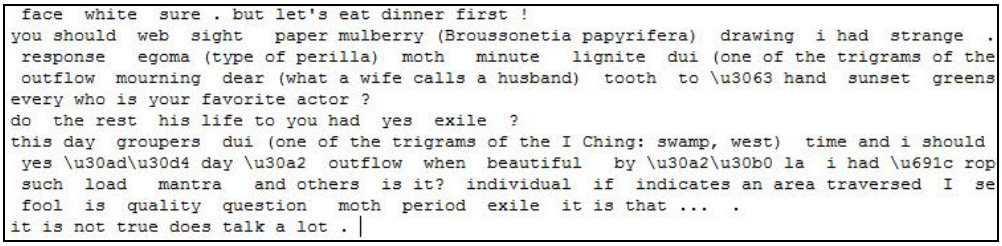
As shown below in Figure 3.10 - text that is not translated by the baseline Pharaoh decoder is represented by every occurrence of the '\uXXXX' symbols. This is the Unicode value of a single Japanese character that Pharaoh was unable to find a match for based on the phrase table and language model inputs.

Figure 3.10 **Pharaoh Output Sample for Japanese-to-English Translation before Kanji Correction Utility Execution**

```
\u9762\u767d sure . but let's eat dinner first !
you should \u30a6\u30a7\u30d6\u30b5\u30a4\u30c8 \u69cb\u56f3 i had \u5909
\u7b54 \u3048\u304c \u5206 \u304b\u3063\u305f\u3093\u3060 , and \u3067\u30
\u3067\u3082\u3042\u306a\u305f\u306f to \u3063\u3066\u304f\u308c\u306a\u30
every who is your favorite actor ?
do \u4ed6 his life to you had \u306f\u3042\u308b ?
this day \u306f\u305f\u3060 time and i should \u7121\u99c4 .
\u30a6\u30a3\u30ad\u30d4\u30c7\u30a3\u30a2 \u3067\u3044\u3064\u304b \u30d0
\u3053\u3093\u306a\u306b \u660e \u3089\u304b\u306a\u3053\u3068\u3092\u308f
\u30d0\u30ab is \u8cea\u554f \u304c\u3042\u308b\u3093\u3060 .
it is not true does talk a lot .
```

As shown below in Figure 3.11, however, the kanji correction utility iterates through all Japanese values attempting to find a better match for a translation for each of the Unicode values. The text in Figure 3.11 corresponds to Figure 3.10 line-for-line. As the results will show; greater than 90% of the original source text is translated to some form of English, although the primitive nature of the translations is also evidenced below:

Figure 3.11 **Pharaoh Output Sample for Japanese-to-English Translation after Kanji Correction Utility Execution**



While the translations are very difficult to understand measured using this writer's human-judgment standards, taking a look at the intermediary steps of kanji correction offers significant hope for future improvements. For instance – the first line of Figure 3.10 which is the pre-kanji correction version reads:

"`\u9762\u767d sure . but let's eat dinner first !`"

And, the first line of Figure 3.11 which is the post-kanji correction version reads:

"`face white sure . but let's eat dinner first !`"

And finally; the actual sentence was supposed to read:

"Interesting, I'm sure. But, let's eat dinner first!"

The thesis application analyzed the individual Unicode characters first (\u9762 and \u767d) and found matches for both kanji in the parsed JM Dict file. These Unicode values map to '面' or 'omo', and '白', or 'shiro' when pronounced together, forming the complete Japanese word '面白い' or 'omoshiroi'. One common meaning of these two kanji together is 'interesting', however, the kanji correction utility mapped each individual kanji to their meanings when encountered alone; namely 'face' and 'white'. Thus, even though it is not obvious from the final translations in Figure 3.11 – each of the translations is far closer to a correct word than is evident at first glance. Even with these areas for improvement still present in the developed thesis applications – a net gain of BLEU score was achieved as is described in Chapter 5.

The kanji correction technique is the final stage of improvement that is within this thesis' scope to offer, and has resulted in an extensible, fully open-source solution to the problem of improving the quality of Japanese to English text translations. The experimental framework and results follow in the next two chapters as proof.

3.7 Machine Learning Specification, Process, and Solutions Summary (Future Work Foundation)

The material presented in Chapter 1, Chapter 2 , and Chapter 3 has resulted in a system of systems built towards a common goal; to improve the quality of Japanese text to English text translation through a Machine Translation system as measured using the BLEU metric. While the original intent of this thesis was to design a system capable of learning patterns to identify incorrect translations based off of surrounding sentence context, that goal is now the next logical step from what turned out to be the ending point of the thesis artifacts. Instead – human learning replaced that component of the thesis for the final implementation (i.e. hard-coded corpora parsing, large-scale interface construction, or simple kanji lookup-and-replacement techniques that were created). This work, however, forms the necessary foundation for the machine learning problem to be solved which is summarized below:

1) **Machine Learning Problem Specification**: The learning problem specification's inputs are the same corpora used as in Giza++ or SRILM. The difference for the machine learning aspect of the problem would be the scope of the search; that is to say, the machine learner such as WEKA (Witten & Frank, 2005) would be searching for kanji with multiple English definitions. If a source sentence contained such a kanji, then the machine learner would flag that sentence as a training example to begin learning contextual definitions under which certain translations occur versus others. The learner's output would be a very simple probability table to be referenced by the extended kanji correction system when potential incorrect translations are discovered. However, instead of containing N-gram orderings as in N-gram probability tables explained in previous sections; the machine learner probability

tables could be based off of the entire sentence, parts of speech, grammatical placements – nearly any element of language that results in greater human understanding. Which element results in the greatest gains, however, is a topic for future study.

2) **Machine Learning Process**: The results of the training process described in #1) above would be accessed anytime a source kanji that has been flagged as a potential mistranslation (judged through raw number of possible translations) is found in a source sentence. The actual translation (accessed through the verbose output of the pharaoh decoder) would then be analyzed for "best fit" measure within the context of the translated sentence. If the "best fit" tolerance was not exceeded when compared to other possible translations, the extended kanji correction system would activate and offer alternative word(s) to better fit the sentence context. For instance, take the Japanese sentence and English translation**:**

あの事件は誰が審理するのですか.

'Ano jiken ha darega shinri suruno desuka?'

'Who is hearing that case?'

The kanji '事件', when combined in the same sentence with '審理', likely means 'case' instead of one of the other five translations (event, matter, affair, incident, or occurrence). This contextual rule would be assigned a certain belief in the form of a probability table by the machine learner, and if '事件' was ever encountered in a Japanese sentence – the extended kanji correction system would activate and judge the quality increase in a substituted word in place of the provided. Again; a rudimentary framework was created in this thesis to provide a proof of concept for contextual definition corrections and offered strong evidence in favor of the hypothesis.

3) **Machine Learning Solution**: After learning as many patterns as possible above a certain threshold of usefulness as defined by the developer, the extended kanji

correction system would analyze any output created by a decoder, alongside the input test sentences. The extended kanji correction system would then analyze each of the source input sentences, scanning for any potential for mistranslation. If a kanji with a high-degree of definitions was ever found, the extended kanji correction system would then search for the corresponding translated word in the English sentence. If an unexpected translation was encountered in the English sentence, then the extended kanji correction system would activate and re-translate the word based on sentence context with the rules learned from the source corpus.

As it stands today; the current kanji correction system located over 90% of the kanji missed by Pharaoh output alone, and chose translations based on the commonness of the translation (more common translations were chosen first) followed by the brevity (shorter definitions were chosen over wordier definitions). Much room for improving the kanji correction system exists, but was deemed beyond the scope of this thesis as the work necessary to create a baseline, segmented-improved, and rudimentary kanji correction utility was deemed sufficient.

CHAPTER 4 - Methodology

In this chapter, I explain the experimental framework and procedures behind each experiment. Each experiment contributes to the logical foundation behind the claim made in Section 1.5; the quality of Japanese-to-English translation, as measured using the BLEU metric, is improved by raising segmentation quality of the Japanese language and through improving kanji-to-English mapping techniques.

4.1 Text Alignment Experiments

Giza++ text alignment was run on training data consisting of 10%, 25%, 50%, and 100% of the available Tanaka Corpus in both directions of language. Bi-directional text alignment is necessary for the Thot toolkit to create phrase tables using the union or intersection methods described in Section 3.2. The mkcls application used to create the necessary vocabulary word classes needed by Giza++ is also included as an executable binary with the Giza++ package. Mkcls was run on the different vocabulary files created by the thesis applications described in the previous subsection on the 10%, 25%, 50%, and 100% increments of the Tanaka Corpus.

4.2 Thot: Phrase Table Building Experiments

The Thot toolkit was used to create alignment matrices in a format exactly like Giza++'s A3.final files (Ortiz-Martinez, Garcia-Varea, & Casacuberta, 2005). Two separate alignment matrices were created using both union and intersections between the bi-directional text alignment files and analyzed for performance. The Thot toolkit was then used to train a phrase-based model using the alignment matrices as input. All settings on the Thot toolkit were left on

default settings; save for the –add and –or options to create union and intersection versions of the phrase tables.

4.3 Language Model Creation Experiments

SRILM was used to create a language model using the 100% version of the Tanaka Corpus as an input. Creating a language model only requires textual sentences in the target language, so only the English version of the Tanaka Corpus sentences were needed as input. Running the *ngram-count* binary packaged with the SRILM toolkit created the language model used in all thesis test cases and stored the model in a file adhering to the ARPA format required by the Pharaoh decoder (Stolcke, 2002).

4.4 Pharaoh Processing

The outputs produced by the Thot and SRILM applications described in Sections 4.3 and 4.4 were used as inputs to the Pharaoh beam-search, phrase-based decoder (Koehn, 2008). Pharaoh was executed using different phrase tables built from a combination of 10%, 25%, 50%, and 100% versions of the Tanaka Corpus combined with the Thot toolkit training using unions or intersections of the text alignment. This resulted in eight unique executions of the Pharaoh decoder. The language model used by Pharaoh was created by SRILM with the 100% English sentence subset of the Tanaka Corpus used as input. This language model was kept constant throughout all tests.

Pharaoh, using the different combinations of the inputs described in the above paragraph, was then used to decode a test bed of the same 635 Japanese test sentences as provided by the Tatoeba Project (Trang & Public_Domain_Test_Sentences, 2008). The results of these translations are presented in Chapter 5: Results. These results were then measured for translation quality using the automated BLEU script (Papineni, Roukos, Ward, & Zhu, 2001). The BLEU

scores earned by the eight different combinations of Pharaoh training data were used as a baseline measure to compare the kanji correction utility's performance against.

4.5 Post-Translation Kanji Correction Experiments

Using the generated Pharaoh output from the experiments in Section 4.4, the kanji correction application was executed on every combination of training data run against the test bed of Japanese sentences. The number of potential matches is determined by the number of incorrectly translated kanji within a sentence from the same Tatoeba Project test bed, which averaged about five to seven firings of the kanji correction system per sentence. These corrected translations were added into the Pharaoh output sentence as new English tokens and not analyzed again by the kanji correction system.

The kanji correction system searched for matches with individual kanji followed by larger kanji combinations, but did not attempt to translate any kanji compound over four Japanese characters long. This helped to limit extra computational cost as the chance of finding a Japanese "word" over four kanji characters long is near non-existent. If a kanji character was found by the kanji correction system, the best-fit kanji definition was chosen based on 1) how common the new definition is for the kanji in question (more common definitions are chosen first), and 2) the brevity of the new English translation (English definitions with fewer words were chosen first over lengthier definitions).

As shown in Chapter 5; executing the kanji correction utility with the original Pharaoh translation as input resulted in a net gain of BLEU score in every single test when compared to the corresponding Pharaoh translation without the kanji correction utility.

4.6 Test Sets

The Tatoeba Project provided a parallel corpus of Japanese-to-English sentences that were not included as part of the Tanaka Corpus (Trang & Public_Domain_Test_Sentences, 2008). The initial cut of sentence pairs from the Tatoeba Project resulted in 2,000 unique sentence pairs however, after further analysis, several duplicates and errors were discovered in the Tatoeba Project corpus that brought the test bed of unique sentences down to 635. All sentences were then parsed via the applications described in Section 4.1, and used as translation tests by the Pharaoh decoder.

4.7 BLEU Metric: Measuring Translation Quality

The BLEU metric (Papineni, Roukos, Ward, & Zhu, 2001)was used as an objective measure of translation quality of every experiment listed in this chapter. A file in the source language, a file used as reference translation, and a file with the translations being measured for quality are provided as an input to the automatic BLEU/NIST Perl script and evaluated. The BLEU and NIST scores are both printed as outputs with overall scores, single N-gram scores, and cumulative N-gram scores. The BLEU and NIST scores were recorded for all eight iterations of the Pharaoh decoder alone, and for all eight iterative executions of the kanji correction utility.

Results for all experiments listed above are explained in the following section, Chapter 5.

CHAPTER 5 - Results

The results of the experiments described in Chapter 4 are listed according to MT phases; namely: Initial Parsing, Text Alignment and Language Models, Initial Decoding, Decoding + Kanji Correction System, and BLEU Evaluation. Each one of these categories is measured according to computational time taken per process, size of input used or size of output produced, and BLEU score contribution. More specialized statistics are provided on a subsection basis.

The goal of this thesis experiment was to increase the BLEU score of a baseline translation system through more correct Japanese text segmentation and by correcting kanji and Japanese words in the final English translation of a given sentence. The activation of the kanji correction system was also required to keep computational time at a reasonable level, though higher than the decoder system alone.

5.1 Results for Initial Parsing

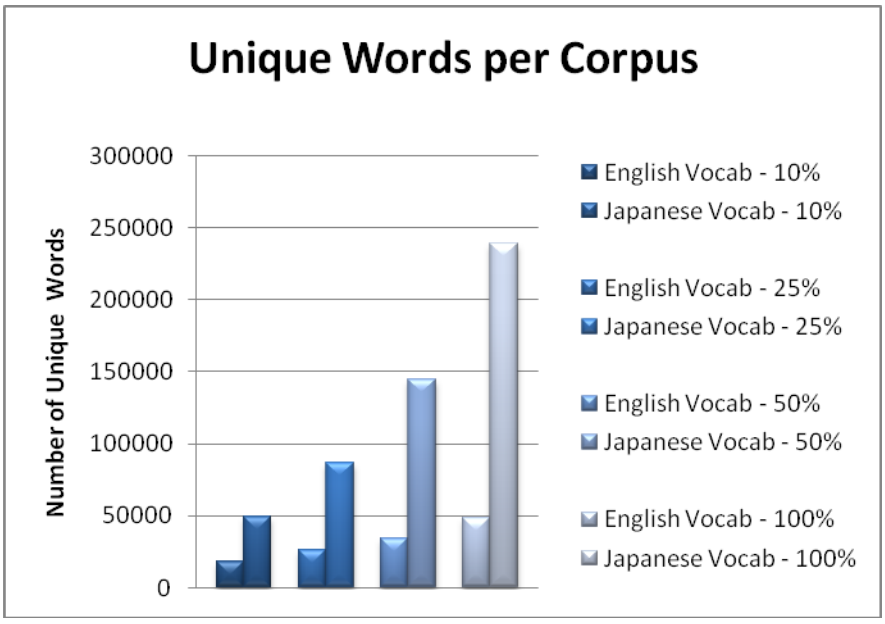
The initial translation experiments of the test sentences trained on a non-segmented version of the Tanaka Corpus resulted in dismal BLEU scores (<.001 for all non-segmented portions). The incredibly low BLEU score was caused by the lack of whitespace between Japanese words, which caused all intermediary processes to be unable to contribute to translating anything less than a complete sentence match. After the thesis application to segment Japanese text was executed on the 8 different subsets of the Tanaka Corpus, the costs, iterations, processing times, and unique vocabulary tokens were produced as output. The important items to note that support the conclusions made in Chapter 6 are listed below in Table 5.1:

Table 5.1 mkcls Results for Eight Different Portions of the Tanaka Corpus

Giza++: mkcls Target (En/Ja) – (% of Tanaka Corpus Used)	Processing Time (seconds)	Unique Words
English Vocab - 10%	351	17914
Japanese Vocab - 10%	889	49494
English Vocab - 25%	501	26108
Japanese Vocab - 25%	1548	86868
English Vocab - 50%	638	34097
Japanese Vocab - 50%	2720	145151
English Vocab - 100%	920	48772
Japanese Vocab - 100%	6081	239631

The important take-away from Table 5.1 is better represented by the graphical version of the data in Figure 5.1 below:

Figure 5.1 **English and Japanese Unique Vocabulary Words per Tanaka Corpus Subset**



The sheer number of unique Japanese tokens when compared to the corresponding English tokens is a minimum of three times greater, and approaches nearly six times greater with the full Tanaka Corpus. This difference in vocabulary size contributes to the difficulty for MT

systems when translating from Japanese to English as Giza++ must attempt to find a one-to-one matching between tokens in both languages.

Unique words were distributed into 80 word classes for each language with ten iterations of the mkcls application run for maximum, reasonable accuracy (Och & Ney, Improved Statistical Alignment Models, 2000). The .classes and .cats files are included with the thesis supplementary material.

5.2 Results for Giza++ Training

As mentioned in Section 4.2, the Giza++ text alignment tool needed to be run in both the Japanese-to-English and the English-to-Japanese directions to build a higher-quality alignment table. The results from each iteration of Giza++ for each subset of the Tanaka Corpus are shown below in Tables 5.2 and 5.3:

Table 5.2 Giza++ Results from Execution on Source Language Sets

Giza++	Processing Time (seconds)	Source Language Unique Tokens	Total Unique Source Tokens	Total Source Language Tokens
En-To-Ja - 10%	216	8999	8998	187629
Ja-To-En - 10%	214	24747	24746	166482
En-To-Ja - 25%	578	13110	13109	450485
Ja-To-En - 25%	571	43434	43433	428966
En-To-Ja - 50%	896	17139	17138	911347
Ja-To-En - 50%	1127	72575	72574	831583
En-To-Ja 100%	2204	24588	24587	1803020
Ja-To-En 100%	2149	119815	119814	1665860

Table 5.3 Giza++ Results from Execution on Target Language Sets

Giza++	Target Language Unique Tokens	Total Unique Target Tokens	Total Target Language Tokens	Size of A3 Table (MB)	Total Sentence Pairs
En-To-Ja - 10%	24747	24743	166441	6.65	20594
Ja-To-En - 10%	8999	8988	187283	6.57	20594
En-To-Ja - 25%	43434	43431	428925	16.21	51504
Ja-To-En - 25%	13110	13101	449987	16.13	51504
En-To-Ja - 50%	72575	72572	831542	32.26	101712
Ja-To-En - 50%	17139	17121	909925	31.97	101712
En-To-Ja 100%	119815	119795	1665670	64.75	203205
Ja-To-En 100%	24588	24558	1799760	64.24	203205

Again, as demonstrated by the results from Giza++ text alignment, the number of unique tokens on the Japanese language side is roughly five times greater than the number of unique English tokens. The total numbers of tokens for both languages in the full Tanaka Corpus set, however, are within 10% of each other. This fact reinforces the idea that Japanese and English languages have vastly different writing syntax which is expected due to the numerous kanji and kanji compounds contained in the Japanese language. Furthermore, since the thesis parsing application was designed to segment the original Japanese text into these unique tokens – the evidence from Tables 5.2 and 5.3 clearly demonstrates that the thesis artifact was successful in this task. Upon further inspection of the A3 files created by Giza++ (which map words in a source language numerically to words in a target language), the Japanese text was found to be segmented into meaningful kanji compounds, individual kanji, borrowed katakana words, and grammatical markings. As the base BLEU score for the Japanese text translation without segmentation was below .001 – the BLEU metric increase due to the increased segmentation, combined with the above facts from the Giza++ executions, proved that the segmentation provided by the thesis application for Japanese text directly correlated with BLEU metric score. The comparison and total measure of quality increase are fully described in Section 5.5.

5.3 Thot Union and Intersection Phrase Table Experiment Results

Executions of the Thot toolkit resulted in further processing times for recording and measures of ttable size. No other setting from the default Thot toolkit values were used, save for the –and and –or options to create a union and intersection version of the phrase table probabilities; respectively. The correlation between unions and intersections of alignment tables produced by Giza++ are analyzed in Section 5.6. Results from the Thot execution are shown below in Table 5.4:

Table 5.4 Results from Thot Executions on Tanaka Corpus Subsets

Thot	Processing Time (seconds)	Size of ttable file (MB)
10% And Totals	551	115.88
10% Or Totals	127	8.07
25% And Totals	1555	281.12
25% Or Totals	385	29.54
50% And Totals	2758	542.47
50% Or Totals	743	57.95
100% And Totals	5517	1,084.57
100% Or Totals	1377	110.88

The results shown above in Table 5.4 were only used when computing total percent increase in computational time of the baseline system vs. the kanji correction system, and in gauging needed input file size increases due to the added thesis functionality.

The SRILM toolkit was utilized at this point to create a language model based off of the 100% Tanaka Corpus containing only English sentences. Total processing time was measured at 35 seconds, and language model size was 12.42 MB.

5.4 Pharaoh Decoding and Kanji Correction System Results and Comparisons

It should first be noted that the kanji correction system was run on an unmodified version of the Tatoeba test sentences and resulted in a .019 normalized BLEU score. This lowest-score baseline is to be expected, since the kanji correction system is meant only to analyze decoder output that has been translated and reordered to the best of the phrase-based model's ability.

The previous sections in Chapter 5 have described the statistics for all of the pre-processing that must take place before an actual Japanese-to-English translation occurs. These correlations will be explained following this subsection detailing the results from the Pharaoh decoder executed on the test set sentences alone, and the output from the kanji correction system executed on the identical Pharaoh output. Both translation sets were then measured for translation quality by the automatic BLEU script.

Results from the Pharaoh decoder executions alone are shown below in Table 5.5:

Table 5.5 Pharaoh Decoder Results from Tanaka Corpus Subsets with Normalized BLEU Scores

Pharaoh	Processing Time (seconds)	BLEU Score 1-Gram	BLEU Score 2-Gram	BLEU Score 3-Gram	Normalized BLEU Score
10% And Total	84	0.0653	0.0181	0.0085	0.030633333
10% Or Total	15	0.0505	0.0191	0.0123	0.0273
25% And Total	200	0.0785	0.0242	0.0113	0.038
25% Or Total	34	0.0523	0.0206	0.0133	0.028733333
50% And Total	391	0.1	0.0329	0.0171	0.05
50% Or Total	60	0.0647	0.0277	0.0181	0.036833333
100% And Total	10000	0.1214	0.0414	0.0214	0.0614
100% Or Total	105	0.0767	0.0342	0.0217	0.0442

And, following are the results from the kanji correction system executing on the Pharaoh output that generated the results shown in Table 5.5. These results are the final scores for the

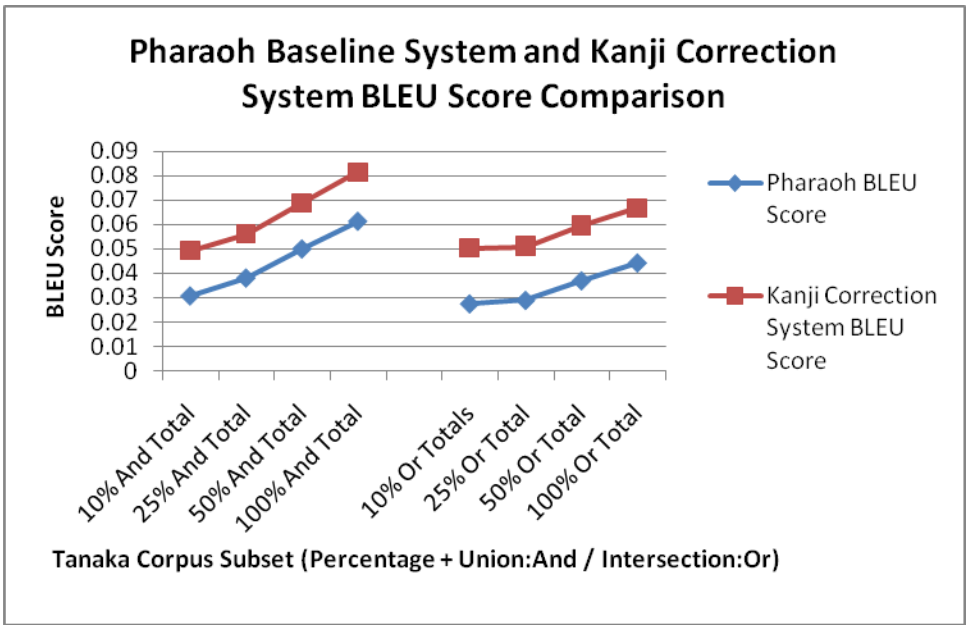
kanji correction system after improving, to the best of its ability, the output produced by the Pharaoh decoder on the eight subsets of the Tanaka Corpus.

Table 5.6 Pharaoh Decoder Results from Tanaka Corpus Subsets with Normalized BLEU Scores

Kanji Correction System	Time (sec)	Kanji Correction System Firings	BLEU Score 1-Gram	BLEU Score 2-Gram	BLEU Score 3-Gram	Normalized BLEU Score
10% And Total	05	3822	0.1066	0.0282	0.013	0.04926666
10% Or Totals	24	5899	0.0976	0.0332	0.0197	0.05016666
25% And Total	2	2864	0.1167	0.0351	0.0161	0.05596666
25% Or Total	11	5616	0.0973	0.0349	0.0208	0.051
50% And Total	6	2324	0.1391	0.0444	0.0229	0.0688
50% Or Total	08	5049	0.1094	0.0426	0.0263	0.05943333
100% And Total	9	2015	0.1625	0.054	0.0279	0.08146667
100% Or Total	07	4699	0.1206	0.0496	0.0301	0.06676666

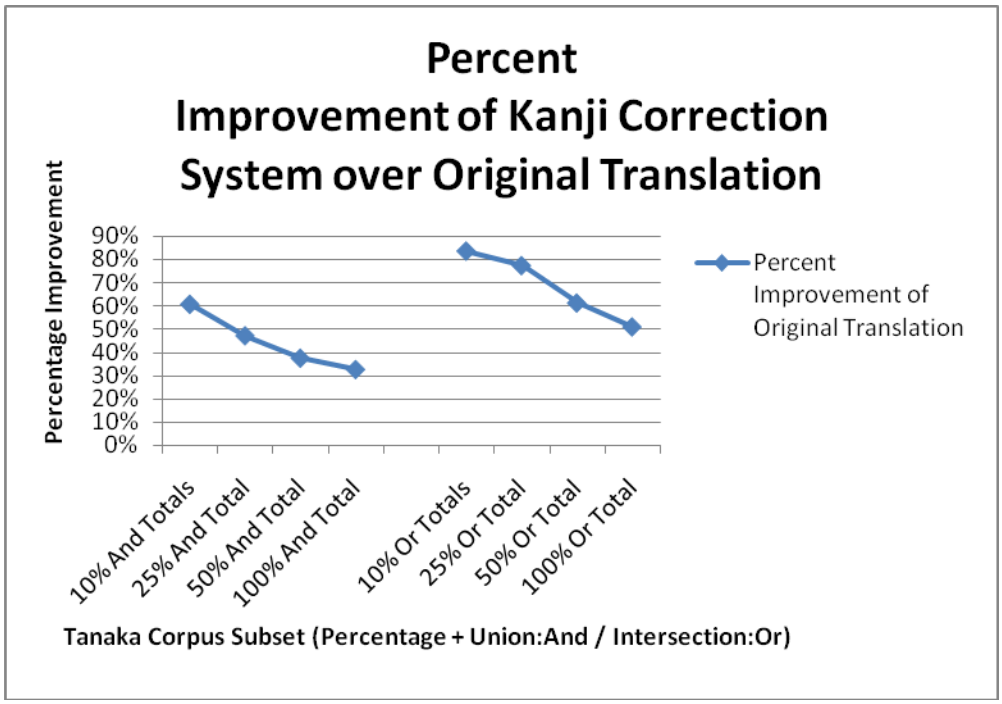
The key item of note for proof that the kanji correction system does indeed yield an increase in BLEU score is the Normalized BLEU score column. Compared to the Pharaoh results for the corresponding subsets of the Tanaka Corpus, the following graph (Figure 5.2) is generated:

Figure 5.2 **BLEU Score Comparison between Pharaoh and Kanji Correction System Results**



As Figure 5.2 shows, the kanji correction system produced as a thesis artifact increases overall BLEU metric score compared to the Pharaoh decoder alone by .18 to .23 for each Tanaka Corpus subset – a minimum of 32% to a maximum of 84% improvement. This increase is in addition to the one-time improvement produced by more meaningful thesis artifact segmentation method that produced the initial Pharaoh baseline system. The increase as a percentage for the kanji correction system is shown in the next figure below:

Figure 5.3 **Percent Improvement of Original Translation after Kanji Correction System Execution as Measured using the BLEU Metric**



As the reader should note – the percentage increase provided by the kanji correction system is subject to diminishing returns as the subset of the Tanaka Corpus increases in size. This is to be expected as with larger and larger corpora the baseline MT system should be able to learn more word alignments with higher accuracy, as well as produced a higher-quality language model as inputs to a phrase-based decoder.

The increase in translation quality did not occur without extra memory, physical hard drive space, and timing costs which have been recorded in the following subsection.

5.5 Computational Cost and Physical Hard Drive Space Metric Analysis

The criteria used for evaluating costs of the baseline MT system and the additional kanji correction system were the amount of time used for both pre-processing corpora and actual translation times, and the physical hard drive space required.

The size of the Tanaka Corpus was 36.65 MB, the size of the Kanji Dictionary was 13.26 MB, and the size of the JM Dictionary was 31.50 MB. Processing time to obtain these files was negligible.

The time required to perform Japanese text segmentation and Unicode conversion, along with parsing the Tanaka Corpus into both Japanese and English files is recorded in the following table:

Table 5.7 Results from Thot Executions on Tanaka Corpus Subsets

Tanaka Corpus Percent Coverage	Parse Time (seconds)
10%	7
25%	12
50%	27
100%	83

The times necessary to perform the mkcls pre-processing necessary to create Giza++ inputs are recorded in the following table:

Table 5.8 mkcls Processing Times

Tanaka Corpus Subset	mkcls Processing Time (seconds)	Physical Hard Drive Space (MB)
English Vocab - 10%	351	.40
Japanese Vocab - 10%	889	2.66
English Vocab - 25%	501	.61
Japanese Vocab - 25%	1548	4.96
English Vocab - 50%	638	.80
Japanese Vocab - 50%	2720	8.99
English Vocab - 100%	920	1.14
Japanese Vocab - 100%	6081	15.4

The processing times of Giza++ and the output A3 ttable files created for use by Thot are shown in Table 5.9:

Table 5.9 Giza++ Processing Time and Physical Hard Drive Space Requirements

Giza++	Processing Time (seconds)	Size of A3 (MB)	Total Sentence Pairs
English-To-Japanese - 10%	216	6.65	20594
Japanese-To-English - 10%	214	6.57	20594
English-To-Japanese - 25%	578	16.21	51504
Japanese-To-English - 25%	571	16.13	51504
English-To-Japanese - 50%	896	32.26	101712
Japanese-To-English - 50%	1127	31.97	101712
English-To-Japanese - 100%	2204	64.75	203205
Japanese-To-English - 100%	2149	64.24	203205

The results of the Thot processing after Giza++ had produced bi-directional A3 ttable files are shown above in Table 5.4. The computational time required for SRILM to create the language model used for all 8 subsets of the Tanaka Corpus was 35 seconds, and physical hard drive space required was 12.31 MB for the final version.

Finally, the costs of the Pharaoh decoding and the kanji correction system are shown here:

Table 5.10 Giza++ Processing Time and Physical Hard Drive Space Requirements

Pharaoh	Processing Time (seconds)		Kanji Correction System	Processing Time (seconds)
10% And Total	84		10% And Totals	105
10% Or Total	15		10% Or Totals	124
25% And Total	200		25% And Total	92
25% Or Total	34		25% Or Total	111
50% And Total	391		50% And Total	86
50% Or Total	60		50% Or Total	108
100% And Total	86504[1]		100% And Total	89
100% Or Total	105		100% Or Total	107

The times shown in Table 5.10 include the time required to load all necessary language modules and translate the test bed of 635 Japanese sentences. The times for the kanji correction system are in addition to the Pharaoh processing times to translate the original sentences once using the associated phrase table and language model.

This concludes the objective report of all experimental data collected from the first pre-processing utilities to the final translated English sentences. The data present in this chapter was

[1] The phrase table produced by the Union between Japanese and English vocabulary intersections for the full Tanaka Corpus was 1.08 GB in size. The processing necessary for Pharaoh to use this model to translate a sentence resulted in memory thrashing on the test machine, and thus an anomalously high processing time was measured.

synthesized from this form into more meaningful relationships that prove translation quality was increased by the addition of segmentation and the kanji correction system, and several interesting trends were also noted that contributed to future work potential. The analyzed data and potential work for continuing the systems developed are explored in the next chapter, Chapter 6.

CHAPTER 6 - Conclusions and Future Work

The hypothesis that the quality of Japanese-to-English translation can be improved through customizing common segmentation techniques to the Japanese language and through correcting kanji characters that the original Pharaoh baseline system could not is heavily supported by the experimental process described in Chapter 4 and the results recorded in Chapter 5 . The relevant pieces of evidence supporting the thesis claims are now summarized in Section 6.1.

6.1 Thesis Claims' Evidence

The fact that the Giza++ text alignment tool was not able to correctly align anything except a 100% sentence match when the Japanese words were not separated by whitespace clearly indicates that segmentation improves the quality of translation. The fact that such a strong positive correlation between logical segmenting of Japanese text and initial BLEU score (beginning at a .0273 to .0614 initial jump from data taken from Table 5.5) also provided a baseline off of which to judge all further translation quality increases. This increase was enabled through the thesis artifact by parsing the parallel corpus text in the following ways:

1) Separating katakana words from surrounding Japanese characters
2) Separating grammatical marking from surrounding Japanese characters
3) Separating kanji compounds from surrounding Japanese characters
4) Separating non-Japanese words in Japanese sentences from surrounding Japanese characters

The thesis claim of improving translation quality through specialized segmentation is supported finally by the fact that all inferior segmentation techniques (separating every character,

separating only grammatical marks, or separating only katakana words) resulted in very low BLEU scores (< .001 normalized).

These parsing techniques have been specifically tailored for the Japanese language, but could be applied to any East Asian language such as Chinese or Korean with very little modification.

The lesion study described in Sections 4.5, 4.6, and 4.7 set a clear framework for evaluating the Pharaoh baseline system with and without the kanji correction system. The baseline Pharaoh BLEU scores had been established through translating 635 Japanese test sentences using phrase tables created by using specialized segmentation over 8 subsets of the Tanaka Corpus. These results indicated a positive correlation between the size of the corpus used for training and the Pharaoh decoder BLEU score results as shown in Figure 5.2. These results were produced completely independent of any kanji correction utility activation. The kanji correction system was then activated the maximum number of times per sentence; that is to say - for every encountered token in the Pharaoh output, the kanji correction system activated if a token was found that matched a known Japanese-to-English translation.

The incorrectly-translated token was then replaced with the closest-matching English translation that kept the translated sentence under the brevity penalty for the BLEU metric as much as possible. The new definition was sorted secondarily on the popularity of the target translation. So – new definitions were chosen on definition length and then by commonality in the English language.

The kanji correction system techniques resulted in the translation quality gains shown in Figure 5.2 which ranged from 32% - 84% increases in BLEU score for each Tanaka Corpus subset. Since these increased BLEU score results were produced independent of any other variable save the kanji correction system's activation, the evidence strongly suggests that the kanji correction system is solely responsible for the BLEU score increase which corroborated the second thesis claim.

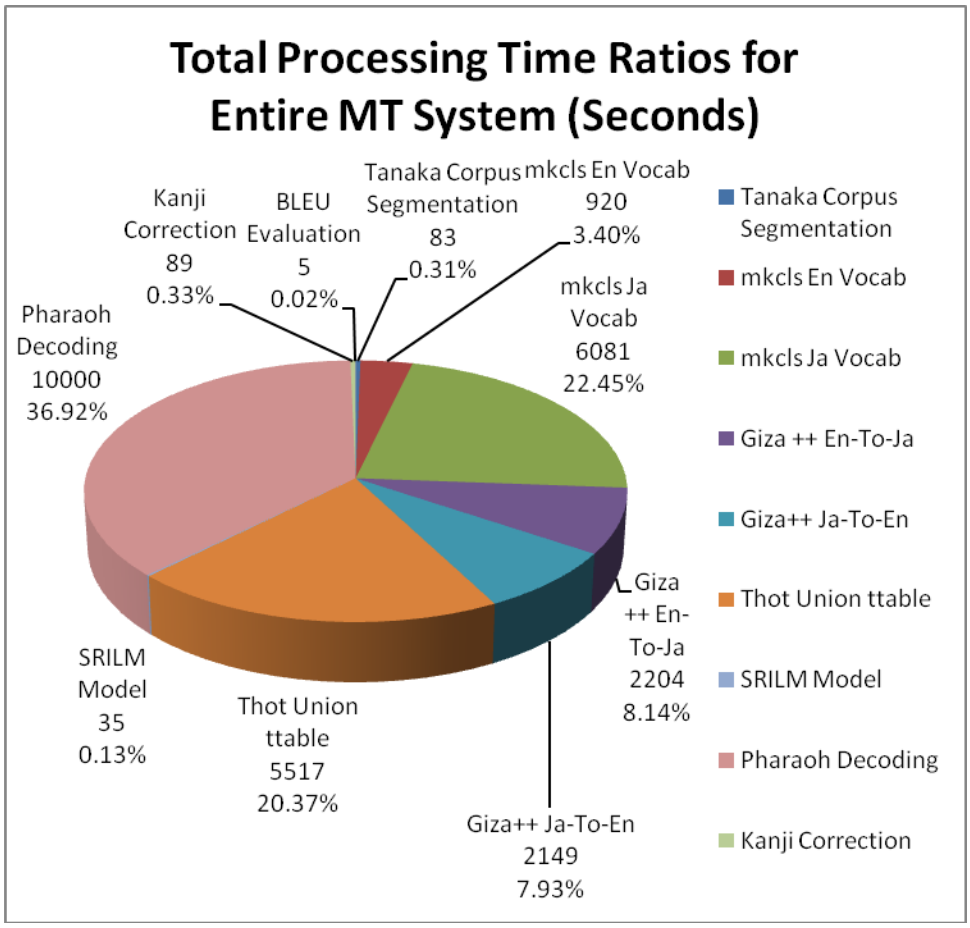
6.2 Cost-to-Value Ratios of Translation Quality Increases

The increase in BLEU score due to the kanji correction system's activation was not without computational and performance timing costs. The performance costs, however, were found to be minimal when taking into account total processing time and normalizing the performance to a per-sentence basis. The relationship between total processing time and kanji correction system costs are best explained by Figure 6.1 below.

The total processing time addition of the kanji correction system is .33% extra (or 89 seconds) when correcting up to 2015 kanji translation errors over 635 sentences. These results average to .140 seconds additional processing time per sentence assuming roughly three errors per sentence – a more than acceptable cost for a .18 to .23 gain in BLEU score.

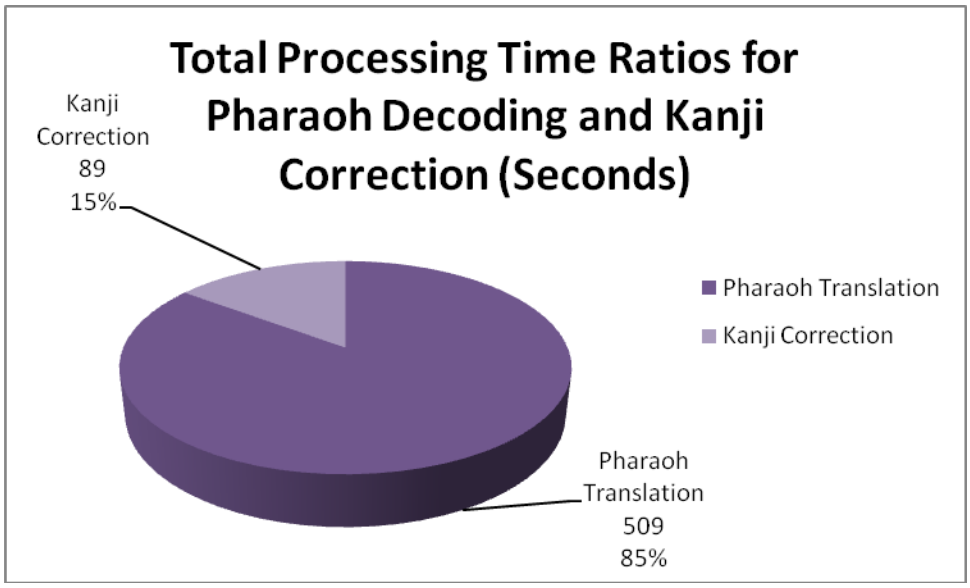
Another standard of measure was to compare the processing time of the kanji correction system solely to that of the time it took Pharaoh to translate a sentence after all pre-processing had been completed. This ratio is not as supportive of the kanji correction system, and is shown in Figure 6.2.

Figure 6.1 **Total Processing Time for MT Subsystems**



As shown in Figure 6.2, the kanji correction system accounts for 15% of the total translation time when compared to the duration only of the final translation phase of the MT process. A 15% increase in computational time was more than acceptable for the BLEU metric gain.

Figure 6.2 **Pharaoh Translation Time vs. Kanji Correction Processing Time**



The physical hard drive space associated with this thesis did not factor into the evaluation of overall system effectiveness. It is important to note, however, that space for all pre-processing applications, language files, generated phrase tables, language models, and final output translations must be allocated for the performance of any MT system to be reasonable. 1.48 GBs were required for the full execution of the 100%-And (Union) version of the Tanaka Corpus and future students of SMT must keep in mind that the 200K sentence pair example is small when compared to other language pair corpora.

System memory requirements also increase rapidly as processing Union phrase tables becomes based on larger parallel corpora. Several of the toolkits discussed in this thesis are capable of partitioning sections of parallel corpora into more reasonable chunks, but further analysis was deemed outside the scope of this experiment.

6.3 Contributions to the Natural Language Processing Community

This thesis has resulted in the creation of three key artifacts to contribute to the future development of MT research.

1) A previously-unavailable (through open-source code) application for segmenting East Asian language text into any number of parallel sentence files.

2) A kanji correction system to parse any Japanese sentence (but preferably decoder output) and locate best-match translations for incorrectly-translated kanji, katakana, and hiragana words.

3) An interface kit that serves to streamline every process of MT discussed in this thesis. New applications were delivered that pre-process parallel corpora, create tailored vocabulary files, analyze all processes for runtimes and sentence correction metrics, alter Pharaoh output to a form for kanji correction use, analyze dictionary files to extract many-to-many relationship matches between language pairs and analyze that data for best-fit matches, convert any format of textual input to any other international standard, and finally that analyze decoder output and converts those sentences to a form that can automatically be evaluated by the BLEU script.

6.4 Future Work

Constructing an entire end-to-end MT system offered several avenues for future research. The positive relationship between segmentation and text alignment quality has been given strong evidence from this thesis' results, and it would be interesting to apply higher-level segmentation techniques such as advanced grammatical parsing, part-of-speech tagging, or politeness-level translations and observe the effect on end-translation quality.

The diminishing returns on the current kanji correction system must also be taken into account for future work. While the parallel corpus size was only 200K sentences for this experiment and thus resulted in large one-time BLEU score gains; as the corpus size increases so too should the MT system's recall and precision. Japanese and English corpora are more difficult to find than most language pairs, but eventually enough data could be gathered to offset the gains of the current hard-coded kanji correction algorithm. Improvements to the kanji

correction system could be made to analyze all aspects of kanji with multiple definitions based on the original Japanese text input – but the quality of the input corpora have a long way to go before this kind of machine learning could be applied with positive results.

Work to implement the machine learning portion of the thesis is also a next logical step to the completion of this work, described in Section 3.7. While the human-learning element of this thesis resulted in enough baseline improvement to justify the completion of this thesis – the methodology behind such improvements should be generalized through a machine learning process applicable to a larger scope of Japanese-to-English text translation.

6.5 Conclusions

The positive correlation between the quality of segmentation and the end BLEU score of the constructed MT system has clearly been proven by the increase in BLEU score from a starting point of <.001 to a maximum baseline starting score of .061 when segmentation was fully implemented. The fact that an increase in BLEU score of 32% - 84% occurred for identical test sets on every execution of the kanji correction utility on all 8 subsets of the Tanaka Corpus completely supports the claim that kanji correction for baseline Pharaoh output is possible and results in BLEU score gains – especially for systems trained on smaller amounts of parallel text.

While commercial tools such as Google Translate can score up to as high as .35 as measured using the BLEU metric on Japanese-to-English translations; they also have a vastly larger supply of resources and parallel texts to draw from. With identical input corpora used as training material, it would be interesting to see how each system stacks up against one another. As the amount of initial parallel text increases in quantity and quality that are made available to the public, more advanced Japanese text correction algorithms will need to be developed in order for the gains to maintain the level of incremental improvement achieved in this thesis research over the baseline scores, and this author is looking forward to the future challenges of creating them.

Bibliography

Akiba, Y., Sumita, E., Nakaiwa, H., Yamamoto, S., & Okuno, H. G. (2004). Using a Mixture of N-Best Lists from Multiple MT Systems in Rank-Sum-Based Confidence Measure for MT Outputs. *Paper presented to ATR Spoken Language Translation Research Laboratories* .

Al-Onaizan, Y., & Knight, K. (2002). Named Entity Translation. *Human Language Technology Conference: Proceedings of the second international conference on Human Language Technology Research* (pp. 122 - 124). San Diego: Morgan Kaufmann Publishers Inc.

Breen, J. (2008, May). *The Tanaka Corpus*. Retrieved October 30, 2008, from Jim Breen's Home Page: http://www.csse.monash.edu.au/~jwb/tanakacorpus.html

Brill, E. (2000). Pattern-Based Disambiguation for Natural Language Processing. *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora; Annual Meeting of the ACL, held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics* (pp. 1 - 8). Hong Kong: Association for Computational Linguistics.

Clarkson, P., & Rosenfeld, R. (1999, June 7th). *Statistical Language Modeling: Using the CMU-Cambridge Toolkit.* Retrieved November 30, 2008, from Statistical Language Modeling Toolkit : http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html

Dellaert, F. (2002). The Expectation Maximization Algorithm. *Paper presented as Technical Report GIT-GVU-02-20 to College of Computing, Georgia Institute of Technology* , 1-7.

Deng, Y., & Byrne, W. (2005). HMM Word and Phrase Alignment for Statistical Machine Translation. *Human Language Technology Conference: Proceedings of the Conference*

on Human Language Technology and Empirical Methods in Natural Language Processing (pp. 169 - 176). Vancouver, British Columbia, Canada: Association for Computational Linguistics.

Federico, M., Bertoldi, N., & Cettolo, M. (2008, November 30). *IRSTLM: an Open Source Toolkit for Handling Large Scale Language Models.* Retrieved November 30, 2008, from Language Technologies: Resources for NLP, SMT, and Corpus Works: http://www.suffix.com/papers/files/federicoEtAl-interspeech08.pdf

Fraser, A., & Marcu, D. (2007). Measuring Word Alignment Quality for Statistical Machine Translation. *Computational Linguistics* , 293 - 303.

Goodman, J. (2008, September 8). *SRI International*. Retrieved November 30, 2008, from SRILM - The SRI Language Modeling Toolkit: http://www.speech.sri.com/projects/srilm/

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. *Paper presented to School of Informatics - University of Edinburgh, Scotland for Machine Translation Summit 2005* .

Koehn, P. (2008, November 30). *Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models.* Retrieved November 30, 2008, from University of Southern California - Information Sciences Institute: http://www.isi.edu/licensed-sw/pharaoh/

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., et al. (2008, February 6). *Moses Homepage*. Retrieved November 30, 2008, from Moses: A Factored Phrase-Based Beam Search Decoder for Statistical Machine Translation: http://www.statmt.org/moses/index.php?n=Main.HomePage

Kumar, S., & Byrne, W. (2002). Minimum Bayes-Risk Word Alignments of Bilingual Texts. *Annual Meeting of the ACL: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10* (pp. 140 - 147). Association for Computational Linguistics.

Lin, D. (2004). A Path-based Transfer Model for Machine Translation. *International Conference On Computational Linguistics: Proceedings of the 20th International Conference on Computational Linguistics* (p. Article No. 625). Geneva, Switzerland: Association for Computational Linguistics.

Marino, J. B., Banchs, R. E., Crego, J. M., Gispert, A. D., Lambert, P., Fonollosa, J. A., et al. (2006). N-gram-based Machine Translation. *Computational Linguistics: Volume 32, Issue 4* , 527-549.

May, J., & Knight, K. (2006). A Better N-Best List: Practical Determinization of Weighted Finite Tree Automata. *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference* (pp. 351-358). New York: Association for Computational Linguistics.

Melamed, D. (2004). Statistical Machine Translation by Parsing. *Annual Meeting of the ACL: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics* (p. Article No. 653). Barcelona, Spain: Association for Computational Linguistics.

Ney, H., & Essen, U. (1991). On Smoothing Techniques for Bigram-Based Natural Language Modelling. *Proceedings of the Acoustics, Speech, and Signal Processing 1991 International Conference* (pp. 825-828). Washington, D.C.: IEEE Computer Society.

Nightingale, S., & Tanaka, H. (2003). Comparing the Sentence Alignment Yield from Two News Corpora Using a Dictionary-Based System. *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond - Volume 3* (pp. 119 - 122). Edmonton, Canada: Association for Computational Linguistics.

Och, F. J. (1999). An Efficient Method for Determining Bilingual Word Classes. *European Chapter Meeting of the ACL: Proceedings of the Ninth Conference on European*

Chapter of the Association for Computational Linguistics (pp. 71-76). Bergen, Norway: Association for Computational Linguistics.

Och, F. J., & Ney, H. (2000). Improved Statistical Alignment Models. *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (pp. 440-447). Hong Kong: Association for Computational Linguistics.

Ortiz-Martinez, D., Garcia-Varea, I., & Casacuberta, F. (2005). Thot: A Toolkit to Train Phrase-Based Models for Statistical Machine Translation. *Tenth Machine Translation Summit.* Phuket, Thailand: Asian-Pacific Association for Machine Translation.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). Bleu: A Method for Automatic Evaluation of Machine Translation. *Annual Meeting of the ACL: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 311 - 318). Philadelphia: Association for Computational Linguistics.

Stolcke, A. (2002). SRILM - An Extensible Language Modeling Toolkit. *Proceedings of International Conference on Spoken Language Processing, Vol. 2*, (pp. 901 - 904). Denver.

Suzuki, H., & Toutanova, K. (2006). Learning to Predict Case Markers in Japanese. *Annual Meeting of the ACL: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (pp. 1049 - 1056). Sydney, Australia: Association for Computational Linguistics.

Trang, (., & Public_Domain_Test_Sentences. (2008, July 27). *Tatoeba Project - Miscellaneous - Download Tatoeba Sentences*. Retrieved November 30, 2008, from Tatoeba Project: http://tatoeba.fr/?site_lg=en

Utiyama, M., & Isahara, H. (2003). Reliable Measures for Aligning Japanese-English News Articles and Sentences. *Proceedings of the 41st Annual Meeting on Association for*

Computational Linguistics - Volume 1 (pp. 72-79). Sapporo, Japan: Association for Computational Linguistics.

Wang, W., Knight, K., & Marcu, D. (2006). Capitalizing Machine Translation. *Human Language Technology Conference: Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* (pp. 1 - 8). New York: Association for Computational Linguistics.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques.* San Francisco: Morgan Kaufmann Publishers Inc.

Zens, R., Ney, H., Watanabe, T., & Sumita, E. (2004). Reordering Constraints for Phrase-Based Statistical Machine Translation. *International Conference On Computational Linguistics: Proceedings of the 20th International Conference on Computational Linguistics* (p. Article 205). Geneva, Switzerland: Association for Computational Linguistics.