

Curious George Learns About Machine Translation

by

Eric H. Davis

11-731 Machine Translation

Carnegie Mellon University

Language Technologies Institute

Professor Teruko Mitamura

May 7, 2007

Acknowledgments

Many thanks to Erik Peterson for calmly and patiently walking me through the intricacies of the AVENUE system. He greatly aided in debugging my rules, explaining all the facets of the system, and lending advice when needed. Without his help and sage advice, this project would not have been nearly as good.

Thanks also go to Greg Hanneman, who provided me with two language models.

I would also like to thank Teruko Mitamura for advising this project and offering helpful tips and advice about designing an MT system in general. Her advice and support kept me from losing my mind at times during this project.

Next, thanks go to the professors of the MT class. Your dedication to presenting the material in an interesting and fascinating way has definitely piqued my interest. Without the knowledge gained in this class, this project would not have been possible in the first place.

METEOR provided free of cost courtesy of Alon Lavie from <http://www.cs.cmu.edu/~alavie/METEOR/>.

BLEU and NIST provided free of charge courtesy of National Institute of Standards and Technology. <http://www.nist.gov/speech/tests/mt/resources/scoring.htm>

Finally, I would like to thank my fiancée, Julie, for her love and support during this project. She provided countless insights about the Korean language, as well as two reference translations. Having her as a native consultant made life much easier and ultimately lead to better results.

1. Introduction

"Is not easy however." In a stunted way, this describes the overall sentiment about machine translation (MT). Due to syntactic, semantic, and computational issues, machine translation is definitely not easy. However, that is not to say that machine translation is an intractable problem. With a careful analysis of the both the source and target languages as well as access to ample data, a native informant or both, a feasible solution to the problem is possible. Still, various complications, such as polysemy, ambiguity, and divergent syntax arise, and a sound MT system must find ways to resolve such issues. (MT class lecture)

There are several different types of MT systems, each with their strengths and weaknesses. First, and lowest level, direct systems translate word-by-word with some possible local word reordering. Due to the low level or almost complete lack of analysis, these systems typically perform fairly poorly, especially if the source and target language differ significantly in terms of structure. Next, transfer-based systems analyze the syntax and or the semantics of the source and target languages and attempt to transfer a structure from the source language to an equivalent structure on the target side. Naturally, since the level of analysis is deeper, a transfer-based system typically outperforms a direct system. However, such systems still have trouble handling idiomatic and other non-compositional phrases. Specialized rules are necessary to handle such constructions for both syntactic and semantic transfer systems (Transfer-Methods.pdf). Finally, Interlingua systems provide a language independent representation of language and the deepest level of analysis. Yet, a language independent representation of all language is difficult and very time consuming to produce. Also, how deep the analysis should be is an issue: if the representations are too shallow, meaning will be lost; if the representations are too deep, analysis and generation is too difficult (Interlingua-MT.pdf).

The ultimate goal is to design a system that is fully automatic, produces high quality output, and is general purpose so that it applies to any domain. Currently, any two out of three of these goals is possible, but no system can achieve all three. Corpus-based methods, such as statistical machine translation (SMT) and example-based machine translation (EBMT) are both fully automatic and general purpose; however, they do not always produce high quality output. Knowledge-Based machine translation (KBMT), on the other hand, produces high quality output and is fully automatic, but it usually works in limited domains. Finally, human-in-the-loop MT produces high quality output and is general purpose, but it is not fully automatic (Interlingua-MT.pdf).

1.1 Details

My background is primarily in Linguistics, and as such, I was immediately drawn to a transfer-based system. Prior to coming to Carnegie Mellon, I lived in Korea for over two years. Thus, I chose to implement a Korean to English MT system. The system would be based with a limited, hand-created lexicon, as well as applicable hand-crafted transfer rules.

1.2 Goals

The ultimate goal of this project was to implement a transfer-based system that worked and produced coherent, understandable translations. In addition, I hoped to apply methods and techniques learned in class and gain hands-on experience implementing a transfer-based system. I also hoped to learn more about Korean morphology and syntax furthering my study and appreciation of the language. Due to the

limited time constraints (we only had a semester to implement this), I chose to limit the domain. My plan was to translate two children's books from Korean to English. The hope was that the limited domain would reduce the size of the lexicon, thereby reducing the overall ambiguity (at least at the lexical level). Finally, and equally importantly, I was hoping that the system would be fun to design and create.

2. Korean Background

Before talking about the tools I used and components of the system, I want to provide some background on the Korean language and some of the unique problems it poses.

2.1 Morphology

Korean is an agglutinative language, and its morphology is like "beads on a string." Words are formed by stacking morphemes one after another in a mandated order. For example, 먹+으시+겠+습니+까 (eat+honorific+future+highest+interrogative) is made up of a verb stem and 4 morphemes as denoted in the parentheses. Moreover, Korean has particles that mark three cases: nominative, accusative, and topic. These particles attach directly to nouns, thus marking the case of the noun. The plural morpheme also attaches directly to a noun. Verbs pose even more of a challenge. They inflect for passive or causative, honorific (highest, high, middle, and low), tense, mood (declarative, interrogative, propositive, imperative, and exclamatory), and continuation (conjunctions attach directly to the verbs), but only mood is mandatory. This rich morphology and its agglutinative nature poses a segmentation problem, as in order to derive the full meaning of a highly inflected form, it is necessary to segment each morpheme and derive the full meaning from these morphemes (Lee 139-196)

2.2 Syntax

Foremost, Korean has a much different word order than English. English is a Subject Verb Object language, producing sentences such as "Fatima ate the poisoned apple." On the other hand, Korean is a Subject Object Verb language, producing sentences such as "Fatima poisoned apple ate." Also, because of the explicit case marking, word order is relatively free, and it is possible (although rare) to place the object before the subject. The relatively free word order means that quantifiers and numbers can come before or after nouns and thus "one apple" and "apple one" are both perfectly valid. Thus, transfer rules must account for this freedom of word order and provide rules to account for both pre-nominal and post-nominal modifiers. In addition, Korean is a head final language. As a result function words such as prepositions (actually postpositions) and conjunctions come after the noun. More interestingly, such words are clitics of sorts and attach directly to the noun. For instance, "to the zoo" would be "zoo-to" in Korean. Finally, like many other Asian languages, Korean has measure words, which are used when counting various types of objects. For example, Koreans use the measure word 채(chae) when counting houses and 벌 (beol) when counting clothes. Hence, "one house" is really '집 한 채 (jip han chae)' 'house one measure word' in Korean. This is an issue because such words must be translated as null because there is not a true translation for the measure word in English (thing comes closest, but something like 'apple one thing' sounds a bit odd) (Lee 197-222).

2.3 Relevant Issues

In addition to the above issues with morphology and syntax, Korean also poses numerous other challenges for an MT system. As with all other languages, ambiguity is a major problem when

translating from Korean to English. First, there is a large amount of polysemy in Korean, including one to many mappings. For example, 우리 (uri) translates as both 'we' and 'cage' and 가 (ka) means both 'go' and topic case. Equally problematic are words that have no true translation in English. Measure words are one such example. In addition to polysemy, Korean words can have the same form but have entirely different meanings based on context. For instance, 먹을 (food-ACC, eat-FUT, food-ADJ) is the noun food with an accusative particle attached, the verb eat with a future marker, and possibly food with a marker for adjectives attached. Ambiguity at the lexical level is pervasive and is a challenge because the transfer MT system must take this ambiguity into account at the lexical, syntactic, and semantic levels as well as during the transfer and generation stages.

Korean syntactic issues also need to be addressed. The language in general is a pro-drop language, and as such subject as well as object can be dropped if they are understood from context. An example of this is: '기다리고 있다 (kitariko issta) 'waiting,' where the subject has been dropped presumably because it is understood from context. Conjunction and negation are also interesting. As already mentioned, conjunctions attach to nouns and verbs. However, two possible orderings are possible: the conjunction can occur between the two nouns it conjoins N-CONJ N, or it can occur after both nouns N N-CONJ. Negation also takes on two forms. The first, or short form, is pre-verbal and does not attach to the verb. The second form, or long form, attaches post-verbally and requires an additional negation verb roughly translating as 'not.' Hence, it is necessary to create a transfer rule to take both possibilities for conjunction and negation into account (Lee 202-3).

Moreover, Korean verbs provide several challenges for a MT system. First, Korean has a large amount of conflation. This includes one verb with multiple meanings encoded (actually two verbs joined together in Korean by a connector) such as '바다먹다 (patameokta)' 'pick up and eat' and separate verbs with one translation in English such as '던져 주다 (teonjyeo juta)' literally 'throw give' but better translated as 'give.' Conflation must be handled either via transfer rules that remove 'throw' in the second example and produce 'give' based on the constraint that 'throw' precedes 'give.' However, this is quite cumbersome and error prone. Thus, a better solution would be to encode such multi-verb instances in the lexicon as single lexical entries. Light verbs are also pervasive in Korean but may not necessarily be a major problem. 주다 (juta) is both a main verb, meaning 'give' and a light verb in the construction '먹이를 주다 (meokileul juta),' literally meaning 'give food.' A better translation would be 'feed,' but 'give food' is not a bad translation and still produces the intended meaning of 'feeding' or 'giving food to animals.' This happens to be a compositional case, where a direct translation produces a result that is not too bad. However, if a phrase with a light verb is not compositional, it becomes necessary to encode said phrase in the lexicon or create a lexicalized transfer rule to handle the construction. Finally, Korean stative verbs lack an explicit copula, meaning that 'be tall' is actually just 'tall' in Korean. There are again two different ways of handling stative verbs. The first approach involves encoding verbs with an explicit 'be' in the lexicon. Another approach is to mark such verbs as 'stative' in the lexicon and create transfer rules to explicitly insert 'be' if the verb is stative (Lee 100-104).

3. The System

As already mentioned, I attempted to implement a transfer-based MT system for Korean to English. I based the system on the AVENUE XFER framework and created the lexicon and transfer rules by hand.

3.1 The Corpora

I translated two children's books from Korean to English. The hope was that the children's books would contain simpler syntax and less ambiguity than Korean from newspapers. This "simpler" language in turn would be easier to translate and hopefully produce better results. I chose Curious George Feeds the Animals as the development set. This book had 91 sentences, and the average sentence length was 10.04 words per sentence. At first glance, my hope that the story would have simple syntax did not appear to be validated, as the text contained relative clauses, embedded questions, gerund subjects, reported speech, and numerous different tenses. However, the sentence structure was still simpler and the average words much smaller than typical written Korean. In addition to the development set, I chose Curious George Goes Camping as the test set. This book had 95 sentences, but the average words per sentence was slightly less than the development set at 9.25. The test set had similar syntactic structures to the development set, and the domain was exactly the same for both, so translation quality should be approximately the same.

3.2 Tools

The main component of the system was the AVENUE XFER framework. This is a transfer-based MT system with select statistical features such as rule weighting implemented by Erik Peterson. The system is compact and efficient in that analysis, transfer, and generation come in one package. Analysis and generation occur synchronously from the bottom up. The system begins with lexical items and attempts to build constituents from the lexical items based on entries in the transfer rules. The system returns results on a continuum. The best case is that a sentence level rule or rules apply, and the system returns a full parse and generation. The worst case is that no rules apply that produce a valid parse. In this case, the system produces a direct word-by-word translation. The system is quite robust, and if a word is not present in the lexicon, the system merely skips the word, continues with the translation, and then outputs the skipped word in its original form (Peterson 47-48). In order to run, the AVENUE framework needs a lexicon in one file and transfer rules in a separate file. The system also requires a file with one source sentence per line. The source sentences can contain punctuation, and if the punctuation is not present in the lexicon, it is simply translated as an unknown lexical entry.

I also used the Elicitation Tool and typological elicitation corpus in this project. The typological corpus contains approximately 800 sentences in English and is designed to elicit specific phenomena such as whether and how a language marks number and gender. Sentences are built compositionally with shorter sentences coming first and longer, more complicated sentences near the end. The sentences are then translated by a native informant or advanced bilingual speaker and the words in the source sentence are mapped to words in the elicited target sentence (Stat-XFER-Mar07.pdf). Many to one alignments are possible as are words with no alignments. The end result is word alignment for 800 sentences demonstrating word alignment for various constructions such as modifiers and relative clauses. The tool also produces a high-quality lexicon using the word alignments from each sentence as guides.

4. Methodology

The first step in this project was typing up the books into an electronic format with one sentence per line so that the transfer engine could process the input. I typed up Curious George Feeds the Animals and Jaedy Kim graciously typed up Curious George Goes Camping. For the development set, I added subjects and objects by hand to try to combat the pro-drop issue mentioned above. I also segmented the text by hand, as a accurate off-the-shelf morphological analyzer was not readily available for Korean. The goal of this step was to gain a better understanding of the material the system would be

translating, and the segmenting helped ensure the lexicon had maximum coverage. However, the ultimate goal of the project was to create a system that was fully automatic. Obviously, segmenting by hand and inserting subjects and object did not meet this goal. Hence, for the test set, I wrote a perl script that automatically segmented the input based on function words and particles in the lexicon. The script had a list of exceptions so as not to segment words that did not need segmenting. An exception is 원숭이 (weonsungi) 'monkey.' Usually, 이 (i) is the topic particle, but in this case it is part of the word, so it should not be segmented. By the end of the script, I had 95 such exceptions. I then proceeded through the data word by word and segmented where necessary. The rules were set up in a particular order to ensure that the longest possible segment was segmented when there was overlap between different possibilities (구나 (kuna) 'surprise' and 나 (na) 'or' for example). I looked for case particles, conjunctions, verb endings, and prepositions and ended up segmenting 15 verb endings, 15 conjunctions, 10 prepositions, 5 copular verbs, 1 adverb, 1 negation particle, and 6 case particles. The program still over-segmented a bit, so the final step was a serious of replace rules to put back together constructions that were segmented unnecessarily. For example, s/을 때/을 때/g; re-attaches the accusative case particle to the previous word when the case particle is followed by 때 (tae) 'when,' as the form in the lexicon is a verb with the 을 (eul) marking. It makes sense to have a global re-write rule here because there are many such forms where 'tae' follows a verb with the 'eul' ending, so adding exceptions for each case would have been tedious and unnecessary when a global replace rule can capture them all.

4.1 Elicitation Tool

I then wanted to gain a better understanding of various phenomena, such as word-alignment, gender, number, and modifiers in Korean. To better achieve this, I used the Elicitation Tool implemented by Erik Peterson and created by Allison Alvarez and Lori Levin. As discussed in class, the typological elicitation corpus takes a definition of the set of the desired features and a multiplier algorithm generates a comprehensive set of feature structures. A natural language generator then generates natural language sentences from these feature structures (Stat-XFER-Mar07.pdf). As such, the corpus covers a wide range of possible features and is as comprehensive as possible. I translated 802 sentences from English to Korean and then hand-aligned the words in a given sentence. The word order of Korean is quite different from that of English, so there were many crossing lines and long distance alignments. Also, there were many one-to-many, many-to-one, and one or many to none alignments because many words do not translate into Korean. For example, Korean does not make use of determiners very often, so a determiner in English was often a null mapping in Korean. In addition to the structural and typological information I gained from aligning the sentences, I also gained a high-quality hand-aligned lexicon with approximately 400 words. However, very few of the words showed up in my development and test texts, so I decided not to use this lexicon.

4.2 Lexicon

I created the initial lexicon by creating a perl script to read the words in the development text word by word and creating a file with the word in the proper AVENUE form. A lexical entry includes the part of speech for the source side, the part of speech for the target side, the lexical form for the source side, the lexical entry for the target side, and any features for both the source and target sides. The initial lexicon for the development set had 543 entries and 15 features taking on 70 values. The final lexicon including the test set had 749 entries and 15 features, including 143 nouns, 249 verbs, 29 conjunctions, 88 adverbs, and 36 prepositions. Due to a lack of a morphological analyzer, most verbs occur in fully inflected forms in the lexicon, and I only split off a verb ending if it was both consistent and irregular

and did not have various different forms because of phonological rules or morphological rules.

4.2.1 Typical Lexicon Entries and Features

I chose common labels to represent typical parts of speech. For example, N represents noun, V represents verbs, and NEG represents negation. The labels are not important in the AVENUE system as long as they are consistent, but I decided on common labels for understandability and readability reasons.

Korean does not mark verbs for agreement, but agreement is still important when translating to English (although English has minimal agreement as well). I handled agreement with an agr feature that took on values of 1sg for first person singular, 2sg for second person singular, 3sg for 3rd person singular, and pl for plural. Luckily agreement was not much of an issue because most of the verbs were in the past tense. In addition, most Korean nouns do not mark definiteness due to the rare use of the definite article in Korean. However, definiteness is important so as not to create a noun phrase with two articles. If definiteness is not marked, it might be possible to insert two articles when building a NP in the case when Korean does insert an article. Without marking definiteness when the NP is created with an article and N, a later rule may apply that inserts an article into a NP with a noun of a certain type. To combat this problem, I have a definiteness feature with + for the definite article and - for the indefinite article. When an NP is created with an article, the definiteness feature is set to + or -, and then the subsequent rule to insert an article into a NP does not apply because definiteness is defined.

Similarly, number marked by num, which takes on sg and pl values, is important for noun verb agreement on the target side. I also mark the type of conjunction so as to be able to move structures to the proper place in the sentence in the transfer rules. For example, if the conjunction is of type 'sub,' such as if, the dependent and independent clauses must be switched. The conjtype feature enables this to happen. Moreover, I mark the semantic type of nouns and verbs and prepositions with the semtype feature. Semtype takes on 13 different values, and the main goal is to ensure the proper selection of a preposition given a certain verb. For instance, 'give' has the semtype 'recipient' as does the preposition 'to.' The hope is that only sentences with semantic agreement between the preposition and verb or preposition and noun will be created. This is necessary because many prepositions in Korean have the same form but quite different meanings. An example is '에게 (eke),' which can mean 'by' and 'to.' The lexical entry to try to combat this polysemy is as follows:

```
src POS tar POS src form tar form
PREP::PREP |: ["에게"] -> ["to"]
(
(x1::y1) ; alignment (x is for source side, y is for target side)
((x0 form) = 에게) ; source side lexical form
((y0 form) = to) ; target side lexical form
((y0 semtype) = recipient) ; semantic type to try to disambiguate
)
```

Other features and values follow in line with typical unification features and values such as noun type, ntype (gerund, demonstrative, wh-pronoun), adverb type, advtype (manner, continuous, intensifier, habit, time), and gender (M, F, and N). Demonstrative nouns are the typical 'this' and 'that' and are distinguished from the definite articles 'this' and 'that.' Intensifier adverbs, such as 'very' intensify

nouns or adjectives, whereas continuous adverbs such as 'however' come at the front of the sentence and mark that a sentence continues in essence.

```
N::N |: ["어떻게"] -> ["how"]
(
(x1::y1)
((x0 form) = 어떻게)
((y0 form) = how)
((y0 ntype) = pro-wh) ; noun type is a wh-pronoun
)
```

4.2.2 Special Lexical Entries and Features

Naturally, Korean is quite different from English, so some special lexical entries and features are necessary. I handled regular final verb endings with the ASP part of speech. In the case where endings were consistent enough and regular, I split the ending from the verb and added it as a lexical entry. ASP also includes some words marking tense in Korean. For example, the future tense is marked by a separate word in English, but this is not a full form and requires the presence of a main verb (much like English). To distinguish this from a main verb I marked the future tense with an ASP part of speech. Also, the infinitive attaches to the verb in Korean, but it is regular. I split the infinitival particle off from the verb and created a ASP lexical entry for this particle as well. I chose to distinguish non-final verb endings providing aspectual information from the final verb endings; although such endings could have been lumped into the ASP category as well. I wanted a lexicon that maximally distinguished verb endings where possible, so I felt this distinction was necessary. MOR parts of speech mark the infinitive (albeit with a different form than the ASP infinitival) as well as necessity. Moreover, I handled measure words with M, and mapped them to null because there is no translation in English.

```
ASP::ASP |: ["거야"] -> ["will"]
(
(x1::y1)
((x0 form) = 거야)
((x0 vtype) = vfinal) ; occurs in final position
((y0 form) = will)
((y0 tense) = future)
)

MOR::MOR |: ["러"] -> ["to"]
(
(x1::y1)
((x0 form) = 러)
((y0 form) = to)
((y0 tense) = inf)
)
```

As mentioned, Korean nouns do not often make use of the definite or indefinite articles. Common nouns in English sound odd without such articles, so I created two special noun types to handle nouns where articles should be inserted. The 'common' value for ntype marks a common noun in English that takes the definite article. On the other hand, the 'icommon' value marks nouns that usually take the indefinite article. This is far from perfect, and once a noun is labeled as 'common' or 'icommon' it is that type and that type only, so a sequence like "the man saw a bird. The bird was big" is not possible. However, most noun phrases sound much more natural with an article than without even if the article is not completely correct (indefinite versus definite for example). The final lexicon had 54 common nouns and 26 icommon nouns.

```
N::N |: ["원숭이"] -> ["monkey"]
(
```

```
(x1::y1)
((x0 form) = 원숭이)
((x0 ntype) = icommon)
((y0 form) = monkey)
((y0 num) = sg)
((y0 agr) = 3sg)
)
```

Korean verbs also mark continuity but do not have an explicit conjunction attached. To handle this issue, I have special vtypes 'need_and' and 'need_that.' If such a value is present, an explicit 'and' or 'that' is inserted in the transfer rules. This allows constructions such as "knew that he was ugly" and the conjunction of two VPs where the conjunction is not explicit in Korean. Another important value for vtype is 'stative.' This value marks stative verbs in Korean, where 'be' must be explicitly inserted in a transfer rule and inflected as necessary.

```
VS::ADJ |: ["귀여웠"] -> ["cute"]
(
(x1::y1)
((x0 form) = 귀여웠)
((x0 vtype) = stative)
((x0 tense) = past)
((y0 form) = cute)
)
```

```
V::V |: ["내밀어"] -> ["reached out"]
(
(x1::y1)
((x0 form) = 내밀어)
((x0 vtype) = need_and)
((y0 form) = "reached out")
((y0 root) = "reach out")
((y0 tense) = past)
)
```

Adverbs occurred frequently in the data and marking their position was key. The advtype feature with a value of 'front' marks that an adverb must occur at the front of the sentence. 'need_and' is also a possible value, meaning that the conjunction 'and' is not explicitly marked in Korean but necessary in English.

The feature act attempts to distinguish questions from declarative sentences from interrogative ones. Its sole value 'quest' marks a V or ASP as a question, and the construction is handled by the appropriate transfer rules. Finally, the feature 'flag' is a catch-all of sorts that attempts to capture necessary values that do not fall neatly into any categories. It is impossible to have a feature with 2 different values in a lexical entry, so this entry is necessary when marking 2 different values for the same feature. For example, I mark all verbs as vfinal if they occur in the final position. This is to distinguish them from non-final verbs, so as to be able to attach ASP or MOR correctly. Some verb final verbs also are progressive, but the vtype feature already has a vfinal value. Thus, I created the flag feature to be able to mark such verbs as progressive and final. flag also takes on a value of 'rel' for verbs that initialize relative clauses, and a transfer rule explicitly inserts 'that' if the verb is of type 'rel.'

```
V::V |: ["있었어요"] -> ["were"]
(
(x1::y1)
((x0 form) = 있었어요)
((x0 root) = 있다)
((x0 vtype) = vfinal)
((y0 form) = were)
((x0 flag) = prog)
)
```

```
((y0 root) = be)
((y0 tense) = past)
((y0 agr) = (*or* 2sg pl))
)V::V |: ["떨어진"] -> ["had broken"]
(
(x1::y1)
((x0 form) = 떨어진)
)
```

```
((x0 flag) = rel)
((y0 form) = "had broken")
)
```

Finally, some constructions translated better when kept as chunks rather than attempting to derive them compositionally. Such constructions are represented as single unit in the lexicon and marked with a PH part of speech for phrase.

Hence, the lexicon is highly detailed with numerous features and values to help reduce lexical ambiguity. Certain values also ensure the insertion of certain explicit words such as 'and' and 'the' that are not present in Korean but necessary in English.

4.3 Transfer Rules

Korean is a complex language with relatively free word, thus a large number of transfer rules was necessary to capture possible word orders and combinations. All in all there were 126 transfer rules, covering noun phrases, verb phrases, adverbial phrases, prepositional phrases adjectival phrases, and sentences. The final set of transfer rules had 36 rules to cover NPs, 44 rules to cover VPs, 35 rules to cover sentences, 3 rules to cover PPs, 6 rules to cover ADVPs, and 2 rules to cover ADJPs. The large number of sentential rules is largely to handle the wide variety of questions (wh-questions, adverbial questions, do-support) and possible insertion of ADVPs before and after sentences, conjoining two sentences, and inserting interjections, communicators, and vocatives before sentences.

The transfer rules build constituents from the bottom up and as such begin by combining lexical entries into simple phrases. These simple phrases denoted by X1, where X is either NP or VP, are then built into more complex phrases, typically consisting of two or more X1 phrases denoted by X1 or X2 if they involve quantifiers or numbers. This process continues until a phrasal level (NP, VP,...) constituent has been built.

In addition, it is possible to assign weights to rules to favor certain rules over other rules. A rule without a weight is given a score of 0 by the decoder, whereas a rule with a weight is given a non-zero score. Therefore, it is possible to favor a certain translation over another one by assigning heavier weights to more common rules. The engine is more likely to choose a translation with the heavier weights as its top choice. This is one of many methods of resolving ambiguity in the transfer rules. Each rule is marked with a unique ID such as VP, 22 to aid in debugging and assist in identifying which rules applied in the process. In the interest of space I will discuss only a few rules of the major types: NP, VP, and S.

4.3.1 Noun Phrases

Noun phrases, like all other phrases are built from the bottom up. The first rule attaches a case particle to a bare noun if one is present. This creates a NP1, denoting that the N is now part of a phrase but not a full-fledged NP yet. If a case particle is not present, the N remains a bare noun until a suitable rule is found, combining it with another NP. Sometimes a noun cannot combine with another NP, and if this happens, a NP2 is created after many rules have failed, thereby ensuring that such nouns can join with other phrases later. Rules pass features such as ntype from the source (X) side to the created constituent via pseudo-unification. The same exact process passes features such as agr and num on the target side. The phrase thus takes on the features of the lexical entries, and these features can then be

used as constraints in later rules to ensure that only grammatical and well-formed constituents are formed in the first place. For example, if a NP had an agr value of 3sg, it could not later combine with a VP that had an agr of 1sg at the sentential level. The final NP rules conjoin already formed NPs if a conjunction is present or if conjunction is possible.

Three NP rules are particularly important to my system. First, one NP rule takes NPs with undefined definiteness and the value 'common' for the ntype feature and inserts 'the.' A similar rule takes NPs with undefined definiteness and the value 'icommon' for the ntype and inserts 'a.'

```
{NP, 21}
;;TL: the _____ ; Kor doesn't normally use DET --> insert
NP::NP : [NP2] -> ["the" NP2]
(
  (x1::y2) ; align NP2 on Korean side with NP2 on English side
  (*score* 0.33) ; assign weight to this rule
  ((x1 ntype) =c common) ; constrain ntype to common
  (x0 = x1) ; unify features in NP2 with NP
  (y0 = x0) ; transfer features in NP to English NP
  ((y0 agr) = (y2 agr)) ; transfer value of NP2 agr to NP
  ((y0 ntype) = (y2 ntype)) ; transfer value of NP2 ntype to NP
  ((y0 num) = (y2 num)) ; transfer value of NP2 num to NP
  ((y2 definiteness) = *undefined*) ; ensure that English definiteness is undefined
  ((y0 definiteness) = +) ; assign + value to definiteness on English side
)
```

This rule takes an NP2 with undefined definiteness and a ntype of 'common' and inserts 'the.' I have given it a weight of 0.33 because I want to favor insertion of 'the' over a shorter analysis without 'the' whenever it is possible. ((x1 ntype) =c common) ensures that the NP2 on the Korean side is of the proper type, 'common.'

The next key rule creates a compound NP with the name of one of the main characters in the Curious George stories: the Man in the Yellow Hat. A direct Korean translation of this phrased would produce 'Yellow Hat Man,' which is not bad but not ideal. Thus, I created a specific transfer rule to handle just this case, insert 'in the' and produce the correct word order. The first 'the' is handled by the above rule. Naturally, I could have created a lexical entry, but it was just as easy to create a transfer rule to derive the name.

```
; create specific compound adjectival NP
{NP, 8}
;;TL: man in the yellow hat
NP1::NP1 : [NP1 N] -> [N "in the" NP1]
(
  (x1::y3) ; constituent 1 on the Korean side aligns with word 3 on the English side
  (x2::y1) ; constituent 2 aligns with constituent 1
  (*score* 0.5) ; this rule applies frequently, so favor it
  ((x1 form) =c 모자) ; 1st Korean N must have this form = hat
  ((x2 form) =c 아저씨) ; 2nd Korean NP must have this form = man
  ((x0 ntype) = (x2 ntype)) ; transfer ntype from N to NP1 on Korean side
  (x0 = x2) ; transfer features from N to NP1 on Korean side
```

```

(y0 = x0) ; transfer features from Korean side to English side
((y0 agr) = (y1 agr)) ; transfer agr from 1st N to NP1 on English side
((y0 num) = (y1 num)) ; transfer num from 1st N to NP1 on English side
)

```

Again, I added a weight to this rule to favor this rule over other rules that created NPs but produced something like 'yellow hat man.'

The final key NP rule deleted certain Korean Ns to ensure better sounding translations. For instance, it is possible to say 'zookeeper' and 'zookeeper man' in Korean, whereas only 'zookeeper' sounds natural in English. Hence, I have a rule that deletes a N but only if it is of the form 'man.'

```

; create compound NP where 2nd NP unnecessary in Korean & 1st N case undefined
{NP, 5}
;;TL: zookeeper man
NP1::NP1 : [N NP1] -> [N]
(
(x1::y1) ; align N with N...no alignment for NP1 so ignored
(*score* 0.36) ; this rule applies frequently, so favor it
((x2 form) = (*or* 아저씨 언니)) ; the form should be 'man' or 'woman'
((x0 case) = (x2 case)) ; transfer case
((x0 ntype) = (x1 ntype)) ; transfer ntype
(y0 = x0) ; transfer from Korean to English side
((y0 num) = (y1 num)) ; transfer num
((y0 agr) = (y1 agr)) ; transfer agr
)

```

This rule is also weighted because it occurs frequently. However, there is a possible conflict with 'the man in the yellow hat' rule. To try to resolve this issue, I weighted the this rule less heavily than the above rule, to favor it slightly less often than the above rule.

4.3.2 Verb Phrases

Like NPs, VPs are also built compositionally from the bottom up. The first step is to attach either an ASP or MOR to a verb assuming one is present. If ASP has a value 'quest' it is not attached at the VP level. Instead, it is attached at the sentential level, ensuring that questions are formed correctly. The VP rules then proceed to build VPs, beginning with basic phrases denoted by VP1 and ending with phrasal level VPs, denoted by VP. The rules look for NP complements, PP complements (namely for ditransitive verbs such as give), as well as ADVP adjuncts and PP adjuncts. The final VP rules conjoin VPs if a conjunction is present or it is marked that the VP needs an 'and.' As with NPs, constraints try to limit the number of possibilities and ensure that only well-formed and grammatical constituents are formed.

Three VP rules are also quite interesting. The first because it is used most frequently and the next two because they insert words on the English side producing better translations. First, the transitive VP rule was by far the most frequently used. The object occurs before the verb in Korean, so it was necessary to invert the order. Also it is important that the case of the NP not be nominative because this is reserved for subjects, and this rule attempts to create a transitive VP. The rule also transfers the object

from the Korean NP to the Korean VP and finally to the English VP.

```
; Transfer transitive verb constructions
{VP, 12}
;;TL: was eating food
VP::VP : [NP VP1] -> [VP1 NP]
(
  (x1::y2) ; alignment
  (x2::y1) ; alignment
  ((x2 type) = (*not* passive)) ; passive handled by different rule
  ((x1 case) = (*not* nom)) ; want V object...so can't be nominative case
  ((x0 obj) == x1) ; Korean VP obj is now 1st NP
  (x0 = x2)
  (y0 = x0) ; transfer features from Korean to English
  ((y0 obj) <= (x0 obj)) ; transfer Obj form Korean side to English side
  ((y0 agr) = (y1 agr)) ; pass agr for final check at sentential level
  ((y0 tense) = (y1 tense))
  (y0 = y1) ; pass features from VP1 to VP on English side
)
```

Next Korean stative verbs do not explicitly contain 'be,' and so 'be' must be inserted on the English side. The transfer rules have a plethora of different rules to account for the above fact. Certain rules handle insertion when negation is present before and after the stative verb, certain rules handle insertion of 'be' when an intensifying adverb such as 'very' is present, and certain rules just transfer constructions where the insertion of 'be' is necessary. Without morphological analysis, it is necessary to account for agreement. Thus many of the rules are of the same form but 'be' has different agreements and tenses.

```
{VP, 1629}
;;TL: were not good
VP::VP : [VS NEG] -> ["were" NEG ADJ]
(
  (x1::y3) ;alignment
  (x2::y2) ; alignment
  ((x2 tense) =c past) ; be past tense so Korean must mark NEG for past tense
  (x0 = x1)
  (y0 = x0)
  ((y0 agr) = (*or* 2sg pl)) ; agr for were is 2nd person singular or plural
  ((y0 tense) = past) ; tense for VP is now past
)
```

The rule creates a VP because stative verbs do not take arguments other than 'be' followed by the adjective.

The final VP rule that is intriguing is inserting an explicit 'and' when one is not present in Korean. Korean marks continuation on verbs, but the conjunction is not explicit, so inserting 'and' or other conjunctions becomes necessary. This rule only applies to VPs with the value 'need_and.'

```
; conjoin 2 VPs...CONJ not explicitly marked in Korean...so insert it
```

```
{VP, 9}
;;TL: slipped and entered
VP::VP : [VP VP] -> [VP CONJ VP]
(
  (x1::y1) ; alignment
  (x2::y3) ; alignment
  ((x1 vtype) =c need_and) ; vtype must be 'need_and'
  (x0 = x2) ; transfer features from 2nd VP to 0th VP
  (y0 = x0)
  ((y2 form) =c and) ; conjunction on English side must be 'and'
  ((y0 agr) = (y3 agr)) ; transfer pertinent features from 2nd VP on English side
  ((y0 tense) = (y3 tense))
)
```

4.3.3 Sentence Rules

Sentence rules combine previously constructed NPs and VPs and have constraints to ensure that only proper constituents combine. There are quite a few sentence level rule due to the relative free word order in Korean, and the need to explicitly insert 'do' and 'have' in certain types of questions. As with stative verbs, 'do' and 'have' take on many different tenses and forms depending on agreement. The number of such rules could have been significantly decreased with a morphological analyzer.

The rule to create declarative transitive sentences was used most frequently and has a constraint to ensure that the subject agrees with the verb as well as a constraint to ensure that the subject is the proper case. The subject is transferred to the target side. However, this is not strictly necessary, but it is nice to see such information in the final parse tree.

```
; Transfer declarative sentences
{S, 1}
;;TL: Someone saw the parrot
S::S : [NP VP] -> [NP VP]
(
  (x1::y1) ; alignment
  (x2::y2) ; alignment
  (*score* 0.25) ; most common S rule so favor
  ((x1 case) = *defined*) ; want case to be defined...usually nominative but can be topic
  ((x1 case) = (*not* acc)) ; don't want subject to be acc case
  ((x2 act) = (*not* quest)) ; other rules transfer questions
  ((x0 subj) == x1) ; Korean subj = NP
  ((x0 subj case) = (x1 case)) ; transfer case
  (x0 = x2)
  (y0 = x0)
  ((y0 subj) <= (x0 subj)) ; transfer subj to English side
  ((y0 agr) = (y2 agr)) ; transfer agr on English side
  ((y1 agr) = (y2 agr)) ; ensure that NP agr unifies with VP agr
)
```

I assigned a weight to this rule to favor this analysis over others in the case of a conflict. There is also

a constraint to ensure that this rule does not apply to questions, as other rules handle various types of questions.

Much as it was possible to insert a conjunction at the VP level, it is possible to do so at the sentential level as well. The VP rule applies if there is no explicit subject, such as 'slipped and fell', so the conjunction must take place at a lower level. The sentential level rule applies when there are two complete S, and one of the verbs has a 'need_and' value. This rule actually applied much more frequently than the VP level rule because I explicitly inserted subjects in the development set. The engine favored maximal length constructions and thus built S level constructions and conjoined these whenever possible rather than conjoining VPs.

```
; insert and in sentence with no explicit and
{S, 999}
;;TL: The zookeeper was throwing fish to the seal and a figure surfaced
S::S : [S S] -> [S CONJ S]
(
  (x1::y1) ; alignment
  (x2::y3) ; alignment
  ((x1 vtype) =c need_and) ; unfulfilled constraint...so applies at S level rather than VP
  (x0 = x2)
  (y0 = x0)
  ((y2 form) =c and) ; conjunction form must be 'and'
)
```

The engine first builds 'the zookeeper was throwing fish to the seal.' Then it builds 'a figure surfaced.' Finally, it conjoins the two sentences, as 'throwing' has a 'need_and' value on the Korean side. This value did not apply at the VP level because it was possible to create two sentences of maximal length and conjoin them at the highest level: the S level.

Questions are also handled at the sentential level. ASP with 'quest' values do not attach until the S level to ensure that proper word order is achieved with minimal fuss. Compound VPs can still be created, but the final question particle does not attach until the question rule applies.

```
;Transfer suggestion questions (should, etc)
{S, 11}
;;TL: should we go to see other animals?
S::S : [NP VP ASP] -> [ASP NP VP]
(
  (x1::y2) ; alignments
  (x2::y3)
  (x3::y1)
  ((x1 case) = (*not* acc)) ; don't want subj to be acc case
  ((x3 act) =c quest) ; ASP must have 'quest' value
  ((x0 subj) == x1) ; transfer NP to S subj
  ((x0 subj case) = (x1 case)) ; transfer case on Korean side
  ((x0 obj) = (x2 obj)) ; transfer obj
  ((x0 asp) == x3) ; S asp becomes ASP
  (x0 = x2) ; transfer features from VP to S
```

```

(y0 = x0) ; transfer features from Korean to English
((y2 agr) = (y3 agr)) ; ensure that NP agr and VP agr unify on English side
((y0 asp) <= (x0 asp))
((y0 subj) <= (x0 subj)) ; transfer subj
((y0 obj) <= (x0 obj)) ; transfer obj
)

```

As with other S rules, this question rule checks that the VP agreement unifies with the NP agreement on the target side. If the agreement is not the same, unification fails, and the constituent is not produced. The rule also moves the ASP to the its proper location the target side: the initial position.

Therefore, the transfer rules cover a wide variety of phenomena and do an adequate job of covering every possible construction in the source text for both the development and test sets. Of course, the 80-20 rule applies (a few rules are used 80% of the time and many rules are 20% of the time). This does not mean, however, that many rules are extraneous. Every rule applied at least once. More than likely, some rules can be collapsed into a single rule, but I leave that for future work.

5. Experiments

After I was content with the coverage of my transfer rules (based on looking at data from the development set as well as the 800 sentences from the Elicitation Tool), I set out to translate the 91 sentences of the development set. First, I obtained two reference translations to evaluate the output of the system. My fiancée, a native speaker, provided one of the reference translations, and I provided the other. Since, she is a native speaker, her translation was more natural, whereas mine was more literal and direct.

The AVENUE system requires a language model to assist in making decoding decisions. After some testing on the development set, I decided that the `xin_all.gpp.220M.3gram` language model produced the best results. This trigram language model was trained on 220 million words of newswire text. The domain was not a good match for what I was translating, but the hope was that the sentences in my corpus would not be overly complicated and the complex nature of newswire sentences would be more than enough to handle anything in my data. I used an init file that saved my settings and automatically ran the engine. In this file, I set a number of important parameters for the engine. First, I specified how wide the beam would be for the beam search during decoding. Ultimately, a beam width of 20 proved to be ample, as the engine rarely returned more than 20 translations for any given sentence. I also set the rule weight, the length weight, the fragmentation weight, and the length ratio in this file. The rule weight how heavily the decoder should weigh the aggregate weights of all the rules in the parse, and a rule with no score had a weight of zero. The weights of the rules used in the translation are summed, and the higher the number, the more heavily favored a hypothesis is. The length weight is a penalty for a hypothesis being too long or too short. The length weight goes hand in hand with the length ratio: a ratio of the length of the average length of a source side example to the average length of a target side example (Peterson). I calculated this ratio using the sentences from the Elicitation tool as well as two gold standard translations from the development set. After separating morphemes and function words, the Korean sentences and the English ones were almost the same length, with the English sentences being slightly longer. The empirically calculated length ratio ended up being 0.98. Finally, the fragmentation weight relates to how much fragmentation is acceptable in the output. Basically determining how good an idea it is to use non-contiguous pieces in the lattice in the final hypothesis. The weight is a penalty, penalizing a hypothesis that is too fragmented. I set the initial

weights for three features by hand and derived the length ratio from data. In order to get optimal results from the test set, I used a script from Erik Peterson. The script examined the output from the development set and the reference translations and set the optimal weights for the above features based on matching the output from the system to the correct or closest reference translation.

I was now ready to run the system. Typical run time was approximately 15 minutes each to translate all the sentences in the development and test sets. The engine output a lattice with the top twenty hypotheses for each sentence. The hypotheses were ranked based on an overall score. The score came from the confidence of the language model, and the four weights described above. By assigning high weights to certain rules, it was possible to favor one hypothesis over another, even if the language model ranked it slightly lower than other hypotheses. As mentioned above, I weighted the insert determiner rule quite heavily. As a result, hypotheses with explicitly inserted determiners were usually near the top of the n-best list.

5.1 Development Set Results

The development set provided a good test of the coverage of the transfer rules and the rules that needed refinement. I examined the output from the system numerous times and refined the rules until I was happy with the result. This process was quite time consuming, but it paid off in the end. After tweaking the rules for an extended period, all the sentences in the development set parsed, and the engine produced relatively high-quality translations. Out of 91 sentences, the correct translation (at least the one matching one of the reference translations) occurred first in the list of 20 hypotheses 72 times. Moreover, the best and most correct translation was the second best in the list an additional 10 times when the initial hypothesis was not the correct one. Also, if the correct hypothesis was not first or second, it occurred in the top five an additional five times. Three sentences did not produce correct translations in any of the 20 hypotheses. However, the quality of even these translations was still relatively good and did not prove to be a major detriment to the overall BLEU, METEOR, or NIST scores.

I wanted to determine the best possible score and worst possible score of my system. To accomplish this, I attempted to translated without using the transfer rules. In order to determine the best possible score for the system, I went through the output and hand-picked the best translation for each sentence. Results of the three experiments can be seen below:

	BLEU	METEOR	NIST
Lexicon Only	0.17	0.61	6.65
Lexicon & Grammar	0.74	0.91	9.2
Oracle	0.77	0.92	9.3

The BLEU score is quite poor when only the lexicon was used. This is largely due to the fact that Korean and English are quite divergent languages and lexical entries do not adequately capture the large amounts of re-ordering at the sentential level and sub-sentential level required to produce decent translations. The system using the lexicon and grammar rules produced decent results. METEOR reported an overall precision of 0.8967193, recall of 0.9201995 and Fmean of 0.9177963. Fragmentation was moderate at 0.25. Moreover, BLEU reports a score of 0.93 on unigrams, 0.85 on bigrams, and 0.79 on trigrams for cumulative n-gram scoring and 0.9253, 0.7865, 0.6866 for the three measures respectively for individual n-gram scoring. The high scores should not be too surprising, considering the system was hand-crafted and tuned to translate precisely this input. Moreover, I

segmented the file by hand and inserted subjects and objects. Naturally, doing so more than likely increased the scores.

The oracle experiment proved interesting as well. Looking at the table, it is apparent that the language model and decoder are selecting the best parse nearly every time. A slight 0.3 improvement would have been possible on the BLEU score had the hypotheses been hand-selected and modest 0.01 improvement on METEOR would have been possible. These results help demonstrate that although I set the length weight, fragmentation weight, and rule weight by hand, the values are close to the optimal values.

5.2 Test Set Results

Before I ran the test set, I used Erik Peterson's script to attempt to optimize the rule weight and fragmentation for my system. I ran the system with my original weights and compared it to the output of the system with the optimal weights. Overall, the optimal weights produced two more correct translations than the original weights hand-tuned by me. I thus decided to keep these weights and use the output obtained with the optimized weights to evaluate the test set.

As expected, results were not quite as good for the test set. This is largely because I used a perl script to segment the data, leading to good but not perfect segmentation. Additionally, I did not insert subjects or objects and left sentences in their original forms. Finally, since this was the test set, I could not tune the rules to the data as I had in the development set.

Despite this, the results for the test set were still quite good. Out of 95 sentences, the correct translation occurred first in the list of 20 hypotheses 46 times. Moreover, the best and most correct translation was the second best in the list an additional five times when the initial hypothesis was not the correct one. Also, if the correct hypothesis was not first or second, it occurred in the top five an additional eight times. Finally, the correct translation occurred in the top half the n-best list an additional three times. Unfortunately, 33 sentences did not have a correct translation in the n-best list. The overall quality of these translations was far from terrible and a large number occurred because of a lack of an explicit subject. As a result, the BLEU, METEOR, and NIST scores dropped but not catastrophically.

I ran the same three experiments I ran on the development set and looked at the results, which can be seen below:

	BLEU	METEOR	NIST
Lexicon Only	0.15	0.62	6.56
Lexicon & Grammar	0.56	0.85	8.23
Oracle	0.6	0.86	8.54

The score for the lexicon only test is similar to those for the development set. I take this to mean that the data sets were similar in nature, which is not too surprising, since they were two books from the same series. The score for the system dropped on all three measures as compared to the development set. As mentioned, about a third of the sentences did not have a fully correct translation anywhere in the n-best list, and this definitely contributed to the decline in the scores. The METEOR scores were lower than the development set but not by excessively large amounts. Precision was 0.86, recall was 0.87, the Fmean was 0.87. On the other hand, fragmentation was almost 1.5 times greater on the test set: 0.40 compared to 0.25 for the development set. The higher fragmentation should not be surprising, considering that test set had significantly more sentences without complete analyses. The BLEU scores

for individual n-gram scoring and cumulative n-gram scoring were also slightly lower. The individual n-gram scores were 0.88, 0.61, and 0.47 for unigram, bigram, and trigram respectively; the cumulative n-gram scores were: 0.88, 0.73, and 0.63 for the same n-grams. The unigram scores are close to those of the development set, but the bigram and trigram scores dropped significantly. The drop for trigrams from 0.78 to 0.47 reflects a less "fluent" output and a much higher fragmentation and coincides with the relatively high fragmentation measure from METEOR.

The oracle experiment again confirmed again that the language model and decoder are choosing the correct hypothesis most of the time. An improvement of 0.03 was possible for the BLEU score on the development set, and a similar improvement of 0.04 would have been possible on the test set. Finally, the exact same improvement of 0.01 would have been possible on both the development and development sets. This either demonstrates that the system is making good choices, or that all hypotheses are equally decent and choosing any one of them would not have a significant impact. From the numbers above, it is my belief that the former is the correct analysis, and the language model and decoder did do a nice job of selecting the correct hypothesis.

6. Error Analysis

To further evaluate the overall performance of the system, I looked at the n-best output for both the development and test sets. I was then able to categorize the errors into size main types, which I will discuss below. The error types overlap and compound each other. For example, an incorrect lexical selection may have lead to the inability to form a compound NP, which in turn may have made an incorrect analysis exceedingly difficult.

6.1 Wrong Word Ordering

Word ordering was the most common error occurring a total of 16 times. The causes varied, but the overall reason seemed to be that no complete parse could be found. When this happened, the decoder pieced together chunks in the lattice. However, many re-ordering rules occurred at the sentential level and verbal level. If a complete analysis was not possible, such re-orderings did not take place, and the end result was fragmented and odd sounding. For example, this is the output from a sentence in the test set: GEORGE MARSHMALLOWS DELICIOUS WITH THE OWNER A TENT FORGOT TOO AND CHASED AFTER A DEER. The VP level did not fire and is in a mangled order. As a result, the sentence level rule did not fire, and forgot is left in its initial position in the Korean sentence and not in its correct position next to George. It is also interesting to note that both NPs 'delicious marshmallows' and 'tent owner' did not form correctly. Review and refinement of the adjectival NP rule and compound NP rule is necessary to resolve the issue. The subject is actually a long NP 'marshmallows delicious with the owner a tent.' Lexical selection is partially to blame as well. The engine chose 'with' over 'and,' and thus the correct analysis creating a conjoined NP with 'delicious marshmallows and tent owner' as the object of 'forgot' could not be created in the first place. The second VP formed correctly and is not an issue here.

Another common error in terms of word ordering was combining NPs to form a larger NP. Often times, the engine mistook an NP as the object of a V instead of a larger NP forming the subject of the sentence. For instance, incorrect analysis of the NP 'the tent owner' produced this comical result: A TENT STARTED CHASING THE OWNER GEORGE. 'owner' and 'George' joined to form the object of the V 'chasing,' and 'tent' was left as the subject. The correct analysis does occur, but it is only ranked fourth by the engine.

A couple of quick fixes would resolve this issue. First, adding a constraint that two NPs cannot conjoin, unless their cases are the same. In this example, 'George' has accusative case and 'owner' has topic case. In this case 'tent' does not have a case. The correct analysis would have 'tent' joining with 'owner' and receiving topic case. Then, 'tent owner' would become the subject of the sentence. Not allowing NPs with two defined and unequal cases combine would give a greater chance of 'tent' joining with 'owner' and producing a correct analysis. The other fix would be to add a constraint to the sentence rules that the NP in the subject position must have a defined case. Then, 'tent' would not be able to become the subject of 'chasing' unless it combined with 'owner' first.

6.2 Attachment of ADVP in the Incorrect Position

This type of error was the second most common and occurred 10 times. Typically the engine found complete parses but attached a modifying ADVP phrase in an incorrect position. Overall, the result is understandable and a decent translation, but it sounds a bit odd. Good examples include:

NOW I HAVE TO GO TO FIND THE PARROT THAT FLEW AWAY SO DON'T WRIGGLE AND HERE WAIT from the development set and GEORGE EVEN RAN FASTER from the test set.

In the first sentence, the system attached 'here' before the VP instead of after it. In the second example, attached one adverb 'faster' in the correct position but the other 'even' incorrectly. 'even' is a modifying adverb and modifies other adverbs and as such should have come before 'faster' and not the verb.

I made wide use of semantic features for adverbs in the lexicon but did not use these features consistently enough in the transfer rules. 'here' was of semtype 'loc' short for location. One of the transfer rules attaching ADVPs to VPs should have had a constraint that adverbs of semtype 'loc' occur after the verb and not before it. A similar solution is possible for 'even.' I labeled 'even' as an intensifying rule in the lexicon. Unfortunately, there is no rule combining an intensifying adverb and manner adverb such as 'faster.' This is because I did not see any such examples in the development set and thus did not create a rule to handle such a construction. Adding a transfer rule where an adverb of semtype 'intens' modifies another adverb of semtype 'manner' would be easy to add and would resolve the problem.

6.3 Wrong Preposition

Another common error was selection of the incorrect preposition based on context. This error type is a subtype of a lexical selection error and such errors occurred eight times. Korean has numerous prepositions that have the same form but various different meanings. The only way to distinguish these prepositions is from context. In an attempt to assist the language model in making a decision, I added a variety of semantic information to both verbs and the prepositions themselves. For example, I labeled the verb 'give' as semtype 'recipient' as well as the preposition 'to.' I then had constraints in the transfer rules that the semtype for a VP with a PP object or adjunct had to be the same. The constraints and semantic features worked okay most of the time, but for some reason the constraints were ignored in some cases. DO YOU WANT TO GO TOGETHER ON THE BIRD CAGE is one example from the development set. Here, despite the fact that 'go' has a 'motion' semtype and 'on' has a 'loc' semtype, the engine still successfully unified the two.

To resolve this issue, it would be necessary to look at the constraints and transfer rules and refine them as necessary. It may also be necessary to examine the transfer engine and determine why the constraint was violated. Finally, a better language model may produce 'to' with 'go' because 'go' and 'to' occur much more frequently close together than 'go' and 'on.'

6.4 Subject versus Object Selection

Usually this determination is straight forward, as nominative case marks subjects and accusative case marks objects. However, Korean also has topic case, and a topic can either be a subject or an object. The engine often placed a NP marked by topic case in the object position when in actuality it was the subject of the sentence, and this type of error occurred six times. An example from the development set is: THERE HAD BROKEN THE WIRE MESH. 'wire mesh' serves better in the subject position in this construction, although it is easy to see why it is the object of 'broken,' as break normally licenses an object. Thus, placing 'wire mesh' as the object is not a bad decision at all and an easy mistake to make.

This issue is particularly difficult to resolve. Adding additional constraints to the transfer rules and features to the lexicon does not resolve the problem. Topics can be both subjects and objects creating a constraint that limits NPs with topic case to one position or the other is counter-productive. One feasible solution is developing a large training corpus and training the language model on such occurrences. With enough data, it may be possible to statistically disambiguate whether the topic should be a subject or object. Using other NPs in the sentence (assuming they occur) is also a great help. The language model made mistakes when the topic was the sole NP in the sentence. If there was an additional NP with accusative case or nominative case, the engine could accurately determine whether the topic NP was a subject or object largely based on the case of the other NP.

6.5 Ambiguity

Words with the same form but different meanings were common in the data. As already mentioned, the topic particle '가' (ka) has the same form as the verb 'go.' In addition, '우리' (uri) means both 'cage' and 'we.' The engine struggled a bit with such lexical selection and the language model did not aid greatly in disambiguation. Quite often, a sentence that should have read 'George-Topic...' came out as "George goes..." On sentence in the development set was particularly troubling. The engine selected 'cage' over 'we' frequently and produced comical results such as GEORGE THE CAGE WILL HAVE TO COME AGAIN. Luckily such errors were not that common and only occurred six times overall: three in the development set and three in the test set. This means that the language model, lexical features, and constraints in the transfer rules did quite a good job of resolving ambiguity

There are a couple ways to resolve this issue. First, a better language model would help greatly. A good language model should be able to distinguish between 'we come' and 'cage come' and ultimately weight 'we' more heavily. The second solution would be to add lexical weights to the system. 'we' occurs much more frequently than 'cage,' and hence a hypothesis with 'we' should be favored over a similarly scoring hypothesis with 'we.'

6.6 Special Test Set Issues

The test set differed from the development set along one key dimension: I inserted subjects and object into the development set but did not do so for the test set. The lack of subjects and object directly affected the translation quality. Approximately 1/4 or 13 out of 49 of the errors in the test set occurred because of a lack of subject or object. The missing NP meant that there was no correct parse, and as such, word-ordering was often incorrect. For instance, the sentence A CHILD WATCHED POURING WATER TOO IN THE CAMPFIRE lacks an explicit subject. 'George' is the implied subject, and the 'child pouring water on the campfire' is the object of the verb 'watched.' Other than selecting the

incorrect preposition, the resulting translation is about the best possible translation without inserting an explicit subject. 'child' has topic case, meaning that it could easily be the subject, but this then leaves 'pouring' without a subject.

I did not have time to attempt a solution to this problem. Therefore, I kept sentences without subjects and objects in their original form. With more time, I would have tried adding features to insert dummy subjects and objects. The dummy subject would have been 'one' for verbs that take animate subjects and 'it' for verbs that take inanimate subjects, and the dummy object would have been 'it.' I would then add features to verbs stipulating whether they took animate subjects or inanimate subject or were transitive or intransitive. After that, I would create transfer rules to insert subject and objects with constraints that the verbs be of a certain type that would agree with the dummy subject and or object. The rules would be ordered last as a kind of last resort if all else failed. I believe that my feature and constraint based approach would have been successful given more time.

The last error was an oversight on my part. The development set did not have any existential verbs, so I did not add any transfer rules to handle such constructions. I should have thought more about this and realized this gaffe, as existential verbs are actually quite common in Korean. On the bright side, the output sounded "Yodaish" thanks to the inability of the engine to re-order the words due to a lack of a rule. YUCK SICKENINGLY WRETCHED SMELL THERE WAS. The output is understandable and ideal if you are small, green, and a puppet, but 'there was' is clearly in the incorrect position.

Handling such a problem would be quite simple. The easiest solution would be to have a feature existential with values + and =. There would be a special transfer rule to insert 'there' if the existential value of a verb was +. Inserting 'there' would produced a full parse and adding 'yuck' at the sentence level would then be possible. Not much re-ordering would be necessary, as 'sickeningly wretched smell' would be the object of 'was,' creating 'was a sickeningly wretched smell.'

Thus, the engine produced a wide variety of errors, but these errors could be grouped into errors of certain types. Most of the errors were not too major and would be solvable with more time to analyze and utilize proper features and constraints. Changing the language model might have also had a positive impact on disambiguating selection. However, the vast majority of the errors only hindered translation quality slightly, and as a result, the system produced quality output that received high BLEU, NIST, and METEOR score.

7. Conclusion and Future Work

Korean proved quite challenging to translate as expected. The large divergence from English in terms of syntax and word order, the polysemy, and the ambiguous topic marker all contributed to this challenge. Despite these issues, I think my system did quite a good job in producing quality output. The translations were far from perfect, but at the very least, even the worst ones were comprehensible. The robustness of the AVENUE engine and its ability to translate at the lexical level played a large role in the success. However, the hand-crafted lexicon and transfer rules played an equally important part. Translating within the domain of children's books also aided, as the sentences truly were shorter than newswire text and less ambiguous. Hence, thanks to the engine itself, the carefully hand-crafted lexicon and transfer rules, and the limited domain, the system met if not exceeded my expectations.

When I started the system, I had three goals. First, I wanted to design and create a system that was fully automatic. Second, I hoped that this system would produce good if not high-quality translations. Finally, I wanted to have fun. The first goal was not met for the development set but was met for the

test set. Running a script to pre-process the data is necessary, but the process is still fully automatic from start to finish. The second goal was also close to being met. My system did produce quality output, and whether it is high-quality or not depends on the task. This work could not be disseminated without some post-processing to fix sentences with word-order issues and the other six types of errors listed above. However, for an assimilation goal, the quality is quite good, and each output sentence is easily understandable. Finally, the third goal was met enthusiastically. I can honestly say that this is the most fun I have ever had on a school project. Not only did I get to learn more about Korean syntax and increase my vocabulary, but I also got to marvel at building the system from the ground up. There were frustrations at times, but overall, it was thoroughly enjoyable.

Where does that leave me? If I continued working on the system, I would focus on a few things. First, I would add a morphological analyzer. Doing this would decrease the size of the lexicon, as I would only have to store base forms and make the system more scalable to other domains. Next, I would refine and prune the transfer rules. 126 rules seems to be entirely too many, as a general purpose Chinese system has 48 rules, and the Hebrew system has even less. It would be interesting to use Ariadna's tool and see which rules could be pruned. I would also add more constraints to the current rules to resolve the attachment of ADVPs and ensure that subjects without case could not be attached. Finally, I would experiment with different language models and see if a more detailed model would give me better results. A more detailed language model might be able to resolve some of the preposition issues mentioned above, since it would employ larger n-grams and more than likely be trained on a larger amount of data. The ultimate goal would be to scale the system so that it could be used as a general purpose translation engine. This class project provided an excellent base should I choose to do that some time in the future.

Works Cited

- 11-731: Machine Translation. Basic Strategies. LTI: Spring, 2007. pp. 69-116.
- 11-731: Machine Translation. Experiments with Hindi-to-English Transfer-based MT System under a Miserly Data Scenario. LTI: Spring, 2007. pp. 1-20.
- Chung-hye Han, Na-Rae Han and Eon-Suk Ko. Development and Evaluation of a Korean Treebank and its Application to NLP. Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002).
- Chung-hye Han, Na-Rae han, Eon-Suk Ko. Bracketing Guidelines for Penn Korean TreeBank. Technical Report, IRCS-01-10 (2001).
- Chung-hye Han, Na-Rae Han. Part of Speech Tagging Guidelines for Penn Korean Treebank. Technical Report, IRCS-01-09 (2001).
- Chung-hye Han, Juntae Yoon, Nari Kim and Martha Palmer. A Feature-Based Lexicalized Tree Adjoining Grammar for Korean, Technical Report, IRCS-00-04 (2000)
- Chung-hye Han, Benoit Lavoie, Martha Palmer, Owen Rambow, Richard Kittredge, Tanya Korelsky, Nari Kim and Myunghee Kim. Handling Structural Divergences and Recovering Dropped Arguments in a Korean/English Machine Translation System. Proceedings of the Association for Machine Translation in the Americas '2000. Published in Lecture Notes in AI series of Springer-Verlag, © Springer-Verlag (2000).
- Chung-hye Han, Fei Xia, Martha Palmer, Joseph Rosenzweig. Capturing Language Specific Constraints on Lexical Selection with Feature-Based Lexicalized Tree-Adjoining Grammars. Proceedings of International Conference on Chinese Computing '96 (ICCC '96).
- Juntae Yoon, Chung-hye Han, Nari Kim and Mee-sook Kim. Customizing the XTAG system for efficient grammar development for Korean. Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Formalisms, TAG+ 5 (2000).
- Lavie, Alon, et al. METEOR. <http://www.cs.cmu.edu/~alavie/METEOR/> Carnegie Mellon University. December 18, 2006.
- Lee, Iksop and S. Robert Ramsey. The Korean Language. SUNY Press. 2000.
- Linguistic Data Consortium. University of Pennsylvania. <http://www ldc.upenn.edu/>. 1992-2007.
- Moonjin Media Co. 동물들에게 먹을 것을 주지 마세요. Translated 2002. Curious George Feeds the Animals. Houghton Mifflin. 1998.
- Moonjin Media Co. 캠핑 가는 것 세웠어요. Translated 2002. Curious George Goes Camping. Houghton Mifflin. 1998.
- MT Class Notes: Interlingua-MT.pdf. Teruko Mitamura.

MT Class Notes: Stat-XFER-Mar07.pdf. Alon Lavie.

MT Class Notes: Transfer-Methods.pdf. Alon Lavie.

NIST. mteval-v11b.pl. BLEU and NIST metrics.

<http://www.nist.gov/speech/tests/mt/resources/scoring.htm> NIST. May 20, 2004.

Peterson, Erik. Adapating a Transfer Engine for Rapid Development Machine Translation. Masters Research Paper. Georgetown. 2002.

Martha Palmer, Dania Egedi, Chunghye Han, Fei Xia, and Joseph Rosenzweig. Constraining Lexical Selection Across Languages Using TAGs. Tree Adjoining Grammars: Formal, Computational and Linguistic Aspects (TAG+ 3 Workshop Proceedings). Eds. Anne Abeille and Owen Rambow, CSLI, Stanford (2000).

Appendix

This section of the paper contains the complete lexicon with all the features and values. I feel the segment script is important, so I will include that as well. It also contains the transfer rules with some comments. For reference and edification, I have included a gloss of both the development and test texts. The gloss contains the original sentence, translations at the word level maintaining Korean word order and containing morphemes, and a full translation. Finally, I am including the output from the AVENUE system for both the development and test sets.

In the interest of space, I will just include the files with this write-up. The paper would be well over 1000 pages if I included everything listed above.