Pomona College
Department of Cognitive Science

Re-Ranking Text Simplification

Vivian Brown

April 25, 2011

Submitted as part of the senior exercise for the degree of
Bachelor of Arts in Cognitive Science

Professors Deborah Burke and David Kauchak, advisors

Abstract

Text simplification is the process of translating a complex text into a simpler text with more accessible sentence structure and vocabulary, while preserving the meaning of the original. Kauchak and Coster (2010) describe an approach to automatic text simplification utilizing a probabilistic phrase-based translation model. The goal of this project is to improve upon their result by re-ranking the n-best output of the baseline model. We identify features that pertain to grammaticality and simplicity in word-choice and structure. We train an SVM re-ranker based on these features, and use the trained model to generate a re-ranked n-best list. We output the new top ranked sentence in the re-ranked n-best list.

Contents

List of Tables

Chapter 1

Introduction

The goal of automatic text simplification is to transform a complex sentence into a simpler sentence while preserving the meaning and grammaticality of the original. Research from cognitive science indicates a large number of applications for text simplification. Deficits in processing complex sentences have been shown across a large number of groups, including young children (Smith and van Kleeck 1986), patients with amnesia (Shapiro, McNamara, Lazoni and Cermak 1992), older adults (Kemper 1987), and those diagnosed with Alzheimers disease (Small, Kemper and Lyons 1997). Automatic text simplification could make difficult-to-read documents newly accessible to these groups. Text simplification also has the potential to make technical or domain-specific articles accessible to the general public, by simplifying jargon and minimizing complex language.

One way to accomplish this is to view the problem as a translation problem. We can then leverage prior research in Machine Translation. In particular, we build on techniques for sentence compression (Knight and Marcu, 2002; Cohn and Lapata, 2007), which involves shortening a sentence by removing some subset of the words while preserving the original word order and meaning.

The application of that technique to text simplification is outlined by Kauchak and Coster (2010). Training data comes from Simple English Wikipedia (http://simple.wikipedia.org), as well as traditional English Wikipedia. Simple Wikipedia contains articles that use common English words and simple grammar, and attempt to cover the same content as traditional Wikipedia. To create the training corpus for Machine Translation, pairs of articles on the same topic are extracted from Simple English Wikipedia and traditional English Wikipedia. Within those articles, pairs of sentences

are identified where both sentences represent the same information, but one is simple and the other is complex. A statistical machine translation engine (www.statmt.org/moses/) is trained on these sentences pairs, with a few modifications to better accommodate simplification.

Like most statistical machine translation systems, the simplification engine produces a ranked list of the most likely simplifications for a given input sentence, called an n-best list. The highest ranked simplification is chosen as the best simplification. To test the system, we compare the best simplification against a manually simplified version from Simple Wikipedia. We compare the sentences using BLEU score (Papineni, Roukos, Ward, and Zhu, 2002), a measure of the similarity of two sentences based on word frequency and word order. BLEU score is designed to mimic human judgments of the similarity between two sentences, and is explained further in the methods section.

Additionally, we can ask how good the best sentence on our n-best list is, called an Oracle score. To compute the Oracle score, we input a complex sentence and get an n-best list. We compare every candidate sentence in the n-best list to the manually simplified version from Simple Wikipedia, and calculate a BLEU score for each. The Oracle score is the highest BLEU score on the list. This score represents the quality of the best translation in the n-best list. Prior research (Kauchak and Coster 2010) indicates that the Oracle score for the existing simplification engine is 6.2 percent higher than the BLEU score for the highest-ranked sentence, which represents a modest improvement in translation quality. If we could choose the best sentence on the n-best list, we could improve the BLEU score of our output by 6.2 percent.

The goal of this project is to reorder the output of the existing text simplification engine in order to select the best simplified sentence on the list. To do this, we identify features that indicate a good translation. These features could pertain to simplicity, grammaticality (whether the sentence in a grammatical English sentence), or preservation of meaning. We calculate these features for the members of the n-best list, and adjust the probability of each sentence accordingly in order to move better sentences to the top of the list. Table 1 gives an example of re-ranking a 3-best list. We expect to use a much larger (on the order of 10,000-best) list for our project.

Original sentence: "This sentence, being quite convoluted, thus has the ability to confuse those who read it."
Target simplification: "This sentence is confusing."

Before re-ranking

1. This sentence thus has the ability to confuse those who read.
2. This sentence is confusing.
3. This ability is quite convoluted.

We select sentence 1: "This sentence thus has the ability to confuse those who read."

After re-ranking

1. This sentence is confusing.
2. T his sentence thus has the ability to confuse those who read.
3. This ability is quite convoluted.

We select sentence 1: "This sentence is confusing", and improve our BLEU score.

Figure 1.1: Example of re-ranking using a 3-best list

Chapter 2

Literature Review

2.1 Models of Linguistic Complexity

When considering automatic text simplification, it is useful to ask: What makes a sentence simple or complex? Understanding linguistic complexity is important to our project for a number of reasons. First, it helps us recognize what kind of output we want, so we can judge whether our system is successful. Although we use simple sentences from Simple English Wikipedia to automatically test our output, it is useful to also have an understanding of the types of sentences we hope to see.

Identifying individual features that signal a complex or simple sentence is also important for re-ranking. We want to re-order a list of candidate simplifications so that the best ones are at the top. Knowing what makes a good simplification will let us move simpler sentences up the list. Finally, studies of linguistic complexity help to motivate our work by showing possible applications. Numerous studies demonstrate deficits in processing complex sentences across various groups including young children (Smith and van Kleeck, 1986), older adults (Kemper 1987), patients with amnesia (Shapiro, McNamara, Lazoni and Cermak, 1992), and those diagnosed with Alzheimers disease (Small, Kemper and Lyons, 1997). A system that simplifies complicated documents is useful because it would make difficult-to-understand documents accessible to more readers from a variety of backgrounds.

Linguistic complexity refers to the ease or difficulty with which a reader or listener is able to parse a sentence. It is closely tied to the notion of computational load how much work is the listener doing? How much information do they need to store in working memory? While it may be easy to identify sentences that seem very simple or complex, precise definitions

of linguistic complexity vary. A good theory of linguistic complexity will predict which sentence structures are difficult to comprehend, and which are easier. This section reviews a few theories of linguistic complexity, as well as some implications and empirical evidence for each.

One system for understanding linguistic complexity is the Dependency Locality Theory (DLT). Gibson (1998) describes DLT as a theory of human computational resources in sentence parsing. He divides the computational resources needed to parse a sentence into two components. The first involves structural integrations, connecting new words into the structure so far. The second involves holding the structure so far in working memory. Based on these two kinds of resource use, Gibson proposes that the cost of integrating two elements (for example, a pronoun and its antecedent) depends on the distance between them.

According to Dependency Locality Theory, center-embedded clauses should be a feature of complex sentences. Gibson provides the following sentences as evidence:

(a) The reporter disliked the editor.

(b) The reporter [s who the senator attacked] disliked the editor.

(c) The reporter [s who the senator [s who John met] attacked] disliked the editor.

Psycholinguistic data demonstrates that sentences like the ones above become increasingly complex as more center-embedded clauses are added. Miller and Isard (1964), for example, demonstrate this by asking subjects to memorize sentences with varying degrees of self-embedding. All sentences had the same length, but subjects had more difficulty recalling sentences with multiple embeddings. With increasing numbers of embedded clauses, the structures become unprocessable.

According to Dependency Locality Theory, sentences like (b) and (c) are difficult to process because of the computational resources needed during their processing. In (b), the listener must hold the incomplete dependency between the Noun Phrase reporter and the verb in working memory while processing the phrase who the senator attacked. Sentence (c) is even more difficult because the listener must store an incomplete dependency involving the reporter and one dependency involving the senator while processing the phrase who John met. Other forms of embedding such as left-branching tree structures are also shown to have similar effect on computational load (Cheung and Kemper 1992). Based on these findings, the presence and

structure of embeddings in parse trees may be a useful feature for re-ranking simplified sentences.

A second theory of linguistic complexity links computational load to idea density. Kintsch and Keenan (1973) propose that propositions are the basic units for understanding a text. A sentence that contains more propositions will be more difficult to comprehend. Propositions are defined as concepts within a sentence that can be true or false separately from others. For example, the sentence The old gray mare has a very large nose can be broken up into the following propositions (Brown, Snodgrass, Kemper and Herman, 2008):

```
(HAS, MARE, NOSE)
(OLD, MARE)
(GRAY, MARE)
(LARGE, NOSE)
(VERY, (LARGE, NOSE))
```

All of these propositions come from the main verb and its arguments, or from additional descriptive elements like adjectives and adverbs. Therefore, proposition density can be computed by counting the occurrence of certain parts of speech. Drawing on Kinstchs theory of proposition density, Brown et. al. propose flagging all conjunctions, numerals, prepositions, adjectives, adverbs, possessives, verbs, relatives and interrogatives as propositions. This measure is particularly useful because it can be computed automatically from a parsed sentence.

There is significant psycholinguistic evidence for the link between proposition density and sentence complexity, as measured by reading time and recall. Kintsch and Keenan (1973) measure the time it takes for subjects to read texts with varying proposition densities. They found that subjects needed an additional 1.5 seconds to process each proposition. High proposition densities also affect recall. Subjects found it more difficult to remember texts with more propositions, supporting the notion of propositions as a basic unit for language comprehension. This evidence supports the usefulness of proposition density for measuring the simplicity or complexity of a sentence because proposition density appears to be tied to linguistic complexity.

A third characteristic of complex language is the presence of uncommon words. Lexical complexity refers to that appear infrequently in written or spoken language. Rayner and Duffy (1986) show that lexical complexity increases a words processing time, indicating a greater strain on the brains computational resources. By tracking subjects eye movements, the authors

are able to show that subjects fixated for longer on infrequent words than frequent ones. Lexical ambiguity also contributed to processing time. Subjects spent longer looking at words with two equally likely meanings than on words with just one likely meaning.

For our project, lexical complexity can be measured by counting the frequency of words not present on a list of common words. Using a list of common words allows us to reduce the amount of information we need to store in order to estimate work frequency. Reducing lexical complexity is a promising application of automatic simplification because it does not require large changes to the structure of the sentence. Uncommon words and phrases can be swapped for simpler alternatives while maintaining the overall tree structure. Reducing lexical complexity will make reading easier for people with limited vocabularies, including young children and people learning English as a second language. It may also make documents about highly technical or specialized topics accessible to a larger group of readers by substituting commonly understood terminology for domain-specific jargon.

Finally, it is worth considering the instructions for simple writing provided by the Simple English Wikipedia project. These recommendations are available online as guidelines for editors contributing to Simple Wikipedia, and they inform the training data we use from Simple Wikipedia articles to train our machine translation system. While the rules listed are not motivated by specific cognitive studies, they do provide an additional model for simplified text that is supported by much of the data reviewed so far.

Writers for Simple Wikipedia are directed to use commonly occurring words to minimize lexical complexity. As a guideline, writers are encouraged to use words from Basic (British American Scientific International Commercial) English, a controlled language that includes a list of 850 basic words, and an expanded list of 1500 words. However, writers are directed to use slightly less common words when very common words sound strange. For instance, Basic English 850 requires drops from eyes in place of the less common word tears, but tears is recommended in that situation because the alternative sounds strange. Therefore it is worth noting that more common words are not always preferred.

Writer are furthermore encouraged to avoid ambiguous language. In the phrase hard work, for example, language learners who are not familiar with common English phrases could understand the word hard to mean solid or difficult to understand. Much work is recommended instead. This rule is consistent with the model of lexical complexity outlined above, which suggests that word ambiguity contributes to computational load.

Simple Wikipedia also provides guidelines for simple sentence structures. While more complex structures are permitted where necessary, the preferred sentence forms (in order) are:

1. Subject-Verb-DirectObject.
2. Subject-Verb-IndirectObject.
3. Subject-Verb-DirectObject-IndirectObject.
4. Subject-Verb-DirectObject-SubordinateClause.
5. Subject-Verb-DirectObject-IndirectObject-SubordinateClause.

Favoring these structures is supported by Distance Locality Theory. In all cases the verb immediately follows the subject, so the dependency between the two can be resolved with minimal strain on working memory. Similarly, the direct object (when present) immediately follows the verb. As predicted, Subject-Verb-Object sentences are preferred to sentences with embedded clauses, and center-embedded clauses are not preferred. Writers are encouraged to break up compound sentences linked by conjunctions into separate, shorter sentences. This is supported by the notion of proposition density, resulting in fewer propositions in each sentence. Lastly, it is noted that shorter sentences are not necessarily simpler. At times, for instance, it is necessary to add filler words that create natural pauses within the sentence so readers have a moment to process the sentence so far. Therefore, sentence length may not be a useful feature for identifying simple sentences.

2.2 Applications

Cognitive research also demonstrates the many possible applications of text simplification. Studies show that deficits in processing linguistic complexity are highly correlated with Alzheimers disease. Small, Kemper and Lyons (1997) study sentence comprehension in patients with Alzheimers disease compared to healthy older adults. They demonstrate significant declines in sentence comprehension among those with Alzheimers disease, and hypothesize that loss of working memory is an important factor for explaining those deficits. Patients have a particularly hard time comprehending embedded sentences, where the relativized noun phrase is follow by an object gap in the embedded clause. This is consistent with theories of linguistic complexity outlined above, which predict that processing dependencies over

large distances creates a larger burden on working memory. More generally, the study finds a robust correlation between scores on working memory tests and sentence comprehension measures. These findings indicate that text simplification can be used to make documents accessible to older adults by rephrasing sentences so that they can be parsed with less demand on working memory.

Linguistic complexity can also be used to predict the onset of Alzheimers disease. A well-known study by Snowdon et al. found that linguistic ability in early life is a strong predictor of Alzheimers and similar declines in cognitive function 58 years later. The authors analyzed autobiographies written by a group of nuns at a median age of 22. To measure linguistic ability in the women in early life, the authors examined proposition density and grammatical complexity. Approximately 58 years later, they assessed the cognitive function of the same group of women, now aged 79 to 96. They found a strong correlation between proposition density and grammatical complexity at a young age and cognitive declines later in life; low proposition density early in life was present in 90% of those with Alzheimers disease, and just 13% of those without Alzheimers disease. Clearly, linguistic complexity is an important tool for predicting Alzheimers disease later in life. Therefore, the ability to measure the complexity of a text could be useful for predicting and individuals risk of developing Alzheimers Disease.

The effects of linguistic complexity can also be seen in studies of aging. Kemper (1987) shows age-related changes in syntactic complexity by analyzing diaries kept by adults for 7 or more decades. A selection of sentences from each half-decade were analyzed for the presence of left- and right-branching embeddings and of coordinate and subordinate phrases and sentence fragments. Average sentence length was also considered. Kemper demonstrated a decline in syntactic complexity over the participants lifetime, as measured by a decrease in both left- and right-branching embedded constructions. Although this study measured language production, it is possible that similar effects may exist in language comprehension. These examples indicate just a few of the groups that might benefit from a successful text simplification engine. In each case, simplified text has the potential to open up a range of documents to groups that would otherwise find them too difficult to process.

2.3 Sentence Compression

Text simplification builds on automatic text summarization and compression. The foundational technique for sentence compression comes from

Knight and Marcu (2002). The authors describe two data-driven approaches to sentence-level summarization. In both cases the system is trained on a parallel corpus of full-text scholarly articles and abstract for those articles. The goal is to compress sentences by deleting some words, leaving the original word order and meaning intact. The system takes as inputs a sequence of words, and drops any subset of those words. Note, sentence compression only allows for deletions, which is simpler than simplification where words and clauses can be rearranged. However, techniques from sentence compression can be adapted for sentence simplification.

The first approach described by Knight and Marcu utilizes a probabilistic noisy channel model. Training data is used to develop a probabilistic context-free grammar (PCFG) that gives the likelihood for the parse trees of compressed sentences given the original sentence. Possible compressions are scored based on the PCFG and a language model consisting of word bi-grams. To calculate the probability of a given compression, we multiply two probabilities. The first, P(Compressed), describes the likelihood that the compressed sentence C that were considering is a valid, grammatical sentence. We calculate this probability using word bi-grams: if a sentence is grammatical, the bigrams in that sentence will appear frequently in other English sentences. The second probability, P(Long — Compressed), describes the relationship between the compression C and the original, longer sentence L. Specifically, P(Long — Compressed) represents the probability that original sentence L is an expanded version of the compressed sentence C. We calculate P(Long — Compressed) using the PCFG. To compute the overall probability of the compression, we multiply P(Long — Compressed) times P(Compressed). The compression with the highest probability is selected. The probabilities can be easily adjusted to favor compressed sentences of a particular length. Because of the way new trees are generated from the original sentence, this model only allows the deletion of subtrees. It does not allow tree reorganization.

Cohn and Lapata (2007) describe a second compression method using tree-to-tree transduction. Again, compressed sentences are generated by removing words or phrases without altering the word order of the original sentence. Grammar rules are generated from a parallel corpus of compressed and uncompressed sentences. The rules are pairs of tree fragments whose leaves are linked by word alignment. Individual words or larger tree fragments can be rewritten to produce simplified sentences. Support Vector Machines are used to score the grammar rules. This model also allows tree transformations, so it can alter the tree structure of the input sentence. Subtrees can be substituted or inserted to produce new structures. Experi-

mental results show significant improvements over Knight and Marcu.

2.4 Text Simplification

Our project seeks to re-rank the probabilistic output of an existing simplification system in order to choose the best output sentence. We plan to use a phrase-based statistical machine translation system (Och and Ney 2004). Words and phrases in simplified and unsimplified training data are aligned to produce translation rules. Building on the noisy channel model described above, the system effectively models local changes in word order, which is important for sentence simplification.

Kauchak and Coster (2010) describe the changes made to the phrase-based model above to better handle simplification. First, because we often wish to delete words or phrases, we allow word and phrases to be aligned to nothing (null alignment). Second, a post-processing step is added to address an error where a word is aligned to itself and another non-sequential word. Other work on simplification comes from Yatskar, Pang, Danescu-Niculescu-Mizil and Lee (2010). Instead of training on paired articles from Simple Wikipedia and traditional Wikipedia, the authors use revision history from simple Wikipedia. They use metadata about revisions to identify instances of sentence simplification in the revision history. They use these revisions to extract lexical simplifications - words or short phrases that can be swapped for a simpler alternative. Our project expands on this work by allowing lexical swapping as well as insertions, deletions and changes in sentence structure.

2.5 Re-ranking

The goal of this project is to re-rank the probabilistic output of an existing simplification engine in order to choose a better simplification. Re-ranking techniques have led to significant improvements in automatic parsing. Machine learning techniques are often applied to re-rank parses, including Boosting (Collins, 2000) and Support Vector Machines (Shen and Joshi, 2003). Another approach to parse re-ranking is presented by Huang (2002), who describes an algorithm for re-ranking a packed forest instead of an n-best list. Forest re-ranking allows re-ranking over a much larger set of possible parses. In general, re-ranking of probabilistic output has led to a 13.5 improvement in labeled recall/precision over the previous state of the art (Shen, Sarkar and Och 2004). Re-ranking has also been applied

to the problem of machine translation. Shen, Sarkar and Och (2004) use an existing machine translation system to output a ranked n-best list of candidate translations, and present two re-ranking algorithms. They use a perception-like machine learning algorithm differentiate between good and bad translations, and then reorder the n-best list based on the learned rankings. Finally, the top choice is selected from the improved ranking. Shen and Joshi (2005) propose a general framework for ranking and re-ranking in natural language processing. Perceptron algorithms are applied to both parse re-ranking and machine translation re-ranking.

A variety of useful features for re-ranking are identified by Och et al. (2004). The authors describe a methodology for comparing the usefulness of candidate features for re-ranking. Features they identify are based on part-of-speech tagging, alignment of the original parse tree to the translation parse tree, and alignment of the original parse tree to the translation sentence. These features are expanded by Chiang, Knight and Marcu (2009) to include an additional 11,001 features identified using the Margin Infused Relaxed Algorithm.

Identifying useful features for re-ranking simplified sentences involves selecting features that indicate a good simplification. Readability assessments attempt to measure the difficulty of understanding a text. Because readability is closely related to simplicity, readability assessment often involves selecting features that reflect the simplicity of a document. Feng, Jansche, Huenerfauth and Elhaded (2010) provide a comparison of features for automatic readability assessment. The authors classify documents based on grade levels ranging from 2 to 5, indicating the years of education required to understand a text. They compare a broad range of features. Some choices include frequency of uncommon words, entity-density (general nouns, and named entities such as names, locations and organizations) and lexical chains (entities connected by semantic relations and linked through the text). Entity density proves particularly useful in determining reading level.

Other readability tests rely on surface features of the document. The Flesch-Kincaid readability test, for examples, considers average words for sentence and average syllables per word (Kincaid 1975). The Dale-Chall formula considers average sentence length and percentage of uncommon words (i.e. words that dont appear on a list of 3000 frequently occurring words). Features directly related to automatic text simplification are identified by Napoles and Dredze (2010) in order to demonstrate the usefulness of Wikipedia and Simple Wikipedia as training data for simplification. They identify cognitively motivated features as well as measurements of a documents lexical, syntactic and surface features. The authors compare

paired articles on the same topic from Wikipedia and Simple Wikipedia. They also extract data from revision history and flags signaling unsimplified text. Some features examined include lexical choices (simples versus complex words), bi-gram part-of-speech tags, and sentence length. Several learning algorithms are used to test classification based on the features described. Using Support Vector Machines, the authors are able to attain 99.9% accuracy on document classification.

Chapter 3

Methods

3.1 Data Set

The data set for the re-ranking problem comes from a corpus of 137K aligned sentences between English Wikipedia and Simple English Wikipedia. To generate aligned sentence pairs, complete copies of English Wikipedia and Simple English Wikipedia were obtained in May 2010. The articles were paired by sentence title. Very short articles, disambiguation pages, and meta-pages about Wikipedia were removed from the data set. Next, articles were divided into paragraphs based on formatting information. Each paragraph in a Simple Wikipedia article was aligned to a paragraph in the corresponding Traditional Wikipedia article based on TF-IDF, cosine similarity (Sulton 1988). Finally, pairs of corresponding sentences between aligned paragraphs were identified based on order of appearance as well as TF-IDF, cosine similarity (Coster and Kauchak 2011).

A total of 10,588 aligned, content-filtered articles were extracted by the Simple English and Traditional English corpus. These articles yielded a total 137K sentence pairs. Of that 137K, 124K pairs were used to train the baseline translation model. Translation was done using a variant of Moses (cite), which uses a probabilistic phrase-based approach.

A set of 482 sentence pairs was used to train the re-ranking system described here. For each of those 482 pairs, the baseline system was used to generate a 1000-best list of candidate translations. And additional set of 450 sentence pairs was used to test the re-ranker. Again, the baseline system was used to generate a 1000-best list of candidate translations to be re-ranked.

3.1.1 Preprocessing

To identify the syntactic features described below, it was first necessary to compute a parse tree and part-of-speech tags for each candidate translation. To do this we used the Stanford Parser (Klein and Manning 2003). The Stanford Parser computes the grammatical structure of each input sentence using a Probabilistic Context-Free Grammar.

3.2 BLEU Score

In order to measure the quality of our translations, we used BLEU (Bilingual Evaluation Understudy), an algorithm for evaluating the quality of text that has been used extensively to evaluate machine translation from one natural language to another. Quality in this case refers to correlation between the machines output and a human translators: we prefer translations that most closely resemble a professional human translation. The BLEU metric was selected because it has been shown to correlate with human assessments of translation quality (Papineni et. al. 2002).

BLEU score is based on n-gram precision: In order to calculate BLEU score for a candidate translation, we sum the number of times each unique word in the candidate translation appears in the reference translation, and divide by the length of the candidate translation. This is repeated for a range of n-gram lengths. Shorter n-gram lengths generally measure retention of information, while longer n-gram lengths measure the fluency of the candidate translation. The n-gram precision scores are combined using a geometric mean. Since short sentences can receive artificially high scores, the combined n-gram precision is multiplied by a brevity penalty based on the length of the sentence. BLEU score is always a value between 0 and 1, with larger values representing more similar texts.

BLEU score can suffer from the fact that there might be more than one good translation for a text. A good translation may receive a bad score if it matches poorly with the one particular human translation to which it is being compared. Generally speaking, however, we believe good translations in our project will share a large number of words and phrases with the reference translation, and will therefore receive a higher BLEU score. The large size of the data set also helps to minimize the effect of noise due to individual translations.

3.3 N-Best Lists

We begin the re-ranking task with an n-best list output by the baseline simplification model. Because the baseline model probabilistically computes candidate translations, each candidate translation t_i it assigned a probability signifying how likely it is the input sentence L translates to t_i. The n-best list consists of the n most likely translations for a given input sentence, as judged by the baseline model. In this case, we are considering a 1000-best list for each input sentence.

Currently, the baseline model selects the highest-ranked candidate translation as its output sentence. We can use BLEU score to estimate how similar the output sentence is to a reference translation created by a human translator. We would like to maximize the BLEU score of our output. We can also compute an Oracle score for the 1000-best list. To do this, we compute the BLEU score for every sentence on the 1000-best list, and take the maximum of all those scores. This indicates the quality of the best translation on the 1000-best list. Over all the sentences in the test set, we found the Oracle score is 6.2 percent better than the BLEU score for the output sentence. This indicates that there is often a better translation somewhere in the 1000-best list that is not being outputted. The goal of this project is to find a higher-scoring translation than the current output in the 1000-best list and output that higher-scoring translation.

3.4 Re-Ranking

The output of our re-ranker is the top-ranked sentence in the 1000-best list. In order to output a better translation, we would like to re-order the sentence in the 1000-best list so that the best translation moves to the top. To do this, we compute a variety of features to describe each sentence (described in detail in section 4). We train a re-ranking algorithm to recognize features that indicate a good translation, and move sentences with those features to the top of the list.

To re-rank sentences we used SVMrank, a re-ranking algorithm that relies on Linear Support Vector Machines to rank re-rank input (T. Joachims 2006). We calculated 72 features (described below), based on output from the baseline model, words present in the sentence, and syntax. For each input sentence in the training set, we have a list of 1000 candidate translations. For each candidate translation, we assigned a desired rank (based on BLEU score), and a list of feature values that describe that sentence. All

candidate translations in the training set were used to train the model.

When training and testing the re-ranker, we varied the following parameters:

- *Training error vs. margin* Trade-off between training error and margin during training
- *Normalization of features* Features were either not normalized, or normalized to the range [0,1]
- *Significance threshold* We included either all features, just those features with t-score $\geq .1$, or just those features with t-score $\geq .3$

Next, we generated re-ranked output for each sentence in the test set. For each sentence in the test set, we had 1000 candidate translations. For each candidate translation, we outputted a list of feature values describing that sentence. We used the trained model described above to re-rank the 1000 candidate translations for each sentence in the test set. We selected the new top translation as the as the output for the system.

Chapter 4

Features

4.1 Translation Probabilities

We included in our feature set statistical output from the existing translation engine. The language model output computes the bigram probability for the candidate translation. To compute the bigram probability, the candidate translation is divided into all possible substrings of length 2. We compute the probability that any of those substrings occurs in an English language text. We multiply all the bigram probabilities to produce an estimate of the probability that the candidate translation is a grammatical English sentence. Bigram probabilities are a good measure of grammaticality because an ungrammatical sentence is likely to contain a large number of word pairs that occur infrequently in grammatical English.

We also included a word-level translation model, which represents the probability that a given word in the input sentence translates to some word in the candidate translation. These probabilities were calculated based on training data from English Wikipedia and Simple English Wikipedia.

4.2 Word-level Features

These features are based directly on the words in the candidate translation. They rely on word banks and syllable counts to predict lexical complexity. These features are intended to identify sentences where too much, or not enough, lexical simplification has occurred during the initial translation.

4.2.1 Word Bank

In order to predict lexical complexity, we used a set of words outlined by Basic English, a controlled language created by Charles Kay Ogden (Ogden 1930). The word set consisted of 1200 root words: the 850-word Basic English set, plus 350 international words. These roots were modified by 10 possible affixes such as ed and est, for a total of 5389 words. Ogdens Basic English was selected for two reasons. First, it is widely used in teaching English to language learners and therefore we believe it represents a reasonable set of basic words. Second, Basic English is suggested as a reference for writers and editors of Simple Wikipedia articles, so we hoped that words on the list would have a high correspondence to words in the target sentences.

We counted the number words in a candidate translation that appear on the Basic English word list, and normalized that count by the total number of words in the sentence in order to calculate estimates for percent simple vocabulary.

4.2.2 Syllable Counts

We also used syllable counts in words to estimate lexical complexity. We used a syllable estimator written by Gregory Fast (Fast accessed February 2011). We believe higher syllable counts will correspond to greater linguistic complexity. We computed average number of syllables per word, as well as percent complex words, the percent of words in the sentence with a syllable count of three or more.

4.2.3 Readability Tests

We calculated three standard readability measures for candidate translations. The first, the Gunning Fog Index (Gunning 1952), consists of the average number of words per sentence, plus the percent of words in the text that are complex. Complex words are defined as words with three or more syllables, not counting proper nouns, familiar jargon, or compound words. The result is scaled by .4 to produce an estimate of the number of years of formal education needed to understand a text on the first reading.

$$0.4(\frac{words}{sentence} + 100\frac{complexwords}{words})$$

Equation to calculate Gunning Fog Index

The second readability test we calculated is Flesch Reading Ease (Flesch 1948). Flesch Reading Ease measures the number of words per sentence, minus the average number of syllables per word. Those measures are scaled by constants and subtracted from 206.835, so that higher scores indicate material that is easier to read.

$$206.835 - 1.015\frac{totalwords}{totalsentences} - 84.6\frac{totalsyllables}{totalwords}$$

Equation to calculate Flesch Reading Ease

Finally, we calculated Flesch-Kincaid Reading Level (Kincaid 1975). While this essentially captures the same information as Flesch Reading Ease, we thought it might be interesting because, like the Fog Index, it returns an estimate of the number of years of formal education required to understand a text. The formula for Flesch-Kincaid Reading Level is given below.

$$0.39\frac{totalwords}{totalsentences} + 11.8\frac{totalsyllables}{totalwords} - 15.59$$

Equation to calculate Flesch-Kincaid Reading Level

4.2.4 Words Per Sentence

Finally, we also included as a feature the number of words in the candidate translation, which may correspond to the overall complexity of the sentence.

4.3 Shallow Syntactic Features

The following features were calculated based on the output of the Stanford Part-Of-Speech Tagger. They relate to the presence of various parts of speech, or combinations of parts of speech, in the sentence.

4.3.1 Part-of-Speech Counts

We calculated the total number of words tagged as nouns, verbs, adjectives, adverbs or conjunctions in the candidate translation, and normalized those counts by the length of the sentence. The proportions of various parts of speech may be descriptive of the complexity of the sentence. A high proportion of adjectives and adverbs, for example, may indicate large amounts of description and therefore greater complexity.

We also calculated the frequency of certain parts of speech occurring in pairs. This was motivated by an examination of the output of the translation engine. We noticed that common grammatical errors included pairs of quotes or commas with no content between them, so we checked for pairs of quotes (" "), and pairs of commas (, ,), occurring one after the other. We also checked for nouns followed immediately by commas, and prepositions followed immediately by nouns, in order to detect other grammatical errors.

4.3.2 Punctuation

A common error in the output of the translation error was a misplaced period (not located at the end of the sentence. We represented period position in two ways: as the index of the position of the period, and as a Boolean indicating whether the period is located at the end of the sentence or not.

In order to capture the error involving paired quotes and commas described above, we also calculated the number of words within quotations, and the number of words between commas.

4.4 Tree-Level Syntactic Features

The following features are calculated based on the parse tree for each sentence. The describe the syntax of the sentence.

4.4.1 General Tree Descriptors

We calculated the tree depth normalized by sentence length, and the number of nodes in the tree, also normalized by sentence length. We used these features to capture the overall shape and structure of the tree. We also calculated the largest branching factor of any node in the tree, which may indicate a particularly complex clause with a large number of components.

4.4.2 Top-Level Structure

To describe the top-level structure of the sentence, we looked at the children of the root node S. In particular, we calculated a feature indicating whether the first child of the root was a noun-phrase, and whether the second child of the root was a verb-phrase, in order to look for subject-verb structure. We also checked for the presence of a number of other parts of speech in various positions beneath the root. We calculated the total number of children of the root as another descriptor of the complexity of the sentence.

4.4.3 Non-Terminal Node Counts and Average Branching Factors

We calculated counts of several different types of internal nodes including noun phrases, verb phrases, and clauses, in order to describe the overall structure of the tree. Sentences with a large number of clauses, for example, may be more complex and contain more embeddings. We also calculated the average number of children for each type of internal node. Because of issues described above with noun-phrase position, we also calculated the average number of nodes whose first child is a noun-phrase, and the average number of prepositional phrases whose first child is a noun-phrase.

4.4.4 Clause Descriptors

Several features were added to describe the structure of individual clauses. This was motivated in part by issues we noticed with noun phrases moving away from the beginning of a clause. We calculated frequency of various parts of speech occurring as the first or second child of a clause, and normalized by the number of clauses in the sentence.

Chapter 5

Results

5.1 Significant Features

In order to select features for re-ranking, it was necessary to estimate the usefulness of candidate features. For each input in our tuning set we extracted both the highest-ranked sentence in the 1000-best list (the output of the baseline model) and the top BLEU-scoring sentence in the 1000-best list. We computed features for each pair of sentences. For each feature, we used a t-test to compute the statistical significance between the two sets. This allowed us to identify the ways in which high-scoring translations might differ from the translations we are currently choosing.

The features that performed best were related to the syntax of the candidate translation. In particular, the translation engine consistently made a few particular grammatical errors. Leading noun phrases were sometimes swapped out of their position at the beginning of a clause, so it we found high significance for features that described both the presence of a noun phrase at the beginning of the sentence-level clause (t = .950), and the presence of a noun phrase at the beginning of other clauses (t = .929).

There were also significant differences in the punctuations between the top-scoring sentences and the baseline output. The baseline output contained a large number of misplaced periods. Therefore we found significant scores for normalized period position (t = .994), or for a Boolean feature indicating whether or not a period occurs at the end of the sentence (t = 1.000). For similar reasons, we found significant scores related to grouping of quotation marks and commas.

Finally, we found significant differences in the translation probabilities between the baseline model output and the top-scoring sentence. This may be a result of comparing the highest-ranked output from the baseline model

to sentences that may have been assigned significantly lower probabilities.

Among the least significant features we measured were the word-level features, which pertained to the presence or absence of certain words in each sentence. We found that word use between the two sets was almost identical ($t < .050$), and average sentence length was identical between the two sets. This indicates that the words present in the baseline output sentence differ very little from the top-scoring sentence on the n-best list, and we are unlikely to improve the translation by altering word choice.

** indicates features normalized by input length.*
*** indicates features normalized by the total number of occurrences of the parents node type.*

Translation Features	T-Score	Baseline Mean	Baseline StDev	Top BLEU Mean	Top BLEU StDev
Total Probability	0.0031	-2383.9414	1322.8185	-2384.2747	1322.8029
Distortion Model Output 0	1.0000	-4.6784	1.7997	-5.9149	3.4860
Distortion Model Output 1	0.5003	-1.3218	0.8456	-1.2851	0.8416
Distortion Model Output 2	1.0000	-0.3754	1.4319	-1.4776	2.8354
Distortion Model Output 3	1.0000	-3.3745	2.1834	-4.4616	2.3112
Distortion Model Output 4	0.8107	-1.0593	0.7124	-0.9994	0.7012
Distortion Model Output 5	0.9989	-2.9657	3.1425	-3.6556	3.3953
Distortion Model Output 6	1.0000	-0.1306	0.5365	-1.9496	2.5339
Language Model Output	0.3945	-116.5978	72.1297	-118.9967	71.8601
Translation Model Output	0.0000	-23.8029	13.2088	-23.8029	13.2088

Table 5.1: Word Level Feature Scores for Baseline vs. Top BLEU-Scoring Training Sentences

Word Level Features	T-Score	Baseline Mean	Baseline StDev	Top BLEU Mean	Top BLEU StDev
Word Count	0.0000	23.8029	13.2088	23.8029	13.2088
Flesch Reading Ease	0.0226	39.3681	24.9205	39.4137	24.9209
Gunning Fog Index	0.0375	15.4565	5.9282	15.4385	5.9320
Flesch-Kincaid Reading Level	0.0440	12.4324	4.9265	12.4149	4.9267
Complex Words	0.0000	19.8666	11.2621	19.8666	11.2621
Syllables Per Word	0.0000	1.7543	0.2773	1.7543	0.2773
Basic English Words*	0.0000	0.4310	0.1197	0.4310	0.1197

Table 5.2: Word Level Feature Scores for Baseline vs. Top BLEU-Scoring Training Sentences

Surface Level Parse Features	T-Score	Baseline Mean	Baseline StDev	Top BLEU Mean	Top BLEU StDev
Starts With Determiner	0.4331	0.1245	0.3301	0.1369	0.3438
Nouns*	0.1392	0.3311	0.1059	0.3323	0.1068
Verbs*	0.1865	0.1221	0.0588	0.1212	0.0587
Adjectives*	0.2251	0.0724	0.0610	0.0713	0.0607
Adverbs*	0.3393	0.0333	0.0457	0.0346	0.0462
Prepositions*	0.1138	0.1226	0.0616	0.1231	0.0615
Conjunctions*	0.1228	0.0297	0.0342	0.0294	0.0341
Words between Quotes	0.0186	0.3029	2.8122	0.2988	2.7166
Words between Quotes*	0.0306	0.0064	0.0536	0.0065	0.0541
Words between Commas	0.0658	3.3589	6.9966	3.3963	7.0295
Words between Commas*	0.1189	0.0943	0.1651	0.0959	0.1677
Period Position	0.1521	22.7801	13.2302	22.6162	13.2709
Period Position*	0.9948	0.9437	0.0350	0.9338	0.0689
Period Last	1.0000	0.9772	0.1493	0.8921	0.3102
Noun After Comma Count	0.1736	0.4191	1.4254	0.4398	1.5071
Prep. After Noun Count	0.2026	1.6432	1.5166	1.6183	1.4898
Comma After Comma Count	0.8209	0.0021	0.0455	0.0083	0.0907
Quote After Quote Count	0.4746	0.0124	0.1109	0.0083	0.0907

Table 5.3: Surface Level Parse Feature Scores for Baseline vs. Top BLEU-Scoring Training Sentences

Tree Level Parse Features	T-Score	Baseline Mean	Baseline StDev	Top BLEU Mean	Top BLEU StDev
Top Level: 1st Child VP	0.1887	0.0809	0.2727	0.0768	0.2662
Top Level: 1st Child AdjP	0.6822	0.0000	0.0000	0.0021	0.0455
Top Level: 1st Child NP	0.9499	0.6141	0.4868	0.5519	0.4973
Top Level: 1st Child PP	0.5711	0.1120	0.3154	0.1286	0.3348
Top Level: 1st Child S	0.6698	0.1452	0.3523	0.1680	0.3739
Top Level: 2nd Child VP	0.7536	0.5373	0.4986	0.5000	0.5000
Top Level: 2nd Child AdjP	0.6822	0.0000	0.0000	0.0021	0.0455
Top Level: 2nd Child NP	0.7203	0.0685	0.2525	0.0871	0.2820
Top Level: 2nd Child PP	0.4044	0.0373	0.1896	0.0311	0.1736
Top Level: Num Children	0.1605	3.4917	1.0589	3.5062	1.1617
Tree Depth	0.2942	10.4855	3.4166	10.5685	3.3995
Tree Depth*	0.3201	0.5097	0.1792	0.5145	0.1830
Largest Branching Factor	0.0202	4.7842	2.4855	4.7801	2.5970
Node Count	0.0570	66.8983	36.3983	67.0664	36.4187
Node Count*	0.7337	2.8263	0.1449	2.8369	0.1496
NP Count*	0.7273	0.3590	0.0980	0.3661	0.1039
Num Children of NPs**	0.0399	2.2759	0.6059	2.2782	0.8097
VP Count*	0.3427	0.1323	0.0741	0.1302	0.0735
Num Children of VPs**	0.7959	2.4620	0.6274	2.4107	0.6246
ADJP Count*	0.4873	0.0136	0.0344	0.0150	0.0360
Num Children of ADJPs**	0.3635	0.4684	0.9422	0.4972	0.9518
ADVP Count*	0.5383	0.0205	0.0348	0.0222	0.0364
Num Children of ADVPs**	0.4891	0.3980	0.5922	0.4232	0.5983
PPP Count*	0.1240	0.1245	0.0645	0.1252	0.0643
Num Children of PPs**	0.1731	1.8857	0.5724	1.8938	0.5703
S Count*	0.1342	0.1061	0.0639	0.1068	0.0650
Num Children of S**	0.1773	2.7508	0.8710	2.7377	0.9369
1st Child of S: VP**	0.1323	0.1635	0.2668	0.1607	0.2572
1st Child of S: ADJP**	0.4364	0.0010	0.0228	0.0021	0.0321
1st Child of S: NP**	0.9288	0.5599	0.3927	0.5141	0.3941
1st Child of S: PP**	0.6157	0.0840	0.2385	0.0981	0.2627
2nd Child of S: VP**	0.6211	0.5027	0.4027	0.4798	0.4019
2nd Child of S: ADJP**	0.5049	0.0048	0.0420	0.0032	0.0329
2nd Child of S: NP**	0.8205	0.0464	0.1693	0.0625	0.2014
2nd Child of S: PP**	0.4605	0.0161	0.1098	0.0121	0.0959
1st Child of any: NP**	0.2453	0.0535	0.0225	0.0540	0.0240
1st Child of PP: NP**	0.3618	0.0004	0.0026	0.0004	0.0024

Table 5.4: Tree Level Parse Scores for Baseline vs. Top BLEU-Scoring Training Sentences

5.2 Re-Ranking Results

To test the re-ranked output, we selected the new highest-ranked translation after re-ranking: the top sentence on the re-ranked 1000-best list. We computed a BLEU score for the top sentence, and compared the BLEU score to the output of the baseline model.

The BLEU score for the baseline model was 0.7739. For each combination of training and testing parameters, only two sentences from the baseline outputted were swapped for different sentences in the n-best list. In all cases, the BLEU score for the re-ranked output was 0.7743. Because so few sentences were swapped does not represent a significant increase in BLEU score. In the conclusion, we discuss some possible ways to improve this result. Possible solutions might include using more training and test examples, and experimenting with different parameters to the training algorithm in order to induce more swapping.

	BLEU Score
Baseline	0.7739
After Re-Ranking	0.7743

Table 5.5: Average BLEU Score for Baseline Model and Re-Ranked Output

5.3 Describing Simplicity

We were curious about the extent to which the features we selected described simple versus complex sentences. To test the ability of the features we selected to describe the Simple Wikipedia data, we compared the set of simple reference sentences in the test set (extracted from Simple Wikipedia) to the sentences in the test set extracted from traditional English Wikipedia. We computed features for each sentence, and used a t-score to describe the difference between the Simple Wikipedia sentences and the traditional Wikipedia sentences. Features related to translation probabilities are omitted because these sentences are not the output of the baseline translation engine they were extracted directly from the Simple and traditional Wikipedia Projects.

We found moderate significance ($t = .690$) for word count between the Simple and traditional Wikipedia sets. However we found high significance ($t > .990$) for the Flesch, Fog, and Flesch-Kincaid readability tests, which

also take into account average number of syllables or percents of words with 3 or more syllables. This indicates that sentences in the traditional set may use more words than the Simple set, but use significantly longer words, as judged by number of syllables. It is also indicated that readability tests are a reliable measure for distinguishing the complex and simple sentences in our data set.

There were also significant differences in the highest-level structure of the sentences, as indicated by the children of the highest-level clause. Simple sentences were significantly more likely to begin with an embedded clause (t = .906). This structure may represent an alternative to embedded clauses within simple sentences. This is consistent with theories of linguistic complexity, which predict that simple sentences will be less likely to contain center-embedded clauses (Gibson 1998). We also found that simple sentences are more likely to have a noun phrase as the second child of the highest-level clause (t = .978). This result was surprising because the Subject-Verb-Object form is preferred by the Simple Wikipedia project.

** indicates features normalized by input length.*
*** indicates features normalized by the total number of occurrences of the parents node type.*

Word Level Features	T-Score	Simple Mean	Simple StDev	Complex Mean	Complex StDev
Word Count	0.6904	22.5867	11.2687	23.3578	11.4587
Flesch Reading Ease	0.9947	41.9247	24.9569	37.2414	25.2263
Gunning Fog Index	0.9992	14.6870	5.4191	15.9071	5.3991
Flesch-Kincaid Reading Level	0.9993	11.7604	4.5385	12.7925	4.5718
Complex Words	0.9562	19.2123	11.2254	20.7378	11.4163
Syllables Per Word	0.9405	1.7393	0.2928	1.7763	0.2958
Basic English Words*	0.7434	0.4447	0.1228	0.4354	0.1233

Table 5.6: Word Level Feature Scores for Simple vs. Traditional Test Sentences

Surface Level Parse Features	T-Score	Simple Mean	Simple StDev	Complex Mean	Complex StDev
Starts With Determiner	0.1846	0.2378	0.4257	0.2444	0.4298
Nouns*	0.0654	0.3320	0.1095	0.3326	0.1067
Verbs*	0.6406	0.1215	0.0572	0.1180	0.0571
Adjectives*	0.1557	0.0707	0.0687	0.0715	0.0661
Adverbs*	0.5169	0.0279	0.0421	0.0299	0.0420
Prepositions*	0.6173	0.1163	0.0603	0.1198	0.0604
Conjunctions*	0.5092	0.0273	0.0329	0.0289	0.0345
Words between Quotes	0.0000	0.0911	1.1168	0.0911	1.1168
Words between Quotes*	0.0818	0.0032	0.0395	0.0030	0.0362
Words between Commas	0.5975	2.9511	5.8922	3.2844	6.0328
Words between Commas*	0.5562	0.0952	0.1679	0.1038	0.1692
Period Position	0.6929	21.5422	11.3096	22.3200	11.4974
Period Position*	0.5103	0.9416	0.0559	0.9442	0.0554
Period Last	0.2381	0.9867	0.1147	0.9889	0.1048
Noun After Comma Count	0.2744	0.3311	0.7684	0.3489	0.7487
Prep. After Noun Count	0.6814	1.4756	1.4144	1.5711	1.4547
Comma After Comma Count	NaN	0.0000	0.0000	0.0000	0.0000
Quote After Quote Count	NaN	0.0000	0.0000	0.0000	0.0000

Table 5.7: Surface Level Parse Feature Scores for Simple vs. Traditional Test Sentences

Tree Level Parse Features	T-Score	Simple Mean	Simple StDev	Complex Mean	Complex StDev
Top Level: 1st Child VP	0.0000	0.0022	0.0471	0.0022	0.0471
Top Level: 1st Child AdjP	NaN	0.0000	0.0000	0.0000	0.0000
Top Level: 1st Child NP	0.6781	0.6533	0.4759	0.6844	0.4647
Top Level: 1st Child PP	0.1598	0.1222	0.3275	0.1267	0.3326
Top Level: 1st Child S	0.9063	0.1933	0.3949	0.1511	0.3582
Top Level: 2nd Child VP	0.2169	0.6222	0.4848	0.6311	0.4825
Top Level: 2nd Child AdjP	NaN	0.0000	0.0000	0.0000	0.0000
Top Level: 2nd Child NP	0.9779	0.0844	0.2781	0.0467	0.2109
Top Level: 2nd Child PP	0.0000	0.0067	0.0814	0.0067	0.0814
Top Level: Num Children	0.1508	3.6756	1.0248	3.6889	1.0754
Tree Depth	0.2777	10.2422	3.4002	10.3222	3.3441
Tree Depth*	0.8136	0.5118	0.1815	0.4962	0.1720
Largest Branching Factor	0.7735	4.7200	1.6362	4.8511	1.6104
Node Count	0.6005	63.3244	31.5549	65.1067	31.8112
Node Count*	0.8712	2.8123	0.1594	2.7963	0.1558
NP Count*	0.5892	0.3527	0.0972	0.3475	0.0925
Num Children of NPs**	0.7247	2.3094	0.6142	2.3538	0.6053
VP Count*	0.6234	0.1335	0.0730	0.1292	0.0727
Num Children of VPs**	0.8856	2.4000	0.5576	2.4611	0.5989
ADJP Count*	0.6635	0.0109	0.0291	0.0092	0.0235
Num Children of ADJPs**	0.2955	0.3902	0.8589	0.3689	0.8227
ADVP Count*	0.5881	0.0162	0.0300	0.0179	0.0314
Num Children of ADVPs**	0.5180	0.3490	0.5902	0.3768	0.5933
PPP Count*	0.5502	0.1202	0.0613	0.1233	0.0609
Num Children of PPs**	0.6446	1.8523	0.6041	1.8880	0.5488
S Count*	0.9379	0.1077	0.0588	0.1005	0.0559
Num Children of S**	0.1324	2.8771	0.8028	2.8861	0.8254
1st Child of S: VP**	0.6293	0.1069	0.1831	0.1182	0.1927
1st Child of S: ADJP**	0.3441	0.0022	0.0333	0.0014	0.0241
1st Child of S: NP**	0.0653	0.6141	0.3656	0.6121	0.3814
1st Child of S: PP**	0.2349	0.0836	0.2478	0.0886	0.2514
2nd Child of S: VP**	0.3971	0.5867	0.3780	0.5733	0.3939
2nd Child of S: ADJP**	0.1154	0.0014	0.0198	0.0012	0.0169
2nd Child of S: NP**	0.4858	0.0437	0.1544	0.0369	0.1595
2nd Child of S: PP**	NaN	0.0000	0.0000	0.0000	0.0000
1st Child of any: NP**	0.5275	0.0523	0.0212	0.0513	0.0206
1st Child of PP: NP**	0.0000	0.0000	0.0004	0.0000	0.0004

Table 5.8: Tree Level Parse Scores for Simple vs. Traditional Test Sentences

Chapter 6

Conclusion

The goal of this project to improve the output of a baseline text simplification engine. First, sought to generate features to describe the differences between the baseline translation, and more desirable translation in the n-best list. Second, we hoped to use those features to train a re-ranking algorithm to reorder the n-best list and select a more desirable translation.

This project was successful in identifying features that describe the different between the output of the current baseline model and higher BLEU-scoring sentences on the n-best list. Significant features generally pertained to grammaticality of the sentence, particularly the position of noun phrases and punctuation. We also found the word-level features were not useful because sentences in the baseline output generally used similar words to higher-scoring sentences.

We did not find a significant improvement in average BLEU-score after re-ranking. This is likely because a very small percentage of top-ranked sentences were swapped during re-ranking. In the future, we might experiment with changing the parameters of the re-ranking algorithm in order to create more aggressive changes in ranking. We might also experiment with different re-ranking algorithm. Identifying more features that differentiate desirable translations from the baseline output might also improve the performance of the re-ranking.

We might also learn more about performance by using a larger set of training and test sentences. In particular, we were limited by the fact that we considered 1000 candidate translations for each input sentence. If we reduced the size of the n-best list, perhaps to 100, we could increase the number of input sentences without sacrificing training or test time. This is a particularly promising option because desirable translations were generally

found close to the top of the 1000-best list, so we would be likely to retain a large number of high-scoring translations.