Pomona College
Department of Computer Science

WikidSimple: A Data-Driven Text Simplifier Using Tree Transducers Trained on Wikipedia

Daniel Feblowitz

April 17, 2011

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

Professor David Kauchak, advisor

Abstract

Automated text simplification has the potential to make documents accessible to a broader readership than they were originally written for. In this paper, we introduce a data-driven, syntax-based text simplifier that models this task as an English-to-English translation problem. Based on a novel corpus of aligned sentences from English and Simple English Wikipedia, our system learns a probabilistic synchronous tree substitution grammar, which is used as a tree transducer to rewrite sentences as their most probable simplifications. Using BLEU score, we compare this system to a state-of-the-art abstractive text compression program, as well as three other phrase- and syntax-based text simplifiers. We find that our program outperforms all but the phrase-based system, and is cable of producing higher-scoring candidate outputs than any other approach tested.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Simplified versions of texts make information available to a broader audience. For instance, while the concepts presented in a research journal article might be accessible to readers without expertise in a given scientific field, the complex vocabulary and syntax of these articles can render them incomprehensible. Science articles in the popular media adapt the language of the original texts in a way that opens their content to many more readers.

Other simplified texts are used to tailor existing works to early readers and foreign-language students, introducing them to the content of the originals while allowing them to develop their reading skills [PO07]. In this spirit, the Wikimedia Foundation recently introduced Simple English Wikipedia, a site designed to render the information in Wikipedia in more accessible terms. According to Wikipedia's internal entry on the wiki, it serves "people with different needs, such as students, children, adults with learning difficulties and people who are trying to learn English. Other people may use the Simple English Wikipedia because simple language helps them to understand unfamiliar topics or complex ideas." [Ano11b]

Producing simplified texts currently requires the labor of human editors with an understanding of both the source material and how to express complex thoughts in a simple vocabulary. The articles in Simple English Wikipedia, for instance, are maintained by over 170,000 registered users [Ano11b]. Even with this large a community supporting it, Simple Wikipedia has less than 2% of the number of articles in the general English Wikipedia. [Ano11c] [Ano11a] However, the last few decades have seen editorial tasks similar to text simplification, like language translation and summarization, being automated with increasing success.

Data-driven methods from these tasks can be readily adapted to the

problem of text simplification. In this paper, we approach text simplification as a form of translation from English to a simplified subset of English. We infer the correspondence between these languages from a novel corpus of aligned sentences from English Wikipedia and Simple English Wikipedia. This corpus, the largest ever assembled for the text simplification task [CK11], contains 137,000 sentence pairs. A pair of these sentences can be viewed as two representations of the same content, transformed from one language to another through a set of edit operations including insertion and deletion of words, reordering, substitution, and word changes. Using a grammar formalism expressive enough to capture the full range of these operations, we train a generative text simplification system on this corpus.

Our approach begins from the parse tree of each sentence as given by a statistical parser [PK07]. We pair the nodes of aligned trees based on their word span and a word-level alignment of their respective sentences. From each node-aligned tree pair, we extract a set of synchronous tree substitution grammar (STSG) rules that, when used as a transducer, can rewrite the traditional English parse tree into the parse tree of the corresponding simple sentence. We combine these into a single probabilistic grammar with weights obtained by maximum likelihood estimation. We annotate this grammar in several ways to improve its specificity, and combine the different levels of annotation in a simple backoff model. We then generate the 10,000 most likely trees in each grammar and adjust their ranking based on language models and a length penalty to return a single best simplification.

Using BLEU [PPR$^+$02], a standard evaluation metric from machine translation, we test the performance of this system against several others. These include T3 [CL09], an abstractive text compression system using the same robust grammar formalism and discriminative learning, as well as a phrase-based system employing an approach that has been successful in machine translation and does not require syntactic information. [KHB$^+$07]

This paper is laid out as follows: Chapter 2 describes the development of methods for the text simplification task and the related problem of text compression. Chapter 3 describes the specific strategy implemented by our program and its theoretical basis. Chapter 4 covers the experiments we conducted to compare this system against several others, and conclusions are drawn from the results in Chapter 5.

Chapter 2

Background

2.1 Sentence Compression

The earliest approaches to automated text simplification were rule-based systems for assisting aphasic readers [CTAC00, CMC+98], generating summaries [Mah97, JM00], and preprocessing text for other NLP tasks [CS97]. Interest in text simplification accelerated following [KM02], which formalized the problem, proposed the first data-driven approaches, and introduced the corpus and evaluation metrics that would become standard in later experiments. Knight and Marcu define a restricted version of the simplification task called "sentence compression," which requires simplified sentences to be produced by deleting a subsequence of the original sentence's words. They seek compressions that are shorter than the original sentence, retain as much information as possible, and remain grammatical.

Knight and Marcu approach sentence compression as text-to-text generation problem similar to machine translation – it is analogous to translating sentences from English to the language of compressed English. Knight and Marcu's work and subsequent research have adopted techniques that have been successfully applied in machine translation. Among these are the noisy-channel model [BPPM93], the alignment template method for word alignment [ON04], synchronous grammars [Shi04, Chi06], and the use of discriminative learning [McD06, CL07].

Recent work by Cohn and Lapata [CL08] extends the definition of sentence compression beyond the deletion-only model, which they refer to as "extractive" sentence compression. Cohn and Lapata define "abstractive" sentence compression to allow reordering, substitution, and insertion in addition to deletion.

Our approach to text simplification is based on abstractive compression. It employs the same group of edit operations, and shares the requirement to preserve the grammaticality and information content of the input. However, it differs from compression in its goal – it aims to produce text that it easier to understand rather than just shorter. This can involve deleting complex words or syntactic structures, or replacing them with simpler content, even if it lengthens the sentence.

2.2 The Noisy Channel Model

In addition to defining the sentence compression task, Knight and Marcu develop two different systems to carry it out. Both of their approaches are data-driven and operate on the syntax trees of input sentences. The first of the two models trains a decision tree for an extended shift-reduce parser that rewrites its input into a compressed form. The other, which has provided the basis for a great deal of subsequent research, applies a noisy channel model to the sentence compression task.

Knight and Marcu offer the following generative story to illustrate this model: a news editor composes a document in short sentences and passes it to a writer to "flesh it out." The writer does so by adding a number of extra words to each sentence — the noise in the model's channel. The compressor's task is then to reconstruct the original short sentences given only the padded results.

This decomposes into three smaller problems. The first is to determine a source model (i.e., language model), which assigns every string s a probability $P(s)$ representing the likelihood that s is the original short sentence. The second is to provide a channel model, or translation model, which maps each string pair (s, t) to a probability $P(t|s)$ that s will be expanded to t during the "fleshing-out" process. The remaining task is that of decoding — that is, given a long string t as input, finding the short string s that maximizes $P(s|t)$. This is equivalent to maximizing $P(t|s) * P(s)$.

For their source model, Knight and Marcu train a probabilistic context-free grammar (PCFG) on the Penn Treebank and a bigram language model on the Wall Street Journal corpus. They calculate $P(s)$ by multiplying the probabilities of each bigram in s and each PCFG rule in the derivation of s.

Their channel model is a weighted synchronous context-free grammar (SCFG). SCFGs consist of paired CFG rules that are applied in parallel to aligned nonterminals, deriving a pair of recursively related parse trees [Chi06]. Though SCFGs cannot delete nodes, they can expand nonterminals

to the terminal symbol ϵ, effectively removing them from the resulting tree's *yield* — the string that can be read off of its leaves [Chi06]. Knight and Marcu estimate weights for their SCFG rules based on frequency counts from a parsed, aligned corpus. To align nodes between parse trees, they employ a simple method that matches a source and target subtree when they share the same root symbol and the source's children are a subsequence of the target's.

For decoding, Knight and Marcu store all possible compressions of the input sentence in a packed forest structure and extract the highest-scoring tree at each of several compression rates. This model has an inherent tendency to favor the shortest compression, since shorter compressions involve fewer rules and bigrams, and multiplying in each additional rule probability lowers the overall score. To counteract this, they normalize the score of each sentence by its word length before outputting a single best compression.

2.3 Refined Generative Models

In order to compensate for the scarcity of available training data, Knight and Marcu make the simplifying assumption that the probability of a string being a valid short sentence, $P(s)$, is equal to that string's probability as an English sentence in general. This allows them to train their source model on the full Penn Treebank, rather than only the short sentences from their training corpus. However, as Turner and Charniak [TC05] point out, this assumption causes the model to strongly favor outputting uncompressed sentences, and it is only by weighting in favor of short sentences that the system performs any compression at all. To counteract this effect, they train a syntactic language model, and use it to calculate the probability that adding the deleted subtrees back into the short sentence will recreate the original. They use this in place of the source model. Turner and Charniak also contribute unsupervised and semi-supervised training methods to cope with data sparsity, but they ultimately conclude that the noisy channel model *per se* is not suitable for this task due to its tendency to reproduce the original sentence.

Galley and McKeown [GM07] offer a number of improvements to Knight and Marcu's model. In the first place, they avoid the above problem by seeking to maximize $P(s|t)$ directly rather than breaking it down into $P(t|s) * P(s)$. This can be done using only a trained SCFG – the most likely compressed tree is the one derived by the highest-scoring set of rules that synchronously produce the uncompressed input tree. Sentence probabilities are

obtained by summing the probabilities for all trees with the same yield.

Galley and McKeown's modifications to the model also include methods for estimating more accurate rule probabilities. *Lexicalization* produces finer-grained rules by annotating each node with the underlying head word and/or its part of speech. *Head-driven Markovization* compensates for the sparsity introduced by flat Penn Treebank trees by making an n-order Markov assumption around the head of each rule. That is, rather than conditioning on the full right-hand side production, a Markovized rule conditions on only the parent node and the n children closest to the head. *Parent annotation* increases the specificity of rules by marking nodes with the category labels of one or more ancestors.

Optimal parameters for these adjustments are determined during an automatic evaluation stage using Simple String Accuracy (SSA). The authors find the best results using interpolated Witten-Bell smoothing, conditioning on one additional node of context both around the head and above the parent, and all possible lexicalization. However, the improvements from both Markovization and parent annotation are slight, and the authors do not compare against a model that includes all horizontal context, which would be the default without Markovization. Further, SSA is a similar metric to word-level accuracy, which Clarke and Lapata [CL06] later show not to correlate with human judgements of compression quality. It is therefore unclear whether these are actually the best possible settings [Pit10].

Galley and McKeown's model also includes a novel method for parse tree alignment, using an approximation algorithm to choose the alignment that minimizes edit distance between the trees. Despite using SCFGs, they are able to capture some tree adjunction operations by allowing nodes on the compressed side to copy themselves into pairs of adjacent nodes that can then be expanded using two separate CFG rules. These operations allow them to align 25% of the abstract sentences in Knight and Marcu's Ziff-Davis corpus, while Knight and Marcu themselves were able to align only 1.75%. With the parameters set to the values determined above, Galley and McKeown's system outperforms Knight and Marcu's in human judgements of grammaticality and importance, even while compressing at a higher rate.

2.4 Discriminative Models

McDonald [McD06] takes a substantially different approach. Rather than define a grammar from the training data, McDonald extracts features from training sentences and weights them using discriminative learning. These

features are based on sentences' constituent and dependency parses, part-of-speech tags, bigrams, and unigrams. Thus, a degree of tolerance for noisy parses is built in – if incorrect syntactic features are harming performance, the learning algorithm should decrease the weight assigned to them. McDonald chooses features that decompose over sequential words in the output, allowing a search for the optimal compression using dynamic programming. For learning, he employs the Margin Infused Relaxed Learning Algorithm (MIRA) with a loss function based on word-level accuracy, which, as previously noted, does not necessarily correlate with human judgements [CL06].

Nomoto's Generic Sentence Trimmer (GST) [Nom08] employs a similar setup but adds restrictions derived from a dependency parse to constrain the space of possible compressions. Using a heuristic based on the depth in the parse tree and the inverse document frequency (IDF) of deleted words, Nomoto conducts an N-best search over the allowable compressions. This is intended to find those compressions that preserve the most relevant parts of the sentence. This list is pruned using hard-coded, language-specific rules, and then converted into features for re-ranking by conditional random fields (CRFs).

Both of these methods are designed for deletion-only sentence compression and cannot be readily extended to the abstractive summarization or simplification tasks. McDonald's must optimize over all possible compressions, the number of which is exponential within the deletion model but infinite if insertions or substitutions are allowed. Nomoto's dependency paths restrict the size of this set, but limit possible compressions to a subset of the subsequences of the original sentence. Though GST achieves better performance than more complicated models, even when training on data that is not deletion-only [Nom09], it is not appropriate for our goal of full text simplification.

2.4.1 Tree Transducers for Non-Deletion Edits

The deletion model employed by Knight and Marcu, Turner and Charniak, McDonald, and Nomoto cannot realize the full range of operations required for summarization or simplification. This is supported by research from Marsi et al. [MKHD10], who constructed a corpus of Dutch-language subtitle/full text pairs, and found that within it, only 16.11% of subtitles were subsequences of their original sentences. Even some constituent deletions permitted by the deletion-only model require multi-level changes to tree structure that cannot be expressed by SCFGs. Galley and McKeown narrow this gap with their improved alignment methods and simulated tree adjunc-

tion. However, capturing the full set of edit operations requires the adoption of a more robust grammar formalism. Cohn and Lapata's "Tree Transducer Toolkit" (T3) [CL09] is based on one such formalism. It combines a large-margin discriminative approach like McDonald's with a synchronous tree substitution grammar (STSG) [Chi06], an expressive grammar that can be used to insert subtrees, change words, and both reorder and delete constituents.

Unlike CFGs, which expand nonterminal symbols into sequences of terminals and nonterminals, TSGs expand nonterminals into structured parse tree fragments. STSG derivations can then recursively continue from aligned nonterminals at the leaves of each fragment. The additional structure represented in these fragments allows STSGs to capture multi-level alignments, as well insertions and more complex substitutions than can be modeled using SCFGs.

Starting from a parsed, word-aligned corpus of sentence pairs, T3 learns a constituent alignment based on the alignment template method of Och and Ney [ON04]. Their method defines two nodes to be aligned if at least one word under each node is aligned to a word in the yield of the other and no word under either node is aligned to a word that is not under the other.

Based on these alignments, T3 uses a top-down algorithm to extract the minimal set of STSG rules describing each tree. These rules are generalized by replacing subtrees with variables, and the minimal set includes no rules that can be generalized further without violating the word alignment. At runtime, copy rules are derived from the input to cover structures that have not been seen in training. The rules induced over the entire corpus, along with these copy rules, constitute a grammar that generates the set of all possible compressions.

Each STSG rule is converted to a vector of features representing syntactic, lexical, and compression-specific information. These features are used to train a structured SVM, which attempts to separate the correct structure from the rest of the training set with a large margin. The SVM rescales its slack variables using a loss function based on a compression's Hamming distance from a gold standard sentence.

T3's decoding phase uses a dynamic program to find the highest-scoring derivation of the input tree. A derivation's score can be calculated by adding the scores of each rule involved and the score assigned to each n-gram in the yield by an n-gram language model. The score of a single rule is the sum of its features weighted by the vector learned in SVM training. Because the rules are synchronous, the highest-scoring set of rules that derives the input simultaneously derives the best compression. The decoding algorithm fills

a three-dimensional chart, with backpointers, containing the best-scoring partial derivation. Each cell $C[v, t, l]$ in the table contains the best derivation transducing a tree rooted at node v to a tree with root t and terminal n-gram context l. The algorithm iterates over the source tree in postorder, computing the set of possible partial derivations and storing the result in the chart. To limit the complexity of this search, Cohn & Lapata use a beam search and cube pruning [CL07].

This differs from McDonald's approach in several significant ways. First, McDonald's system does not rely on an acquired grammar to define the space of possible outputs. Instead, it limits itself to deletion only and searches over all subsequences of the input. T3, on the other hand, considers only outputs licensed by its grammar. In practice, this is a much greater set of possibilities, since this grammar is capable of reproducing any insertion or substitution extracted from the training corpus. T3 also differs from McDonald's system in its choice of features, learning algorithm, and loss function.

The source code for T3 is available online (`http://staffwww.dcs.shef.ac.uk/people/T.Cohn/t3/`), and several other researchers have published experimental results with it. Nomoto compares the dependency-path-based GST [Nom09] to T3 on a corpus of sentence summarizations from New York Times RSS headers and the corresponding articles. He finds that GST outperforms T3 in human ratings of intelligibility and representativeness, scales that correspond to Knight and Marcu's original measures of grammaticality and importance [KM02]. He also notes a tendency in T3 to introduce unrelated material from its training set into its compressions, and speculates that it is ill-suited to corpora not designed specifically for the deletion-only task.

Marsi et al [MKHD10] also test T3 on their Dutch subtitle corpus. Though they perform no formal evaluation, they report observing the same tendency to insert unrelated material. They also mention instances of T3 violating agreement constraints, dropping required arguments, and eliminating or incorrectly replacing function words. Further, they find that if T3 is not allowed to derive back-off rules from the test data and retrain its classifier at test time – a step they deem to be prohibitively expensive – it is only able to produce output for 7% of input sentences.

Chapter 3

Program Description

We model text simplification as tree-to-tree transduction based on an STSG acquired from a parsed parallel corpus. We implement modified versions of Cohn & Lapata's [CL09] tree alignment and grammar extraction algorithms in order to acquire a probabilistic STSG from our training corpus. This grammar is used to define a weighted finite tree-to-tree transducer to which each input tree is applied, giving a weighted regular tree grammar. We generate the 10,000 most likely trees in this grammar, and rerank them using two n-gram models and a length penalty. The yield of the top-scoring tree is output as the simplification of the input sentence.

This chapter describes the workings of the program in detail. Section 3.1 explains the STSG formalism and its use in tree transduction. Sections 3.2 and 3.3 discuss the algorithms used for tree alignment and grammar extraction. Section 3.5 covers the decoding and reranking stages.

3.1 Synchronous Grammars

3.1.1 SCFGs

Syntax-based sentence compression systems like those of Knight and Marcu [KM02] and Galley and McKeown [GM07] model the tree-to-tree rewriting problem using what is essentially a synchronous context-free grammar (SCFG). SCFGs resemble CFGs, but simultaneously generate pairs of recursively related strings rather than single strings [Chi06]. They can be used for synchronous parsing, but here we focus on their application to translation. Whereas a CFG may have a rule of the form

$$\mathrm{S} \rightarrow \mathrm{NP\ VP}$$

an SCFG rule takes the form

$$\mathrm{S} \rightarrow \langle \mathrm{NP}_1\ \mathrm{VP}_2\ ,\ \mathrm{NP}_1\ \mathrm{VP}_2 \rangle,$$

Where the subscripts on nonterminals represent alignment.

A CFG production replaces the nonterminal symbol on its left-hand side with the string of terminals and nonterminals on its right. Likewise, one SCFG production replaces a single nonterminal in the source string and a single nonterminal in the target string. These nonterminals must be aligned, and they must be the same symbol as the left hand side of the rule. Applying the rule expands the nonterminal in the source string to the sequence on the left of the rule, and the nonterminal in the target string to the sequence on the right of the rule. Thus, if the production

$$\mathrm{VP} \rightarrow \langle \mathrm{V}_1\ \mathrm{NP}_2\ ,\ \mathrm{NP}_2\ \mathrm{V}_1 \rangle$$

is applied to the string pair

$$\langle \mathrm{NP}_1\ \mathrm{VP}_2\ ,\ \mathrm{NP}_1\ \mathrm{VP}_2 \rangle,$$

it is expanded into

$$\langle \mathrm{NP}_1\ \mathrm{V}_3\ \mathrm{NP}_4\ ,\ \mathrm{NP}_1\ \mathrm{NP}_4\ \mathrm{V}_3 \rangle.$$

Here, variable indices are changed to avoid unintended capture. [Chi06]

SCFGs are only capable of reordering and relabeling nodes [Chi06]. For a grammar to be used for sentence compression or text simplification, it must also be able to perform deletions. This can be accomplished by the inclusion of an empty string symbol ϵ in the grammar's alphabet of terminals. Nodes are deleted by special productions that expand all preterminals underneath them to ϵ [Pit10].

SCFGs are not flexible enough to capture a number of the edit operations needed for text simplification. Since they are limited to reordering and relabeling nodes, SCFGs cannot insert new constituents. Also, since SCFG productions only involve a parent nonterminal and its immediate children, they cannot represent alignments that span more than a single level of tree structure. For instance, in the pair of aligned trees shown in Figure 3.1, the two VPs written in boldface are aligned to each other under our learned constituent alignment. The top-level S nodes are also aligned. However, since an SCFG production can only contain a parent node and its immediate children, and since the VP in the simple tree is a child of the S, but the VP in the traditional tree is a grandchild of the aligned S, there is no SCFG rule that can derive this structure with the correct alignment.

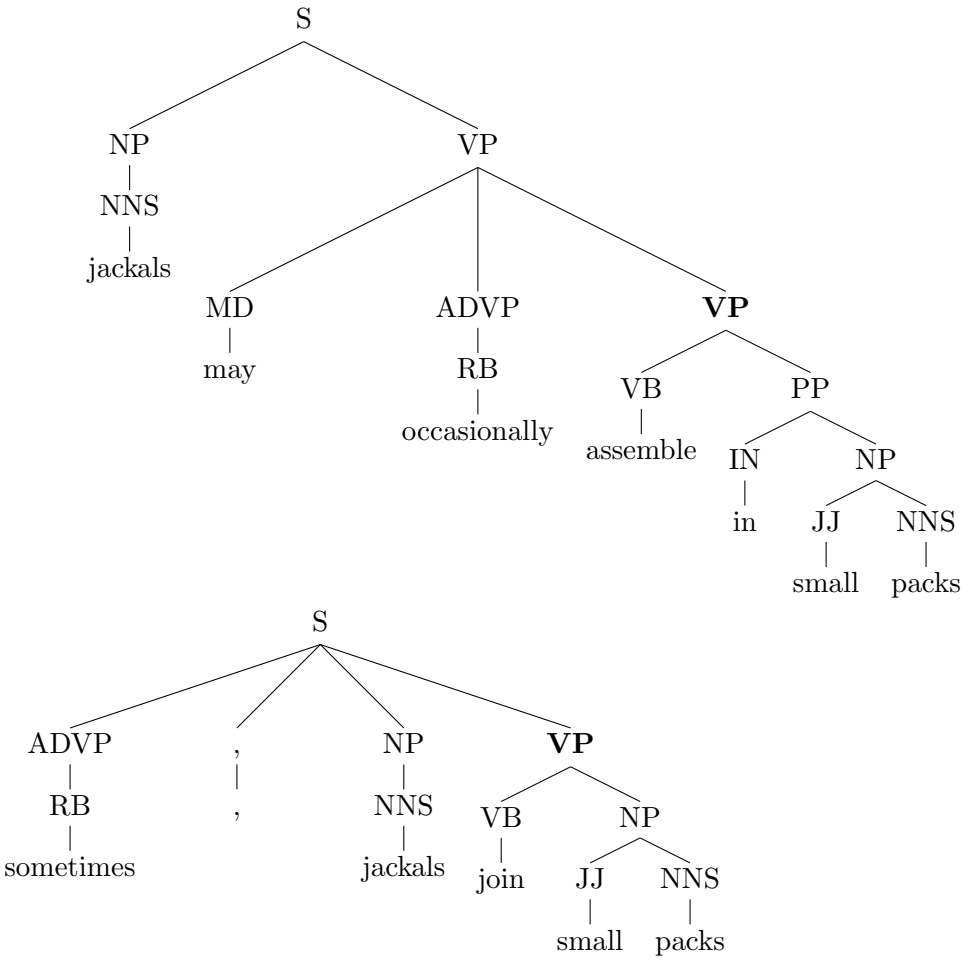

Figure 3.1: A pair of aligned trees from sentences in our Wikipedia corpus. Boldface VPs represent an alignment not expressible by an SCFG.

3.1.2 STSGs

Chiang [Chi06] discusses the strategy of eliminating this problem by flattening trees with multi-level alignments. However, this causes a loss of structural information. An alternative described by Chiang and implemented in a sentence compression setting by Cohn and Lapata is to use the more expressive Synchronous Tree Substitution Grammar (STSG) formalism [CL07] [Chi06].

Rather than replacing nonterminals with strings of terminals or nonterminals, as in SCFG productions, STSG productions replace nonterminals with tree fragments called *elementary trees*. The leaves of an elementary tree can be either terminals or aligned nonterminals. Thus, STSG productions allow the replacement of aligned nonterminals with arbitrarily deep tree structures. This allows them to derive many aligned trees inexpressible with SCFGs. For instance, the boldface VPs in Figure 3.1 can be synchronously derived with the following STSG rule:

$$\langle \text{S, S} \rangle \rightarrow$$

$$\langle \text{S(NP}_0 \text{ VP(MD(may) ADVP(RB(occasionally)) } \mathbf{VP}_1 \text{)), S(ADVP(RB(sometimes)) ,(,) NP}_0 \ \mathbf{VP}_1 \text{)} \rangle$$

In addition to enabling multi-level alignments, STSGs also allow new nodes to be inserted. This is accomplished by expanding a nonterminal higher up in the tree to a tree fragment that includes the new nodes. For instance, the above rule simultaneously inserts the MD headed by "may" and the ADVP headed by "occasionally" on the source side and the ADVP headed by "sometimes" on the target side.

Either form of synchronous grammar can be used as a tree transducer. Rather than expanding a pair of nonterminals into elementary trees, in a transduction setting the rules of the grammar read one elementary tree and rewrite it as the other tree of the pair. As in synchronous derivation, transduction then continues recursively on the nonterminal leaves. The resulting transducer can rewrite any source tree in the grammar as the target tree that would be synchronously derived [CL09].

3.2 Tree Alignment

Our program trains on a set of pairs of parsed, word-aligned sentences from a bitext. The word-level alignments for each sentence pair are lists of integer pairs giving the position in the sentence of aligned words. We implement Cohn & Lapata's [CL09] constituent alignment algorithm to infer a node-level correspondence from these indices. A pair of parse tree nodes are

aligned if all nodes under each are either aligned to a node under the other or unaligned. Cohn & Lapata formalize the definition of constituent alignment as a set C of all node pairs (v_t, v_s) such that there exists a pair of words (t, s) in the word-level alignment with t occurring in the yield of v_t and s occurring in the yield of v_s, and there is no pair (s, t) in the word alignment with only one of s or t in the yield of its respective node. Cohn and Lapata define constituent alignment in set notation, with A representing the word alignment and Y a yield operator that returns the indices of all words spanned by a given node, specifically:

$$C = \{(v_t, v_s) | (\exists (t, s) \in A \wedge t \in Y(v_t) \wedge s \in Y(v_s)) \wedge (\neg \exists (t, s) \in A \wedge t \in Y(v_t) \oplus s \in Y(v_s))\}$$

We add one additional stipulation to the original definition. We require that if a node b with and its parent a are both aligned to a node z and its parent y, the alignment retain only the pairs (a, y) and (b, z). This eliminates a common occurrence where too many associations are made between a pair of preterminal nodes and their children. Under the original definition, the following subtrees would contribute four node pairs to the alignment:

Specifically, the pairs (VB, VB), (assemble, join), (VB, join) and (VB, assemble) meet the criterion for inclusion. Our revised definition eliminates the latter two pairs. This reduces the size of the alignment, decreasing the number of cases which must be checked during grammar extraction while preserving the intuitive correspondence between similar nodes.

3.3 Grammar Extraction

Once a pair of input trees have been aligned, the next step is to extract a set of STSG rules that can simultaneously derive them. Used as a transducer, these rules will be able to completely rewrite the traditional tree into the simple tree.

Because STSG rules can have arbitrary depth, the rules we extract from aligned tree pairs can vary greatly in specificity. For instance, given the pair of aligned trees in Figure 3.1, it would be possible to extract a single STSG rule that derives both trees in their entirety in one step. This rule would not be particularly useful during transduction — in order to apply it, we would have to see the entire first tree as a subtree of an input, and as a result it

would be converted completely to the second tree. A rule producing smaller fragments, like this pair:

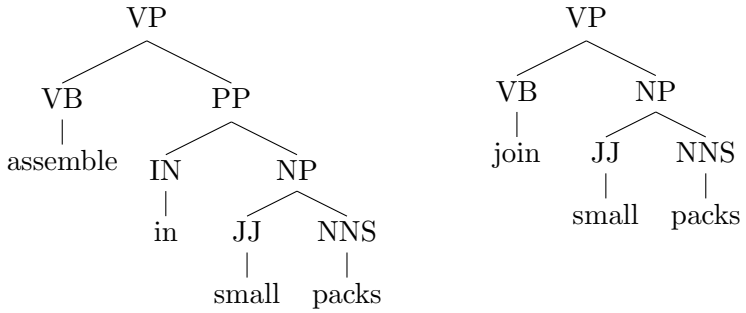

Figure 3.2: A pair of tree fragments

would be more useful in a transduction setting, as it could apply to a wider variety of inputs. A rule becomes even more general if aligned subtrees are replaced with linked variable nodes. For instance, substituting variable nodes for the subtrees (VB assemble) and (VB join), and both instances of (NP (JJ small) (NNS packs)) in the above tree fragments gives the following pair:

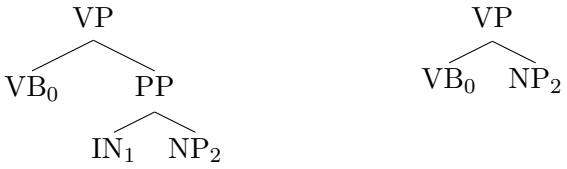

Figure 3.3: Tree fragments with variables in place of aligned subtrees.

By dividing a pair of trees into the smallest segments that respect the constituent alignment, and replacing all aligned subtrees with variable nodes, it is possible to extract what Cohn and Lapata [CL09] refer to as the *maximally general* rule set for the pair. This set of rules is capable of synchronously deriving the original pair, but is also applicable to the greatest breadth of possible input trees. The maximally general rule set extracted from the input tree pair in Figure 3.1 is given in Table 3.1.

In order to extract the maximally general rule set from an aligned pair of input trees, we implement the *extract* algorithm described in [CL09]. Pseudocode for *extract* is given in Figure 3.4. This algorithm takes as input the roots of two trees and the constituent alignment. It maintains a *frontier alignment*, a set of pairs of nodes that will be replaced with common variables

$\langle$S, S$\rangle \rightarrow$	$\langle$S(NP$_0$ VP(MD(may) ADVP(RB(occasionally))) VP$_1$),
	S(ADVP(RB(sometimes)) ,(,) NP$_0$ VP$_1$)$\rangle$
$\langle$PP, PP$\rangle \rightarrow$	$\langle$PP(IN$_0$ NP$_1$) , NP$_1\rangle$
$\langle$NP, NP$\rangle \rightarrow$	$\langle$NP(NNS$_0$) , NP(NNS$_0$)$\rangle$
$\langle$VP, VP$\rangle \rightarrow$	$\langle$VP(VB$_0$ PP$_1$) , VP(VB$_0$ PP$_1$)$\rangle$
$\langle$VB, VB$\rangle \rightarrow$	$\langle$VB(assemble) , VB(join)$\rangle$
$\langle$JJ, JJ$\rangle \rightarrow$	$\langle$JJ(small) , JJ(small)$\rangle$
$\langle$NNS, NNS$\rangle \rightarrow$	$\langle$NNS(packs) , NNS(packs)$\rangle$
$\langle$NNS, NNS$\rangle \rightarrow$	$\langle$NNS(jackals) , NNS(jackals)$\rangle$

Table 3.1: The maximally general STSTG rule set for the trees in Figure 3.1

in the rule produced by the current call. It proceeds through the nodes of the traditional tree in preorder. For each aligned node of the tree, it nondeterministically chooses the corresponding simple tree node that results in the maximally general rule set for that subtree.

Upon choosing a match for the current node, the algorithm makes a recursive call on the subtrees rooted at this pair. When this call returns, it adds the selected node pair to a frontier alignment for the current call and removes all tree structure underneath both nodes. If a node is the root of a completely unaligned subtree, all nodes underneath are removed without a recursive call, and that node is paired with ϵ in the frontier alignment, signifying that it will be deleted. Once iteration completes, the algorithm adds a rule that expands the two trees' roots to the undeleted remainder of each tree, with pairs of frontier-aligned nodes replaced with common variables. The top-level call returns the maximally general rule set.

The nondeterministic choice mentioned above is accomplished by means of a pair of mutually recursive methods. One method saves the state of both trees, and passes control to the other. This method iterates through the set of nodes aligned to the current node, choosing each in turn, and proceeding to recurse on the subtrees rooted at that pair. When this recursive call returns, the size of the extracted rule set is checked against the largest set seen so far. If the current choice of node did not result in a larger rule set, the changes are rolled back, and that choice is forgotten. Otherwise, the rule set and changes to the trees are retained. Ultimately, the algorithm acts upon only the choice that resulted in the largest set of rules.

Because of the computational expense involved in the nondeterministic choice of aligned nodes, we configured our program to reject any input for which rule extraction took longer than a predetermined time. For example,

extract(t, s, A, R): extracts minimal rule set from trees t and s with constituent alignment A into set R

$F \leftarrow \emptyset$ {initialize frontier alignment F to empty set}
for all nodes v_t in tree rooted at t, in preorder **do**
  if t is aligned to some node(s) under s **then**
    choose node v_s
    $R \leftarrow extract(v_t, v_s, A)$
    $F \leftarrow F \cup (v_t, v_s)$
    delete children of v_t
    delete children of v_s
  else if all nodes in subtree rooted at v_t are unaligned **then**
    $F \leftarrow F \cup (v_t, \epsilon)$
    delete children of v_t
  end if
end for
$R \leftarrow R \cup (\langle root(t), root(s) \rangle \rightarrow \langle t, s, F \rangle)$ {Add new rule to minimal rule set.}
return R

Figure 3.4: Pseudocode for the *extract* algorithm, adapted from [CL09]

setting this threshold at 2 minutes resulted in the abandonment of only 311 out of the 110,000 sentence pairs in the training corpus.

3.4 Grammar Augmentation

Providing more general rules can be both a help and a hindrance in transduction. Highly general rules allow the resulting transducer to handle more potential inputs, but can also result in unwanted transformations. For instance, applying the rule from Figure 3.3 to the the subtree VP(VB PP(IN NP)) transforms it to VP(VB NP). If this rule were applied to an unseen phrase that matches this structure, for instance "eat in the cafeteria", it could produce a malformed output like "eat the cafeteria".

This problem can be mitigated by adding some more specific rules back into the rule set, allowing the system to prefer transformations that are attested in the training set, and generalize only when better information is not available. Cohn and Lapata [CL09] accomplish this by incorporating rules of varying depth, allowing their extraction algorithm to make the additional nondeterministic choice to continue recursing on aligned subtrees.

We adopt a different approach, applying within the STSG framework several of the annotations shown to improve SCFG-based sentence compression by Galley and McKeown [GM07]. Specifically, we implement parent annotation and head-lexicalization using both the head word and its part of speech.

3.4.1 Parent Annotation

Extracting shallower rules provides better generality but reduces the amount of structural information contained in each rule. Many rules will manifest some of the same problems seen with SCFGs. A simple way to build more structural information back into these rules is to condition on the label of each node's parent as well as the label of the node itself. With parent annotation, the traditional tree fragment in Figure 3.3 is rewritten as:

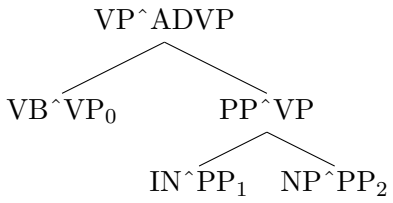

Figure 3.5: Parent-annotated version of the traditional tree fragment from Figure 3.3

Rather than applying to any input subtree of type VP(VB PP(IN NP)), the resulting rule will now only match this structure when it occurs as a child of an ADVP. Thus, a VP of this shape directly under the root symbol S, which is likely a larger and more important subtree of the input with different properties than the training instance, will not be transformed by this rule.

3.4.2 Lexicalization

Another way of obtaining more specific rules is to tag each node with the lexical head underneath it. This allows our grammar to differentiate between subtrees with the same overall structure but different words at the leaves. Suppose in training our system had observed both the tree pair from Figure 3.2 and the tree pair shown in Figure 3.6.

Without lexicalization, the system would observe instances of the rule

$$\langle \text{VP,VP} \rangle \rightarrow \langle \text{VP}(\text{VB}_0\ \text{PP}(\text{IN}_1\ \text{NP}_2)),\ \text{VP}(\text{VB}_0\ \text{NP}_2) \rangle$$

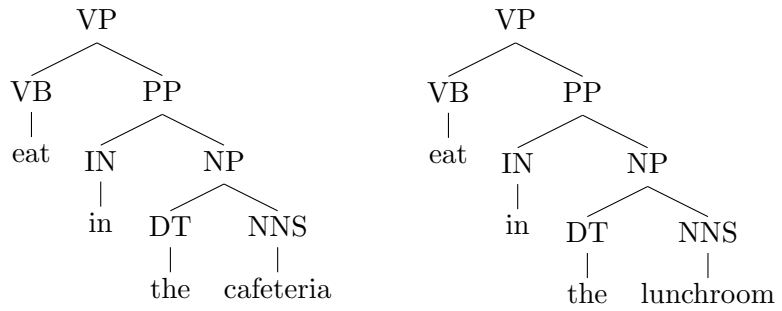

Figure 3.6: Another pair of tree fragments

from both, and if the phrase "eat in the cafeteria" appears as input, it will be incorrectly transformed as described above. However, with lexicalization, the system would learn both the rule

$$\langle \text{VP[assemble]}, \text{VP[join]} \rangle \rightarrow$$

$$\langle \text{VP[assemble]}(\text{VB[assemble]}_0\ \text{PP}(\text{IN[in]}_1\ \text{NP[packs]}_2)),\ \text{VP[join]}(\text{VB[join]}_0\ \text{NP[packs]}_2) \rangle$$

and the rule

$$\langle \text{VP[eat]}, \text{VP[eat]} \rangle \rightarrow$$

$$\langle \text{VP[eat]}(\text{VB[eat]}_0\ \text{PP}(\text{IN[in]}_1\ \text{NP[cafeteria]}_2)),\ \text{VP[eat]}(\text{VB[eat]}_0\ \text{PP}(\text{IN[in]}_1\ \text{NP[lunchroom]}_2)) \rangle,$$

applying only the rule appropriate in the context of the input sentence.

A downside of lexicalization is that it requires a different rule for every combination of head words. This results in a large, sparse rule set that is capable of making specific predictions but has comparatively poor coverage of the space of possible inputs.

A middle ground is to tag nodes with the part of speech of the head word rather than the word itself. This results in rules that are more specific than unannotated rules, but provide better coverage than fully lexicalized rules.

3.4.3 Probability Estimation and Copy Rules

In order to have both good coverage of possible inputs and specific rules, we train a four-level backoff model using varying degrees of annotation. At the highest backoff level, rules are augmented with parent annotation and both

kinds of lexicalization. The subsequent levels drop, in order, word-level lexicalization, part of speech-level lexicalization, and parent annotation, leaving the unannotated grammar at the lowest level.

We train this model using maximum likelihood estimation over our training corpus. Each backoff level learns the probability distribution over every traditional tree fragment observed at its level of annotation. The probability of rewriting a traditional tree t into a simple tree s is estimated as

$$P(s|t) = \frac{count(s \wedge t)}{count(t)} \ .$$

If a tree fragment in an input has never been observed at the current level of annotation, we remove one level of annotation and use the probability distribution given by the next less specific model. If no model contains a probability distribution for this tree fragment, we introduce a rule that copies the tree fragment with probability 1.

Two types of out-of-vocabulary problem can occur in this context, and the strategy of adding copy rules provides robustness against them both. In the first, an input contains a tree fragment whose structure has never been seen in training. In this case, copy rules allow the structure to be reproduced, leaving the system to make more informed changes lower down in the tree. In the second, the input contains an unknown word. This only affects transduction at the leaves of the tree, since, at the lower backoff levels, nodes are not annotated with words. Adding probability 1 copy rules allows the program to retain, replace, or delete unseen words based only on the probabilities of rules higher up in the tree for which it does have estimates.

3.5 Decoding and Reranking

Once the grammar has been acquired, it is converted to a finite tree to tree transducer with one state and transitions defined by the grammar's individual rules. Transitions are weighted with the probabilities assigned to the corresponding grammar rule. We use the tree automata package Tiburon [MK06] to apply input sentence to the transducer. This yields a weighted regular tree grammar that generates every tree that can result from rewriting the input tree using the transducer. The probability of each tree in this grammar is equal to the product of the probabilities of all rewrite rules used to produce it during transduction.

We generate the 10,000 most probable trees for each input, and then rerank them based on a log-linear combination of several features, namely:

- The tree's probability.
- The probability of the output tree's yield, as given by an n-gram language model trained on the simple side of the training corpus.
- The probability of the sequence of preterminal nodes in the output tree, as given by an n-gram model trained on the part-of-speech tags of the simple corpus.
- A two-sided length penalty decreasing the score of output sentences whose length, normalized by the length of the input, deviates from the in-corpus mean, which was found empirically to be 0.85.

The first three of these features are intended to ensure that outputs are well-formed according to the grammar of Wikipedia's Simple English. The length penalty is used to prevent both the over-deletion and insertion of out-of-source phrases that T3 encountered.

Each output is ultimately ranked using a score of the form

$$\lambda_0 log(P(s|t)) + \lambda_1 log(P_{LM}(s)) + \lambda_2 log(P_{POS-LM}(s)) + (|0.85 - \frac{length(s)}{length(t)}|)^{\lambda_3}$$

Since we take the logarithm of all three probabilities, we raise the length penalty to λ_3 rather than multiplying to maintain log-linearity. The weights of this score are set using random-restart hill-climbing search [RN03] with BLEU [PPR+02] as a heuristic function. More details on BLEU are given in Section 4.1.3.

Chapter 4

Evaluation

4.1 Experimental Setup

4.1.1 Competing Systems

We compared the performance of our transducer-based program, Wikid-Simple, against several other text simplification and sentence compression programs. These were:

- **T3** [CL09], which is currently the state of art for abstractive text compression.
- **K&M**, a deletion-only, SCFG-based text compressor based on Knight and Marcu's [KM02] approach. K&M employs a naïve, left-to-right, deletion based tree-alignment heuristic, which defines two nodes as aligned if their labels match and the simple node's children are a subsequence of the traditional node's children. This results in a large number of unaligned nodes, especially close to the leaves of each parse tree.
- **Lexicalized K&M**, an implementation of K&M with head-lexicalization and backoff.
- **Moses+Del**, a deleting phrase-based system, Moses+Del, that performed well in preliminary tests.
- A **baseline** system that always returns an unaltered copy of the input sentence.

All syntax-based systems were provided with parse trees for their input sentences by the Berkeley Parser [PK07]. WikidSimple, T3, and Moses+Del also used word-level alignments from GIZA++ [ON00]. The IRST LM Toolkit [MF08] was used to compute WikidSimple's n-gram language models.

4.1.2 Corpus

All programs used in our experiments were trained on our Wikipedia corpus. This corpus contains 137,362 sentence pairs automatically extracted from the English and Simple English sites. Pages between the two sites were aligned if their titles matched exactly. Stubs, one-line pages, meta pages, and disambiguation pages were removed. Paragraphs within aligned documents with normalized TF-IDF cosine similarity above a given threshold were aligned. Sentences within paragraphs were aligned using a dynamic programming approach based on sentence-level edit operations, and then pairs with a TF-IDF cosine similarity at or below 0.5 were pruned. For more detail on this corpus, see Kauchak and Coster[CK11].

The corpus was partitioned into training and test sets that differed slightly between experiments. Our phrase-based and deletion-model systems were trained on 123,626 sentences. 12,363 were reserved as a development set for the phrase-based system. Due to time and memory issues, we were unable to train T3 on the full training set — attempting to do so took used over 100GB of memory and ran on our server for over a week without terminating. T3 was ultimately trained the first 30,000 sentence pairs, the largest subset of the training corpus it successfully completed processing. All systems were tested on the same 1373-pair test set, except for WikidSimple, which required the first 15 of these sentences to be reserved as a development set. The remaining 1358 were used for testing.

4.1.3 Evaluation Metrics

We depart from the standard form of evaluation used in previous sentence compression experiments, which requires eliciting human judgements of grammaticality and importance. Instead, we opt to evaluate our system using BLEU score. BLEU, developed by Papineni et al [PPR$^+$02], rates the similarity of an output sentence to a gold-standard reference, generally produced by a human. It is calculated as the geometric mean of modified n-gram precisions with respect to the reference, combined with a corpus-level brevity penalty [PPR$^+$02]. BLEU is a standard metric for machine

translation and has been demonstrated to correlate with human judgements of translation quality [PPR+02] [Cou03]. We use the Simple Wikipedia sentences aligned to our inputs as reference sentences.

We also report the Oracle BLEU score, which is obtained by allowing an oracle to greedily select the single highest-scoring sentence from the thousand best candidate outputs considered by the system. Oracle scores provide a sense of the quality of candidate sentences each system is capable of generating. They give an upper bound on the BLEU score attainable by each system through improved reranking alone.

In addition to these scores, we measure the mean ratio of the length in words of output sentences to input sentences. This ratio, commonly reported in sentence compression experiments as the *compression rate*, provides a rough indication of the amount of deletion or insertion each system prefers. Between the traditional and simple sides of our corpus, the length ratio averaged 0.85.

Finally, we also document the percentage of input sentences that each program leaves unmodified. Since Simple English is a subset of traditional English, a proportion of input sentences are likely to be valid simple sentences. Over the training corpus, 26.7% of traditional sentences were identical to the aligned simple sentence. While a system that leaves too many inputs unmodified is less useful, some unchanged sentences are to be expected. This percentage is provided to give a sense of whether each system is modifying too many or too few of its inputs.

4.2 Results

The results of our tests are summarized in Table 4.1.

Out of all the systems tested, only the phrase-based system Moses+Del outperformed the baseline for one-best BLEU score. Moses+Del achieved the highest BLEU score, 0.6046. Making no change to the input resulted in a BLEU score of 0.5937. All four syntax-based systems — WikidSimple, Lexicalized K&M, K&M, and T3 — produced one-best BLEU scores that were significantly worse than doing nothing, scoring 0.5640, 0.4976, 0.4061, and 0.2437, respectively.

The high performance of the baseline system can be explained by the close similarity between the input and reference sides of the corpus. Unlike in machine translation, where an unchanged input sentence is in a different

[1] We were unable to obtain an Oracle score for T3, since it outputs only a single best simplification and not an n-best list.

System	BLEU	Oracle	Length Ratio	% Unmodified
WikidSimple	0.5640	**0.6627**	0.8487	57.5%
T3	0.2437	–[1]	0.5808	23.3%
Moses+Del (best phrase-based)	**0.6046**	0.6421	0.9907	56.9%
K&M (deletion only)	0.4061	0.6021	0.6758	10.5%
Lexicalized K&M	0.4976	0.6087	0.8256	20.7%
Baseline (no change)	0.5937	0.5937	1.0	100%
In-Corpus Mean	–	–	0.85	26.7%

Table 4.1: BLEU, Oracle score, mean length ratio, and mean percentage of inputs unmodified for all systems tested, with mean values from training corpus.

language than the reference and is therefore very unlikely to have many n-grams in common with the reference, in text simplification there is likely to be substantial overlap between the input and the reference. Fully 26.7% of simple sentences in the training corpus were identical to the aligned traditional sentence. On average, simple sentences in the training corpus were only 15% shorter than their traditional equivalents, suggesting that a substantial portion of the source sentence was retained even in sentences that underwent some simplification. Also, since reference sentences tend to be shorter than the corresponding inputs, leaving the input unchanged escapes BLEU's one-sided, multiplicative brevity penalty.

T3 showed the poorest performance of any system evaluated here, with a one-best BLEU score of 0.2437. This low score was the result of several problematic behaviors not produced by the other systems. T3 tended to aggressively over-delete, retaining on average only 58% of the original sentence's length, less than any other system tested. This is apparent in many of the sentences T3 produces as output — for instance, it compacts the sentence:

> "In earlier times, they frequently lived on the outskirts of communities, generally in squalor."

to just

> "A lived"

Such short outputs result in high brevity penalties, dragging down BLEU.

In addition to over-shortening, T3 also tended to insert out-of-source material, as previously observed by both Nomoto and Marsi et al. [Nom09] [MKHD10]. For instance, as a simplification of the sentence

> "A cameo role or cameo appearance is a brief appearance of a known person in a work of the performing arts, such as plays, films, video games and television."

T3 returns

> "Later on, Japanese called a cameo role and cameo appearance films."

WikidSimple came the closest to the in-corpus length ratio, most likely because of the two-sided length penalty used in reranking. Moses+Del was far more conservative with deletions, tending to produce output that was only about 1% shorter than the input. Both WikidSimple and Moses+Del left almost 60% of inputs unchanged, more than twice as many as were unchanged in the training corpus.

WikidSimple tended to simplify by deleting prepositional, adjective, and adverbial phrases, and by truncating conjunction phrases to one of their conjuncts. This often resulted in outputs that were syntactically well-formed with some degree of information loss. In some cases, this was a desirable outcome — for example, it converts

> "The Haiti national football team is the national team of Haiti and is controlled by the Fédération Hatïenne de Football."

to

> "The Haiti national football team is the national football team of Haiti.",

which differs from the reference by only one word. In other cases, these changes were destructive to the meaning of the sentence, as when WikidSimple reduced the input

> "An easily visualized metaphor is a group of separate soap bubbles, in which observers living on one bubble can not interact with those on other soap bubbles, even in principle."

down to

> "An easily visualized metaphor is a group."

WikidSimple also produces a number of interesting lexical and phrasal substitutions, including generalizing "football striker" and "football defender" to "football player", "in order to" to "to", "known as" to "called", and "member" to "part".

Table 4.2 shows several sample inputs, along with the gold standard sentence and the output produced by each system.

System	Output
Input	After Anton Szandor Lavey's death, his position as head of the Church of Satan passed on to Blanche Barton.
Reference	After Anton Szandor Lavey 's death, his partner Blanche Barton became head of the Church of Satan.
WikidSimple	After Anton Szandor Lavey 's death, his position passed on to Blanche Barton.
T3	His partner Blanche Barton
M+D	(same as input)
K&M	His position as head of the Church of passed on to Blanche Barton.
LK&M	After Anton Szandor Lavey's death, his position as passed on.
Input	Overall Bamberga is the tenth brightest main belt asteroid after, in order, Vesta, Pallas, Ceres, Iris, Hebe, Juno, Melpomene, Eunomia and Flora.
Reference	(same as input)
WikidSimple	Overall Bamberga is the tenth brightest main belt asteroid.
T3	Overall Bamberga is the main belt.
M+D	(same as input)
K&M	(same as input)
LK&M	(same as input)
Input	Raúl Vicente Amarilla is a former Paraguayan football striker.
Reference	Raúl Vicente Amarilla is a former Paraguayan football player.
WikidSimple	Raúl Vicente Amarilla is a former Paraguayan football player.
T3	(same as input)
M+D:	Raúl Vicente Amarilla is a former Paraguayan football player.
LK&M	is a former Paraguayan football striker.
K&M	(same as input)

Table 4.2: Comparison of sample outputs for all systems tested.

4.2.1 Oracle BLEU Scores

While WikidSimple's one-best BLEU score fell short of the baseline, its Oracle BLEU score was the highest of any system we examined, at 0.6627. This is over 2 BLEU points greater than the Oracle score attained by Moses+Del, and about 7 BLEU points greater than the baseline.

This high score indicates that WikidSimple was able to generate candidate outputs with greater similarity to the reference than any produced by Moses+Del or the baseline. Although its reranker could not reliably promote them to the top of the list, these sentences were given relatively high probabilities, all in the top 10% of the 10,000 sentences generated by the grammar. With improved reranking, WikidSimple has the potential to outperform both the baseline and the phrase-based approach.

Chapter 5

Conclusions

In this paper, we introduced WikidSimple, a data-driven, syntax-based text simplification system designed to work with a larger corpus than had been previously used for this task. This program acquired a probabilistic synchronous tree substitution grammar from a collection of parsed, aligned sentences culled from English and Simple English Wikipedia. It used this grammar as a tree transducer to rewrite input sentences as their most likely simplified form, then reranked these outputs using language models and a length penalty.

We evaluated WikidSimple, the state-of-the-art abstractive compression system T3, and several other text simplification programs on our Wikipedia corpus using BLEU score. While only the phrase-based system Moses+Del produced one-best outputs scoring higher than the baseline approach of returning the input sentence, WikidSimple outscored T3 and two other syntax-based systems. WikidSimple was able to train faster and more completely then T3, and to generate substantially higher scoring sentences without requiring computationally expensive discriminative learning.

WikidSimple also produced higher-scoring sentences in its thousand best candidate outputs than any other system tested, surpassing Moses+Del by 2 BLEU points and the baseline by 7. Its synchronous grammar consistently generated sentences that scored higher than any produced by the phrase-based system, and it has the potential to return these sentences as output with improved reranking.

5.1 Future work

Data-driven text simplification is a relatively new problem, with myriad opportunities for future work. All of our systems that generated n-best lists showed a wide discrepancy between the BLEU score of their one-best output and the score of outputs chosen by an oracle. Better reranking of output sentences could close this gap across all these systems, without requiring deep changes to any particular model. For WikidSimple, we experimented with a small set of features – only an n-gram language model, a part of speech n-gram model, and a length penalty. An investigation into which features improve simplification quality would help all of these systems realize their potential.

Future work should also include a detailed study of evaluation metrics for this domain. No such study has yet been undertaken, and our results raise some questions about the appropriateness of BLEU for this task. Both the similarity of simplified English sentences to their traditional forms and BLEU's one-sided brevity penalty often favor systems that avoid making changes that a human reader would recognize as simplifications. It is possible that using multiple reference simplifications might improve this, but other evaluation metrics should be investigated. Other metrics have been explored in a sentence compression context, including Simple String Accuracy (SSA) and F-score of the grammatical relations between the input and the reference. Clarke and Lapata [CL06] report that this F-score correlates with human judgements of sentence compression quality, so it may prove worthwhile to investigate its applicability to this domain.

Finally, better alignment of sentences between English and Simple English Wikipedia would improve the performance of data-driven text simplifiers in general. Our corpus was aligned using simple text similarity heuristics, resulting in some alignments that are recognizably not simplifications. When these mis-alignments appear in training, they cause text simplification systems to learn incorrect edit operations. When they occur in testing, they cause output sentences to be compared to an inappropriate standard. Raising the quality of sentence alignment could eliminate these sources of error.

Bibliography

[Ano11a] Anonymous. Wikipedia main page. `http://en.wikipedia.org/wiki/Main_Page`, April 2011.

[Ano11b] Anonymous. Wikipedia:simple english wikipedida. `http://simple.wikipedia.org/wiki/Wikipedia:Simple_English_Wikipedia`, April 2011.

[Ano11c] Anonymous. Wikipedia:simple english wikipedida. `http://simple.wikipedia.org/wiki/Main_Page`, April 2011.

[BPPM93] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *CL.*, 19:263–311, June 1993.

[Chi06] David Chiang. An introduction to synchronous grammars. Part of a tutorial given at ACL., 2006.

[CK11] William Coster and David Kauchak. Simple english wikipedia: A new text simplification task. In *Proceedings of ACL (Short Paper)*, 2011.

[CL06] James Clarke and Mirella Lapata. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of COLING-ACL*, pages 377–384, 2006.

[CL07] Trevor Cohn and Mirella Lapata. Large margin synchronous generation and its application to sentence compression. In *Proceedings of EMNLP-CoNLL*, pages 73–82, 2007.

[CL08] Trevor Cohn and Mirella Lapata. Sentence compression beyond word deletion. In *Proceedings of COLING*, pages 137–144, 2008.

[CL09] Trevor Cohn and Mirella Lapata. Sentence compression as tree transduction. *JAIR*, 34(1):637–674, 2009.

[CMC+98] John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10, 1998.

[Cou03] Deborah Coughlin. Correlating automated and human assessments of machine translation quality. In *Proceedings of MT Summit IX*, pages 63–70, 2003.

[CS97] Raman Chandrasekar and Bangalore Srinivas. Automatic induction of rules for text simplification. *KBS*, 10(3):183–190, 1997.

[CTAC00] Yvonne Canning, John Tait, Jackie Archibald, and Ros Crawley. Cohesive generation of syntactically simplified newspaper text. In *Proceedings of TSD*, pages 145–150, 2000.

[GM07] Michel Galley and Kathleen McKeown. Lexicalized Markov grammars for sentence compression. In *Proceedings of HLT-NAACL*, pages 180–187, 2007.

[JM00] Hongyan Jing and Kathleen R. McKeown. Cut and paste based text summarization. In *Proceedings of NAACL*, pages 178–185, 2000.

[KHB+07] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris C. Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180. ACL, 2007.

[KM02] Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *AI*, 139(1):91–107, 2002.

[Mah97] K. Mahesh. Hypertext summary extraction for fast document browsing. In *Working Notes of the AAAl Spring Symposium on NLP for WWW*, pages 95–103, 1997.

[McD06] Ryan McDonald. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the ACL*, 2006.

[MF08] M. Cettolo M. Federico, N. Bertoldi. Irstlm: an open source toolkit for handling large scale language models. In *Proceedings of Interspeech*, Brisbane, Australia, 2008.

[MK06] Jonathan May and Kevin Knight. Tiburon: A weighted tree automata toolkit. In Oscar H. Ibarra and Hsu-Chun Yen, editors, *Proceedings of CIAA 2006*, volume 4094 of *Lecture Notes in Computer Science*, pages 102–113, Taipei, Taiwan, August 2006. Springer.

[MKHD10] Erwin Marsi, Emiel Krahmer, Iris Hendrickx, and Walter Daelemans. On the limits of sentence compression by deletion. In E. Krahmer and M. Theune, editors, *Empirical Methods in NLG*, volume 5790 of *LNCS*, pages 45–66. 2010.

[Nom08] Tadashi Nomoto. A generic sentence trimmer with CRFs. In *Proceedings of ACL-HLT*, pages 299–307, 2008.

[Nom09] Tadashi Nomoto. A comparison of model free versus model intensive approaches to sentence compression. In *Proceedings of EMNLP*, pages 391–399, 2009.

[ON00] F. J. Och and H. Ney. Improved statistical alignment models. In *ACL00*, pages 440–447, Hongkong, China, October 2000.

[ON04] Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *COLI*, 30(4):417–449, 2004.

[Pit10] Emily Pitler. Methods for sentence compression. Technical Report MS-CIS-10-20, University of Pennsylvania Department of Computer and Information Science, 2010.

[PK07] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *In Proceedings of HTL-NAACL 2007*, pages 404–411, Rochester, New York, April 2007. Association for Computational Linguistics.

[PO07] Sarah E. Petersen and Mari Ostendorf. Text simplification for language learners: a corpus analysis. In *In Proc. of Workshop on Speech and Language Technology for Education*, 2007.

[PPR+02] Kishore Papineni, Kishore Papineni, Salim Roukos, Salim Roukos, Todd Ward, Todd Ward, Wei jing Zhu, and Wei jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318, 2002.

[RN03] S Russell and P Norvig. Artificial intelligence: A modern approach, 2003.

[Shi04] Stuart M.. Shieber. Synchronous Grammars As Tree Transducers. In *In Proceedings of TAG+7*, pages 88–95, 2004.

[TC05] Jenine Turner and Eugene Charniak. Supervised and unsupervised learning for sentence compression. In *Proceedings of ACL*, pages 290–297, 2005.

[TJHA05] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, September 2005.

[YN08] Elif Yamangil and Rani Nelken. Mining wikipedia revision histories for improving sentence compression. In *Proceedings of HLT-NAACL*, pages 137–140, 2008.