Lexicalized Markov Grammars for Sentence Compression*

Michel Galley and **Kathleen R. McKeown**
Columbia University
Department of Computer Science
New York, NY 10027, USA
{galley,kathy}@cs.columbia.edu

Abstract

We present a sentence compression system based on synchronous context-free grammars (SCFG), following the successful noisy-channel approach of (Knight and Marcu, 2000). We define a head-driven Markovization formulation of SCFG deletion rules, which allows us to lexicalize probabilities of constituent deletions. We also use a robust approach for tree-to-tree alignment between arbitrary document-abstract parallel corpora, which lets us train lexicalized models with much more data than previous approaches relying exclusively on scarcely available document-compression corpora. Finally, we evaluate different Markovized models, and find that our selected best model is one that exploits head-modifier bilexicalization to accurately distinguish adjuncts from complements, and that produces sentences that were judged more grammatical than those generated by previous work.

1 Introduction

Sentence compression addresses the problem of removing words or phrases that are not necessary in the generated output of, for instance, summarization and question answering systems. Given the need to ensure grammatical sentences, a number of researchers have used syntax-directed approaches that perform transformations on the output of syntactic parsers (Jing, 2000; Dorr et al., 2003). Some of them (Knight and Marcu, 2000; Turner and Charniak, 2005) take an empirical approach, relying on formalisms equivalent to probabilistic synchronous context-free grammars (SCFG) (Lewis and Stearns, 1968; Aho and Ullman, 1969) to extract compression rules from aligned Penn Treebank (PTB) trees. While their approach proved successful, their reliance on standard maximum likelihood estimators for SCFG productions results in considerable sparseness issues, especially given the relative flat structure of PTB trees; in practice, many SCFG productions are seen only once. This problem is exacerbated for the compression task, which has only scarce training material available.

In this paper, we present a head-driven Markovization of SCFG compression rules, an approach that was successfully used in syntactic parsing (Collins, 1999; Klein and Manning, 2003) to alleviate issues intrinsic to relative frequency estimation of treebank productions. Markovization for sentence compression provides several benefits, including the ability to condition deletions on a flexible amount of syntactic context, to treat head-modifier dependencies independently, and to lexicalize SCFG productions.

Another part of our effort focuses on better alignment models for extracting SCFG compression rules from parallel data, and to improve upon (Knight and Marcu, 2000), who could only exploit 1.75% of the Ziff-Davis corpus because of stringent assumptions about human abstractive behavior. To alleviate their restrictions, we rely on a robust approach for aligning trees of arbitrary document-abstract sentence pairs. After accounting for sentence pairs with both substitutions and deletions, we reached a retention of more than 25% of the Ziff-Davis data, which greatly benefited the lexical probabilities incorporated into our Markovized SCFGs.

Our work provides three main contributions:

*This material is based on research supported in part by the U.S. National Science Foundation (NSF) under Grant No. IIS-05-34871 and the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or DARPA.

Proceedings of NAACL HLT 2007, pages 180–187,
Rochester, NY, April 2007.

(1) Our lexicalized head-driven Markovization yields more robust probability estimates, and our compressions outperform (Knight and Marcu, 2000) according to automatic and human evaluation. (2) We provide a comprehensive analysis of the impact of different Markov orders for sentence compression, similarly to a study done for PCFGs (Klein and Manning, 2003). (3) We provide a framework for exploiting document-abstract sentence pairs that are not purely compressive, and augment the available training resources for syntax-directed sentence compression systems.

2 Synchronous Grammars for Sentence Compression

One successful syntax-driven approach (Knight and Marcu, 2000, henceforth K&M) relies on synchronous context-free grammars (SCFG) (Lewis and Stearns, 1968; Aho and Ullman, 1969). SCFGs can be informally defined as context-free grammars (CFGs) whose productions have two right-hand side strings instead of one, namely *source* and *target* right-hand side. In the case of sentence compression, we restrict the target side to be a sub-sequence of the source side (possibly identical), and we will call this restricted grammar a *deletion SCFG*. For instance, a deletion SCFG rule that removes an adverbial phrase (ADVP) between an noun phrase (NP) and a verb phrase (VP) may be written as follows:

$$\text{S} \rightarrow \langle \text{NP ADVP VP, NP VP} \rangle$$

In a sentence compression framework similar to the one presented by K&M, we build SCFGs that are fully trainable from a corpus of document and reduced sentences. Such an approach comprises two subproblems: (1) transform tree pairs into synchronous grammar derivations; (2) based on these derivations, assign probabilities to deletion SCFG productions, and more generally, to compressions produced by such grammars. Since the main point of our paper lies in the exploration of better probability estimates through Markovization and lexicalization of SCFGs, we first address the latter problem, and discuss the task of building synchronous derivations only later in Section 4.

2.1 Stochastic Synchronous Grammars

The overall goal of a sentence compression system is to transform a given input sentence $\mathbf{f}$ into a concise and grammatical sentence $\mathbf{c} \in \mathbf{C}$, which is a subsequence of $\mathbf{f}$. Similarly to K&M and many successful syntactic parsers (Collins, 1999; Klein and Manning, 2003), our sentence compression system is *generative*, and attempts to find the optimal compression $\hat{\mathbf{c}}$ by estimating the following function:[1]

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c} \in \mathbf{C}} \left\{ p(\mathbf{c}|\mathbf{f}) \right\} = \arg\max_{\mathbf{c} \in \mathbf{C}} \left\{ p(\mathbf{f}, \mathbf{c}) \right\} \quad (1)$$

If $\tau(\mathbf{f}, \mathbf{c})$ is the set of all tree pairs that yield $(\mathbf{f}, \mathbf{c})$ according to some underlying SCFG, we can estimate the probability of the sentence pair using:

$$p(\mathbf{f}, \mathbf{c}) = \sum_{(\pi_\mathbf{f}, \pi_\mathbf{c}) \in \tau(\mathbf{f}, \mathbf{c})} P(\pi_\mathbf{f}, \pi_\mathbf{c}) \quad (2)$$

We note that, in practice (and as in K&M), Equation 2 is often approximated by restricting $\tau(\mathbf{f}, \mathbf{c})$ to a unique full tree $\hat{\pi}_\mathbf{f}$, the best hypothesis of an off-the-shelf syntactic parser. This implies that each possible compression $\mathbf{c}$ is the target-side yield of at most one SCFG derivation.

As in standard PCFG history-based models, the probability of the entire structure (Equation 2) is factored into probabilities of grammar productions. If θ is a derivation $\theta = r^1 \circ \cdots \circ r^j \cdots \circ r^J$, where r^j denotes the SCFG rule $l^j \rightarrow \langle \alpha_f^j, \alpha_c^j \rangle$, we get:

$$p(\pi_\mathbf{f}, \pi_\mathbf{c}) = \prod_{j=1}^{J} p(\alpha_f^j, \alpha_c^j | l^j) \quad (3)$$

The question we will now address is how to estimate the probability $p(\alpha_f^j, \alpha_c^j | l^j)$ of each SCFG production.

2.2 Lexicalized Head-Driven Markovization of Synchronous Grammars

A main issue in our enterprise is to reliably estimate productions of deletion SCFGs. In a sentence compression framework as the one presented by K&M, we use aligned trees of the form of the Penn Treebank (PTB) (Marcus et al., 1994) to acquire and score SCFG productions. However, the use of the PTB structure faces many challenges also encountered in probabilistic parsing.

[1] In their noisy-channel approach, K&M further break down $p(\mathbf{c}, \mathbf{f})$ into $p(\mathbf{f}|\mathbf{c}) \cdot p(\mathbf{c})$, which we refrain from doing for reasons that will become obvious later.

Firstly, PTB tree structures are relatively flat, particularly within noun phrases. For instance, adjective phrases (ADJP)—which are generally good candidates for deletions—appear in 90 different NP-rooted SCFG productions in Ziff-Davis,[2] 61 of which appear only once, e.g., NP → ⟨DT ADJP JJ NN NN, DT JJ NN NN⟩. While it may seem advantageous to maintain many constituents within the same domain of locality of an SCFG production, as we may hope to exploit its large syntactic context to condition deletions more accurately, the sparsity of such productions make them poor candidates for relative frequency estimation, especially in a task with limited quantities of training material. Indeed, our base training corpus described in Section 4 contains only 951 SCFG productions, 593 appearing once.

Secondly, syntactic categories in the PTB are particularly coarse grained, and lead to many incorrect context-free assumptions. Some important distinctions, such as between arguments and adjuncts, are beyond the scope of the PTB annotation, and it is often difficult to determine out of context whether a given constituent can safely be deleted from a right-hand side.

One first type of annotation that can effectively be added to each syntactic category is its lexical head and head part-of-speech (POS), following work in syntactic parsing (Collins, 1999). This type of annotation is particular beneficial in the case of, e.g., prepositional phrases (PP), which may be either complement or adjunct. As in the case of Figure 1 (in which adjuncts appear in italic), knowing that the PP headed by "from" appears in a VP headed by "fell" helps us to determine that the PP is a complement to the verb "fell", and that it should presumably not be deleted. Conversely, the PP headed by "because" modifying the same verb is an adjunct, and can safely be deleted if unimportant.[3] Also, as discussed in (Klein and Manning, 2003), POS annotation can be useful as a means of backing off to more frequently occurring head-modifier POS occurrences (e.g., VBD-IN) when specific bilexical co-

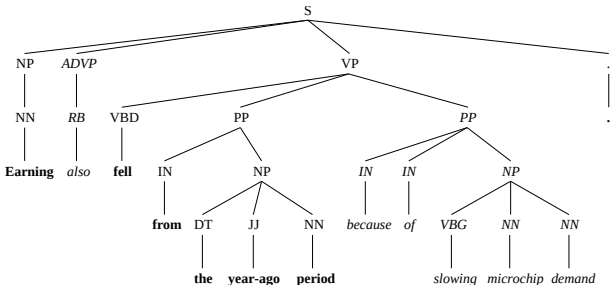

Figure 1: Penn Treebank tree with adjuncts in italic.

occurrences are sparsely seen (e.g., "fell"-"from"). At a lower level, lexicalization is clearly desirable for pre-terminals. Indeed, current SCFG models such as K&M have no direct way of preventing highly improbable single word removals, such as deletions of adverbs "never" or "nowhere", which may turn a negative statement into a positive one.[4]

A second type of annotation that can be added to syntactic categories is the so-called *parent annotation* (Johnson, 1998), which was effectively used in syntactic parsing to break unreasonable context-free assumptions. For instance, a PP with a VP parent is marked as PP^VP. It is reasonable to assume that, e.g., that constituents deep inside a PP have more chances to be removed than otherwise expected, and one may seek to increase the amount of vertical context that is available for conditioning each constituent deletion.

To achieve the above desiderata for better SCFG probability estimates—i.e., reduce the amount of sister annotation within each SCFG production, by conditioning deletions on a context smaller than an entire right-hand side, and at the same time increase the amount of ancestor and descendent annotation through parent (or ancestor) annotation and lexicalization—we follow the approach of (Collins, 1999; Klein and Manning, 2003), i.e., factorize n-ary grammar productions into products of n right-hand side probabilities, a technique sometimes called *Markovization*.

Markovization is generally head-driven, i.e., reflects a decomposition centered around the head of each CFG production:

$$l \rightarrow \Delta L^m \cdots L^1 H R^1 \cdots R^n \Delta \qquad (4)$$

[2]Details about the SCFG extraction procedure are given in Section 4. In short, we refer here to a grammar generated from 823 sentence pairs.

[3]The PP headed by "from" is an optional argument, and thus may still be deleted. Our point is that lexical information in general should help give lower scores to deletions of constituents that are grammatically more prominent.

[4]K&M incorporate lexical probabilities through n-gram models, but such language models are obviously not good for preventing such unreasonable deletions.

where H is the head, $L^1, \ldots, L^m$ the left modifiers, $R^1, \ldots, R^n$ are right modifiers, and Δ termination symbols needed for accurate probability estimations (e.g., to capture the fact that certain constituents are more likely than others to be the rightmost constituent); for simplicity, we will ignore Δ in later discussions. For a given SCFG production $l \rightarrow \langle \alpha_f, \alpha_c \rangle$, we ask, given the source RHS α_f that is assumed given (e.g., provided by a syntactic parser), which of its RHS elements are also present in α_c. That is, we write:

$$p(\alpha_c|\alpha_f, l) = \\ p(k_l^m, \cdots, k_l^1, k_h, k_r^1, \cdots, k_r^n|\alpha_f, l) \quad (5)$$

where k_h, k_l^i, k_r^j ('k' for keep) are binary variables that are true if and only if constituents H, L_i, R^j (respectively) of the source RHS α_f are present in the target side α_c. Note that the conditional probability in Equation 5 enables us to estimate Equation 3, since $p(\alpha_f, \alpha_c|l) = p(\alpha_c|\alpha_f, l) \cdot p(\alpha_f|l)$. We can rely on a state-of-the-art probabilistic parser to effectively compute either $p(\alpha_f|l)$ or the probability of the entire tree $\pi_{\mathbf{f}}$, and need not worry about estimating this term. In the case of sentence compression from the one-best hypothesis of the parser, we can ignore $p(\alpha_f|l)$ altogether, since $\pi_{\mathbf{f}}$ is the same for all compressions.

We can rewrite Equation 5 exactly using a head-driven infinite-horizon Markovization:

$$p(\alpha_c|\alpha_f, l) = p(k_h|\alpha_f, l) \\ \cdot \prod_{i=1\ldots m} p(k_l^i|k_l^1, \cdots, k_l^{i-1}, k_h, \alpha_f, l) \\ \cdot \prod_{i=1\ldots n} p(k_r^i|k_r^1, \cdots, k_r^{i-1}, k_h, \Lambda, \alpha_f, l) \quad (6)$$

where $\Lambda = (k_l^1, \cdots, k_l^m)$ is a term needed by the chain rule. One key issue is to make linguistically plausible assumptions to determine which conditioning variables in the terms should be deleted. Following our discussion in the first part of this section, we may start by making an order-s Markov approximation centered around the head, i.e., we condition each binary variable (e.g., k_r^i) on a context of up to s sister constituents between the current constituent and the head (e.g., $(R^{i-s}, \ldots, R^i)$). In order to incorporate bilexical dependencies between the head and each modifier, we also condition all modifier probabilities on head variables H (and k_h). These assumptions are overall quite similar to the ones made in Markovized parsing models. If we assume that all other conditioning variables in Equation 6 are irrelevant, we write:

$$p(\alpha_c|\alpha_f, l) = p_h(k_h|H, l) \\ \cdot \prod_{i=1\ldots m} p_l(k_l^i|L^{i-s}, ..., L^i, k_l^{i-s}, ..., k_l^{i-1}, H, k_h, l) \\ \cdot \prod_{i=1\ldots n} p_r(k_r^i|R^{i-s}, ..., R^i, k_r^{i-s}, ..., k_r^{i-1}, H, k_h, l) \quad (7)$$

Note that it is important to condition deletions on both constituent histories $(R^{i-s}, \ldots, R^i)$ and non-deletion histories $(k_r^{i-s}, \ldots, k_r^{i-1})$; otherwise we would be unable to perform deletions that must operate jointly, as in production S $\rightarrow \langle$ADVP COMMA NP VP, NP VP$\rangle$ (in which the ADVP should not be deleted without the comma). Without binary histories, we often observed superfluous punctuation symbols and dangling coordinate conjunctions appearing in our outputs.

Finally, we label l with an order-v ancestor annotation, e.g., for the VP in Figure 1, $l = \epsilon$ for $v = 0$, $l =$VP^S for $v = 2$, and so on. We also replace H and modifiers L^i and R^i by lexicalized entries, e.g., $H =$(VP,VBD,*fell*) and $R^i =$(PP,IN,*from*). Note that to estimate $p_l(k_l^i|\cdots)$, we only lexicalize L^i and H, and none of the other conditioning modifiers, since this would, of course, introduce too many conditioning variables (the same goes for $p_r(k_r^i|\cdots)$). The question of how much sister and vertical (s and v) context is needed for effective sentence compression, and whether to use lexical or POS annotation, will be evaluated in detail in Section 5.

3 The Data

To acquire SCFG productions, we used Ziff-Davis, a corpus of technical articles and human abstractive summaries. Articles and summaries are paired by document, so the first step was to perform sentence alignment. In the particular case of sentence compression, a simple approach is to just consider compression pairs (**f**,**c**), where **c** is a substring of **f**. K&M identified only 1,087 such paired sentences in the entire corpus, which represents a recall of 1.75%.

For our empirical evaluations, we split the data as follows: among the 1,055 sentences that were taken

to train systems described in K&M, we selected the first 32 sentence pairs to be an auxiliary test corpus (for future work), the next 200 sentences to be our development corpus, and the remaining 823 to be our base training corpus (ZD-0), which will be augmented with additional data as explained in the next section. We feel it is important to use a relatively large development corpus, since we will provide in Section 5 detailed analyses of model selection on the development set (e.g., by evaluating different Markov structures), and we want these findings to be as significant as possible. Finally, we used the same test data as K&M for human evaluation purposes (32 sentence pairs).

4 Tree Alignment and Synchronous Grammar Inference

We now describe methods to train SCFG models from sentence pairs. Given a tree pair $(\mathbf{f}, \mathbf{c})$, whose respective parses $(\pi_{\mathbf{f}}, \pi_{\mathbf{c}})$ were generated by the parser described in (Charniak and Johnson, 2005), the goal is to transform the tree pair into SCFG derivations, in order to build relative frequency estimates for our Markovized models from observed SCFG productions. Clearly, the two trees may sometimes be structurally quite different (e.g., a given PP may attach to an NP in $\pi_{\mathbf{f}}$, while attaching to VP in $\pi_{\mathbf{c}}$), and it is not always possible to build an SCFG derivation given the constraints in $(\pi_{\mathbf{f}}, \pi_{\mathbf{c}})$. The approach taken by K&M is to analyze both trees and count an SCFG rule whenever two nodes are "deemed to correspond", i.e., roots are the same, and α_c is a sub-sequence of α_f. This leads to a quite restricted number of different productions on our base training set (ZD-0): 823 different productions were extracted, 593 of which appear only once. This first approach has serious limitations; the assumption that sentence compression appropriately models human abstractive data is particularly problematic. This considerably limits the amount of training data that can be exploited in Ziff-Davis (which contains overall more than 4,000 documents-abstract pairs), and this makes it very difficult to train lexicalized models.

An approach to slightly loosen this assumption is to consider document-abstract sentence pairs in which the condensed version contains one or more substitutions or insertions. Consider for example

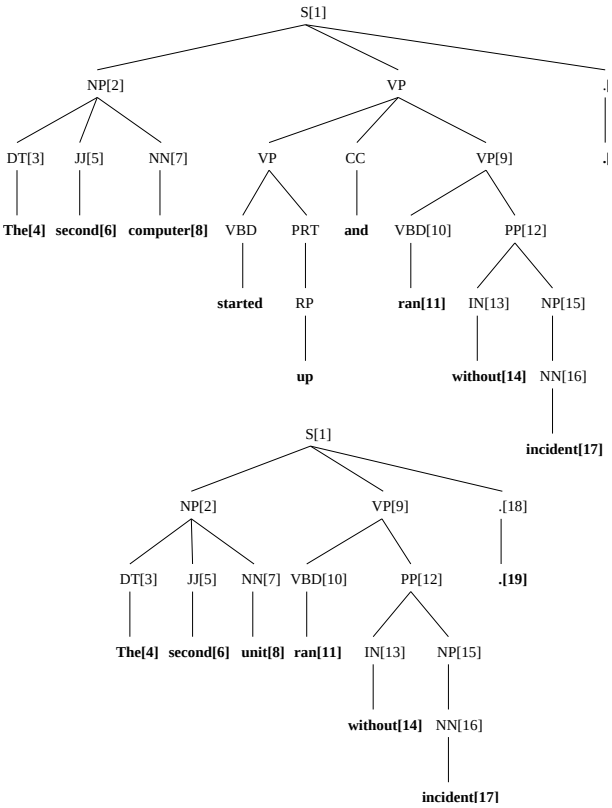

Figure 2: Full sentence and its revision. While the latter is not a compression of the former, it could still be used to gather statistics to train a sentence compression system, e.g., to learn the reduction of a VP coordination.

the tree pair in Figure 2: the two sentences are syntactically very close, but the substitution of "computer" with "unit" makes this sentence pair unusable in the framework presented in K&M. Arguably, there should be ways to exploit abstract sentences that are slightly reworded in addition to being compressed. To use sentence pairs with insertions and substitutions, we must find a way to align tree pairs in order to identify SCFG productions. More specifically, we must define a constituent alignment between the paired abstract and document sentences, which determine how the two trees are synchronized in a derivation. Obtaining this alignment is no trivial matter as the number of non-deleting edits increases. To address this, we synchronized tree pairs by finding the constituent alignment that minimizes the edit distance between the two trees, i.e., minimize the number of terminals and non-terminals insertions, substitutions and deletions.[5] While criteria

[5]The minimization problem is known to be NP hard, so we used an approximation algorithm (Zhang and Shasha, 1989) that

other than minimum tree edit distance may be effective, we found—after manual inspections of alignments between sentences with less than five non-deleting edits—that this method generally produces good alignments. A sample alignment is provided in Figure 2. Once a constituent alignment is available, it is then trivial to extract all deletion SCFG rules available in a tree pair, e.g., NP $\rightarrow \langle$DT JJ NN, DT JJ NN$\rangle$ in the figure.

We also exploited more general tree productions known as synchronous tree substitution grammar (STSG) rules, in an approach quite similar to (Turner and Charniak, 2005). For instance, the STSG rule rooted at S can be decomposed into two SCFG productions if we allow unary rules such as VP $\rightarrow$ VP to be freely added to the compressed tree. More specifically, we decompose any STSG rule that has in its target (compressed) RHS a single context free production, and that contains in its source (full) RHS a single context free production adjoined with any number of tree adjoining grammar (TAG) auxiliary trees (Joshi et al., 1975). In the figure, the initial tree is S $\rightarrow$ NP VP, and the adjoined (auxiliary) tree is VP $\rightarrow$ VP CC VP.[6] We found this approach quite helpful, since most useful compressions that mimic TAG adjoining operations are missed by the extraction procedure of K&M.

Since we found that exploiting sentence pairs containing insertions had adverse consequences in terms of compression accuracies, we only report experiments with sentence pairs containing no insertions. We gathered sentence pairs with up to six substitutions using minimum edit distance matching (we will refer to these sets as ZD-0 to ZD-6). With a limit of up to six substitutions (ZD-6), we were able to train our models on 16,787 sentences, which represents about 25% of the total number of summary sentences of the Ziff-Davis corpus.

5 Experiments

All experiments presented in this section are performed on the Ziff-Davis corpus. We note first that all probability estimates of our Markovized grammars are smoothed. Indeed, incorporating lexical dependencies within models trained on data sets as small as 16,000 sentence pairs would be quite futile without incorporating robust smoothing techniques. Different smoothing techniques were evaluated with our models, and we found that interpolated Witten-Bell discounting was the method that performed best. We used relative frequency estimates for each of the models presented in Section 2.2 (i.e., p_h, p_l, p_r), and trained p_l separately from p_r. We interpolated our most specific models (lexical heads, POS tags, ancestor and sister annotation) with lower-order models.[7]

Automatic evaluation on development sets is performed using word-level classification accuracy, i.e., the number of words correctly classified as being either deleted or not deleted, divided by the total number of words. In our first evaluation, we experimented with different horizontal and vertical Markovizations (Table 1). First, it appears that vertical annotation is moderately helpful. It provides gains in accuracy ranging from .5% to .9% for $v = 1$ over a simpler models ($v = 0$), but higher orders ($v > 1$) have a tendency to decrease performance. On the other hand, sister annotation of order 1 is much more critical, and provides 4.1% improvement over a simpler model ($s = 0, v = 0$). Manual examinations of compression outputs confirmed this analysis: without sister annotation, deletion of punctuation and function words (determiners, coordinate conjunctions, etc.) is often inaccurate, and compressions clearly lack fluency. This annotation is also helpful for phrasal deletions; for instance, we found that PPs are deleted in 31.4% of cases in Ziff-Davis if they do not immediately follow the head constituent, but this percentage drops to 11.1% for PPs that immediately follow the head. It seems, however, that increasing sister annotation beyond $s > 1$ only provide limited improvements.

In our second evaluation reported in Table 2, we

runs in polynomial time.

[6] To determine whether a given one-level tree is an auxiliary, we simply check the following properties: all its leaves but one (the "foot node") must be nodes attached to deleted subtrees (e.g., VP and CC in the figure), and the foot node (VP[9]) must have the same syntactic category as the root node.

[7] We relied on the SRI language modeling (SRILM) toolkit library for all smoothing experiments. We used the following order in our deleted interpolation of p_h: lexical head, head POS, ancestor annotation, and head category. For p_l and p_r, we removed first: lexical head, lexical head of the modifier, head POS, head POS of the modifier, sister annotation (L^i deleted before k_l^i), k_h, category of the head, category of the modifier. We experimented with different deletion interpolation orderings, and this ordering appears to work quite well in practice, and was used in all experiments reported in this paper.

assessed the usefulness of lexical and POS annotation (setting s and v to 0). In the table, we use M to denote any of the modifiers L_i or R_i, and c, t, w respectively represent syntactic constituent, POS, and lexical conditioning. While POS annotation is clearly advantageous compared to using only syntactic categories, adding lexical variables to the model also helps. As is shown in the table, it is especially important to know the lexical head of the modifier we are attempting to delete. The addition of w_m to conditioning variables provides an improvement of 1.3% (from 66.5% to 67.8%) on our optimal Ziff-Davis training corpus (ZD-6). Furthermore, bilexical head-modifier dependencies provide a relatively small improvement of .5% (from 69.8% to 70.3%) over the best model that does not incorporate the lexical head w_h. Note that lexical conditioning also helps in the case where the training data is relatively small (ZD-0), though differences are less significant, and bilexical dependencies actually hurt performance. In subsequent experiments, we experimented with different Markovizations and lexical dependency combination, and finally settled with a model ($s = 1$ and $v = 1$) incorporating all conditioning variables listed in the last line of Table 2. This final tuning was combined with human inspection of generated outputs, since certain modifications that positively impacted output quality seldom changed accuracies.

We finally took the best configuration selected above, and evaluated our model against the noisy-channel model of K&M on the 32 test sentences selected by them. We performed both automatic and human evaluation against the output produced by Knight and Marcu's original implementation of their noisy channel model (Table 3). In the former case, we also provide Simple String Accuracies (SSA).[8] For human evaluation, we hired six native-speaker judges who scored grammaticality and content (importance) with scores from 1 to 5, using instructions as described in K&M. Both types of evaluations favored our Markovized model against the noisy channel model.

Table 4 shows several outputs of our system

[8]SSA is defined as: $SSA = 1 - (I + D + S)/R$. The numerator terms are respectively the number of inserts, deletes, and substitutions, and R is the length of the reference compression.

Vertical Order	Horizontal Order $s=0$	$s=1$	$s=2$	$s=3$
$v=0$	63	67.1	67.2	67.2
$v=1$	63.9	67.6	**67.7**	67.7
$v=2$	65.7	66.6	66.9	66.9
$v=3$	65.2	66.8	67.1	67

Table 1: Markovizations accuracies on Ziff-Davis devel set.

Conditioning Variables		ZD-0	ZD-3	ZD-6
$M=c_m$	$H=c_h$	62.2	62.4	64.4
$M=(c_m,t_m)$	$H=c_h$	63.0	63.4	66.5
$M=(c_m,w_m)$	$H=c_h$	64.2	65.2	66.7
$M=(c_m,t_m,w_m)$	$H=c_h$	63.8	65.8	67.8
$M=(c_m,t_m,w_m)$	$H=(c_h,t_h)$	66.7	68.6	69.8
$M=(c_m,t_m,w_m)$	$H=(c_h,w_h)$	66.9	68.9	**70.3**
$M=(c_m,t_m,w_m)$	$H=(c_h,t_h,w_h)$	66.3	69.1	69.8

Table 2: Accuracies on Ziff-Davis devel set with different head-modifier annotations.

Models	Acc	SSA	Grammar	Content	Len(%)
NoisyC	61.3	14.6	4.37 ± 0.5	3.87 ± 1.2	70.4
Markov	67.9	31.7	4.68 ± 0.4	4.22 ± 0.4	62.7
Human	-	-	4.95 ± 0.1	4.43 ± 0.3	53.3

Table 3: Accuracies on Ziff-Davis test set.

(Markov) that significantly differed from the output of the noisy channel model (NoisyC), which confirms our finding that Markovized models can produce quite grammatical output. Our compression for the first sentence underlines one of the advantages of constituent-based classifiers, which have the ability of deleting a very long phrase (here, a PP) at once. The three next sentences display some advantages of our approach over the K&M model: here, the latter model performs deletion with too little lexico-syntactic information, and accidentally removes certain modifiers that are sometimes, but not always, good candidates for deletions (e.g., ADJP in Sentence 2, PP in sentences 3 and 4). On the other hand, our model keeps these constituent intact. Finally, the fifth and last example is one of the only three cases (among the 32 sentences) where our model produced a sentence we judged clearly ungrammatical. After inspection, we found that our parser assigned particularly errorful trees to those inputs, which may partially explain these ungrammatical outputs.

6 Related Work

A relatively large body of work addressed the problem of sentence compression. One successful recent approach (McDonald, 2006) combines a discriminative framework with a set of features that capture information similar to the K&M model. Mc-

Input	*Many debugging features, including user-defined break points and variable-watching and message-watching windows, have been added.*
NoisyC	Many debugging features, including user-defined points and variable-watching and message-watching windows, have been added.
Markov	Many debugging features have been added.
Human	Many debugging features have been added.
Input	*The chemical etching process used for glare protection is effective and will help if your office has the fluorescent-light overkill that 's typical in offices.*
NoisyC	The process used for glare protection is and will help if your office has the overkill
Markov	The chemical etching process used for glare protection is effective.
Human	Glare protection is effective.
Input	*The utilities will be bundled with Quickdex II in a $90 package called super quickdex, which is expected to ship in late summer.*
NoisyC	The utilities will be bundled
Markov	The utilities will be bundled with Quickdex II.
Human	The utilities will be bundled with Quickdex II.
Input	*The discounted package for the SparcServer 470 is priced at $89,900, down from the regular $107,795.*
NoisyC	The package for the 470 is priced
Markov	The discounted package for the SparcServer 470 is at $89,900.
Human	The SparcServer 470 is priced at $89,900, down from the regular $107,795.
Input	*Prices range from $5,000 for a microvax 2000 to $179,000 for the vax 8000 or higher series.*
NoisyC	Prices range from $5,000 for a 2000 to $179,000 for the vax 8000 or higher series.
Markov	Prices range from $5,000 for a microvax for the vax.
Human	Prices range from $5,000 to $179,000.

Table 4: Compressions of sample test sentences.

Donald's features include compression bigrams, as well as soft syntactic evidence extracted from parse trees and dependency trees. The strength of McDonald's approach partially stems from its robustness against redundant and noisy features, since each feature is weighted proportionally to its discriminative power, and his approach is thus hardly penalized by uninformative features. In contrast, our work puts much more emphasis on feature analysis than on efficient optimization, and relies on a statistical framework (maximum-likelihood estimates) that strives for careful feature selection and combination. It also describes and evaluates models incorporating syntactic evidence that is new to the sentence compression literature, such as head-modifier bilexical dependencies, and nth-order sister and vertical annotation. We think this work leads to a better understanding of what type of syntactic and lexical evidence makes sentence compression work. Furthermore, our work leaves the door open to uses of our factored model in a constituent-based or word-based discriminative framework, in which each elementary lexico-syntactic structure of this paper can be discriminatively weighted to directly optimize compression quality. Since McDonald's approach does not incorporate SCFG deletion rules, and conditions deletions on less lexico-syntactic context, we believe this will lead to levels of performance superior to both papers.

7 Conclusions

We presented a sentence compression system based on SCFG deletion rules, for which we defined a head-driven Markovization formulation. This Markovization enabled us to incorporate lexical conditioning variables into our models. We empirically evaluated different Markov structures, and obtained a best system that generates particularly grammatical sentences according to a human evaluation. Our sentence compression system is freely available for research and educational purposes.

Acknowledgments

We would like to thank Owen Rambow, Michael Collins, Julia Hirschberg, and Daniel Ellis for their helpful comments and suggestions.

References

A. Aho and J. Ullman. 1969. Syntax directed translations and the pushdown assembler. 3:37–56.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*.

M. Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, U. of Pennsylvania.

B. Dorr, D. Zajic, and R. Schwartz. 2003. Hedge: A parse-and-trim approach to headline generation. In *Proc. of DUC*.

H. Jing. 2000. Sentence reduction for automatic text summarization. In *Proc. of NAACL*, pages 310–315.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

A. Joshi, L. Levy, and M. Takahashi. 1975. Tree adjunct grammar. *Journal of Computer and System Science*, 21(2).

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

K. Knight and D. Marcu. 2000. Statistics-based summarization — step one: Sentence compression. In *Proc. of AAAI*.

P. Lewis and R. Stearns. 1968. Syntax-directed transduction. In *Journal of the Association for Computing Machinery*, volume 15, pages 465–488.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

R. McDonald. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proc. of EACL*.

J. Turner and E. Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proc. of ACL*.

K. Zhang and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.