Large Margin Synchronous Generation and its Application to Sentence Compression

Trevor Cohn and **Mirella Lapata**
School of Informatics
University of Edinburgh
Edinburgh, United Kingdom
{tcohn,mlap}@inf.ed.ac.uk

Abstract

This paper presents a tree-to-tree transduction method for text rewriting. Our model is based on synchronous tree substitution grammar, a formalism that allows local distortion of the tree topology and can thus naturally capture structural mismatches. We describe an algorithm for decoding in this framework and show how the model can be trained discriminatively within a large margin framework. Experimental results on sentence compression bring significant improvements over a state-of-the-art model.

1 Introduction

Recent years have witnessed increasing interest in text-to-text generation methods for many natural language processing applications ranging from text summarisation to question answering and machine translation. At the heart of these methods lies the ability to perform rewriting operations according to a set of prespecified constraints. For example, text simplification identifies which phrases or sentences in a document will pose reading difficulty for a given user and substitutes them with simpler alternatives (Carroll et al., 1999). Sentence compression produces a summary of a single sentence that retains the most important information while remaining grammatical (Jing, 2000).

Ideally, we would like a text-to-text rewriting system that is not application specific. Given a parallel corpus of training examples, we should be able to learn rewrite rules and how to combine them in order to generate new text. A great deal of previous work has focused on the rule induction problem (Barzilay and McKeown, 2001; Pang et al., 2003; Lin and Pantel, 2001; Shinyama et al., 2002), whereas relatively little emphasis has been placed on the actual generation task (Quirk et al., 2004). A notable exception is sentence compression for which end-to-end rewriting systems are commonly developed (Knight and Marcu, 2002; Turner and Charniak, 2005; Galley and McKeown, 2007; Riezler et al., 2003; McDonald, 2006). The appeal of this task lies in its simplified formulation as a single rewrite operation, namely word deletion (Knight and Marcu, 2002).

Solutions to the compression task have been cast mostly in a supervised learning setting (but see Clarke and Lapata (2006a), Hori and Furui (2004), and Turner and Charniak (2005) for unsupervised methods). Rewrite rules are learnt from a parsed parallel corpus and subsequently used to find the best compression from the set of all possible compressions for a given sentence. A common assumption is that the tree structures representing long sentences and their compressions are *isomorphic*. Consequently, the models are not generally applicable to other text rewriting problems since they cannot readily handle structural mismatches and more complex rewriting operations such as substitutions or insertions. A related issue is that the tree structure of the compressed sentences is often poor; most algorithms delete words or constituents without paying too much attention to the structure of the compressed sentence. However, without an explicit generation mechanism that allows tree transformations, there is no guarantee that the compressions will have well-formed syntactic structures. And it will not be easy to process them for subsequent generation or analysis tasks.

In this paper we present a text-to-text rewriting

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 73–82, Prague, June 2007. ©2007 Association for Computational Linguistics

model that scales to *non-isomorphic* cases and can thus naturally account for structural and lexical divergences. Our approach is inspired by *synchronous tree substitution grammar* (STSG, Eisner (2003)) a formalism that allows local distortion of the tree topology. We show how such a grammar can be induced from a parallel corpus and propose a large margin model for the rewriting task which can be viewed as a weighted tree-to-tree transducer. Our learning framework makes use of the algorithm put forward by Tsochantaridis et al. (2005) which efficiently learns a prediction function to minimise a given loss function. Experiments on sentence compression show significant improvements over the state-of-the-art. Beyond sentence compression and related text-to-text generation problems (*e.g.,* paraphrasing), our model is generally applicable to tasks involving structural mapping. Examples include machine translation (Eisner, 2003) or semantic parsing (Zettlemoyer and Collins, 2005).

2 Related Work

Knight and Marcu (2002) proposed a noisy-channel formulation of sentence compression based on synchronous context-free grammar (SCFG). The latter is a generalisation of the context-free grammar (CFG) formalism to simultaneously produce strings in two languages. In the case of sentence compression, the grammar rules have two right hand sides, one corresponding to the *source* (long) sentence and the other to its *target* compression. The synchronous derivations are learnt from a parallel corpus and their probabilities are estimated generatively.

Given a long sentence, l, the aim is to find the corresponding compressed sentence, s, which maximises $P(s)P(l|s)$ (here $P(s)$ is the source model and $P(l|s)$ the channel model.) Modifications of this model are reported in Turner and Charniak (2005) and Galley and McKeown (2007) with improved results. The channel model is limited to tree deletion and does not allow any type of tree re-organisation.

Non-isomorphic tree structures are common when translating between languages. It is therefore not surprising that most previous work on tree rewriting falls within the realm of machine translation. Proposals include Eisner's (2003) synchronous tree substitution grammar (STSG), Melamed's (2004) multitext grammar, and Graehl and Knight's (2004) tree-to-tree transducers. Despite differences in formalism, all these approaches model the translation process using tree-based probabilistic transduction rules. The grammar induction process requires EM training which can be computationally expensive especially if all synchronous rules are considered.

Our work formulates sentence compression in the framework of STSG (Eisner, 2003). We propose a novel grammar induction algorithm that does not require EM training and is coupled with a separate large margin training process (Tsochantaridis et al., 2005) for weighting each rule. McDonald (2006) also presents a sentence compression model that uses a discriminative large margin algorithm. However, we differ in two important respects. First, our generation algorithm is more powerful, performing complex tree transformations, whereas McDonald only considers simple word deletion. Being tree-based, the generation algorithm is better able to preserve the grammaticality of the compressed output. Second, our model can be tuned to a wider range of loss functions (*e.g.,*tree-based measures).

3 Problem Formulation

We formulate sentence compression as an instance of the general problem of learning a mapping from input patterns $\mathbf{x} \in \mathcal{X}$ to discrete structured objects $\mathbf{y} \in \mathcal{Y}$. Our training sample consists of a parallel corpus of input (uncompressed) and output (compressed) pairs $(x_1, y_1) \ldots (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ and our task is to predict a target labelled tree $\mathbf{y}$ from a source labelled tree $\mathbf{x}$. As we describe below, $\mathbf{y}$ is not precisely a target tree, but instead derivations which generate both the source and the target tree. We model the dependency between $\mathbf{x}$ and $\mathbf{y}$ as a weighted STSG. Grammar rules are of the form $\langle X, Y \rangle \rightarrow \langle \gamma, \alpha, \beta \rangle$ where γ and α are *elementary trees* composed of a mixture of terminal and non-terminals rooted with non-terminals X and Y respectively, and β is a set of variable correspondences between pairs of frontier non-terminals in γ and α. A grammar rule specifies that we can substitute the trees γ and α for corresponding X and Y nodes in the source and target trees respectively. For example, the rule:

$$\langle NP,\ NP \rangle \rightarrow \langle [DT_{\boxed{1}}\ ADJP\ NN_{\boxed{2}}]_{NP}, [DT_{\boxed{1}}\ NN_{\boxed{2}}]_{NP} \rangle$$

allows adjective phrases to be dropped from the source tree within an NP. The indices $\boxed{x}$ are used to specify the variable correspondences, β.

Each grammar rule has a score from which the overall score of a compression $\mathbf{y}$ for sentence $\mathbf{x}$ can be derived. These scores are learnt discriminatively using the large margin technique proposed by Tsochantaridis et al. (2005). The synchronous rules are combined using a chart-based parsing algorithm (Eisner, 2003) to generate the derivation (i.e., compressed tree) with the highest score.

We begin by describing our STSG generation algorithm in Section 3.1. We next explain how a synchronous grammar is induced from a parallel corpus of original sentences and their compressions (Section 3.2) and give the details of our learning framework (Section 3.3).

3.1 Generation

Generation aims to find the best target tree for a given source tree using the transformations specified by the synchronous grammar. (We discuss how we obtain this grammar in the following section.)

$$\mathbf{y}^* = \max_{\mathbf{y} \in \mathcal{Y}} score(\mathbf{x}, \mathbf{y}; \mathbf{w}) \qquad (1)$$

where $\mathbf{y}$ ranges over all target derivations (and therefore trees), $\mathbf{w}$ is a parameter vector and $score(\cdot)$ is an objective function measuring the quality of the derivation. In common with many parsing methods, we encounter a problem with *spurious ambiguity*: i.e., there may be many *derivations* (sequences of rule applications) which produce the same target tree. Ideally we would sum up the scores over all these derivations, however for the sake of tractability we instead take the maximum score. This allows us to pose the maximisation problem over *derivations* rather than target trees.

The generation algorithm uses a dynamic program defined over the constituents in the source tree as shown in Figure 1 (see also Eisner (2003)). The algorithm makes the assumption that the scoring function decomposes with the derivation, such that a partial score can be evaluated at each step, i.e., $score(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{r \in \mathbf{y}} score(r; \mathbf{w})$ where r are the rules used in the derivation. This method builds a *chart* of the best scoring partial derivation for each source subtree headed by a given target non-terminal. The inductive step is applied recursively bottom-up, and involves applying a grammar rule to a node in the source tree. Rules with substitution variables in their frontier are scored with reference to the chart for the matching nodes and target non-terminal categories. Once the process is complete, we can read the best score from the chart cell for the root node, and the best derivation can be constructed by traversing *back-pointers* also stored in the chart. This is illustrated in Figure 2 where the rule $\langle VP, VP \rangle \rightarrow \langle [[\text{is}_{AUX}\ ADJP_{\boxed{1}}]_{VP}\ CC\ VP]_{VP}, [\text{is}_{AUX}\ NP_{\boxed{1}}]_{VP} \rangle$ is applied to the top VP node. The score of the resulting tree would reference the chart to calculate the score for the best target tree at the *ADJP* node with syntactic category *NP*.

```
for all nodes, n, in source tree (bottom-up) do
    for all rules, r with left side matching node, n_r = n do
        s = score(r)
        for all variables v in r do
            score = score + chart[n_v, c_v]
        end for
        update chart[n, c_r] with score, s, if better than current
    end for
end for
c_best = argmax_c chart[root, c]
find best derivation using back-pointers from (root, c_best)
```

Figure 1: Generation algorithm to find the best derivation. n_r and n_v are the source nodes indexed by the rule's source side (root and variable), while c_r and c_v are the non-terminal categories of the rule's target side (root and variable).

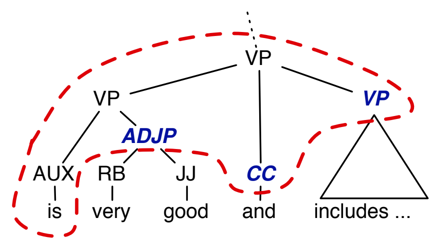


Figure 2: Example of a rule application during generation. The dashed area shows a matching rule for the VP node.

3.2 Grammar Induction

Our induction algorithm automatically finds grammar rules from a word-aligned parsed parallel corpus. The rules are pairs of *elementary trees* (i.e., tree fragments) whose leaf nodes are linked by the word alignments. These leaves can be either terminal or non-terminal symbols. Initially, the algorithm extracts tree pairs from word aligned text by choosing aligned constituents in the source and the target. These pairs are then generalised using subtrees which are also extracted, resulting in synchronous rules with variable nodes. The set of aligned tree pairs are extracted using the alignment template method (Och and Ney, 2004), constrained to syntactic constituent pairs:

$$\mathcal{C} = \{(n_S, n_T),\ (\exists(s,t) \in \mathcal{A} \wedge s \in Y(n_S) \wedge t \in Y(n_T)) \wedge (\nexists(s,t) \in \mathcal{A} \wedge (s \in Y(n_S) \veebar t \in Y(n_T)))\}$$

where n_S and n_T are source and target tree nodes (subtrees), $\mathcal{A} = \{(s,t)\}$ is the set of word alignments (pairs of word-indices), $Y(\cdot)$ returns the yield span for a subtree and $\veebar$ is the exclusive-or operator.

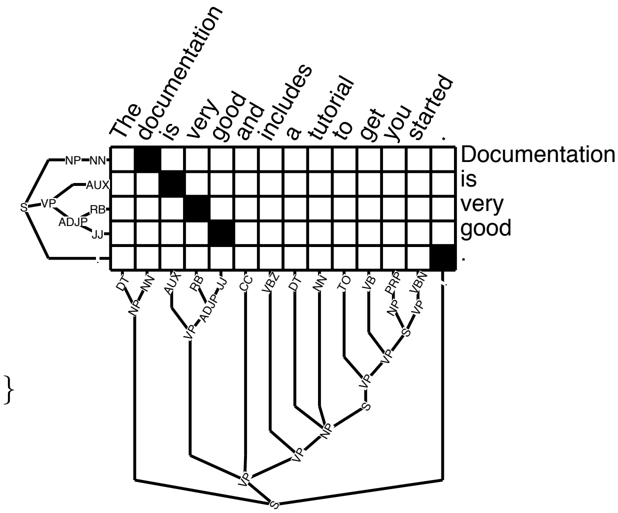
The next step is to generalise the candidate pairs by replacing subtrees with variable nodes. We could fully trust the word alignments and adopt a strategy in which the rules are generalised as much as possible and thus include little lexicalisation. Figure 3 shows a simple sentence pair and the resulting synchronous rules according to this generalisation strategy. Alternatively, we could extract every possible rule by including unlexicalised rules, lexicalised rules and their combination. The downside here is that the total number of possible rules is factorial in the size of the candidate set. We address this problem by limiting the number of variables and the recursion depth, and by filtering out singleton rules.

There is no guarantee that the induced rules will generalise well to a testing set. For example, the testing data may have a rule which was not seen in the training set (*e.g.,* a new terminal or non terminal). In this case no rule can be applied and subsequently generation fails. For this reason we allow the model to duplicate any CFG production from the source tree, and uses a feature to flag that this rule was unseen in training. These SCFG rules are then merged with the induced rules and fed into the feature detection module (see Section 3.3 for details).

3.3 The Large Margin Model

We now describe how the parameters of our STSG generation system are fit to a supervised training set. For a given source tree, the space of sister target trees implied by the synchronous grammar is often very large, and the majority of these trees are ungrammatical or are poor compressions. The training procedure learns weights such that the model can discriminate between these trees and predict a good target tree. For this we develop a discriminative training process which learns a weighted tree-to-tree transducer. Our model is based on Tsochantaridis et al.'s (2005) framework for learning Support Vector Machines (SVMs) with structured output spaces, using the SVM^{struct} implementation.[1] We briefly summarise the approach below; for a more detailed description we refer the interested reader to Tsochantaridis et al. (2005).

Traditionally SVMs learn a linear classifier that separates two or more classes with the largest possible margin. Analogously, structured SVMs attempt to separate the correct structure from all other



$$\begin{aligned}
\langle S,S\rangle &\rightarrow \langle [NP_{\boxed{1}}\ VP_{\boxed{2}}\ ._{\boxed{3}}]_S, [NP_{\boxed{1}}\ VP_{\boxed{2}}\ ._{\boxed{3}}]_S\rangle\\
\langle NP,NP\rangle &\rightarrow \langle [DT\ NN_{\boxed{1}}]_{NP}, [NN_{\boxed{1}}]_{NP}\rangle\\
\langle NN,NN\rangle &\rightarrow \langle \text{documentation}_{NN}, \text{Documentation}_{NN}\rangle\\
\langle VP,VP\rangle &\rightarrow \langle VP_{\boxed{1}}\ CC\ VP, VP_{\boxed{1}}\rangle\\
\langle VP,VP\rangle &\rightarrow \langle AUX_{\boxed{1}}\ ADJP_{\boxed{2}}, AUX_{\boxed{1}}\ ADJP_{\boxed{2}}\rangle\\
\langle AUX,AUX\rangle &\rightarrow \langle \text{is}_{AUX}, \text{is}_{AUX}\rangle\\
\langle ADJP,ADJP\rangle &\rightarrow \langle [RB_{\boxed{1}}\ JJ_{\boxed{2}}]_{ADJP}, [RB_{\boxed{1}}\ JJ_{\boxed{2}}]_{ADJP}\rangle\\
\langle RB,RB\rangle &\rightarrow \langle \text{very}_{RB}, \text{very}_{RB}\rangle\\
\langle JJ,JJ\rangle &\rightarrow \langle \text{good}_{ADJ}, \text{good}_{ADJ}\rangle\\
\langle .,.\rangle &\rightarrow \langle ._., ._.\rangle
\end{aligned}$$

Figure 3: Induced synchronous grammar from a sentence pair using a strategy that extracts general rules.

[1] http://svmlight.joachims.org/svm_struct.html

structures with a large margin. Given an input instance $\mathbf{x}$, we search for the optimum output $\mathbf{y}$ under the assumption that $\mathbf{x}$ and $\mathbf{y}$ can be adequately described using a combined feature vector representation $\Psi(\mathbf{x},\mathbf{y})$. Recall that $\mathbf{x}$ are the source trees and $\mathbf{y}$ are synchronous derivations which generate both $\mathbf{x}$ and a target tree.

$$f(\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{y\in\mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y})\rangle \quad (2)$$

The goal of the training procedure is to find a parameter vector $\mathbf{w}$ such that it satisfies the condition:

$$\forall i, \forall y \in \mathcal{Y}\backslash y_i : \langle \mathbf{w}, \Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\mathbf{y})\rangle \geq 0 \quad (3)$$

where $\mathbf{x}_i, \mathbf{y}_i$ are the ith training source tree and target derivation. To obtain a unique solution — there will be several parameter vectors $\mathbf{w}$ satisfying (3) if the training instances are linearly separable — Tsochantaridis et al. (2005) select the $\mathbf{w}$ that maximises the minimum distance between $\mathbf{y}_i$ and the closest runner-up structure.

The framework also incorporates a loss function. This property is particularly appealing in the context of sentence compression and generally text-to-text generation. For example, a compression that differs from the gold standard with respect to one or two words should be treated differently from a compression that bears no resemblance to it. Another important factor is the length of the compression. Compressions whose length is similar to the gold standard should be be preferable to longer or shorter output. A loss function $\Delta(\mathbf{y}_i,\mathbf{y})$ quantifies the accuracy of prediction $\mathbf{y}$ with respect to the true output value $\mathbf{y}_i$. We give details of the loss functions we employed for the compression task below.

We are now ready to state the learning objective for the structured SVM. We use the soft-margin formulation which allows errors in the training set, via the slack variables ξ_i:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i,\ \xi_i \geq 0 \quad (4)$$
$$\forall i, \forall \mathbf{y} \in \mathcal{Y}\backslash \mathbf{y}_i : \langle \mathbf{w}, \delta\Psi(\mathbf{y})\rangle \geq 1 - \frac{\xi_i}{\Delta(\mathbf{y}_i,\mathbf{y})}$$

Slack variables ξ_i are introduced here for each training example x_i, C is a constant that controls the trade-off between training error minimisation and margin maximisation, and $\delta\Psi(\mathbf{y})$ is a shorthand for $\Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\mathbf{y})$ (see (3)). Note that slack variables are rescaled with the inverse loss incurred in each of the linear constraints.[2]

The optimisation problem in (4) is approximated using a polynomial time cutting plane algorithm (Tsochantaridis et al., 2005). This optimisation crucially relies on finding the constraint incurring the maximum cost. The cost function for slack rescaling can be formulated as:

$$H(\mathbf{y}) = (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w}\rangle)\Delta(\mathbf{y}_i,\mathbf{y}) \quad (5)$$

In order to adapt this framework to our generation problem, we must provide the feature mapping $\Psi(\mathbf{x},\mathbf{y})$, a loss function $\Delta(\mathbf{y}_i,\mathbf{y})$, and a maximiser $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y})$ (see (5)). The following sections describe how these are instantiated in the sentence compression task.

Feature Mapping We devised a general feature set suitable for compression and paraphrasing. Our feature space is defined over source trees ($\mathbf{x}$) and target derivations ($\mathbf{y}$). All features apply to a single grammar rule; a feature vector for a derivation is expressed as the sum of the feature vectors for each rule in this derivation.

We make use of syntactic, lexical, and compression specific features. Our simplest syntactic feature is the identity of a synchronous rule. Specifically, we record its source tree, its target tree and their combination. We also include rule frequencies $\phi(target|source)$, $\phi(source|target)$ and $\phi(source, target)$. Another feature records the frequencies of the CFG productions used in the target side of a rule. This allows the model to learn the weights of a CFG generation grammar, as a proxy for a language model. Using scores from a pre-trained CFG grammar or an n-gram language model might be preferable when the training sample is small, however we leave this as future work. Our last syntactic feature keeps track of the source root and the target root non-terminals. Our lexical features contain the list of tokens in the source yield, target yield, and both. We also use words as features.

[2] Alternatively, the loss function can be used to rescale the margin. This approach is less desirable as it is not scale invariant (Tsochantaridis et al., 2005). We also found empirically that slack-rescaling slightly outperforms margin rescaling on our compression task.

Finally, we have implemented a set of compression-specific features. These include a feature that detects if the yield of the target side of a synchronous rule is a subset of the yield of its source. We also take note of the edit operations (i.e., removal, insertion) required to transform the source side into the target. Edit operations are recorded separately for trees and their yields. In order to encourage compression, we also count the number of words on the target, the number of rules used in the derivation and the number of dropped variables.

Loss Functions The large margin configuration sketched above is quite modular and in theory a wide range of loss functions could be specified. Examples include edit-distance, precision, F-score, BLEU and tree-based measures. In practice, the loss function should be compatible with our maximisation algorithm which requires the objective function to decompose along the same lines as the tree derivation.[3]

Given this restriction, we define a loss based on position-independent unigram precision (Prec) which penalises errors in the yield independently for each word. Although fairly intuitive, this loss is far from ideal. First, it maximally rewards repeatedly predicting the same word if the latter is in the reference target tree. Secondly, it may bias towards overly short output which drops core information — one-word compressions will tend to have higher precision than longer output. To counteract this, we introduce two brevity penalty measures (BP) inspired by BLEU (Papineni et al., 2002) which we incorporate into the loss function, using a product, $loss = 1 - \text{Prec} \cdot \text{BP}$:

$$\begin{aligned} \text{BP1} &= \exp(1 - \max(1, \frac{r}{c})) \\ \text{BP2} &= \exp(1 - \max(\frac{c}{r}, \frac{r}{c})) \end{aligned} \tag{6}$$

where r is the reference length and c is the candidate length.

BP1 is asymmetric, it has value one when $c \geq r$ and decays to zero when $c < r$. Note that precision should decay when $c > r$ as extra output will often not match the reference. BP2 is two-sided: it has value one when $c = r$ and decays towards zero for $c < r$ and $c > r$. In both cases, brevity is assessed against the gold standard target (not the source) to allow the system to learn the correct degree of compression from the training data.

Maximisation Algorithm Our algorithm finds the maximising derivation for $H(\mathbf{y})$ in (5). This derivation will have a high loss and a high score under the model, and therefore represents the most-violated constraint which is then added to the SVM's working set of constraints (see (4)).

The standard generation method from Section 3.1 cannot be used without modification to find the best scoring derivation since it does not account for the loss function or the gold standard derivation. Instead, we *stratify* the generation chart with the number of true and false positive tokens predicted, as described in Joachims (2005). These contingency values allow us to compute the precision and brevity penalty (see (6)) for each complete derivation. This is then combined with the derivation score and the gold standard derivation score to give $H(\mathbf{y})$.

The gold standard derivation features, $\Psi(\mathbf{x}_i, \mathbf{y}_i)$, must be calculated from a derivation linking the source tree to the gold target tree. As there may be many such derivations, we find a *unique* derivation using the smallest rules possible (for maximum generality). This is done using a dynamic program, similar to the inside-outside algorithm used in parsing. Other strategies are also possible, however we leave this to future work. Finally, we can find the global maximum $H(\mathbf{y})$ by maximising over all the root chart entries.

4 Evaluation Set-up

In this section we present our experimental set-up for assessing the performance of the max margin model described above. We give details of the corpora used, briefly introduce McDonald's (2006) sentence compression model used for comparison with our approach, and explain how system output was evaluated.

Corpora We evaluated our system on two different corpora. The first is the compression corpus of Knight and Marcu (2002) derived automatically from the document-abstract pairs of the Ziff-

[3]Optimising non-decompositional loss functions complicates the objective function, which then cannot be solved efficiently using a dynamic program.

Davis corpus. Previous compression work has almost exclusively used this corpus. Our experiments follow Knight and Marcu's partition of training, test, and development sets (1,002/36/12 instances). We also present results on Clarke and Lapata's (2006a) Broadcast News corpus.[4] This corpus was created manually (annotators were asked to produce compressions for 50 Broadcast news stories) and poses more of a challenge than Ziff-Davis. Being a speech corpus, it often contains incomplete and ungrammatical utterances and speech artefacts such as disfluencies, false starts and hesitations. Furthermore, spoken utterances have varying lengths, some are very wordy whereas others cannot be reduced any further. Thus a hypothetical compression system trained on this domain should be able to leave some sentences uncompressed. Again we used Clarke and Lapata's training, test, and development set split (882/410/78 instances).

Comparison with State-of-the-art We evaluated our approach against McDonald's (2006) discriminative model. This model is a good basis for comparison for several reasons. First, it achieves competitive performance with Knight and Marcu's (2002) decision tree and noisy channel models. Second, it also uses large margin learning. Sentence compression is formulated as a string-to-substring mapping problem with a deletion-based Hamming loss. Recall that our formulation involves a tree-to-tree mapping. Third, it uses a feature space complementary to ours. For example features are defined between adjacent words, and syntactic evidence is incorporated indirectly into the model. In contrast our model relies on synchronous rules to generate valid compressions and does not explicitly incorporate adjacency features. We used an implementation of McDonald (2006) for comparison of results (Clarke and Lapata, 2007).

Evaluation Measures In line with previous work we assessed our model's output by eliciting human judgements. Participants were presented with an original sentence and its compression and asked to rate the latter on a five point scale based on the information retained and its grammaticality. We conducted two separate elicitation studies, one for the

O:	I just wish my parents and my other teachers could be like this teacher, so we could communicate.
M:	I wish my teachers could be like this teacher.
S:	I wish my teachers could be like this, so we could communicate.
G:	I wish my parents and other teachers could be like this, so we could communicate.
O:	Earlier this week, in a conference call with analysts, the bank said it boosted credit card reserves by $350 million.
M:	Earlier said credit card reserves by $350 million.
S:	In a conference call with analysts, the bank boosted card reserves by $350 million.
G:	In a conference call with analysts the bank said it boosted credit card reserves by $350 million.

Table 1: Compression examples from the Broadcast news corpus (O: original sentence, M: McDonald (2006), S: STSG, G: gold standard)

Ziff-Davis and one for the Broadcast news dataset. In both cases our materials consisted of 96 source-target sentences. These included gold standard compressions and the output of our system and McDonald's (2006). We were able to obtain ratings on the entire Ziff-Davis test set as it has only 32 instances; this was not possible for Broadcast news as the test section consists of 410 instances. Consequently, we randomly selected 32 source-target sentences to match the size of the Ziff-Davis test set.[5] We collected ratings from 60 unpaid volunteers, all self reported native English speakers. Both studies were conducted over the Internet. Examples of our experimental items are given in Table 1.

We also report results using F1 computed over grammatical relations (Riezler et al., 2003). We chose F1 (as opposed to accuracy or edit distance-based measures) as Clarke and Lapata (2006b) show that it correlates reliably with human judgements.

5 Experiments

The framework presented in Section 3 is quite flexible. Depending on the grammar induction strategy, choice of features, loss function and maximisation algorithm, different classes of models can be derived. Before presenting our results in detail we discuss the specific model employed in our experiments and explain how its parameters were instantiated.

In order to build a compression model we need

[4]The corpus can be downloaded from `http://homepages.inf.ed.ac.uk/s0460084/data/`.

[5]A Latin square design ensured that subjects did not see two different compressions of the same sentence.

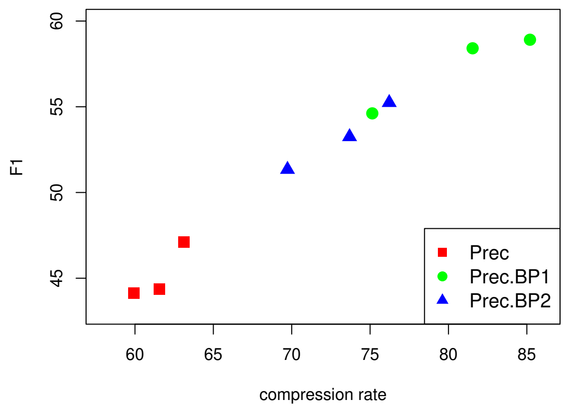


Figure 4: Compression rate vs. grammatical relations F1 using unigram precision alone and in combination with two brevity penalties.

a parallel corpus of syntax trees. We obtained syntactic analyses for source and target sentences with Bikel's (2002) parser. Our corpora were automatically aligned with Giza++ (Och et al., 1999) in both directions between source and target and symmetrised using the intersection heuristic (Koehn et al., 2003). Each word in the lexicon was also aligned with itself. This was necessary in order to inform Giza++ about word identity. Unparseable sentences and those longer than 50 tokens were removed from the data set.

We induced a synchronous tree substitution grammar from the Ziff-Davis and Broadcast news corpora using the method described in Section 3.2. We extracted all maximally general synchronous rules. These were complemented with more specific rules from conjoining pairs of general rules. The specific rules were pruned to remove singletons and those rules with more than 3 variables. Grammar rules were represented by the features described in Section 3.3.

An important parameter for our compression task is the appropriate choice of loss function. Ideally, we would like a loss function that encourages compression without overly aggressive information loss. Figure 4 plots compression rate against grammatical relations F1 using each of the loss functions presented in Section 3.3 on the Ziff-Davis development set.[6] As can be seen with unigram precision alone (Prec)

[6]We obtained a similar plot for the Broadcast News corpus but omit it due to lack of space.

Ziff-Davis	CompR	RelF1
McDonald06	66.2	45.8
STSG	56.8	54.3
Gold standard	57.2	—

Broadcast News	CompR	RelF1
McDonald06	68.6	47.6
STSG	73.7	53.4*
Gold standard	76.1	—

Table 2: Results using grammatical relations F1 (*: sig. diff. from McDonald06; $p < 0.01$ using the Student t test)

the system produces overly short output, whereas the one-sided brevity penalty (BP1) achieves the opposite effect. The two-sided brevity penalty (BP2) seems to strike the right balance: it encourages compression while achieving good F-scores. This suggests that important information is retained in spite of significant compression. We also varied the regularisation parameter C (see (4)) over a range of values on the development set and found that setting it to 0.01 yields overall good performance across corpora and loss functions.

We now present our results on the test set. These were obtained with a model that uses slack rescaling and a precision-based loss function with a two-sided brevity penalty ($C = 0.01$). Table 2 shows the average compression rates (CompR) for McDonald (2006) and our model (STSG) as well as their performance according to grammatical relations F1. The row 'Gold standard' displays human-produced compression rates. Notice that our model obtains compression rates similar to the gold standard, whereas McDonald tends to compress less on Ziff-Davis and more on Broadcast news. As far as F1 is concerned, we see that STSG outperforms McDonald on both corpora. The difference in F1 is statistically significant on Broadcast news but not on Ziff-Davis (which consists solely of 32 sentences).

Table 3 presents the results of our elicitation study. We carried out an Analysis of Variance (ANOVA) to examine the effect of system type (McDonald06, STSG, Gold standard) on the compression ratings. The ANOVA revealed a reliable effect on both corpora. We used post-hoc Tukey tests to

Model	Ziff-Davis	Broadcast news
McDonald06	$2.82^{\dagger}$	$2.16^{\dagger}$
STSG	$3.20^{\dagger *}$	2.63^{*}
Gold standard	3.72	3.05

Table 3: Mean ratings on compression output elicited by humans (*: sig. diff. from McDonald06 ($\alpha < 0.05$); † sig. diff. from Gold standard ($\alpha < 0.01$); using post-hoc Tukey tests)

examine whether the mean ratings for each system differed significantly. The Tukey tests showed that STSG is perceived as significantly better than McDonald06. There is no significant difference between STSG and the gold standard compressions on the Broadcast news; both systems are significantly worse than the gold standard on Ziff-Davis.

These results are encouraging, indicating that our highly expressive framework is a good model for sentence compression. Under several experimental conditions we obtain better performance than previous work. Importantly, the model described here is not compression-specific, it could be easily adapted to other tasks, corpora or languages (for which syntactic analysis tools are available). Being supervised, our model learns to fit the compression rate of the training data. In this sense, it is somewhat inflexible as it cannot easily adapt to a specific rate given by a user or imposed by an application (*e.g.,* when displaying text on small screens). Compression rate can be indirectly manipulated by adopting loss functions that encourage or discourage compression (see Figure 4), but admittedly in other frameworks (*e.g.,* Clarke and Lapata (2006a)) the length of the compression can be influenced more naturally.

In our formulation of the compression problem, a derivation is characterised by a single inventory of features. This entails that the feature space cannot in principle distinguish between derivations that use the same rules, applied in a different order. Although, this situation does not arise often in our dataset, we believe that it can be ameliorated by intersecting a language model with our generation algorithm (Chiang, 2005).

6 Conclusions and Future Work

In this paper we have presented a novel method for sentence compression cast in the framework of structured learning. We develop a system that generates compressions using a synchronous tree substitution grammar whose weights are discriminatively trained within a large margin model. We also describe an appropriate algorithm than can be used in both training (i.e., learning the model weights) and decoding (i.e., finding the most plausible compression under the model). The proposed formulation allows us to capture rewriting operations that go beyond word deletion and can be easily tuned to specific loss functions directly related to the problem at hand. We empirically evaluate our approach against a state-of-the art model (McDonald, 2006) and show performance gains on two compression corpora.

Future research will follow three directions. First, we will extend the framework to incorporate position dependent loss functions. Examples include the Hamming distance or more sophisticated functions that take the tree structure of the source and target sentences into account. Such functions can be supported by augmenting our generation algorithm with a beam search. Secondly, the present paper used a relatively simple feature set. Our intention was to examine our model's performance without extensive feature engineering. Nevertheless, improvements should be possible by incorporating features defined over *n*-grams and dependencies (McDonald, 2006). Finally, the experiments presented in this work use a grammar acquired from the training corpus. However, there is nothing inherent in our formalisation that restricts us to this particular grammar. We therefore plan to investigate the potential of our method with unsupervised or semi-supervised grammar induction techniques for additional rewriting tasks including paraphrase generation and machine translation.

Acknowledgements The authors acknowledge the support of EPSRC (grants GR/T04540/01 and GR/T04557/01). We are grateful to James Clarke for sharing his implementation of McDonald (2006) with us. Special thanks to Philip Blunsom for insightful comments and suggestions.

References

R. Barzilay, K. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL/EACL*, 50–57, Toulouse, France.

D. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of HLT*, 24–27, San Diego, CA.

J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, J. Tait. 1999. Simplifying text for language impaired readers. In *Proceedings of EACL*, 269–270, Bergen, Norway.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd ACL*, 263–270, Ann Arbor, MI.

J. Clarke, M. Lapata. 2006a. Constraint-based sentence compression: An integer programming approach. In *Proceedings of COLING/ACL Main Conference Poster Sessions*, 144–151, Sydney, Australia.

J. Clarke, M. Lapata. 2006b. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of COLING/ACL*, 377–384, Sydney, Australia.

J. Clarke, M. Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of EMNLP-CoNLL*, Prague, Czech Republic.

J. Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the ACL Interactive Poster/Demonstration Sessions*, 205–208, Sapporo, Japan.

M. Galley, K. McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Proceedings of NAACL/HLT*, 180–187, Rochester, NY.

J. Grael, K. Knight. 2004. Training tree transducers. In *Proceedings of NAACL/HLT*, 105–112, Boston, MA.

C. Hori, S. Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems*, E87-D(1):15–25.

H. Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of ANLP*, 310–315, Seattle, WA.

T. Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of ICML*, 377–384, Bonn, Germany.

K. Knight, D. Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

P. Koehn, F. J. Och, D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*, 48–54, Edmonton, Canada.

D. Lin, P. Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):342–360.

R. McDonald. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proceedings of EACL*, 297–304, Trento, Italy.

I. D. Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of ACL*, 653–660, Barcelona, Spain.

F. J. Och, H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. J. Och, C. Tillmann, H. Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of EMNLP/VLC*, 20–28, College Park, MD.

B. Pang, K. Knight, D. Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of NAACL*, 181–188, Edmonton, Canada.

K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, 311–318, Philadelphia, PA.

C. Quirk, C. Brockett, W. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*, 142–149, Barcelona, Spain.

S. Riezler, T. H. King, R. Crouch, A. Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of HLT/NAACL*, 118–125, Edmonton, Canada.

Y. Shinyama, S. Sekine, K. Sudo, R. Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT*, 40–46, San Diego, CA.

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

J. Turner, E. Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of ACL*, 290–297, Ann Arbor, MI.

L. S. Zettlemoyer, M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*, 825–830, Edinburgh, UK.