Bayesian Synchronous Tree-Substitution Grammar Induction and its Application to Sentence Compression

Elif Yamangil and **Stuart M. Shieber**
Harvard University
Cambridge, Massachusetts, USA
{elif, shieber}@seas.harvard.edu

Abstract

We describe our experiments with training algorithms for tree-to-tree synchronous tree-substitution grammar (STSG) for monolingual translation tasks such as sentence compression and paraphrasing. These translation tasks are characterized by the relative ability to commit to parallel parse trees and availability of word alignments, yet the unavailability of large-scale data, calling for a Bayesian tree-to-tree formalism. We formalize nonparametric Bayesian STSG with epsilon alignment in full generality, and provide a Gibbs sampling algorithm for posterior inference tailored to the task of *extractive sentence compression*. We achieve improvements against a number of baselines, including expectation maximization and variational Bayes training, illustrating the merits of nonparametric inference over the space of grammars as opposed to sparse parametric inference with a fixed grammar.

1 Introduction

Given an aligned corpus of tree pairs, we might want to learn a mapping between the paired trees. Such induction of tree mappings has application in a variety of natural-language-processing tasks including machine translation, paraphrase, and sentence compression. The induced tree mappings can be expressed by synchronous grammars. Where the tree pairs are isomorphic, synchronous context-free grammars (SCFG) may suffice, but in general, non-isomorphism can make the problem of rule extraction difficult (Galley and McKeown, 2007). More expressive formalisms such as synchronous tree-substitution (Eisner, 2003) or tree-adjoining grammars may better capture the pairings.

In this work, we explore techniques for inducing synchronous tree-substitution grammars (STSG) using as a testbed application extractive sentence compression. Learning an STSG from aligned trees is tantamount to determining a segmentation of the trees into elementary trees of the grammar along with an alignment of the elementary trees (see Figure 1 for an example of such a segmentation), followed by estimation of the weights for the extracted tree pairs.[1] These elementary tree pairs serve as the *rules* of the extracted grammar. For SCFG, segmentation is trivial — each parent with its immediate children is an elementary tree — but the formalism then restricts us to deriving isomorphic tree pairs. STSG is much more expressive, especially if we allow some elementary trees on the source or target side to be unsynchronized, so that insertions and deletions can be modeled, but the segmentation and alignment problems become nontrivial.

Previous approaches to this problem have treated the two steps — grammar extraction and weight estimation — with a variety of methods. One approach is to use word alignments (where these can be reliably estimated, as in our testbed application) to align subtrees and extract rules (Och and Ney, 2004; Galley et al., 2004) but this leaves open the question of finding the right level of generality of the rules — how deep the rules should be and how much lexicalization they should involve — necessitating resorting to heuristics such as *minimality* of rules, and leading to

[1] Throughout the paper we will use the word STSG to refer to the tree-to-tree version of the formalism, although the string-to-tree version is also commonly used.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 937–947,
Uppsala, Sweden, 11-16 July 2010.

large grammars. Once a given set of rules is extracted, weights can be imputed using a discriminative approach to maximize the (joint or conditional) likelihood or the classification margin in the training data (taking or not taking into account the derivational ambiguity). This option leverages a large amount of manual domain knowledge engineering and is not in general amenable to latent variable problems.

A simpler alternative to this two step approach is to use a generative model of synchronous derivation and simultaneously segment and weight the elementary tree pairs to maximize the probability of the training data under that model; the simplest exemplar of this approach uses expectation maximization (EM) (Dempster et al., 1977). This approach has two frailties. First, EM search over the space of all possible rules is computationally impractical. Second, even if such a search were practical, the method is degenerate, pushing the probability mass towards larger rules in order to better approximate the empirical distribution of the data (Goldwater et al., 2006; DeNero et al., 2006). Indeed, the optimal grammar would be one in which each tree pair in the training data is its own rule. Therefore, proposals for using EM for this task start with a precomputed subset of rules, and with EM used just to assign weights within this grammar. In summary, previous methods suffer from problems of *narrowness* of search, having to restrict the space of possible rules, and *overfitting* in preferring overly specific grammars.

We pursue the use of hierarchical probabilistic models incorporating sparse priors to simultaneously solve both the narrowness and overfitting problems. Such models have been used as generative solutions to several other segmentation problems, ranging from word segmentation (Goldwater et al., 2006), to parsing (Cohn et al., 2009; Post and Gildea, 2009) and machine translation (DeNero et al., 2008; Cohn and Blunsom, 2009; Liu and Gildea, 2009). Segmentation is achieved by introducing a prior bias towards grammars that are compact representations of the data, namely by enforcing *simplicity* and *sparsity*: preferring simple rules (smaller segments) unless the use of a complex rule is evidenced by the data (through repetition), and thus mitigating the overfitting problem. A Dirichlet process (DP) prior is typically used to achieve this interplay. Interestingly, sampling-based nonparametric inference further allows the possibility of searching over the infinite space of grammars (and, in machine translation, possible word alignments), thus side-stepping the narrowness problem outlined above as well.

In this work, we use an extension of the aforementioned models of generative segmentation for STSG induction, and describe an algorithm for posterior inference under this model that is tailored to the task of extractive sentence compression. This task is characterized by the availability of word alignments, providing a clean testbed for investigating the effects of grammar extraction. We achieve substantial improvements against a number of baselines including EM, support vector machine (SVM) based discriminative training, and variational Bayes (VB). By comparing our method to a range of other methods that are subject differentially to the two problems, we can show that both play an important role in performance limitations, and that our method helps address both as well. Our results are thus not only encouraging for grammar estimation using sparse priors but also illustrate the merits of nonparametric inference over the space of grammars as opposed to sparse parametric inference with a fixed grammar.

In the following, we define the task of extractive sentence compression and the Bayesian STSG model, and algorithms we used for inference and prediction. We then describe the experiments in extractive sentence compression and present our results in contrast with alternative algorithms. We conclude by giving examples of compression patterns learned by the Bayesian method.

2 Sentence compression

Sentence compression is the task of summarizing a sentence while retaining most of the informational content and remaining grammatical (Jing, 2000). In *extractive* sentence compression, which we focus on in this paper, an order-preserving subset of the words in the sentence are selected to form the summary, that is, we summarize by deleting words (Knight and Marcu, 2002). An example sentence pair, which we use as a running example, is the following:

- Like FaceLift, much of ATM's screen performance depends on the underlying application.
- ATM's screen performance depends on the underlying application.

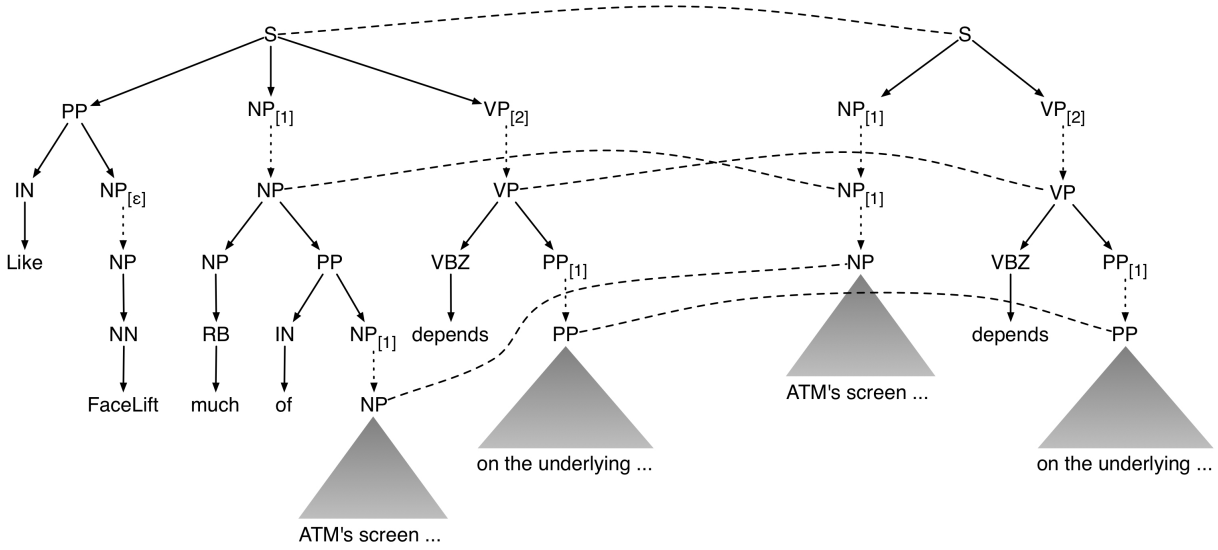


Figure 1: A portion of an STSG derivation of the example sentence and its extractive compression.

where the underlined words were deleted. In *supervised* sentence compression, the goal is to generalize from a parallel training corpus of sentences (source) and their compressions (target) to unseen sentences in a test set to predict their compressions. An *unsupervised* setup also exists; methods for the unsupervised problem typically rely on language models and linguistic/discourse constraints (Clarke and Lapata, 2006a; Turner and Charniak, 2005). Because these methods rely on dynamic programming to efficiently consider hypotheses over the space of all possible compressions of a sentence, they may be harder to extend to general paraphrasing.

3 The STSG Model

Synchronous tree-substitution grammar is a formalism for synchronously generating a pair of non-isomorphic source and target trees (Eisner, 2003). Every grammar rule is a pair of elementary trees aligned at the leaf level at their frontier nodes, which we will denote using the form

$$c_s/c_t \rightarrow e_s/e_t, \quad \gamma$$

(indices s for source, t for target) where c_s, c_t are root nonterminals of the elementary trees e_s, e_t respectively and γ is a 1-to-1 correspondence between the frontier nodes in e_s and e_t. For example, the rule

$$\text{S / S} \rightarrow \text{(S (PP (IN Like) NP}_{[\epsilon]}\text{) NP}_{[1]}\text{ VP}_{[2]}\text{) / (S NP}_{[1]}\text{ VP}_{[2]}\text{)}$$

can be used to delete a subtree rooted at PP. We use square bracketed indices to represent the alignment γ of frontier nodes — $\text{NP}_{[1]}$ aligns with $\text{NP}_{[1]}$, $\text{VP}_{[2]}$ aligns with $\text{VP}_{[2]}$, $\text{NP}_{[\epsilon]}$ aligns with the special symbol ϵ denoting a deletion from the source tree. Symmetrically ϵ-aligned target nodes are used to represent insertions into the target tree. Similarly, the rule

$$\text{NP / } \epsilon \rightarrow \text{(NP (NN FaceLift)) / } \epsilon$$

can be used to continue deriving the deleted subtree. See Figure 1 for an example of how an STSG with these rules would operate in synchronously generating our example sentence pair.

STSG is a convenient choice of formalism for a number of reasons. First, it eliminates the isomorphism and strong independence assumptions of SCFGs. Second, the ability to have rules deeper than one level provides a principled way of modeling lexicalization, whose importance has been emphasized (Galley and McKeown, 2007; Yamangil and Nelken, 2008). Third, we may have our STSG operate on trees instead of sentences, which allows for efficient parsing algorithms, as well as providing syntactic analyses for our predictions, which is desirable for automatic evaluation purposes.

A straightforward extension of the popular EM algorithm for probabilistic context free grammars (PCFG), the inside-outside algorithm (Lari and Young, 1990), can be used to estimate the rule weights of a given unweighted STSG based on a corpus of parallel parse trees $\mathbf{t} = t_1, \ldots, t_N$ where $t_n = t_{n,s}/t_{n,t}$ for $n = 1, \ldots, N$. Similarly, an

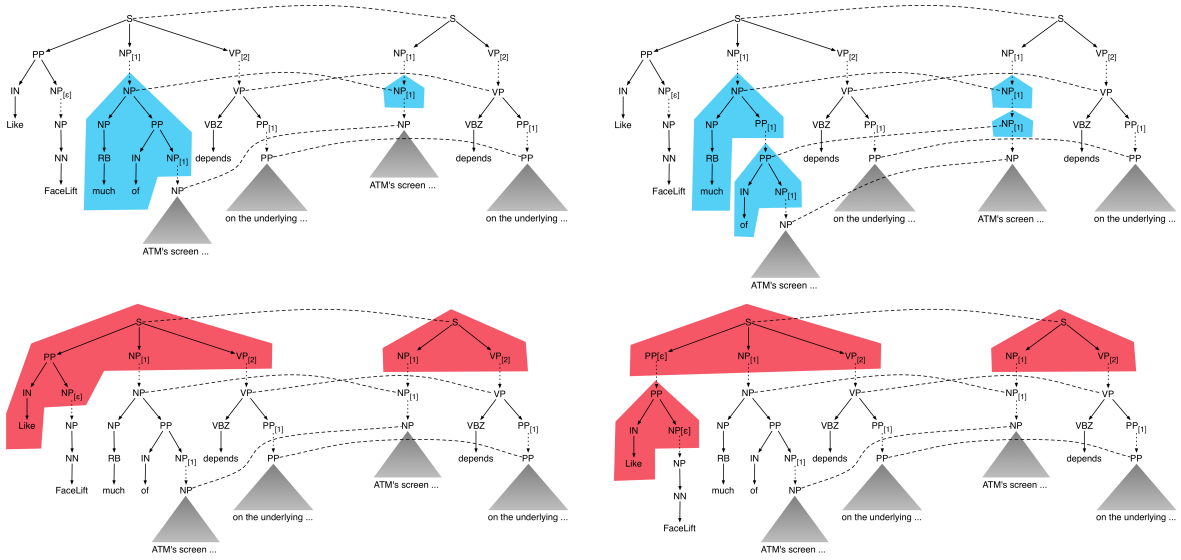


Figure 2: Gibbs sampling updates. We illustrate a sampler move to align/unalign a source node with a target node (top row in blue), and split/merge a deletion rule via aligning with ϵ (bottom row in red).

extension of the Viterbi algorithm is available for finding the maximum probability derivation, useful for predicting the target analysis $t_{N+1,t}$ for a test instance $t_{N+1,s}$. (Eisner, 2003) However, as noted earlier, EM is subject to the narrowness and overfitting problems.

3.1 The Bayesian generative process

Both of these issues can be addressed by taking a nonparametric Bayesian approach, namely, assuming that the elementary tree pairs are sampled from an independent collection of Dirichlet process (DP) priors. We describe such a process for sampling a corpus of tree pairs $\mathbf{t}$.

For all pairs of root labels $c = c_s/c_t$ that we consider, where up to one of c_s or c_t can be ϵ (e.g., S / S, NP / ϵ), we sample a sparse discrete distribution G_c over infinitely many elementary tree pairs $e = e_s/e_t$ sharing the common root c from a DP

$$G_c \sim \mathrm{DP}(\alpha_c, P_0(\cdot \mid c)) \quad (1)$$

where the DP has the concentration parameter α_c controlling the sparsity of G_c, and the base distribution $P_0(\cdot \mid c)$ is a distribution over novel elementary tree pairs that we describe more fully shortly.

We then sample a sequence of elementary tree pairs to serve as a derivation for each observed derived tree pair. For each $n = 1, \ldots, N$, we sample elementary tree pairs $e_n = e_{n,1}, \ldots, e_{n,d_n}$ in a derivation sequence (where d_n is the number of rules used in the derivation), consulting G_c whenever an elementary tree pair with root c is to be sampled.

$$e \overset{\text{iid}}{\sim} G_c, \quad \text{for all } e \text{ whose root label is } c$$

Given the derivation sequence e_n, a tree pair t_n is determined, that is,

$$p(t_n \mid e_n) = \begin{cases} 1 & e_{n,1}, \ldots, e_{n,d_n} \text{ derives } t_n \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The hyperparameters α_c can be incorporated into the generative model as random variables; however, we opt to fix these at various constants to investigate different levels of sparsity.

For the base distribution $P_0(\cdot \mid c)$ there are a variety of choices; we used the following simple scenario. (We take $c = c_s/c_t$.)

Synchronous rules For the case where neither c_s nor c_t are the special symbol ϵ, the base distribution first generates e_s and e_t independently, and then samples an alignment between the frontier nodes. Given a nonterminal, an elementary tree is generated by first making a decision to expand the nonterminal (with probability β_c) or to leave it as a frontier node ($1 - \beta_c$). If the decision to expand was made, we sample an appropriate rule from a PCFG which we estimate ahead

of time from the training corpus. We expand the nonterminal using this rule, and then repeat the same procedure for every child generated that is a nonterminal until there are no generated nonterminal children left. This is done independently for both e_s and e_t. Finally, we sample an alignment between the frontier nodes uniformly at random out of all possible alingments.

Deletion/insertion rules If $c_t = \epsilon$, that is, we have a deletion rule, we need to generate $e = e_s/\epsilon$. (The insertion rule case is symmetric.) The base distribution generates e_s using the same process described for synchronous rules above. Then with probability 1 we align all frontier nodes in e_s with ϵ. In essence, this process generates TSG rules, rather than STSG rules, which are used to cover deleted (or inserted) subtrees.

This simple base distribution does nothing to enforce an alignment between the internal nodes of e_s and e_t. One may come up with more sophisticated base distributions. However the main point of the base distribution is to encode a controllable preference towards simpler rules; we therefore make the simplest possible assumption.

3.2 Posterior inference via Gibbs sampling

Assuming fixed hyperparameters $\boldsymbol{\alpha} = \{\alpha_c\}$ and $\boldsymbol{\beta} = \{\beta_c\}$, our inference problem is to find the posterior distribution of the derivation sequences $\mathbf{e} = e_1, \ldots, e_N$ given the observations $\mathbf{t} = t_1, \ldots, t_N$. Applying Bayes' rule, we have

$$p(\mathbf{e} \mid \mathbf{t}) \quad \propto \quad p(\mathbf{t} \mid \mathbf{e}) p(\mathbf{e}) \tag{3}$$

where $p(\mathbf{t} \mid \mathbf{e})$ is a $0/1$ distribution (2) which does not depend on G_c, and $p(\mathbf{e})$ can be obtained by collapsing G_c for all c.

Consider repeatedly generating elementary tree pairs $e_1, \ldots, e_i$, all with the same root c, iid from G_c. Integrating over G_c, the e_i become dependent. The conditional prior of the i-th elementary tree pair given previously generated ones $e_{<i} = e_1, \ldots, e_{i-1}$ is given by

$$p(e_i \mid e_{<i}) \quad = \quad \frac{n_{e_i} + \alpha_c P_0(e_i \mid c)}{i - 1 + \alpha_c} \tag{4}$$

where n_{e_i} denotes the number of times e_i occurs in $e_{<i}$. Since the collapsed model is exchangeable in the e_i, this formula forms the backbone of the inference procedure that we describe next. It also makes clear DP's inductive bias to reuse elementary tree pairs.

We use Gibbs sampling (Geman and Geman, 1984), a Markov chain Monte Carlo (MCMC) method, to sample from the posterior (3). A derivation $\mathbf{e}$ of the corpus $\mathbf{t}$ is completely specified by an alignment between the source nodes and the corresponding target nodes (as well as ϵ on either side), which we take to be the state of the sampler. We start at a random derivation of the corpus, and at every iteration resample a derivation by amending the current one through local changes made at the node level, in the style of Goldwater et al. (2006).

Our sampling updates are extensions of those used by Cohn and Blunsom (2009) in MT, but are tailored to our task of extractive sentence compression. In our task, no target node can align with ϵ (which would indicate a subtree insertion), and barring unary branches no source node i can align with two different target nodes j and j' at the same time (indicating a tree expansion). Rather, the configurations of interest are those in which only source nodes i can align with ϵ, and two source nodes i and i' can align with the same target node j. Thus, the alignments of interest are not arbitrary relations, but (partial) functions from nodes in e_s to nodes in e_t or ϵ. We therefore sample in the direction from source to target. In particular, we visit every tree pair and each of its source nodes i, and update its alignment by selecting between and within two choices: (a) unaligned, (b) aligned with some target node j or ϵ. The number of possibilities j in (b) is significantly limited, firstly by the word alignment (for instance, a source node dominating a deleted subspan cannot be aligned with a target node), and secondly by the current alignment of other nearby aligned source nodes. (See Cohn and Blunsom (2009) for details of matching spans under tree constraints.)[2]

[2] One reviewer was concerned that since we explicitly disallow insertion rules in our sampling procedure, our model that generates such rules wastes probability mass and is therefore "deficient". However, we regard sampling as a separate step from the data generation process, in which we can formulate more effective algorithms by using our domain knowledge that our data set was created by annotators who were instructed to delete words only. Also, disallowing insertion rules in the base distribution unnecessarily complicates the definition of the model, whereas it is straightforward to define the joint distribution of all (potentially useful) rules and then use domain knowledge to constrain the support of that distribution during inference, as we do here. In fact, it is pos-

More formally, let e_M be the elementary tree pair rooted at the closest aligned ancestor i' of node i when it is unaligned; and let e_A and e_B be the elementary tree pairs rooted at i' and i respectively when i is aligned with some target node j or ϵ. Then, by exchangeability of the elementary trees sharing the same root label, and using (4), we have

$$p(\text{unalign}) = \frac{n_{e_M} + \alpha_{c_M} P_0(e_M \mid c_M)}{n_{c_M} + \alpha_{c_M}} \quad (5)$$

$$p(\text{align with } j) = \frac{n_{e_A} + \alpha_{c_A} P_0(e_A \mid c_A)}{n_{c_A} + \alpha_{c_A}} \quad (6)$$

$$\times \frac{n_{e_B} + \alpha_{c_B} P_0(e_B \mid c_B)}{n_{c_B} + \alpha_{c_B}} \quad (7)$$

where the counts $n_{e.}, n_{c.}$ are with respect to the current derivation of the rest of the corpus; except for n_{e_B}, n_{c_B} we also make sure to account for having generated e_A. See Figure 2 for an illustration of the sampling updates.

It is important to note that the sampler described can move from any derivation to any other derivation with positive probability (if only, for example, by virtue of fully merging and then resegmenting), which guarantees convergence to the posterior (3). However some of these transition probabilities can be extremely small due to passing through low probability states with large elementary trees; in turn, the sampling procedure is prone to local modes. In order to counteract this and to improve mixing we used simulated annealing. The probability mass function (5-7) was raised to the power $1/T$ with T dropping linearly from $T = 5$ to $T = 0$. Furthermore, using a final temperature of zero, we recover a maximum a posteriori (MAP) estimate which we denote $\mathbf{e}_{\text{MAP}}$.

3.3 Prediction

We discuss the problem of predicting a target tree $t_{N+1,t}$ that corresponds to a source tree $t_{N+1,s}$ unseen in the observed corpus $\mathbf{t}$. The maximum probability tree (MPT) can be found by considering all possible ways to derive it. However a much simpler alternative is to choose the target tree implied by the maximum probability derivation (MPD), which we define as

$$\begin{aligned} e^* &= \operatorname*{argmax}_e p(e \mid t_s, \mathbf{t}) \\ &= \operatorname*{argmax}_e \sum_{\mathbf{e}} p(e \mid t_s, \mathbf{e}) p(\mathbf{e} \mid \mathbf{t}) \end{aligned}$$

where e denotes a derivation for $t = t_s/t_t$. (We suppress the $N+1$ subscripts for brevity.) We approximate this objective first by substituting $\delta_{\mathbf{e}_{\text{MAP}}}(\mathbf{e})$ for $p(\mathbf{e} \mid \mathbf{t})$ and secondly using a finite STSG model for the infinite $p(e \mid t_s, \mathbf{e}_{\text{MAP}})$, which we obtain simply by normalizing the rule counts in $\mathbf{e}_{\text{MAP}}$. We use dynamic programming for parsing under this finite model (Eisner, 2003).[3]

Unfortunately, this approach does not ensure that the test instances are parsable, since t_s may include unseen structure or novel words. A workaround is to include all zero-count context free copy rules such as

NP / NP → (NP NP$_{[1]}$ PP$_{[2]}$) / (NP NP$_{[1]}$ PP$_{[2]}$)
NP / ϵ → (NP NP$_{[\epsilon]}$ PP$_{[\epsilon]}$) / ϵ

in order to smooth our finite model. We used Laplace smoothing (adding 1 to all counts) as it gave us interpretable results.

4 Evaluation

We compared the Gibbs sampling compressor (GS) against a version of maximum a posteriori EM (with Dirichlet parameter greater than 1) and a discriminative STSG based on SVM training (Cohn and Lapata, 2008) (SVM). EM is a natural benchmark, while SVM is also appropriate since it can be taken as the state of the art for our task.[4]

We used a publicly available extractive sentence compression corpus: the Broadcast News compressions corpus (BNC) of Clarke and Lapata (2006a). This corpus consists of 1370 sentence pairs that were manually created from transcribed Broadcast News stories. We split the pairs into training, development, and testing sets of 1000,

sible to prove that our approach is equivalent up to a rescaling of the concentration parameters. Since we fit these parameters to the data, our approach is equivalent.

[3] We experimented with MPT using Monte Carlo integration over possible derivations; the results were not significantly different from those using MPD.

[4] The comparison system described by Cohn and Lapata (2008) attempts to solve a more general problem than ours, abstractive sentence compression. However, given the nature of the data that we provided, it can only learn to compress by deleting words. Since the system is less specialized to the task, their model requires additional heuristics in decoding not needed for extractive compression, which might cause a reduction in performance. Nonetheless, because the comparison system is a generalization of the extractive SVM compressor of Cohn and Lapata (2007), we do not expect that the results would differ qualitatively.

	SVM	EM	GS
Precision	55.60	58.80	58.94
Recall	53.37	56.58	64.59
Relational F1	54.46	57.67	61.64
Compression rate	59.72	64.11	65.52

Table 1: Precision, recall, relational F1 and compression rate (%) for various systems on the 200-sentence BNC test set. The compression rate for the gold standard was 65.67%.

	SVM	EM	GS	Gold
Grammar	$2.75^{\dagger}$	2.85^{*}	3.69	4.25
Importance	2.85	2.67^{*}	3.41	3.82
Comp. rate	68.18	64.07	67.97	62.34

Table 2: Average grammar and importance scores for various systems on the 20-sentence subsample. Scores marked with $*$ are significantly different than the corresponding GS score at $\alpha < .05$ and with † at $\alpha < .01$ according to post-hoc Tukey tests. ANOVA was significant at $p < .01$ both for grammar and importance.

170, and 200 pairs, respectively. The corpus was parsed using the Stanford parser (Klein and Manning, 2003).

In our experiments with the publicly available SVM system we used all except paraphrasal rules extracted from bilingual corpora (Cohn and Lapata, 2008). The model chosen for testing had parameter for trade-off between training error and margin set to $C = 0.001$, used margin rescaling, and Hamming distance over bags of tokens with brevity penalty for loss function. EM used a subset of the rules extracted by SVM, namely all rules except non-head deleting compression rules, and was initialized uniformly. Each EM instance was characterized by two parameters: α, the smoothing parameter for MAP-EM, and δ, the smoothing parameter for augmenting the learned grammar with rules extracted from unseen data (add-$(\delta - 1)$ smoothing was used), both of which were fit to the development set using grid-search over $(1, 2]$. The model chosen for testing was $(\alpha, \delta) = (1.0001, 1.01)$.

GS was initialized at a random derivation. We sampled the alignments of the source nodes in random order. The sampler was run for 5000 iterations with annealing. All hyperparameters α_c, β_c were held constant at α, β for simplicity and were fit using grid-search over $\alpha \in [10^{-6}, 10^{6}], \beta \in [10^{-3}, 0.5]$. The model chosen for testing was $(\alpha, \beta) = (100, 0.1)$.

As an automated metric of quality, we compute F-score based on grammatical relations (relational F1, or RelF1) (Riezler et al., 2003), by which the consistency between the set of predicted grammatical relations and those from the gold standard is measured, which has been shown by Clarke and Lapata (2006b) to correlate reliably with human judgments. We also conducted a small human subjective evaluation of the grammaticality and informativeness of the compressions generated by the various methods.

4.1 Automated evaluation

For all three systems we obtained predictions for the test set and used the Stanford parser to extract grammatical relations from predicted trees and the gold standard. We computed precision, recall, RelF1 (all based on grammatical relations), and compression rate (percentage of the words that are *retained*), which we report in Table 1. The results for GS are averages over five independent runs. EM gives a strong baseline since it already uses rules that are limited in depth and number of frontier nodes by stipulation, helping with the overfitting we have mentioned, surprisingly outperforming its discriminative counterpart in both precision and recall (and consequently RelF1). GS however maintains the same level of precision as EM while improving recall, bringing an overall improvement in RelF1.

4.2 Human evaluation

We randomly subsampled our 200-sentence test set for 20 sentences to be evaluated by human judges through Amazon Mechanical Turk. We asked 15 self-reported native English speakers for their judgments of GS, EM, and SVM output sentences and the gold standard in terms of *grammaticality* (how fluent the compression is) and *importance* (how much of the meaning of and important information from the original sentence is retained) on a scale of 1 (worst) to 5 (best). We report in Table 2 the average scores. EM and SVM perform at very similar levels, which we attribute to using the same set of rules, while GS performs at a level substantially better than both, and much closer to human performance in both criteria. The

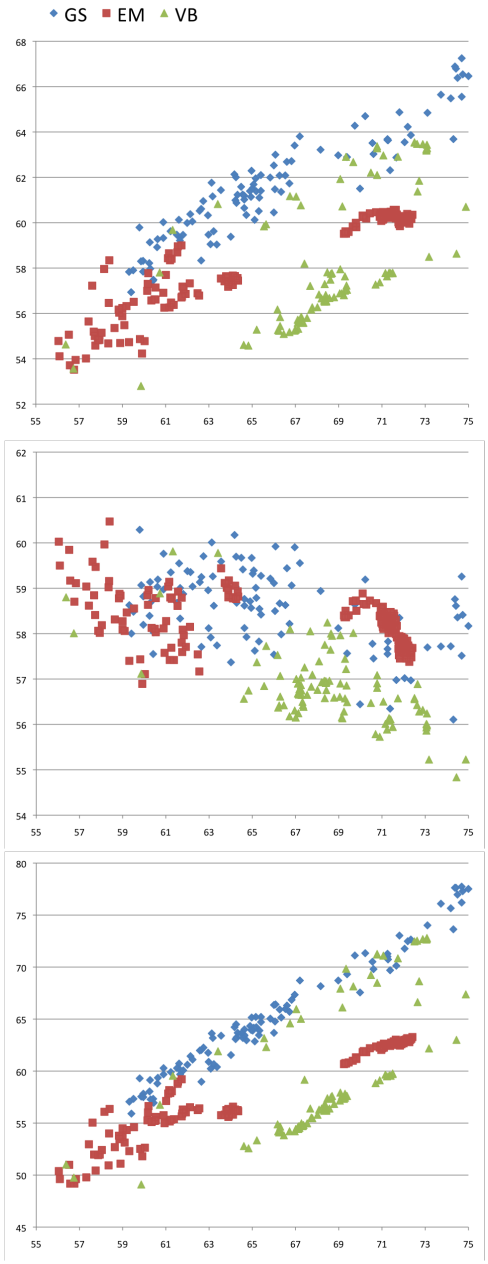


Figure 3: RelF1, precision, recall plotted against compression rate for GS, EM, VB.

human evaluation indicates that the superiority of the Bayesian nonparametric method is underappreciated by the automated evaluation metric.

4.3 Discussion

The fact that GS performs better than EM can be attributed to two reasons: (1) GS uses a sparse prior and selects a compact representation of the data (grammar sizes ranged from 4K-7K for GS compared to a grammar of about 35K rules for EM). (2) GS does not commit to a precomputed grammar and searches over the space of all grammars to find one that bests represents the corpus. It is possible to introduce DP-like sparsity in EM using variational Bayes (VB) training. We experiment with this next in order to understand how dominant the two factors are. The VB algorithm requires a simple update to the M-step formulas for EM where the expected rule counts are normalized, such that instead of updating the rule weight in the t-th iteration as in the following

$$\theta_{c,e}^{t+1} = \frac{n_{c,e} + \alpha - 1}{n_{c,\cdot} + K\alpha - K}$$

where $n_{c,e}$ represents the expected count of rule $c \rightarrow e$, and K is the total number of ways to rewrite c, we now take into account our DP$(\alpha_c, P_0(\cdot \mid c))$ prior in (1), which, when truncated to a finite grammar, reduces to a K-dimensional Dirichlet prior with parameter $\alpha_c P_0(\cdot \mid c)$. Thus in VB we perform a *variational* E-step with the subprobabilities given by

$$\theta_{c,e}^{t+1} = \frac{\exp\left(\Psi(n_{c,e} + \alpha_c P_0(e \mid c))\right)}{\exp\left(\Psi(n_{c,\cdot} + \alpha_c)\right)}$$

where Ψ denotes the digamma function. (Liu and Gildea, 2009) (See MacKay (1997) for details.) Hyperparameters were handled the same way as for GS.

Instead of selecting a single model on the development set, here we provide the whole spectrum of models and their performances in order to better understand their comparative behavior. In Figure 3 we plot RelF1 on the test set versus compression rate and compare GS, EM, and VB ($\beta = 0.1$ fixed, (α, δ) ranging in $[10^{-6}, 10^{6}] \times (1, 2]$). Overall, we see that GS maintains roughly the same level of precision as EM (despite its larger compression rates) while achieving an improvement in recall, consequently performing at a higher RelF1 level. We note that VB somewhat bridges the gap between GS and EM, without quite reaching GS performance. We conclude that the mitigation of the two factors (narrowness and overfitting) both contribute to the performance gain of GS.[5]

4.4 Example rules learned

In order to provide some insight into the grammar extracted by GS, we list in Tables (3) and (4) high

[5]We have also experimented with VB with parametric independent symmetric Dirichlet priors. The results were similar to EM with the exception of sparse priors resulting in smaller grammars and slightly improving performance.

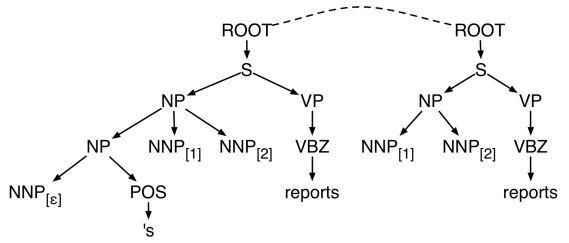
(ROOT (S CC$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S NP$_{[1]}$ ADVP$_{[\epsilon]}$ VP$_{[2]}$ (. .)))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$ (. .)))
(ROOT (S ADVP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .)))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$ (. .)))
(ROOT (S PP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .)))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$ (. .)))
(ROOT (S PP$_{[\epsilon]}$,$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S NP$_{[\epsilon]}$ (VP VBP$_{[\epsilon]}$ (SBAR (S NP$_{[1]}$ VP$_{[2]}$))) .$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S ADVP$_{[\epsilon]}$ NP$_{[1]}$ (VP MD$_{[2]}$ VP$_{[3]}$) .$_{[4]}$))	/ (ROOT (S NP$_{[1]}$ (VP MD$_{[2]}$ VP$_{[3]}$) .$_{[4]}$))
(ROOT (S (SBAR (IN as) S$_{[\epsilon]}$) ,$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S S$_{[\epsilon]}$ (, ,) CC$_{[\epsilon]}$ (S NP$_{[1]}$ VP$_{[2]}$) .$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S PP$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S S$_{[1]}$ (, ,) CC$_{[\epsilon]}$ S$_{[2]}$ (. .)))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$ (. .)))
(ROOT (S S$_{[\epsilon]}$,$_{[\epsilon]}$ NP$_{[1]}$ ADVP$_{[2]}$ VP$_{[3]}$.$_{[4]}$))	/ (ROOT (S NP$_{[1]}$ ADVP$_{[2]}$ VP$_{[3]}$.$_{[4]}$))
(ROOT (S (NP (NP NNP$_{[\epsilon]}$ (POS 's)) NNP$_{[1]}$ NNP$_{[2]}$) (VP (VBZ reports)) .$_{[3]}$))	/ (ROOT (S (NP NNP$_{[1]}$ NNP$_{[2]}$) (VP (VBZ reports)) .$_{[3]}$))

Table 3: High probability ROOT / ROOT compression rules from the final state of the sampler.

(S NP$_{[1]}$ ADVP$_{[\epsilon]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S INTJ$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S (INTJ (UH Well)) ,$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S PP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S ADVP$_{[\epsilon]}$ (, ,) S$_{[1]}$ (, ,) (CC but) S$_{[2]}$.$_{[3]}$)	/ (S S$_{[1]}$ (, ,) (CC but) S$_{[2]}$.$_{[3]}$)
(S ADVP$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S NP$_{[\epsilon]}$ (VP VBP$_{[\epsilon]}$ (SBAR (IN that) (S NP$_{[1]}$ VP$_{[2]}$))) (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S NP$_{[\epsilon]}$ (VP VBZ$_{[\epsilon]}$ ADJP$_{[\epsilon]}$ SBAR$_{[1]}$))	/ S$_{[1]}$
(S CC$_{[\epsilon]}$ PP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S NP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S NP$_{[1]}$ (, ,) ADVP$_{[\epsilon]}$ (, ,) VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S CC$_{[\epsilon]}$ (NP PRP$_{[1]}$) VP$_{[2]}$)	/ (S (NP PRP$_{[1]}$) VP$_{[2]}$)
(S ADVP$_{[\epsilon]}$,$_{[\epsilon]}$ PP$_{[\epsilon]}$,$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S ADVP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)

Table 4: High probability S / S compression rules from the final state of the sampler.

probability subtree-deletion rules expanding categories ROOT / ROOT and S / S, respectively. Of especial interest are deep lexicalized rules such as



a pattern of compression used many times in the BNC in sentence pairs such as "NPR's Anne Garrels reports" / "Anne Garrels reports". Such an informative rule with nontrivial collocation (between the possessive marker and the word "reports") would be hard to extract heuristically and can only be extracted by reasoning across the training examples.

5 Conclusion

We explored nonparametric Bayesian learning of non-isomorphic tree mappings using Dirichlet process priors. We used the task of extractive sentence compression as a testbed to investigate the effects of sparse priors and nonparametric inference over the space of grammars. We showed that, despite its degeneracy, expectation maximization is a strong baseline when given a reasonable grammar. However, Gibbs-sampling–based nonparametric inference achieves improvements against this baseline. Our investigation with variational Bayes showed that the improvement is due both to finding sparse grammars (mitigating overfitting) and to searching over the space of all grammars (mitigating narrowness). Overall, we take these results as being encouraging for STSG induction via Bayesian nonparametrics for monolingual translation tasks. The future for this work would involve natural extensions such as mixing over the space of word alignments; this would allow application to MT-like tasks where flexible word reordering is allowed, such as abstractive sentence compression and paraphrasing.

References

James Clarke and Mirella Lapata. 2006a. Constraint-based sentence compression: An integer programming approach. In *Proceedings of the 21st Interna-*

tional Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 144–151, Sydney, Australia, July. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2006b. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 377–384, Sydney, Australia, July. Association for Computational Linguistics.

Trevor Cohn and Phil Blunsom. 2009. A Bayesian model of syntax-directed tree to string grammar induction. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 352–361, Morristown, NJ, USA. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning*, pages 73–82, Prague. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 137–144, Manchester, United Kingdom. Association for Computational Linguistics.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Morristown, NJ, USA. Association for Computational Linguistics.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 (Series B):1–38.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *StatMT '06: Proceedings of the Workshop on Statistical Machine Translation*, pages 31–38, Morristown, NJ, USA. Association for Computational Linguistics.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 314–323, Morristown, NJ, USA. Association for Computational Linguistics.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 205–208, Morristown, NJ, USA. Association for Computational Linguistics.

Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 180–187, Rochester, New York, April. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

S. Geman and D. Geman. 1984. Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images. pages 6:721–741.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia, July. Association for Computational Linguistics.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315, Morristown, NJ, USA. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS*, pages 3–10. MIT Press.

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

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

Ding Liu and Daniel Gildea. 2009. Bayesian learning of phrasal tree-to-string templates. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1308–1317, Singapore, August. Association for Computational Linguistics.

David J.C. MacKay. 1997. Ensemble learning for hidden markov models. Technical report, Cavendish Laboratory, Cambridge, UK.

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

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48, Suntec, Singapore, August. Association for Computational Linguistics.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 118–125, Morristown, NJ, USA. Association for Computational Linguistics.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 290–297, Morristown, NJ, USA. Association for Computational Linguistics.

Elif Yamangil and Rani Nelken. 2008. Mining wikipedia revision histories for improving sentence compression. In *Proceedings of ACL-08: HLT, Short Papers*, pages 137–140, Columbus, Ohio, June. Association for Computational Linguistics.