Investigation of Compound Probabilistic Context-Free Grammars and their Extensions using Semantic Knowledge and Updating Priors

Alex Li
alexjli@mit.edu

Tong Zhao
tzhao@mit.edu

Abstract

Grammar induction is a field that many researchers are interested, ranging from linguists to those studying artificial intelligence. Recently, Kim et al. demonstrated a neural-parameterized Compound Probabalistic Context-Free Grammar (CPCFG) that was able learn English syntax structure with better than state-of-the-art performance. In this paper we investigate the parameter space of CPCFGs, including models with varying priors and variational posteriors. We also extend the model, among which include SS-CPCFGs (Semantic-Sensitive CPCFGs), AP-CPCFGs (Adaptive-Prior CPCFGs), and their combination SSAP-CPCFGs. These models were evaluated on PTB-10 to see their performance in learning the syntactic structure of English. We found that, while most models fail to perform as well as Kim et al.'s vanilla model, models that swap either the prior or the variational posterior with Laplace distributions tend to outperform their Gaussian counterparts. SSAP-CPCFGs also seem to have better performance than the vanilla model.

Introduction

Languages have words. A language's syntax is its system and associated rules by which it constructs sentences from words. Knowledge of a language's syntactic structure is powerful, as it empowers greater control over the languages such as allowing for the generation of novel sentences. As such, discovering a language's syntax structure is useful to both linguists and those studying natural language processing. However, the current process of discovering syntax is very difficult. Linguists must spend thousands of hours collecting and processing data manually, and the top NLP models in the field require costly labeled data. Hence, many scientists are interested in the field of grammar induction, the unsupervised learning of syntactic structure based on raw text, which is readily available via sources such as written text and the Internet.

The best methods of grammar induction are based on statistical methods. These methods require probabilistic grammars that are learned by optimization of the appropriate loss function.

Early work found that the induction of probabilistic context-free grammars (PCFGs) from linguistic data problematic (Lari & Young, 1990; Carroll & Charniak, 1992). This has been attributed by some to the overly strict independence assumptions of PCFGs and the difficult optimization landscape (Kim et al., 2019). More recent work has gotten around these problems with auxiliary objectives (Klein & Manning, 2002), priors and non-parametric models (Kenichi Kurihara and Taisuke Sato, 2006; Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater, 2007; Percy Liang & Klein, 2007; Wang & Blunsom, 2013), and manually set features (Yun Huang & Tan, 2012; Dave Golland & Uszkoreit, 2012).

Very recently, Kim et al. demonstrated that their compound PCFG model could be used to great success in grammar induction (Kim et al., 2019). In this model, they employ a continuous mixture of PCFGs with a novel "neural" parameterization.

In this project, we explore Kim's compound PCFG and experiment with changes to it. We experiment with different priors and variational posteriors, as well as augment the model with adaptive priors and semantic sensitivity.

PCFGs

First, we start with context free grammars. We will consider a context free grammar (CFG) consisting of a 5-tuple $\mathcal{G} = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R})$, where S is the start symbol, $\mathcal{N}$ is the set of nonterminals, $\mathcal{P}$ is the set of preterminals, Σ is the set of terminals, and $\mathcal{R}$ is the set of rules of the form

$$\begin{aligned} S &\to A, & A &\in \mathcal{N} \\ A &\to B\ C, & A \in \mathcal{N};\ B, C &\in \mathcal{N} \cup \mathcal{P} \\ T &\to w, & T \in \mathcal{P};\ w &\in \Sigma. \end{aligned}$$

A PCFG consists of a grammar $\mathcal{G}$ along with a set of rule probabilities $\boldsymbol{\pi} = \{\pi_r\}_{r \in \mathcal{R}}$ such that π_r is the probability of rule r. Let $\mathcal{T}_{\mathcal{G}}$ denote the set of all parse trees over $\mathcal{G}$. Given a parse tree $\boldsymbol{t} \in \mathcal{T}_{\mathcal{G}}$, a PCFG defines a distribution $p_{\boldsymbol{\pi}}(\boldsymbol{t})$ as

$$p_{\boldsymbol{\pi}}(\boldsymbol{t}) = \prod_{r \in \mathcal{R}_{\boldsymbol{t}}} \pi_r,$$

where $\mathcal{R}_{\boldsymbol{t}} \subseteq \mathcal{R}$ is the set of rules used in $\boldsymbol{t}$. Similarly, given a string of terminals $\boldsymbol{x} \in \Sigma^{|\boldsymbol{x}|}$ where $|\boldsymbol{x}|$ denotes the length of $\boldsymbol{x}$, let $\mathcal{T}_{\mathcal{G}}(\boldsymbol{x}) = \{\boldsymbol{t} \mid \text{yield}(\boldsymbol{t}) = \boldsymbol{x}\}$ be the set of trees $\boldsymbol{t}$ whose leaves are $\boldsymbol{x}$. A PCFG defines a distribution $p_{\boldsymbol{\pi}}(\boldsymbol{x})$ as

$$p_{\boldsymbol{\pi}}(\boldsymbol{x}) = \sum_{t \in \mathcal{T}_{\mathcal{G}}(x)} p_{\boldsymbol{\pi}}(t).$$

This allows us to give the quantity

$$p_{\boldsymbol{\pi}}(\boldsymbol{t} \mid \boldsymbol{x}) = \frac{\mathbb{1}[\text{yield}(\boldsymbol{t} = \boldsymbol{x})] p_{\boldsymbol{\pi}}(\boldsymbol{t})}{p_{\boldsymbol{\pi}}(\boldsymbol{x})}$$

as an expression for the posterior distribution of unobserved latent trees given an observation of sentence $\boldsymbol{x}$, where $\mathbb{1}$ represents the indicator function.

Parameterization in PCFGs

A naive parameterization of a PCFG is to associate scalars for each rule π_r such that the scalars form a valid probability distribution. However, previous work has shown that learning meaningful grammars from natural language data is difficult with this parameterization (Carroll & Charniak, 1992). Thus, Kim et al.'s compound PCFG model uses a so-called "neural parameterization" with rule probabilities based on distributed representations.

For each $N \in \{S\} \cup \mathcal{N} \cup \mathcal{P}$ (a symbol that appears on the left side of a rule), we associate an input embedding $\boldsymbol{w}_N$; for each $N \in \mathcal{N} \cup \mathcal{P} \cup \Sigma$ (a symbol that appears on the right side of a rule), we associate an output embedding $\boldsymbol{u}_N$. Let $\mathcal{M}$ denote the product space $\mathcal{N} \cup \mathcal{P} \times \mathcal{N} \cup \mathcal{P}$, and let f_1 and f_2 be instances of neural network f_λ parameterized by parameters λ; we parameterize π_r for each rule type r:

$$\pi_{S\to A} = \frac{\exp(\boldsymbol{u}_A^T f_1(\boldsymbol{w}_S))}{\sum_{A'\in\mathcal{N}} \exp(\boldsymbol{u}_{A'}^T f_1(\boldsymbol{w}_S))},$$
$$\pi_{A\to BC} = \frac{\exp(\boldsymbol{u}_{BC}^T \boldsymbol{w}_A)}{\sum_{B'C'\in\mathcal{M}} \exp(\boldsymbol{u}_{B'C'}^T \boldsymbol{w}_A)},$$
$$\pi_{T\to w} = \frac{\exp(\boldsymbol{u}_w^T f_2(\boldsymbol{w}_T))}{\sum_{w'\in\Sigma} \exp(\boldsymbol{u}_{w'}^T f_2(\boldsymbol{w}_T))}.$$

Let $\boldsymbol{E}_\mathcal{G} = \{\boldsymbol{w}_N \mid N \in \{S\} \cup \mathcal{N} \cup \mathcal{P}\}$ denote the global set of symbol embeddings for the grammar $\mathcal{G}$.

Compound PCFG

This section follows an explanation of Kim et al.'s compound PCFG model, for a more detailed treatment and analysis see Kim et al. (2019).

A compound probability distribution is a distribution whose parameters are also drawn from a distribution. These distributions allow the modeling of complex generative processes. The compound PCFG model uses a generative process to leverage a compound rule probability distribution for the PCFG.

Let $p_\gamma(\boldsymbol{z})$ be a prior with parameters γ. The rule probabilities are obtained via

$$\boldsymbol{z} \sim p_\gamma(\boldsymbol{z}), \qquad \boldsymbol{\pi_z} = f_\lambda(\boldsymbol{z}, \boldsymbol{E}_\mathcal{G}),$$

where f_λ denotes a neural network with parameters λ which concatenates input embeddings with $\boldsymbol{z}$ and outputs rule probabilities $\boldsymbol{\pi_z}$,

$$\pi_{S\to A} \propto \exp(\boldsymbol{u}_A^T f_1([\boldsymbol{w}_S;\boldsymbol{z}])),$$
$$\pi_{A\to BC} \propto \exp(\boldsymbol{u}_{BC}^T [\boldsymbol{w}_A;\boldsymbol{z}]),$$
$$\pi_{T\to w} \propto \exp(\boldsymbol{u}_w^T f_2([\boldsymbol{w}_T;\boldsymbol{z}])),$$

where $[;]$ denotes vector concatenation. The tree is then sampled from the PCFG with probabilities given by $\boldsymbol{\pi_z}$, and the sentence is yielded from the tree:

$$\boldsymbol{t} \sim \text{PCFG}(\boldsymbol{\pi_z}), \qquad \boldsymbol{x} = \text{yield}(\boldsymbol{t}).$$

It is important to note that the compound PCFG can be viewed as Bayesian PCFG with a prior on sentence-level rule probabilities parameterized by $\boldsymbol{z}, \lambda, \boldsymbol{E}_\mathcal{G}$. Context-free assumptions hold conditioned on $\boldsymbol{z}$, but not in general. For example, two rules $r_1, r_2 \in \mathcal{R}$ are independent conditioned on $\boldsymbol{z}$, but not independent otherwise, as there exists a dependence path through $\boldsymbol{z}$.

Inference in Compound PCFGs

We use an inference network to generate parameters for a variational posterior. Our inference network first embeds sentence $\boldsymbol{x}$, then feeds this embedded sentence through a bidirectional LSTM, followed by an affine layer over the max-pooled hidden states. This generates mean and log-variance vectors for our diagonal Gaussian variational posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$.

The evidence lower bound ELBO for the entire model is given by

$$\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})] - \text{KL}[q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \mid\mid p_\gamma(\boldsymbol{z})]$$
$$= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \sum_{\boldsymbol{t}\in\mathcal{T}_\mathcal{G}(\boldsymbol{x})} p_\theta(\boldsymbol{t} \mid \boldsymbol{z})\right] - \text{KL}[q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \mid\mid p_\gamma(\boldsymbol{z})],$$

where we marginalize over $\mathcal{T}_\mathcal{G}(\boldsymbol{x})$ for tractability in calculation. We can find low-variance estimators for $\nabla_{\theta,\phi}\text{ELBO}(\theta,\phi;x)$ by using the reparameterization trick on the expected reconstruction likelihood term and the analytical expression for the KL term (Kingma & Welling, 2014)

Note that under the Bayesian PCFG view, since we estimate the parameters of the prior from the data, our approach is an instance of empirical Bayes (Robbins, 1956).

Experimental Detail

In the following sections, we discuss our exploration of the Compound PCFG model.

Varying Priors and Posteriors

The Compound PCFG has two big Bayesian components: a prespecified prior $p_\gamma(\boldsymbol{z})$ and an estimated variational posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$. How does changing the nature of the prior and the posterior change how the model performs? We investigated moving the prior location and changing the distribution family of the prior and the variational posterior.

In the following discussion, we will be referring to the mean and variance of the prior in scalar form (e.g. "the mean of the prior was set to 1"). It is important to note that in actuality the mean and variance of the prior are vectors with the same dimensions of the latent $\boldsymbol{z}$ vector; however, these vectors have all their elements set to the same scalar value. For ease of discussion, we simply describe this vector by the scalar value it's composed of.

Prior Location and Scale

First we consider the parameters of the prior, which is initially Gaussian with mean 0 and variance 1. What if this isn't representative of the correct prior on $\boldsymbol{z}$? Perhaps the prior more apt if located away from the origin, or has a different scale. Hence, we ran two sets of experiments consisting of changing the location and scale of the prior. In the mean experiments, the mean was varied from -2 to 2 in integer increments. In the scale experiment, the scale was varied from 0.2 to 1.0 in increments of 0.2 and 1.0 to 3.0 in increments of 1.

Varying Probability Distribution Families

Next we consider the distributional families the prior and variational posterior are from. Perhaps Gaussian is not the best way to describe the prior and variational posteriors for z? Hence, we varied the distribution of the prior and variational posteriors to see their effects on performance of the Compound PCFG. Unfortunately, because of the nature of the model's ELBO, we are limited to functions with closed forms of KL divergence.

Firstly, we changed the family of the variational posterior while keeping the prior a Gaussian of mean 0 and variance 1. This limited us to only testing the Gumbel and Laplace distribution families.

Secondly, we change both the prior and variational posterior to be from the same family. In this case, we set both to be either Gumbel distributions, Laplace distributions, or Log-Normal distributions.

Adaptive-Prior Compound PCFG

We did a lot of manually testing of priors. What if the Compound PCFG can update its prior as it learns? If we take the view that z represents the latent structure of a sentence, then updating the prior as the Compound PCFG sees sentences could function similar to learning more information about the inherent structure of English. This was implemented by changing our prior's mean and scale values to be parameters rather than constants. Since the KL divergence of all prior/variational posterior pairs has a closed form, we are able to propagate the gradients from the ELBO back to the prior, hence causing the prior to update every backprop session. Hence, we call this the Adaptive-Prior Compound PCFG (AP-CPCFG).

One question to consider is what to initialize the scale vector to. Although the model uses Xavier Uniform initialization, our intuition suggests that having small variances around 0 would make it difficult for the model to begin gradient decent. Hence, we tested the difference between initializing the scale vector to all 1s compared to the standard Xavier Uniform initialization.

We also wanted to test the difference between distribution families for the prior/variational posterior using this adaptive-prior model. We tested the AP-CPCFG using Gumbel, Laplace, Log-Normal, and Normal distributions for the prior and variational posterior.

Semantic-Sensitive Compound PCFGs

Being in the era of powerful neuronal language modelling architectures like ELMo, BERT, and XLNet, we wondered if we could use these softwares to our advantage to produce better syntax parse trees. Our intuition was that the semantics of a sentence can be useful for parsing the syntax of the sentence, so perhaps feeding the model semantic information will make it better at generating parse trees. We decided to investigate BERT extensions of our model, given the availability of a pre-trained BERT model through the `transformers` Python package (Jacob Devlin & Toutanova, 2018). We decided to augment our model in three ways: through the variational LSTM, through the rule-embedding network, and a combination of both.

The first model, which we call BERT1, involves replacing the embedding layer of the variational inference network with BERT-preprocessed versions of the original sentence. The input dimentionality of the variational LSTM is adjusted accordingly to accept a sequence of 768-dimensional vectors (the dimensionality of the output of BERT, where each vector represents a word).

The second model, which we call BERT2, involves appending BERT-preprocessed versions of the original sentence to the input of the rule-probability neural nets.

Let $\mathcal{B}$ be the last hidden state of obtained by running a bidirectional LSTM on the hidden states of BERT. BERT2 calculates rule probabilities by concatenating $\mathcal{B}$ to z and adjusting the dimensionality of the neural networks / embeddings accordingly:

$$\begin{aligned}
\pi_{S\to A} &\propto \exp(\boldsymbol{u}_A^T f_1^*([\boldsymbol{w}_S; z; \mathcal{B}])), \\
\pi_{A\to BC} &\propto \exp(\boldsymbol{u}_{BC}^{T*}[\boldsymbol{w}_A; z; \mathcal{B}]), \\
\pi_{T\to w} &\propto \exp(\boldsymbol{u}_w^T f_2^*([\boldsymbol{w}_T; z; \mathcal{B}])).
\end{aligned}$$

The final model, aptly called BERT3, is a combination of BERT1 and BERT2. BERT-preprocessed sentences are both fed through the variational LSTM and fed into the rule-embedding neural network via a separate LSTM.

Combining Semantic Sensitivity and Adaptive Priors: SSAP-CPCFG

Intuitively, it seems that extending the model both via semantic sensitivity and an adaptive prior would make the model more powerful than either extension alone. Hence, we created the Semantic-Sensitive and Adaptive Prior CPCFG (SSAP-CPCFG). The variants SSAP1, SSAP2, SSAP3 parallel BERT1, BERT2, and BERT3, respectively, but with an adaptive prior built in. [1]

Experimental Setup

Experiments were run on the BCS OpenMind High Performance Computing Cluster. Models were parallelized over 4 GPUs with at least 11 GB of GPU-RAM and at least 10GB of CPU-RAM per GPU.

[1]This naming convention is not the same as the raw names during data collection. See Appendix for more details

Data

The main dataset used in these experiment is a trimmed version of the Penn Tree Bank (PTB) PTB-10, a version of the PTB which only contains sentences with 10 or fewer words. Train/validation/test splits were obtained from Tristan Thrush.

Hyperparameters

Models were run with batch sizes of 100 split among 4 GPUs. Outside of modifications, hyperparameter settings were kept the same as in Kim et al. (see "Hyperparameters" under 4: Experimental Setup of Kim et al. (2019)).

Baselines and Evaluation

Evaluation of a model was done training 4 copies of the model and evaluating each model on the test set. The Sentence F1 of each was computed, and the average, max, and standard deviation of the F1s are calculated.

Baselines were obtained by running Kim et al. (2019) on the PTB-10 dataset in quadruplicate and performing the same analysis as described above.

Results and Analysis

Prior Location

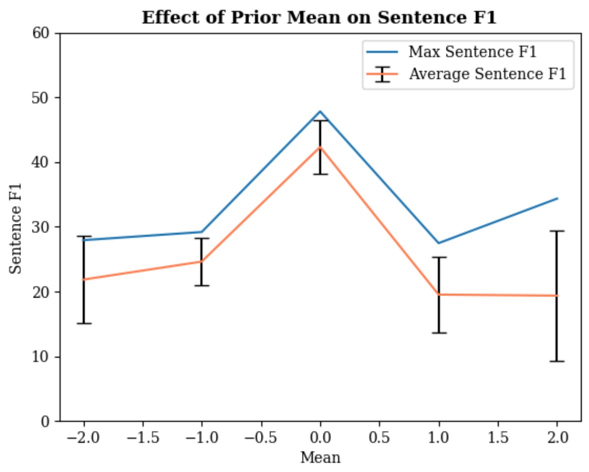

Figure 1: Effect of changing the mean of the prior on the result model's sentence F1. Prior is a Gaussian with standard deviation 1.

As can be seen in Figure 1, we see that the best mean for a Gaussian prior is around 0. Perhaps the best value is slightly negative, as we can see that the negative means give higher sentence F1s than positive means.

We can see that maximizing ELBO means minimizing the KL divergence between the variational posterior and the prior – this implies that the variational posterior is optimized towards the prior while still fitting the data. Thus, the mean of the prior pulls the mean of the variational posterior from which z is drawn.

We do not have a clear explanation for why the highest performing mean is around 0. In our current understanding of z, z encodes latent information about sentence structure, but how it does so is unclear.

One interpretation is that z represents a particular kind of sentence structure, and two distant points in z-space represent two very different sentence structures; we call this interpretation the **Euclidian** interpretation. Under this interpretation, we can interpret our loss of performance due to mean-moving as due to selecting priors on z that just happen to represent possible sentences structures that are very different from English. Perhaps choosing priors that had means differing along a different axis would lead to better performance.

Another interpretation is that cosine similarity is related to the syntactic similarity between two sentences; we call this interpretation the **cosine** interpretation. Under this interpretation, shifting the prior mean limits the range of syntactic structures that can be expressed by the model. This could explain the drop in performance, as the model tries to apply a similar syntactic structure to each sentence, whose true syntactic structure may be very different from the one that the model tries to apply.

A third interpretation is that the model is failing due to normalized weights and values, and that due to the high dimensionality of z, shifting the prior mean by integer amounts in each dimension increases $||z||$ so much that the model cannot function properly, which would also explain the drop in performance.

Prior Scale

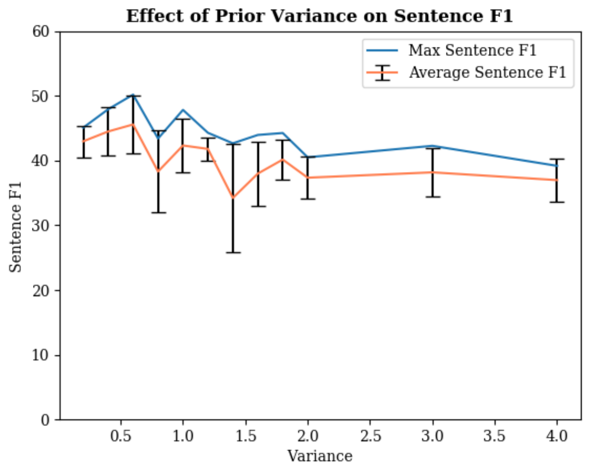

Figure 2: Effect of changing the variance of the prior on the result model's sentence F1. Prior is a Gaussian with mean 0

Based on Figure 2, we observe the best variance for a Gaussian prior of mean 0 is around 0.6. Like with the mean, we do not have a clear explanation for why the highest performing variance is 0.6.

The **Euclidean** interpretation is strongly consistent with this observation. A z drawn from a large variance distribution implies sentences have large structural variation, and hence don't bear much resemblance to one another. On the flip side, a z drawn from a small variance distribution implies sentences have low variation, and hence bear similarities that can be used to the model's advantage. In the space of short sentences, structural variation tends to be minimal: there aren't that many unique sentence structures given so few words. Hence, the model with the best performance is the one with a low variance prior.

With respect to the **cosine** interpretation, we note that while this interpretation does not imply the observed trend, the evidence does not necessarily preclude this interpretation either. Since the effect of prior variance on model performance is very small, the evidence is not strong enough to reject the interpretation outright.

Changing Variational Posterior Family Alone

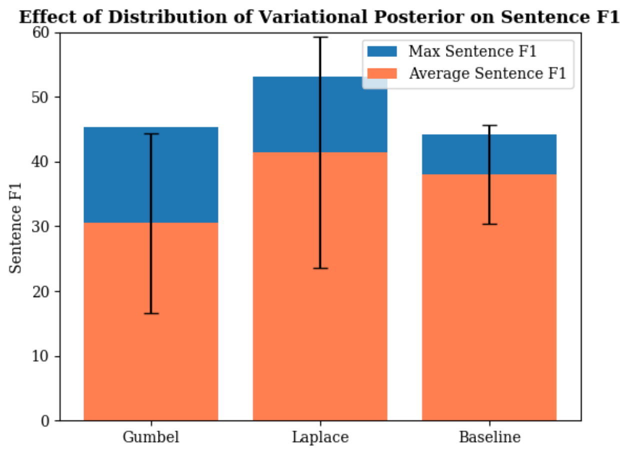

Figure 3: Effect of changing the distribution of the variational posterior on the result model's sentence F1.

As seen in Figure 3, we see that using a Laplace variational posterior outperforms both its Gumbel and Normal (baseline) counterparts. We can explain this with explanations similar to the ones presented in the two previous sections. The **Euclidean** interpretation would predict better performance with a variational posterior with higher concentration near the origin, since they would have both lower magnitudes and have lower differences; hence, it is more apt to draw z vectors from a Laplace distribution than a Gaussian distribution.

Changing Prior and Variational Posterior Family Concurrently

As seen in Figure 4, we again see that using the Laplace distribution for both the variational posterior and the prior out-

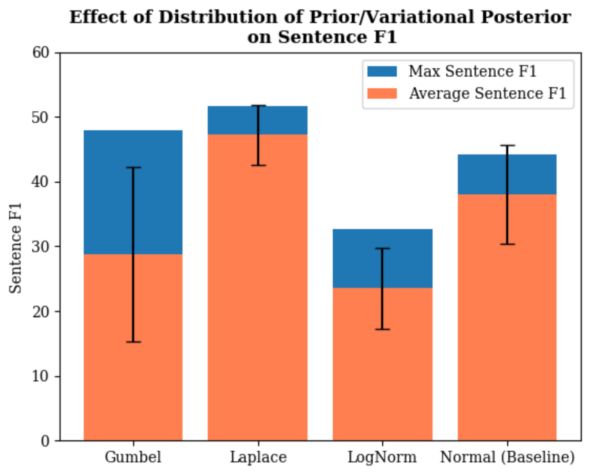

Figure 4: Effect of changing the distribution of both the prior and the variational posterior on the result model's sentence F1.

performs all other distributions. Similar to previous discussions, this can be rationalized by interpreting z as encoding latent information about the sentence. Since a Laplace distribution seems to be a better distribution to set priors and draw z's from, it makes sense that setting both the prior and the variational posteriors to be Laplace distributions would lead to better performance.

AP-CPCFGs: Should Scale be initialized to 1s or 0s?

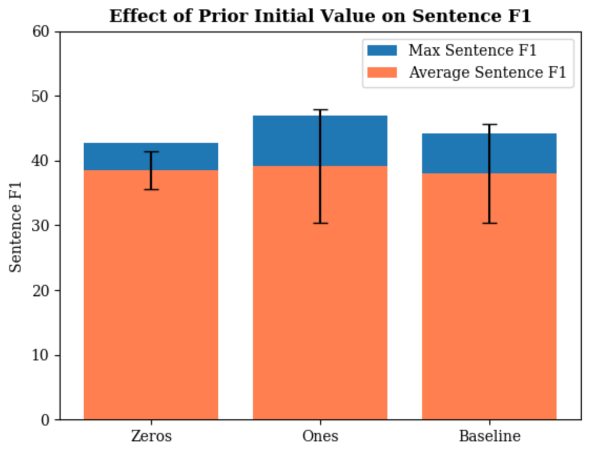

Figure 5: Effect of the initial value of the variance vector on Sentence F1 under the Adaptive Prior model. Baseline provided for reference.

We were initially concerned that initialization of the variance vector could potentially affect the ability of the model to learn and adapt its prior. Fortunately, it appears this is not the case. As seen in Figure 5, initialization doesn't seem to matter too much. The baseline AP models also appear to show no difference in performance when compared to the baseline, which is a little surprising given our intuitions. For future models using AP, we decided to initialize the variance to all 1s, per our prior intuitions.

AP: Different Distributions

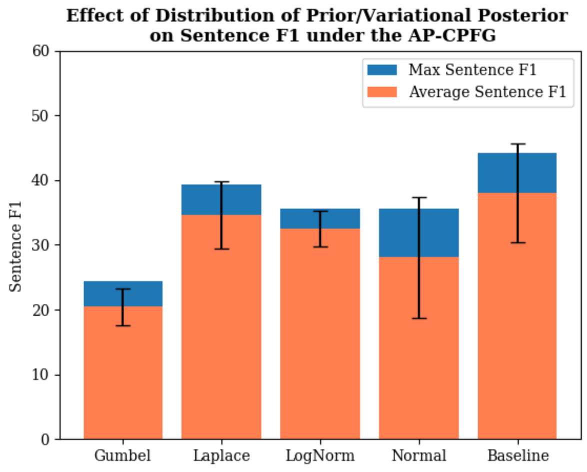

Figure 6: Effect of the prior/variational posterior distribution on Sentence F1 under the Adaptive Prior model. Baseline provided for reference.

As seen in Figure 6, we can see that once again the Laplace distribution model outperform all other models. Note, however, that performance in general suffers. The baseline model outperforms all other AP models.

This could be due to the ELBO landscape being difficult to traverse, leading to worse optimization and lower model performance.

SS-CPCFGs: BERT1, BERT2, BERT3

Among the three SS-CPCFG models, BERT1 appears to perform best. This is an interesting observation, especially since it suggests that semantic meaning is more important information for the prior on sentence structure rather than the particular rules of the PCFG. We can interpret this as semantics having a high-level impact on syntax but little local impact per sentence. It's also intriguing that BERT3 performs worse than BERT1. We'd expect that since BERT3 is strictly more expressive than BERT1 that it would, at worst, perform at the same level as BERT1. One possible explanation as to why this isn't the case is that the outputs of BERT are detrimental to the rule embedding network, making it more difficult to optimize the entire network properly and leading to worse performance.

However, as a whole, BERT1 outperforms the baseline. This makes sense, as the variational LSTM is meant to generate a variational posterior based on the sentence, and BERT-

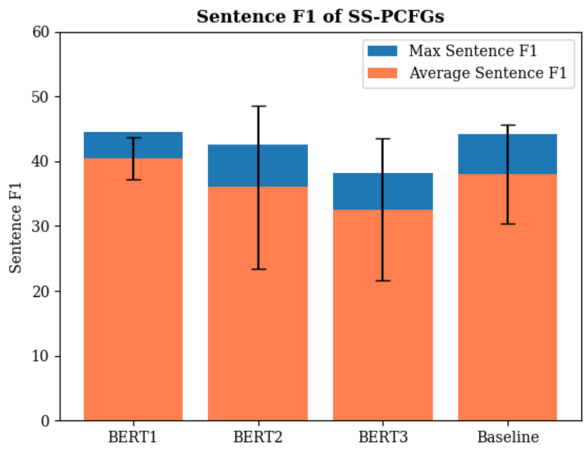

Figure 7: Sentence F1s of different SS-CPCFG architectures. Baseline provided for reference.

preprocessed sentences undergo much more powerful finetuning versus using raw sentences. This would likely make it easier for LSTM to extract the information necessary to perform variational inference.

SSAP-CPCFGs

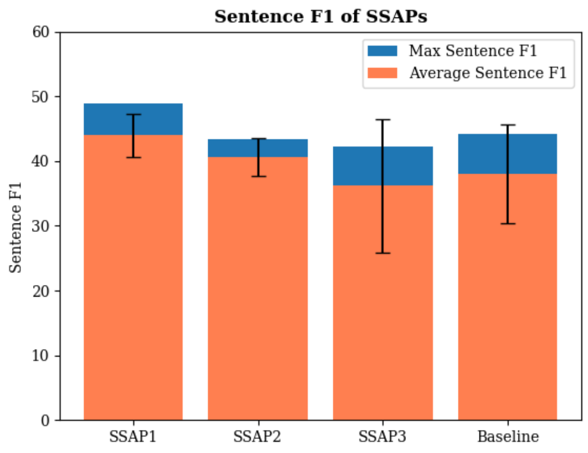

Figure 8: Sentence F1s of different SSAP-CPCFG architectures. Baseline provided for reference.

Based on Figure 8, we see that adding AP to our SS-PCFGs increased all their performances. Now, SSAP1 and SSAP2 outperform the baseline, with SSAP3 only trailing behind. We didn't expect that the AP addition would improve the BERT models based on their performance on the original model, but this isn't unreasonable either. Feeding BERT-preprocessed sentences into the model changes the ELBO

Table 1: Summary of Results

Model	Best Version	Avg Sentence F1	F1 Stdev	Max F1
Laplace	Prior/Vpost Same	47.22	4.58	51.59
AP	Laplace AP	34.60	5.19	39.37
SS / BERT	BERT1	40.47	3.23	44.44
SSAP	SSAP1	43.99	3.34	48.82
Baseline	N/A	38.00	7.57	44.12

landscape, and perhaps AP functions much better in this landscape compared to the original model.

Discussion

First, it is worthy to note that every model tested in this project is extremely sensitive to initialization – a bad seed can kill model performance across all models. Empirically, models with Gaussian distributions seem to be the most resistant to this, but even those models are not immune, as evidenced by the ubiquitous high error bars.

We note that PTB-10 is a small dataset, and that tangible differences exist between PTB and PTB-10. In addition to having more data points, PTB features longer sentence lengths, which means that a model trained on PTB is exposed to a greater breadth of syntactic structure than a model trained on PTB-10.

Preliminary results seem to suggest that on PTB-15, using a Gaussian distribution for the variational posterior offers better performance than using a Laplace distribution. We attribute this change to an increase in the complexity of the sentences observed by the model. If we use the **Euclidean** interpretation for z, we would expect that more complex sentences correspond to z's which are further apart. Indeed, this is what a Gaussian variational posterior offers: it is less clustered about its mean than the Laplace, allowing for more variation between z's and ultimately greater differences in sentence structure.

The linguistic phenomenon known as "movement" is an example of how longer sentences can have more syntactic variation. In broad terms, "movement" is the movement of one part of a built syntax tree to another part of the tree to create a new sentence. An simplistic example involves the sentence pair "I will be eating later today" and "Later today, I will be eating". These two sentences are derived from the same syntax tree, but movement of the constituent phrase "later today" occurs in the second sentence. However, because PCFGs have no way to model movement, they must derive separate rules that account these surface syntax differences, increasing the complexity of the syntax parsing problem. Movement is more likely to occur in longer sentences than shorter sentences simply because you need a sufficiently large syntax tree for movement to occur. Hence movement, and other linguistic phenomenon that arise at scale, can make the PTB dataset much more complex than the PTB-10 dataset.

Future Work

There are a lot of options still to be explored in the Compound PCFG model. First and foremost, this project was limited by computing resources and time – given the proper setup, it would be interesting to investigate the performance of each of the models proposed in this paper on the full PTB dataset.

Additionally, in our procedure, when we changed the mean and variance of our prior, we only changed them by scaling them along an axis; an investigation which could be fruitful is a more thorough search of the space of means and variances for the optimal prior parameters.

Another difficulty that the model faces is its optimization – the ELBO landscape is difficult to traverse, perhaps there exist more clever methods of optimization for the model, or even perhaps a better objective than ELBO.

One thing we did not investigate in this project is the use of Laplace prior and variational posterior distributions with the SSAP model. This may be promising because both the use of Laplace for prior and variational posterior distributions and the use of semantic sensitivity with adaptive prior yielded models with performance better than baseline. Combining the two ideas may lead to fruitful results as well.

A final extension that may lead to better performance is the introduction of tree-adjoining grammars (TAGs), which can help account for linguistic phenomenon like movement and may improve model performance, especially on longer sentences with more complex syntax.

Conclusions

We conclude that on PTB-10, using a Laplace distribution in place of a Gaussian for both prior and variational posterior improves the performance of a compound PCFG. Additionally, the addition of semantic sensitivity and adaptive priors to a compound PCFG improves its performance on small datasets. However, we note that the difference between PTB-10 and PTB are significant, and that the results that hold for PTB-10 need not necessarily hold for PTB, given the added complexity of the sentences in PTB compared to PTB-10.

Contributions

Alex and Tong collaborated on model design and writing the paper. Alex was most focused on implementation and data collection, while Tong was most focused on paper writing and data interpretation.

References

Carroll, G., & Charniak, E. (1992). *Two Experiments on Learning Probabilistic Dependency Grammars from Corpora.*

Dave Golland, J. D., & Uszkoreit, J. (2012). *A Feature-Rich Constituent Context Model for Grammar Induction.*

Jacob Devlin, K. L., Ming-Wei Chang, & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*

Kenichi Kurihara and Taisuke Sato. (2006). *Variational bayesian grammar induction for natural language.*

Kim, Y., Dyer, C., & Rush, A. M. (2019). *Compound Probabilistic Context-Free Grammars for Grammar Induction.*

Kingma, D. P., & Welling, M. (2014). *Auto-Encoding Variational Bayes.*

Klein, D., & Manning, C. (2002). *A Generative Constituent-Context Model for Improved Grammar Induction.*

Lari, K., & Young, S. (1990). *The Estimation of Stochastic Context-Free Grammars Using the Inside Outside Algorithm.*

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. (2007). *Bayesian inference for pcfgs via markov chain monte carlo.*

Percy Liang, M. I. J., Slav Petrov, & Klein, D. (2007). *The Infinite PCFG using Hierarchical Dirichlet Processes.*

Robbins, H. (1956). *An Empirical Bayes Approach to Statistics.*

Wang, P., & Blunsom, P. (2013). *Collapsed Variational Bayesian Inference for PCFGs.*

Yun Huang, M. Z., & Tan, C. L. (2012). *Improved Constituent Context Model with Features.*