# Latent syntax in a deep generative model of language

**Daan van Stigt**
Institute for Logic, Language and Computation

# Contents

# Latent syntax in a deep generative model of language

---

ABSTRACT

In this thesis I investigate the question: *What are effective ways of incorporating syntactic structure into neural language models?*

In this thesis I:

- study a class of neural language models that merges generative transition-based parsing with recurrent neural networks in order to model sentences together with their latent syntactic structure;
- propose a new globally trained chart-based parser as an alternative proposal distribution used in the approximate marginalization;
- propose effective methods for semisupervised learning, making the syntactic structure a latent variable;
- perform targeted syntactic evaluation and compare the model's performance with that of alternative models that are based on multitask learning.

I find that:

- ...
- ...

---

# Acknowledgements

# 1

# Introduction

This thesis investigates the question: *What are effective ways of incorporating syntactic structure into neural language models?*

We study a class of neural language models that explicitly model the hierarchical syntactic structure in addition to the sequence of words (Dyer *et al.*, 2016; Buys and Blunsom, 2015b; Buys and Blunsom, 2018). These models merges generative transition-based parsing with recurrent neural networks in order to model sentences together with their latent syntactic structure. The syntactic structure that decorates the words can be latent, and marginalized over, or can be given explicitly, for example as the prediction of an external parser. Although these are fundamentally joint model, they can be evaluated as regular language models (modeling only words) by (approximate) marginalization of the syntactic structure. In the case of the RNNG (Dyer *et al.*, 2016), exact marginalization is intractable due to the parametrization of the statistical model, but importance sampling provides an effective approximate method. An externally trained discriminative parser is used to obtain proposal samples. Other models provide exact marginalization, but this typically comes at the cost of a less expressive parametrization, for example one in which the features cannot be structure-dependent (Buys and

Blunsom, 2018).

In this thesis I study the RNNG (Dyer *et al.*, 2016) and investigate:

**The approximate marginalization** I propose an alternative proposal distribution and investigate the impact.

- I propose a new discriminative chart-based neural parser that is trained with a global, Conditional Random Field (CRF), objective. The parser is an adaptation of the minimal neural parser proposed in Stern *et al.* (2017), which is trained with a margin-based objective.

- This contrast with the choise of Dyer *et al.* (2016) for a transition-based parser as proposal, a discrminatively trained RNNG.

- We posit that a globally trained model is a better proposal distribution than a locally trained transition based model: a global model has ready access to competing analyses that can be structurally dissimilar but close in probability, whereas we hypothesize that a locally trained model is prone to produce locally corrupted structures that are nearby in transition-space.

- In a transition based parser more diverse samples can be obtained by flattening the transition distributions. This causes the model to be less confident in its predictions. A downside is that this approach causes the model to explore parts of the probability space which it has not encountered during training.

- The above is a general challenge for greedy transition based models that can be answered to by training with dynamic oracles (Goldberg and Nivre, 2013), also called 'exploration' ((Ballesteros *et al.*, 2016; Stern *et al.*, 2017). These approaches can be considered instances of imitation learning (Vlachos, 2013; He *et al.*, 2012).

- We do not consider these directions in this thesic. Dynamic oracles can produce substantial improvements in constituency parsing performance, but they must be custom designed for each transition system (Fried and Klein, 2018).

**Semi-supervised training by including unlabeled data** To make joint models competitive language models they need to make use of the vast amounts of unlabeled data that exists.

- A major drawback of these syntactic language models is that they require annotated data to be trained, and preciously little of such data exists.
- We extend the training to the unsupervised domain by optimizing a variational lower bound on the marginal probabilities that jointly optimizes the parameters of proposal model ('posterior' in this framework) with the joint model.
- We obtain gradients for this objective using the score function estimator (Fu, 2006), also known as REINFORCE (Williams, 1992), which is widely used in the field of deep reinforcement learning, and we introduce an effective baseline based on argmax decoding (Rennie *et al.*, 2017), which significantly reduces the variance in this optimization procedure.
- Our CRF parser particularly excels in the role of posterior thanks the independence assumptions that allow for efficient exact computation of key quantities: the entropy term in the lower bound can be computed exactly using Inside-Outside algorithm, removing one source of variance from the gradient estimation, and the argmax decoding can be performed exactly thanks to Viterbi, making the argmax baseline even more effective.

**Alternative, simpler, models** There are alternatives to the methods that this thesis investigates.

- Multitask learning of a neural language model with a syntactic side objective is a competitive and robust alternative method to infuse neural language models with syntactic knowledge.
- Training the syntactic model on data that mixes gold trees with predicted 'silver' trees for unlabeled data is a competitive and robust alternative to fully principled semi-supervised learning.

- We propose a simple multitask neural language model that predicts labeled spans from the RNN hidden states, using a feature function identical identical to that used in the CRF parser.
- We consider these alternatives in order to quantify significance of the latent structure and the semisupervised training as measured by some external performance metric.

**Targeted syntactic evaluation** TBA

# 2

# Background

In this chapter I give background required to read this thesis.

## 2.1 Syntax

Introduce the background on syntax relevant for the chapters on parsing, and the syntactic evaluation. My aim is to provide a succinct and compelling answer to the inevitable question: *Why do we care about constituency structure?* In particular I want to firmly establish the concept of constituents, and secondly I want to set some of the ground for the syntactic evaluation that we perform in the final chapter.

- Introduce the central question: what are sentences, strings or structures? Is a sentence a string of words in linear order, or are the words combined in a hierarchical structure? (Everaert *et al.*, 2015; Frank *et al.*, 2012)
- The notion of a constituent, and constituent tests Carnie (2010) and Huddleston and Pullum (2002).
- Hierarchical structure (of these constituents) Everaert *et al.* (2015).
- The lexical and phrasal categories of constituent analysis.

- Evidence for structured sentence processing from psycholinguistic research (Hale, 2001; Levy, 2008; Brennan *et al.*, 2016).
- Introduce the concept of *acceptability judgements*, with the final chapter on syntactic evaluation in mind.

In the following exposition we primarily follow Huddleston and Pullum (2002), with some excursions into Carnie (2010) and Everaert *et al.* (2015). The first source is a well established reference grammar of the English language that is relatively agnostic with respect to theoretical framework, whereas the the later two sources are more firmly rooted in a particular framework[1]. We take the following three pricinpiles (about English syntax!) from (Huddleston and Pullum, 2002) as guiding:

(i) Sentences consist of parts that may themselves have parts.

(ii) These parts belong to a limited range of types.

(iii) The constituents have specific roles in the larger parts they belong to.

**Constituents** Sentences consist of parts that may themselves have parts. The parts are groups of words that function as units, and they are called *constituents*. Consider the simple sentence *A bird hit the car*. The immediate constituents are *a bird* (the subject) and *hit the car* (the predicate). The phrase *hit the car* can be further analyzed as containing the constituent *the car*. The ultimate constituents of a sentence are the atomic words, and the entire analysis is called the constituent structure of the sentence. This structure can be indicated succinctly with the use of brackets

[ A bird [ hit [ the car ] ] ]

or less succinctly as a tree as in figure 2.1. Evidence for the existence of these constituents can be provided by examples such as the following, which are called constituent tests (Carnie, 2010). Consider inserting the adverb *apparently* into our example sentence, indicating the alleged

[1] Broadly subsumable under the label *generative grammar*.

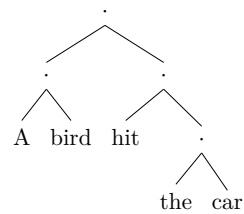



**Figure 2.1:** Sentence 2.1 as a tree. (TODO: draw this without labels.)

status of the event described in the sentence. In principle there are six positions available for the placement of *apparently* (including before, and after the sentence). However, only three of these placements are actually permissible[2]:

1. *Apparently* a bird hit the car.
2. *An *apparently* bird hit the car.
3. A bird *apparently* hit the car.
4. *A bird hit *apparently* the car.
5. *A bird hit the *apparently* car.
6. A bird hit the car, *apparently*.

Based on the bracketing in 2.1 we can formulate a general constraint: the adverb must not interrupt any constituent. Indeed, this explains why *actually* cannot be placed anywhere inside *hit the car* and not between *a* and *bird*. For full support, typically results from many more such test are gathered, and in general these tests can be much more controversial than in our simple example (Carnie, 2010).

**Syntactic categories** The constituents of a sentence belong to a limited range of types that form the set of syntactic categories (Huddleston and Pullum, 2002). Two types are distinguished: lexical categories (part

[2]We use an asterisk '*' to indicate a sentence that is judged ungrammatical, as is customary in linguistics.

of speech) and phrasal categories. The tree in figure 2.1 can be represented in more detail by adding syntactic categories, see figure 2.2.

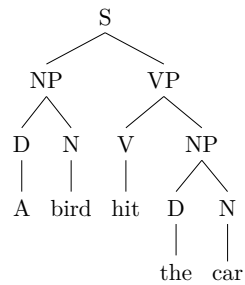
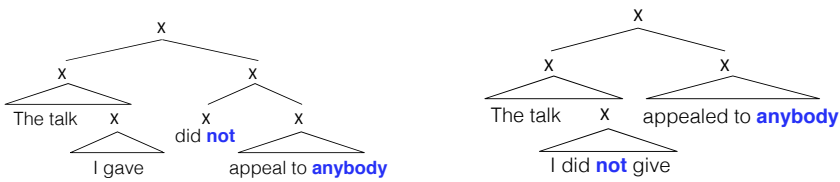



**Figure 2.2:** A tee specified with lexical (D, N, V) and phrasal categories (S, NP, VP).

**Hierarchical structure** The constituents have specific roles in the larger parts they belong to (Huddleston and Pullum, 2002). This structure provides constraints that are not explainable from the linear order of the words themselves. Consider the following example about the syntactic behaviour of *negative polarity items* (NPIs)[3] such as *anybody*:

1. The book that I bought did *not* appeal to *anybody*.




**(a)** In licensing context. **(b)** Not in licensing context.

**Figure 2.3:** Negative Polarity. Figure taken from Everaert *et al.*, 2015. (TODO: draw in qtree, but how to make the nested roofs?)

[3] A negative polarity item is, to first approximation, a word or group of words that is restricted to negative context (Everaert *et al.*, 2015). More generaly they are words that need to be licensed by a specific *licencing context* (Giannakidou, 2011).

2. * The book that I bought appealed to *anybody*.

From this example we might formulate the hypothesis that the word *not* must linearly precede the word *anybody*. A counter example refutes this linear hypothesis:

1. *The book I did *not* buy appealed to *anybody*.

Instead, the constraints that govern this particular pattern depend on hierarchical structure: the word *not* must "structurally precede" the word *anybody* (Everaert *et al.*, 2015). Figure shows the constituent structure of both sentences. The explanation goes as follows (put this in my own words): "In sentence 2.3 (a) the hierarchical structure dominating *not* also immediately dominates the hierarchical structure containing *anybody*. In sentence 2.3 (b), by contrast, *not* sequentially precedes *anybody*, but the triangle dominating *not* fails to also dominate the structure containing *anybody*."

**Cognitive reality** Does all this exist in the human brain? These people say *yes*: (Hale, 2001; Levy, 2008; Brennan *et al.*, 2016).

**Controversy** Theoretical syntax is rife with controversy, and wildly differing viewpoints exist. In fact, for each point made in our short discussion, the exact opposite point has been made as well:

- Constituents are fundamental (Huddleston and Pullum, 2002; Carnie, 2010) *versus* word-word relations are all you need (Tesnière, 1959; Nivre, 2005; Hudson, 2010).
- Hierarchical structure is a core feature of language (Everaert *et al.*, 2015) *versus* sequential sentence structure has enough explanatory power Frank *et al.*, 2012. The claim is that (2) is cognitively more fundamental than (1),

es [ [can [be analysed ] [as [hierarchically structured] ] ] ]

Sentences [can be analysed] [as hierarchically structured]

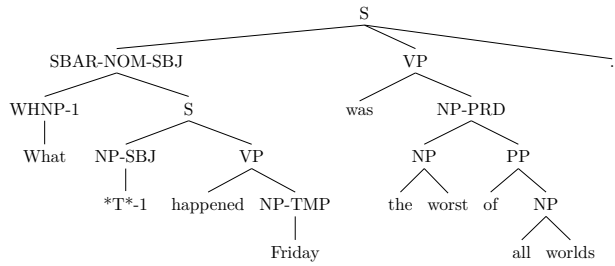
which is exactly contrary to the analyses above. This is more similar to the NLP task of *chunking*, or shallow parsing.

- Studies in cognitive neuroscience and psycholinguistics show that human sentence processing is hierachical (Hale, 2001; Levy, 2008; Brennan *et al.*, 2016) *and* they show that it is not (Conway and Pisoni, 2008; Christiansen *et al.*, 2012; Gillespie and Pearlmutter, 2011; Gillespie and Pearlmutter, 2013) (selected from the survey in Frank *et al.* (2012)).

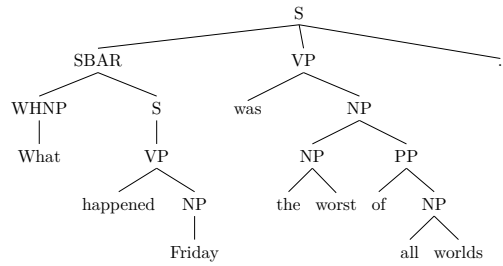
In this research I take a position of extreme pragmatism with respect to syntax: it is whatever our dataset says it is. Which means in our case, the Penn Treebank has the final word on the subject. And is language hierachical or linear? That is exactly what we inted to ivestigate from a statistical and computational viewpoint.

## 2.2 Parsing

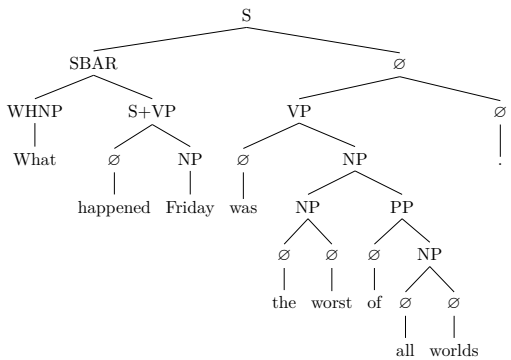
- Treebanks, in particular the Penn Treebank. Treebank preprocessing. CFGs, CNF, spans. Reference figure 2.4.
- The two conceptions of a tree: as a set of *labeled spans* or as a set of *anchored rules*.
- A labeled span is a triple $(\ell, i, j)$ of a syntactic label $\ell$ together the left and right endpoints $i$, $j$ that the label spans.
- An *anchored rule* is a triple $(r, i, j)$ or four-tuple $(r, i, k, j)$, containing a CNF rule $r$ with span endpoints $i$, $j$, and a split-point $k$ of the left and right child $r$ is not a lexical rule.
- For the difference, consider the following two representations of the tree in figure 2.4d given in table 2.1.
- Algorithms for parsing: global chart based, local transition based
- Dynamic programming inference versus search heuristics.
- Modelling types: generative, discriminative, log-linear, count-based, feature-based, neural network features.



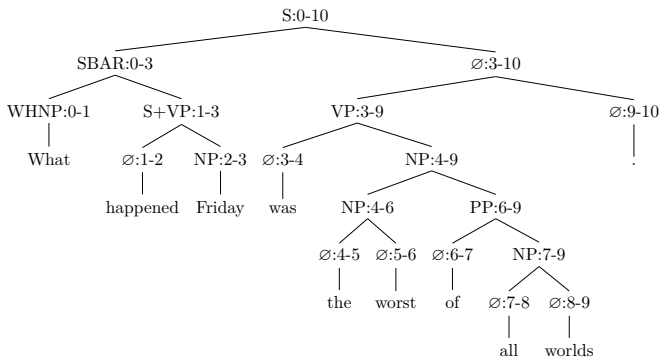

**(a)** Original Penn Treebank tree.




**(b)** Function tags and traces removed.




**(c)** Converted to normal form.




**(d)** In normal form with spans.

**Figure 2.4:** Converting a treebank tree (withouth part-of-speech tags).

| Labeled spans | Anchored rules |
|---|---|
| (S, 0, 10) | (S → SBAR ∅, 0, 3, 10) |
| (SBAR, 0, 3) | (SBAR → WHNP S+VP, 0, 1, 3) |
| (VP, 1, 3) | (S+VP → ∅ NP, 1, 2, 3) |
| ⋮ | ⋮ |
| (NP, 7, 9) | (NP → ∅ ∅, 7, 8, 9) |

**Table 2.1:** Two conceptions of the tree in 2.4d.

## 2.3 Language models

- Briefly mention some typical approaches for langugage modelling: count based n-gram with smoothing (Chen and Goodman, 1999; Kneser and Ney, 1995), neural n-gram (Bengio *et al.*, 2003) and recurrent neural network (Mikolov *et al.*, 2010). Also mention some (early) syntactic approaches: count-based (Chelba and Jelinek, 2000; Pauls and Klein, 2012), neural (Emami and Jelinek, 2005), and top-down parsing related (Roark, 2001).
- Explain the metric perplexity.
- Briefly mentions some typical datasets and some benchmarks (dataset, perplexity, number of parameters, training time).
- Mention some downsides of the perplexity metric: conflating different sources of succes in next-word prediction (simple collocations, semantics, syntax).
- Note that there exists some alternatives to perplexity: adversarial evaluation (Smith, 2012), subject-verb agreement (Linzen *et al.*, 2016) and grammatical acceptability judgments (Marvin and Linzen, 2018).

## 2.4 Neural networks

Introduce all the neural networks.

- We consider the neural networks as abstractions denoting certain parametrized functions FeedForward, rnn, Lstm, etc.

- Let $\mathbf{x}$ and $\mathbf{y}$ be vectors in respectively $\mathbf{R}^n$ and $\mathbf{R}^m$.

  A *feedforward neural network* is a parametrized function FeedForward from $\mathbf{R}^n$ to $\mathbf{R}^m$ such that

  $$\textsc{FeedForward}(\mathbf{x}) = \mathbf{y}.$$

  A *recurrent neural network* is a parametrized function rnnthat takes a sequence of vectors $(\mathbf{x}_i)_{i=1}^n = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ in $\mathbf{R}^n$ and produces a sequence of output vectors in $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ in $\mathbf{R}^m$:

  $$\textsc{rnn}((\mathbf{x}_i)_{i=1}^n) = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n).$$

  Internally, the rnnrecursively applies a function $f : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^m$ that is defined by the recursion

  $$\begin{aligned}\mathbf{y}_t &= f(\mathbf{x}_t, \mathbf{y}_{t-1}) \\ &= f(\mathbf{x}_t, f(\mathbf{x}_{t-1}, \mathbf{y}_{t-2})) \\ &\vdots \\ &= f(\mathbf{x}_t, f(\mathbf{x}_{t-1}, f(\ldots f(\mathbf{x}_1, \mathbf{y}_0)))),\end{aligned}$$

  which is how the vectors $\mathbf{y}_i$ are computed. That is, $f$ takes the input $\mathbf{x}$ of the current timestep $t$ and the output $\mathbf{y}$ of the previous timestep $t-1$ and returns a new ouput for timestep $t$. The initial vector $\mathbf{y}_0$ does not depend on the input, and can be fixed or part of the function's set of parameters.

  An rnncan be applied to the input sequence in reverse, that is, in the *backward* direction:

  $$\begin{aligned}\textsc{rnn}_B((\mathbf{x}_i)_{i=1}^n) &= \text{reverse}(\textsc{rnn}(\text{reverse}((\mathbf{x}_i)_{i=1}^n))) \\ &= \text{reverse}(\textsc{rnn}(\mathbf{x}_n, \mathbf{x}_{n-1}, \ldots, \mathbf{x}_1) \\ &= \text{reverse}(\mathbf{y}_n, \mathbf{y}_{n-1}, \ldots, \mathbf{y}_1) \\ &= (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n),\end{aligned}$$

where we defined a

$$\text{reverse}((\mathbf{x}_i)_{i=1}^n) \triangleq (\mathbf{x}_n, \mathbf{x}_{n-1}, \ldots, \mathbf{x}_1), \tag{2.1}$$

For consistency we will refer to the RNNin the regular, *forward*, direction as $\text{RNN}_F$. To stress the difference between the different outputs obtained from the two directions, we will denote the output vectors obtained in the regular, forward, direction with $\mathbf{f}_i$ and the vectors obtained in the backward direction with $\mathbf{b}_i$:

$$\text{RNN}_F((\mathbf{x}_i)_{i=1}^n) = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)$$
$$\text{RNN}_B((\mathbf{x}_i)_{i=1}^n) = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$$

The two functions above can used to construct a *bidirectional* RNNby combining the output of each as

$$\text{BiRNN}((\mathbf{x}_i)_{i=1}^n) = (\mathbf{f}_1 \circ \mathbf{b}_1, \mathbf{f}_2 \circ \mathbf{b}_2, \ldots, \mathbf{f}_n \circ \mathbf{b}_n), \tag{2.2}$$

where we use $\circ$ to denote vector concatenation, *i.e.*$\mathbf{x} \circ \mathbf{y}$ is a vector in $\mathbf{R}^{n+m}$.

An LSTMis a particular way to construct the RNNinternal function $f$, and similarly has a bidirectional equivalent denoted BILSTM.

- The function FEEDFORWARDis defined as follows:
- The function LSTMis defined as follows
- (Minibatch) SGD optimization.

# 3

# Recurrent Neural Network Grammars

## 3.1 Model

I describe the model.

- A discriminative RNNG is a discriminative transition based parser that uses regular RNNs to summarize the actions in the history and the words on the buffer into vectors, and uses a special RNN with a syntax-dependent recurrence—a StackLSTM (Ballesteros *et al.*, 2017) outfitted with a custom 'composition' function to summarize closed constituents—to obtain a vector representation of items on the the stack.

- Depending on the perspective, the generative RNNG is either a structured model of language that predicts words together with their structure, or a generative version of the discriminative RNNG that jointly models the words in the instead of conditioning on them. From the view of parsing, it simply dispends with the buffer of the discriminative RNNG to instead predict the words that dscorate the tree. As a model of sentences, it can be understood as kind of structured RNN: it predicts words, but also compresses and labels them recursively whenever they form

a complete constituents.

- Specify the transition-system.

- Fundamentally, the model is a probability distribution over action sequences $\mathbf{a} = (a_1, \ldots, a_T)$ that generate trees $\mathbf{y}$ conditionally given a sequence of words $\mathbf{x}$ in the discriminative model, and jointly with $\mathbf{x}$ in the the generative model.

  Put simply, the model is defined as

  $$p(\mathbf{a}) = \prod_{t=1}^{T} p(a_t \mid \mathbf{a}_{<t}). \tag{3.1}$$

  However, the exact model is slightly more complicated, a consequence of the difference between the discrminative and the generative actions, and a consequence of practical concerns regarding the implementation.

- First we define a set of discriminative actions and a set of generative models,

  $$A_{\mathcal{D}} = \{\textsc{shift}, \textsc{open}, \textsc{reduce}\}, \tag{3.2}$$

  and

  $$A_{\mathcal{G}} = \{\textsc{gen}, \textsc{open}, \textsc{reduce}\}. \tag{3.3}$$

  And we define a finite set of nonterminal symbols

  $$N = \{\mathrm{S}, \mathrm{NP}, \ldots, \mathrm{WHNP}\},$$

  and a finite alphabet

  $$\Sigma = \{\text{all}, \text{Friday}, \ldots, \text{worst}\}.$$

  For the discriminative model $\mathbf{a}$ is an element of $A_{\mathcal{D}}^T$, and for the generative model $\mathbf{a}$ is an elemtnt of $A_{\mathcal{G}}^T$, both with the restriction that they form a valid tree $\mathbf{y}$. A sequence of nonterminals $\mathbf{n}$ in $N^K$ is the sequence of nonterminal nodes in $\mathbf{y}$ in pre-order. A sentence $\mathbf{x}$, finally, is an element of $\Sigma^N$.

- Then let $\mathbf{1}_{\{a_t = \text{OPEN}\}}$ be the indicator function for the event that the action $a_t$ is to open a new nonterminal, and similarly let $\mathbf{1}_{\{a_t = \text{GEN}\}}$ be the indicator function for the event that the action $a_t$ is to generate a word. Furthermore we introduce two functions that maps between sets of indices to indicate the number of times a particular action has been taken at each time step:

$$\mu : \{1, \ldots, T\} \to \{1, \ldots, M\} : \mu(t) = \sum_{i<t} \mathbf{1}_{\{a_i = \text{OPEN}\}},$$
$$\nu : \{1, \ldots, T\} \to \{1, \ldots, N\} : \nu(t) = \sum_{i<t} \mathbf{1}_{\{a_i = \text{GEN}\}}.$$

- Let $\mathbf{a}$ be a sequeunce from $A^T_{\mathcal{D}}$. Then the model for the discrminative RNNG is

$$p(\mathbf{a} \mid \mathbf{x}) = \prod_{t=1}^{T} p(a_t \mid \mathbf{x}, \mathbf{a}_{<t}) p(n_{\mu(t)} \mid \mathbf{x}, \mathbf{a}_{<t})^{\mathbf{1}_{\{a_t = \text{OPEN}\}}}. \tag{3.4}$$

- Let $\mathbf{a}$ be a sequeunce from $A^T_{\mathcal{G}}$, which include the actions that generate words[1], then the model for the generative RNNG is

$$p(\mathbf{a}) = \prod_{t=1}^{T} p(a_t \mid \mathbf{a}_{<t}) p(n_{\mu(t)} \mid \mathbf{a}_{<t})^{\mathbf{1}_{\{a_t = \text{OPEN}\}}} p(x_{\nu(t)} \mid \mathbf{a}_{<t})^{\mathbf{1}_{\{a_t = \text{GEN}\}}}. \tag{3.5}$$

- The probabilities are given by linear regression classifiers on a feature vector $\mathbf{u}_t$[2],

$$p(a_t \mid \mathbf{a}_{<t}) = \frac{\exp(\mathbf{w}_{a_t}^\top \mathbf{u}_t + b_{a_t})}{\sum_{a \in A} \exp(\mathbf{w}_a^\top \mathbf{u}_t + b_a)}, \tag{3.6}$$

$$p(n_{\mu(t)} \mid \mathbf{a}_{<t}) = \frac{\exp(\mathbf{v}_{n_{\mu(t)}}^\top \mathbf{u}_t + b_{n_{\mu(t)}})}{\sum_{n \in N} \exp(\mathbf{v}n^\top \mathbf{u}_t + b_n)}, \tag{3.7}$$

$$p(x_{\nu(t)} \mid \mathbf{a}_{<t}) = \frac{\exp(\mathbf{r}_{x_{\nu(t)}}^\top \mathbf{u}_t + b_{x_{\nu(t)}})}{\sum_{x \in \Sigma} \exp(\mathbf{r}_x^\top \mathbf{u}_t + b_x)}, \tag{3.8}$$

$$\tag{3.9}$$

[1] Note that under this action set $p(a_t \mid \mathbf{a}_{<t}, \mathbf{x}_{<t}) = p(a_t \mid \mathbf{a}_{<t})$, given the fact that the words in $\mathbf{x}_{<t}$ are contained in $\mathbf{a}_{<t}$.

[2] For brevity we omit the conditioning on $\mathbf{x}$, which was redundant already in the case of the generative model.

where $A$ can denote either $A_{\mathcal{D}}$ or $A_{\mathcal{G}}$, and $\mathbf{w}_i$, $\mathbf{v}_i$, $\mathbf{r}_i$, and $b_i$ are parameters.

- How the feature vector $\mathbf{u}_t$ is constructed is outlined in the next section.

- Note that could have defined

$$A_{\mathcal{D}} = \{\text{REDUCE}, \text{SHIFT}\} \cup \{\text{OPEN}(n) \mid n \in N\},$$

and

$$A_{\mathcal{G}} = \{\text{REDUCE}\} \cup \{\text{OPEN}(n) \mid n \in N\} \cup \{\text{GEN}(x) \mid x \in \Sigma\},$$

and defined $p(\mathbf{a})$ as in 3.1. However, in the case of the generative model this is particularly inefficient from a computational perspective. Note that the set $\Sigma$ is generally very very large[3], and observe that the normalization in 3.6 requires a sum over all actions, while a large number of the actions do not generate words. For consistency we extend this modelling choice to the discriminative RNNG. Besides, the presentation in 3.4 and 3.5 is conceptually cleaner: first choose an action, then, if required, choose the details of that action. For these reasons combined we opt for the two-step prediction. And although it appears that Dyer *et al.* (2016) model the sequences according to 3.1, followup work takes our approach and models the actions of the generative RNNG as in equation 3.5 (Hale *et al.*, 2018).

### 3.1.1 Features

The feature vector $\mathbf{u}_t$ from which the transition probabilities are computed are computed from the stack configuration's entire history, and in a syntax-dependent way.

- The StackLSTM computes incremental features for the sequences on the three datastructures of the transition-system, with unbounded history.

[3] On the order of tens of thousands.

- A composition function computes representations of closed constituents on the stack.
- There are two options for the composition function: simple BiRNN and attention-based. The attention-based composition performed best in earlier research, so we focus on this function.
- There is evidence that the stack-datastructure is all that is needed. However, we focus on the models that compute representations of all the datastructures.

## 3.2 Syntax and cognition

Here I describe the research into syntax and cognition using the RNNG.

**Cognition** RNNGs can tell us something about our brains.

- Psycholinguistic research that indicates that top-down parsing is a cognitively plausible parsing strategy (Brennan *et al.*, 2016).
- RNNGs are good statistical predictors in psycholinguistic research (Hale *et al.*, 2018). More precisely: the sequential word-probabilities that are derived from a generative RNNG in combination with word-synchronous beam-search (**Stern+2017:beam**) provide per-word complexity metrics that predict human reading difficulty well.

**Syntax** What do RNNGs learn about syntax?

- RNNGs learn a number of syntactic phenomena as a side product of the main objective. The attention mechanism in the composition function learns a type of 'soft' head-rules, and when trained on trees without syntactic labels the RNNG still learns representations for constituents that cluster according to their withheld gold label (Kuncoro *et al.*, 2017).
- RNNGs are better at a long-distance verb-argument agreement task than LSTMS (Linzen *et al.*, 2016; Kuncoro *et al.*, 2018).

### 3.3 Experiments

We perform three types of experiments with the RNNG:

- We reproduce the parsing f-scores and perplexities from (Dyer *et al.*, 2016), and some more.
- We evauluate 'how good the model is' as a sampler.

**Supervised model** We investigate the following.

- We train with standard hyperparameter settings and optimizer, and replicate the original results. We will get a little lower with the discriminative model because we do not use tags.
- We evaluate F-score with 100 samples (as many proposal trees as possible).
- We evaluate perplexity with varying number of samples: 1 (argmax), 10, 20, 50, 100 (default). The peplexity evaluation with the argmax prediction gives an impression of the uncertaty in the model (Buys and Blunsom, 2018).

**Sampler** We investigate the following:

- We asses the conditional entropy of the model. This is most quantitative. Recall that conditional entropy is defined as

$$\mathrm{H}(Y \mid X) = \sum_{x \in \mathcal{X}} p_X(x) \mathrm{H}(Y \mid X = x), \tag{3.10}$$

  where

$$\mathrm{H}(Y \mid X = x) = -\sum_{y \in \mathcal{Y}} p_{Y|X}(y \mid x) \log p_{Y|X}(y \mid x). \tag{3.11}$$

  We estimate the quantity $\mathrm{H}(Y \mid X = x)$ with the model samples. We estimate the quantity $\mathrm{H}(Y \mid X)$ by a sum over the development dataset. For the probabilities $p_X(x)$ we use the marginalized probabilities of the joint RNNG (with samples from the discrminative parser $p_{Y|X}$).

- We asses for some cherry picked sentences. This is more qualitative. These sentences should be difficult or ambiguous. Or they can be ungramatical when taken from the syneval dataset. We can evaluate their entropy, and the diversity of samples, for example to see if there are clear modes. We can make violinplots of the probabilities of the samples. We can compute the f-scores of the samples compared with the argmax tree.

## 3.4 Related work

- Generative dependency parsing and language modelling (Titov and Henderson, 2007; Buys and Blunsom, 2015a; Buys and Blunsom, 2015b; Buys and Blunsom, 2018)
- Top-down parsing and language modelling (**Roark2001**).
- Brain research with top-down parsing (Brennan *et al.*, 2016; Hale *et al.*, 2018).

# 4

# Conditional Random Field parser

In this chapter I introduce an alternative parser to act as proposal model in the approximate marginalization.

- The parser is a neural Conditional Random Field (CRF) parser that combines the efficient exact inference of chart-based parsing with the rich nonlinear features of neural networks.
- The chart-based approach allows efficient exact inference, while the neural features can be relatively rich and can condition on the entire sentence.
- The neural network is used exclusively to learn good representation from which to predict local scores, while the global structured interactions are
- The parser is an adaptation of the chart-based parser introduced in Stern *et al.* (2017), where it is trained with a margin-based objective.

**Notation** Let a sentence be $x_1, \ldots, x_n$, where each $x_i$ is a word. We are given a CFG $(N, \Sigma, R, S)$ in Chomsky normal form. Let $\psi$ be a function

that maps any rule production $r \in R$ of the form $\langle A \to B\ C, i, k, j\rangle$ or $\langle A, i, i+1\rangle$ to a value $\psi(r) \geq 0$. Let a tree $T$ be a set of such rules $r$ with the only constraint that these rules make up a tree.

## 4.1 Model

I describe the probabilistic model of the parser.

1. Introduce probabilistic model, reference the appendix on CRFs C.
2. Desribe how this is an adaptation from Stern *et al.* (2017) to probabilistic training.

Following the minimal span parser we a scoring function $\psi$ as defined on spans

$$\log \psi(A \to B\ C, i, k, j) = \log \psi(A, i, j) \tag{4.1}$$
$$\tag{4.2}$$

discarding the rest of the span information. The function is then defined as

$$\log \psi(A, i, j) \triangleq s(i, j, A), \tag{4.3}$$

and thus the potential of a tree as

$$\log \Psi(T) = \sum_{r \in T} \log \psi(r) \tag{4.4}$$
$$= \sum_{\langle A,i,j\rangle \in T} s(i, j, A), \tag{4.5}$$
$$\tag{4.6}$$

Note that the potential function as defined in **??** disregards most of the information in a binary rule. In particular we see that $B$, $C$ and $k$, the labels and split-point of the children, are discarded.

Now note that equation 4.4 corresponds exactly to the second formula in section 3 of the minimal span-based parser paper

$$s_{tree}(T) = \sum_{(\ell(i,j))\in T} [s(i, j, \ell)]. \tag{4.7}$$

which is how I derived that **??** is the correct formula for the rule score.

We obtain our CRF objective when we normalize this score globally

$$P(T) = \frac{\prod_{r \in T} \psi(r)}{\sum_{T' \in \mathcal{T}} \prod_{r' \in T'} \psi(r')} \tag{4.8}$$

$$\tag{4.9}$$

or equivalently

$$\log P(T) = \sum_{r \in T} \log \psi(r) - \log \sum_{T \in \mathcal{T}} \prod_{r \in T} \psi(r) \tag{4.10}$$

$$\tag{4.11}$$

### 4.1.1 Features

I describe how the local scores are computed using neural networks.

- Give formal expression for feature function: 'LSTM minus features' with Feedforward scoring function.

### 4.1.2 Motivation

- Key point to make: this model regards a constituency tree as a collection of *labeled spans* over a sentence. Earlier models, both log-linear and neural, regard a constituency tree as a collection of *anchored rules* over a sentence (Finkel *et al.*, 2008; Durrett and Klein, 2015).

- A model over spanned rules puts more expressiveness in the state space of the dynamic program, because the correlations between subparts of the trees are modeled through the rich rules. The model in Stern *et al.* (2017) instead puts the expressiveness in the input space by using rich neural feature representations. The state space in contrast is less structured, because the score-function is agnostic to the composition of it's children.

- Earlier approaches went even farther. These approaches enriched the grammar by lexicalizing the rules (Collins, 2003) or by breaking the grammar's independence assumptions by annotating the rule with parent and sibling labels (Klein and Manning, 2003).

- The choice to model labeled spans makes dramatically improves the speed of this model. In the section on inference we will show precisely how.

## 4.2 Inference

Due to the parametrization, the model allows efficient inference. In this section we describe efficient solutions to three related problems:

- Find the best parse $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$
- Compute the normalizer $Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{a=1}^{A} \Psi(\mathbf{x}, \mathbf{y}_a)$, where $F = \{\Psi_a\}_{a=1}^{A}$ is the set of factors in the graph.
- Compute the entropy conditioned on $\mathbf{x}$, $H(\mathbf{y}|\mathbf{x})$.

All three problems can be solved with a different instance of the same two algorithm: the inside algorithm and the outside algorithm.

### 4.2.1 Inside recursion

In this derivation we follow Michael Collins notes on the Inside-Outside Algorithm.[1] We have the following general result for the inside value $\alpha$. For all $A \in N$, for all $0 \leq i < n$

$$\alpha(A, i, i+1) = \psi(A, i, i+1) \tag{4.12}$$

and for all $(i, j)$ such that $1 \leq i < j \leq n$:

$$\alpha(A, i, j) = \sum_{A \to BC} \sum_{k=i+1}^{j-1} \psi(A \to B\ C, i, k, j) \cdot \alpha(B, i, k) \cdot \alpha(C, k, j) \tag{4.13}$$

Note that we are considering a CFG in which the rule set is complete, i.e.

$$\langle A \to B\ C \rangle \in R \text{ for each } (A, B, C) \in N^3, \tag{4.14}$$

[1] http://www.cs.columbia.edu/~mcollins/io.pdf

and recall that the labels $B$ and $C$ do not appear in the scoring functions in **??**. These facts will allow us to simplify the expression in formula 4.13 as

$$\begin{aligned}\alpha(A,i,j) &= \sum_{B\in N}\sum_{C\in N}\sum_{k=i+1}^{j-1}\tilde{s}(i,j,A)\cdot\alpha(B,i,k)\cdot\alpha(C,k,j)\\ &= \tilde{s}(i,j,A)\cdot\sum_{k=i+1}^{j-1}\sum_{B\in N}\alpha(B,i,k)\cdot\sum_{C\in N}\alpha(C,k,j)\\ &= \tilde{s}(i,j,A)\cdot\sum_{k=i+1}^{j-1}S(i,k)\cdot S(k,j)\end{aligned}$$

where we've introduced a number of notational abbreviations

$$\begin{aligned}\tilde{s}(i,j,A) &= \exp(s(i,j,A))\\ S(i,j) &= \sum_{A\in N}\alpha(A,i,j)\end{aligned}$$

Note that this is the exact same formula as **??**.

From equation 4.17 we can deduce that we in fact do even need to store the values $\alpha(i,j,A)$ but that it suffices to only store the marginalized values $S(i,j)$. In this case, the recursion simplifies even further:

$$\begin{aligned}S(i,j) &= \sum_{A\in N}\alpha(A,i,j)\\ &= \sum_{A\in N}\tilde{s}(i,j,A)\cdot\sum_{k=i+1}^{j-1}S(i,k)\cdot S(k,j)\\ &= \left[\sum_{A\in N}\tilde{s}(i,j,A)\cdot\right]\left[\sum_{k=i+1}^{j-1}S(i,k)\cdot S(k,j)\right]\end{aligned}$$

where we put explicit brackets to emphasize that independence of the subproblems of labeling and splitting. We can now recognize this as the 'inside' equivalent of the expression from the paper[2]

$$s_{best}(i,j) = \max_{\ell}[s(i,j,\ell)] + \max_{k}[s_{split}(i,k,j)]. \tag{4.17}$$

[2] I believe there is actually an error in this equation: it should read $s(i,j,\ell) + s_{span}(i,j)$ instead of just $s(i,j,\ell)$. This is implied by the score for a single node, which is given by equation **??**, taken directly from the paper.

The recursions are the same; the semirings are different. The viterbi recursion given above is in the ViterbiSemiring, which uses the max operator as $\oplus$; the inside recursion given in 4.17 has standard addition $(+)$ instead.

### 4.2.2 Outside recursion

$$\begin{aligned}
\beta(A,i,j) &= \sum_{B\to CA\in R}\sum_{k=1}^{i-1}\psi(B\to CA,k,i-1,j)\cdot\alpha(C,k,i-1)\cdot\beta(B,k,j) \\
&\quad + \sum_{B\to AC\in R}\sum_{k=j+1}^{n}\psi(B\to A,C,i,j,k)\cdot\alpha(C,j+1,k)\cdot\beta(B,i,k) \\
&= \sum_{B\in N}\sum_{C\in N}\sum_{k=1}^{i-1}\psi(B,k,j)\cdot\alpha(C,k,i-1)\cdot\beta(B,k,j) \\
&\quad + \sum_{B\in N}\sum_{C\in N}\sum_{k=j+1}^{n}\psi(B,i,k)\cdot\alpha(C,j+1,k)\cdot\beta(B,i,k) \\
&= \sum_{k=1}^{i-1}\left[\sum_{B\in N}\psi(B,k,j)\cdot\beta(B,k,j)\right]\cdot\left[\sum_{C\in N}\alpha(C,k,i-1)\right] \\
&\quad + \sum_{k=j+1}^{n}\left[\sum_{B\in N}\psi(B,i,k)\cdot\beta(B,i,k)\right]\cdot\left[\sum_{C\in N}\alpha(C,j+1,k)\right] \\
&= \sum_{k=1}^{i-1}S'(k,j)\cdot S(k,i-1) + \sum_{k=j+1}^{n}S'(i,k)\cdot S(j+1,k)
\end{aligned}$$

where

$$\begin{aligned}
S(i,j) &= \sum_{A\in N}\alpha(A,i,j) \\
S'(i,j) &= \sum_{A\in N}\psi(A,i,j)\beta(A,i,j)
\end{aligned}$$

## 4.3 Experiments

We perform three types of experiments with the CRF parser:

- We show that the model is a good supervised parser. We train the model supervised on the PTB and show the f-score on the PTB test set.

- We evaluate the joint RNNG with samples from the CRF parser. We compare the perplexity and fscore with RNNG case.

- We evauluate 'how good the model is' as a sampler.

**Supervised model** We investigate the following.

- We have some optimization and hyperparameter choices here. The original paper uses Adam with 0.001 and a LSTM of dimension 250, which gives the model around 2.5 million parameters. For the discriminative RRNG we use SGD with 0.1, and hidden sizes of 128 gives the model around 800,000 parameters.

- I suggest two experiments: (1) use the default setting from (Stern *et al.*, 2017) and (2) use the settings for the RNNG with a hidden size to match the 800,000 parameters.

**Proposal model** We investigate the following:

- We evaluate validation F-score and perplexity.

- We evaluate F-score with 100 samples (as many proposal trees as possible).

- We evaluate perplexity with varying number of samples: 1 (argmax), 10, 20, 50, 100 (default). The peplexity evaluation with the argmax prediction gives an impression of the uncertaty in the model (**Buys+2018**).

- We perform learning rate decay and model selection based on a development score computed with the samples from the discriminative RNNG. Undecided: should we train a separate joint RNNG with CRF samples?

**Sampler** We investigate the following:

- We asses the conditional entropy of the model. This is most quantitative. Recall that conditional entropy is defined as

$$\mathrm{H}(Y|X) = \sum_{x \in \mathcal{X}} p_X(x) \mathrm{H}(Y|X = x), \tag{4.18}$$

  where

$$\mathrm{H}(Y|X = x) = -\sum_{y \in \mathcal{Y}} p_{Y|X}(y|x) \log p_{Y|X}(y|x). \tag{4.19}$$

  The quantity $\mathrm{H}(Y|X = x)$ can computed exactly with the CRF parser. We estimate the quantity $\mathrm{H}(Y|X)$ by a sum over the development dataset. For the probabilities $p_X(x)$ we use the marginalized probabilities of the joint RNNG (with samples from the CRF parser $p_{Y|X}$).

- We asses for some cherry picked sentences. This is more qualitative. These sentences should be difficult or ambiguous. Or they can be ungramatical when taken from the syneval dataset. We can evaluate their entropy, and the diversity of samples, for example to see if there are clear modes. We can make violinplots of the probabilities of the samples. We can compute the f-scores of the samples compared with the argmax tree.

## 4.4 Related work

Here I describe related work, and in particular earlier approaches to (neural) CRF-parsing.

1. Of course (Stern *et al.*, 2017)
2. CRFs (Sutton and McCallum, 2012)
3. CRF parsing with linear and nonlinear features (Finkel *et al.*, 2008; Durrett and Klein, 2015)
4. Attempts to simplify the grammar and thus the state-space of the dynamic program (Hall *et al.*, 2014).

5. Recent extension of Stern *et al.* (2017), with same model but different features (Kitaev and Klein, 2018).

# 5

# Semisupervised learning

In this chapter we show how the RNNG can be trained on unlabeled data. Together with the regular, supervised, objective, this derives a way to perform semisupervised training.

- I formulate an unsupervised objective for the RNNG that we can combine with the supervised objective to perfom semisupervised training.
- I introduce an approximate posterior in the form of a discriminative parser and derive a variational lower bound on the unsupervised objective.
- I show how to obtain gradients for this lowerbound by rewrinting the gradient into a form that is called the score function estimator (Williams, 1992; Fu, 2006).

**Notation** In this chapter we write $\mathbf{x}$ for a sentence, $\mathbf{y}$ for a (latent) constituency tree, and $\mathcal{Y}(\mathbf{x})$ for the *yield* of $\mathbf{x}$, all trees that can be assigned to $\mathbf{x}$. Furthermore, let $\mathcal{D}_L = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ be a labeled dataset of sentences $\mathbf{x}$ with gold trees $\mathbf{y}$, and let $\mathcal{D}_U = \{\mathbf{x}_i\}_{i=1}^M$ be an unlabeled dataset consisting of just sentences $\mathbf{x}$. We denote our generative RNNG

with $p_\theta$ and when we write $q_\lambda$ we will mean either the discriminative RNNG or the CRF parser.

## 5.1 Objective

We define the following general semi-supervised objective

$$\mathcal{L}(\theta, \lambda) \triangleq \mathcal{L}_S(\theta) + \mathcal{L}_U(\theta, \lambda).$$

The supervised objective $\mathcal{L}_S$ is optimized over $\mathcal{D}_L$ and $\mathcal{L}_U$ the unsupervised objective optimized over $\mathcal{D}_U$. We introduce $\alpha \in \mathbb{R}_{\geq 0}$ as an arbitrary scalar controlling the contribution of the unsupervised objective.

**Supervised objective** We define the supervised objective $\mathcal{L}_S(\theta)$ as

$$\mathcal{L}_S(\theta) \triangleq \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_L} \log p_\theta(\mathbf{x}, \mathbf{y})$$

This objective is optimized as usual using stochastic gradient estimates:

$$\nabla_\theta \mathcal{L}_S(\theta) \approx \frac{N}{K} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{B}} \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{y}),$$

where $\mathcal{B} \subseteq \mathcal{D}_U$ is a mini-batch of size $K$ sampled uniformly from the dataset. We rely on automatic differentiation to compute $\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{y})$ (**baydin2017automatic**).

**Unsupervised objective** We define the unsupervised objective $\mathcal{L}_U(\theta, \lambda)$ as

$$\begin{aligned}\mathcal{L}_U(\theta, \lambda) &\triangleq \sum_{\mathbf{x}\in\mathcal{D}_U} \log p(\mathbf{x}) \\ &= \sum_{\mathbf{x}\in\mathcal{D}_U} \log \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} p_\theta(\mathbf{x}, \mathbf{y})\end{aligned}$$

This is a language modelling objective, in which we treat $\mathbf{y}$ as latent. We have noted the a consequence the lack of independence assumptions

of the RNNG is that the sum over trees $\mathbf{y}$ is not tractable. To optimize this objective we must thus fall back on approximate methods.

## 5.2 Variational approximation

We optimize the unsupervised objective using variational inference (Blei *et al.*, 2016). We introduce a posterior $q_\lambda(\mathbf{y}|\mathbf{x})$ parametrised by $\lambda$ and use Jensen's inequality to derive a variational lower bound on the objective $\mathcal{L}_U(\theta, \lambda)$. First we bound the likelihood of one $\mathbf{x}$ in $\mathcal{D}_U$

$$\begin{aligned}
\log p(\mathbf{x}) &= \log \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} q_\lambda(\mathbf{y}|\mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{y})}{q_\lambda(\mathbf{y}|\mathbf{x})} \\
&= \log \mathbf{E}_q \left[ \frac{p_\theta(\mathbf{x}, \mathbf{y})}{q_\lambda(\mathbf{y}|\mathbf{x})} \right] \\
&\geq \mathbf{E}_q \left[ \log \frac{p_\theta(\mathbf{x}, \mathbf{y})}{q_\lambda(\mathbf{y}|\mathbf{x})} \right] \\
&= \mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) - \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big]
\end{aligned}$$

and write

$$\begin{aligned}
\mathcal{E}(\theta, \lambda) &\triangleq \sum_{\mathbf{x} \in \mathcal{D}_U} \mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) - \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big] \\
&\leq \sum_{\mathbf{x} \in \mathcal{D}_U} \log p(\mathbf{x}) \\
&= \mathcal{L}_U(\theta, \lambda) \qquad (5.1)
\end{aligned}$$

as a lower bound on our true objective $\mathcal{L}_U$. This is a particular instance of the evidence lower bound (ELBO) (Blei *et al.*, 2016). The quantity can be rewritten to reveal an entropy term $\mathrm{H}(q)$

$$\begin{aligned}
\mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) - \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big] &= \mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) \Big] - \mathbf{E}_q \Big[ \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big] \\
&= \mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) \Big] + \mathrm{H}(q),
\end{aligned}$$

which gives this objective an intuitive interpretation. On the one hand, the objective aims to

**Posterior** The posterior $q$ can be any kind of models, with the only condition that for all $\mathbf{x}$ and $\mathbf{y} \in \mathcal{Y}(\mathbf{x})$,

$$p(\mathbf{x}, \mathbf{y}) > 0 \Rightarrow q(\mathbf{y}|\mathbf{x}) > 0.$$

This condition is fulfilled by any discriminatively trained parser with the same support as the joint RNNG $p$. We have two obvious choices at hand: the discriminatively trained RNNG, and the CRF parser that we introduced in chapter 4.

An interesting advantage of the CRF parser is that we *can* compute the entropy $\mathrm{H}(q)$ exactly. This contrasts with the discriminative RNNG, where $\mathrm{H}(q)$ can only be approximated. To make this explicit we introduct separate ELBO objectives:

$$\mathcal{E}_{\text{RNNG}}(\theta, \lambda) \triangleq \sum_{\mathbf{x} \in \mathcal{D}_U} \mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) - \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big] \tag{5.2}$$

$$\mathcal{E}_{\text{CRF}}(\theta, \lambda) \triangleq \sum_{\mathbf{x} \in \mathcal{D}_U} \mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) \Big] + \mathrm{H}(q). \tag{5.3}$$

## 5.3 Optimization

Just like the supervised objective $\mathcal{L}_U$ we optimize the lower bound $\mathcal{E}$ by gradient optimization, which means that we need to compute the gradients $\nabla_\theta \mathcal{E}(\theta, \lambda)$ and $\nabla_\lambda \mathcal{E}(\theta, \lambda)$.

**Gradients of joint parameters** The first gradient is easy and permits a straightforward Monte-Carlo estimate:

$$\begin{aligned}
\nabla_\theta \mathcal{E}(\theta, \lambda) &= \nabla_\theta \mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) - \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big] \\
&= \mathbf{E}_q \Big[ \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{y}) \Big] \\
&\approx \frac{1}{K} \sum_{i=1}^{K} \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{y}_i)
\end{aligned}$$

where $y_i \sim q_\lambda(\cdot|\mathbf{x})$ for $i = 1, \ldots, K$ are samples from the approximate posterior. We can move the gradient inside the expectation because $q$ does not depend on $\theta$, and note that $\nabla_\theta \log q_\lambda(\mathbf{y}|\mathbf{x}) = 0$.

**Gradients of posterior parameters** The second gradient is not so straightforward and requires us to rewrite the objective into a form that is called the *score function estimator* (Fu, 2006). Firstly we define a *learning signal*

$$L(\mathbf{x}, \mathbf{y}) \triangleq \log p_\theta(\mathbf{x}, \mathbf{y}) - \log q_\lambda(\mathbf{y}|\mathbf{x}), \tag{5.4}$$

and use the identity in equation D.1 that we derive in the appendix

$$\begin{aligned}\nabla_\lambda \mathcal{E}(\theta, \lambda) &= \nabla_\lambda \mathbf{E}_q \Big[L(\mathbf{x}, \mathbf{y})\Big] \\ &= \mathbf{E}_q \Big[L(\mathbf{x}, \mathbf{y}) \nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x})\Big].\end{aligned}$$

In this rewritten form the gradient is in the form of an expectation, and that does permit a straightforward MC estimate:

$$\mathbf{E}_q \Big[L(\mathbf{x}, \mathbf{y}) \nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x})\Big] \approx \frac{1}{K} \sum_{i=1}^{K} L(\mathbf{x}, \mathbf{y}_i) \nabla_\lambda \log q_\lambda(\mathbf{x}|\mathbf{y}_i) \tag{5.5}$$

where again $y_i \sim q_\lambda(\cdot|\mathbf{x})$ for $i = 1, \ldots, K$ are independently sampled from the approximate posterior. This estimator has been derived in slightly different forms in Williams (1992), Paisley *et al.* (2012), Mnih and Gregor (2014), Ranganath *et al.* (2014), and Miao and Blunsom (2016) and is also known as the REINFORCE estimator (Williams, 1992).

## 5.4 Variance reduction

We introduce the two baselines:

- Feedforward baseline (Miao and Blunsom, 2016).
- Argmax baseline from Rennie *et al.* (2017) which is exact in the CRF, and approximate in the RNNG.
- The CRF has no variance in estimating the entropy.

## 5.5 Experiments

- Experiments with the two baselines and the two posteriors.

- Compare to a simple baseline: supervised learing on mixed gold-silver trees (partially predicted).
- Analyze the variance reduction provided by the different baselines.

## 5.6 Related work

- Discrete latent variables in neural models (Miao and Blunsom, 2016; Yin *et al.*, 2018).
- Semisupervised training for the RNNG (Cheng *et al.*, 2017).
- Argmax baseline Rennie *et al.*, 2017.

# 6

# Syntactic evaluation

Language models are typically evaluated by the perplexity they assign on held out data, and so did we evaluate our language models in the previous chapters. In this chapter we look at alternative evaluation. In particular, we are interested in evaluation that specifically probes the syntactic abilities of language models. To this end we take the dataset introduced by Marvin and Linzen (2018), which presents a comprehensive set of syntactic challenges, and evaluate our models against it. The dataset consists of constructed sentence pairs that differ in only one word, where one sentence is grammatical and the other is not. The task is to assign higher probability to the grammatical sentence. This task can be thought of as soliciting comparative *acceptability judgements*, a key concept in linguistics. We will refer to this dataset as SYNEVAL, for *syntactic evaluation*.

The chapter is organized as follows. First, I introduce the SYNEVALdataset and describe syntactic phenomana that it tests, and I summarize earlier work on syntactic evaluation. I then describe multitask learning as an approach that has been taken to improve language models for this task, and introduce a novel multitask model based on span labeling. I then evaluate all the models introduced in this thesis on this dataset and

describe the results.

## 6.1 Syntactic evaluation

A shortcoming of perplexity is that the metric conflates various sources of succes. A language model can make use of many features of language to predict the probability of a sequence of words. Although we would like the probability of a sentence to depend on high-level phenomena such as syntactic well-formedness or global semantic coherence, a language model can also obtain low peplexity by assiging high probability to collocations and semantic relations that can predicted from local context, since most sentences are grammatically simple (Marvin and Linzen, 2018). Arguably, this conflation is also the appeal: perplexity is a one size fits all metric. But to get a more fine-grained analysis of language models we must resort to fine-grained metrics.

Recently, a series of papers has introduced tasks that specifically evaluate the syntactic abilities of language models (Linzen *et al.*, 2016; Gulordava *et al.*, 2018; Marvin and Linzen, 2018). And that can be very informative. The task introduced by Linzen *et al.* (2016) is to predict the correct conjugation of a verb in long-distance subject-verb agreement—especially in the precense of distracting subjects that intervene. This task has revealed that lower perplexity does not imply greater succes on this task (Tran *et al.*, 2018), and that an explicitly syntactic model like the RNNG significanlty outperforms purely sequential models, especially when the distance between the subject and the verb increases (Kuncoro *et al.*, 2018). This type of agreement is one of the many phenomena evaluated in SYNEVAL.

### 6.1.1 Dataset

The SYNEVALdataset consists of contrastive sentence pairs where one sentence is grammatical and the other is not. These sentences differ in only one word. Let $(\mathbf{x}, \mathbf{x}')$ be this minimal pair, with grammatical sentence $\mathbf{x}$ and an ungrammatical sentence $\mathbf{x}'$. Then a language model $p$ makes a correct prediction on this pair if $p(\mathbf{x}) > p(\mathbf{x}')$.

The classification is based on the probability that the model assigns

to the entire sentence. This means that the task can be applied to grammatical phenomena that are not local, because they that depend not on a single word, or where the phenomena is not purely sequential, which contrasts with the task introduced in Linzen *et al.* (2016) that revolves around the prediction for one word. This approach is also more natural for a model like the RNNG, where the probability of the sentence is computed by marginalizing over all latent structures, whereas individual word probabilities can only be obtained when conditioning on a single structure—for example the predicted parse—as is done in Kuncoro *et al.*, 2018.

The sentence pairs fall into three categories that linguists consider to depend on hierachical syntactic structure (Everaert *et al.*, 2015; Xiang *et al.*, 2009):

1. Subject-verb agreement (The farmer *smiles.*)
2. Reflexive anaphora (The senators embarassed *themselves.*)
3. Negative polarity items (*No* authors have *ever* been famous.)

In each category the dataset contains sentences of increasing difficulty. For example, the distance between two words in a syntactic dependency can be increased by separating them with a prepositional phrase: *The farmer next to the guards smiles.* In this example *the guards* additionally forms a distractor for the proper conjugation of *smiles*, making the example extra challenging.

The dataset is constructed automatically using handcrafted context-free grammars. The lexical rules are finegrained so that the resulting sentences reasonably coherent semantically. In particular there are rules for animate and innanimate objects so a sentence like *The apple laughs* cannot be constructed. The total dataset consists of around 350,000 sentence pairs.

**Categories** We give the entire list of categories with examples, taken from Marvin and Linzen (2018). Most of these categories come in the two version described above: with animate and innanimate subjects and verbs. For the full list of lexical items used in the constructions we refer the reader to appendix A of Marvin and Linzen (2018).

1. **Simple agreement**:
   (a) The farmer *smiles*.
   (b) *The farmer *smile*.
2. **Agreement in a sentential complement**:
   (a) The mechanics said the author *laughs*.
   (b) *The mechanics said the author *laugh*.
3. **Agreement in short VP coordination**:
   (a) The authors laugh and *swim*.
   (b) *The authors laugh and *swims*.
4. **Agreement in long VP coordination**:
   (a) The author knows many different foreign languages and *enjoys* playing tennis with colleagues.
   (b) *The author knows many different foreign languages and *enjoy* playing tennis with colleagues.
5. **Agreement across a prepositional phrase**:
   (a) The author next to the guards *smiles*.
   (b) *The author next to the guards *smile*.
6. **Agreement across a subject relative clause**:
   (a) The author that likes the security guards *laughs*.
   (b) *The author that likes the security guards *laugh*.
7. **Agreement across an object relative**:
   (a) The movies that the guard likes *are* good.
   (b) *The movies that the guard likes *is* good.
8. **Agreement in an object relative**:
   (a) The movies that the guard *likes* are good.
   (b) *The movies that the guard *like* are good.
9. **Simple reflexive anaphora**:
   (a) The author injured *himself*.

(b) *The author injured *themselves*.

10. **Reflexive in sentential complement**:
    (a) The mechanics said the author hurt *himself*.
    (b) *The mechanics said the author hurt *themselves*.
11. **Reflexive across a relative clause**:
    (a) The author that the guards like injured *himself*.
    (b) *The author that the guards like injured *themselves*.
12. **Simple NPI**:
    (a) *No* authors have ever been famous.
    (b) **Most* authors have ever been famous.
13. **NPI across a relative clause**:
    (a) *No* authors that *the* guards like have ever been famous.
    (b) **Most* authors that *no* guards like have ever been famous.
14. **NPI across a relative clause (the)**:
    (a) *No* authors that *the* guards like have ever been famous.
    (b) **The* authors that *no* guards like have ever been famous.

### 6.1.2 Related work

In this section I give a short review the literature on related syntactic evaluations. Linzen *et al.* (2016) introduce the task of long distance subject-verb agreement and Gulordava *et al.* (2018) make this test more challenging by turning the sentences nonsensical while keeping them grammatical. Both datasets are extracted from a wikipedia corpus based on their predicted dependency structure. To make the sentences nonsensical, Gulordava *et al.* (2018) randomly substitute words from the same grammatical category.[1] Warstadt *et al.* (2018) fine-tune neural models to learn to immitate grammatical acceptability judgments gathered from linguistics textbooks. McCoy *et al.* (2018) Train a neural machine translation model to learn how to turn a declarative sentence into a

[1] An approach inspired by Chomsky's (in)famous sentence *Colorless green ideas sleep furiously.*.

question. Linguist have argued that the transformations required to generate one from the other provide strong evidence for the existence of hierarchical structure in language Everaert *et al.*, 2015.[2]

## 6.2 Multitask learning

One method to improve language models for the tasks described in this chapter is multitask learning Collobert and Weston, 2008; Collobert *et al.*, 2011; Zhang and Weiss, 2016; Søgaard and Goldberg, 2016, which has already been applied succesfully to improve the perfomance of neural language models on syntactic evaluation tasks (Enguehard *et al.*, 2017; Marvin and Linzen, 2018). Multitask learning is a simple and effective way of providing additional supervision to neural models by combining multiple tasks in one single objective. The model is thus by encouraged to compute representations that are useful in all the tasks.

In this section I describe two simple baselines for the syntactic evaluation task that are based on multitask learning. Both methods are based on language modelling with a syntactic side objective. The first side objective is to predict combinatory categorical grammar (CCG) supertags (Bangalore and Joshi, 1999) for each word in the sentence, which is proposed in (Enguehard *et al.*, 2017). The second side objective is to label spans of words with labels from a treebank. This objective is inspired by the label scoring function in the CRF parser introduced in chapter 4. This is similar to a recent work in semantic parsing where it this side-objective is is called a 'syntactic scaffold' (Swayamdipta *et al.*, 2018). The exact form of our side-objective is novel, as far as the author is aware of, and is parametrized in a considerably simpler way. Both objectives are challenging and require representations that encode a fair amount of syntactic information.

In our case we combine a language model $p$ with a model $q$ that learns syntactic side-objective, and optimize these jointly over a single

[2]These pairs play a central role as empirical evidence in the argument—known as the *argument from the poverty of the stimulus*—that humans have an innate predisposition for generalizations that rely on hierarchical structure rather than linear order (Chomsky, 1980). Sequence-to-sequence neural machine translation, on the other hand, is a fully sequential model that involves no hierarchical structure or transformations.

labeled dataset $\mathcal{D}$. In this case, the objective is to maximize

$$\mathcal{L}(\theta, \lambda, \xi) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \log p_{\theta,\lambda}(\mathbf{x}) + \log q_{\theta,\xi}(\mathbf{y}|\mathbf{x}) \tag{6.1}$$

with respect to all the parameters $\theta$, $\lambda$ and $\xi$. The key point of multitask learning is that the the two models $p$ and $q$ share the set of parameters $\theta$ and that in objective 6.1 the parameters $\theta$ will be optimized to fit both objectives well. The parameters in $\lambda$ and $\xi$ are optimized to their respective objectives separately. The proportion and the nature of the parameters that belong to $\theta$ is a choice of the modeller and the objective. How we made this choice we specify in the following paragraphs.

**Language model** The main model $p$ is a straightforward RNN language model on sentences $\mathbf{x} = (x_1, \ldots, x_n)$:

$$\log p_{\theta,\lambda} = \sum_{i=1}^{n} \log p_{\theta,\lambda}(x_i \mid \mathbf{x}_{<i}),$$

where the probabilities are computed by linear regression on a feature vector $\mathbf{f}_{i-1}$ as

$$p_{\theta,\lambda}(x_i \mid \mathbf{x}_{<i}) \propto \exp\left[\mathbf{W}^\top \mathbf{f}_{i-1} + \mathbf{b}\right]_{x_i}.$$

where the parameters $\xi = \{\mathbf{W}, \mathbf{b}\}$ are specific to the language model. The features $\mathbf{f}_i$ are computed using a forward RNNparametrized by $\theta$,

$$[\mathbf{f}_1, \ldots, \mathbf{f}_n] = \text{RNN}_\theta^f(\mathbf{x}).$$

These features are also used in the side objective, and it is in this precise sense that the parameters $\theta$ are shared between $p$ and $q$.

**Word labeling** Let $\mathbf{y} = (y_1, \ldots, y_n)$ be a sequence of CCG supertags for the sentence $\mathbf{x}$, with one tag for each word. The side model is then a simple greedy tagging model:

$$\log q_{\theta,\xi}(\mathbf{y} \mid \mathbf{x}) = \sum_{i=1}^{n} \log q_{\theta,\xi}(y_i \mid \mathbf{x}).$$

The probability over tags for position $i$ are computed from $\mathbf{f}_i$ using a feedforward network parametrized by $\xi$

$$q_{\theta,\xi}(y_i \mid \mathbf{x}) \propto \exp\Big[\textsc{FeedForward}_\xi(\mathbf{f}_i)\Big]_{y_i}.$$

**Span labeling** Let $\mathbf{y}$ be a sequence of labeled spans $\mathbf{y}_k = (\ell_k, i_k, j_k)$ that are obtained from a gold parse tree for $\mathbf{x}$. Then the side model $q$ predicts a label $\ell$ for a given sentence $\mathbf{x}$ and span endpoints $i$ and $j$:

$$\begin{aligned}\log q_{\theta,\xi}(\mathbf{y} \mid \mathbf{x}) &= \sum_{k=1}^{K} \log q_{\theta,\xi}(\mathbf{y}_k \mid \mathbf{x}) \\ &= \sum_{k=1}^{K} \log q_{\theta,\xi}(\ell_k \mid \mathbf{x}, i_k, j_k).\end{aligned}$$

We define a span feature as in the CRF parser

$$\mathbf{s}_{ij} = \mathbf{f}_j - \mathbf{f}_j$$

and compute a distribution over labels using a feedforward network parametrized by $\xi$

$$q_{\theta,\xi}(\ell \mid \mathbf{x}, i, j) \propto \exp\Big[\textsc{FeedForward}_\xi(\mathbf{s}_{ij})\Big]_{\ell}.$$

## 6.3 Experiments

### 6.3.1 Setup

### 6.3.2 Results

# 7

# Conclusion

Here is a narrative summary of what I have shown in this thesis.

## 7.1 Main contributions

The main $n$ contributions of thesis are:

**Global training of a chart based neural parser.** Here I describe what that entails.

**Semisupervised training of RNNGs.** Here I describe what that entails.

**Effective baselines for the score function estimator.** Here I describe what that entails.

## 7.2 Future work

We have identified possibilities for future work:

**Something.** Here I describe what that entails.

# Appendices

# A

# Figures

In this appendix I will put figures, for cases where there are just too many. For example:

- The barplots of the syntactic evaluation
- The training losses for the various models
- The valuation perplexity and f-score during training.

## A.1 Training plots

We show training plots that are means with standard deviation bands over 10 runs. We have two types of plots: losses and development scores.

**Development scores** Plots with development scores.

- DiscRNNG + GenRNNG-disc + GenRNNG-crf + CRF (small) development fscores.
- CRF parser lossses: our (128d + SGD) and original (250d + Adam).

- GenRNNG-disc + GenRNNG-crf development perplexity
- Language models: lstm + lstm-ccg + lstm-span development perplexity
- GenRNNG + LSTM development perplexity.

**Training losses** Plots with training losses.

### A.2 Test scores

Here we show violin plots that show the distribution of the test scores.

### A.3 Samples

Here we put figures to illustrate the samples.

### A.4 Syntactic evaluation

Here we put more bar-charts for the syntactic evaluation.

# B

# Implementation

In this appendix I summarize all choices made concerning data, implementation, and optimization.

- We choose simplicity over maximal performance: minimal UNKing; embeddings learned from scratch; basic SGD optimization.
- We surmise that more elaborate UNKin, or more elaborate embeddings should improve performance,
- Our goal however is to compare in the most vanilla setting possible, to maximally focus on the essential differences between the approaches.

## B.1 Data

**Penn Treebank** We follow Stern *et al.* (2017) in all choices of data processing and unk-ing. Most importantly, we use only a single unknown token, and perform dynamic

- We use the Penn Treebank (Marcus *et al.*, 1993) for our experiments, with standard splits of sections 2-21 for training, section 22 for development, and section 23 for testing.

- We perform no token preprocessing, and only a single `<UNK>` token is used for unknown words at test time.
- For all discriminative models the vocabulary contains all words in the training data and words are replaced by `<UNK>` with probability $1/(1 + \text{freq}(w))$ during training, where $\text{freq}(w)$ is the frequency of $w$ in the training data.
- For all generative models the vocabulary contains all words with a count greater that in the training data and words are replaced by `<UNK>` with probability $1/(1 + \text{freq}(w))$ during training, where $\text{freq}(w)$ is the frequency of $w$ in the training data.
- We use automatically predicted tags for training and testing, obtaining predicted part-of- speech tags for the Penn Treebank using the Stan- ford tagger (Toutanova et al., 2003) with 10-way jackknifing, and using the provided predicted part- of-speech and morphological tags for the French Treebank.

**Penn Treebank**

## B.2 Implementation

- Implemented in Dynet (Neubig *et al.*, 2017).
- Surrogate objective and gradient blocking Schulman *et al.*, 2015.
- All embeddings are learned from random initialization. More elaborate word embeddings (GloVE, FastText, ELMO) will no doubt improve results, but this is not the aim of our researcg.

## B.3 Hyperparameters

For the RNNG we follow exactly the hyperparameters from the published papers. For the CRF parser

-

## B.4 Optimization

I describe the choices made with regards to optimization.

- Automatic differentiation (Baydin *et al.*, 2018) for gradients.
- SGD and not Adam or other adaptive methods (Wilson *et al.*, 2017) and choices of hyperparameters.
- Learning rate schedule based on development scores.

# C

## Conditional Random Fields

In this appendix I describe CRF's for factor graphs, together with the message passing algorithms from which forward-backward and inside-outside are derived.

# D

# Variational Inference

In this appendix we give an account of variational inference in general of amortized variational inference in particular. We focus on amortized inference with discrete latent variables, and in particular when the variables are structured. We then derive the score function gradient, as used in chapter 5, and we describe techniques to reduce the variance of this estimator.

## D.1 Variational Inference

We will write some generic things about variational inference.

- Variational inference for exponential families, with conjugate priors (Jordan *et al.*, 1999; Wainwright and Jordan, 2008; Blei *et al.*, 2016).
- With amortized inference, using neural networks, for non-conjugate models (Kingma and Welling, 2014; Rezende *et al.*, 2014) and in particular the reparametrization trick that makes these models efficiently trainable.
- Reparametrization for discrete latent variables (Maddison *et al.*,

2017; Jang *et al.*, 2017).

- The generalization of this reparametrization trick in automatic differentiation variational inference (Kucukelbir *et al.*, 2017).
- Black box variational inference, which uses the same combination of score function gradient with variance reduction that we resort to (Ranganath *et al.*, 2014).

### D.2 Score function estimator

In this section we provide a detailed derivation of the score function estimator:

$$\nabla_\lambda \mathbf{E}_q \Big[ L(\mathbf{x}, \mathbf{y}) \Big] = \mathbf{E}_q \Big[ L(\mathbf{x}, \mathbf{y}) \nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big] \tag{D.1}$$

where

$$L(\mathbf{x}, \mathbf{y}) \triangleq \log p_\theta(\mathbf{x}, \mathbf{y}) - \log q_\lambda(\mathbf{y}|\mathbf{x})$$

$$\begin{aligned}
\nabla_\lambda \mathbf{E}_q \Big[ L(\mathbf{x}, \mathbf{y}) \Big] &= \nabla_\lambda \mathbf{E}_q \Big[ \log p_\theta(\mathbf{x}, \mathbf{y}) - \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big] \\
&= \nabla_\lambda \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big\{ q_\lambda(\mathbf{y}|\mathbf{x}) \log p_\theta(\mathbf{x}, \mathbf{y}) - q_\lambda(\mathbf{y}|\mathbf{x}) \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big\} \\
&= \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big\{ \nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x}) \log p_\theta(\mathbf{x}, \mathbf{y}) - \nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x}) \log q_\lambda(\mathbf{y}|\mathbf{x}) \\
&\qquad - q_\lambda(\mathbf{y}|\mathbf{x}) \nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big\} \\
&= \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big\{ \nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x}) \log p_\theta(\mathbf{x}, \mathbf{y}) - \nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x}) \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big\} \\
&= \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big\{ L(\mathbf{x}, \mathbf{y}) \nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x}) \Big\} \\
&= \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big\{ L(\mathbf{x}, \mathbf{y}) q_\lambda(\mathbf{y}|\mathbf{x}) \nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big\} \\
&= \mathbf{E}_q \Big[ L(\mathbf{x}, \mathbf{y}) \nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x}) \Big].
\end{aligned}$$

In this derivation we used the identity

$$\nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x}) = q_\lambda(\mathbf{y}|\mathbf{x}) \nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x}),$$

which follows from the derivative

$$\nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x}) = \nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x}) q_\lambda(\mathbf{y}|\mathbf{x})^{-1}.$$

We

$$\begin{aligned}
\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} q_\lambda(\mathbf{y}|\mathbf{x})\nabla_\lambda \log q_\lambda(\mathbf{y}|\mathbf{x}) &= \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} q_\lambda(\mathbf{y}|\mathbf{x}) \frac{\nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x})}{q_\lambda(\mathbf{y}|\mathbf{x})} \\
&= \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \nabla_\lambda q_\lambda(\mathbf{y}|\mathbf{x}) \\
&= \nabla_\lambda \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} q_\lambda(\mathbf{y}|\mathbf{x}) \\
&= \nabla_\lambda 1 \\
&= 0.
\end{aligned}$$

## D.3 Optimization

We use automatic differentiation (**baydin2017automatic**) to obtain all our gradients. In order to obtain the gradients in formula D.1 using this method we rewrite it in the form of a *surrogate objective* (Schulman *et al.*, 2015):

$$\mathcal{L}_{\text{SURR}}(\theta, \lambda) = \frac{1}{K}\sum_{i=1}^{K} \log q_\lambda(\mathbf{x}|\mathbf{y}_i)\textsc{BlockGrad}(L(\mathbf{x}, \mathbf{y}_i)). \tag{D.2}$$

The function BLOCKGRADdetaches a node from its upstream computation graph. This turns it effectively into a scalar. More precisely, let $f$ be function (computed by a node in the computation graph) with parameters $\theta$ and input $\mathbf{x}$, then

$$\textsc{BlockGrad}(f_\theta(\mathbf{x})) \triangleq f(\mathbf{x}),$$

such that

$$\nabla_\theta \textsc{BlockGrad}(f_\theta(\mathbf{x})) = \nabla_\theta f(\mathbf{x}) = 0.$$

Automatic differentiation of equation D.2 with respect to $\lambda$ will give us the exact expression we are looking for

$$\nabla_\lambda \mathcal{L}_{\text{SURR}}(\theta, \lambda) = \frac{1}{K} \sum_{i=1}^{K} L(\mathbf{x}, \mathbf{y}) \nabla_\lambda \log q_\lambda(\mathbf{x}|\mathbf{y}_i),$$

hence the adjective *surrogate*.

## D.4 Variance reduction

We have derived an estimator for the gradient of the posterior parameters in the unsupervised objective. This estimator is unbiased, but is known to have high variance, often too much to be useful (Paisley *et al.*, 2012). Two effective methods to counter this are control variates and baselines (Ross, 2006).

**Variance of estimator** First, let's analyze the variance of our estimator. Note that our expectation is of the general form

$$\mu \triangleq \mathbf{E}\left[f(X)\right]$$

and that we estimate this quantity by generating $n$ independent samples $X_1, \ldots, X_n \sim P(X)$ and computing

$$\hat{\mu} \triangleq \frac{1}{n} \sum_{i=1}^{n} f(X_i).$$

This is an unbiased estimator for $\mu$ with error

$$\left[(\mu - \hat{\mu})^2\right] = \mathbf{var}\left[\hat{\mu}\right] = \frac{\mathbf{var}\left[\hat{\mu}\right]}{n},$$

which means that the error

$$\mu - \hat{\mu} = O\left(\sqrt{\frac{\mathbf{var}\left[\hat{\mu}\right]}{n}}\right)$$

and reducing it linearly requires a quadratic number of samples.

In our particular case, the function $f$ is

$$f_{X=x}(Y) \triangleq L(X, Y) \nabla_\lambda \log q_\lambda(Y|X = x)$$

where we have made explicit that $y$ is the random variable, and $x$ is given.

**Control variates** Consider a function $g(X)$ with known expectation

$$\mu_g \triangleq \mathbf{E}\left[g(X)\right]$$

Then we can define a new function $\hat{f}$ such that

$$\hat{f}(X) \triangleq f(X) - g(X) + \mu_g.$$

This function is also an estimator for $\mu$, since

$$\begin{aligned}\mathbf{E}\left[\hat{f}(X)\right] &= \mathbf{E}\left[f(X)\right] - \mu_g + \mu_g \\ &= \mathbf{E}\left[f(X)\right],\end{aligned}$$

and a computation shows that the variance of the new function is

$$\begin{aligned}\mathbf{var}\left[\hat{f}(X)\right] &= \mathbf{E}\left[(f(X) - g(X) + \mu_g) - \mu)^2\right] \\ &= \mathbf{E}\left[(f(X) - g(X) + \mu_g)^2\right] - 2\,\mathbf{E}\left[(f(X) - g(X) + \mu_g)\mu\right] + \mathbf{E}\left[\mu^2\right] \\ &= \mathbf{E}\left[(f(X) - g(X) + \mu_g)^2\right] - 2\,\mathbf{E}\left[(f(X) - g(X) + \mu_g)\right]\mu + \mu^2 \\ &= \mathbf{E}\left[(f(X) - g(X) + \mu_g)^2\right] - 2\mu^2 + \mu^2 \\ &= \mathbf{E}\left[f(X)^2 + g(X)^2 + \mu_g^2 - 2f(X)g(X) + 2f(X)\mu_g - 2g(X)\mu_g\right] - \mu^2 \\ &= \mathbf{E}\left[f(X)^2\right] - \mathbf{E}\left[f(X)\right]^2 \\ &\qquad - 2(\mathbf{E}\left[f(X)g(X)\right] - \mathbf{E}\left[f(X)\right]\mathbf{E}\left[g(X)\right]) \\ &\qquad + \mathbf{E}\left[g(X)^2\right] - \mathbf{E}\left[g(X)\right]^2 \\ &= \mathbf{var}\left[f(X)\right] - 2\,\mathbf{cov}\left[f(X), g(X)\right] + \mathbf{var}\left[g(X)\right]\end{aligned}$$

This means we can get a reduction in variance whenever

$$\mathbf{cov}\left[f(X), g(X)\right] > \frac{1}{2}\,\mathbf{var}\left[g(X)\right].$$

The function $g$ is called a *control variate*—it allows us to control the variance of $f$.

From the equality above we can see that this will be the case whenever $f(X)$ and $g(X)$ are strongly correlated. Our choice of control

variate will be made with the that in mind. Furthermore, $\mathbf{E}\left[g(X)\right]$ must be known. What is an optimal control variate? Typically a control variate of the form $ag$ is chosen with fixed, and $a$ is optimized to maximize the correlation. This brings us to the generic formulation of a control variate:

$$\hat{f}(X) \triangleq f(X) - a(g(X) - \mathbf{E}\left[g(X)\right])$$

with variance

$$\mathbf{var}\left[\hat{f}(X)\right] = \mathbf{var}\left[f(X)\right] - 2a\,\mathbf{cov}\left[f(X), g(X)\right] + a^2\,\mathbf{var}\left[g(X)\right]$$

We take a derivative of this with respect to $a$

$$\frac{d}{da}\,\mathbf{var}\left[\hat{f}(X)\right] = -2\,\mathbf{cov}[f(X), g(X)\right] + 2a\,\mathbf{var}\left[g(X)\right]$$

Setting this to zero and solving for $a$ we obtain the optimal choice for $a$

$$a = \frac{\mathbf{cov}\left[f(X), g(X)\right]}{\mathbf{var}\left[g(X)\right]}. \tag{D.3}$$

Plugging in this solution into the expression for $\mathbf{var}\left[\hat{f}(X)\right]$ and dividing by $\mathbf{var}\left[f(X)\right]$ we get

$$\frac{\mathbf{var}\left[\hat{f}(X)\right]}{\mathbf{var}\left[f(X)\right]} = 1 - \frac{\mathbf{cov}[f(X), g(X)\right]}{\mathbf{var}\left[f(X)\right]\mathbf{var}\left[g(X)\right]} \tag{D.4}$$

$$= 1 - \mathbf{corr}^2\left[f(X), g(X)\right], \tag{D.5}$$

which shows that given this choice of $a$ the reduction in variance is directly determined by the correlation between $f(X)$ and $g(X)$.

Bringing this all together, we let our new estimator be

$$\mathbf{E}\left[f(X)\right] = \mathbf{E}\left[\hat{f}(X)\right] \approx \frac{1}{n}\sum_{i=1}^{n}[f(X_i) - ag(X_i)\right] - \mu_g$$

**Example** (Ross, 2006) Suppose we want to use simulation to determine

$$\mathbf{E}\left[f(X)\right] = \mathbf{E}\left[e^X\right] = \int_0^1 e^x dx = e - 1$$

with $X \sim \mathcal{U}(0,1)$. A natural control variate to use in this case is the random variable $X$ itself: $g(X) \triangleq X$. We thus define the new estimator

$$\begin{aligned}\hat{f}(X) &= f(X) - g(X) + \mathbf{E}\left[g(X)\right] \\ &= e^X - X + \frac{1}{2}.\end{aligned}$$

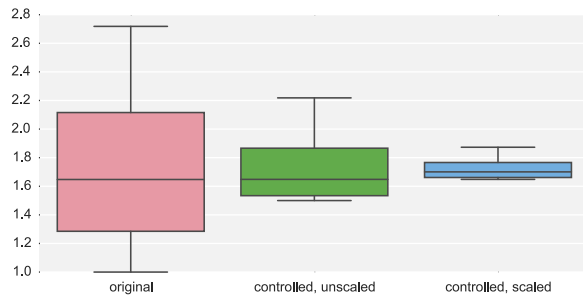
To compute the decrease in variance with this new estimator, we first note that

$$\begin{aligned}\mathbf{cov}(e^X, X) &= \mathbf{E}\left[Xe^X\right] - \mathbf{E}\left[X\right]\mathbf{E}\left[e^X\right] \\ &= \int_0^1 xe^x dx - \frac{e-1}{2} \\ &= 1 - \frac{e-1}{2} \approx 0.14086 \\ \mathbf{var}\left[e^X\right] &= \mathbf{E}\left[e^{2X}\right] - (\mathbf{E}\left[e^X\right])^2 \\ &= \int_0^1 e^{2x} dx - (1-e^x)^2 \\ &= \frac{e^2-1}{2} - (1-e^x)^2 \approx 0.2420 \\ \mathbf{var}\left[X\right] &= \mathbf{E}\left[X^2\right] - (\mathbf{E}\left[X\right])^2 \\ &= \int_0^1 x^2 dx - \frac{1}{4} \\ &= \frac{1}{3} - \frac{1}{4} = \frac{1}{12}.\end{aligned}$$

When we choose $a$ as in formula D.3 we can use formula D.4 to compute that

$$\begin{aligned}\frac{\mathbf{var}\left[\hat{f}(X)\right]}{\mathbf{var}\left[f(X)\right]} &= 1 - \frac{(0.14086)^2}{\frac{0.2420}{12}} \\ &\approx 0.0161.\end{aligned}$$

This is a reduction of 98.4 percent! A simulation illustrates what this looks like in practice with ... samples:


2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
original
controlled, unscaled
controlled, scaled

# References


Ballesteros, M., C. Dyer, Y. Goldberg, and N. A. Smith (2017). "Greedy Transition-Based Dependency Parsing with Stack LSTMs". *Computational Linguistics*. 43: 311–347.

Ballesteros, M., Y. Goldberg, C. Dyer, and N. A. Smith (2016). "Training with Exploration Improves a Greedy Stack LSTM Parser". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics. 2005–2010.

Bangalore, S. and A. K. Joshi (1999). "Supertagging: An approach to almost parsing". *Computational linguistics*. 25(2): 237–265.

Baydin, A. G., B. A. Pearlmutter, A. A. Radul, and J. M. Siskind (2018). "Automatic differentiation in machine learning: a survey". *Journal of Marchine Learning Research*. 18: 1–43.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin (2003). "A neural probabilistic language model". *Journal of machine learning research*. 3(Feb): 1137–1155.

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2016). "Variational Inference: A Review for Statisticians". *ArXiv e-prints*. Jan.

Brennan, J. R., E. P. Stabler, S. E. Van Wagenen, W.-M. Luh, and J. T. Hale (2016). "Abstract linguistic structure correlates with temporal activity during naturalistic comprehension". *Brain and language*. 157: 81–94.

Buys, J. and P. Blunsom (2015a). "A Bayesian Model for Generative Transition-based Dependency Parsing". In: *Proceedings of the Third International Conference on Dependency Linguistics, DepLing 2015, August 24-26 2015, Uppsala University, Uppsala, Sweden*. 58–67.

Buys, J. and P. Blunsom (2015b). "Generative incremental dependency parsing with neural networks". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Vol. 2. 863–869.

Buys, J. and P. Blunsom (2018). "Neural Syntactic Generative Models with Exact Marginalization". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Vol. 1. 942–952.

Carnie, A. (2010). *Constituent Structure. Oxford linguistics*. Oxford University Press. ISBN: 9780199583461.

Chelba, C. and F. Jelinek (2000). "Structured language modeling". *Computer Speech & Language*. 14(4): 283–332.

Chen, S. F. and J. Goodman (1999). "An empirical study of smoothing techniques for language modeling". *Computer Speech & Language*. 13(4): 359–394.

Cheng, J., A. Lopez, and M. Lapata (2017). "A Generative Parser with a Discriminative Recognition Algorithm". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics. 118–124.

Chomsky, N. (1980). "Rules and representations". *Behavioral and brain sciences*. 3(1): 1–15.

Christiansen, M. H., C. M. Conway, and L. Onnis (2012). "Similar neural correlates for language and sequential learning: evidence from event-related brain potentials". *Language and cognitive processes*. 27(2): 231–256.

Collins, M. (2003). "Head-driven statistical models for natural language parsing". *Computational linguistics*. 29(4): 589–637.

Collobert, R. and J. Weston (2008). "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 160–167.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa (2011). "Natural language processing (almost) from scratch". *Journal of Machine Learning Research*. 12(Aug): 2493–2537.

Conway, C. M. and D. B. Pisoni (2008). "Neurocognitive basis of implicit learning of sequential structure and its relation to language processing". *Annals of the New York Academy of Sciences*. 1145(1): 113–131.

Durrett, G. and D. Klein (2015). "Neural CRF Parsing". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics. 302–312.

Dyer, C., A. Kuncoro, M. Ballesteros, and N. A. Smith (2016). "Recurrent Neural Network Grammars". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics. 199–209.

Emami, A. and F. Jelinek (2005). "A neural syntactic language model". *Machine learning*. 60(1-3): 195–227.

Enguehard, É., Y. Goldberg, and T. Linzen (2017). "Exploring the Syntactic Abilities of RNNs with Multi-task Learning". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics. 3–14.

Everaert, M. B., M. A. Huybregts, N. Chomsky, R. C. Berwick, and J. J. Bolhuis (2015). "Structures, not strings: linguistics as part of the cognitive sciences". *Trends in cognitive sciences*. 19(12): 729–743.

Finkel, J. R., A. Kleeman, and C. D. Manning (2008). "Efficient, Feature-based, Conditional Random Field Parsing". In: *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics. 959–967.

Frank, S. L., R. Bod, and M. H. Christiansen (2012). "How hierarchical is language use?" *Proceedings of the Royal Society B: Biological Sciences*. 279(1747): 4522–4531.

Fried, D. and D. Klein (2018). "Policy Gradient as a Proxy for Dynamic Oracles in Constituency Parsing". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics. 469–476.

Fu, M. C. (2006). "Gradient Estimation". In: *Handbooks in Operations Research and Management Science (Volume 13)*. Ed. by B. L. N. Edited by Shane G. Henderson. Elsevier. Chap. 19. 575–616.

Giannakidou, A. (2011). "Negative and positive polarity items: Variation, licensing, and compositionality". *The Handbook of Natural Language Meaning (second edition). Mouton de Gruyter, Berlin*.

Gillespie, M. and N. J. Pearlmutter (2011). "Hierarchy and scope of planning in subject–verb agreement production". *Cognition*. 118(3): 377–397.

Gillespie, M. and N. J. Pearlmutter (2013). "Against structural constraints in subject–verb agreement production." *Journal of Experimental Psychology: Learning, Memory, and Cognition*. 39(2): 515.

Goldberg, Y. and J. Nivre (2013). "Training Deterministic Parsers with Non-Deterministic Oracles". *Transactions of the Association for Computational Linguistics*. 1(Oct): 403–414.

Gulordava, K., P. Bojanowski, E. Grave, T. Linzen, and M. Baroni (2018). "Colorless Green Recurrent Networks Dream Hierarchically". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics. 1195–1205.

Hale, J. (2001). "A probabilistic Earley parser as a psycholinguistic model". In: *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Association for Computational Linguistics. 1–8.

Hale, J., C. Dyer, A. Kuncoro, and J. Brennan (2018). "Finding syntax in human encephalography with beam search". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics. 2727–2736.

Hall, D., G. Durrett, and D. Klein (2014). "Less grammar, more features". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 228–237.

He, H., J. Eisner, and H. Daume (2012). "Imitation learning by coaching". In: *Advances in Neural Information Processing Systems*. 3149–3157.

Huddleston, R. D. and G. K. Pullum (2002). *The Cambridge Grammar of the English Language*. Cambridge University Press.

Hudson, R. (2010). *An introduction to word grammar*. Cambridge University Press.

Jang, E., S. Gu, and B. Poole (2017). "Categorical reparameterization with gumbel-softmax". *ICLR*.

Jordan, M., Z. Ghahramani, T. Jaakkola, and L. Saul (1999). "An Introduction to Variational Methods for Graphical Models". *Machine Learning*. 37(2): 183–233.

Kingma, D. P. and M. Welling (2014). "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations*.

Kitaev, N. and D. Klein (2018). "Constituency Parsing with a Self-Attentive Encoder". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics. 2676–2686.

Klein, D. and C. D. Manning (2003). "Accurate Unlexicalized Parsing". In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1. ACL '03*. Sapporo, Japan: Association for Computational Linguistics. 423–430.

Kneser, R. and H. Ney (1995). "Improved backing-off for m-gram language modeling". In: *icassp*. Vol. 1. 181e4.

Kucukelbir, A., D. Tran, R. Ranganath, A. Gelman, and D. M. Blei (2017). "Automatic differentiation variational inference". *The Journal of Machine Learning Research*. 18(1): 430–474.

Kuncoro, A., M. Ballesteros, L. Kong, C. Dyer, G. Neubig, and N. A. Smith (2017). "What Do Recurrent Neural Network Grammars Learn About Syntax?" In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics. 1249–1258.

Kuncoro, A., C. Dyer, J. Hale, D. Yogatama, S. Clark, and P. Blunsom (2018). "LSTMs Can Learn Syntax-Sensitive Dependencies Well, But Modeling Structure Makes Them Better". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics. 1426–1436.

Levy, R. (2008). "Expectation-based syntactic comprehension". *Cognition*. 106(3): 1126–1177.

Linzen, T., E. Dupoux, and Y. Goldberg (2016). "Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies". *Transactions of the Association for Computational Linguistics*. 4: 521–535.

Maddison, C. J., A. Mnih, and Y. W. Teh (2017). "The concrete distribution: A continuous relaxation of discrete random variables". *ICLR*.

Marcus, M. P., M. A. Marcinkiewicz, and B. Santorini (1993). "Building a large annotated corpus of English: The Penn Treebank". *Computational linguistics*. 19(2): 313–330.

Marvin, R. and T. Linzen (2018). "Targeted Syntactic Evaluation of Language Models". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 1192–1202.

McCoy, R. T., T. Linzen, and R. Frank (2018). "Revisiting the poverty of the stimulus: hierarchical generalization without a hierarchical bias in recurrent neural networks". *arXiv preprint arXiv:1802.09091*.

Miao, Y. and P. Blunsom (2016). "Language as a Latent Variable: Discrete Generative Models for Sentence Compression". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics. 319–328.

Mikolov, T., M. Karafiát, L. Burget, J. Černock, and S. Khudanpur (2010). "Recurrent neural network based language model". In: *Eleventh Annual Conference of the International Speech Communication Association*.

Mnih, A. and K. Gregor (2014). "Neural Variational Inference and Learning in Belief Networks". In: *ICML*.

Neubig, G., C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn, *et al.* (2017). "Dynet: The dynamic neural network toolkit". *arXiv preprint arXiv:1701.03980*.

Nivre, J. (2005). "Dependency grammar and dependency parsing". *MSI report*. 5133(1959): 1–32.

Paisley, J., D. Blei, and M. Jordan (2012). "Variational Bayesian Inference with Stochastic Search". In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. Ed. by J. Langford and J. Pineau. *ICML '12*. New York, NY, USA: Omnipress. 1367–1374.

Pauls, A. and D. Klein (2012). "Large-scale syntactic language modeling with treelets". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics. 959–968.

Ranganath, R., S. Gerrish, and D. Blei (2014). "Black box variational inference". In: *Artificial Intelligence and Statistics*. 814–822.

Rennie, S. J., E. Marcheret, Y. Mroueh, J. Ross, and V. Goel (2017). "Self-Critical Sequence Training for Image Captioning". *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 1179–1195.

Rezende, D. J., S. Mohamed, and D. Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. 1278–1286.

Roark, B. (2001). "Probabilistic top-down parsing and language modeling". *Computational linguistics*. 27(2): 249–276.

Ross, S. M. (2006). *Simulation, Fourth Edition*. Orlando, FL, USA: Academic Press, Inc.

Schulman, J., N. Heess, T. Weber, and P. Abbeel (2015). "Gradient estimation using stochastic computation graphs". In: *Advances in Neural Information Processing Systems*. 3528–3536.

Smith, N. A. (2012). "Adversarial evaluation for models of natural language". *arXiv preprint arXiv:1207.0245*.

Søgaard, A. and Y. Goldberg (2016). "Deep multi-task learning with low level tasks supervised at lower layers". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics. 231–235.

Stern, M., J. Andreas, and D. Klein (2017). "A Minimal Span-Based Neural Constituency Parser". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics. 818–827.

Sutton, C. and A. McCallum (2012). "An Introduction to Conditional Random Fields". *Found. Trends Mach. Learn.* 4(4): 267–373.

Swayamdipta, S., S. Thomson, K. Lee, L. Zettlemoyer, C. Dyer, and N. A. Smith (2018). "Syntactic Scaffolds for Semantic Structures". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 3772–3782.

Tesnière, L. (1959). "Elements of structural syntax, translated by Timothy Osborne and Sylvain Kahane".

Titov, I. and J. Henderson (2007). "A Latent Variable Model for Generative Dependency Parsing". In: *Proceedings of the 10th International Conference on Parsing Technologies. IWPT '07*. Prague, Czech REpublic: Association for Computational Linguistics. 144–155. ISBN: 978-1-932432-90-9.

Tran, K., A. Bisazza, and C. Monz (2018). "The Importance of Being Recurrent for Modeling Hierarchical Structure". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 4731–4736.

Vlachos, A. (2013). "An investigation of imitation learning algorithms for structured prediction". In: *European Workshop on Reinforcement Learning*. 143–154.

Wainwright, M. J. and M. I. Jordan (2008). "Graphical Models, Exponential Families, and Variational Inference". *Found. Trends Mach. Learn.* 1(1-2): 1–305.

Warstadt, A., A. Singh, and S. R. Bowman (2018). "Neural Network Acceptability Judgments". *arXiv preprint arXiv:1805.12471*.

Williams, R. J. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". *Machine learning*. 8(3-4): 229–256.

Wilson, A. C., R. Roelofs, M. Stern, N. Srebro, and B. Recht (2017). "The marginal value of adaptive gradient methods in machine learning". In: *Advances in Neural Information Processing Systems*. 4148–4158.

Xiang, M., B. Dillon, and C. Phillips (2009). "Illusory licensing effects across dependency types: ERP evidence". *Brain and Language*. 108(1): 40–55.

Yin, P., C. Zhou, J. He, and G. Neubig (2018). "StructVAE: Tree-structured Latent Variable Models for Semi-supervised Semantic Parsing". *arXiv preprint arXiv:1806.07832*.

Zhang, Y. and D. Weiss (2016). "Stack-propagation: Improved Representation Learning for Syntax". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics. 1557–1566.