Scalable Syntax-Aware Language Models Using Knowledge Distillation

Adhiguna Kuncoro♠♢ Chris Dyer♠ Laura Rimell♠
Stephen Clark♠ Phil Blunsom♠♢
♠DeepMind, London, UK
♢Department of Computer Science, University of Oxford, UK
{akuncoro,cdyer,laurarimell,clarkstephen,pblunsom}@google.com

Abstract

Prior work has shown that, on small amounts of training data, syntactic neural language models learn structurally sensitive generalisations more successfully than sequential language models. However, their computational complexity renders scaling difficult, and it remains an open question whether structural biases are still necessary when sequential models have access to ever larger amounts of training data. To answer this question, we introduce an efficient knowledge distillation (KD) technique that transfers knowledge from a syntactic language model trained on a small corpus to an LSTM language model, hence enabling the LSTM to develop a more structurally sensitive representation of the larger training data it learns from. On targeted syntactic evaluations, we find that, while sequential LSTMs perform much better than previously reported, our proposed technique substantially improves on this baseline, yielding a new state of the art. Our findings and analysis affirm the importance of structural biases, even in models that learn from large amounts of data.

1 Introduction

Language models (LMs) based on sequential LSTMs (Hochreiter and Schmidhuber, 1997) have numerous practical applications, but it has also been shown that they do not always develop accurate syntactic generalisations (Marvin and Linzen, 2018). Thus, one strategy for improving LSTMs is to change their biases to facilitate more linguistically valid generalisations.

This paper introduces a scalable method for introducing syntactic biases to LSTMs (and indeed, to any left-to-right language model trained with a cross-entropy objective) by distilling knowledge (Bucilă et al., 2006; Hinton et al., 2015) from recurrent neural network grammars (Dyer et al., 2016, RNNGs). RNNGs have been shown to successfully capture non-local syntactic dependencies (Kuncoro et al., 2018), achieve excellent parsing performance (Kuncoro et al., 2017; Fried et al., 2017), and correlate well with encephalography signals (Hale et al., 2018). Unfortunately, these benefits come at the expense of *scalability*, since the hierarchical constituent composition process (§3) within RNNGs means that the structure of the computation graph for a sentence varies according to its tree structure. Even with the help of automatic dynamic batching (Neubig et al., 2017a,b), RNNGs can be ten times slower to train than a comparable LSTM as they benefit less from specialised hardware like GPUs. As such, RNNGs are an impractical alternative to computationally convenient architectures that are used to build language models from massive corpora (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2019; Radford et al., 2019).

As RNNGs are hard to scale, we instead use the predictions of an RNNG *teacher model* trained on a small training set, to guide the learning of syntactic structure in a sequential LSTM *student model*, which is trained on the training set in its entirety. We denote the resulting lanugage model (i.e., the student LSTM) as a *distilled syntax-aware* LSTM LM (**DSA-LSTM**). Intuitively, the RNNG teacher is an expert on syntactic generalisation, although it lacks the opportunity to learn the relevant semantic and common-sense knowledge from a large training corpus. By learning from both, the DSA-LSTM therefore learns from a signal that is informative for syntactic generalisation, but without sacrificing the semantic richness contained in a large corpus.

Since the DSA-LSTM differs from a conventional LSTM only in its training loss, it has the same hardware-friendly computational structure as a conventional LSTM, and is therefore much

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3472–3484
Florence, Italy, July 28 - August 2, 2019. ©2019 Association for Computational Linguistics

faster to train. On targeted syntactic evaluations, it achieves better accuracy than: (i) a strong LSTM LM which, through careful hyperparameter tuning, performs much better than previously thought (§2); (ii) the teacher RNNG that exploits a hierarchical inductive bias but lacks scalability (§3); and (iii) a born-again network (Furlanello et al., 2018) that similarly learns from KD, albeit without a hierarchical bias from the teacher. We analyse the DSA-LSTM's internal representation through the syntactic probe (Shi et al., 2016; Adi et al., 2017) of Blevins et al. (2018), and find that the learned representations encode hierarchical information to a large extent, despite the DSA-LSTM lacking direct access to syntactic annotation.

While not directly comparable, on subject-verb agreement both the teacher RNNG and student DSA-LSTM outperform BERT (Devlin et al., 2019; Goldberg, 2019), which benefits from bidirectional information and is trained on 30 times as much data. Altogether, these findings suggest that structural biases continue to play an important role, even at massive data scales, in improving the linguistic competence of LMs.

2 Replication of Targeted Syntactic Evaluations of LSTM LMs

In this section, we replicate the targeted syntactic evaluations reported by Marvin and Linzen (2018), which assess LMs' ability to assign higher probability in grammatical/ungrammatical minimal pairs within a variety of complex syntactic structures. This will serve as our primary evaluation instrument in this paper.

The following example illustrates the *subject-verb agreement across an object relative clause (no complementiser)* construction:

- The farmer the parents love swims/*swim.

An LM succeeds on each example iff it assigns a higher probability to the grammatical sentence. Marvin and Linzen (2018) report that LSTMs, even with multi-task syntactic supervision, on aggregate still lag far behind human performance.

Experimental settings. Following Marvin and Linzen (2018), we use LSTMs with 650 hidden units trained on the Wikipedia corpus of Gulordava et al. (2018). Hyperparameters are optimised based on a grid search and can be found in the Appendix. As the targeted syntactic evaluations are based on individual sentences, our LSTM models each sentence separately.[1]

Discussion. We present our findings in Table 1 (**"Ours"**); for all our models we report mean and standard deviation of 10 identical models from different random seeds. Our LSTM LM achieves much better perplexity than the LSTM LM (32% ppl. reduction) and even the multi-task LSTM (12% reduction) of Marvin and Linzen (2018). As our LSTM has the same number of hidden units, we attribute this gap to differences in optimisation and codebases. On aggregate, our LSTM LM outperforms the LSTM multi-task model from Marvin and Linzen (2018) that exploits explicit CCG annotations, and is able to match or exceed human performance on 7 out of all 15 constructions, thus confirming earlier findings that neural language models are able to acquire complex syntactic generalisation without explicit syntactic supervision (Gulordava et al., 2018; Goldberg, 2019).

Despite the small variance in perplexity (stdev 0.16 ppl.), the trained LMs exhibit large variance in accuracy for some constructions (up to stdev 0.12 for NPI across a relative clause). This observation is consistent with earlier findings that models with similar perplexity may exhibit different patterns of syntactic generalisation (Kuncoro et al., 2018; Tran et al., 2018), and serves as a caution against reporting results based on single runs.

3 Syntactic Evaluation with RNNG

To what extent can a model that leverages syntactic bias and annotation do well on targeted syntactic evaluations, even when trained on less data?

Here we briefly describe and assess the performance of the stack-only RNNG (Kuncoro et al., 2017) that we use as the teacher. Our choice of RNNG is motivated by its excellent number agreement performance on the Linzen et al. (2016) dataset,[2] achieving 92.9% for four attractors under purely incremental decoding (Kuncoro et al., 2018).

[1] By modelling each sentence separately, our setup is consistent with that of Marvin and Linzen (2018) but differs from those with cross-sentential context (Mikolov et al., 2010).

[2] While BERT (Devlin et al., 2019) achieves even better number agreement performance (Goldberg, 2019), the results are not directly comparable since BERT operates non-incrementally and was trained on 500 times as much data. The current state of the art among models trained on the Linzen et al. (2016) training set is the adaptive universal transformer model (Dehghani et al., 2019).

	Marvin & Linzen models		Ours	Ours (small training)		
	M&L-LSTM	**M&L-Multi**	**Our LSTM**	**Small LSTM†**	**RNNG†**	**Humans**
Gulordava et al. (2018) **test perplexity**	78.65	61.10	**53.73** (±0.16)	94.54 (±0.21)	**92.30** (±0.27)	N/A
SUBJECT-VERB AGREEMENT						
Simple	0.94	**1.00**	**1.00** (±0.00)	0.89 (±0.03)	**0.99** (±0.01)	0.96
In a sentential complement	**0.99**	0.93	0.97 (±0.02)	0.89 (±0.01)	**0.93** (±0.02)	0.93
Short VP coordination	0.90	0.90	**0.96** (±0.02)	0.90 (±0.03)	**0.96** (±0.02)	0.94
Long VP coordination	0.61	0.81	**0.82** (±0.05)	0.78 (±0.03)	**0.94** (±0.03)	0.82
Across a prepositional phrase	0.57	0.69	**0.89** (±0.02)	0.83 (±0.02)	**0.95** (±0.01)	0.85
Across a subject relative clause	0.56	0.74	**0.87** (±0.02)	0.81 (±0.04)	**0.95** (±0.03)	0.88
Across an object relative clause	0.50	0.57	**0.77** (±0.11)	0.54 (±0.08)	**0.95** (±0.03)	0.85
Across an object relative clause (no *that*)	0.52	0.52	**0.70** (±0.05)	0.55 (±0.07)	**0.93** (±0.02)	0.82
In an object relative clause	0.84	0.89	**0.90** (±0.03)	0.79 (±0.05)	**0.96** (±0.01)	0.78
In an object relative clause (no *that*)	0.71	0.81	**0.86** (±0.05)	0.72 (±0.03)	**0.96** (±0.02)	0.79
Average of subject-verb agreement	0.71	0.79	**0.87** (±0.02)	0.77 (±0.02)	**0.95** (±0.01)	0.86
REFLEXIVE ANAPHORA						
Simple	0.83	0.86	**0.91** (±0.01)	**0.93** (±0.01)	0.83 (±0.02)	0.96
In a sentential complement	**0.86**	0.83	0.81 (±0.02)	**0.77** (±0.03)	0.46 (±0.05)	0.91
Across a relative clause	0.55	0.56	**0.64** (±0.02)	0.63 (±0.02)	**0.82** (±0.02)	0.87
Average of reflexive anaphora	0.75	0.75	**0.79** (±0.01)	**0.78** (±0.01)	0.70 (±0.02)	0.91
NEGATIVE POLARITY ITEMS						
Simple	0.40	0.48	**0.96** (±0.04)	**0.93** (±0.06)	0.28 (±0.05)	0.98
Across a relative clause	0.41	0.73	**0.75** (±0.12)	**0.82** (±0.09)	0.78 (±0.06)	0.81
Average of negative polarity items	0.41	0.61	**0.86** (±0.06)	**0.88** (±0.05)	0.53 (±0.04)	0.90
Average of all constructions	0.68	0.75	**0.85** (±0.02)	0.79 (±0.02)	**0.85** (±0.02)	0.88

Table 1: Replication of Marvin and Linzen (2018) results. M&L-Multi is the Marvin and Linzen (2018) LSTM trained on LM and CCG supertagging (Bangalore and Joshi, 1999; Clark and Curran, 2007) losses with an interpolation factor of 0.5. We report our LSTM LM, small LSTM†, and RNNG† performance (†smaller training data; §3) in the format of *mean (±standard deviation)* of 10 identical models from different seeds. Results in bold denote the best among models trained on similar amounts of training data.

3.1 Recurrent Neural Network Grammars

An RNNG defines the joint probability of surface string $\boldsymbol{x}$ and phrase-structure tree $\boldsymbol{y}$, denoted as $t(\boldsymbol{x}, \boldsymbol{y})$. The model generates phrase-structure trees in a top-down, left-to-right manner through a series of action sequences in a process reminiscent of shift-reduce parsing. At any given state, the decision over which action to take is parameterised by a stack LSTM (Dyer et al., 2015) encoding partially-completed constituents. Let $\mathbf{h}_t$ be the stack LSTM hidden state at time t. The next action $a_t \in \{\text{GEN}, \text{NT}, \text{REDUCE}\}$ is sampled according to a categorical distribution defined by an affine transformation and a softmax:

$$a_t \sim \text{softmax}(\mathbf{W_a}\mathbf{h}_t + \mathbf{b_a}).$$

- If $a_t \in \{\text{GEN}, \text{NT}\}$, the model samples a terminal x or a non-terminal n from each respective categorical distribution as the next input:

  $$x \sim \text{softmax}(\mathbf{W_x}\mathbf{h}_t + \mathbf{b_x}),$$
  $$n \sim \text{softmax}(\mathbf{W_n}\mathbf{h}_t + \mathbf{b_n}).$$

- If $a_t = \text{REDUCE}$, the topmost stack elements going back to the last incomplete non-terminal are popped, and a *composition function* (here a bidirectional LSTM) is executed to represent the completed phrase on the stack. This recursive composition function constitutes a primary difference with the syntactic LM of Choe and Charniak (2016) that operates sequentially, and has been found to be crucial for achieving good number agreement (Kuncoro et al., 2018) and correlation with brain signals (Hale et al., 2018).

The stack LSTM, composition function, lookup embeddings, and pairs of affine transformation weights and biases $\{\mathbf{W}, \mathbf{b}\}$ are model parameters.

3.2 Experiments

Here we outline the experimental settings and present our RNNG findings.

Experimental settings. We implement the RNNG with DyNet and enable autobatching on GPU. Predicted phrase-structure trees for the training and validation sets of the Gulordava et al. (2018) Wikipedia dataset are obtained with a pre-trained Berkeley parser (Petrov and Klein, 2007). Since training the RNNG on the full training set with the same number of hidden units as the LSTM would take more than a month,[3] we train the RNNG on $\sim 20\%$ of the training set (600,000 sentences), and use a smaller hidden state size of 256 (vs. 650 for the full LSTM). As the dataset is pre-processed, we select this subset such that all word types occur at least once in this smaller training set.

Incremental decoding and marginal probability. To preserve incrementality constraints, at test time we use a word-synchronised beam search (Fried et al., 2017) with fast-tracking (Stern et al., 2017), using word and action beam sizes of $k = 50$ and $k \times 10 = 500$, respectively. As exact inference of $t(\boldsymbol{x})$ is intractable, we evaluate with a lower bound of the marginal probability by summing over the top k hypotheses $\boldsymbol{y}_1^{b(\boldsymbol{x})}, \ldots, \boldsymbol{y}_k^{b(\boldsymbol{x})}$ on the beam $b(\boldsymbol{x})$ once parsing finishes:

$$t(\boldsymbol{x}) = \sum_{\boldsymbol{y}' \in \mathcal{T}(\boldsymbol{x})} t(\boldsymbol{x}, \boldsymbol{y}') \geq \sum_{i=1}^{k} t(\boldsymbol{x}, \boldsymbol{y}_i^{b(\boldsymbol{x})}),$$

where $\mathcal{T}(\boldsymbol{x})$ denotes the set of all possible phrase-structure trees for a sentence $\boldsymbol{x}$. On targeted syntactic evaluations, the model succeeds iff $\log t(\boldsymbol{x}_{\text{correct}}) > \log t(\boldsymbol{x}_{\text{incorrect}})$.

Discussion. We present the results in Table 1 (sixth column: **"RNNG"**), and compare with LSTMs trained on: (i) the full dataset (fourth column: **"Our LSTM"**), and (ii) the same (smaller) training set as the RNNG (fifth column: **"Small LSTM"**). Our findings clearly reaffirm the benefits of *both* hierarchical bias and data scale. In terms of hierarchical bias, an RNNG that leverages syntactic annotations and explicit composition operators outperforms a comparable small LSTM on 11 out of 15 constructions, and on aggregate improves accuracy on targeted syntactic evaluations from 79% to 85% (29% error reduction), thus matching the aggregate performance of the full LSTM trained on 5 times as much data, although we remark that their success and failure modes appear to be different.

In terms of data scale, the LSTM LM trained on the full training set substantially outperforms the LSTM trained on the smaller training set. In particular, the performance difference between the small and full LSTMs sheds light on which constructions are sensitive to variations in the amount of data. For instance, agreement across an object relative clause exhibits large variations across the two training regimes (77% to 54%), suggesting that LSTMs require a large amount of data to learn these constructions well. Our finding on the importance of data scale for LM training is consistent with the success of recent LM pre-training approaches (Peters et al., 2018; Devlin et al., 2019, *inter alia*) and earlier work on noisy channel models for tasks such as machine translation and speech recognition (Jelinek, 1997; Rosenfeld, 2000; Koehn, 2010, *inter alia*).

Despite its smaller training set, the RNNG performs extremely well on subject-verb agreement, substantially outperforming both the full LSTM and a pre-trained BERT (Devlin et al., 2019, Table 2) trained on 150 times as much data, although it still lags behind the full LSTM on reflexive anaphora and NPI.

4 Syntax-Aware Language Model

Given the trade-off between hierarchical operations and scalability, how can we design LMs that can better capture complex syntactic dependencies ***and*** *be easily scalable at the same time?*

4.1 Knowledge Distillation (KD)

The goal of KD is to find a set of student model parameters $\hat{\theta}_{\text{KD}}$ that would minimise the Kullback–Leibler (KL) divergence between the teacher RNNG's marginal probability $t(\boldsymbol{x}) = \sum_{\boldsymbol{y}' \in \mathcal{T}(\boldsymbol{x})} t(\boldsymbol{x}, \boldsymbol{y}')$ and the LSTM student $q_\theta(\boldsymbol{x})$. Expanding the KL term and removing terms that do not depend on θ yields:

$$\hat{\theta}_{\text{KD}} = \arg\min_{\theta} D_{\text{KL}}\left(t(\boldsymbol{x}) \mid\mid q_\theta(\boldsymbol{x})\right), \quad (1)$$

$$= \arg\min_{\theta} - \sum_{\boldsymbol{x} \in \Sigma^*} t(\boldsymbol{x}) \log q_\theta(\boldsymbol{x}), \quad (2)$$

$$= \arg\min_{\theta} - \mathbb{E}_{\boldsymbol{x} \sim t(\boldsymbol{x})} \log q_\theta(\boldsymbol{x}), \quad (3)$$

where Σ denotes the set of all word types in the vocabulary, and Σ^* the set of all possible sentences. As Eq. 2 involves an intractable summation over the set of all possible sentences, one alternative is to approximate this expectation with Monte Carlo sampling to obtain K sentences $D' = \{\boldsymbol{x}'^{(1)}, \ldots, \boldsymbol{x}'^{(K)}\}$ from $t(\boldsymbol{x})$,[4] and train a student LSTM LM on these sampled sentences as opposed to ground-truth LM data:

[3] We tested the speed of RNNGs and LSTMs with similar capacity (40 million parameters) on DyNet. Both models ran on a single Quadro P4000 GPU with automatic batching turned on and a batch size of 20 sentences.

[4] While an RNNG estimates $t(\boldsymbol{x}, \boldsymbol{y})$, a simple way of sampling surface strings $\boldsymbol{x}$ from the RNNG is to sample pairs of $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)}) \sim t(\boldsymbol{x}, \boldsymbol{y})$ and ignore all non-terminals $\boldsymbol{y}^{(k)}$.

$$\mathbb{E}_{\boldsymbol{x}\sim t(\boldsymbol{x})} \log q_\theta(\boldsymbol{x}) \approx \frac{1}{K}\sum_{\boldsymbol{x}'\in D'}\sum_{j=1}^{|\boldsymbol{x}'|}\log q_\theta(x'_j|\boldsymbol{x}'_{<j}),$$

although our preliminary experiments suggest that this procedure performs poorly due to high variance.[5] We instead approximate Eq. 3 by minimising the KL at the *local word-level*:

$$\mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{t(x)}} \log q_\theta(\boldsymbol{x}) \approx$$
$$\mathbb{E}_{\boldsymbol{x}^*\sim p^*(\boldsymbol{x})} \sum_{j=1}^{|\boldsymbol{x}^*|} D_{\mathrm{KL}}\left(t(w \mid \boldsymbol{x}^*_{<j}) \mid\mid q_\theta(w \mid \boldsymbol{x}^*_{<j})\right),$$

where $\boldsymbol{x}^*$ is sampled from the empirical distribution $p^*(\boldsymbol{x})$, rather than from the teacher RNNG. Here $t(w \mid \boldsymbol{x}^*_{<j})$ and $q_\theta(w \mid \boldsymbol{x}^*_{<j})$ respectively parameterise the (marginal) probability of generating the next-word continuation $w \in \Sigma$, given the "ground-truth" conditioning context $\boldsymbol{x}^*_{<j}$, under the teacher and student models.

For a dataset of sentences $D = \{\boldsymbol{x}^{*(1)}, \ldots, \boldsymbol{x}^{*(|D|)}\}$ characterising the empirical distribution $p^*(\boldsymbol{x}^*) = \frac{1}{|D|}$ when $\boldsymbol{x}^* \in D$ (i.i.d. assumption), the word-level objective is:

$$\hat{\theta}_{\mathrm{KD}} \approx \arg\min_\theta -\frac{1}{|D|}\sum_{\boldsymbol{x}^*\in D} \ell_{\mathrm{KD}}(\boldsymbol{x}^*;\theta),$$
$$\ell_{\mathrm{KD}}(\boldsymbol{x}^*;\theta) = \sum_{j=1}^{|\boldsymbol{x}^*|}\sum_{w\in\Sigma} t(w \mid \boldsymbol{x}^*_{<j}) \log q_\theta(w \mid \boldsymbol{x}^*_{<j}).$$

In earlier work, this local word-level approximation to the KD objective for sequence models has been shown to work surprisingly well in the case of neural machine translation[6] (Kim and Rush, 2016) and language modelling (Furlanello et al., 2018, Born-Again Networks).

Interpolation. As the teacher RNNG is trained on a smaller training set, the DSA-LSTM should not only aim to emulate the RNNG's predictions and risk being upper-bounded by the teacher's performance, but also learn from the correct next word x_j^* to fully exploit scalability.[7] We thus interpolate the distillation (left) and LM (right) losses:

$$\hat{\theta}_{\alpha\text{-int}} = \arg\min_\theta -\frac{1}{|D|}\sum_{\boldsymbol{x}^*\in D}$$
$$\left[\alpha\ell_{\mathrm{KD}}(\boldsymbol{x}^*;\theta) + (1-\alpha)\sum_{j=1}^{|\boldsymbol{x}^*|}\log q_\theta(x_j^* \mid \boldsymbol{x}^*_{<j})\right],$$

where α is the interpolation coefficient. We illustrate the effect of this interpolation in Fig. 1.

Furthermore, computing $\ell_{\mathrm{KD}}(\boldsymbol{x}^*;\theta)$ requires the RNNG's estimate of $t(w \mid \boldsymbol{x}^*_{<j})$, which necessitates an expensive marginalisation over all tree prefixes that generate w conditional on $\boldsymbol{x}^*_{<j}$. For efficiency, we approximate this using the one-best predicted tree from a pre-trained Berkeley parser,[8] denoted as $\hat{\boldsymbol{y}}^{\mathrm{berk}}(\boldsymbol{x}^*)$, as follows:

$$t(w \mid \boldsymbol{x}^*_{<j}) \approx t(w \mid \boldsymbol{x}^*_{<j},\ \hat{\boldsymbol{y}}^{\mathrm{berk}}_{<j}(\boldsymbol{x}^*)),$$

where $\hat{\boldsymbol{y}}^{\mathrm{berk}}_{<j}(\boldsymbol{x}^*)$ are all the non-terminals in $\hat{\boldsymbol{y}}^{\mathrm{berk}}(\boldsymbol{x}^*)$ that occur before x_j^*. In other words, we first parse the sentence with a Berkeley parser, and use the resulting tree prefix as conditioning context to compute the probability of generating $w \in \Sigma$ under the RNNG. While this means that the teacher's predictions are not derived from a purely incremental process,[9] the student DSA-LSTM still operates strictly incrementally. This interpolated objective is similar to label smoothing (Szegedy et al., 2016; Pereyra et al., 2017), with the softmax distribution of the RNNG as the smoothing factor as opposed to the uniform distribution.

Intuition. In Fig. 1, we provide an intuition about why the interpolation of the distillation and LM losses could inject hierarchical bias into a sequential model. We consider the interpolated target with $\alpha = 0.5$ for a prefix (suppressing non-terminals) *Parts of the river valley*, where the correct continuation is *have* since the agreement controller *parts* is plural. The standard LM loss is zero only when all word types other than the correct one are assigned zero probability mass, and it is only in expectation (across training contexts) that syntactic regularities are inferred. In contrast, the interpolated target assigns a minimum probability of 0.5 to the correct label, but crucially contains additional information about the *plausibility* of every alternative based on the teacher RNNG's predictions. Under this objective, the plural verbs

[5] This procedure of training a student LSTM LM on string samples from the RNNG with $K \approx 3,000,000$ yields a high validation perplexity of above 1,000, due to the enormity of the sample space and the use of discrete samples.

[6] While Kim and Rush (2016) proposed a technique for sequence-level KD for machine translation through beam search, the same technique is not directly applicable to LM, which is an unconditional language generation problem.

[7] Recall that $\ell_{\mathrm{KD}}(\boldsymbol{x};\theta)$ does not depend on the true next word x_j^*.

[8] We use the same pre-trained Berkeley parser to obtain training and validation trees in §3.

[9] The resulting syntactic prefix $\hat{\boldsymbol{y}}^{\mathrm{berk}}_{<j}(\mathbf{x})$ for approximating $t(w \mid \boldsymbol{x}^*_{<j})$ under the RNNG is obtained from a Berkeley parser that has access to yet unseen words $\boldsymbol{x}_{>j}$.

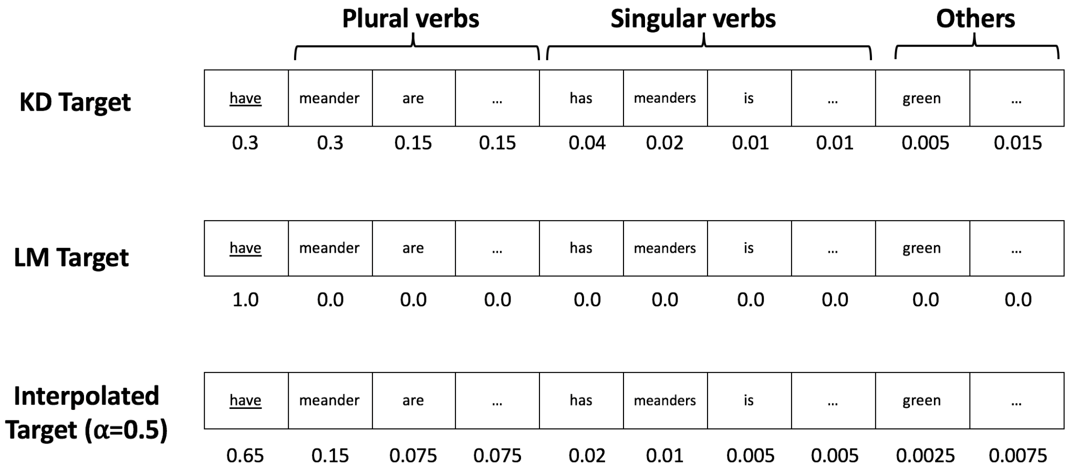

Figure 1: Example of the KD target (top), the standard LM target (middle), and the interpolated target used to train the DSA-LSTM (bottom) with $\alpha = 0.5$, for a prefix (showing only the terminals) *Parts of the river valley*, where the correct continuation is *have* due to the plural subject *parts*.

are and *meander* are assigned relatively high probability mass since they fit both the syntactic and semantic constraints (e.g. Parts of the river valley often meander), while the set of singular verbs *has*, *meanders*, and *is* are assigned much lower probability mass since they are syntactically illicit. Thus, as long as the RNNG makes the accurate structural generalisations (and we have shown that it largely does in §3), every training instance provides the student LSTM with a wealth of information about all the possible legitimate continuations according to the predictions of the hierarchical teacher, thereby making it easier for the student to learn the appropriate hierarchical constraints and generalisations.

Differences with other KD work. Our approach departs from the predominant view of distillation primarily as a means of *compressing* knowledge from a bigger teacher or an ensemble to a compact student (Ba and Caruana, 2014; Kim and Rush, 2016; Liu et al., 2018, *inter alia*) in two important ways. First, here the teacher and student models are different in character, and not just in size: we transfer knowledge from a teacher that models the joint probability of strings and phrase-structure trees through hierarchical operations, to a student that only models surface strings through sequential operations. This setup presents an interesting dynamic since the DSA-LSTM has to mimic the predictions of the RNNG, which conditions on syntactic annotation to guide hierarchical operations, even though the DSA-LSTM itself has no direct access to any syntactic annotations at all.

Second, distillation thus far has mostly been applied in settings where the teacher and student models are trained on the same data. For scalability reasons, we train the RNNG on a subset of the data, and obtain its soft predictions on the rest. We hypothesise that the predictions of the hierarchical teacher—although they come from a model trained on a smaller dataset—can nevertheless encourage the LSTM to develop structurally sensitive representations of the larger dataset it observes.

Born-Again Networks (BA). In practice, the interpolated distillation objective above can be applied to any teacher and student models. Recently, Furlanello et al. (2018) surprisingly finds perplexity improvement in a *born-again* setup that trains an LSTM LM on the gold data, and then uses the resulting model as a teacher to a student LSTM that shares the same architecture as the teacher. To better understand the importance of learning from a hierarchical teacher (which is not the case in a BA-LSTM since the teacher model is also sequential), we present experiments comparing the DSA-LSTM with a BA-LSTM.

4.2 Experiments

Here we describe our experimental settings and present our findings.

Computational challenge. The KD loss necessitates computing the teacher RNNG's predictive softmax distribution for each token in the training set, but pre-computing these for the Gulordava et al. (2018) training set leads to a prohibitive memory footprint.[10] To save space, we instead pre-compute the teacher RNNG's hidden state $\mathbf{h}_t \in \mathcal{R}^M$ for every token x_t in the training set ($M \ll |\Sigma|$), and compute the teacher's softmax on-the-fly with an affine transformation and a softmax, which presents minimal computational overhead.

Experimental settings. The DSA-LSTM has an identical architecture to the LSTM LM (§2), although the learning rate is optimised independently (Appendix). We select the final model based on validation LM perplexity, with targeted syntactic evaluations only applied at test time.

Training speed. Since the DSA-LSTM operates sequentially, it is amenable to batching operations and is five times faster to train than a comparable RNNG. Despite this significant speed-up, training the DSA-LSTM in our basic implementation is still half as fast as the standard LM objective. We attribute this difference to the additional computational overhead associated with the distillation objective, such as I/O operations and computing the cross-entropy between the teacher and student models for the entire vocabulary. These operations, however, only apply at training time; at test time there is no overhead of inferring $q_{\hat{\theta}_{\alpha\text{-int}}}(\boldsymbol{x})$ under the DSA-LSTM.

Baselines. The DSA-LSTM benefits from three main components: (i) a KD objective, which in itself has been shown to be a good regulariser (Furlanello et al., 2018), (ii) the scalability of the sequential architecture, and (iii) a hierarchical bias, which here comes from the teacher RNNG. To understand the benefit of each component, we compare DSA-LSTM with these baselines:

- a strong LSTM LM (§2) that is scalable but lacks a hierarchical bias (**"Full LSTM"**);
- the teacher RNNG trained on a 20% subset of the training set (§3), which benefits from a hierarchical bias but lacks scalability (**"RNNG"**);
- a DSA-LSTM trained on the same smaller subset as the teacher RNNG (**"S-DSA-LSTM"**). This baseline isolates the importance of scalability, since it still benefits from a KD objective and a hierarchical bias from the teacher RNNG;
- a born-again LSTM that benefits from KD and scalability, though it lacks a hierarchical bias due to the sequential teacher (**"BA-LSTM"**).

Discussion. To avoid clutter, for each model variant we present only the mean performance of 10 identical models from different random seeds; results with standard deviations are in the Appendix. We present our findings in Table 2, based on which we derive several observations.

- Of the three models trained on the small subset, the S-DSA-LSTM outperforms the small LSTM trained on standard LM objective, improving overall acccuracy from 0.79 to 0.82 (14% error reduction), even though both models share the same architecture and training set size (i.e. only the training objective is different). On subject-verb agreement, the S-DSA-LSTM successfully narrows the gap with the slower teacher RNNG, which benefits from syntactic bias and annotation. These findings confirm our hypothesis that the KD approach constitutes an efficient way to inject hierarchical bias into sequential models.
- The born-again model (BA-LSTM) outperforms the LSTM LM, albeit by a small margin. This finding suggests that KD helps improve the syntactic competence of LSTMs, even when the teacher model lacks explicit hierarchical bias and shares the same architecture as the student.
- In terms of perplexity, both BA-LSTM and DSA-LSTM perform slightly worse than the full LSTM LM trained without KD loss. We attribute this gap to the smoother target distribution when using KD, which effectively penalises high probabilities on the correct next word x_j^* unless the teacher model is extremely confident. This observation is consistent with earlier findings on label smoothing in machine translation (Pereyra et al., 2017; Vaswani et al., 2017), which often results in better BLEU at the expense of slightly worse likelihood.
- Despite identical architectures, on aggregate the DSA-LSTM substantially improves over the full LSTM (85% to 89%), constituting a 27% error rate reduction and a new state of the art. Our findings suggest that the DSA-LSTM combines the best of both hierarchical bias and data scale: on subject-verb agreement, the DSA-LSTM improves over the LSTM baseline and narrows the gap with the teacher RNNG, while at the same time performing well on reflexive anaphora and

[10] Pre-computing the RNNG's predictions necessitates storing $N \times |\Sigma|$ numbers, where N is the number of tokens. For the Gulordava et al. (2018) training set (∼80M tokens), this requires storing 4 trillion floating points, or 25 terabytes.

	Small Training Set			Full Training Set				
	Small LSTM†	S-DSA-LSTM†	RNNG†	Full LSTM	BA-LSTM	DSA-LSTM	BERT	Humans
Gulordava et al. (2018) **test ppl.**	94.54	93.95	**92.30**	**53.73**	54.64	56.74	N/A	N/A
SUBJECT-VERB AGREEMENT								
Simple	0.89	0.96	**0.99**	**1.00**	**1.00**	**1.00**	1.00	0.96
In a sentential complement	0.89	**0.98**	0.93	0.97	**0.98**	**0.98**	0.83	0.93
Short VP coordination	0.90	0.88	**0.96**	0.96	0.95	**0.99**	0.89	0.94
Long VP coordination	0.78	0.74	**0.94**	**0.82**	0.80	0.80	0.98	0.82
Across a prepositional phrase	0.83	0.88	**0.95**	0.89	0.89	**0.91**	0.85	0.85
Across a subject relative clause	0.81	0.87	**0.95**	0.87	0.87	**0.90**	0.84	0.88
Across an object relative clause	0.54	0.69	**0.95**	0.77	0.81	**0.84**	0.89	0.85
Across an object relative clause (no *that*)	0.55	0.61	**0.93**	0.70	0.74	**0.77**	0.86	0.82
In an object relative clause	0.79	0.87	**0.96**	0.90	0.91	**0.92**	0.95	0.78
In an object relative clause (no *that*)	0.72	0.88	**0.96**	0.86	0.83	**0.92**	0.79	0.79
Average of subject-verb agreement	0.77	0.84	**0.95**	0.87	0.88	**0.90**	0.89	0.86
REFLEXIVE ANAPHORA								
Simple	**0.93**	0.90	0.83	0.91	**0.92**	0.91	0.94	0.96
In a sentential complement	0.77	**0.78**	0.46	0.81	0.81	**0.82**	0.89	0.91
Across a relative clause	0.63	0.67	**0.82**	0.64	0.64	**0.67**	0.80	0.87
Average of reflexive anaphora	**0.78**	**0.78**	0.70	0.79	0.79	**0.80**	0.88	0.91
NEGATIVE POLARITY ITEMS								
Simple	0.93	0.84	0.28	0.96	**0.98**	0.94	N/A	0.98
Across a relative clause	0.82	0.73	0.78	0.75	0.70	**0.91**	N/A	0.81
Average of negative polarity items	**0.88**	0.79	0.53	0.86	0.84	**0.92**	N/A	0.90
Average of all constructions	0.79	0.82	**0.85**	0.85	0.86	**0.89**	N/A	0.88

Table 2: Experimental findings of the **"DSA-LSTM"**. For each column, we report the mean of 10 identical models trained from different random seeds; standard deviation values are reported in the Appendix. **"S-DSA-LSTM"** indicates the DSA-LSTM trained on the smaller RNNG training set, while **"BA-LSTM"** is the born-again model where the teacher is the full LSTM LM. We also compare with the syntactic generalisation of **"BERT"** Base (Devlin et al., 2019; Goldberg, 2019), which is not strictly comparable since it is trained on 30 times as much data. † indicates models trained on the smaller 20% training set (§3). Results in bold denote the best among those trained with the same amounts of data.

NPI, on which the teacher RNNG (but not the full LSTM) fails to achieve a good performance.

- While not directly comparable, the DSA-LSTM outperforms a pre-trained BERT (Devlin et al., 2019; Goldberg, 2019)[11] on subject-verb agreement. Since BERT benefits from bidirectionality and was trained on 30 times as much data as the DSA-LSTM, this finding suggests that, at least in terms of syntactic competence, structural biases continue to be relevant even as the current generation of sequential LMs is able to exploit increasingly large amounts of data.

4.3 Probing for Hierarchical Information

Having established the advantages of the DSA-LSTM on targeted syntactic evaluations, we turn to the question of analysing how its internal representation differs from that of a standard LSTM LM. To this end, we adopt the method of Blevins et al. (2018) and use a probe (Shi et al., 2016; Adi et al., 2017; Belinkov et al., 2017; Conneau et al., 2018; Hewitt and Manning, 2019, *inter alia*) that predicts the *grandparent constituent* of a word token x_t, based on its encoding $\mathbf{h}_t$ under the pretrained LSTM. Under this framework, the accuracy of the probe on a held-out set can be understood as an indication of how well the hidden states encode the relevant syntactic information required to succeed in this task.

We use a linear classifier for the probe and obtain the predicted grandparent constituent label using the same pre-trained Berkeley parser (§3) that we used to obtain predicted phrase-structure trees to train the RNNG. For the probing experiment, we randomly select sentences from each respective training, validation, and test set of the Gulordava et al. (2018) dataset to yield ∼300,000 words for training and ∼10,000 words for each of validation and test sets. For the probe features, we use a concatenation of the LSTM hidden state at the current and next words,[12] i.e. $[\mathbf{h}_t; \mathbf{h}_{t+1}]$, where ; denotes the concatenation operation.

Recall that the DSA-LSTM operates only on word sequences and has no access to the Berkeley parse during training. We summarise the probing

[11] Goldberg (2019) applies an additional pre-processing step, removing sentences in which the focus verb does not appear as a single word in the word piece-based vocabulary; hence, the evaluation sentences are slightly different.

[12] Our probing feature set thus slightly differs from that of Blevins et al. (2018), who concatenated the hidden states of a left-to-right and right-to-left LSTM language models.

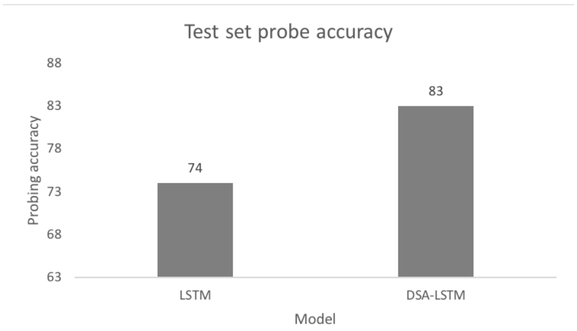

Figure 2: Probing accuracy on the test set. We analyse the hidden states of the LSTM and DSA-LSTM to analyse the structural information encoded in each respective model's hidden state.

result in Fig. 2. Overall, the syntactic probing accuracy for the DSA-LSTM is much higher than for the LSTM LM (83% to 74%; a 34% error rate reduction), suggesting that the means by which the DSA-LSTM achieves better syntactic competence is by tracking more hierarchical information during sequential processing.

5 Related Work

Augmenting language models with syntactic information and structural inductive bias has been a long-standing area of research. To this end, syntactic language models estimate the joint probability of surface strings and some form of syntactic structure (Jurafsky et al., 1995; Chelba and Jelinek, 2000; Roark, 2001; Henderson, 2004; Emami and Jelinek, 2005; Buys and Blunsom, 2015; Mirowski and Vlachos, 2015; Dyer et al., 2016; Kim et al., 2019). In contrast to these approaches, the DSA-LSTM only models the probability of surface strings, albeit with an auxiliary loss that distills the next-word predictive distribution of a syntactic language model.

Earlier work has also explored multi-task learning with syntactic objectives as an auxiliary loss in language modelling and machine translation (Luong et al., 2016; Eriguchi et al., 2016; Nadejde et al., 2017; Enguehard et al., 2017; Aharoni and Goldberg, 2017; Eriguchi et al., 2017). Our approach of injecting syntactic bias through a KD objective is orthogonal to this approach, with the primary difference that here the student DSA-LSTM has no direct access to syntactic annotations; it does, however, have access to the teacher RNNG's softmax distribution over the next word.

Our approach is also closely related to recent work that introduces structurally-motivated inductive biases into language models. Chung et al. (2017) segmented the hidden state update of an RNN through a multi-scale hierarchical recurrence, thereby providing a shortcut to the gradient propagation of long-range, hierarchical dependencies. Yogatama et al. (2018) introduced a stack-structured memory to encourage hierarchical modelling in language models, where the resulting model successfully outperforms standard LSTM variants in number agreement (Linzen et al., 2016) evaluation. Shen et al. (2019) imposed a hierarchical bias on the LSTM cell-updating mechanism, based on the intuition that larger constituents contain information that changes more slowly across the sequence. Our proposed method is orthogonal and can be applied on top of these recent approaches.

6 Conclusion

In this paper, we introduce a distilled syntax-aware LSTM (DSA-LSTM), which combines scalability with structural biases. We achieve this by distilling the predictions about upcoming words in a large training corpus made by a (computationally complex) hierarchical language model trained on a small subset of the data. While we find that LSTM language models achieve better syntactic generalisation than previously thought, on targeted syntactic evaluations our approach improves over this strong baseline, yields a new state of the art, compares favourably to a language model trained on much more data, and results in a language model that encodes hierarchical information to a large extent despite its sequential architecture. Our approach is a general one that can be applied to other student model architectures, such as Transformers (Vaswani et al., 2017). These findings suggest that the question of structural biases continues to be relevant for improving syntactic competence, even in scalable architectures that can benefit from ever-growing amounts of training data.

Acknowledgments

We would like to thank Rebecca Marvin and Tal Linzen for their help in answering questions regarding data preparation. We also thank Dani Yogatama, John Hale, and the three anonymous reviewers for their helpful suggestions.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proc. of ICLR*.

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proc. of ACL*.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *NIPS*.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proc. of ACL*.

Terra Blevins, Omer Levy, and Luke Zettlemoyer and. 2018. Deep rnns encode soft hierarchical syntax. In *Proc. of ACL*.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proc. of KDD*.

Jan Buys and Phil Blunsom. 2015. A Bayesian model for generative transition-based dependency parsing. *CoRR*, abs/1506.04334.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4).

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proc. of EMNLP*.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *Proc. of ICLR*.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *Proc. of ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.

Ahmad Emami and Frederick Jelinek. 2005. A neural syntactic language model. *Machine Learning*, 60:195–227.

Émile Enguehard, Yoav Goldberg, and Tal Linzen. 2017. Exploring the syntactic abilities of rnns with multi-task learning. In *Proc. of CoNLL*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proc. of ACL*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proc. of ACL*.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proc. of ACL*.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born-again neural networks. In *Proc. of ICML*.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proc. of NAACL*.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan R. Brennan. 2018. Finding syntax in human encephalography with beam search. In *Proc. of ACL*.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proc. of ACL*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proc. of NAACL*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proc. of ACL*.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchaman, and N. Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Proc. of ICASSP*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proc. of EMNLP*.

Yoon Kim, Alexander M. Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gabor Melis. 2019. Unsupervised recurrent neural network grammars. In *Proc. of NAACL*.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proc. of EACL*.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proc. of ACL*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*.

Yijia Liu, Wanxiang Che, Huaipeng Zhao, Bing Qin, and Ting Liu. 2018. Distilling knowledge for search-based structured prediction. In *Proc. of ACL*.

Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proc. of ICLR*.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proc. of EMNLP*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of Interspeech*.

Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. In *Proc. of ACL-IJCNLP*.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Predicting target language ccg supertags improves neural machine translation. In *Proc. of WMT*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017a. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980*.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017b. On-the-fly operation batching in dynamic computation graphs. In *Proc. of NIPS*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *Proc. of ICLR*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of NAACL*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2).

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here. In *Proc. of IEEE*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proc. of ICLR*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proc. of EMNLP*.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proc. of EMNLP*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proc. of CVPR*.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proc. of EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS*.

Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. 2018. Memory architectures in recurrent neural network language models. In *Proc. of ICLR*.

Appendix

Here we outline the hyperparameters and the experimental results with standard deviation values.

A Hyperparameters

The hyperparameters for each model is summarised as follows.

LSTM LMs. For the LSTM LMs trained on the full and small training sets, we use the following hyperparameters that achieve the best validation perplexity following a grid search: 2-layer LSTM with 650 hidden units per layer for the full LSTM and 300 hidden units per layer for the small LSTM (similar model capacity as the RNNG trained on the same smaller training set), optimised by stochastic gradient descent (SGD) with a learning rate of 0.45 (decayed exponentially at every epoch with a factor of 0.9 after the tenth epoch), a dropout rate of 0.2 applied on both input and recurrent connections, and a batch size of 20 sentences.

RNNG. For the RNNG, we use the following hyperparameters that achieve the best validation perplexity following a similar grid search: 2-layer stack LSTM with 256 hidden units per layer, optimised by SGD with a learning rate of 0.3 (decayed exponentially at every epoch with a factor of 0.92 after the tenth epoch), a dropout rate of 0.3 applied on both input and recurrent connections, and a batch size of 10 sentences.

DSA-LSTMs and Born-Again LSTMs. We use the same hyperparameters for the DSA-LSTMs trained on both the full and small (S-DSA-LSTM) training sets and the born-again LSTM (BA-LSTM) trained on the full training set. Since the model architectures are identical with the respective LSTM LMs (i.e. only the training objective is different), we only optimise for the learning rates and keep all other hyperparameters the same. We find that a learning rate of 0.4 and an exponential decay factor of 0.9 applied after the tenth epoch works well across all three models trained with the KD objective.

B Experimental Results with Standard Deviation

We summarise the experimental results that include standard deviation values in Table 3.

	Small Training Set		Full Training Set				
	S-DSA-LSTM†	**RNNG**†	**Full LSTM**	**BA-LSTM**	**DSA-LSTM**	**BERT**	**Humans**
Gulordava et al. (2018) **test ppl.**	93.95 (±0.18)	**92.30** (±0.27)	**53.73** (±0.16)	54.64 (±0.25)	56.74 (±0.26)	N/A	N/A
SUBJECT-VERB AGREEMENT							
Simple	0.96 (±0.03)	**0.99** (±0.01)	**1.00** (±0.00)	**1.00** (±0.00)	**1.00** (±0.00)	1.00	0.96
In a sentential complement	**0.98** (±0.02)	0.93 (±0.02)	0.97 (±0.02)	**0.98** (±0.02)	**0.98** (±0.02)	0.83	0.93
Short VP coordination	0.88 (±0.04)	**0.96** (±0.02)	0.96 (±0.02)	0.95 (±0.02)	**0.99** (±0.02)	0.89	0.94
Long VP coordination	0.74 (±0.03)	**0.94** (±0.03)	**0.82** (±0.05)	0.80 (±0.04)	0.80 (±0.02)	0.98	0.82
Across a prepositional phrase	0.88 (±0.02)	**0.95** (±0.01)	0.89 (±0.02)	0.89 (±0.03)	**0.91** (±0.03)	0.85	0.85
Across a subject relative clause	0.87 (±0.02)	**0.95** (±0.03)	0.87 (±0.02)	0.87 (±0.01)	**0.90** (±0.02)	0.84	0.88
Across an object relative clause	0.69 (±0.06)	**0.95** (±0.03)	0.77 (±0.11)	0.81 (±0.05)	**0.84** (±0.03)	0.89	0.85
Across an object relative clause (no *that*)	0.61 (±0.05)	**0.93** (±0.02)	0.70 (±0.05)	0.74 (±0.03)	**0.77** (±0.02)	0.86	0.82
In an object relative clause	0.87 (±0.05)	**0.96** (±0.01)	0.90 (±0.03)	0.91 (±0.03)	**0.92** (±0.04)	0.95	0.78
In an object relative clause (no *that*)	0.88 (±0.03)	**0.96** (±0.02)	0.86 (±0.05)	0.83 (±0.02)	**0.92** (±0.02)	0.79	0.79
Average of subject-verb agreement	0.84 (±0.02)	**0.95** (±0.01)	0.87 (±0.02)	0.88 (±0.01)	**0.90** (±0.01)	0.89	0.86
REFLEXIVE ANAPHORA							
Simple	**0.90** (±0.01)	0.83 (±0.02)	0.91 (±0.01)	**0.92** (±0.03)	0.91 (±0.04)	0.94	0.96
In a sentential complement	**0.78** (±0.01)	0.46 (±0.05)	0.81 (±0.02)	0.81 (±0.02)	**0.82** (±0.03)	0.89	0.91
Across a relative clause	0.67 (±0.03)	**0.82** (±0.02)	0.64 (±0.02)	0.64 (±0.02)	**0.67** (±0.03)	0.80	0.87
Average of reflexive anaphora	**0.78** (±0.01)	0.70 (±0.02)	0.79 (±0.01)	0.79 (±0.02)	**0.80** (±0.03)	0.88	0.91
NEGATIVE POLARITY ITEMS							
Simple	**0.84** (±0.05)	0.28 (±0.05)	0.96 (±0.04)	**0.98** (±0.02)	0.94 (±0.04)	N/A	0.98
Across a relative clause	0.73 (±0.07)	**0.78** (±0.06)	0.75 (±0.12)	0.70 (±0.10)	**0.91** (±0.07)	N/A	0.81
Average of negative polarity items	**0.79** (±0.05)	0.53 (±0.04)	0.86 (±0.06)	0.84 (±0.05)	**0.92** (±0.05)	N/A	0.90
Average of all constructions	0.82 (±0.02)	**0.85** (±0.02)	0.85 (±0.02)	0.86 (±0.01)	**0.89** (±0.01)	N/A	0.88

Table 3: Experimental findings of the **"DSA-LSTM"**. For each column, we report the mean and standard deviation values of 10 identical models trained from different random seeds. **"S-DSA-LSTM"** indicates the DSA-LSTM trained on the smaller RNNG training set, while **"BA-LSTM"** is the born-again model where the teacher is the full LSTM LM. We also compare with the syntactic generalisation of **"BERT"** Base, which is not strictly comparable since it is trained on 30 times as much data. † indicates models trained on the smaller 20% training set (§3). Results in bold denote the best among those trained with the same amounts of data.