LiLT volume 15, issue 2 2017

Using Deep Neural Networks to Learn Syntactic Agreement

JEAN-PHILLIPE BERNARDY, *University of Gothenburg* SHALOM LAPPIN, *University of Gothenburg, King's College London, Queen Mary University of London*

Abstract

We consider the extent to which different deep neural network (DNN) configurations can learn syntactic relations, by taking up Linzen et al.'s (2016) work on subject-verb agreement with LSTM RNNs. We test their methods on a much larger corpus than they used (a ∼24 million example part of the WaCky corpus, instead of their ∼1.35 million example corpus, both drawn from Wikipedia). We experiment with several different DNN architectures (LSTM RNNs, GRUs, and CNNs), and alternative parameter settings for these systems (vocabulary size, training to test ratio, number of layers, memory size, drop out rate, and lexical embedding dimension size). We also try out our own unsupervised DNN language model. Our results are broadly compatible with those that Linzen et al. report. However, we discovered some interesting, and in some cases, surprising features of DNNs and language models in their performance of the agreement learning task. In particular, we found that DNNs require large vocabularies to form substantive lexical embeddings in order to learn structural patterns. This finding has interesting consequences for our understanding of the way in which DNNs represent syntactic information. It suggests that DNNs learn syntactic patterns more efficiently through rich lexical embeddings, with semantic as well as syntactic cues, than from training on lexically impoverished strings that highlight structural patterns.

1 Introduction

The extent to which machine learning systems in general, and neural networks in particular, can learn regularities that depend on syntactic structure is a topic of ongoing debate.[1] Subject-verb number agreement is a paradigmatic case of a syntactic feature that depends on structural properties of a sentence.

Linzen et al. (2016) train a Long Short Term Memory Recurrent Neural Network (LSTM RNN) on a subset of a Wikipedia corpus to predict the number of a verb. As they observe, the task increases in difficulty in relation to the length of the sequence of NPs with the wrong number feature that occur between a subject and the verb that it controls. They refer to such intervening NPs as *attractors*.[2] In (1) the attractors for the *The students-submit* pair are indicated in boldface.

(1a) *The students submit* a final project to complete the course.

(b) *The students* enrolled in **the program** *submit* a final project to complete the course.

(c) *The students* enrolled in **the program** in **the Department** *submit* a final project to complete the course.

(d) *The students* enrolled in **the program** in **the Department** where **my colleague** teaches *submit* a final project to complete the course.

Linzen et. al use a dependency parser to identify the controlling subject of each verb in their corpus of examples. This identification is necessary to compute the number of attractors, but it is not used in the training for the number prediction task, because the number of the verb is morphologically manifest in the raw data.

They train their LSTM RNN on ∼121,500 examples (9% of the total corpus) by supervised learning, in which the system is shown the correct number feature of the verb. They test the number predictions of their RNN on ∼1.21 million examples (90% of their corpus). They encode the input words as vectors in 50 dimensions, and their RNN has 50 hidden units. They report that their system achieves 99% accuracy in the number prediction task for cases with 0 attractors between the subject and its verb, and declines to 83% accuracy for examples with 4 attractors. They do not report scores for examples with more than 4 attractors.

In addition to the supervised number prediction task, they train a generative language model (predicting the next word) and use it to predict the number of the verb in an unsupervised manner. By contrast to their supervised

[1] See Lau et al. (2016) for recent discussion and references.

[2] It would actually be more perspicuous to describe these NPs as *distractors*, given that they are marked for the non-agreeing number. But to avoid confusion, we retain Linzen et al.'s original term.

LSTM RNN, Linzen et al.'s language model goes below chance in its predictions for 4 attractor cases. While the much larger Google LM (Józefowicz et al., 2016) does better, at a $\sim$45% error for 4 attractors, it also performs well below their supervised RNN. They conclude that a considerable amount of syntactic structure is accessible to DNN learning if training is properly supervised.

Our main objective in the work that we report here is to explore the capabilities, and the limitations of DNNs for learning complex syntactic relations which depend on structural properties of sentences.

We used methods similar to Linzen et al.'s to test several DNN models on a much larger corpus. We experimented with different DNN architectures, and with alternative values for the following parameters:

- Ratio of training to testing as a partition of the corpus
- Number of hidden units (memory size)
- Vocabulary size
- Number of layers
- Dropout rate
- Lexical embedding dimension size

In addition we applied our own language model to the number prediction task, testing two distinct methods of predicting verb number from the model's probability distribution.

Our results are broadly compatible with those that Linzen et al. present. However, some of the correlations that we observed between certain parameter settings and levels of performance surprised us.

Specifically, by working with different vocabulary sizes, lexical encodings, and embedding sizes we discovered that our supervised DNN models learn agreement patterns more effectively from rich word embeddings than from abstract syntactically annotated input. We also found that, in general, our models required larger amounts of training relative to testing than Linzen et al. describe for their system, in order to reach the performance that they report.[3] We have observed that increasing the values of hyperparameters (and thereby the number of degrees of freedom in the network itself) generally improves accuracy, even if after a certain point, overfitting is observed. But changes to any individual hyperparameter do not create dramatic effects. However, the total effect of hypeparameter optimisation is quite significant.

Finally, we were able to construct a language model with significantly better (unsupervised) prediction. Yet, our model is much smaller than the

[3] We could not replicate their near perfect accuracy for zero attractors. This could be because we trained and tested our DNNs on much larger corpora with less filtered dependency sequences.

Google LM. All of our supervised DNNs outperformed our language model on the number prediction task.[4]

1 Experimental Design

1.1 Corpus

For our experiments, we used the WaCkypedia English corpus (Baroni et al., 2009), which contains ∼24 million example cases of present tense subject-verb agreement.[5] The corpus is annotated for POS by TreeTagger (Schmid, 1995), and for dependency relations by the MaltParser (Nivre et al., 2007).

Linzen et al. restricted training and testing to one agreement case per sentence in their corpus. We used the full set of number agreement relations in the sentences of our corpus for our experimental work.

They also limit their test, but not their training examples to cases in which all NPs intervening between the subject and the verb that it controls are attractors. We did not adopt this constraint on our test sets of examples. We included the cases in which agreeing, as well as non-agreeing NPs intervened between the subject and its verb. We are interested in measuring the accuracy with which a DNN predicts verb number in complex and possibly confusing syntactic sequences. Training and testing less filtered data of the kind that involves discarding the non-agreement constraint that Linzen et al. apply to their test set provides a realistic indication of the success with which different types of DNN learn number agreement patterns.

Therefore, our test procedures depart slightly from those of Linzen et al. in relaxing both the single agreement case and the (non-) agreement conditions on the test set.

1.2 Models

Models Trained on the Inflection Task

We experiment with three types of DNN. We use a version of Linzen et al.'s LSTM RNN, a Convolutionnal Neural Network (CNN) similar to that presented in Kalchbrenner et al. (2016), and a Gated Recurrent Unit (GRU) network of the kind described in Cho et al. (2014).[6]

[4] However, by limiting the vocabulary of our LM to the 100 most frequent words we have, in effect, introduced an element of supervision in learning. This feature of the model forces it to attend to POS sequences and abstract away from lexical semantic properties (as well as real world knowledge) that would be introduced with a larger vocabulary.

[5] We are grateful to the administrators of the WaCKy corpora (`http://wacky.sslmit.unibo.it`) for giving us access to to this corpus .

[6] We are grateful to Tal Linzen for sharing the code that Linzen et al. used in their work. We note that our experiments were performed from scratch: both the data and our code are new. The code for our models and our data sets are available at `https://github.com/GU-CLASP/DNNSyntax`.

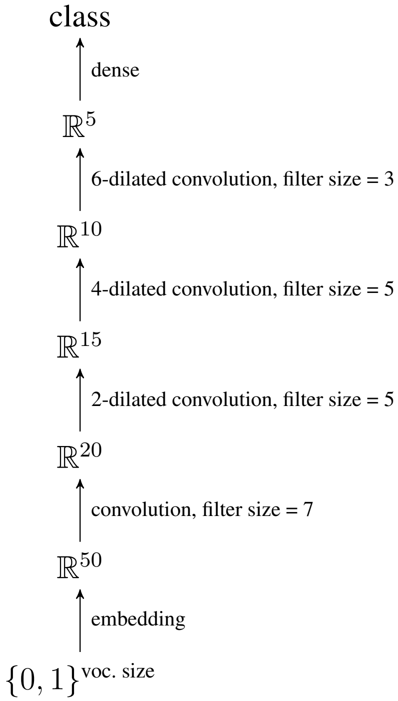

class
dense
$\mathbb{R}^5$
6-dilated convolution, filter size = 3
$\mathbb{R}^{10}$
4-dilated convolution, filter size = 5
$\mathbb{R}^{15}$
2-dilated convolution, filter size = 5
$\mathbb{R}^{20}$
convolution, filter size = 7
$\mathbb{R}^{50}$
embedding
$\{0,1\}^{\text{voc. size}}$

FIGURE 1: Our CNN Architecture

We implemented our own versions of an LSTM RNN, GRU RNN, and CNN, using the Keras library (Chollet, 2015) with a TensorFlow backend. Our LSTM RNN and GRU RNN have standard architectures. We apply a dense layer with sigmoid activation to the output of the latest RNN cell to obtain the verb-number classifier.

Our CNN has 6 levels with filtering successively compressing vector dimensions from 15 through 20, 15, 10 to 5. Convolution at these levels yields 7, 5, 5, and 3 features, respectively. Every convolution layer uses a ReLU activation function. The last layer is a dense layer with sigmoid activation. We represent this CNN graphically in fig.1.

A Generative Language Model

In addition to models which predict verb numbers, we have trained a generative language model. For this purpose we use an LSTM RNN with two layers, 1200 units per layer, and a dropout rate of 0.5 to generate a language model from the WaCky corpus of sentences.[7] We did this by employing the 100 most common words and substituting corresponding POS tags for the others in the sentences. This design is intended both to improve the performance of the language model by restricting the number of possible outputs. It is targetted to focus on syntactic patterns that we seek to recognise by removing most of the semantic (lexical) features in the corpus. We have performed experiments with 100, 10k and 100k common words on the supervised number prediction task, to control for the role of the lexicon in supervised learning.

1.3 Methods

For our supervised learning experiments we followed Linzen et al. in training and testing procedures. We trained each DNN on a portion of our agreement example corpus using customary back propagation and gradient descent, with the Adam (Kingma and Ba, 2014) optimizer. The training set was subdivided into a proper training and validation set. After each run through the proper training set, the average loss was computed for the validation set. We ended training when the validation loss function reached a local minimum value.

Contrary to Linzen et al. we applied a relatively large batch size (1024 instead of 16). We tested the resulting model on the remainder of the corpus by having it predict the number feature of the verb in each example of the test set. As stated above, the dependency parse structures were not used in the training. They were applied only to determine the number of attractors between the subject and the verb that it controls, as well as all other intervening nouns in this sequence. The part-of-speech tagging was employed both to determine the correct number of the verb and to limit the vocabulary in a syntactically meaningful way, as explained below.

We first identified a benchmark of reasonable performance for our DNN configuration and training. We settled on an LSTM with one layer of 150 units and no dropout, a data-set constructed with 10000 words, lexical embeddings of dimension 50, and a training regimen of 90% of the corpus. We then conducted experiments varying each of these parameters independently, holding the others constant. We ran each DNN with training on 10%, 50%, and 90% of the corpus, testing on the remainder for each split. We tested 50, 150, 450 and 1350 units for the LSTM layers. We experimented with embedding vocabulary sizes of 100, 10k and 100k words, substituting corresponding POS tags for the rest. We used 1, 2 and 4 layers for our LSTM RNN. We tested dropout rates of 0, 0.1, 0.2 and 0.5. These dropouts applied to the weights within the LSTM layers, but not at the final dense layer. Finally we tested lexical embedding dimension sizes of 17, 50, 150, and 450.

Our working hypothesis for the reduced vocabulary experiment was that a

[7] We also experimented with a dropout rate of 0.1 with our LM, but the results were significantly worse than those that we achieved with 0.5.

DNN would learn the target syntactic dependency pattern more efficiently if it was exposed to input consisting largely of POS sequences in which number features are marked on noun and verb tags. We thought that such input would highlight the dependency relations more clearly by abstracting away from possibly confounding distributional lexical information contained in richer embeddings. Specifically, we conjectured that if impoverished lexical sequences are used to exhibit a structural relation, with the relevant number feature spotlighted, it would be easier for a DNN to discern the relevant agreement pattern. Adding rich lexical content that includes semantic cues might conceal this pattern by adding irrelevant distributional information.

Predictions Using a Generative Language Model

Our language model provides us with an unsupervised system for predicting agreement. We tested two ways for doing this. Let $p(w_i|w_{i-1},...,w_{i-k})$ be the predicted probability of a word w in a string, given the prefix of $w_{i-1},...,w_{i-k}$ of preceding words in that string.

For the first way of predicting verb agreement we determine for each example in our test set if (1.1) holds, when V^n ranges over verbs with the correct number feature (n) in a particular string (i.e. the number agreeing with that of the verb's controlling subject), and $V^{\neg n}$ ranges over verbs with the wrong number feature in that string.

$$\sum_{V^n} p(V_i^n|w_{i-1},...,w_{i-k}) \quad > \quad \sum_{V^{\neg n}} p(V_i^{\neg n}|w_{i-1},...,w_{i-k}) \qquad (1.1)$$

The second way of predicting verb agreement is to test if (1.2) applies. In this case we check wether the model gives a higher probability for occurrence of the inflected $Verb_i$ which is found in the test set versus the same verb with the opposite inflection.

$$p(Verb_i^n|w_{i-1},...,w_{i-k}) \qquad > \qquad p(Verb_i^{\neg n}|w_{i-1},...,w_{i-k}) \qquad (1.2)$$

The prefix $w_{i-1},...,w_{i-k}$ is identical on both sides of the inequality in each case.

The first method measures the summed conditional probabilities of all correctly number inflected verbs occurring after the prefix in a string against the summed predicted probabilities of all incorrectly number inflected verbs appearing in this position. We will refer to this as *the summing method* of predicting verb number from a LM.

The second procedure compares only the predicted probability of the correctly number-marked form of the actual verb in this position with that of its incorrectly marked form. We will describe it as *the verb targeted method* of predicting verb number from a LM.

In the summing method the LM is not given any semantic cue, whereas in the verb targeted method method it is given the cue of which verb is expected. Yet in our 100-word vocabulary there are only 4 inflectible verb forms: “to be”, “to have”, “to state”, and other verbs lumped together in the “VV” part of speech code, so this semantic information is relatively limited.

1 Results

We present our results in figures 2 to 6. In all the figures, except for fig. 6, the y-axis gives the accuracy rate, and the x-axis the number of NP attractors. We found that 73% of the verbs in our test sets are singular. This provides a majority choice baseline, which is indicated by a straight horizontal line in these figures.

Figures 2a and 2b shows that using a reduced vocabulary of 100 most common words, substituting POS tags for the remainder, consistently reduces accuracy across DNN architectures, for the supervised learning task. Increasing the vocabulary to 100k words generally yields a significant improvement in performance for the LSTM RNN, but gives mixed results for our CNN.

Fig. 3a indicates that increasing the ratio of training to testing examples from 10% to 50% significantly improves the performance of the LSTM RNN (with 150 units and a vocabulary of 10,000 word embeddings). Further increasing it to 90% does not make much of a difference, even degrading accuracy slightly at 6 attractors.

Fig. 3b reveals that the LSTM and GRU RNNs perform at a comparable level, and both achieve significantly better accuracy than our CNN.

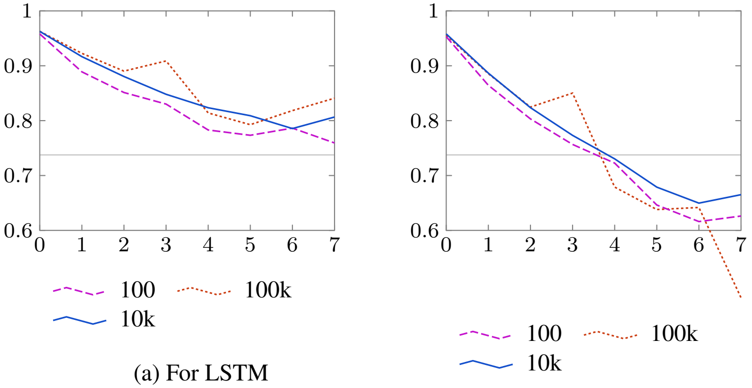
Fig. 3c suggests that increasing the number of units in an LSTM RNN from 50 to 150 improves its performance on the task, while further expanding this number to 450 yields greater accuracy in relation to the number to attractors. Each three-fold increase in the number of units achieves a similar improvement in percentage points for a higher number of attractors. A further increase to 1350 provides only a small overall improvement.

Fig. 3d indicates that increasing the number of layers for an LSTM RNN from 1 150-unit layer to 2 such layers marginally improves its performance. A further increase to 4 150-unit layers makes no clear difference.

Fig. 3e shows that by introducing a dropout rate of 0.1 for the LSTM RNN (1 layer with 150 units) we improve its performance slightly. An increase to 0.2 provides no clear benefit. Furthuer increasing the dropout rate to 0.5 degrades performance.

Fig. 3f indicates that decreasing lexical embedding dimensions from our benchmark 50 to 17 decreases the performance of our LSTM RNN. Increasing dimension size to 150 improves performance, while further expanding it to 450 contributes no benefit, and, for some cases, reduces accuracy.

Fig. 5 shows that while predicting verb number with our language model



(a) For LSTM

(b) For CNN architecture

FIGURE 2: Comparing effect of vocabulary size across architectures.

using the summing method (inequality (1.1)) yields results comparable to Linzen et al.'s LM, the verb targeted method (inequality (1.2)) achieves a far higher level of accuracy than their model, and than the results that they give for the much larger Google LM (Józefowicz et al., 2016). We note that Linzen et al. report using the verb targeted method for their results. However the accuracy of our LM with the verb targeted method is still below our best supervised results for the LSTM RNN. Its performance with the verb targeted method is roughly comparable to that of our CNN.

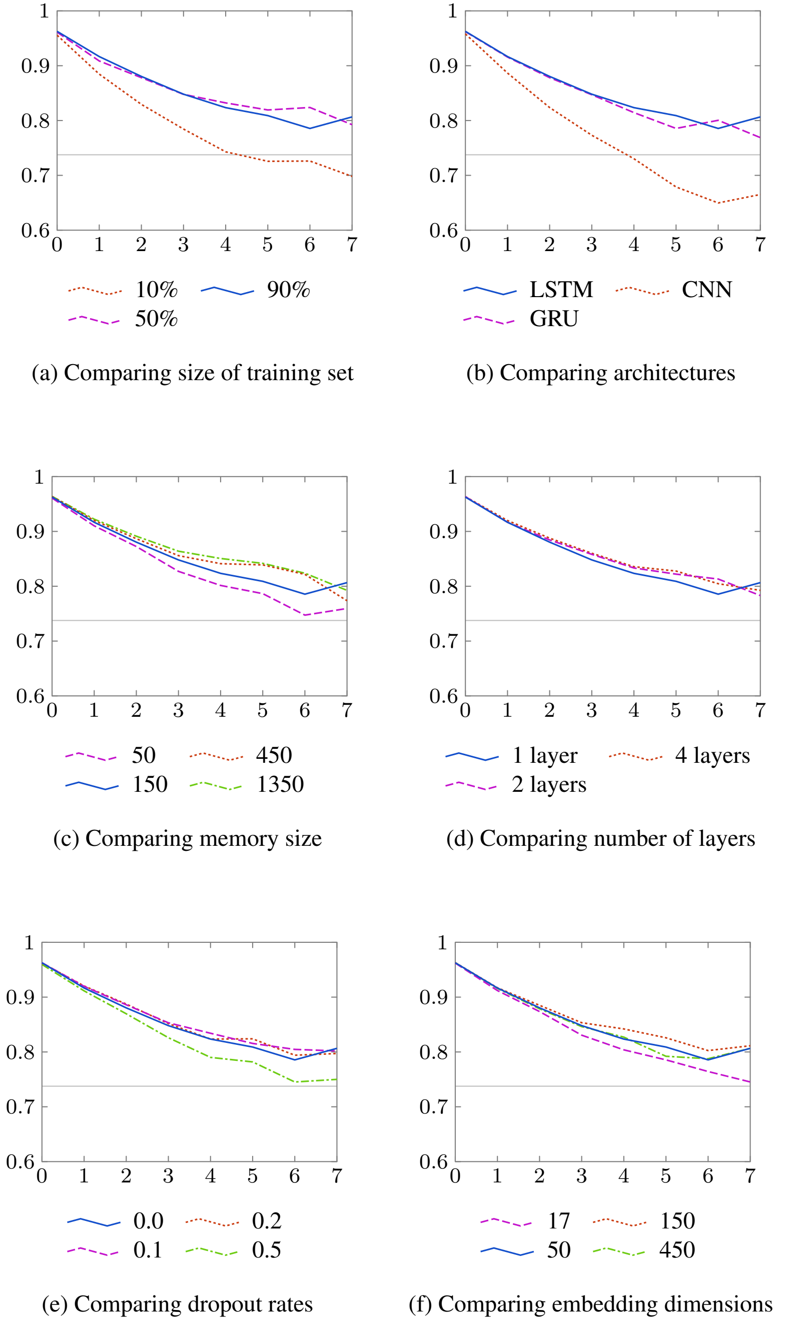
Fig. 4 shows the performance of a DNN configured with the best observed parameter values, which are model = LSTM RNN, layers = 2, number of units = 1350, dropout rate = 0.1, vocabulary size = 100k, training = 90%, of the corpus, and lexical embedding size = 150 dimensions. This DNN provides the highest level of accuracy of all the systems that we tested.

Finally fig. 6 displays the inverse relation between the number of examples and the number of attractor NPs.

1 Discussion

Our results support Linzen et al.'s finding that RNNs, both of the LSTM and GRU variety, are well suited to the task of identifying long distance syntactic dependencies, even when an extended, complex sequence of expressions that could cause confusion intervenes between a controller of a syntactic feature, and the lexical item on which it is realised. However, their success in learning subject-verb agreement scales with the size of the data set on which they train.

Given that RNNs rely on sequential composition of layers of neural units, it is not clear how much hierarchical structure they can identify in their recog-



(a) Comparing size of training set

(b) Comparing architectures

(c) Comparing memory size

(d) Comparing number of layers

(e) Comparing dropout rates

(f) Comparing embedding dimensions

FIGURE 3: Results of testing the effect of various hyper-parameters. The reference (solid blue line) is an LSTM architecture, vocabulary size 10000, training set 90%, single layer, 150 memory units, no drop out.

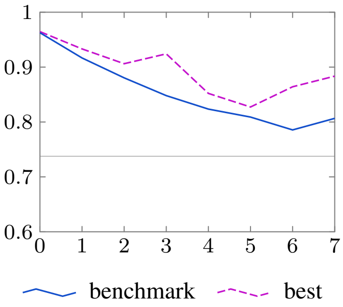


FIGURE 4: Results for a configuration with best parameters values: LSTM RNN with 2 layers of 1350 units, dropout rate 0.1, vocabulary size 100k, training on 90%, and lexical embedding dimension size 150

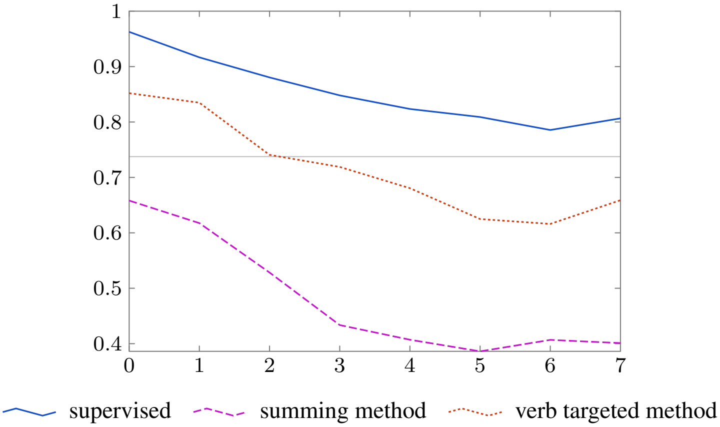


FIGURE 5: Comparing LSTM trained language model (with voc. size 100 and 1000 units) for the two methods of predicting verb number. The solid blue line represents our (supervised) benchmark LSTM RNN.

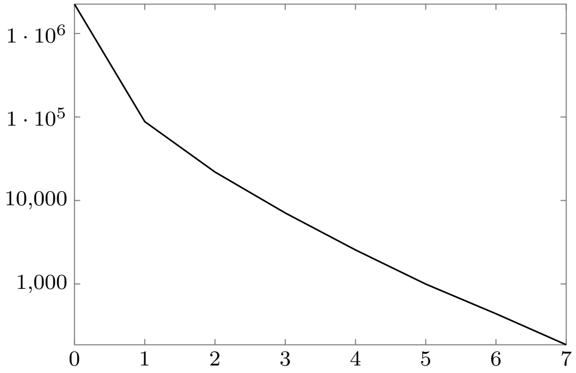


FIGURE 6: Number of test examples per number of attractors

nition of these dependencies. It is also worth noting that we used fairly simple RNNs. It could be that an RNN with a more structured memory that incorporates the equivalent of a stack for encoding the beginning of a dependency and a pop mechanism for releasing it later in a sequence (Grefenstette et al., 2015) would yield even better results.

CNNs have been highly successful in image recognition, which involves constructing levels of successfully more abstract visual feature patterns. Therefore it is surprising that our CNN did not perform particularly well on a task that would, at first glance, appear to require learning hierarchical phrase structure.

It is important to recall that we employed a fairly simple, static CNN model. One with a dynamic memory might yield better performance for this task. In general, there is considerable room for exploring alternative architectures before drawing strong conclusions on the capabilities of the entire class of DNNs for learning syntactic relations.

One of the most striking results to emerge from our experiments is that training DNNs on data that is lexically impoverished, but highlights the syntactic elements between which a relation is to be acquired does not, at least in the current case, facilitate learning. On the contrary, it degrades it. DNNs learn better from data populated by richer lexical sequences. This suggests that DNNs are not efficient at picking up abstract syntactic patterns when they are explicitly marked in the data. Instead they extract them incrementally from lexical embeddings through recognition of their distributional regularities. It is also possible that they use the lexical semantic cues that larger vocabularies introduce to determine agreement preferences for a verb.

It is an open question as to how DNN modes of learning resemble and diverge from human learning. We are making no cognitive claims here. However, it is interesting to note that some recent work in neurolinguistics indicates that syntactic knowledge is distributed through different language centres in the brain, and closely integrated with lexical-semantic representations (Blank et al., 2016). This lexically encoded and distributed way of representing syntactic information is consistent with the role of rich lexical embeddings in DNN syntactic learning.

Finally our results show that a language model can achieve not entirely unreasonable results on the number agreement prediction task, if an appropriate method is applied for comparing the conditional probabilities of alternative number markings on verbs. Our LM is trained on only 100 words, with POS tags substituted for the others in order to highlight the agreement dependency relations that we are seeking to model. This strategy did not improve performance for supervised learning, but it does appear to have been successful for unsupervised learning with a language model.

1 Conclusions and Future Work

Our experimental results strengthen Linzen et al.'s conclusion that DNNs are able to learn long distance syntactic relations to a fairly high degree of accuracy, across extended complex sequences of potentially distracting phrases. We also found that accuracy in the supervised version of this task scales with the amount of training data used, and with the size of the lexical embedding vocabulary.

Performance improves with an increase in the number of hidden units. This effect may be even more pronounced when tracking more complex syntactic relations with multiple features. This is a question that we will explore in future work.

We also found that it is possible to obtain reasonable results with unsupervised learning through a comparatively small language model, when we use a targeted procedure for predicting verb number. The performance of this model, with this procedure, significantly exceeds that of the two models that Linzen et al. present.

In furture work we will experiment with alternative DNN architectures. We are particularly interested in incorporating a structured memory that simulates a stack and pop mechanism for handling long distance dependencies.

We will also test larger language models trained on bigger vocabularies, using our targeted prediction procedure, to see if our conjecture that a smaller vocabulary produces better probabilistic predictions. This proposal did not hold for the supervised learning case, but we do not know if it will be sustained for unsupervised learning with a language model.

One of our main concerns will be to explore syntactic dependencies involving several agreement features. In languages in which gender, and person, as well as number are morphologically realised on verbs the agreement prediction task is more difficult. It requires accuracy across three feature dimensions rather than one. Testing DNNs on agreement in such languages will provide a better sense of their capacity to learn and represent syntactic information.

Acknowledgments

The research reported here was supported by a grant from the Swedish Research Council, which funds the Centre for Linguistic Theory and Studies in Probability in the Department of Philosophy, Linguistics, and Theory of Science at the University of Gothenburg.

We are grateful to an anonymous reviewer for helpful comments on a previous draft of this paper. We thank Tal Linzen for valuable discussion of some of the ideas presented in this paper. Earlier versions of the paper were presented to the Chalmers Machine Learning Seminar in April 2017 and the colloquium of the Cambridge Linguistics Society in May 2017. We thank the audiences of these venues for helpful comments and suggestions. We bear sole responsibility for any remaining errors.

References

Baroni, M., S. Bernardini, A. Ferraresi, and E. Zanchetta (2009). The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation 43*, 209–226.

Blank, I., Z. Balewski, K. Mahowald, and E. Fedorenko (2016). Syntactic processing is distributed across the language system. *NeuroImage 127*, 307–323.

Cho, K., B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014, October). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, pp. 1724–1734. Association for Computational Linguistics.

Chollet, F. (2015). keras. `https://github.com/fchollet/keras`.

Grefenstette, E., K. M. Hermann, M. Suleyman, and P. Blunsom (2015). Learning to transduce with unbounded memory. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, Cambridge, MA, USA, pp. 1828–1836. MIT Press.

Józefowicz, R., O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu (2016). Exploring the limits of language modeling. *CoRR abs/1602.02410.*

Kalchbrenner, N., L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu (2016). Neural machine translation in linear time. *CoRR abs/1610.10099.*

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *CoRR abs/1412.6980.*

Lau, J. H., A. Clark, and S. Lappin (2016). Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*, 1–40.

Linzen, T., E. Dupoux, and Y. Golberg (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association of Computational Linguistics 4*, 521–535.

Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryiğit, S. Kübler, S. Marinov, and E. Marsi (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 95–135.

Schmid, H. (1995). Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*. Association for Computational Linguistics.