Under the Hood: Using Diagnostic Classifiers to Investigate and Improve how Language Models Track Agreement Information

Giulianelli, M.; Harding, J.; Mohnert, F.; Hupkes, D.; Zuidema, W.

Published in:
The 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP

DOI:
10.18653/v1/W18-5426

Under the Hood: Using Diagnostic Classifiers to Investigate and Improve how Language Models Track Agreement Information

Mario Giulianelli University of Amsterdam
Jack Harding University of Amsterdam
Florian Mohnert University of Amsterdam
{mario.giulianelli, jack.harding, florian.mohnert}@student.uva.nl

Dieuwke Hupkes ILLC, University of Amsterdam
d.hupkes@uva.nl

Willem Zuidema ILLC, University of Amsterdam
w.h.zuidema@uva.nl

Abstract

How do neural language models keep track of number agreement between subject and verb? We show that 'diagnostic classifiers', trained to predict number from the internal states of a language model, provide a detailed understanding of how, when, and where this information is represented. Moreover, they give us insight into when and where number information is corrupted in cases where the language model ends up making agreement errors. To demonstrate the causal role played by the representations we find, we then use agreement information to influence the course of the LSTM during the processing of difficult sentences. Results from such an intervention reveal a large increase in the language model's accuracy. Together, these results show that diagnostic classifiers give us an unrivalled detailed look into the representation of linguistic information in neural models, and demonstrate that this knowledge can be used to improve their performance.

1 Introduction

Machine learning models for estimating the probabilities of potential next words (and hence, for predicting the next word) in a running text have seen enormous improvements in performance over the last few years (Merity et al., 2018). These newer models—all based on deep learning techniques such as LSTMs (Hochreiter and Schmidhuber, 1997)—allow some language technologies, such as speech recognisers, to reach 'human parity'. From their high accuracy and from further analysis, it is clear that LSTM-based language models have learned a great deal about both short and long distance relations in sentences and discourse. In particular, Gulordava et al. (2018) report that for several languages, their LSTM-based language model performs remarkably well on a set of long-distance number agreement tasks.

The Gulordava study, however, does not clarify which components of the LSTM are responsible for storing or processing syntactic features, and how such features are represented. Understanding how trained recurrent networks such as LSTMs might represent syntax and other structural information is currently a key area of research. Popular approaches include visualising the state space of these networks, performing ablations to the network, or using the internal states of the networks for some auxiliary task (e.g., Adi et al., 2016; Kádár et al., 2017; Conneau et al., 2018; Khandelwal et al., 2018).

In this paper, we analyse the phenomenon of subject-verb agreement in English using the *diagnostic classification* approach of Hupkes et al. (2018). We start with replicating the results of Gulordava et al. (2018) on English, and we then show that diagnostic classifiers can be used to give a fine-grained analysis of how neural language models capture structural dependencies. In particular, we examine how information about subject-verb agreement is represented by an LSTM (Section 4), (ii) how that information varies across timesteps (Section 5), and (iii) where and how the problems arise that let the model commit agreement errors (Section 5 and 6). Finally, to demonstrate how precisely and accurately this method can identify the network's internal representations, we (iv) show that we can alter the representation to strongly improve the models ability to predict verb number (Section 7). In the next section, after discussing subject-verb agreement, we outline the data used throughout our experiments.

Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 240–248
Brussels, Belgium, November 1, 2018.

2 Data

The work in this paper focuses on understanding how recurrent neural language models can understand subject-verb agreement, which is used as a proxy for understanding syntactic structure. In this section, we discuss subject verb agreement and the type of sentences we look at throughout the rest of this paper. We then briefly describe the data that we use for our investigation.

2.1 Subject-verb agreement

Subject-verb agreement is a variable-distance syntactic dependency, and a classic example of a structural dependency in natural language (Chomsky, 1957; Tesnière, 1959). In English, a present tense verb and the head of its syntactic subject must agree on their number (singular or plural). Thus, "The **dog chases** the cat" is grammatical, whilst "The **dog chase** the cat" is not. In principle, subject and verb can be separated by an arbitrary number of tokens, often including other nouns with a potentially different number (for an example, see Figure 1). We call the number of tokens between the subject head and the mean verb the *context size*.

Without any syntactic analysis, it is unclear how to identify all subject-verb pairs in a sentence within an arbitrarily large window of tokens, especially since intervening nouns can themselves be candidates for agreement. To respect subject-verb agreement, a language model needs to detect the grammatical number of both the subject head and the verb, store this information across timesteps, and identify which nouns correspond to which verbs. When intervening nouns carry the opposite grammatical number from the subject head—as do both intervening nouns in the example sentence in Figure 1—we refer to them as *agreement attractors*, or simply *attractors*.

2.2 Datasets

For the experiments described in this paper we use two different datasets. The first is the one introduced by Gulordava et al. (2018), which contains 410 sentences with at least three tokens occurring between subject head and verb. For each of 41 original sentences, nine 'nonce' variants were generated by substituting each context word in the sentence by a random word with the same part-of-speech tag and morphological features. This data construction method is motivated by the fact that grammaticality judgements should not be influenced by the meaningfulness of a sentence, and ensures that frequency-based confounds are avoided. Every sentence in the dataset is annotated with the correct and incorrect verb forms, the morphological features of the former, the position of the subject head and of the verb, the number of agreement attractors, and the type of construction spanning the long-distance dependency.

Additionally, we extract different subsets of the Universal Dependency (UD) corpus (ca. 1.5 million sentences) for our experiments. The large amount of annotated sentences in this dataset allows us to retrieve sets of sentences that satisfy specific conditions relevant to subject-verb agreement. In particular, we can extract sentences with specific context sizes, and fixed numbers of words before the subject and after the verb. We are also able to specify whether the sentences in the set should have an attractor and—if so—at which index (or, in our terminology, *timestep*) the attractor should appear. Similarly, we can ensure that there is no other noun between subject and verb that has the same number as the subject (we call these *helpful nouns*). As we will see, this allows us to examine the dynamic effect of attractors in the way the LSTM processes subject-verb agreement.

In this paper, the specific subset of the universal dependency dataset we use varies from experiment to experiment, as different experiments require different constraints. We will specify our selection of data for each experiment in the relevant sections. To clarify which subset of the UD corpus is used in an experiment, we use the following notation: *UD-Kk-Ll-Mm-Aa*, where k refers to the minimal number of words appearing before the subject, l to the number of words between the subject and verb (the context size), m to the minimal number of words after the verb, and a to the position of the attractor relative to the subject. We use an asterisk to indicate that no restrictions are placed on one of the above mentioned variables; e.g., $A*$ indicates that there may or may not be an attractor. Finally, we denote datasets of sentences that have no attractor with a minus following the attractor index (i.e., $A-$).

3 Replication

We start with replicating the experiment performed in (Gulordava et al., 2018), using the pretrained LM and the English test set made available

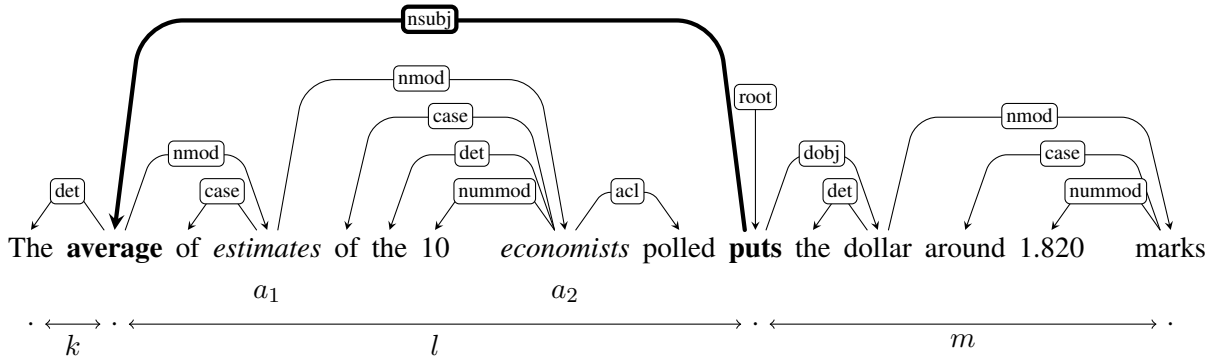

Figure 1: An example dependency parse of a sentence with a singular subject head and main verb (marked in boldface). As the subject *average* and the verb *put* are separated by 7 tokens, the *context size* (l) of this sentence is 7. Within this context, there are two intervening plural nouns, *estimates* (a_1) and *economists* (a_2), which we call *agreement attractors*.

by the authors of the paper.[1] Following Linzen et al. (2016) and Gulordava et al. (2018), we use the LSTM language model to process a corpus of sentences containing long-distance subject-verb relations, and test whether the model assigns a higher probability to the verb that originally occurred in the sentence than to its incongruent counterpart.

	Gulordava et al.	**Our Accuracy**
Original	81.0	78.1
Nonce	74.1	70.7

Table 1: LM accuracy on both English sets from Gulordava et al. (2018). Reported are the percentages of sentences for which the correct verb form is assigned a higher likelihood under the LM than the incorrect form.

In Table 1 we report both Gulordava's original accuracies, and the results from our replication. Overall we obtain similar results, but our accuracy scores are slightly lower[2] than those reported by Gulordava et al. (2018).

4 Diagnostic Classification to Predict Number

After confirming Gulordava et al. (2018)'s results, we now investigate *how* the LSTM represents the required number information, how this information is built up over time and where in the network the representation resides. To this end, we use diagnostic classifiers (DCs, Hupkes et al., 2018). The key idea of diagnostic classification is to test whether an LSTM's intermediate representations contain information about a particular phenomenon—such as subject-verb agreement—by training another model to recognise the information relevant to the phenomenon in the internal activations of the LSTM. More precisely, given a dataset of intermediate LSTM representations and a set of labels that describe the hypothesis to be tested, a meta model can be trained to predict the correct label from the representations. If the model succeeds in this task (i.e. if it achieves a performance significantly above chance on test data disjoint from the training data), this constitutes evidence that the LSTM is in fact computing or keeping track of the hypothesised information.

Training We create a training set containing 1000 sentences that all have 5 words between subject and verb (i.e. the context size is 5), have at least one word before the subject and after the verb, and for which no attractor based constraints are placed on the training set (*UD-K1-L5-M1-A−*). We run the pretrained LM of Gulordava et al. (2018)—a two layer LSTM model with 650 hidden units—on this corpus, and for both layers we extract activation data for both the hidden and gate activations (the hidden activation $\mathbf{h_t}$ and memory cell $\mathbf{c_t}$, and the forget gate $\mathbf{f_t}$, input gate $\mathbf{i_t}$ and output gate $\mathbf{o_t}$). For example, for a single sentence of length n we obtain $5 \times 2 \times n$ activation vectors, because we have 2 layers, *n* timesteps, and 5 types of

[1] `github.com/facebookresearch/colorlessgreenRNNs/tree/master/data`

[2] The results we obtain with our implementation exactly match those we get when running the script publicly shared by Gulordava et al. (2018); we currently have no explanation for the discrepancy in overall scores but consider the differences small enough to proceed with the real purpose of our study: understanding how the models work.

	$\mathbf{h_t}$	$\mathbf{c_t}$	$\mathbf{f_t}$	$\mathbf{i_t}$	$\mathbf{o_t}$
Layer 0	**0.74** / 0.57	**0.76** / 0.58	**0.69** / 0.55	**0.68** / 0.56	**0.69** / 0.56
Layer 1	**0.90** / 0.62	**0.91** / 0.65	**0.86** / 0.61	**0.86** / 0.60	**0.87** / 0.60

Table 2: Mean accuracy of DCs (correct/wrong) across timesteps, averaged over datasets drawn from different context sizes and attractor positions (with $K = 0$, $M = 0$, $5 \leq L \leq 7$ and with a variable number of attractors at different positions).

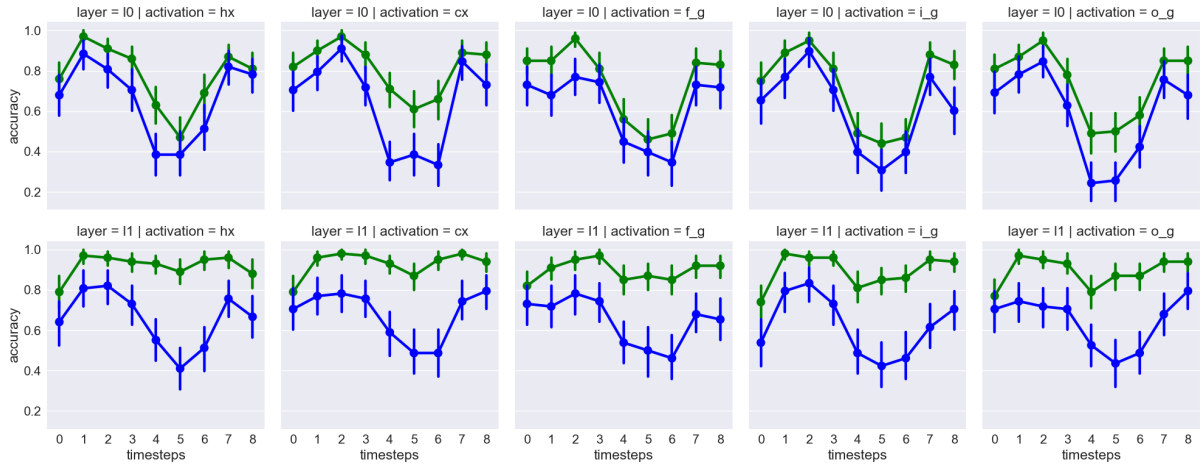

Figure 2: Accuracies over time (on UD-K1-L5-M1-A3) of 10 diagnostic classifiers trained and tested on data from different components of the LSTM. As in this testset one word occurs before the subject, the subject is at timestep 1. Green lines represent sentences for which the LSTM predicts the correct verb, blue lines sentences for which the LSTM assigns a higher probability to the incongruent counterpart.

activations at each time step t: $\mathbf{h_t}, \mathbf{c_t}, \mathbf{f_t}, \mathbf{i_t}, \mathbf{o_t}$). We then label all activations with the number of the main verb of the sentence from which it was generated (either 'singular' or 'plural') and train a separate DCs for each of the 10 components of the LSTM.

Results We test the trained DCs on two test sets, that differ with respect to whether the LM correctly or incorrectly classified the sentences they contain (i.e. a sentence s is in the 'correct' set iff the LM assigns higher probability to the correct form of the sentence than to the incorrect form). Otherwise, the two sets have similar features, containing both sentences from *UD-K1-L5-M1-A3*. While we strive to generate the 'wrong' and 'correct' test sets with 100 sentences each, this is not always possible due to data sparsity. However, we ensure that both test sets have approximately the same size and do contain at least 50 sentences.

In Table 2, we print the average DC accuracies. We observe that for both the 'wrong' and the 'correct' test sets, the accuracies are highest at the second layer (layer 1) across almost all LSTM components, suggesting that the last LSTM layer reaches the level of abstraction which can best capture long-distance dependencies.

In Figure 2, we plot the average DC accuracy at different timesteps when processing sentences (from a set with a context size of 5 and a single attractor located three words after the subject). Unsurprisingly, the DCs obtain their best accuracy scores at (or just after) the subject and verb timestep. This pattern is consistent across context sizes, attractor positions, and number of words before the subject and after the verb, and regardless of whether the LSTM prediction was correct or incorrect. This result illustrates that the LM learns to recognise the number information of subject heads and present tense verbs.

The figure furthermore shows that performance differs between layers and between components. The DC performance of the layer 1 components, moreover, critically differs for 'correct' and 'wrong' sentences, For example, classifiers that make predictions based on $\mathbf{c_t}$ and $\mathbf{h_t}$ activations of 'correct' sentences are the most stable in terms of accuracy, in particular at layer 1. Although all LSTM components outperform the random base-

line of 50%, these results imply that the cell state and the hidden activation are the LSTM components that are most specialised at processing number information. We test this claim in Section 5.

Another cause of differences across diagnostic classification error rates is the presence of agreement attractors. Accuracies for the test sets with an attractor are overall lower than those obtained on sentences without an attractor. While the error rate rises in Figure 2 and diverges between 'correct' and 'wrong' at the position of the attractor, the same does not happen for sentences without attractors (not plotted).

5 Representations Across Timesteps

Results so-far show us that number information is most easily retrieved from the internal states of the LM when the noun or verb have just been presented, but not very well from the internal states at intermediate timesteps. The good performance of the LM in predicting the number of the verb, however, indicates that the LM does retain the subject's number information during those intermediate timesteps—but apparently it does so using a *different* representation. In this section, we focus on these changing representations.

In the previous experiment we trained diagnostic classifiers on activation data for all words in the sentence. In contrast, we now train *separate* diagnostic classifiers for each timestep: each DC_t is trained with activation data at timestep t only. We test, however, each DC_t on data from all other timesteps as well. With a total of T timesteps, this gives us $T \times T$ DC-accuracies that together constitute a *Temporal Generalization Matrix* (King and Dehaene, 2014; Fyshe et al., 2016).

In effect, we are forcing each DC to specialise on timestep-specific representations of subject-verb agreement information. If this information is represented uniformly across timesteps, a classifier trained at the subject timestep should also have a high accuracy when applied to the activations corresponding with the timestep in which the attractor occurs. If, on the other hand, information is dynamically encoded, no such generality of classifiers is to be expected.

Data To test the development of the encoding over time, we create a corpus with sentences that are identical with respect to the position of the subject, attractor and main verb. We train on sentences with 5 intervening words between the subject, containing one attractor 3 timesteps after the subject, and a variable number of words before the subject and after the verb (*UD-K*-L5-M*-A3*). After computing the activations for all sentences, we collect the activations corresponding to all 6 timesteps from subject to verb, in 6 different bins. For each bin, we train a separate DC.

For testing we create again a 'correct' and an 'incorrect' test set, drawing both sets from *UD-K*-L5-M*-A3*. Following the same procedure as for the training data, we split both test sets up into 6 timesteps. In the remainder of this section, position 0 thus always refers to the position of the subject, while the attractor and main verb of the

Figure 3: The temporal generalization matrices for DCs trained on memory cell activation at different timesteps, for correctly (top) and incorrectly classified (bottom) sentences. Timestep 0 corresponds to the subject of the sentence, the attractor and main verb of the sentence occur at timesteps 3 and 6, respectively. The corpus used for testing here is *UD-K*-L5-M*-A3*.

sentence occur at timestep 3 and 6, respectively.

In Figure 3, we plot the Temporal Generalization Matrix for the memory cell ($\mathbf{c}_t^1$) activation data, containing the accuracies of T DC's evaluated on T timestep datasets each. The top figure plots results for 'correct' sentences, the bottom figure for 'incorrect' sentences.

A first observation is that accuracies on the diagonals—which correspond to classifiers that were trained and tested on the same timestep—are typically high for sentences that are processed correctly, while being lower for incorrectly processed sentences. Interestingly, this difference already emerges at the first two timesteps, where no attractor has yet appeared—suggesting that an important part of the problem with misclassified sentences is the encoding of the relevant information already when the subject occurs.

Comparing the plots for correctly and incorrectly processed sentences, we notice that the attractor (timestep 3) has a very large effect on the accuracies for incorrectly classified sentences. For those sentences, the LM's internal states contain no information anymore after the attractor is processed: timesteps 4 and 5 receive below chance accuracies, whereas for correctly processed sentences the attractor prompts only a slight dip in accuracy.

Focussing on the correctly processed sentences, an interesting observation that can be made is the discrepancy between column 0 and 6 (the columns corresponding to the subject and verb of a sentence) and the rest of the columns. While the first and last column generalise poorly to different timesteps, the classifiers trained and tested on timesteps 1-5 show a different pattern: despite potential effects from the attractor at timestep 3, the accuracy scores do not change substantially across timesteps. This implies that the LSTM represents subject-verb agreement information in at least two different ways: a short-term 'surface' level at and around the subject timestep, and a longer-term 'deep' level for successive sequence processing. This deep level information seems to be represented most generically at timestep 4, the classifier for which has the highest accuracy across timesteps.

In the next section, we delve deeper into the representations at this timestep and investigate which components of the LSTM are most crucial in representing this information.

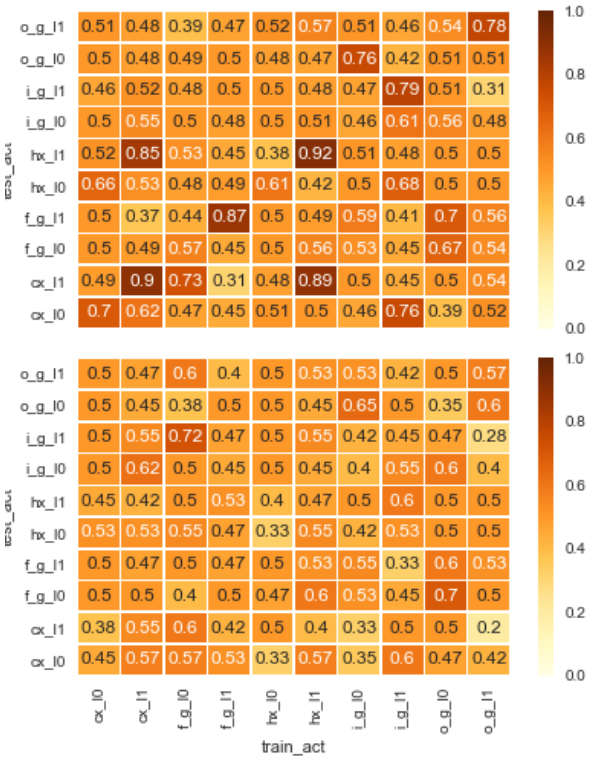

Figure 4: The spatial generalization matrices at timestep 4. Shown are accuracies of DCs trained on activation data of each component separately (horizontal), and tested on each component separately (vertical). Results for correctly (top) and incorrectly (bottom) classified sentences.

6 Comparing Representations Across Components

In this section, we briefly investigate the stability of information across components of the LSTM. Rather than comparing DCs that are trained on different *timesteps*, we now compare DCs that are trained on different *components*. We focus on timestep 4 which, following our previous experiments, optimally represents 'deep' information about subject-verb agreement. For our experiments, we use the same training set as for the previous experiment, with sentences with a context size of 5 and a single attractor located three words after the subject (*UD-K*-L5-M*-A3*).

Figure 4 presents the 'spatial generalization matrix', with DCs trained at timestep 4 with data from each components separately. The matrix shows that deep information is best represented in the hidden activation and memory cell of layer 1, and that the representations in these two components are similar.

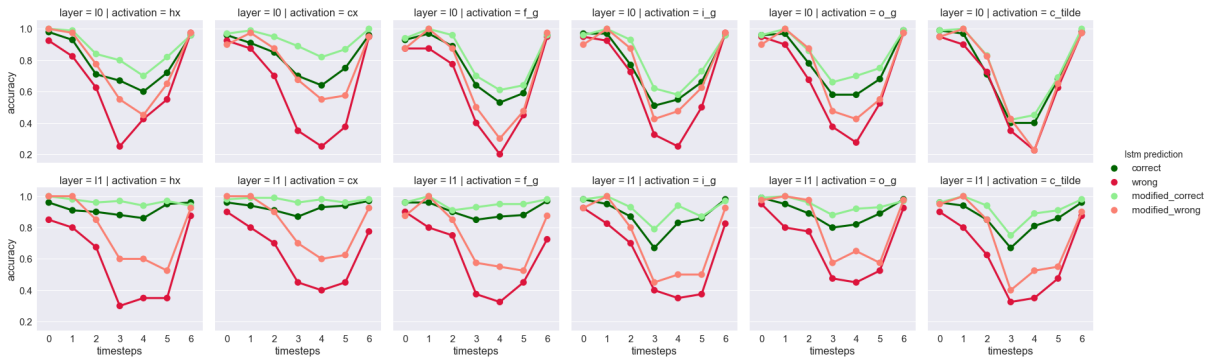

Figure 5: Mean accuracies for each component of the LSTM after an intervention of $\mathbf{c_t}$ and $\mathbf{h_t}$ at the subject timestep 0. An attractor and the agreeing verb occur at timestep 3 and 6, respectively.

	An	official	estimate	issued	in	2003	suggests	suggest
Original		-11.05	-8.426	-8.472	-1.243	-3.951	-5.753	-5.6979
Intervention		-11.05	-8.426	-8.472	-1.268	-3.97	-5.691	-6.4361

Table 3: Example sentence as processed by the neural language model of Gulordava et al. (2018), without and with our intervention. Shown are perplexities per word, for two versions of the sentence (featuring the verb 'suggests' or 'suggest').

7 Improving the Language Model Using Diagnostic Classifiers

In the experiments presented above, we used diagnostic classifiers to investigate the way the LSTM performs the verb number prediction task. In this section, we take one step further: rather than using DCs to analyse what neural networks are encoding, we try to use them to actively influence their behaviour through what they learned.

We use the same data as we used for the experiments described in the previous section: a corpus of sentences with the subject at timestep 0, one attractor 3 timesteps after (at timestep 3) and the main verb at timestep 6 (*UD-K0-L5-M0-A3*). We train 4 DCs to predict the number of the sentence from the hidden layer activations and memory cell activations for both layers, respectively.

We then use the trained DCs to actively influence the course of processing by the LSTM. We start processing sentences from the Gulordava et al. (2018) corpus, but after processing the subject of a sentence—the point where we discovered information is stored in a corrupted way for 'wrong' sentences—we halt the LSTM's processing, extract the hidden activation and the activation of the memory cell, and apply the trained diagnostic classifier to predict whether the main verb in the sentence is singular or plural. We then slightly adapt the activations based on the error that is defined by the difference between the predicted label and the correct label for this particular sentence. We compute the gradients of this error with respect to the activations of the network, and we modify the activations using the delta-rule (we empirically decided on $\eta = 0.5$). In other words, we change the activations such that the prediction of the diagnostic classifier is slightly closer to the gold label. After adapting the activations, we continue to process the rest of the sentence given the adapted activations.

DC accuracy In Figure 5 we plot the accuracies of DCs trained on different components of the LSTM when we apply them on activations resulting from sentences processed with the above described intervention. Trivially, the intervention increases the accuracy of DCs for the hidden activation and memory cell of the network at timestep 1. More interestingly, this effect persists while the processing of the sentence proceeds—in some cases it grows even stronger—and thus in fact *changes* how the LSTM processes the sentence. This effect is not only visible in the components on which the intervention is done, but also displays in the gate-values, that are not directly updated but only changed indirectly through the interventions in the memory cell and hidden activations.

Language modelling To put our interventions to the test, we now assess the predictions made

	without intervention	with intervention
Original	78.1	85.4
Nonce	70.7	75.6

Table 4: Accuracy of the LSTM on the Gulordava et al. (2018) agreement test, with and without an intervention at the subject timestep.

by the LSTM as a consequence of the interventions. First, we confirm that the intervention does not cause strong anomalies in the LSTM, by comparing the perplexity of a small corpus of sentences processed *with* interventions at the subject timestep with sentences processed without any interventions. Table 3 shows an example sentence. We do not find any strong differences, confirming that the intervention is minor with respect to the overall behaviour of the LSTM. On the agreement test described by Gulordava et al. (2018) and conducted earlier in Section 3, however, the intervention *does* have a strong effect, as can be seen in Table 4. The accuracy of predicting the correct verb number increases from 78.1 to 85.4 and from 70.7 to 75.6 for original and nonce sentences, respectively.

These results provide evidence that DCs are able to pick up features that are actually used by the LSTM, rather than relying on idiosyncrasies in the high dimensional spaces that happen to be aligned with the predicted labels. Furthermore, they illustrate how diagnostic classifiers can be used to actively change the course of processing in a recurrent neural network, and with this opens a path that moves from merely *analysing* to actively *influencing* black box neural models.

8 Conclusions

In this paper, we focus on understanding how an LSTM language model processes subject-verb congruence, using a task first presented by Linzen et al. (2016), in which it is tested whether a language model prefers congruent over incongruent verbs. After replicating their results, we train diagnostic classifiers (Hupkes et al., 2018) to discover where and how number information is encoded by the LSTM; we find that number information is encoded *dynamically* over time, rather than remaining constant. Using a cognitive-neuroscience inspired method, we then train different diagnostic classifiers for different timesteps, resulting in a *Temporal Generalisation Matrix*, which provides more information about changing representations over time. We find that while number information is stored in very different ways at the beginning and end of a sentence, in between a relatively stable 'deep' representation is maintained. Additionally, we find that for sentences in which the LSTM prefers an incongruent verb over congruent one, the information appears to be stored wrongly already at the beginning of the sentence, far before the verb is to appear.

Combining this information, we invert the process of diagnostic classification, using the classifiers to *influence* rather than merely observe. To this end, we process sentences with our language model and, at the point where we find information to be often corrupted, we intervene by (slightly) changing the hidden activations of the network using a trained DC. After this intervention, we continue processing the sentence as normal. This small intervention has little effect on the overall course of the LSTM, but a very large effect on the verb prediction at the end: the percentage of sentences for which the model prefers the congruent over the incongruent verb rises from 78.1% to 85.4%.

With these results, we not only show that diagnostic classifiers offer a detailed understanding of where and when information is encoded in a neural model, but also that this information can be used post hoc to change the course of the processing of such a model.

Acknowledgements

DH and WZ are funded by the Netherlands Organization for Scientific Research (NWO), through a Gravitation Grant 024.001.006 to the Language in Interaction Consortium.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations (ICLR)*.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing

sentence embeddings for linguistic properties. In *Association for Computational Linguistics (ACL)*.

Alona Fyshe, Gustavo Sudre, Leila Wehbe, Nicole Rafidi, and Tom M Mitchell. 2016. The semantics of adjective noun phrases in the human brain. *bioRxiv*, page 089615.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco 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)*, volume 1, pages 1195–1205.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Association for Computational Linguistics (ACL)*.

JR King and Stanislas Dehaene. 2014. Characterizing the dynamics of mental representations: the temporal generalization method. *Trends in cognitive sciences*, 18(4):203–210.

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*, 4:521–535.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Lucien Tesnière. 1959. *Eléments de syntaxe structurale*. Klincksieck, Paris.