LM investigation using diagnostic classifiers: LSTMs challenge Transformers' DIBS

607 and **kerkeruil**
Made for an assignment as part of the MSc Artificial Intelligence at the University of Amsterdam.

Abstract

Language models have recently made significant advancements in generating and understanding human language. But whether these models have an understanding of the underlying structures within language remains a subject of ongoing research. A common approach to exploring the mechanics of these models is by using probes or diagnostic classifiers (DCs). These DCs are small models trained to interpret language models and make predictions based on the information encoded within them. One suggested metric to investigate DC performance is minimum description length. Additionally, we suggest a novel metric. Findings include that Transformers and LSTMs both represent POS well, and both aren't as strong on tree representations, and that sentences that get POS tagged correctly generally also get good tree representations.

1 Introduction

In natural language processing, language models have played a major role since 2017. Language models (LMs) are neural models that are pretrained on a large dataset, learning general-purpose features of the input data. LMs can be finetuned on a supervised task (Liu et al., 2023). Contrary to feature engineered models, it is not necessarily clear what LMs represent internally, or why a given model works well (AlKhamissi et al., 2022; Alammar, 2021; Belinkov and Glass, 2019).

In this research, we will draw a comparison between LSTMs (Recurrent Neural Networks) and Transformers (Attention-based Networks). Transformers are commonly known to be easier to train and more stable, achieving similar or higher accuracy than the LSTM, but can also overfit more easily (Zeyer et al., 2019). Ezen-Can (2020) claim that this can be dependent on the tasks and that in some cases LSTM can be faster with comparable or higher accuracy.

One technique to study the contents of LM representations is to train a small model to perform a task based on the embeddings generated by the LM. This technique has been referred to under various names, such as *auxiliary prediction tasks* (Adi et al., 2017), *probes* (Hewitt and Manning, 2019) and *diagnostic classifiers* (Hupkes et al., 2018). We adhere to the latter term.

However, diagnostic classifiers (DCs) do not necessarily use structure present in the embeddings of the model, leading to uncertainty in the interpretability of DC accuracy (Zhang and Bowman, 2018). For this reason Voita and Titov (2020) proposed incorporating a method from the field of information theory to evaluate the DCs, called Minimum Description Length (MDL) (Rissanen, 2001). This method should provide more informative and stable insight into how well an LM captures linguistic properties.

In the present research we use DCs to investigate how well LSTMs and Transformers represent POS and tree structure, evaluating on both classical measures and MDL. Furthermore, we propose a new method to evaluate DCs, requiring less computation power and being easy to implement: *direct iteration-based score* (DIBS).

Experimental findings suggest that Transformers and LSTMs represent POS equally well, outperforming random baselines. For tree structure, they also seem on par, but don't outperform their baselines as much. We find that in general, sentences that get POS tagged correctly also get better trees.

2 Background

Diagnostic classifiers have been used to look into the representation of various types of linguistic information. For example, Tenney et al. (2019) trained diagnostic classifiers for 11 different tasks, among which POS tagging, dependency labeling and relation classification. Like the present research, they considered both LSTM and Trans-

former architectures, comparing to various non-contextual baselines, and finding that contextual models mostly improved over non-contextual models on syntactic tasks rather than semantic tasks. Hewitt and Manning (2019) made a structural diagnostic classifier, which we partly reuse in this work, to investigate the representation of syntax trees in embeddings, and finding evidence that syntax trees are embedded in both LSTMs and Transformers.

The reliability of diagnostic classifiers has been called into question (Belinkov, 2022). Zhang and Bowman (2018) found that diagnostic classifiers can learn POS tagging to almost as high accuracy on a randomly initialized LSTM as on a pretrained LSTM, provided the diagnostic classifier gets enough training data. Hewitt and Liang (2019) investigated the performance of diagnostic classifiers on *control tasks*: tasks similar to linguistic tasks, but associated with random outputs. They showed that under many popular hyperparameter settings, diagnostic classifiers could get similar accuracy on control tasks and linguistic tasks. In response, Voita and Titov (2020) proposed evaluating Minimum Description Length instead of accuracy, which will be further explored in section 3.

Besides diagnostic classifiers, other directions of interpretability research include visualization, challenge sets and adversarial examples. For a full review see Belinkov and Glass (2019).

3 Approach

To investigate how LSTMs and Transformers represent POS and tree structure, we train two types of Diagnostic Classifiers (DCs) on both types of architecture. We also train DCs on embeddings from randomised versions of the same model architectures, as a baseline. A DC is trained like any regular neural network but takes in the output embeddings as input instead of the original sentence. Hence, a DC is specific to a model. The training objective is to predict some property of language. If the DC shows good performance, this may suggest that the investigated model's embeddings represent this property of language (Belinkov, 2022).

However, performance alone might not be enough. Thus we also calculate Minimum Description Length (MDL), where a higher MDL implies the embeddings are harder to decode (Voita and Titov, 2020). Additionally, we evaluate using *direct iteration-based score* (DIBS), a newly proposed method, which we evaluate by comparing its scores against that of MDL. Both methods aim to capture the "effort" it takes the DC to learn structure from the embeddings.

4 Experimental setup

4.1 Data

To train and evaluate our DCs, we used the Universal Dependencies EWT dataset (Silveira et al., 2014)[1], which contains 16621 sentences, annotated with both dependency trees and POS tags.

4.2 Model details

For the evaluated models we use two Transformers and three LSTMs. Specifically, we use DistilGPT2, a distilled version of GPT2 (Radford et al., 2019), and XLNet (Yang et al., 2019) for the Transformers, and for the LSTMs we use the LSTM architecture from Gulordava et al. (2018), one model from that paper and two more from Jumelet et al. (2021), trained in the same way with different seeds.

As our DC for POS, we use a linear model that generates a transformation from the embedding size to the vocabulary size (i.e. amount of different POS tags), using Cross Entropy Loss. As our structural DC for trees, we use the implementation from Hewitt and Manning (2019), which uses L1 Distance Loss.

4.3 Training procedure

To train our DCs, we used Adam (Kingma and Ba, 2015). Following Voita and Titov (2020) and Hewitt and Liang (2019), we start training with a learning rate of 0.001, and anneal by 0.5 every time no improvement is made on the evaluation loss after an epoch. If four such epochs occur consecutively, we stop training. We also stop training if 40 epochs have passed, following Hewitt and Manning (2019).

4.4 Metrics

For DC evaluation we use three different methods: accuracy, MDL and DIBS. For the structural DC accuracy is swapped for *Unlabeled Undirect Attachment Score* (UUAS), measuring the recall of the predicted edges[2].

[1]Available at https://universaldependencies.org/treebanks/en_ewt/index.html.

[2]Accuracy wouldn't work here because a tree would need to be completely correct

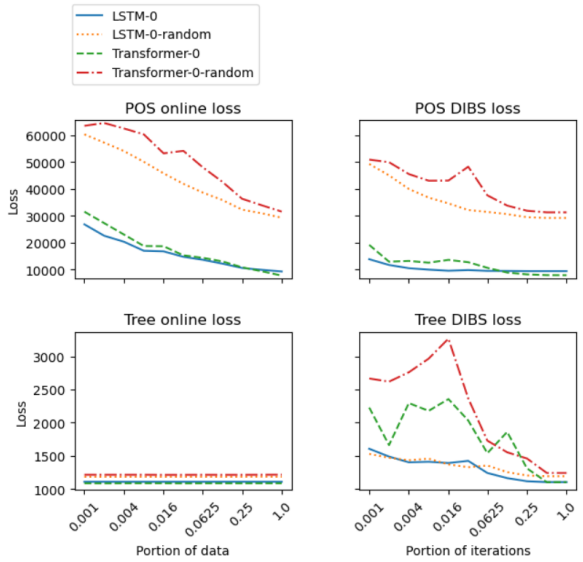

Figure 1: **Test losses for online code and DIBS.** Shown are our first LSTM and our first Transformer, including their random baselines. We see that the metrics follow a similar trend for the POS DC but differ significantly for the structural DC. The online loss for the structural DC shows no improvement on larger data portions but the DIBS loss shows improvement over a longer period of training.

For MDL, we replicate the online code by summing the losses[3] on the test set of DCs trained on increasing portions of the dataset[4] (Voita and Titov, 2020).

Our newly proposed method DIBS requires training a DC just once, and sums the test losses obtained on portions of the amount of iterations. If a model converges early, the losses are padded with the loss obtained from the last finished portion.

5 Results and discussion

In Figure 1 we show the test losses that are used to calculate the online code (for MDL) and the DIBS, for our first LSTM and our first Transformer, including their random baselines. DIBS and MDL show similar trends for the POS DCs on all inspected models, which suggests that DIBS could be an alternative for the online loss. However, for the structural DCs, the shown trend differs greatly. For MCL, the losses start out very low and stay stable. This suggests, according to Voita and Titov (2020), that the models contain a clear representation of syntactic structure, as it can be learned from little data. It is remarkable, however, that this is the case for the random baselines as well. DIBS, on the other hand, more closely matches our expectations, as the loss starts out bigger but gradually decreases. However, here also we do not observe a big difference between the trajectories of the pretrained models and the random baselines. We suggest further research to compare DIBS and MCL, possibly also incorporating the variational loss as detailed in Voita and Titov (2020).

The full end results of our experiment can be found in Table 1. Both model types achieve similar results. DIBS and MDL both trend in favor of LSTMs, although the standard deviation on the Transformer results makes this more debatable due to the difference in Transformer performance, which seems to be present in all Transformer results. We suggest that this happens because two different Transformer architectures were used, of which one (XLNet) performed significantly worse than the other (DistilGPT2), whereas three LSTMs of identical architecture were used. This begs the question whether the results produced accurately compare the differences between LSTMs and Transformers. For future research, we suggest obtaining both more different Transformer architectures and different LSTM architectures, in an attempt to obtain more consistent and clearer results. It should be noted that LSTMs and Transformers are both types of contextual models, so it is not necessarily surprising that they both represent types of syntactic information well (Tenney et al., 2019).

Both LSTMs and Transformers represent POS well, according to our results, in line with Tenney et al. (2019), and greatly outperform their random baselines. Both model types do not strongly outperform their random baselines when training a structural DC, however. Comparing against results from Hewitt and Manning (2019), our DCs perform worse than the DCs on all their tested contextual models, but better than all their non-contextual baselines. The former might be accounted for by our tested models being of similar age but smaller size than the models tested in Hewitt and Manning (2019). The higher baseline performance could be caused by many easy examples in the dataset, as will be elaborated below.

Table 2 shows a comparison of DC performance for POS and trees. There appears to be a correlation between the performance of a DC for assigning

[3]When using L1 Distance Loss summing the losses does not exactly correspond to codelength, but the technique should still apply.

[4]The portions used are 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.25, 12.5, 25, 50 and 100 percent of the dataset.

	POS acc	POS MDL	POS DIBS	UUAS	Tree MDL	Tree DIBS
LTSM	0.86 ± 0.02	211 ± 26	137 ± 17	0.65 ± 0.03	13 ± 1	15 ± 0
LSTM-random	0.62 ± 0.01	485 ± 7	397 ± 7	0.61 ± 0.01	13 ± 0	15 ± 0
Transformer	0.89 ± 0.03	278 ± 88	233 ± 105	0.64 ± 0.08	16 ± 4	49 ± 30
Transformer-random	0.68 ± 0.07	455 ± 94	368 ± 78	0.57 ± 0.02	17 ± 3	48 ± 24

Table 1: **Results of training POS and structural DCs on LSTMs and Transformers, including random baselines.** For the Transformers, results were averaged over two models with different architectures. For the LSTMs, results were averaged over three models with the same architecture. For both types of models, comparable accuracy is achieved while outperforming their random baselines. UUAS for LSTMs and Transformers is also similar, but here the random baselines are not so clearly outperformed. The transformer results have a very high standard deviation overall, complicating the drawing of conclusions. It does seem that Transformers get significantly higher DIBS on trees than LSTMs.

Table 2: **Comparison of performance of tree and POS DCs per sentence.** Sentences are separated in bins based on accuracy and in bins based on UUAS, where 1-25 means the worst 25%, and so forth. The number of sentences that is in each pair of bins is reported in the table, and for each bin the largest pair including that bin is made bold. We note that in most cases, the biggest pairs are on or near the diagonal. One exception is that for DCs trained on LSTM embeddings, there are many sentences that get good tree representations but poor POS tagging. In table 2c examples of sentences in this bin pair can be found.

		Accuracy			
		1-25	26-50	51-75	76-100
UUAS	0-25	**144**	**201**	104	71
	26-50	124	**195**	113	87
	51-75	122	117	**146**	134
	76-100	130	6	**156**	**227**

(a) DCs trained on Transformer embeddings.

		Accuracy			
		1-25	26-50	51-75	76-100
UUAS	0-25	176	**178**	89	77
	26-50	85	161	**180**	93
	51-75	55	149	**158**	157
	76-100	**204**	31	92	**192**

(b) DCs trained on LSTM embeddings.

⟦ http://www.hackneys.net/ ⟧
⟦ - ⟧, ⟦ Will ⟧, ⟦ Rogers ⟧
⟦ VERY ⟧, ⟦ CONCERNED ⟧, ⟦ !!! ⟧

(c) Examples of sentences that get the worst POS accuracy but best UUAS on LSTM embeddings.

POS tags and generating a tree representation, considering that most sentences are on the diagonal.

There is one outlier score for LSTMs, as we find that are many sentences that get good tree representations but poor POS tagging. We looked into the sentences in question and found that they are of a few different types, illustrated in Table 2c. Most of these sentences are difficult to tag for the LSTM since it has never seen most of these 'words', whence they won't be in its vocabulary. The trees for these kinds of sentences, however, tend to be very small and are therefore easier to predict[5]. This problem does not arise for the transformers because they have their own tokenizers. We suspect that those are capable of capturing and recognizing a wider range of word types making them more accurate in POS tagging.

6 Conclusion

Based on the results and explanations presented, we do not support that Transformers better represent syntax than LSTMs. Both model types seem to encode more linguistic features after training when compared to the random baseline, although POS are more strongly represented than trees. Good performance on either POS or Tree DCs indicates good performance for the other aspect as well, but this correlation is more present in Transformers. We propose a novel metric, DIBS, which might be able to replace MDL in DC evaluation. Further research is required to validate this, as well as to get more reliable comparisons between LSTM and Transformer performance.

[5] A predicted tree for a sentence consisting of a single token will in fact trivially get a UUAS of 1.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

J Alammar. 2021. Ecco: An open source library for the explainability of transformer language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 249–257, Online. Association for Computational Linguistics.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Aysu Ezen-Can. 2020. A comparison of lstm and bert for small corpus. *arXiv preprint arXiv:2009.05451*.

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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Jaap Jumelet, Milica Denic, Jakub Szymanik, Dieuwke Hupkes, and Shane Steinert-Threlkeld. 2021. Language models use monotonicity to assess NPI licensing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4958–4969, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jorma Rissanen. 2001. Modelling by shortest data description. *Automatica*, 14.

Natalia Silveira, Timothy Dozat, Marie-Catherine De Marneffe, Samuel R Bowman, Miriam Connor, John Bauer, and Christopher D Manning. 2014. A gold standard dependency corpus for english. In *LREC*, pages 2897–2904. Citeseer.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2019. A comparison of transformer and lstm encoder decoder models for asr. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15.

Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop*

BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 359–361, Brussels, Belgium. Association for Computational Linguistics.