Inspecting Neural Networks

PM Deeplearning summer term 2016

J. Johannsmeier, I. Kurenkov, M. Klotz

August 1, 2016

Project

Looking at the complex architecture of neural networks the following research questions arose:

Questions

1. What exactly is it, what a neural network learns?
2. How is it represented in the network?
3. Can we find and/or track it?
4. What is going on inside a neural network at sampling time?

What are these questions about?

- What exactly is it, what a neural network learns?
- How is it represented in the network?

These two questions depend on the task the network is built to solve and the architecture chosen to do this.

- Can we find and/or track it?
- What is going on inside a neural network at sampling time?

To answer all these questions, common and new means where to be explored to do inspection on several neural *language models*.
More detailed questions are to follow when looking at the experiments.

Outline

- data, tasks and architectures
- the means we used for the analysis
- the experiments in detail
- review of methods
- critique
- references

Data, Tasks and Architectures

Karpathy et al. (2015) ran an analysis on several recurrent neural network architectures learning character sequences of the linux kernel. Observation: Neurons *kept track* of properties like sentence length, line breaks or text within vs. not within quotes.

- train sequence generators with different underlying architectures (simple recurrent, GRU and LSTM)
- train character-level and word-level models
- use the following three datasets:

KJ	King James Bible[a]	
LK	Linux-Kernel[b]	
HDT	Hamburg Dependency Treebank	Foth et al. (2014)

[a] http://www.gutenberg.org/cache/epub/10/pg10.txt
[b] https://github.com/torvalds/linux

Means of Analysis

A short overview

- **Plotting** neuron activations
- Computing various **correlations** of neuron activations with sequence properties
- **P**rinciple **C**omponent **A**nalysis of layer activations
- ERP analysis
- **Autoencoders**

Experiments

Plotting experiments

- E1: Plotting activations of a char-level LSTM for KJ
- E2: Neuron behaviour in a *continuous-state* char-level LSTM for LK

Correlation experiments

- E3: KJ: Do char-level sequence generators learn words?
- E4: Correlating activations of a char-level sequence generator for KJ
- E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

Experiments

Overcoming problems with correlations

- E6: HDT – Clustering correlations
- E7: KJ – PCA and correlations

Further experiments

- E8: Plotting revisited – LSTMVis
- E9: ERP analysis on LK
- E10: Facing the linearity problem with autoencoders

E1: Plotting activations of a char-level LSTM for KJ

motivation:

- Can we reproduce Karpathy et al.'s results or generate similar findings?
- exploring visualizations
- this is addressing all four research questions
- get a first intuition

E1: Plotting activations of a char-level LSTM for KJ

procedure:

- train a 3-layered LSTM (512, 512, 512) on character level for the KJ-data
- feed sentences through the network and plot the activations with hinton diagrams and heat maps
- observe the broad distribution of high and low activation *in a specific layer*
- find neurons, whose activation could be correlating to specific phenomena in the data (a specific character or character sequence, etc.)

E1: Plotting activations of a char-level LSTM for KJ

observations:

- cells tracking *sequence length* and *text structure* (verse number vs. actual verse)

conclusion:

- this experiment (among others) encouraged „more sophisticated“ techniques of analysis, such as computing correlations in a next step
- hinton diagrams where found a useful tool for a quick inspection; I'll be back on this in the correlation chapter

E2: Neuron behaviour in a *continuous-state* char-level LSTM for LK

motivation:

- try to reproduce Karpathy et al.'s results on formal language (character level)
- do especially the cell activations show correlations (visual) with properties of the input text?
- apart from our questions this addresses even more, if it has learnt something and/or if it reveals this something in a way, such that we are able to observe it in the activations

$\rightarrow$ if, where, how, (who)?

E2: Neuron behaviour in a *continuous-state* char-level LSTM for LK

procedure:

- Karpathy et al. (2015) trained their network on continuous sequences with truncated backpropagation through time
- to imitate this behaviour with blocks without running in memory issues, we built a sequence generator, that learns fixed-length sequences (length 250) with batch size 1, while the initial state value for a sequence i is the state's value after reading sequence $i-1$

E2: Neuron behaviour in a *continuous-state* char-level LSTM for LK

results:

- activations of cells in 3rd layer showed reactions on indentation of C-code / spacial characters vs. ńon-white-space characters
- in addition the hinton diagrams suggest cells reacting on capital letters
- “parantheses-effect“ could not be reproduced
- generated C-code never showed perfect syntax, but was sometimes close
- perfect comment-syntax (but words were random letter combinations)

E2: Neuron behaviour in a *continuous-state* char-level LSTM for LK

conclusion:

- method too complicated, in addition too long training times because batch-size was tight to 1
- overgeneration of comments, natural language like variable names: comments in training data should have been removed, model had to learn two languages
- plots indicate correlations, but computation for single but consistently reoccuring events is difficult / impossible (to be shown later)
- correlation with indentation / spaces is basically an effect of character repitition

E3: KJ: Do char-level sequence generators learn words?

motivation:

- addressing mainly research questions 1, 3 and 4
- when learning character sequences of KJ, does the network learn reoccurring sequences (atomic parts)

procedure:

- train a char-level model (3-layered LSTM) on KJ
- a very common word in KJ is "LORD"
- feed sequences containing LORD through the network
- find the cell of the output layer that assigns a high prediction value for $P(\text{“}O\text{”} \mid \text{“}L\text{”})$, $P(\text{“}R\text{”} \mid \text{“}O\text{”})$ and $P(\text{“}D\text{”} \mid \text{“}R\text{”})$
- from there try to find high $w_i h_i \ \forall i \in \{1..H\}$
- correlate $w_i \cdot h_{seq}$ of the neuron with a marking for the current letter of observation

E3: KJ: Do char-level sequence generators learn words?

results:

- no correlation found
- we observed high probabilities for “O”, “R” and “D” after reading an “L”, which is not a surprise on KJ
- even normalizing by the readout-output did not change the picture

conclusion:

- problem of this method: we are again assuming that always the same neuron is responsible for specific network behaviour
- we could not completely solve the normalization problem
- actually `pearsonr` is inappropriate for 0-1-marking vectors
- using `spearman`-correlation instead also showed low values

E4: Correlating activations of a char-level sequence generator for KJ

motivation:

- this experiment was a first computational approach to the ideas of E1 and E2 (exploring the network)
- we aimed at generalizing sequence-specific findings generated with visual techniques (exploring a method) by using a computational approach

procedure:

- to achieve this, we marked particular properties of KJ character sequences and computed their activations
- then we computed correlations of the marking vector and the activations of a 3-layered LSTM for each sequence and average over all sequences

E4: Correlating activations of a char-level sequence generator for KJ

results:

- correlations of almost `1.0` for sequence-length marking (cf. E5, E6) for higher layers
- higher correlations of around `0.6` for word bounderies (also higher layers)

conclusion:

- we could confirm Karpathy et al.'s findings on sequence length
- maybe the assumption of *consistent* behaviour of *single* neurons over all sequences does not hold, this would explain low averages (cf. E5, E6)

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

motivation:

- stepping from character level to word level
- starting point: correlations between activations of an LSTM for formal language phenomena (hierarchy markers BRACE and BRACE)
- what kind of information about natural language is learnt by a neural network? (purely sequential vs. hierarchichal information)
- which network type (simple recurrent, GRU, LSTM) performs best
- this experiment is addressing all presented research questions

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

procedure:

- training data: HDT part A – C
- test data: a 7000-sentences subset of HDT part A
- train several neural networks with different architectures, i. e. all three mentioned types with dimensions (512, 512)
- to get a first idea which model is “most qualified“ compute perplexities of language models
- look at sentences containing dependency relation DET including a *distractor noun* between the determiner and the real head-noun
- mark the sequence from determiner to distractor noun on the one hand and the sequence from determiner to head noun on the other hand
- correlate the activations with both markings for each sentence and average over all

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

results:

simple recurrent	GRU	LSTM	trigram (Laplace)
7500	3300	2500	26000

Table 1: Approximate perplexity values of networks

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

some examples:

- *der die diz über die säule sind dennoch vage nach eigenen angaben von office auf 351 umzusteigen*
- *dort in peking und athlon-mainboards*
- *aus für arbeit*

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

- low average correlations, but for each sentence stronger correlations up to `0.8`

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

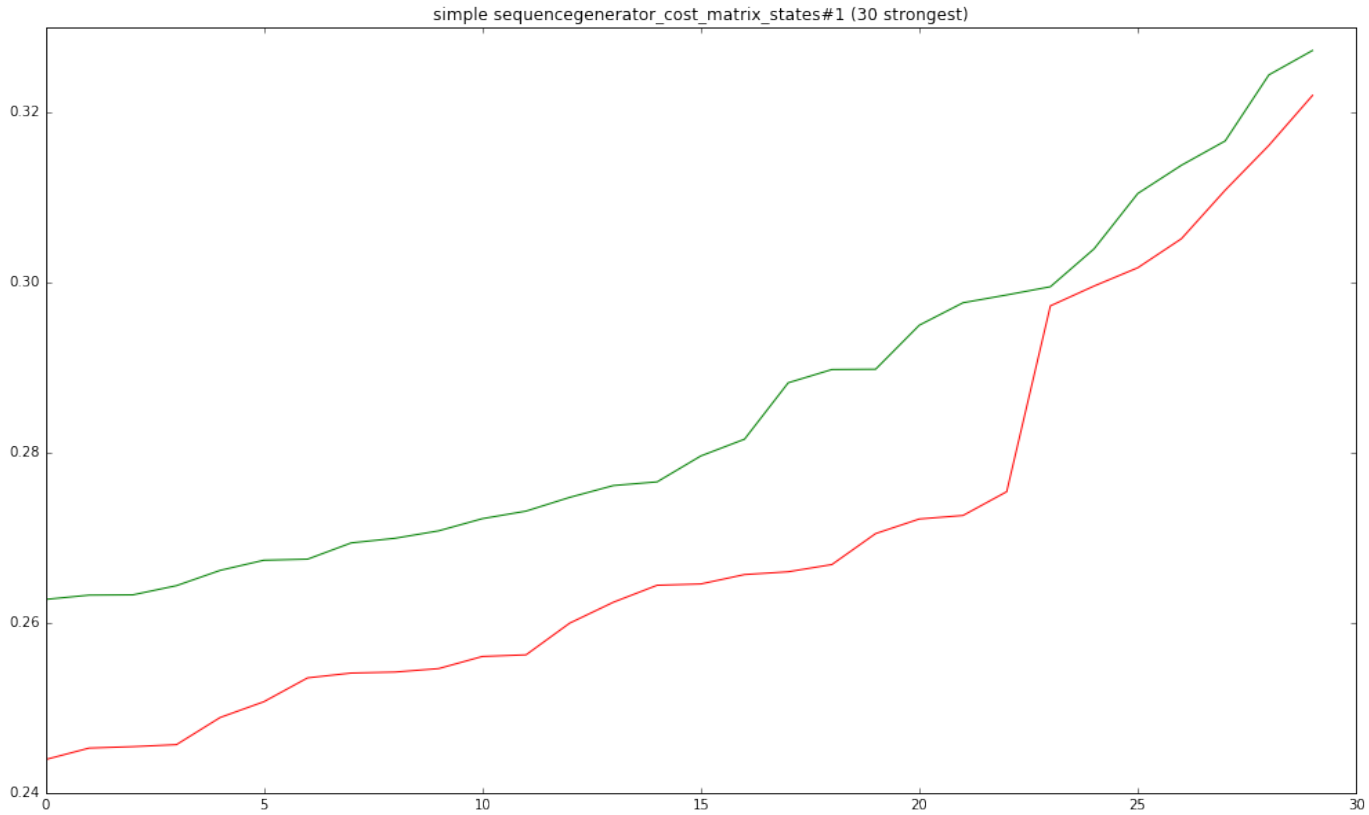

Figure 1: simple recurrent, last hidden layer output (top 30 correlations)

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

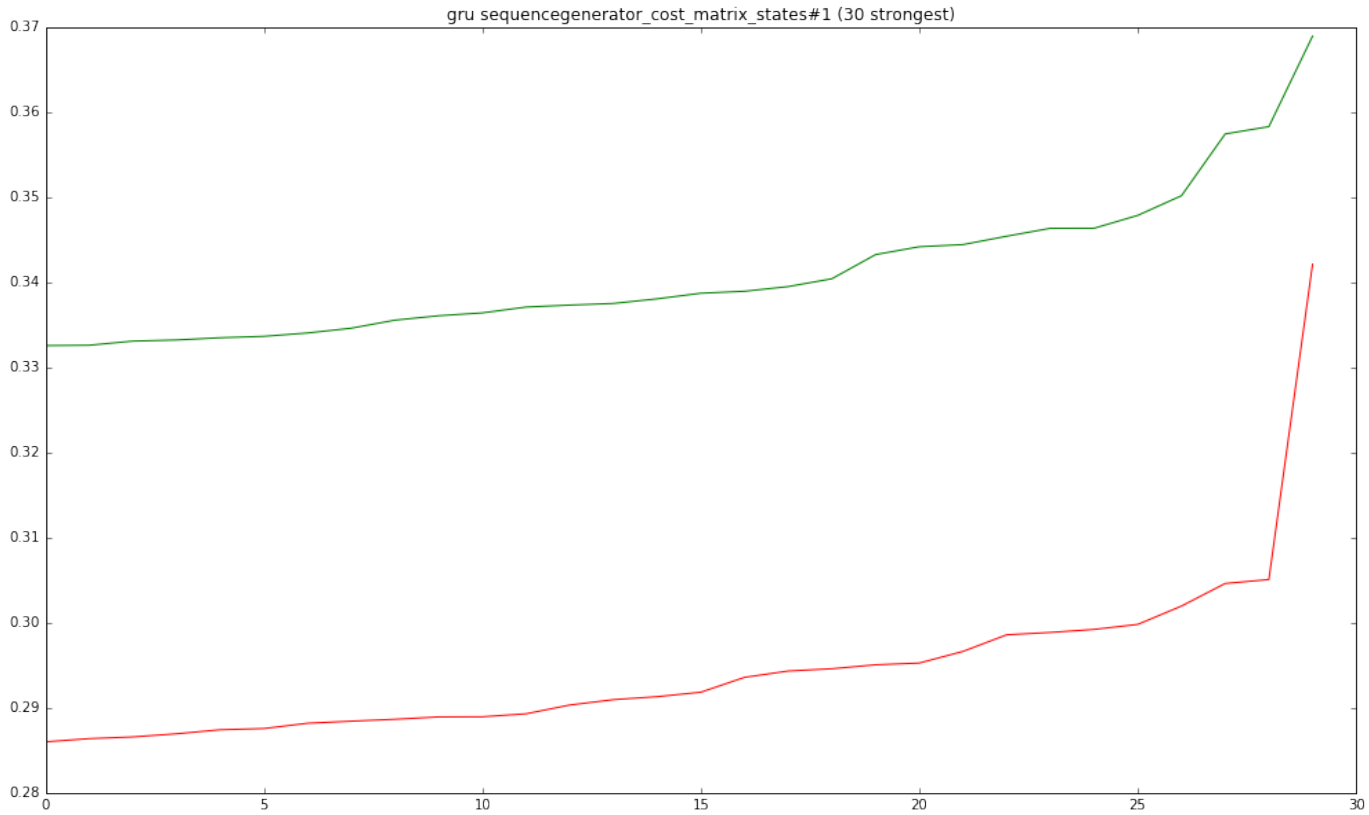

Figure 2: gru, last hidden layer output (top 30 correlations)

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

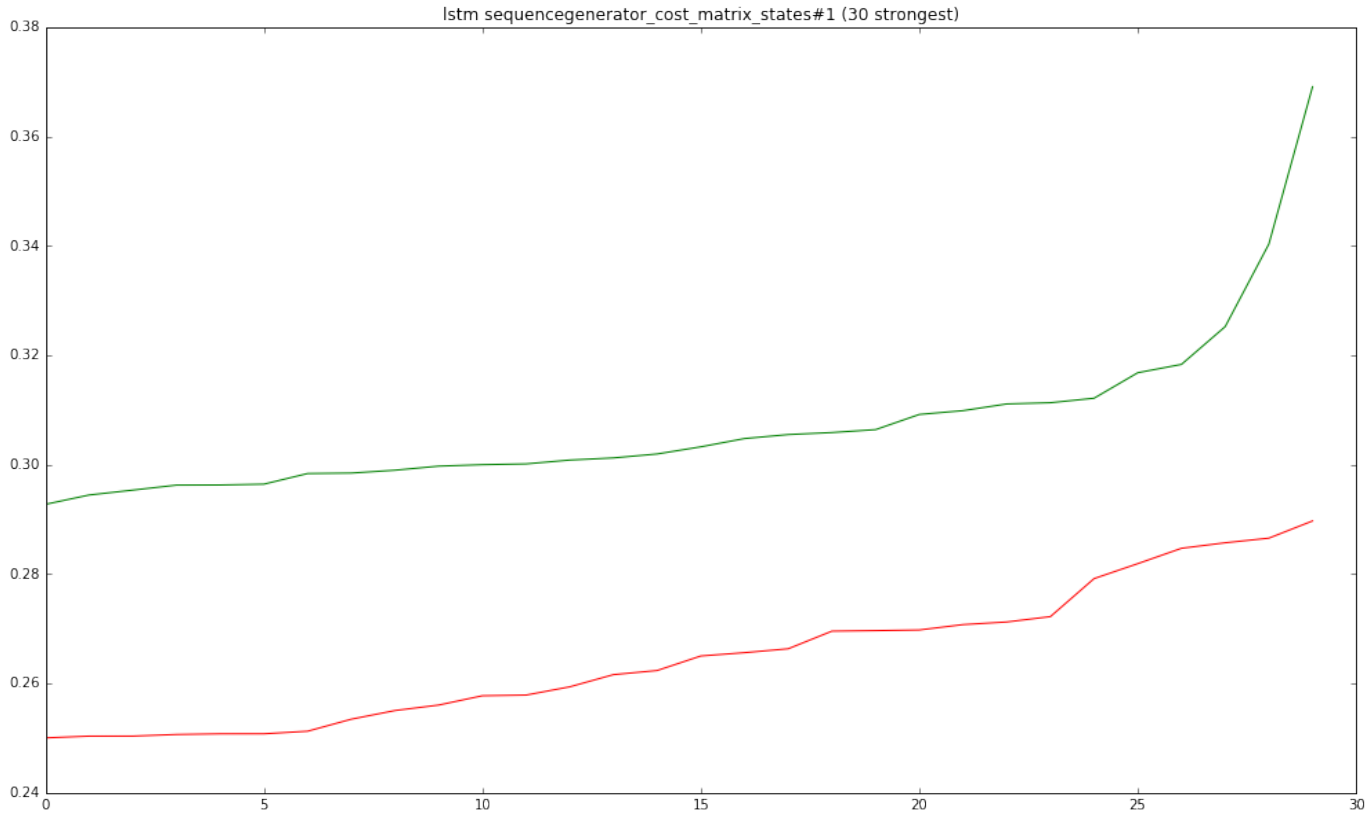

Figure 3: lstm, last hidden layer states output (top 30 correlations)

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

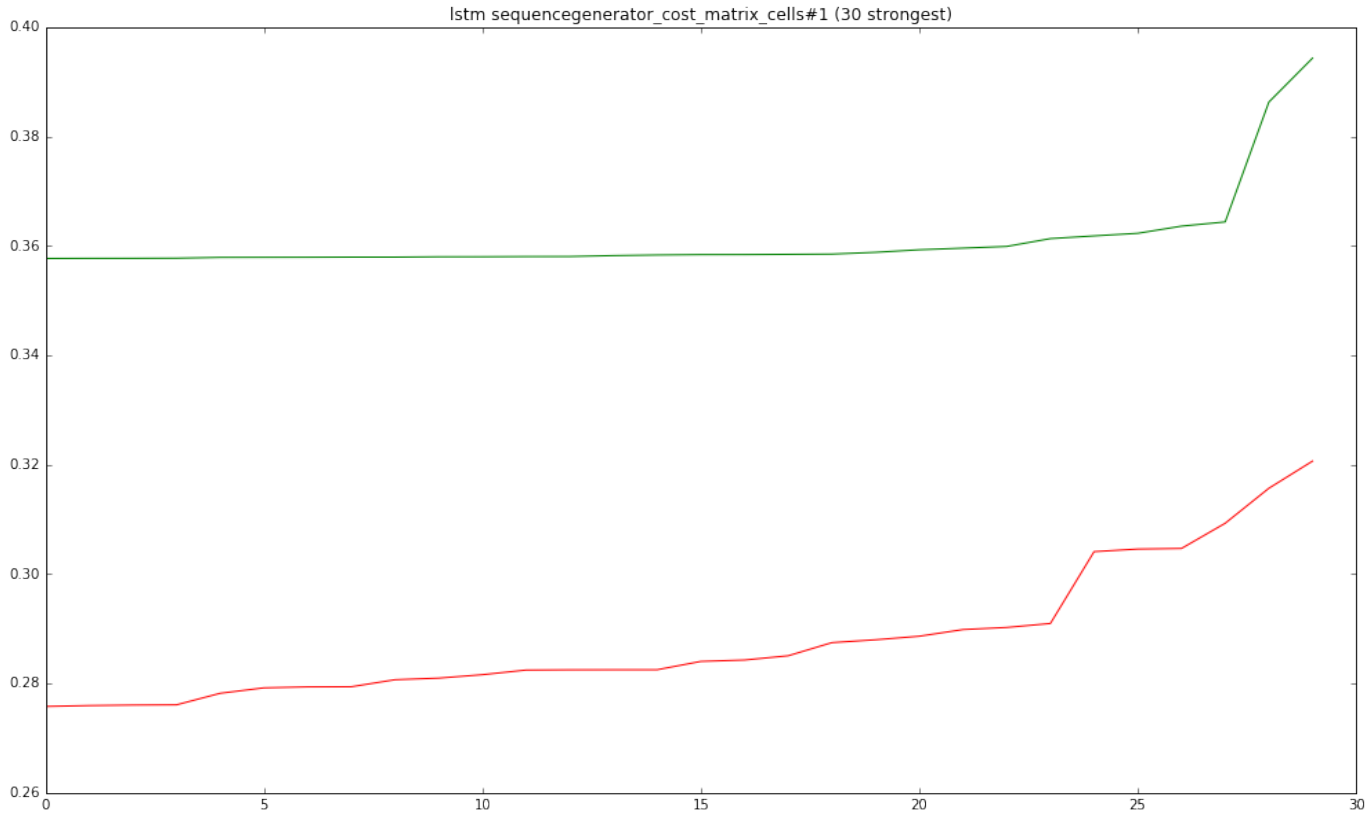

Figure 4: lstm, last hidden layer cells output (top 30 correlations)

E5: Correlating dependency relations and pos tags with activations for HDT(word-level)

conclusion:

- no consistent behaviour of a single neuron over all sequences, but strong correlations of single neurons over a subset of the tested sentences
- do neurons *cooperate* or *share tasks*
- very unclear if these results truly represent a correlation with our phenomenon of choice

E6: HDT – Clustering correlations

motivation:

- we observed problems for computed correlations for both character and word level
- how can we still obtain the neurons showing *consistency* in their behaviour (even if not over all sequences)
- we could plot all sentence activations and count, but well ...
- how can we figure out groups of neurons, that “work together“

E6: HDT – Clustering correlations

procedure:

- take correlation values from E5 and find k strongest neurons for each sentence ($k = 100$ out of 512 per layer $\times$ (cells, states))
- memorize these neurons for each layer
- cluster the sentences (which sentences have the most in common)
- plot some sentences to get an idea what these correlation values are about

E6: HDT – Clustering correlations

results:

- the intersection of top k neurons over all sentences was **empty**
- but ...

E6: HDT – Clustering correlations

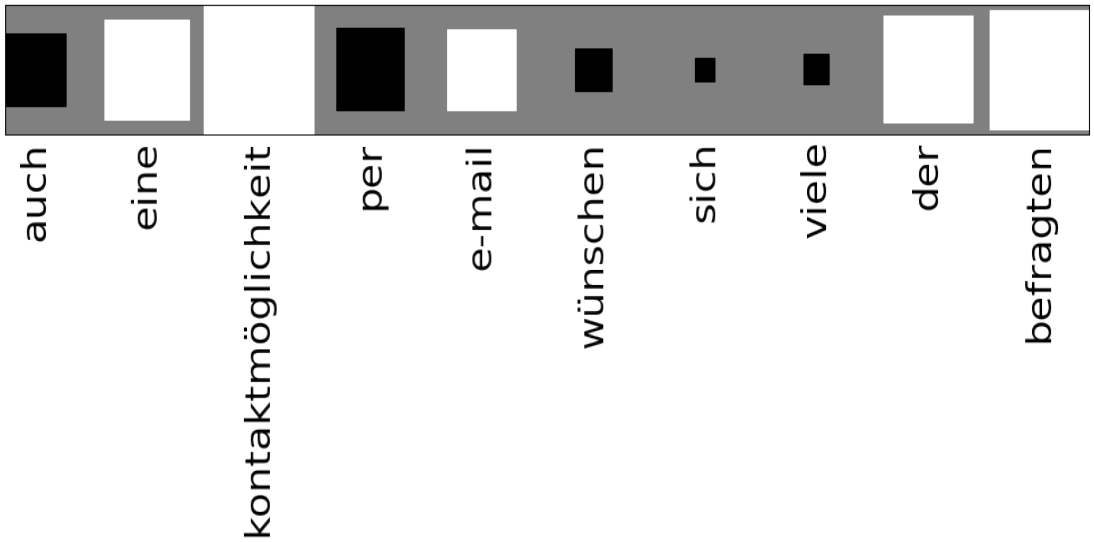

Figure 5: LSTM, first hidden layer cells output (neuron 36)

E6: HDT – Clustering correlations

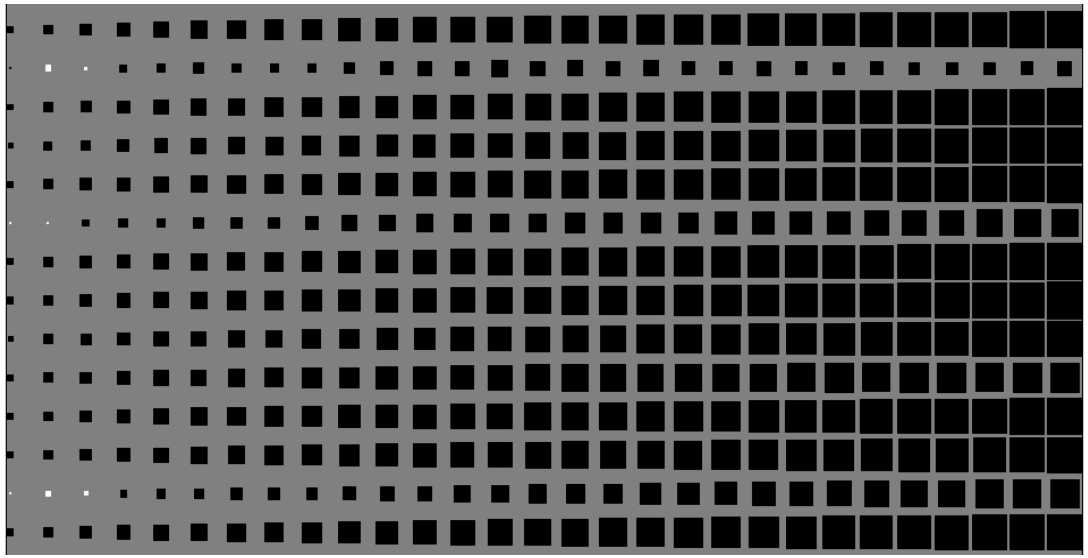

Figure 6: LSTM, last hidden layer cells output (observed way more often)

E6: HDT – Clustering correlations

conclusion:

- a combined method of plotting and correlating only provided a clear answer (so far), both methods on its own seem to be insufficient
- indeed the computed correlations seem to have been mostly about sentence length
- no neuron (in the inspected areas) showed strong correlations over all sentences
- neurons seem to act in clusters, but these are also not constant over time
- there are no hints or observations, that the network captured the hierarchical relation, but somehow distributional properties of nouns with their determiners

E7: KJ – PCA and correlations

motivation:

- as well as in the word-level experiment we need to explore a way to find out about neuron groups correlating with input-specific phenomena

procedure:

- we used PCA to find the neurons explaining between 90 and 99% of the activation variation in a 1-layered simple recurrent ($H = 1024$)
- after that, we correlated word bounderies and sequence length on the remaining components

E7: KJ – PCA and correlations

results:

- dimensionality was reduced on ≈ 500
- sequence length correlation decreased
- word boundery correlation decreased on `0.3` (`0.6` without PCA)

conclusion:

- PCA failed
- we were using a linear reduction method on clearly non-linear data
- this motivates E10

E9: ERP analysis on LK

motivation:

- activation sequences are continues signals
- why not using methods for signal analysis on it?

procedure:

- we decided to use an ERP signal analysis method
- general idea: compare averaged signals to noise to detect *real events*
- feed the network V artificial character sequences of length 2
- all sequences share our character of interest (COI) as first character, the second character is unique for each sequence
- the average of activations for these sequences is our COI's representation
- do the same for each other character of the vocabulary, but average over all gathered activations (noise representation)
- this experiment was performed for LSTM and simple recurrent on the LK-dataset

results:

E10: Facing the linearity problem with autoencoders

motivation:

- for non-linear data like ours (activations) with respect to our research questions, we need a non-linear method of analysis
- autoencoders seemded to be worth a try
- unfortunately this is work in progress without any results so far

procedure:

- training an autoencoder on our activation data of the HDT word model
- reduce dimensions of input down to 300 and back up again
- look at the inner representation of the input vectors and see what we can do with it

results:

- right now only a model for the cell outputs of a 2-layered (512, 512) LSTM exists (last layer)
- training parameters: input dim 512, 1 hidden layer, $H = 300$, batch-size 1000, trained with batch normalization

Review

- searching for the one neuron doing a particular thing is a dead end
- correlation computation needs to be done in a contrastive manner and in the best case be added with plots to get a better impression
- the research questions can only be answered insufficiently and incomplete
- we need methods capable to deal with non-linear data

References

Kilian Foth, Arne Köhn, Niels Beuck, and Wolfgang Menzel. Because size does matter: The hamburg dependency treebank. 2014.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.