Experiments

Dieuwke Hupkes

1 Datasets

I define the following set of languages:

Name	*Numeric leaves*	*Example*
L_2	2	$(x_1\ op\ x_1)$
L_3	3	$((x_1\ op\ x_2)\ op\ x_3)$
L_4	4	$((x_1\ op\ x_2)\ op\ (x_3\ op\ x_4))$
...		

Where $x_i \in \{-19, 19\}$, and $op \in \{+, -\}$. The meaning y of e sentences is the result of the arithmetic expression expressed by the languag. We restrict the languages to include only expressions such that $y \in \{-60, 60\}$.

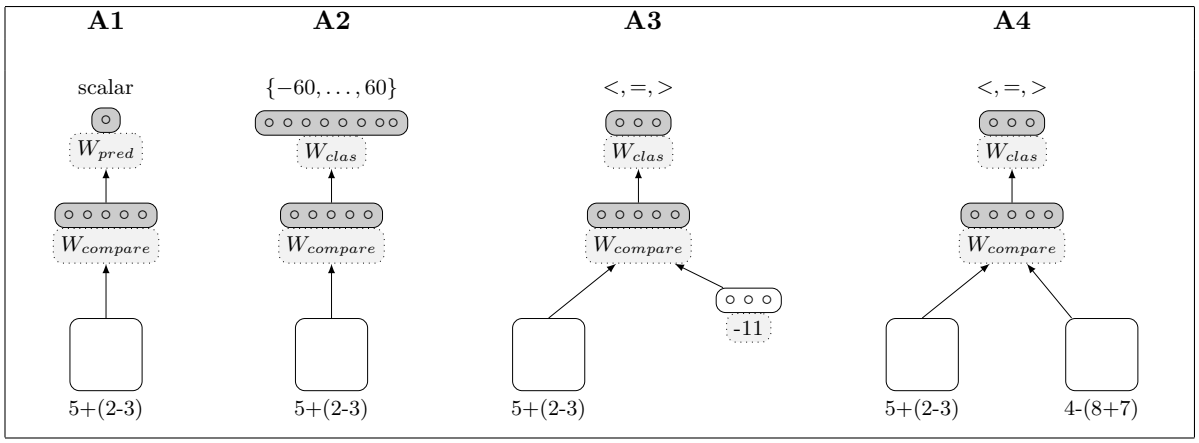
We define the following subsets of the languages defined above:

Name	*Restriction*	*Example*	
L_i+	$op == +$	$(.(.(x_1 + x_2) + \dots x_i)$	Structurally non ambiguous
L_i-	$op == -$	$(.(.(x_1 - x_2) - \dots x_i)$	
$L_i$right	only right branching trees	$(.(.(x_1\ op\ x_2)\ op\ x_3)\ op\ \dots x_i)$	Structurally non ambiguous
$L_i$left	only left branching trees	$(x_1\ op\ (x_2\ op\ (\dots\ op\ (x_{i-1}\ op\ x_i).).)$	Structurally non ambiguous

The datasets that the networks will be trained and tested on are (subsets of) unions of the languages described above.

2 Architectures

I use four different architectures (explanation?):



3 Experiments

I will start by running a sequence of experiments to determine if the networks can learn to compose the meaning of sentences from the structurally non ambiguous languages L_2, L_3+, L_3left and L_3right. Depending on the results I will move on to more complicated languages In principle, I would like to do all (possible) combinations that can be made by combining elements from the following table,[1] starting with architectures A1 and A2 and then expanding to A3 and A4.

Network	Language	Architecture	Dimensionality	Initialisation	Embeddings
SRN	L_2	A1	10	Random	fixed
GRU	L_3+	A2	6	Gray	trained
LSTM		A3	2	one-hot?	
	L_3right	A4			
	L_3left				

4 Results

General settings I am using for all simulations: hidden size = 20, size comparison layer = 10, batch size = 24, optimizer = adagrad.

4.1 L2

I ran a few run with Architecture 1, size_hidden = 20, size_compare=10, size_embeddings = 2 and language = L2. The trainingset contains 1800 sentences, validation set contains 200 sentences, batchsize during training was 24. During most runs the prediction error (= the sum squared differences between the true outcome and the rounded scalar prediction of the network) on the validation set stays high for a while and then rapidly decreases to a value close to 0. Examples of learned embeddings are depicted in 1 and 2.

4.2 Training L+ languages SRN

I did a couple runs with structurally unambiguous plus languages. Settings of the network were still the same: size_hidden = 20, size_compare=10, size_embeddings = 2. I trained for 1000 epochs.

The numbers are still more or less ordered, but not as strongely as before. The brackets and + do not get a meaningful embedding (they are both very close to 0), which is sensible because in the + languages they do not contribute to the meaning of the sentence. Examples of learned embeddings are shown in Figure 3 and Figure 4.

It seems the simple recurrent network does not really learn a very general solution for the addition operator: when the sentence length of the test items is shorter than the longest training item (even though the exact length was not in the trainingset) the network performs the task with a very low training error. When the sentence length is longer than the longest training item, the network does not learn to perform the task.

[1] Of course excluding non-sensical combinations, such as Gray encoding in two dimennions

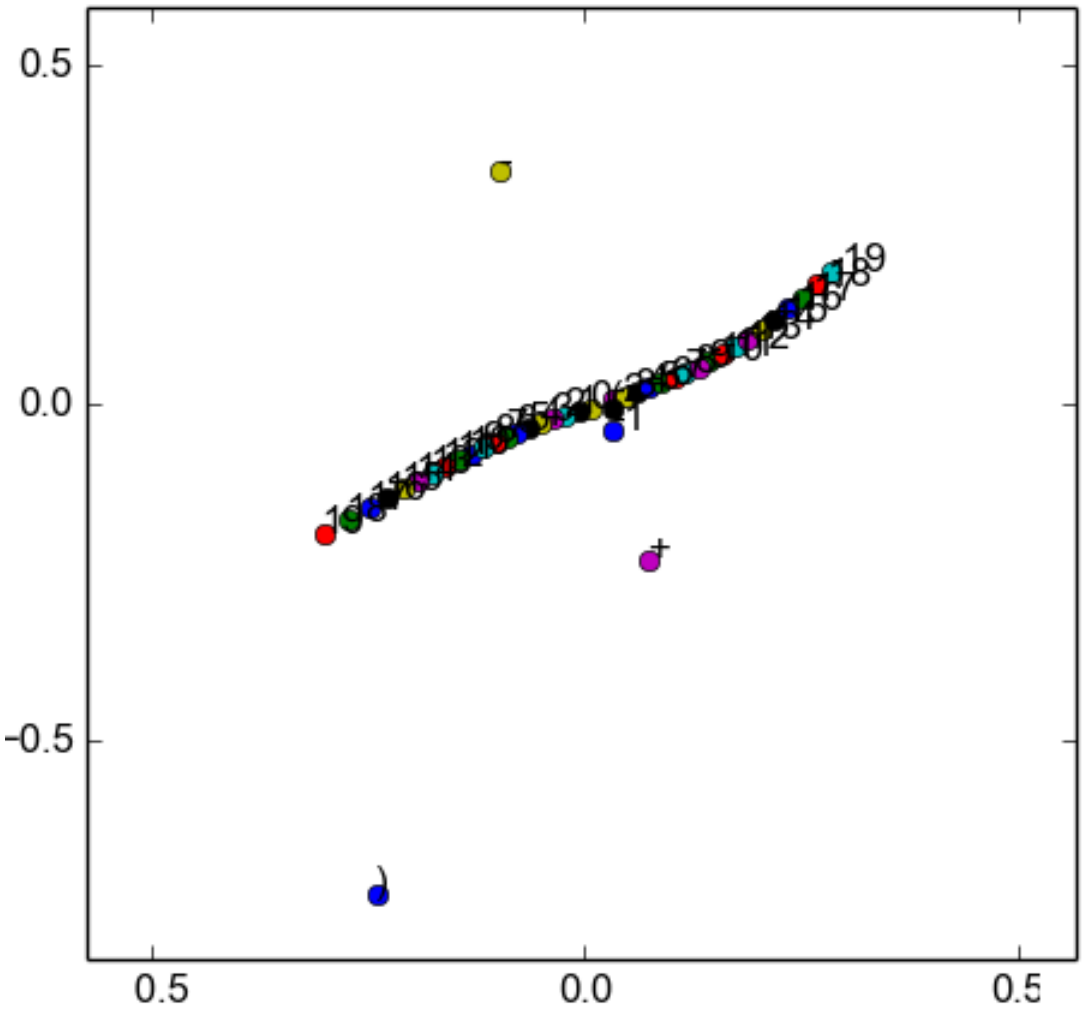


Figure 1: Learned embeddings for network trained on 1800 L2 sentences and tested on 200 different L2 sentences. After training, the mean squared prediction error on the 200 sentences in the validation set was 0.05

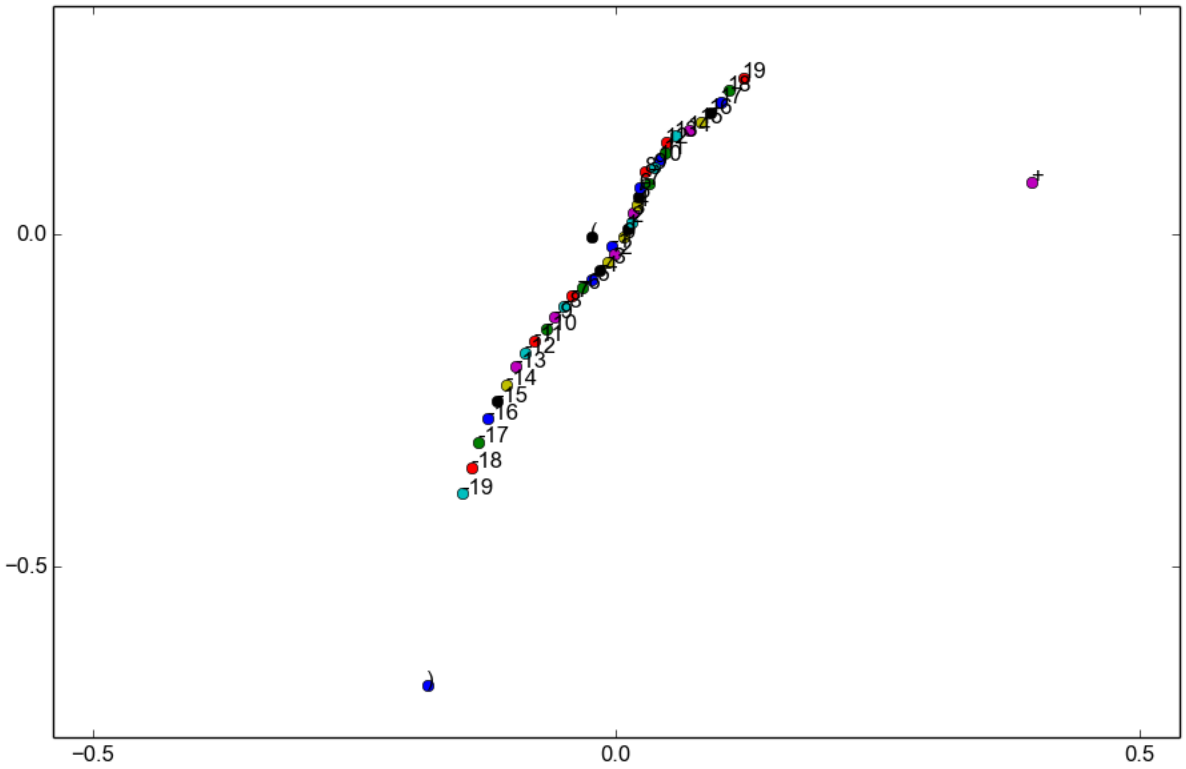


Figure 2: Learned embeddings for network trained on 1800 L2 sentences and tested on 200 different L2 sentences. After 1500 epochs, the mean squared prediction error on the 200 sentences in the validation set was 0.76

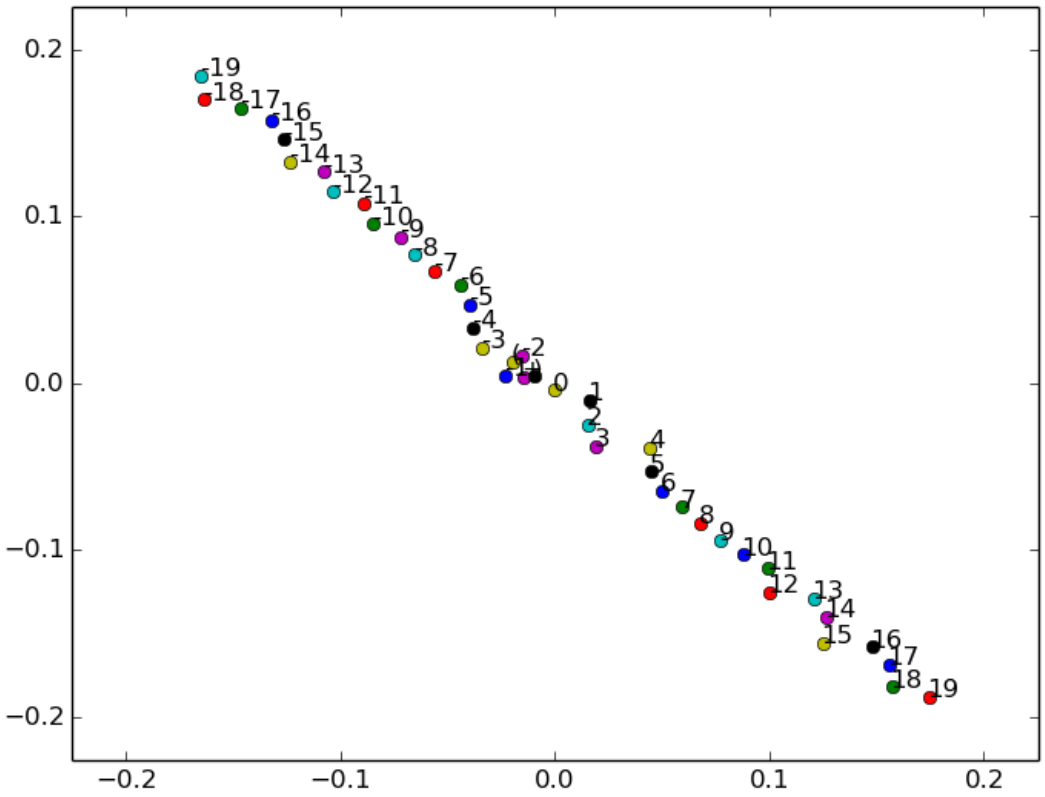


Figure 3: Learned embeddings for SRN trained on 2000 L2+ sentences, 2000 L3+ sentences and 2000 L6+ sentences, and tested on 500 L4 sentences. After 1000 epochs, the mean squared prediction error on the 200 sentences in the validation set was 0.056

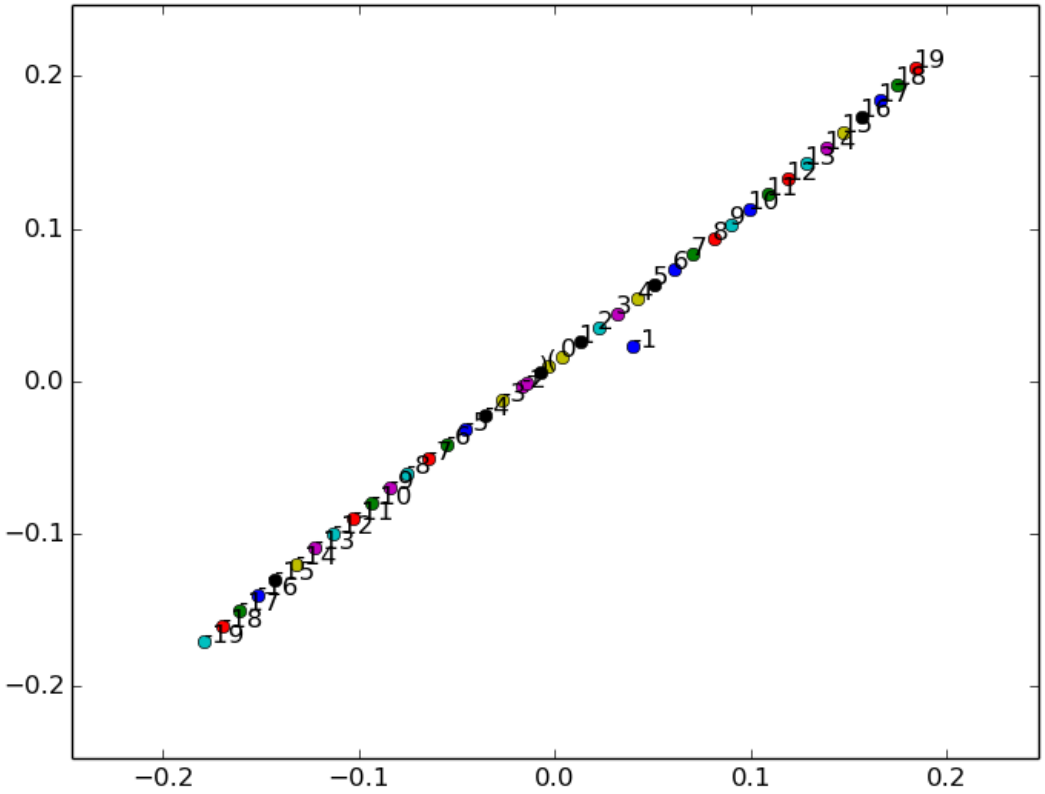


Figure 4: Learned embeddings for SRN trained on 2000 L2+ sentences, 2000 L3+ sentences and 2000 L4+ sentences, and tested on 500 L5+ sentences. After 1000 epochs, the mean squared prediction error on the 200 sentences in the validation set was 24.1 (the mspe of the trainingset was 0.006).

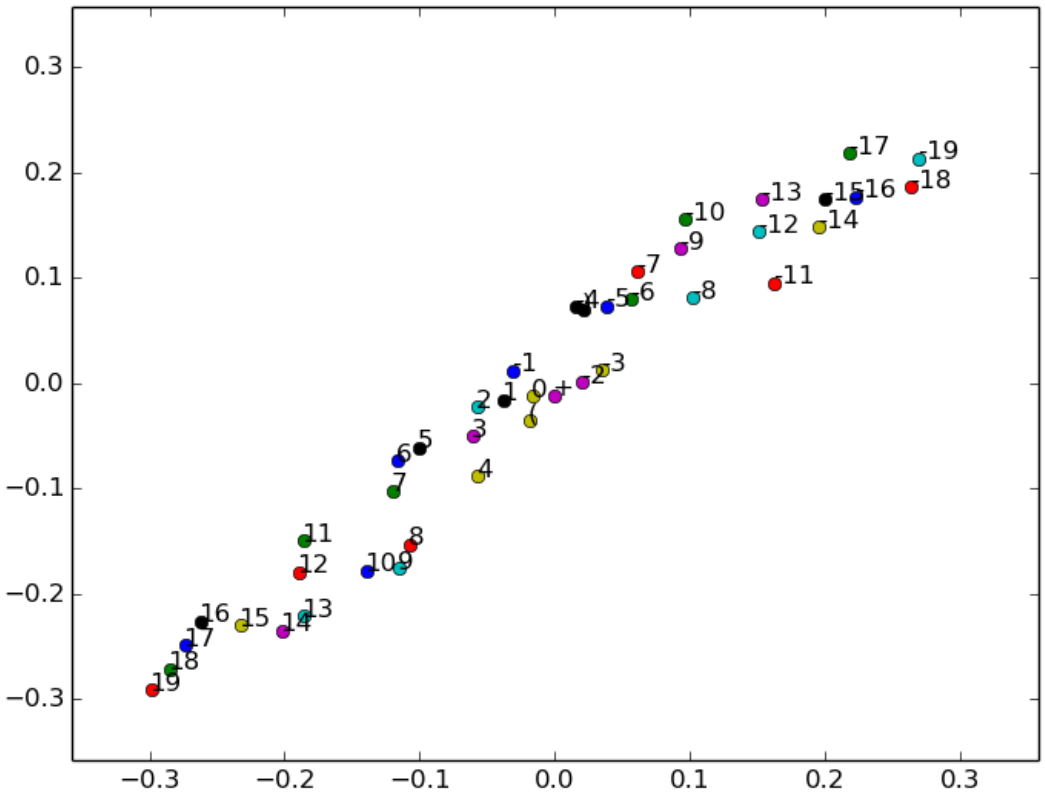


Figure 5: Learned embeddings for network with GRU layer trained on 2000 L2+ sentences, 2000 L3+ sentences and 2000 L4+ sentences, and tested on 500 L5+ sentences. After 1000 epochs, the mean squared prediction error on the 200 sentences in the validation set was 0.04

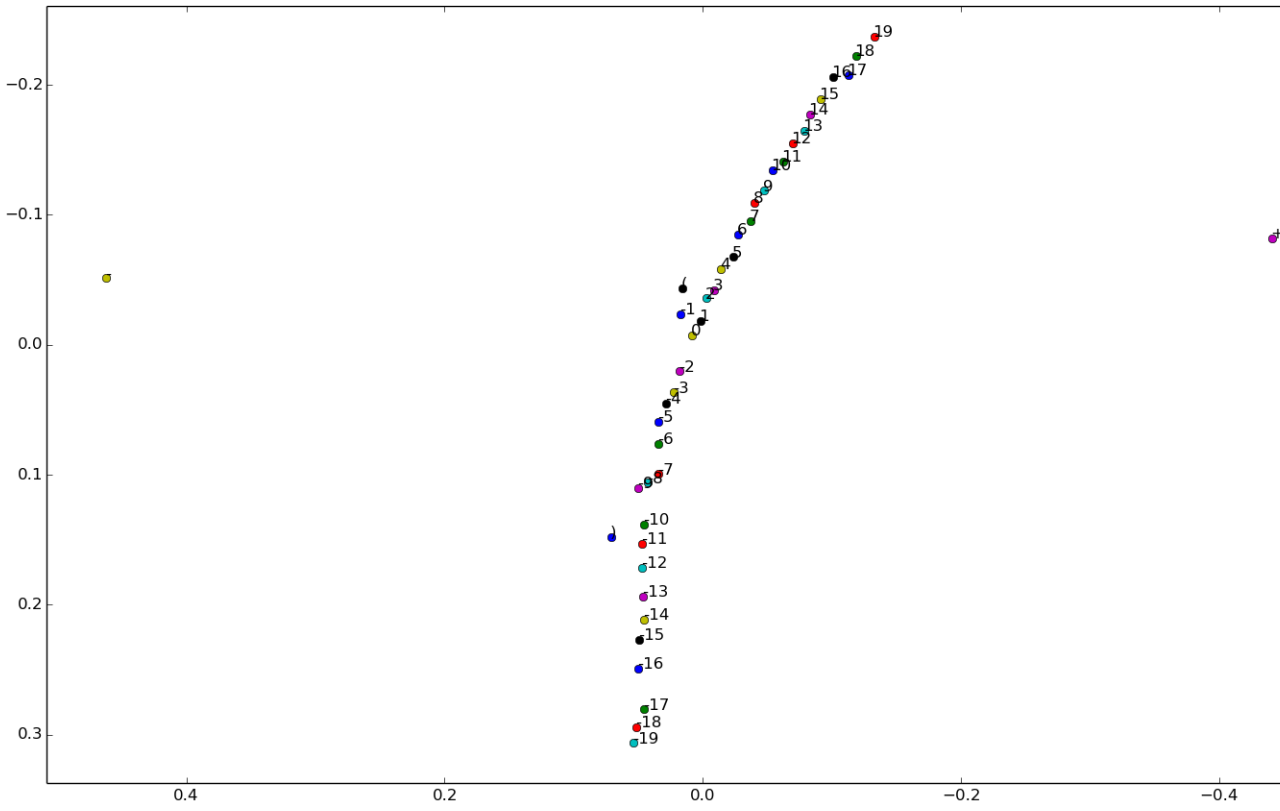


Figure 6: Simple recurrent layer getraind op L3left, training error was nog tamelijk hoog (rond de 5).