DL for NLP - Ass. 3 - report2

Itamar Trainin and Guy Gispan

January 2020

Language - 1

A language that should fail the LSTM is the **palindrome** language, that is sequences that have the same characters when reading from left to right and from right to left ('a', 'aba', 'abba', 'abcba', etc.).
We believe that an LSTM cannot distinguish this language from the language of all non-palindrome languages because in order to correctly classify, the LSTM must learn to remember all the characters that it have seen. Since the sequences can be of arbirary size, this is a very difficult task to achieve with one-directional LSTM.
In practice this language fails the LSTM from distinguishing the two languages (as can be seen in Figure 1: (a),(b)). We have used train size of 300 examples, test size of 100 examples and ran for 50 iterations.
It can be seen from the output graphs that the network failed on learning both the train and the test sets for this task. See Figure 1 for results.

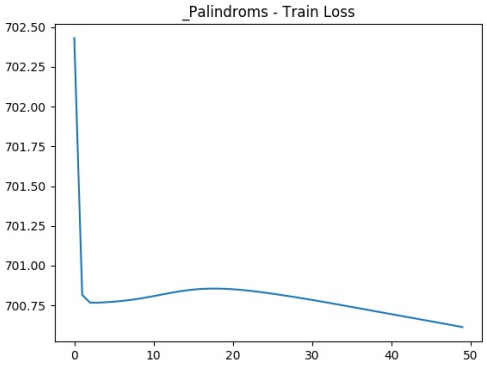
Language - 2

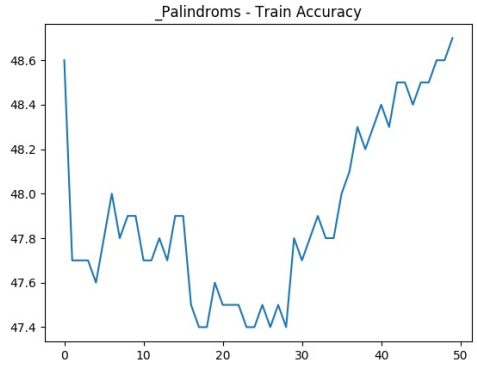
The second language that we propose is the language of **binary representations of prime numbers**. That is sequences over the binary alphabet that represent prime numbers.
We believe that an LSTM network cannot distinguish this language from the language of non-prime numbers because we don't believe that there is an arithmetic rule over a sequence of bits of arbitrary length in a prime number that can determine whether or not a number is prime or not (obviously, if there was it was already in use today).
We have tried passing this language through an LSTM network. Surprisingly, the network **WAS** able to distinguish between the languages.
We believe that the reason for this is that the LSTM learned from the last bit to distinguish odd numbers from even ones, simplifying the task by half. Then, since we have used sequences of max length of 15 bits at most, we believe that the network over-fitted to this case and therefore was able to correctly guess unseen test sequence, we strongly believe that this approach will not hold for arbitrary sized sequences, or at least will not outperform conventional methods

that utilize properties of prime numbers to distinguish them from non-prime ones.
One can see from the attached graphs (figure 2) that the network succeeded in the task both on the train and test sets. We have used train size of 300 examples, test size of 100 examples and ran for 50 iterations.

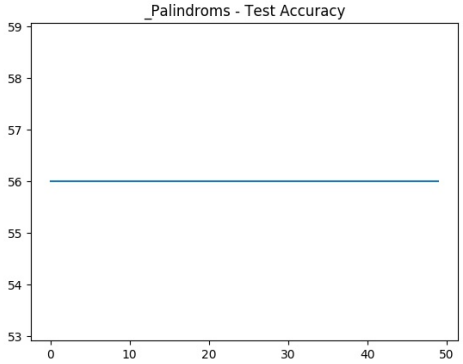
Language - 3

The last language we propose is the language where the first and the last characters are the same.
We believe that an LSTM network cannot learn to distinguish between sequences that start and end with the same symbol from other sequences because this requires the LSTM to remember the first character, however since the implementation of the network is recursive, we believe that the network cannot know which symbol is the first.
We have tested this language as well and we can see the the network **was not** able to distinguish between the different languages as can be seen in figure 3.
We have used train size of 300 examples, test size of 100 examples and ran for 50 iterations.



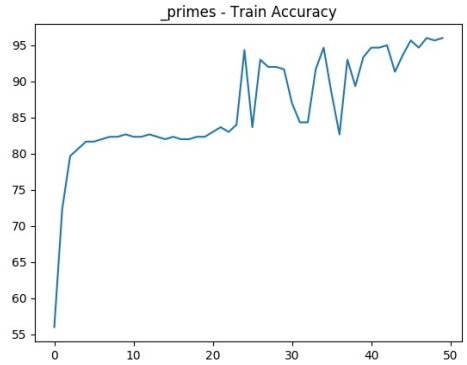


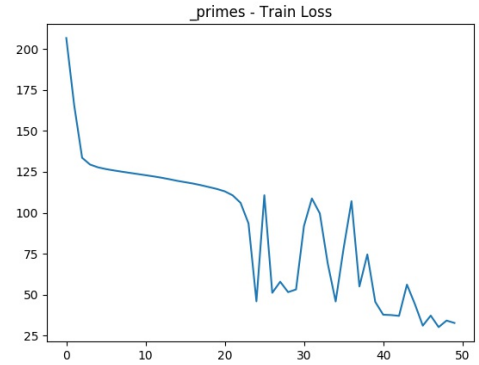
(a) Palindromes Train Accuracy and Loss



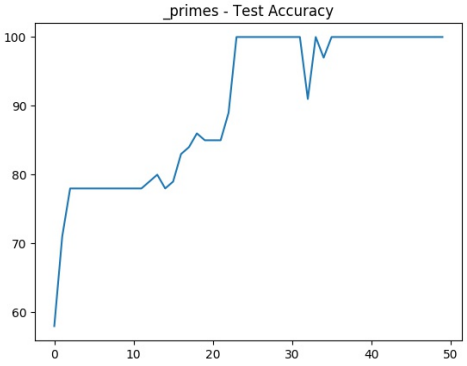
(b) Palindromes Test Accuracy and Loss

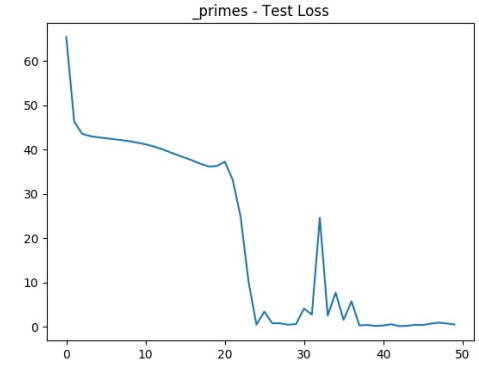
Figure 1





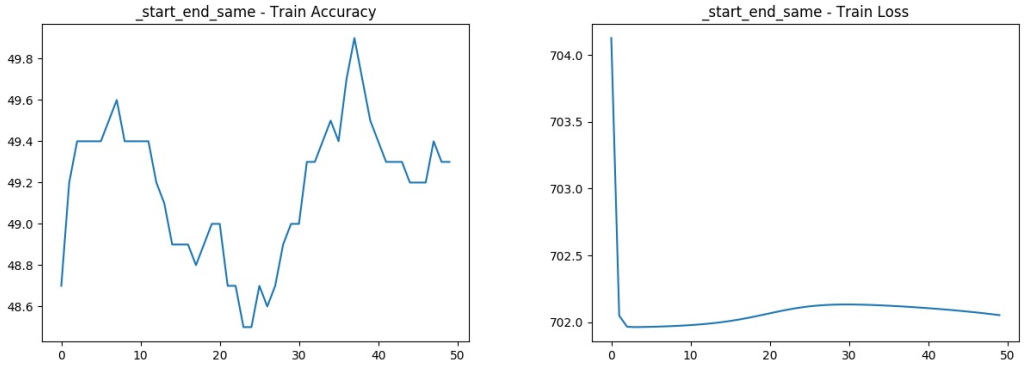
(a) Primes Train Accuracy and Loss



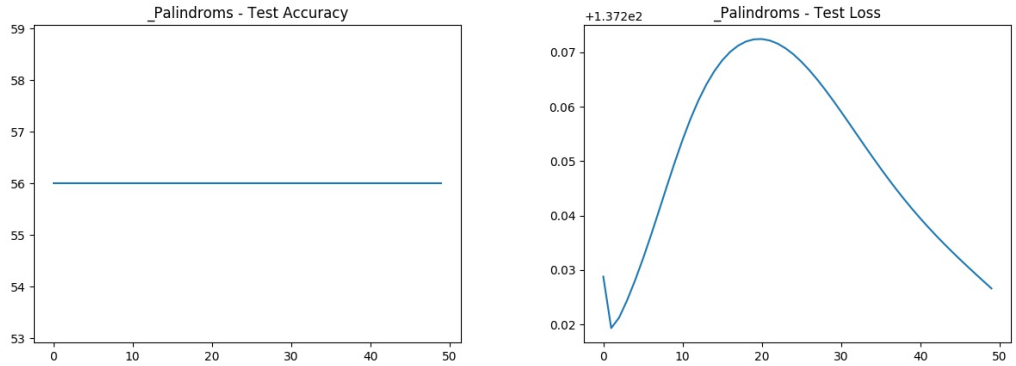


(b) Primes Test Accuracy and Loss

Figure 2



(a) Primes Train Accuracy and Loss



(b) Primes Test Accuracy and Loss

Figure 3