DL4Seq - Assignment 3

Otmazgin, Shon 305394975
Rassin, Royi 311334734

May 15th, 2021

1 Report 2

Q: A description of the languages
A:

1. Palindrome language - Alphabet: {0, 1}. A palindrome is a word, number, phrase, or other sequence of characters which reads the same backward as forward, such as madam or racecar.
2. Divisible by 3 - Alphabet: {1, 2, 3, 4, 5, 6, 7, 8, 9}, any number divisible by 3.
3. Prime Numbers language - Alphabet: {1, 2, 3, 4, 5, 6, 7, 8, 9}, any number which has just two factors: itself and 1.

Q: Why did you think the language will be hard to distinguish?
A:

1. Palindrome language - The model will need to understand that a positive example is one where every character is mirrored in its respective symmetrical location, and that even one deviation from that is considered negative. And we expect words like '010101011' to be especially difficult. The problem forces the model to 'look' arbitrarily far backwards, and the longer a word is, the lower the chances that the model will successfully apply context. In this case, the model needs to remember the first character it saw, and understand its meaning for the last character. Unsurprisingly, if we restrict the model to short words, it will perform well.
2. Divisible by 3 - A number is divisible by 3 if the sum of its digits are divisible by 3. For the LSTM to learn this problem, it needs to sum arbitrarily long inputs; it may successfully recognize short inputs, but when 50-long characters are introduced, the network is more than likely to lose context and forget what it saw in the beginning, and thus, not solve the problem.

3. Prime Numbers language - In essence, this is an even harder version of the 'divisible by 3' language; is x divisible by N (where N are all of the numbers between 1 and x). Further, prime numbers do not follow a pattern. However, the model can distinguish between even and odd numbers, so it may achieve at least a 50% success rate.

4. A final note: all of the languages above are not regular, and thus, LSTMs naturally have a problem to learn them.

Q: Did you manage to fail the LSTM acceptor? (including, train and test set sizes, how many iterations did you train for, did it manage to learn the train but did not generalize well to the test, or did it fail also on train?)

A:

1. Palindrome language - Train size: 2100 examples, 1060 palindromes and 1040 non-palindromes. Test size: 900 examples, 440 palindromes and 460 non-palindromes. 1050 iterations with batch size of 10 examples in each iteration. Both train and test reached 50-52% accuracy it always predicted the same label either 0 or 1. The train loss didn't go down after 10 iterations and remained noisy. The model failed to distinguish between palindromes and non-palindromes numbers. See figure 1-3 for more details.

2. Divisible by 3 - Train size: 3500 examples, 1731 examples divided by 3 and 1769 examples not divided by 3. Test size: 1500 examples, 769 examples divided by 3 and 731 examples not divided by 3. 1750 iterations with batch size of 10 examples in each iteration. Both train and test reached 55% accuracy. The train loss remained noisy and the model was not able to distinguish between positive and negative. See figures 4-6 for more details.

3. Prime Numbers language - Train size: 7190 examples, 3583 primes and 3607 non-primes. Test size: 3082 examples, 1553 primes and 1529 non-primes. 3595 iterations with batch size of 10 examples in each iteration. Both train and test reached 83% accuracy. The train loss did not go down after 20 iterations and it seems like the model reached its limit. It also failed to distinguish between primes and non-primes numbers. Furthermore, we also attempted a much larger data-set: 110K in train-set and 47K in test. We ended up with the exact same results. See figures 7-9 for more details.

Q: How did you generate your data?

A:

1. Palindrome language - A positive sample is created by generating a random sequence and picking its reverse (i.e. mirror) as the other half; we do acknowledge some samples are of odd-length, and do address that. A negative sample is just a random sequence over the alphabet. Also, the length of each sequence was randomly chosen; between 1 to 100.

2. Divisible by 3 and Prime Numbers language - Much like the previous language; we randomly generate numbers over the defined alphabet, and make sure there is a similar number of positive and negative samples. In this case too, a sequence is limited to a length of 100 and as with all cases, we make sure there are no duplicated samples. Specifically, for the divisible by 3 language we randomly concatenate a number, if it is in the dataset, or not divisible by 3 (while collecting positive samples), we continue the process, otherwise, we add it to the dataset.

2 Appendix

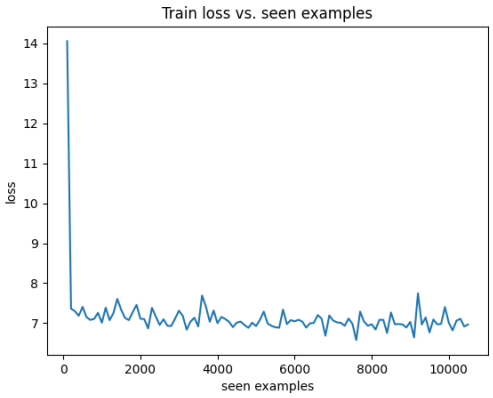

Figure 1: Palindromes Train loss

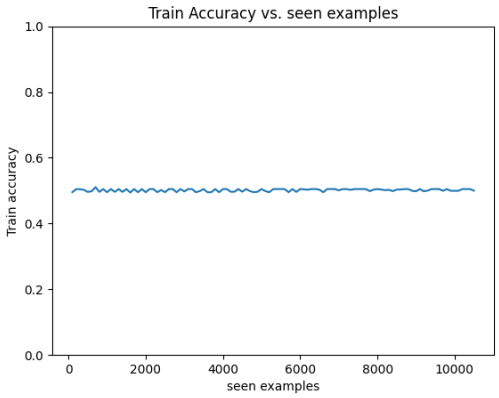

Figure 2: Palindromes Train accuracy

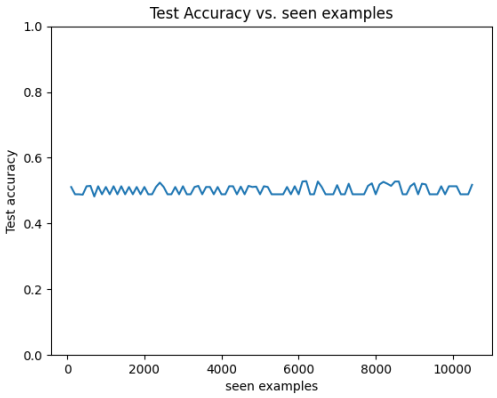

Figure 3: Palindromes test accuracy

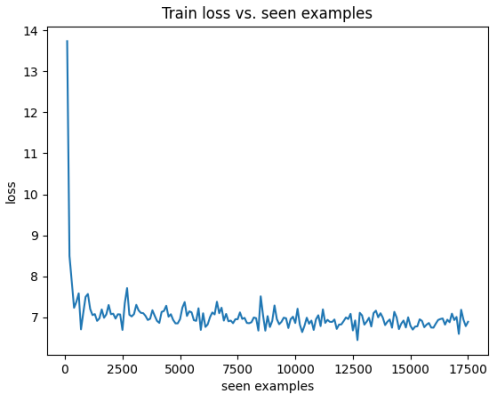

Figure 4: Divided by 3 train loss

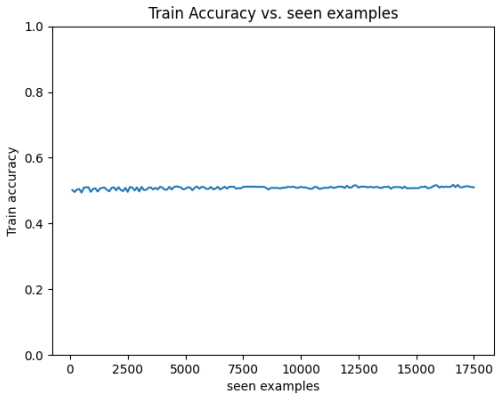

Figure 5: Divided by 3 train accuracy

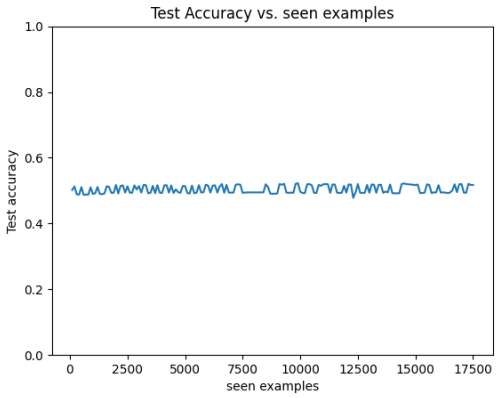

Figure 6: Divided by 3 test accuracy

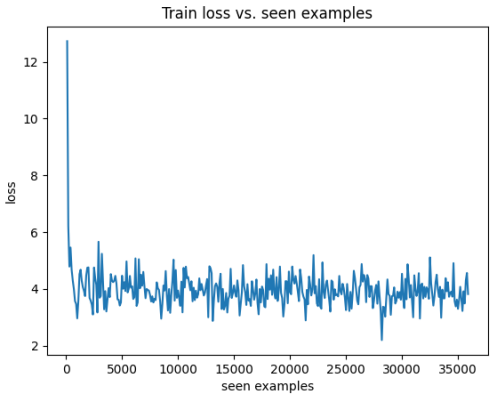

Figure 7: Primes train loss

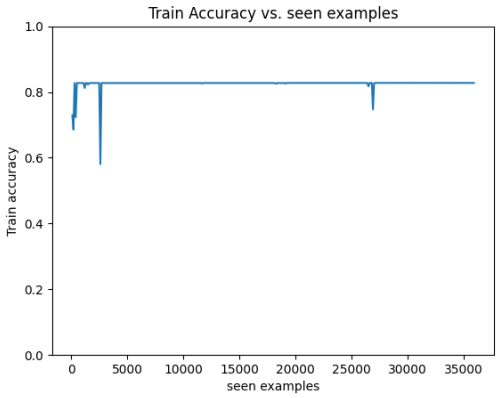

Figure 8: Primes train accuracy

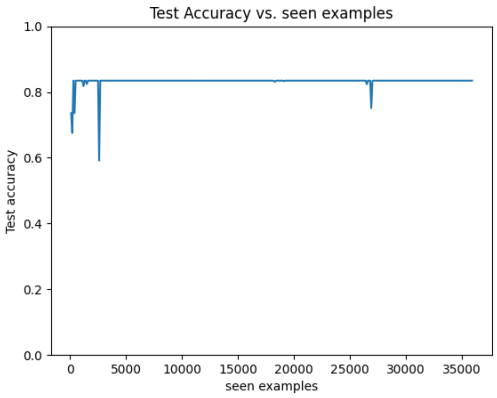

Figure 9: Primes test accuracy