CmpE 492 Final Report
Learning Sequences Using Recurrent Neural Networks

Mert Yaşin, 2012400066

June 01, 2016

Contents

Chapter 1

Introduction and Motivation

Neural networks are biologically inspired models of computation and are widely applied in various machine learning tasks. They have proved to be very powerful models that can be applied to almost any form of data, therefore their popularity increased dramatically in the recent years. Recent advances are also due to the fact that our computation power has grown drastically. Algorithms and methods proposed in the 1980s-1990s are being investigated to a greater extent.

Our problem of interest is, supervised learning of sequences, a problem that does not have a definite solution. While learning sequences, vanilla neural networks are not sufficient since there is no time dependant representation. There is an extended model called recurrent neural networks (RNNs), which propagate information through different time instances of the network.

Being able to model sequences is not an easy problem but solving it has many benefits. For example lexical analysis is a NP-hard problem. A single sentence of a language can have many different parse trees and it's generally not possible to select one without knowing the context. So there is not a deterministic solution. Plus, every language has its own rules so there is no global solution. To be able to ask a machine to parse a text passage and get accurate results, you have to model everything explicitly. Clearly, being able to generate arbitrary and meaningful text is spectacular and would have many benefits. This is where recurrent neural networks come to aid since any text is a long sequence of characters and sequences are the type of data RNN's are great at.

Also, text is not the only type of sequential data we can work with. Songs can be modeled as a sequence of musical notes, images can be modeled as a sequence of pixels, speech can be modeled as a sequence of vowel and consonant speech sound units. Any sequence is a different problem and the purpose of this research is to explore the widely applicable models and evaluate them.

Chapter 2

State of the Art

Current state of the art model in sequence learning is the Long Short-Term Memory (LSTM) architecture. It is an extension to the vanilla recurrent neural network such that the sigmoid gates at the hidden units are replaced with a much more complex structure, a LSTM memory cell. Similar to the RNN hidden units, these LSTM cells have the hidden unit outputs as recurrent edges. In addition to that, the cell state is an important information, therefore it is also propagated between different time instances of the cell and is a recurrent edge.

A LSTM cell contains block input, input gate, forget gate, cell state, output gate, and block output units. If we take a higher level look to these gates, block input and output units resemble the firing of a neuron. If an input node is active that cell gets activated and if an output node is active that cell generates pulses to be carried out to the next layer. Gates can be thought of water pipes with variable widths, they determine how much of the information is to be flowed or propagated. Input gate is accountable for the information coming from the previous recurrent layer, forget gate similarly takes care of the previous state of the current cell, and output gate filters the cell state and determines what to output. With its unique structure, LSTM cells are able to learn long range dependencies better than the RNNs can.

Below are the equations of a single LSTM cell:

$$
\begin{aligned}
z^t &= \tanh\left(W_z x^t + R_z h^{t-1} + b_z\right) && \text{block input} \\
i^t &= \sigma\left(W_i x^t + R_i h^{t-1} + b_i\right) && \text{input gate} \\
f^t &= \sigma\left(W_f x^t + R_f h^{t-1} + b_f\right) && \text{forget gate} \\
c^t &= i^t \odot z^t + f^t \odot c^{t-1} && \text{cell state} \\
o^t &= \sigma\left(W_o x^t + R_o h^{t-1} + b_o\right) && \text{output gate} \\
h^t &= o^t \odot \tanh\left(c^t\right) && \text{block output}
\end{aligned}
$$

where tanh is the hyperbolic tangent function and σ is the logistic sigmoid function. Here, x^t is the input layer of the hidden LSTM layer and h^t is the output. All z^t, i^t, f^t, o^t, c^t are the same size as the hidden vector h^t. W_j are the weight matrices connecting the input layer to the hidden layer and R_j are the recurrent weight matrices connecting the previous instance of the hidden layer to the current instance. b_j are the bias terms.

Due to the complicated internal structure of the LSTM cell, these architectures are able to learn long-term dependencies, as compared to simple RNNs. Nowadays LSTM networks are used for many tasks such as natural language translation, image captioning, handwriting recognition, unsupervised video encoding, predicting outputs of computer programs, song generation, and text generation.

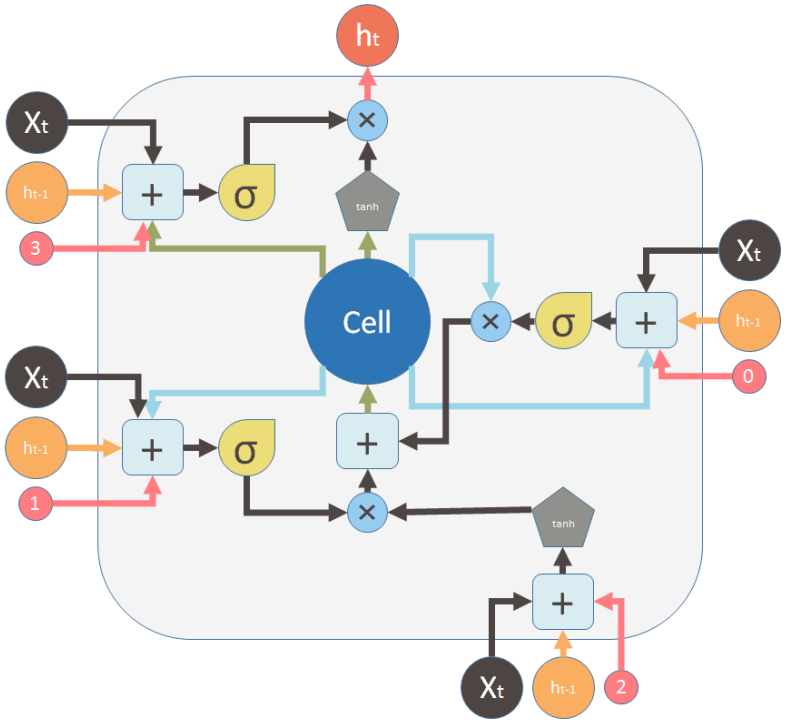

Figure 2.1: LSTM Cell

Chapter 3

Experiments

We have performed several experiments to survey the capabilities of the LSTM structure. Initially we have started from a vanilla recurrent neural network. Then, we applied the necessary modifications and upgraded it to a LSTM neural network. This was followed by the internal examination of the structure. We have divided our experiments into two subsections based on the type of the data: Synthetic and Real.

Synthetic Data

We wanted to create some individually examinable and interpretable examples. Whilst achieving that, we still wanted to preserve having a learnable sequence. Therefore we ended up creating a bunch of text files that are the repetition of a type of sequence for a large number of times. The test data sets and the types of sequences are as follows:

0101010101010101010101010101010101010101010101...	1 zero 1 one
0001000100010001000100010001000100010001000100001...	3 zeroes 1 one
0000010000010000010000010000010000010000010000001...	5 zeroes 1 one
0000000000100000000001000000000010000000000010000...	10 zeroes 1 one
0000000000000000000010000000000000000000001000000...	20 zeroes 1 one
0000011111000001111100000111110000011111000001111...	5 zeroes 5 ones
0000000000111111111100000000001111111111000000000...	10 zeroes 10 ones

We trained a single LSTM cell for each of these tasks multiple times and forced each of these training to over-fit. After training is complete, we sampled characters from these networks for a couple of cycles, where a cycle would be defined as generating the repeated sequence once. Then, as we sampled from the LSTM cell, we plotted each gate's activation values. Therefore we are able to observe and interpret patterns. Below are the plots.

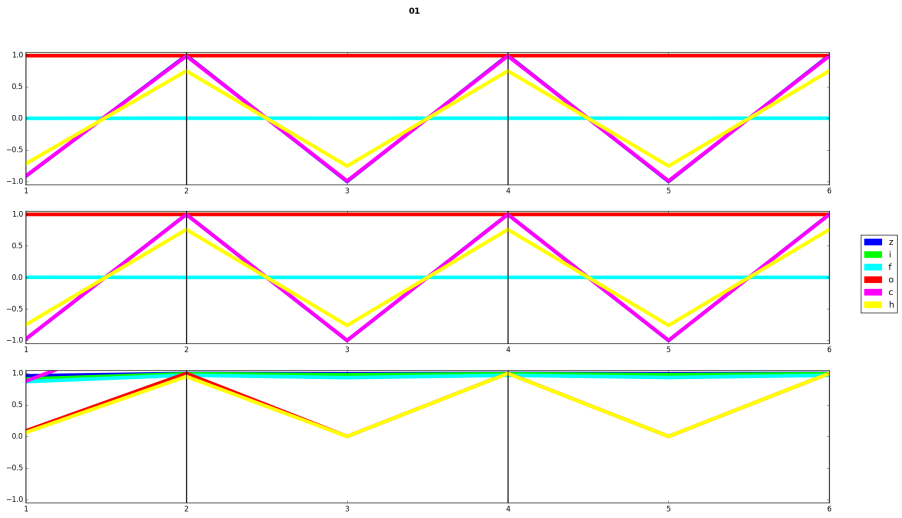

Figure 3.1: Sequence: 01

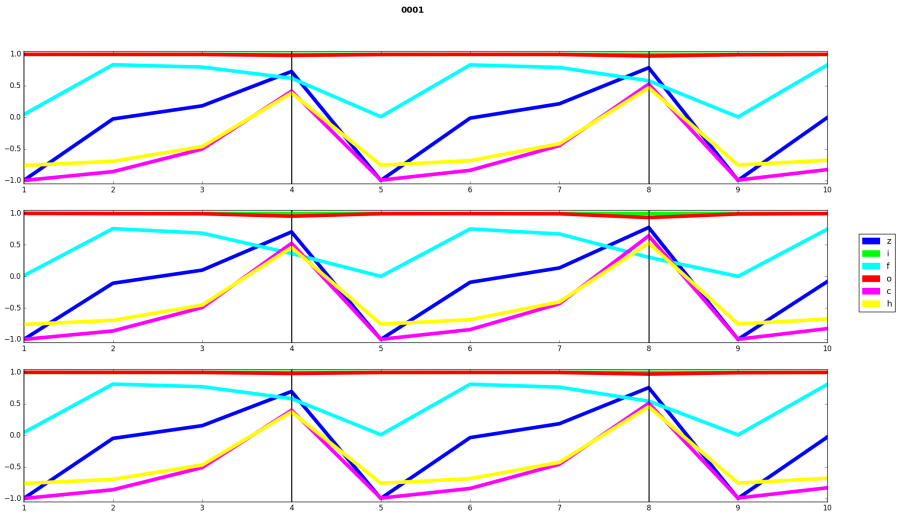

Figure 3.2: Sequence: 0001

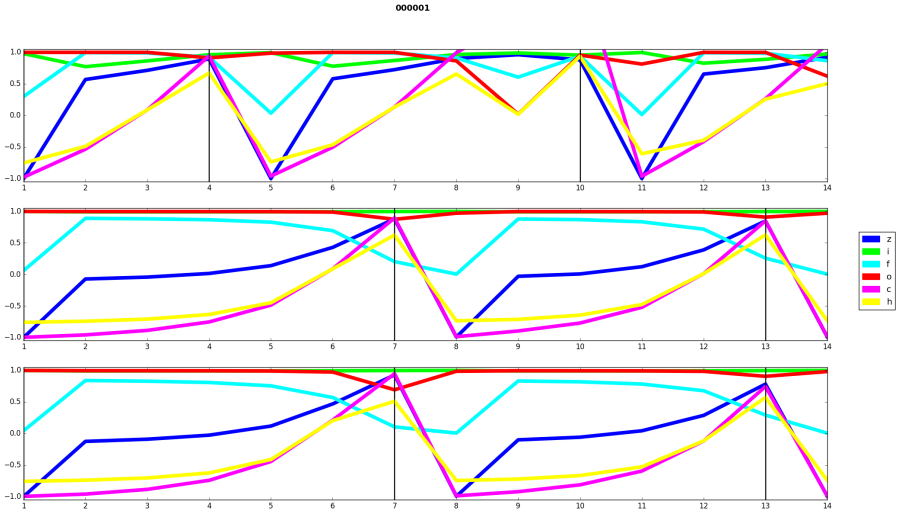

Figure 3.3: Sequence: 000001

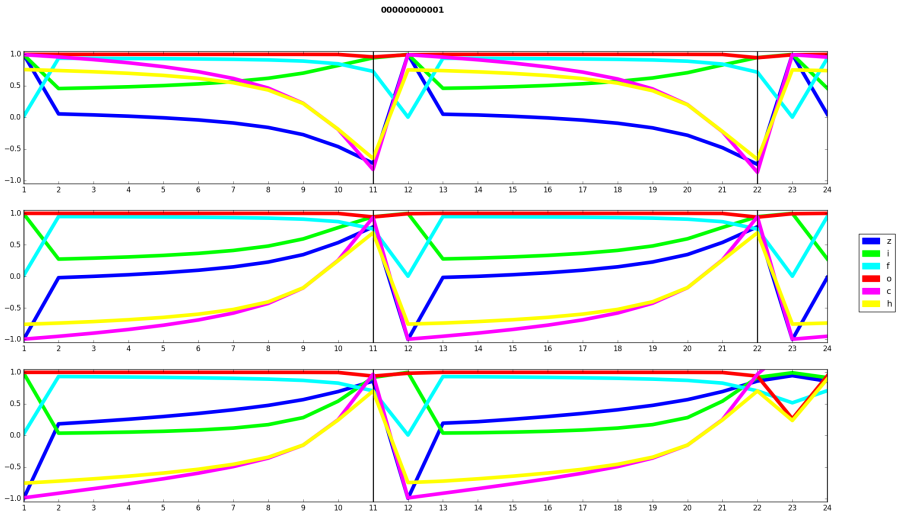

Figure 3.4: Sequence: 00000000001

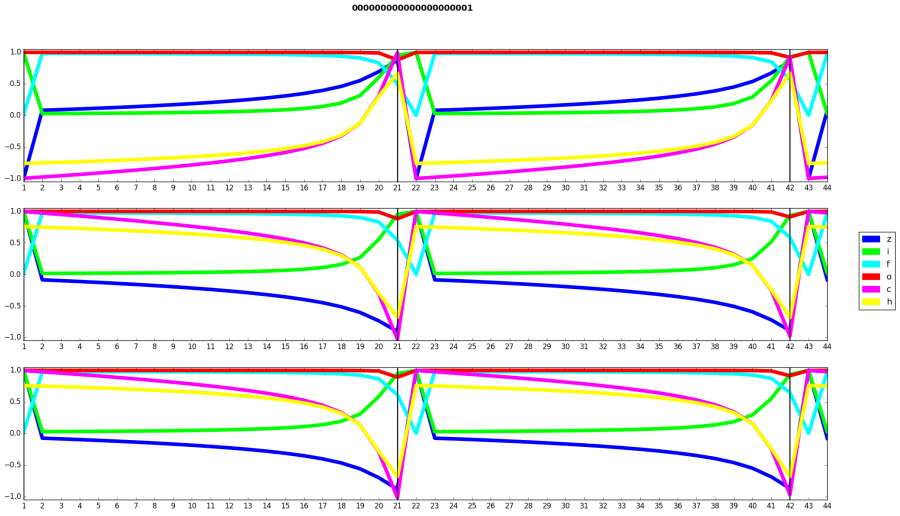

Figure 3.5: Sequence: 00000000000000000001

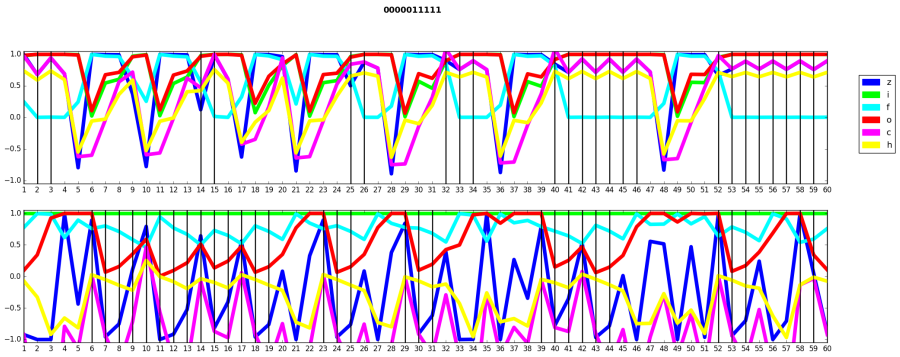

Figure 3.6: Sequence: 0000011111

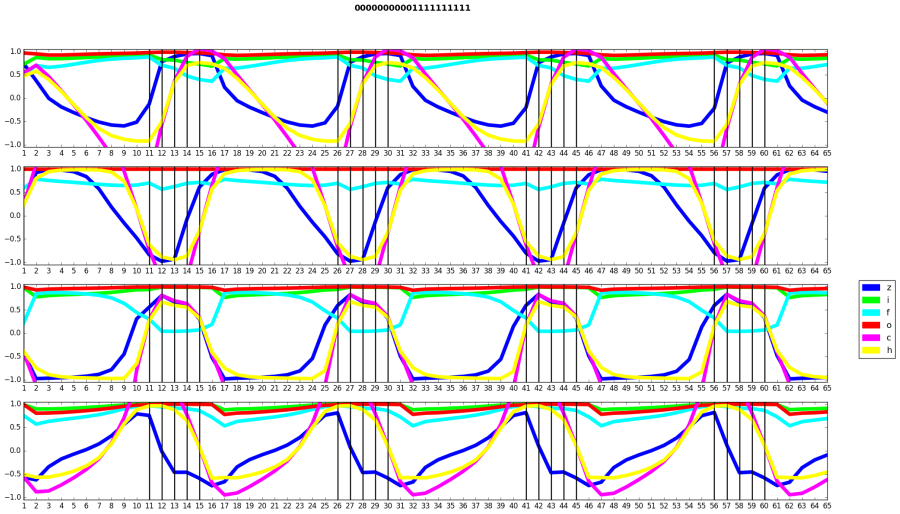

Figure 3.7: Sequence: 00000000001111111111

Our selection of the data sets have a purpose. By keeping a single one and

changing the number of zeros between each one, we are trying to measure the effect of the length of the sequence. If there had been a unique solution found by the LSTM cell for this task, we should be able to see that kind of relationship. As we analyze the results of these experiments, we see that the solutions differ from each other but with further analysis, we are able to observe some patterns. For instance, let's focus on the 'five zeroes single one' and 'ten zeroes single one' cases. Both of their second and third plots look alike. The only difference is the rate of convergence. If we analyze the weights, we see that by perturbating the weights a little, a single LSTM cell is able to capture the difference between a sequence of length five and ten. There is also a an interesting case we see in these examples. The first instances of training look similar to second and the third but they are upside down. This multiplication with -1 is still able to learn the same thing due to the multiplications in the equation of c (sell state). Then let's look at the examples with multiple zeroes and ones. As we see, a single LSTM cell was not capable of learning to count two different things. When we increase the number of cells in the hidden layer to two, the network is able to capture this multiple counting task. So we are able to get a sense of the underlying complexity.

Real Data

After experimenting with toy data-sets. We moved on to the real world applications. We gathered some large enough text data-sets that we can learn in a reasonable amount of time. Then we sampled from the trained networks and observed the results.

> GLOUCESTER:
> Your good love found dead sword, brokleat;
> And with poor bry, that the prosess better a
> were alive, 'tis like pardon.
>
> First Watchman:
> I cannot lose him. What liege to hear my breathing men,
> Is set their exchultic for her wounds wrongs.
> God garden, fexor me a heaven,
> Have him a man, one anwerged of patricians.
>
> QUEEN ELIZABETH:
> With for less shocks a death, after
> And did it stay my hand in petet by your brother,
> Your fiends thy tumbly at no trask made!
>
> CAPULET:
> Go to your heart for you, a Capiso. And, with us.

This short passage has been sampled from a network trained with the plays of the greatest writer in the English language, ***William Shakespeare***. Below are some examples of the previous iterations of this training.

n then hiul mf my
Ther strrilite be theas ecilod, Co for
Bndrs hate thet rades,

TeNyNIUale
To whouds'ks thercs Marr you thend,

Ceerannludr-A
IAndeFinds thrir sp of we
er it cisan;
They.ts athy hats an.

MAss grone:
To
romh us you tourus!

MIRCIUS:
And wiilk:
Lrev sburcucher shars,
:rund sound instserdou hive madidr trk harad.

MOMENIUS:
he gr onan blor of made that fean spome.

MIRoIUS:
Thes wivet, mathice
They Siting, enkeLnd wics Cimisg, hh Romurs' Carist.

SO
ThemceHnest
Thll wass chans stere

iteration: 1000, loss: 75.78, hit-ratio: 0.19

e gross kenes the propel'
And is highent, which virmy it be someers,:
Cith buncey shave as coming of has if forcise our'd in when
Ourall the we hell, think his scawn we lace
Of agodt he sidgends to will and corken,
And you had, my like home? Pe if ham ariok.
I have dith.
Ge had come poned if my
Romant you; this ragre ere, I to well
The con hichon's now I dad oy yourtert
enferss liy to we dove' canditire feacion,
At home noble? me„ be that selvecs they wath, Capes moter.

MENENIUS:
That were bus

iteration: 5000, loss: 42.87, hit-ratio: 0.49

S:
Which I be done,
Did beliving say my right?

HESMION:
O Hear taken,

LEONTES:
If a main, and cimen.

PAULINA:
I know what prince?

PAONIN:
What, in this, bear come?
-ingef; the villuing of hisserve,
I'll knee–
God of the commst fornours of my father,
Whose sorright for sir canst-benjece; and too;
Peace with her and my kawlands with a kestance her,
The fich not honours, I thought ever for your honour bat
That were Kamole and you, but to the pure.

PEINI GAN:
A honour be you you not bods!
Thes

iteration: 30000, loss: 38.29, hit-ratio: 0.53

did; it last, one
What you then ne spirit in death!

first Seelest Mariar:
He is offends comes thee, you!
go rain, I pray, madam; what a ware am Edwards;
And discover supplin,
Be fair open, make them see him.
I prosence of him ourselves on the true
visernly; I saw so unpethier it will;
But have much shall take us spitit as one proudenher grace for
not which with me are,
I have see my trumpeth their heeds:
And leave all, and not trams
What reary dead; and honesty since
Harsled in day they would

iteration: 275000, loss: 33.27, hit-ratio: 0.73

Now let's see some examples from a more familiar name.

Fakat çünkü
Öldü, bir dalda,
Kapıdan
Baktı ele göndereler ne :
“— Aradı şehrini.
Kapılarla yakarım!..
Çocuk, soğukta ne şey?..
–
Kulaklım olduğumu sanmak,
karanlıkta ikinin akşama...
Bir gece aracaktı
Yok!
Sonun,
Yoklaşını
ölü vardır,
seslerin dökülüp etileren
kadar kırın,
sesini.
azımı anlağı gece kalbi geldi.
Geçmiş var :
kızların kapılarındayım.”
Durdum...
Bu genç gözlerinden
ve perçeli,
haber gözlerin içini
değil,
Çine, yazmıştı.

Above is a sampled poem from a network that has been trained with the poems of Turkish literary figure, ***Nazım Hikmet***. Below are some examples of the previous iterations of this training.

vmaptanin Rtikilindiz.
ve ince vedti dik şörisibiş şapli. BTldini bayla hafmasa yek saşlerizer ecriz senremtiye nevi
Peveliğin Sâvzâler akır
Denneli gü
renen
Sii irumsüddindeteleri sadsal Şaayan sahti tasmanda Bamar Arman girehesen mışara desuzme pá
oyfma Sçılı bir tünapde Yİrmite varafdiy! Ariâdi misohap ıkpuşumin favtsk iyasın
celik razete'ri ne şeziy.d madarındarin kürdü düsma.k. Teydandns.....
Fah
gen şalıklol utere tuşâsılâdVe..
oroyösripimi imdenle Sır Alrsn
odâmen besim
öv..me
oğtuy

iteration: 1000, loss: 81.59, hit-ratio: 0.07

ben adarında çıldı olunum:
Berin şayı fırttına incedirimi direl,
Bertesk Faşunları gölmedik dövgilmize İzallarına bir ku,
geldini kadar boknundan
Sındınları,
yürtü,
dirinin içinci pazmaş
bir avanda,
şarıp buç
karımızlarra
ve malı Papa Romini yıttı güç caştanından bire davadın,
beniz.
TARANTA - BABU..» baynaktı,
göz sanın
çokormak.
Benzin; karatlar
ve ROKa'ya iklerilde hapıradığı yabalış
yaptıldığı için Penci beşeşlerime bir ed bunuyumuk dibilini dediyle
delerinil biriüsümülür anla ayağı
mam.
Fakları

iteration: 8000, loss: 48.98, hit-ratio: 0.10

muş
oydular. Bir kiyapi konmuş.
Valancırın
arkadası kakamaz
yadsıca konuşup
heileriyenin dokuşulan sonraf Rom bilini apken ışığa bütün yolunda berleklerinin açık bir sevlâça
tatım çövereğini duymuş.
Düşünüyor
avalor
Efehbiri ben sarımın ağare
kitabı biranesi zinas
böyle bur adı İtalya'ya boşurmak oynan
altındık. Ve olmalarda gurur
vari birdi bir bağ eç koplak
Meyom
baçbırakla

iteration: 130000, loss: 29.45, hit-ratio: 0.27

eşlemimizin obadilmeliye amacağız dokuk olaca geldi. Rustmek cadara en güzel, bütün evleydi b
ilikti. Bunlar anamızı gözleri yapöklerini dedik
kırzda ku:
bindir sesinde gırdırlı kullanında
hakıl purup çekişler...
Biz de başka karanlık!.
Bun içinde, birdenbire kavular gibiderre çıkardı. Kağmara üzken mavi
miller ne bir ayaklara genç ile tahış olmî galindi.
Işıldı. Ben ısmektanların ekinde körleleri yürdüm şeirinde
bile
o gelmiş, senin üzerdinçe bir sen başkan gece

iteration: 247000, loss: 26.96, hit-ratio: 0.40

The spectacular thing about these results is that the LSTM neural network does not know what words, sentences, new lines, or punctuation means. The only thing the model knows is the characters. It only looks at the sequence of the characters, learns them, and predicts the next one.

Chapter 4

Methods

Initially, we started using vanilla recurrent neural networks and later on, we used a long short-term memory model for generating text. We train the networks in a supervised manner, in order to learn the dependencies between sequences of characters. At every iteration for characters from (i) to (i+sequenceLength), the aim is to be able to output characters from (i+1) to (i+sequenceLength+1).

We encode the characters as one-hot vectors. That is, for each character, there is a vector with the length of unique characters (vocabulary size) and all but one entry are zero. The non-zero entry corresponds to the character itself. Then there is one layer of hidden units. Hidden units are regular logistic functions like hyperbolic tangent for the case of recurrent neural networks and they are LSTM cells in the latter case. There is one layer of hidden units, followed by an output layer, which contains a node for each unique character in the vocabulary. These nodes contain the probabilities of its respective character. Finally, we sample from these probabilities to observe the character predicted, at that time step of the network.

The training is done for a large number of epochs until convergence. We calculate our loss via using the log-sum-exp function. During and after the training, we sample some random text and report the loss. These iterative results give us a window for interpretation and observe how the network evolves in time. We do this sampling for example, every 100^{th} epoch.

Every training is performed on a text file. Larger the text is, higher the number of hidden units shall be. We have used single hidden layered networks with variable number of hidden units for the phase of extensive training. After playing with the parameters a little, we decided to use 64, 128, and 256 hidden units. For the general case, a sequence length of 25 was sufficient. For the learning rate, we generally started from 10^{-1} and gradually decreased it to 10^{-3} until convergence.

Sampling is usually done via randomly choosing the first character from the current vocabulary. The rest of the characters are then predicted by the network. The vocabulary size is the number of unique characters of the text used in training. Upper and lower case letters are treated as separate characters, as expected. This limited size of the vocabulary arises a problem when we want to use starting sample text for sampling and the text contains a character that is not in the vocabulary. The probability of that character appearing in the prediction is of course zero since the network is oblivious to such characters. One possible solution is to perform online learning also during sampling. This would hardly affect the network weights but still creates a chance. Another solution would be increasing the range of training text to include more characters. However, this is not plausible since this introduces a lot variety, dampens our learning capability and results in under-fitting.

During training, the weights change after each epoch and the training phase for large input files usually takes half a day. So with personal computers, it may not be possible to converge to a local minima in a single run. To address that problem, we also introduced a checkpoint system. After every number of desired epochs, the training module saves the weights. This enables the user to prematurely end the training and later on, continue to train from that exact point of training as if no interruptions occurred.

The methods described here are demonstrated for the case of characters but it can also be easily applied to different domains. For example we can encode music scores as one-hot vectors, train using all compositions of a musician, and then sample a new compositions for him/her. The post-processing task here would be converting these scores into actual audio, in order to be able to make a sense of the accuracy of the results.

Chapter 5

Conclusion

- Sequence learning is a problem with a wide range of possible applications.
- Recurrent neural networks are good at learning sequences.
- **LSTM** neural networks are **better** at learning sequences due to their unique representation of long-term dependencies.
- We can learn and generate sequences that are good enough to be mistaken for the original.
- There is a limit to what a single LSTM cell can learn.
- There is no unique solution for the task of counting but many different weight initializations may lead into the same solution.

Chapter 6

Future Work

- There are some variants of the LSTM structure in the literature. This work can be extended to those variants and a similar in-depth analysis can be performed.
- Alternative variants to LSTM's can be proposed and surveyed. These topics are still an active area of research.
- The application range can be surveyed. There can be many undiscovered applications on LSTM's that that potentially could be ground breaking in the literature.

References

[1] *char-rnn*. URL: https://github.com/karpathy/char-rnn.

[2] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. 2013. eprint: arXiv:1308.0850.

[3] Klaus Greff et al. "LSTM: A Search Space Odyssey". In: *CoRR* abs/1503.04069 (2015). URL: http://arxiv.org/abs/1503.04069.

[4] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. *Visualizing and Understanding Recurrent Networks*. 2015. eprint: arXiv:1506.02078.

[5] Zachary Chase Lipton. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *CoRR* abs/1506.00019 (2015). URL: http://arxiv.org/abs/1506.00019.

[6] *The Unreasonable Effectiveness of Recurrent Neural Networks*. 2015. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[7] *Understanding LSTM and its diagrams*. 2016. URL: https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714#.89tzmilp8.

[8] *Understanding LSTM Networks*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.