Application of artificial neural networks with attention mechanism for discovering distant dependencies in time series

(Zastosowanie sieci neuronowych z mechanizmem uwagi
do odkrycia odległych zależności w szeregach czasowych)

Szymon Malik

Praca licencjacka

Promotor: dr Jan Chorowski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

9 września 2016

Szymon Malik

. . . .

. . . .

(adres zameldowania)

. . . .

. . . .

(adres korespondencyjny)

PESEL: . . . .

e-mail: . . . .

Wydział Matematyki i Informatyki
stacjonarne studia I stopnia
kierunek: Indywidualne Studia Informatyczno-Matematyczne
nr albumu: 264266

Oświadczenie
o autorskim wykonaniu pracy dyplomowej

Niniejszym oświadczam, że złożoną do oceny pracę zatytułowaną *Application of artificial neural networks with attention mechanism for discovering distant dependencies in time series* wykonałem/am samodzielnie pod kierunkiem promotora, dr. Jana Chorowskiego. Oświadczam, że powyższe dane są zgodne ze stanem faktycznym i znane mi są przepisy ustawy z dn. 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tekst jednolity: Dz. U. z 2006 r. nr 90, poz. 637, z późniejszymi zmianami) oraz że treść pracy dyplomowej przedstawionej do obrony, zawarta na przekazanym nośniku elektronicznym, jest identyczna z jej wersją drukowaną.

Wrocław, 9 września 2016

(czytelny podpis)

Abstract

This work explores the idea of extending known neural models with attention mechanism in order to enhance their performance. The work is focused on overcoming some weaknesses of LSTM networks as well as compressing information for feedforward language models.

Celem niniejszej pracy jest zbadanie idei rozszerzenia pewnych znanych modeli neuronowych o mechanizm uwagi. Praca jest skupiona na zwalczeniu pewnych wad sieci LSTM oraz kompresji informacji dla neuronowego (nierekurencyjnego) modelu językowego.

Contents

Chapter 1

Introduction

1.1 Neural networks

Artificial neural networks (in this thesis also called *neural networks*) are inspired by the way the human brain works. They are used to model data and make data-driven inference. Mathematically, a neural network is a function $g(x, \theta)$, parameterized by vector θ, where vector x is an input of the model. To optimize the model, given a set of points (x, y), one needs to define a loss (score) function, then vector θ is chosen by minimizing (maximizing) the objective function. Optimization is typically performed using gradient-based methods.

1.1.1 Recurrent neural networks (RNN)

When dealing with sequential data RNNs may turn useful as they are able to process data of variable length. At every time step t the hidden state h_t is computed based on the current input x_t and the previous hidden state: $h_t = g(x_t, h_{t-1}, \theta)$. θ is a vector of parameters that define the function g. Training of such model is done by unrolling network in time and applying gradient based optimization as in classical networks. However versatile, general recurrent networks are difficult to train for data with distant dependencies and very long sequences (due to phenomena of exploding or vanishing gradient).

1.1.2 Long Short-Term Memory (LSTM)

LSTM was originally introduced by Hochreiter and Schmidhuber [1]. It is a recurrent network enriched with a memory cell (LSTM cell) - a fixed size vector. At every time-step the network is able to change information stored in the cell (”forget” old information and ”remember” new data). LSTM networks proved to be able to overcome the mentioned above problem with vanishing gradients.

1.2 Attention mechanism

Attention mechanism is a way of creating a fixed size context from a variable-length input. For example a weighted average of sequential data as in work by Cheng et al. [2] or Raffel and Ellis [3]. In this work the attention mechanism will be used to produce an auxiliary context vector ($\sum_i f(input[i]) \cdot allignment\_weight[i]$) or a sequence $\{allignment\_weight[i]\}$.

1.3 Overview of this work

Intention of this thesis is to explore the idea behind the attention mechanism. Whether it is possible, with additional computation and memory usage, to enhance models performance in training time and generalization over unseen data. Towards this goal we conduct experiments to answer two questions:

1. Can we use alignment information obtained from the attention mechanism to ease training of LSTM networks?
2. Can we use the attention mechanism to improve on existing models for sequence prediction?

LSTM resolved major problems with training of recurrent networks (vanishing or exploding gradients due to backpropagation). Additionally networks gained ability to focus only on particular inputs through the use of the input gates, however they struggle to distinguish relevant ones. An idea is to enrich LSTM cell with additional attention mechanism that would help differentiate relevant inputs from irrelevant ones (by changing behavior of cell's gates (the input gate in particular)). One toy-problem with distant dependencies, described in 2.1, was tested as a "proof of concept" and starting point for the future work.

In the next experiment (section 2.2), an autoregressive language model inspired by Cheng et al. [2] and Raffel and Ellis [3] will be presented. Namely, the language model proposed by Bengio et al. [4] enhanced with attention mechanism will be tested.

Chapter 2

Neural models

2.1 Addition problem

The addition task is one of popular synthetic problems used to measure model's ability of tackling distant dependencies in data. The task is as follows:

The input is a sequence of pairs. The first component of each pair is a real number from the interval $[0, 1]$ and the second component is either 0 or 1. There are exactly two elements of the sequence marked with 1, where one is in the firs half of sequence and the other is in the second half. Desired output is the sum of numbers marked with 1's.

Two recurrent models were considered for this problem: a one-layer LSTM network enriched with attention mechanism used to control cell's input gate and a regular one-layer LSTM network as a reference point.

2.1.1 LSTM network

One-layer recurrent network with LSTM cell was used. The last hidden state (h_{final}) is fed to a single-layer neural network to determine the prediction (y):

$$y = f\left(W_{out} h_{final} + b_{out}\right)$$

where $W_{out} \in \mathbb{R}^{1 \times cell\_size}$, $b_{out} \in \mathbb{R}$ and f is a Leaky ReLU (rectified linear unit) activation function: $f(x) = max(x,\ 0.01x)$.

2.1.2 LSTM network with attention mechanism

This model consists of two parts:

- an attention mechanism that produces real-valued weights for every element of the input sequence

- a LSTM recurrent network with additional input - weights from the attention network

Attention mechanism

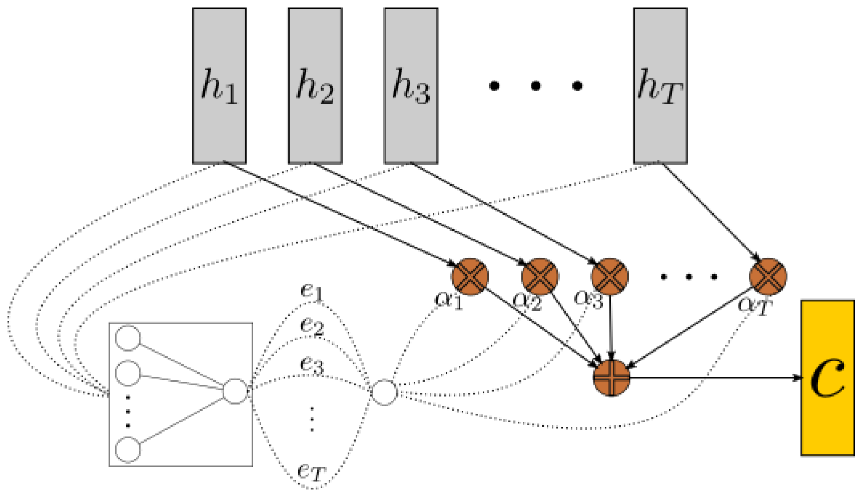

Figure 2.1: Schematic of described below attention mechanism

The following feed-forward model with attention proposed by Raffel and Ellis [3] was trained:
the model first computes hidden state h_t and energy e_t for each input element (independently from one another):

$$h_t = f\left(W_{xh}x_t + b_{xh}\right),\ W_{xh} \in \mathbb{R}^{D\times 2},\ b_{xh} \in \mathbb{R}^D$$

$$e_t = \tanh\left(W_{hc}h_t + b_{hc}\right),\ W_{hc} \in \mathbb{R}^{1\times D},\ b_{hc} \in \mathbb{R}$$

these energies are then normalized over all input elements:

$$\alpha_t = \frac{\exp\left(e_t\right)}{\sum_{k=1}^{T}\exp\left(e_k\right)}$$

and summarized into a context:

$$c = \sum_{t=1}^{T}\alpha_t h_t$$

then the context is fed to fully-connected network:

$$s = f\left(W_{cs}c + b_{cs}\right),\ W_{cs} \in \mathbb{R}^{D\times D},\ b_{cs} \in \mathbb{R}^D$$

$$y = f\left(W_{sy}s + b_{sy}\right),\ W_{sy} \in \mathbb{R}^{1\times D},\ b_{sy} \in \mathbb{R}$$

where x_t is t-th element of input sequence, $f = max(x,\ 0.01x)$ and D is size of hidden state.

"Attention sequence" passed to the recurrent layer consists of logits of attention layer, i.e. sequence $\{e_t\}$.

LSTM network

This model is similar to network described in subsection 2.1.1. The only difference is that this network accepts additional sequence ($\{e_t\}$) as an input and at every time step t the value of e_t is added to the input gate of the LSTM cell. The update of the cell and the computation of the new hidden state is performed according to the following formulas:

$$f_t = \sigma\left(W_{hf}h_{t-1} + W_{xf}x_t + b_f\right)$$

$$i_t = \sigma\left(W_{hi}h_{t-1} + W_{xi}x_t + b_i + e_t\right)$$

$$o_t = \sigma\left(W_{ho}h_{t-1} + W_{xo}x_t + b_o\right)$$

$$g_t = \tanh\left(W_{hg}h_{t-1} + W_{xg}x_t + b_g\right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

note that there is an extra term e_t in the formula for input gate (i_t).

2.2 Language model

The objective of this task is to create an accurate probabilistic language model, i.e. to model the conditional distribution of words given previous words.

2.2.1 Feed-forward network

Bengio et al. [4] proposed the following model over large but finite vocabulary with simultaneous training of feature word vectors, usually called embedding vectors:

$$m = \left(E(w_1), \ldots, E(w_{n-1})\right)^T, \ E \in \mathbb{R}^{V \times S}$$

$$h = f\left(W_{mh}m + b_{mh}\right), \ W_{mh} \in \mathbb{R}^{H \times (n-1)\cdot S}, \ b_{mh} \in \mathbb{R}^H$$

$$l = W_{hl}h + b_{hl}, \ W_{hl} \in \mathbb{R}^{V \times H}, \ b_{hl} \in \mathbb{R}^V$$

$$P\left(w_n = i \mid w_1, \ldots, w_{n-1}\right) = \frac{\exp(l_i)}{\sum_{j=1}^{V} \exp(l_j)}$$

where E is an embedding matrix, V is vocabulary size, S is number of features of individual word embedding, H is size of hidden layer, $f(x) = max(0,\ x)$ is an activation function and $(w_1, \ldots, w_{n-1})$ is an input sequence of words. Number of inputs $(n-1)$ is fixed due to limitations of the architecture.

2.2.2 Extension - attention mechanism

The above model was extended so that it accepts inputs of any length greater or equal to some fixed number n. The high level idea is to extend the range of words input to the model by selecting few words using the attention mechanism and combining them into an auxiliary input to the network. Then the routine is identical to the previous one (2.2.1). Details of attention mechanism follow (note that presented architecture of attention is identical to the one described in 2.1.2, consult figure 2.1 for details):

$$h_i = f\left(W_{wh} w_i + b_{wh}\right),\ W_{wh} \in \mathbb{R}^{S \times S},\ b_{wh} \in \mathbb{R}^S$$

$$e_i = tanh\left(W_{hc} h_i + b_{hc}\right),\ W_{hc} \in \mathbb{R}^{1 \times S},\ b_{hc} \in \mathbb{R}$$

$$\alpha_i = \frac{\exp\left(e_i\right)}{\sum_{k=1}^{t-n+1} \exp\left(e_k\right)}$$

$$c = \sum_{i=1}^{t-n+1} \alpha_i h_i$$

Where S is again size of embedding and $f(x) = max(0,\ x)$ is an activation function. Then vector $m = \left(c, E(w_{t-n+2}), \ldots, E(w_t)\right)^T$ is constructed and the remaining part of the routine is identical with that described previously in subsection 2.2.1.

Chapter 3

Experiments and results

All models were created and trained using TensorFlow library [5]. However slight changes in recurrent networks' routine were needed:

- input gate manipulation was enabled by introducing additional (optional) input - weights that are added to the gate at each time step
- in order to enable gates' saturation, $sigmoid(x)$ was replaced with $1.2sigmoid(x) - 0.1$

3.1 Addition problem

All models were trained using Adam Optimizer [6] with mean squared error as a loss function. Sequences of various lengths between 500 and 550 elements were used as training data. Weight were initialized randomly with entries drawn from the Gaussian distribution (mean zero, std $\frac{1}{\sqrt{output\_dim}}$). Biases were initialized with all entries set to zero. Both recurrent networks used 64-unit LSTM cells. Moreover, to establish the attention vectors a feed-forward network from 2.1.2 with hidden state of size $D = 100$ was trained.

3.1.1 Training pace

Both recurrent models were trained for 15000 time steps. Mini batches consisting of 50 sequences were used. Two following plots present learning pace for proposed models. Advantage of attention model was observed. It starts learning at around 2000th step while regular LSTM needs around 8000 steps.

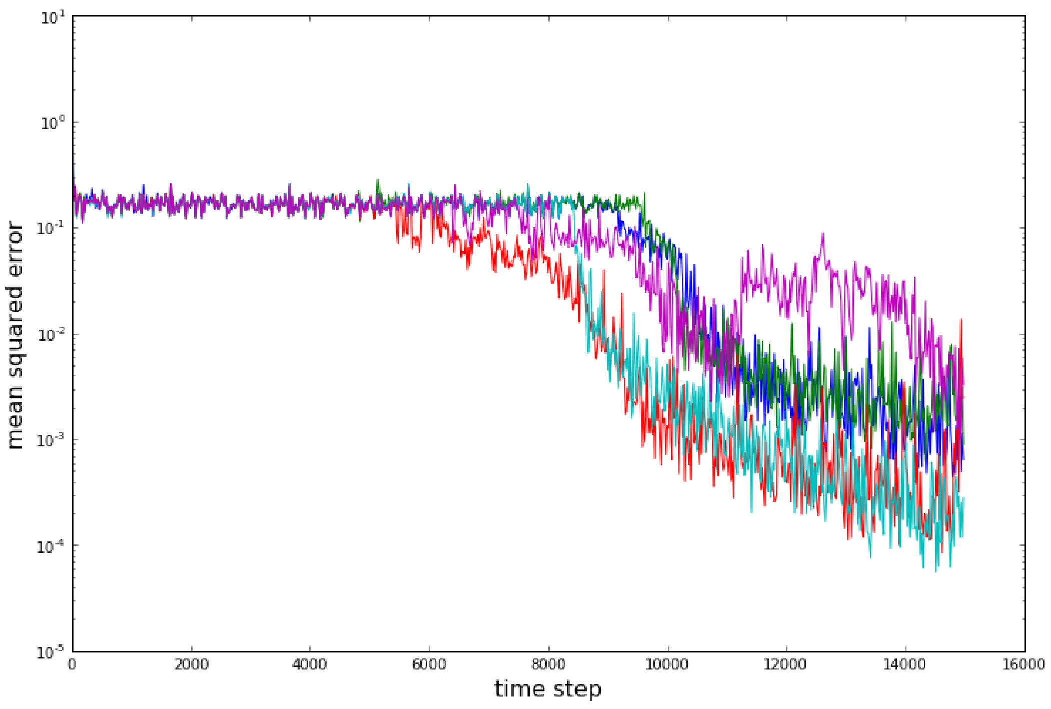

Figure 3.1: Five training runs of LSTM network

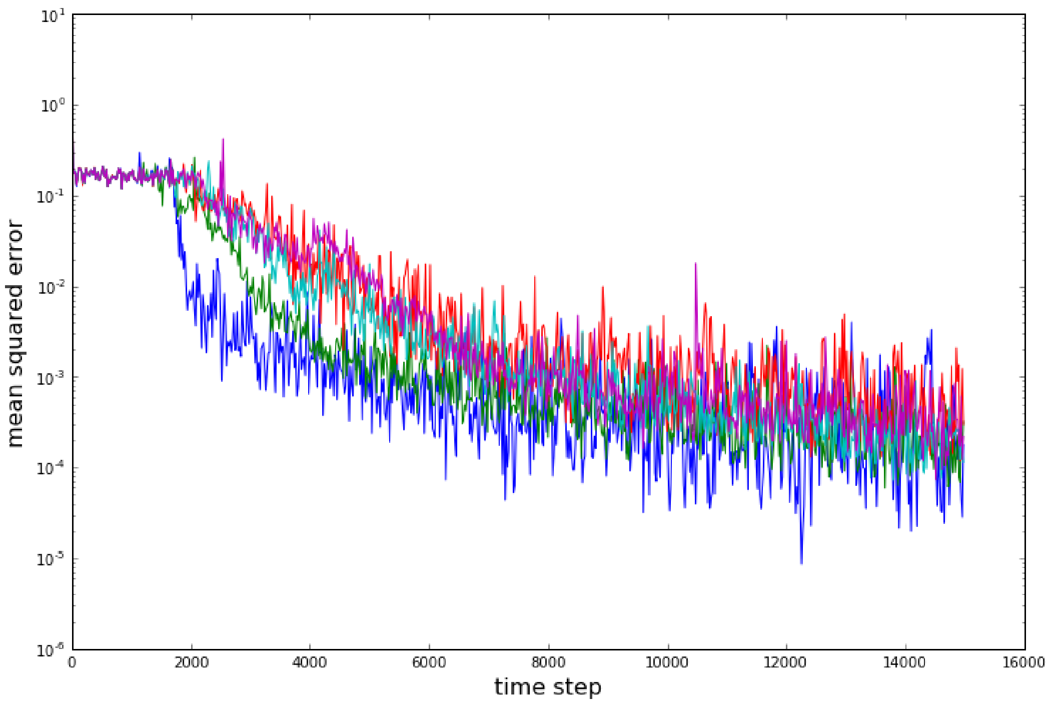

Figure 3.2: Five training runs of LSTM network with attention mechanism

3.1.2 Generalization error

Not only does attention mechanism make the model learn faster but it also helps it generalize better. On average, even for sequences of length 6000, error of LSTM with attention is below the base line (variance of sum of two variables drawn from $Uniform[0, 1]$).

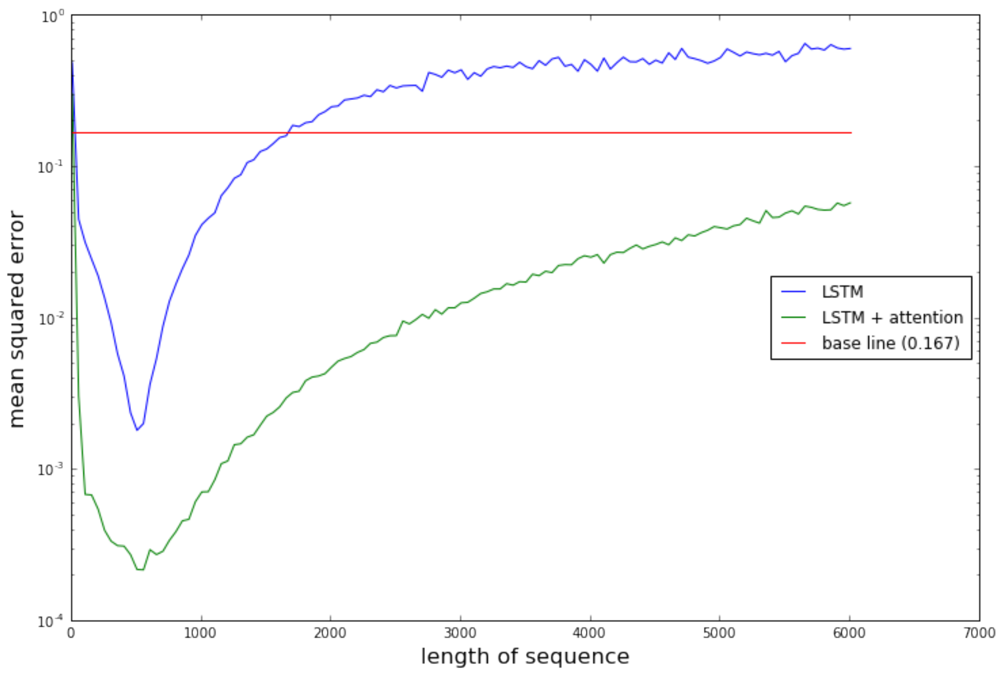

Figure 3.3: Comparison of average generalization errors for both recurrent models

3.2 Language model

3.2.1 Corpus description

To train all these models a fragment of Polish Wikipedia consisting of $7,000,000$ words was used. Vocabulary contain 37235 most frequent words in data (words that occurred at least 10 times) and one artificial word $\langle$UNK$\rangle$ which replaced all less frequent words. $6,000,000$ words were used as training data. Remaining 1 million words formed validation and test data ($500,000$ each).

3.2.2 Training details

All models were trained using Gradient Descent optimizer for 25 epochs with mini batches of 1000 training examples. Objective was to maximize probability of "correct" word for given training sequences. Loss function to be minimized is the mean negative log-likelihood over the mini-batch. Also a weight decay term 10^{-5} for matrices W_{mh}, W_{hl} was used. Embedding vectors of size $S = 100$ were used. Hidden layer size H was set to 200 units. Learning rate in the i-th step was $\epsilon_i = \frac{\epsilon_0}{1+10^{-8} \cdot i}$ where $\epsilon_0 = 0.2$

3.2.3 Results

Architecture selection

Four different models were trained (5 instances each):

- model described in 2.2.1 trained on 5-grams, namely given 4 words predict fifth one
- model trained on 10-grams
- model with attention trained on 5-grams with 2-word prefix, namely a network directly sees 4 words and an auxiliary context vector created from the prefix
- model with attention trained on 5-grams with additional 5-word prefix

Like Bengio et al. [4] models' performance was tested comparing achieved perplexity on validation and test data, where perplexity is exponential of average negative log-likelihood.

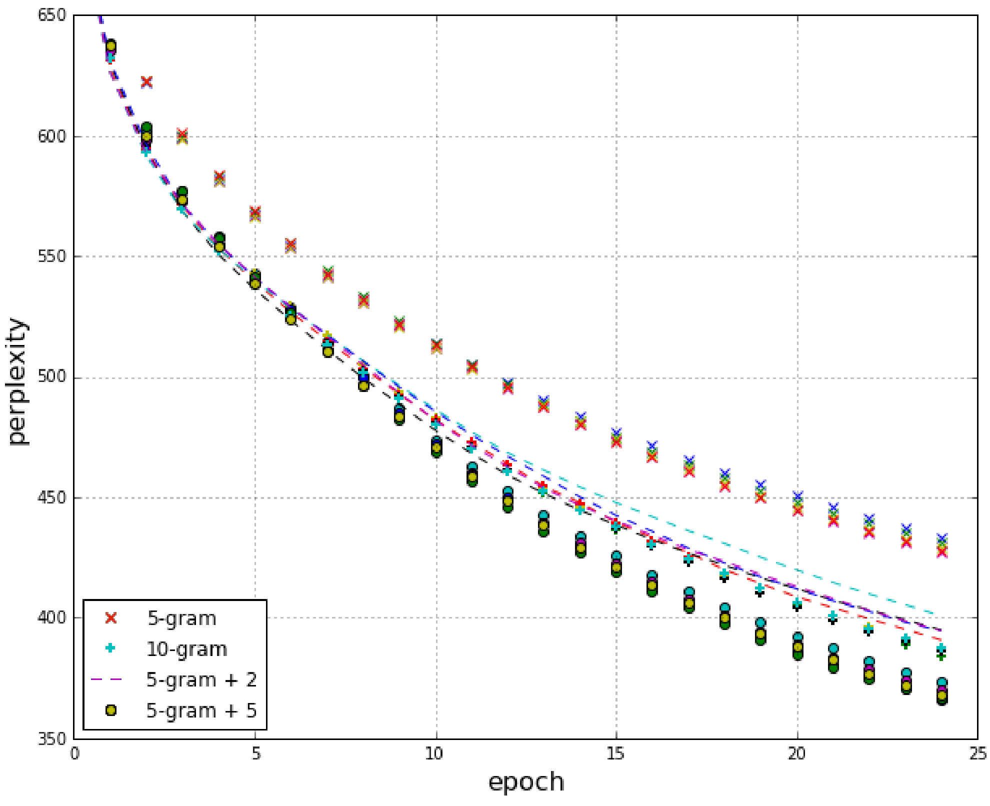

Figure 3.4: Perplexity on the validation set after each epoch

model	mean perplexity	standard deviation
5-gram	544.70	6.67
10-gram	488.78	2.88
5-gram + 2	487.38	6.53
5-gram + 5	460.07	6.16

Table 3.1: Perplexity on the test set

Attention models consistently performed better than regular feed-forward model with advantage of longer prefix over shorter ones. It is worth mentioning that 10-gram model achieved worse results that smaller (5-gram + 5) attention model.

Further tests

After these preliminary experiments three models (most promising - 10-gram and 5-gram + 5 and, as a reference point, 5-gram model) were trained more carefully using momentum = 0.85 in optimization. Learning rate in the i-th step was $\epsilon_i = \frac{\epsilon_0}{1+10^{-5} \cdot i}$ where $\epsilon_0 = 0.15$

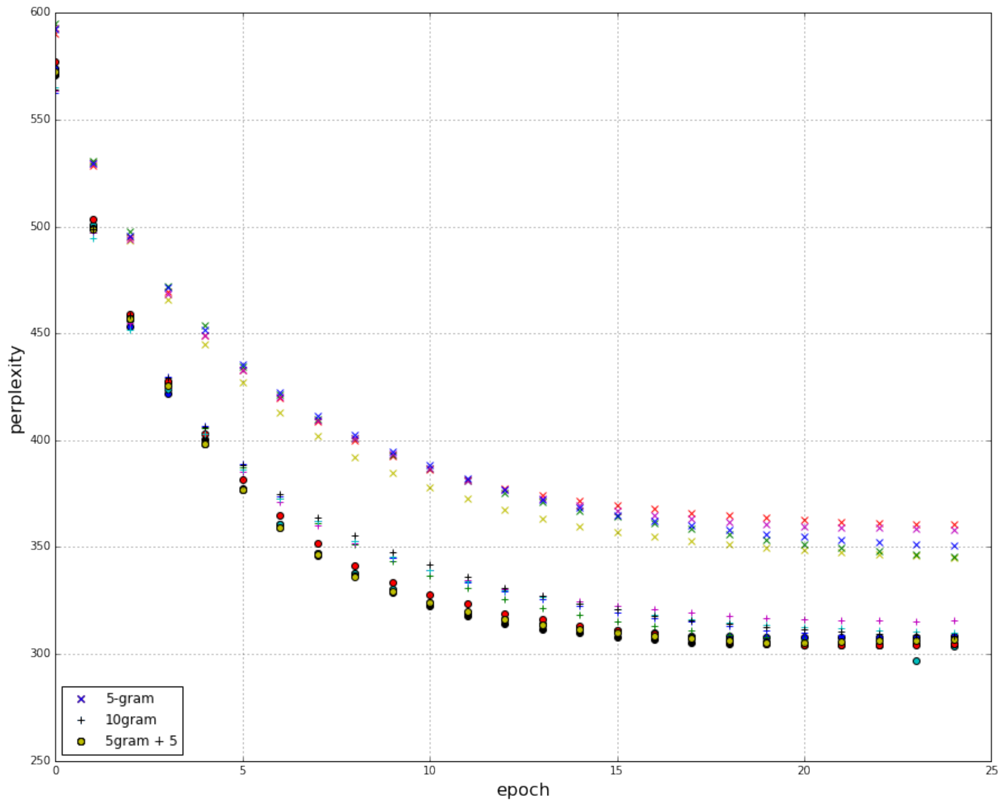

Figure 3.5: Perplexity on the validation set after each epoch

model	mean perplexity	standard deviation
5-gram	493.04	16.41
10-gram	425.90	8.36
5-gram + 5	411.43	8.37

Table 3.2: Perplexity on the test set

Attention model, being much smaller than the 10-gram model, had slightly better performance in terms of perplexity on the test set and convergence time.

3.2.4 Examples of weights assigned by attention

One more model was pre-trained on larger (40 million words) data - articles from *The Wall Street Journal*. Examples of activation of attention are presented below. Whole sentence is given, then results of attention on 5-word substrings are presented. Transparency of words corresponds to weights assigned to them.

1. BRIAN ROSNER AN ASSISTANT DISTRICT ATTORNEY IN MANHATTAN WHO EVENTUALLY PROSECUTED MESSRS. GRAMBLING AND LIBMAN

   BRIAN ROSNER AN ASSISTANT DISTRICT
   ROSNER AN ASSISTANT DISTRICT ATTORNEY
   AN ASSISTANT DISTRICT ATTORNEY IN
   ASSISTANT DISTRICT ATTORNEY IN MANHATTAN
   DISTRICT ATTORNEY IN MANHATTAN WHO
   ATTORNEY IN MANHATTAN WHO EVENTUALLY
   IN MANHATTAN WHO EVENTUALLY PROSECUTED
   MANHATTAN WHO EVENTUALLY PROSECUTED MESSRS.
   WHO EVENTUALLY PROSECUTED MESSRS. GRAMBLING
   EVENTUALLY PROSECUTED MESSRS. GRAMBLING AND
   PROSECUTED MESSRS. GRAMBLING AND LIBMAN

2. THE COMPANY SAID THE PLANT WILL BE COMPLETED BY LATE NINETEEN EIGHTY EIGHT AND WILL EMPLOY BETWEEN ONE THOUSAND AND ONE THOUSAND FIVE HUNDRED PEOPLE

   THE COMPANY SAID THE PLANT
   COMPANY SAID THE PLANT WILL
   SAID THE PLANT WILL BE
   THE PLANT WILL BE COMPLETED
   PLANT WILL BE COMPLETED BY
   WILL BE COMPLETED BY LATE
   BE COMPLETED BY LATE NINETEEN
   COMPLETED BY LATE NINETEEN EIGHTY
   BY LATE NINETEEN EIGHTY EIGHT
   LATE NINETEEN EIGHTY EIGHT AND

NINETEEN EIGHTY EIGHT AND WILL
EIGHTY EIGHT AND WILL EMPLOY
EIGHT AND WILL EMPLOY BETWEEN
AND WILL EMPLOY BETWEEN ONE
WILL EMPLOY BETWEEN ONE THOUSAND
EMPLOY BETWEEN ONE THOUSAND AND
BETWEEN ONE THOUSAND AND ONE
ONE THOUSAND AND ONE THOUSAND
THOUSAND AND ONE THOUSAND FIVE
AND ONE THOUSAND FIVE HUNDRED
ONE THOUSAND FIVE HUNDRED PEOPLE

3. TRADE TENSIONS ARE A GROWING CONCERN IF WE GET INTO A BROAD FRONT TRADE WAR IT COULD SLOW EVERYONE DOWN AND CHANGE THE POLITICAL CLIMATE DRAMATICALLY

TRADE TENSIONS ARE A GROWING
TENSIONS ARE A GROWING CONCERN
ARE A GROWING CONCERN IF
A GROWING CONCERN IF WE
GROWING CONCERN IF WE GET
CONCERN IF WE GET INTO
IF WE GET INTO A
WE GET INTO A BROAD
GET INTO A BROAD FRONT
INTO A BROAD FRONT TRADE
A BROAD FRONT TRADE WAR
BROAD FRONT TRADE WAR IT
FRONT TRADE WAR IT COULD
TRADE WAR IT COULD SLOW
WAR IT COULD SLOW EVERYONE
IT COULD SLOW EVERYONE DOWN
COULD SLOW EVERYONE DOWN AND
SLOW EVERYONE DOWN AND CHANGE
EVERYONE DOWN AND CHANGE THE
DOWN AND CHANGE THE POLITICAL
AND CHANGE THE POLITICAL CLIMATE
CHANGE THE POLITICAL CLIMATE DRAMATICALLY

Chapter 4

Conclusions and future work

Both experiments showed that enhancing known neural models with attention mechanism helps in general performance - learning pace and generalization over unseen data.

4.1 Addition problem

Experiments with addition task presented in 2.1.2 may be perceived as proof of the concept that LSTM's difficulties in distinguishing relevant inputs may be resolved with external gate manipulation. Possible continuation of this work could be to apply this idea to more difficult, ideally not synthetic, problem.

4.2 Language model

Generalization of attention model trained on sequences with moderately short prefixes to longer prefixes should be tested in future work. Possible extension of this model is making network "aware" of location and distance between words in prefix, currently it is indistinguishable. First attempts in implementing this idea were made. So far no significant advantage was observed.

Bibliography

[1] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory.* Neural Computation 9(8):1735-1780, 1997.

[2] Jianpeng Cheng, Li Dong and Mirella Lapata. *Long Short-Term Memory-Networks for Machine Reading.* arXiv:1601.06733, 2016.

[3] Colin Raffel and Daniel P. W. Ellis. *Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems.* arXiv:1512.08756, 2015.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent and Christian Jauvin. *A Neural Probabilistic Language Model.* Journal of Machine Learning Research 3 (2003) 1137–1155, 2003.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.

[6] Diederik Kingma, Jimmy Ba. *Adam: A Method for Stochastic Optimization.* arXiv:1412.6980, 2014.