DNN assignment 3

Tomasz Garbus

June 2018

Contents

1 Input format **1**

2 Loss function **1**

3 Dataset **2**
3.1 Generating bracket sequences . . . 2
3.2 Sequences distribution in dataset . . . 2
3.3 Validation set . . . 2

4 Configurations **2**
4.1 Accuracies after 100 first epochs . . . 2
4.2 Accuracies after 600 epochs . . . 3
4.3 Best model, after 600 epochs . . . 3
4.4 Plots (after 100 epochs) . . . 4
4.4.1 BasicRNN, 64 hidden neurons, 1 bracket at a time . . . 4
4.4.2 BasicRNN, 64 hidden neurons, 10 brackets at a time . . . 5
4.4.3 BasicRNN, 256 hidden neurons, 1 bracket at a time . . . 5
4.4.4 BasicRNN, 256 hidden neurons, 10 brackets at a time . . . 6
4.4.5 LSTM, 64 hidden neurons, 1 bracket at a time . . . 6
4.4.6 LSTM, 64 hidden neurons, 10 brackets at a time . . . 7
4.4.7 LSTM, 256 hidden neurons, 1 bracket at a time . . . 7
4.4.8 LSTM, 256 hidden neurons, 10 brackets at a time . . . 8

5 Script **8**
5.1 Example . . . 8

1 Input format

Opening brackets are denoted with 1, closing with -1. The maximum sequence length equals 100. Shorter sequences are padded with 0s after the sequence.

2 Loss function

As candidates for loss function I have considered Mean Squared Error and Absolute Difference. I decided to choose **Mean Squared Error**, because it gives more significance to the maximum error of 3 statistics.

3 Dataset

3.1 Generating bracket sequences

I have prepared 3 bracket sequence generators:

- `RandomBrackGen`: for given length of the sequence, generates a random sequence, starting from an empty string and inserting `'()'` at random position of the sequence in a loop.
- `FixedParamsGen`: for given length and 3 statistics from the task statement, generates a bracket sequence satisfying both length and the 3 statistics. It also offers a function determining whether such sequence exists.
- `OpenCloseBrackSeqGen`: for given `length`, generates a sequence $\texttt{'('} \cdot \frac{\texttt{length}}{2} + \texttt{')'} \cdot \frac{\texttt{length}}{2}$

3.2 Sequences distribution in dataset

The dataset consists of:

- exactly one sequence for each (achievable) combination of statistics up to length of 100, generated by `FixedParamsGen`.
- 200000 sequences generated by `RandomBrackGen`, equally distributed among all even lengths up to 100.
- exactly one sequence generated with `OpenCloseBrackSeqGen` for each even length up to 100.

3.3 Validation set

Validation set consists of 10% of the dataset extracted randomly.

4 Configurations

All models have been trained with:

- batch size = 1000
- steps per epoch = 200
- number of epochs = 200
- learning rate = 0.01 for LSTM
- learning rate = 0.001 for BasicRNN

I have decreased the learning rate for basic RNN, because otherwise I could not get them off the ground (they would get stuck at predicting the average value for each statistic each time). I have decided to run several configurations for 100 epochs to choose one configuration with LSTM and one without LSTM.

4.1 Accuracies after 100 first epochs

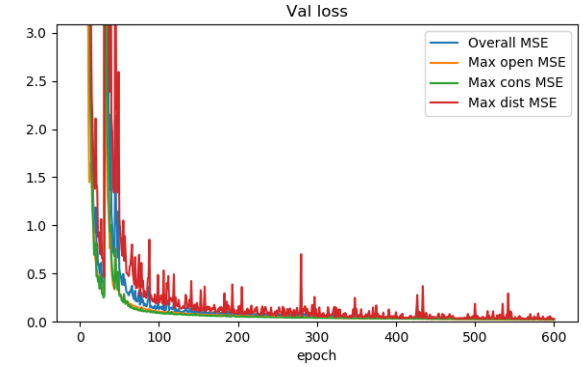
LSTM	hidden neurons	brackets looked at	max_open acc	max_cons acc	max_dist acc	loss
yes	256	1	0.92644	0.92719	0.82781	0.11040
yes	256	10	0.87572	0.82509	0.76313	0.21770
yes	64	10	0.77928	0.55334	0.71935	0.41527
yes	64	1	0.71171	0.72495	0.41282	0.66301
no	256	10	0.0251	0.51816	0.49462	1.25853
no	64	10	0.40900	0.41203	0.28435	3.96005
no	64	1	0.01727	0.02375	0.00554	258.77255
no	256	1	0.01703	0.02357	0.01939	264.50741

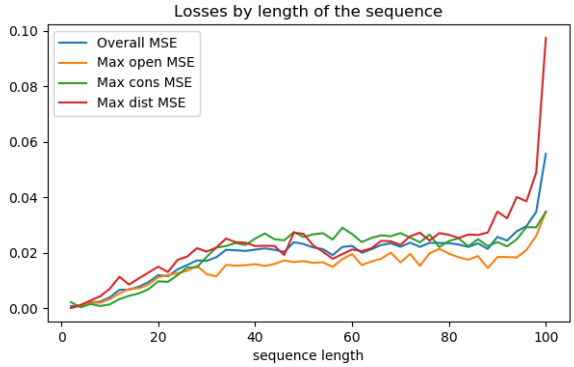
4.2 Accuracies after 600 epochs

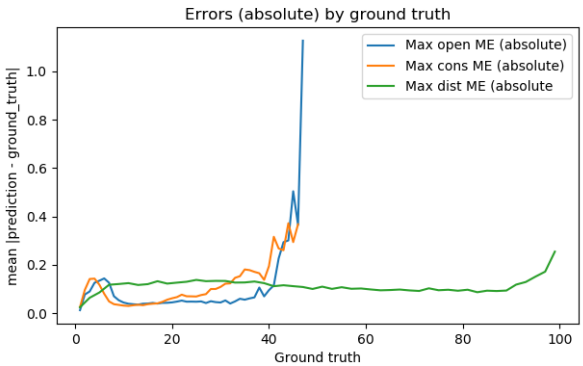
I have trained the best model with LSTM and the best model without LSTM for 500 more epochs.

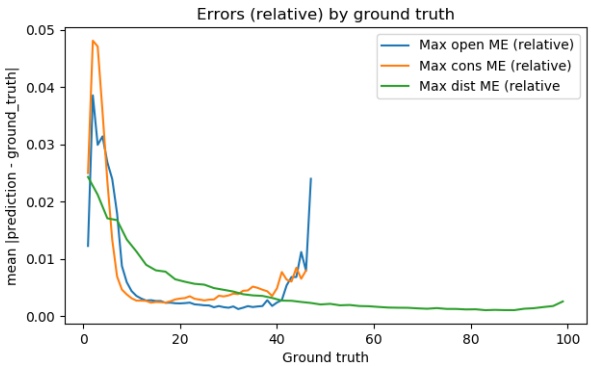
LSTM	hidden neurons	brackets looked at	max_open acc	max_cons acc	max_dist acc	loss
yes	256	1	0.99542	0.99285	0.98143	0.02233
no	256	10	0.74940	0.68584	0.68150	0.44943

4.3 Best model, after 600 epochs



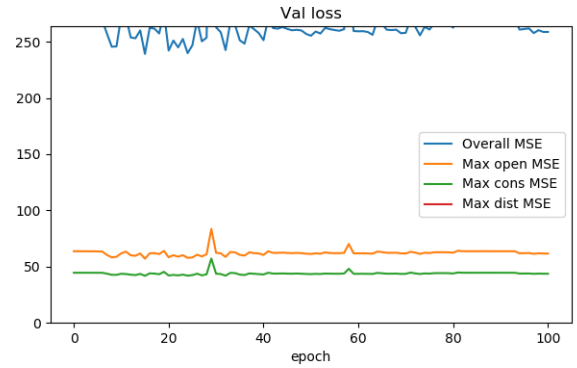


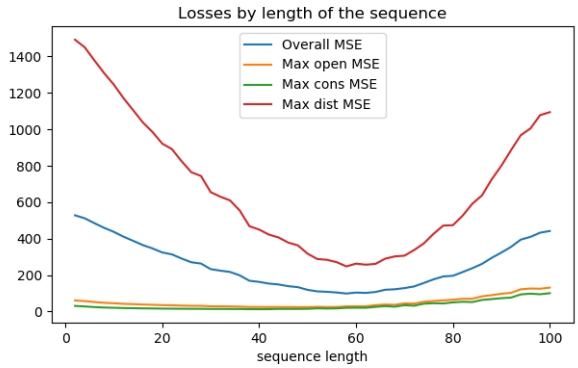


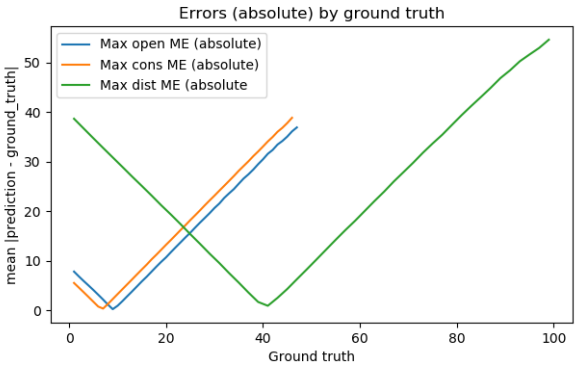


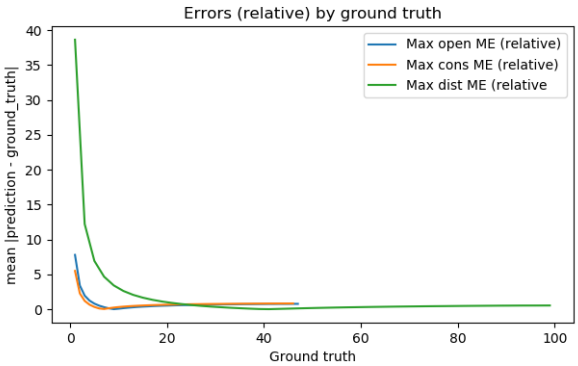
4.4 Plots (after 100 epochs)

4.4.1 BasicRNN, 64 hidden neurons, 1 bracket at a time

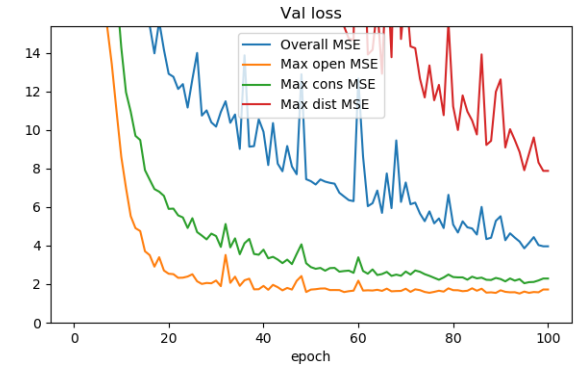


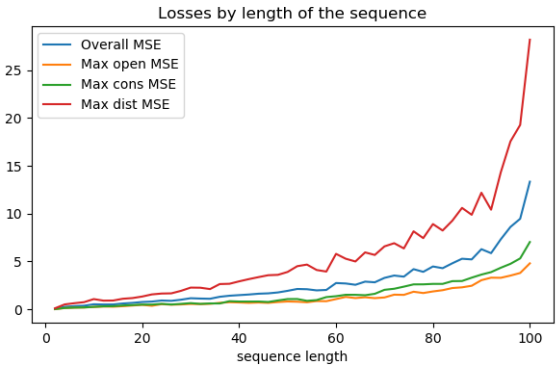


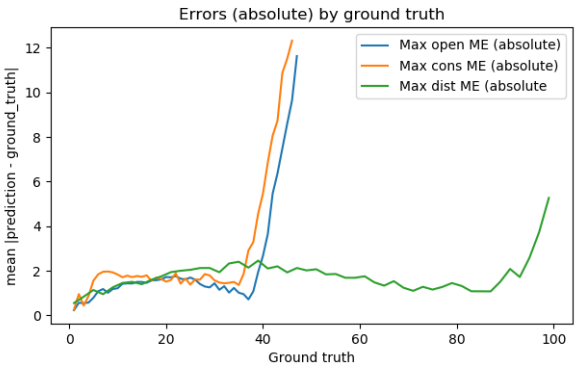


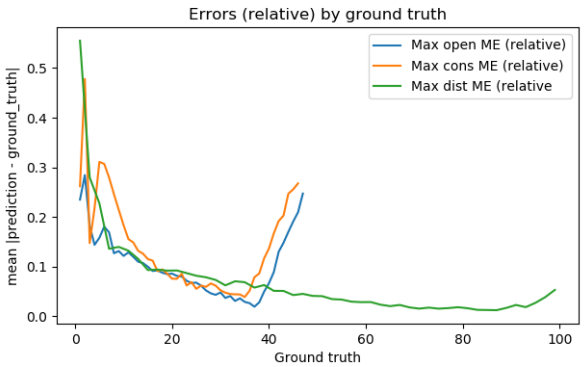


4.4.2 BasicRNN, 64 hidden neurons, 10 brackets at a time

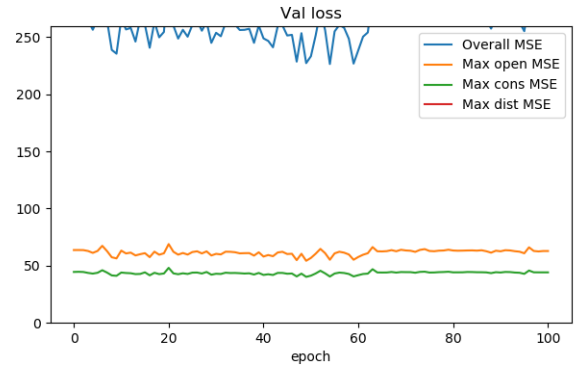


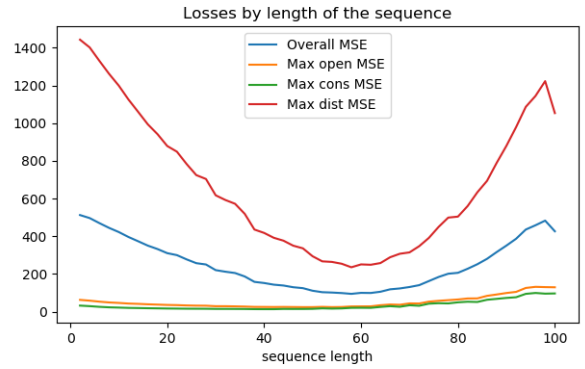


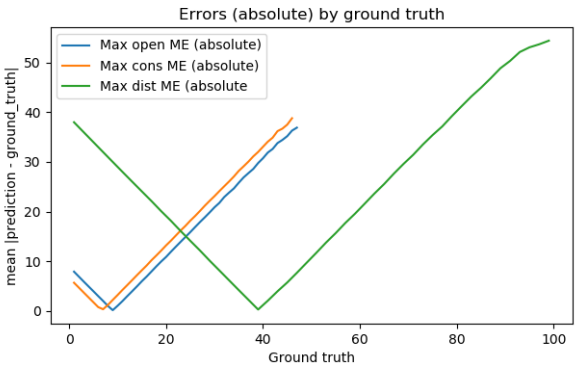


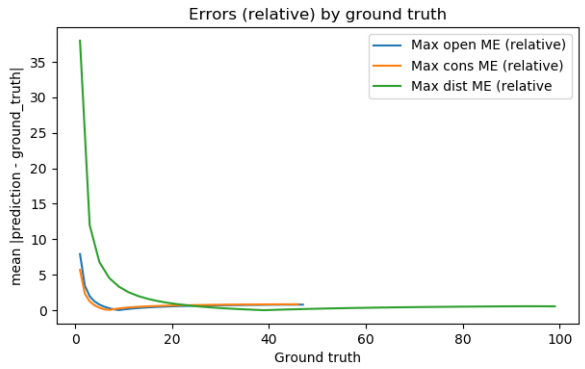


4.4.3 BasicRNN, 256 hidden neurons, 1 bracket at a time

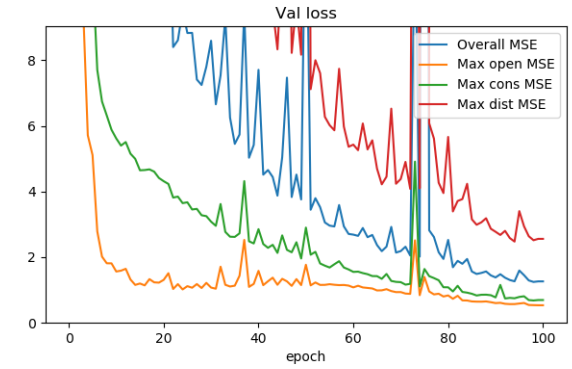


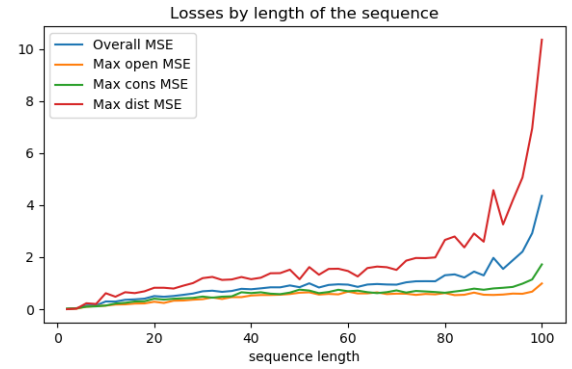


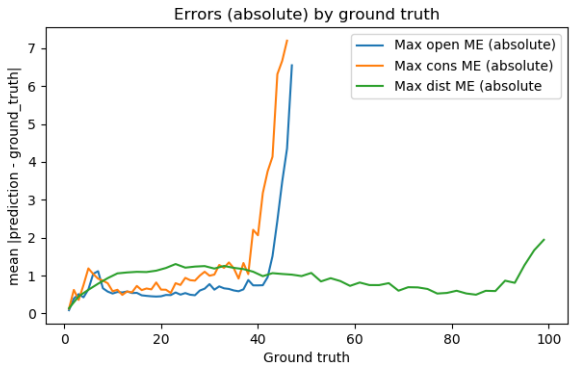


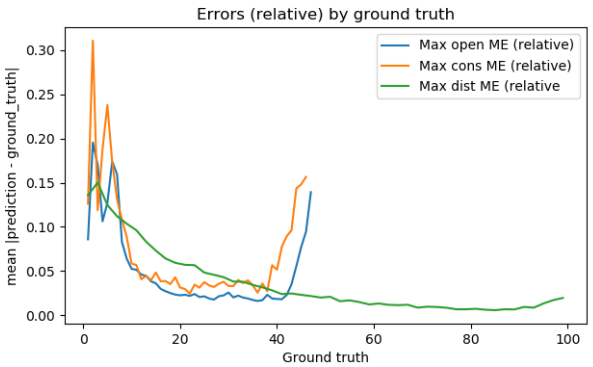


4.4.4 BasicRNN, 256 hidden neurons, 10 brackets at a time

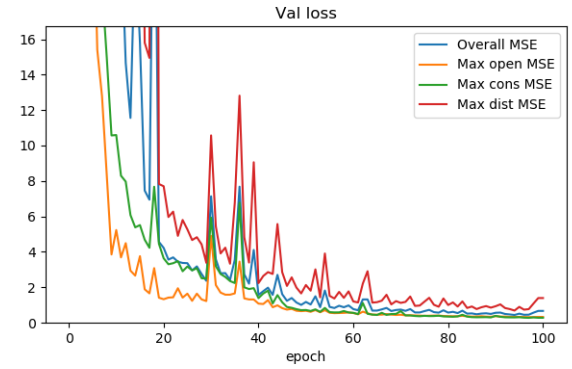


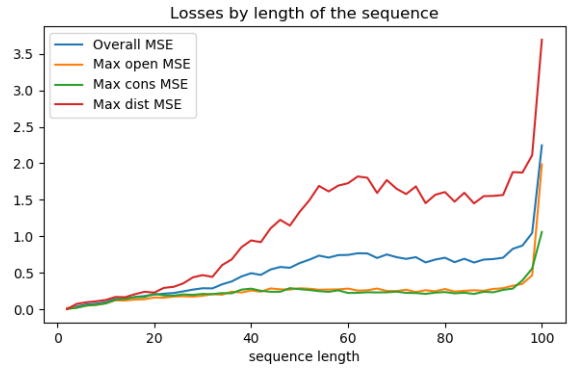


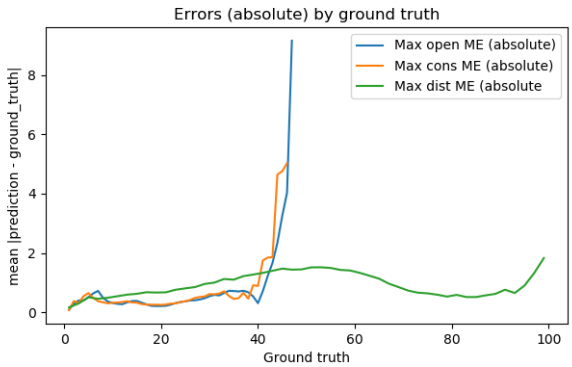


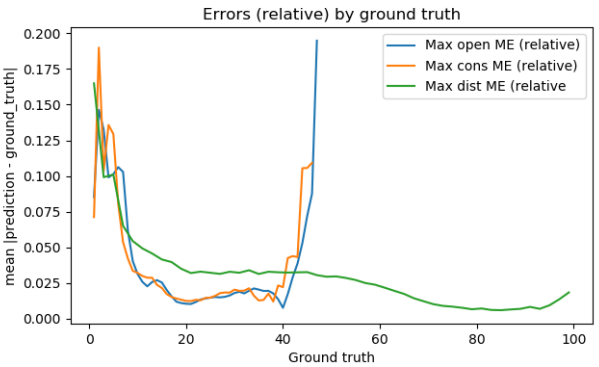


4.4.5 LSTM, 64 hidden neurons, 1 bracket at a time

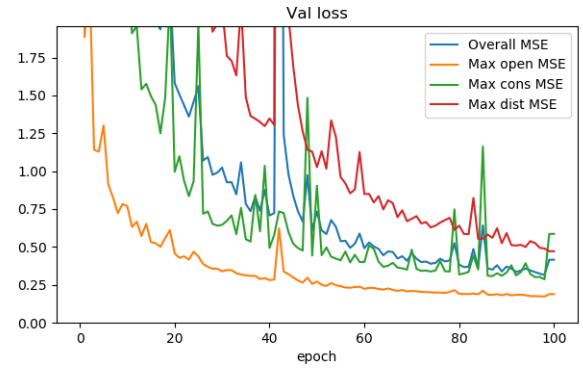


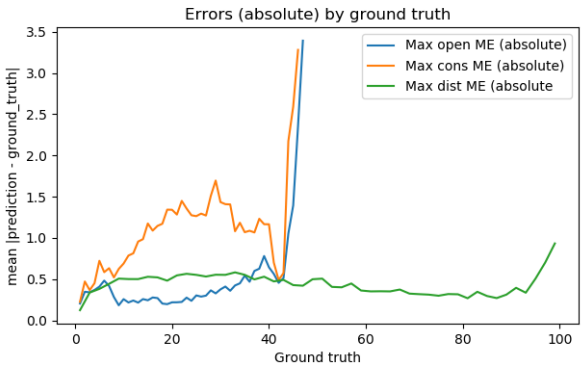


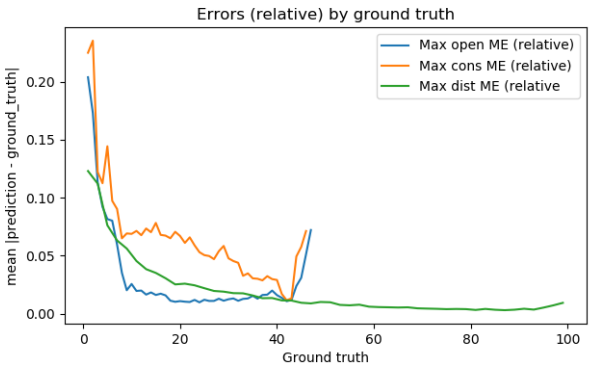




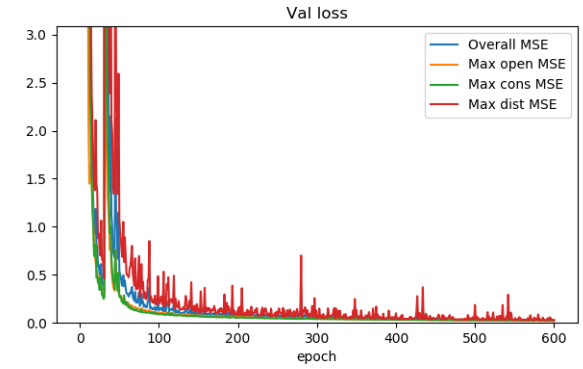
4.4.6 LSTM, 64 hidden neurons, 10 brackets at a time

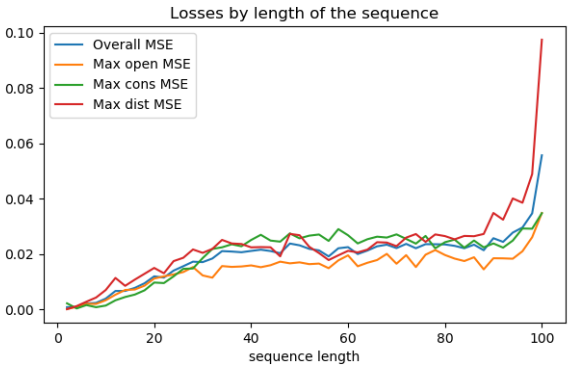


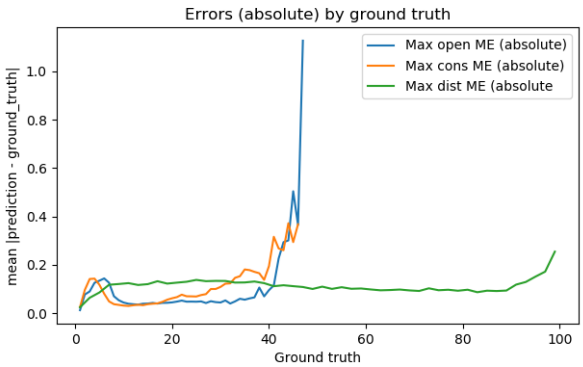


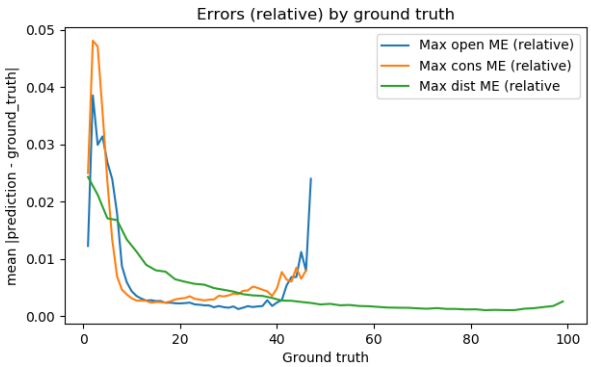


4.4.7 LSTM, 256 hidden neurons, 1 bracket at a time

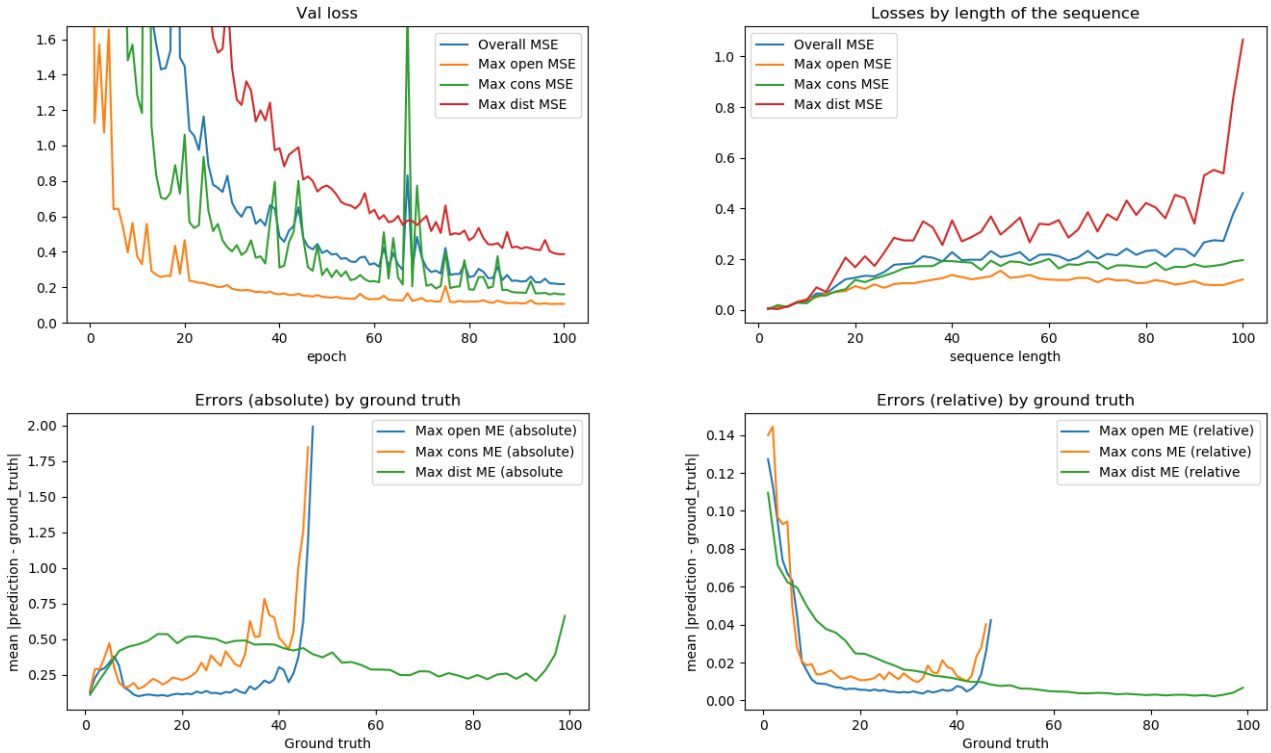








4.4.8 LSTM, 256 hidden neurons, 10 brackets at a time



5 Script

I have prepared a script for both training and interacting with RNN:

```
usage: main.py [-h] [-lr LR] [-n N] [-e E] [-b B] [--mode {lstm,rnn}] [--interact]
Train or play with RNN.
optional arguments:
-h, --help show this help message and exit
-lr LR learning rate
-n N hidden neurons count
-e E number of epochs
-b B brackets batch size
--mode {lstm,rnn}
--interact
```

Without the `--interact` flag, the network trains for `e` epochs and stores a checkpoint in `tmp/` directory and plots in `plots/` directory.

With `--interact` flag, it accepts bracket sequences from stdin and for each, displays predictions and correct answers. Provide the sequence in "natural" format, i.e. without converting to 1s and -1s.

5.1 Example

```
 > ipython main.py -- --mode=lstm -n 256 --interact
(((())))
Predictions:  [[2.9556935 3.099603 5.0717382]] , correct answers:  [[3 3 5]]
()()(()())
Predictions:  [[2.173327 2.0808468 4.8551016]] , correct answers:  [[2 2 5]]
((()))
```

```
AssertionError:  invalid bracket sequence
```