Clause identification with Long Short-Term Memory

James Hammerton
Department of Computer Science
University College Dublin
james.hammerton@ucd.ie

1 Introduction

Long Short-Term Memory (LSTM) is a new recurrent neural network architecture and training algorithm capable of remembering information over long time periods, and thus addresses the problem of recurrent networks forgetting information as a sequence is processed.

In this exploratory work, LSTM is applied to the part 3 of the CoNLL 2001 shared task to see how well it copes with a complex task involving real-world data.

2 LSTM

LSTM networks consist of 3 layers, an input layer, a recurrent hidden layer and an output layer. The main innovation in LSTM is the recurrent hidden layer. This consists of one or more memory blocks each containing one or more memory cells. Typically the inputs are connected to all of the cells and gates. The cells are connected to the outputs and the gates are connected to other cells and gates in the hidden layer.

Figure 1 depicts a single-cell memory block. The block consists of an input gate, the memory cell and an output gate. The memory cell is a linear unit with self-connection with a weight of value 1. When not receiving any input, the cell maintains its current activation over time. The input to the memory cell is passed through a squashing function and gated (multiplied) by the activation of the input gate. The input gate thus controls the flow of activation into the cell.

The memory cell's output is also passed through a squashing function before being gated by the output gate activation. Thus the output gate controls the flow of activation from cells to outputs. During training the input and output gates learn to open and close in order to let new information into the cells and let the cells

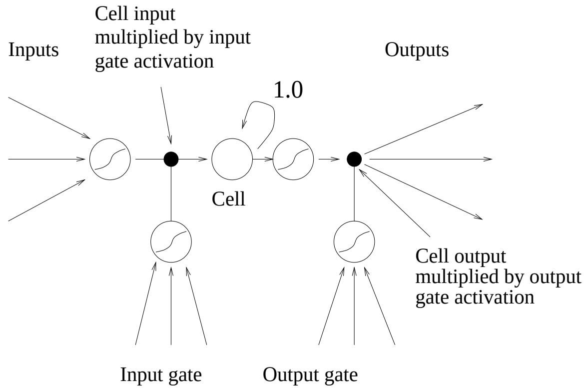

Figure 1: A single-celled memory block

influence the outputs. The cells themselves otherwise hold onto information unless new information is accepted by the input gate. Training of LSTM networks proceeds by a fusion of back-propagation through time and real-time recurrent learning, details of which can be found in (Hochreiter and Schmidhuber, 1997).

LSTM networks thus learns to remember information for arbitrary periods of time. In artificial tasks LSTM has been shown to be capable of remembering information for up-to 1000 time-steps. It thus tackles one of the most serious problems affect the performance of recurrent networks on temporal sequence processing tasks.

3 Approach

The LSTM network was trained to do the task as follows:

- The input to the network consists of vectors representing the current word, its associated part of speech (POS) tag and a tag representing the base chunk the word belongs to. The vector representations of word, the chunk tags and POS tags consisted of 25, 5 and 7 units respectively. The word representations were derived using lexical space encodings (Zavrel and Veenstra, 1996), the rest by hand[1]. An input unit is used to indicate which pass through the current sentence the network is currently in (see below). There are thus a total of 38 input units.

- The outputs were represented by 13 unit vector representations. These were supposed to indicate the number of clauses starting or ending on the current word by having 2 sets of units, N units of which were set to the value "1" when N clauses started or ended on the current word. However a mistake in the encoding was only spotted when all the training and testing on the development data had been done. Note that when presented as targets to the network, the zeros are converted 0.1s and the ones to 0.9s. This helps training, most probably because it helps prevent the weights taking on very high values that make the network resistant to change.
- Each sentence is presented to the network, one word/POS tag/chunk tag at a time, in two passes. On the first pass, the network is not presented with target outputs. On the second pass the network has to output the number of clauses starting and ending on the current word.
- The purpose of the first pass is to enable the network to collect information to use in disambiguation during the second pass.
- A network with 12 memory blocks of 8 cells was used in training.
- The network was trained for 1000 iterations on the first 1000 sentences from the sentences from sections 15 to 18 of the Wall Street Journal corpus. The best set of weights as indicated by the number of erroneous patterns was then used in testing. When deciding if a pattern was erroneous a tolerance of 0.39 was used. I.e. if all output activations came within 0.39 of the target value the pattern was deemed to be correct. The learning rate was 0.3 and no momentum was used.
- The network was tested on the development and test data for the shared task. When evaluating the network's output, the outputs are converted to brackets. These are balanced by assuming that every opening bracket the net outputs is valid, and discarding any extra closing brackets. Should closing brackets need to be added, they are added after the final word in the sentence (i.e. before the final punctuation).

The network used above was much larger than anything used in the LSTM literature and was intensive to train (hence the decision to train only on the first 1000 sentences). Whilst this may in part reflect the difficulty of the task, it may also reflect the possibility that the approach to training the network for the task is not optimal. E.g. correcting the mistake in the encoding of the output tags could improve performance.

During investigation to see whether training of some LSTM networks could be improved for another task (noun-phrase bracketing), it was discovered that outputting all zeroes (as opposed to not using targets) during the first pass improved performance, as did using orthogonal representations for the POS and chunk tags. Applying these changes (and correcting the mistake in the encoding of the output tags described above) enabled a network with only 12 blocks of 4 cells to be trained and to reach a similar level of performance on the same training set as the network above. This network had 90 inputs due to the use of orthogonal vectors for the tags.

Finally, a third network was trained on the first 2000 sentences of the training set, which also employed 12 blocks of 4 cells and the same vector representations of the words, tags and output as with the second network above.

4 Results

Table 1 gives the results of training the networks mentioned above and Table 2 gives the results of testing them. Note that the fscores are below the baseline value given in the specification for the shared task in the first 2 cases but the fscore is just above baseline in the 3rd case. Note also

[1] Contact the author for details of the input/output representations.

training	Iters	MSE	Errs	Weights
Net 1	998	0.028	1047	17883
Net 2	997	0.014	1014	13495
Net 3	983	0.016	2817	13495

Table 1: Results of training 2 LSTM networks on first 1000 sentences of the training set and a 3rd network (Net 3) on the first 2000 sentences.

that the recall is above baseline in all 3 cases. The column headings are as follows:

- Iters. The number of iterations performed when the best error was reached.
- MSE. The mean squared error between the target outputs and the actual outputs.
- Errs. The number of erroneous patterns.
- Weights. The number of weights in the network (the most accurate measure of network size).
- Precision. The percentage of clauses found that were correct.
- Recall. The percentage of clauses defined in the corpus that were found.
- Fscore. $F = \frac{2PR}{P+R}$ where P=precision and R=recall.

5 Concluding Remarks

As noted earlier this work was exploratory in nature as part of an investigation into using LSTM networks for shallow parsing tasks. Given that these preliminary attempts to do this task and the networks were trained on only a small part of the training set, which has a total of 8936 sentences, and that improved performance compared to the first network was achieved whilst using a smaller network, these results are not as disappointing as they might initially appear. Both the fscore and the recall are above baseline in the best case, and recall is above baseline in all cases. There is also scope for improving on these results.

The differences in performance between Net 1 and Net 2 on the development data may simply be due to noise from different weight initialisations. A more thorough experiment would employ averages from several training runs to make allowances for this.

development	Precision	Recall	Fscore
Net 1	55.19%	46.35%	50.38
Net 2	54.74%	45.37%	49.62
Net 3	59.85%	55.56%	**57.62**

test	Precision	Recall	Fscore
Net 1	51.92%	40.59%	45.56
Net 2	52.92%	40.08%	45.61
Net 3	55.81%	45.99%	**50.42**

Table 2: Results of testing the networks on the development and testing data for **Part 3** of the shared task **ONLY**. The best fscores are highlighted in **bold**.

The scope for improving on these results comes from the possibilities of training with more data, training for more iterations and possible further improvements in training times through different ways of presenting the task to the networks.

However it is clear there is some way to go before these networks reach a good level of performance. Future work will look at improving the performance via increasing the amount of training data, using the difference between the current word vector and the previous word vector as an extra input to the network (which thus explicitly provides the transitions from one word to the next) and possibly employing a different processing strategy to the current 2-pass strategy.

Acknowledgements

The author would like to thank Ronan Reilly and Fred Cummins for their comments and advice on this work and for Fred's LSTM code. This work was funded by the EU TMR project "Learning Computational Grammars".

References

S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9:1735–1780.

J. Zavrel and J. Veenstra. 1996. The language environment and syntactic word class acquisition. In Koster C. and Wijnen F., editors, *Proceedings of the Groningen Assembly on Language Acquisition (GALA '95)*.