Some fancy title: followed by some more text

Alyssa P. Hacker [1] Ben Bitdiddle [2] Lem E. Tweakit [2]

[1]Some Institute [2]Another Institute

Introduction

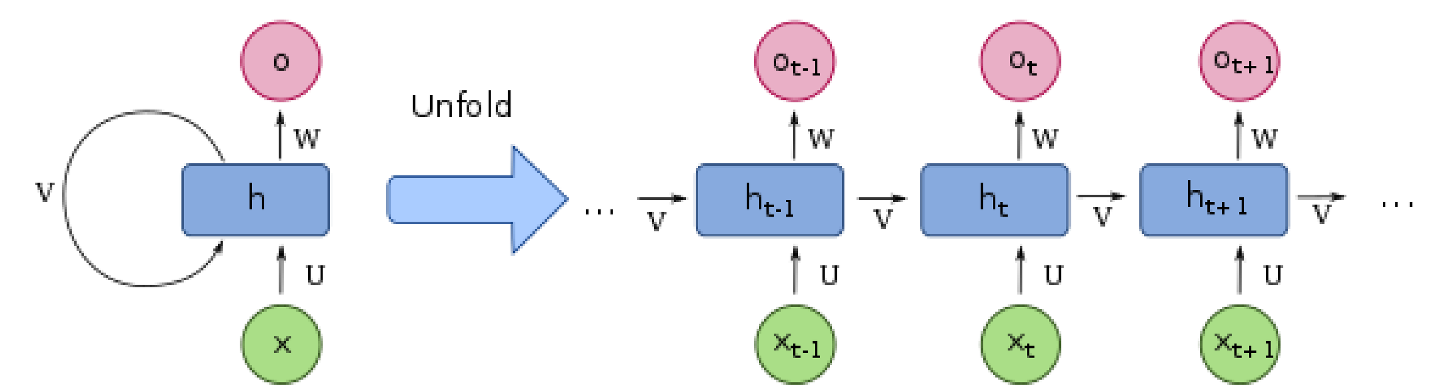
Hidden Markov Models used to be the go-to probabilistic tool to model sequential data; the Markov assumption however proves to often be unreasonably strong. Adding links to form higher order chains is not a scalable solution as the computationnal complexity grows exponentially in the order of the cain Recurrent Neural Networks (RNN) constitute a family of neural network architectures specialized to process sequential data which can forfeit Markovian assumptions (the cost is the lost of a simple probabilistic interpretation).

They do so by leveraging the simple idea of sharing parameters across the model [1]. RNNs have been used in diverse domains for generating sequences such as music and text [2].

Naive RNNs designs are unable to store information about far past inputs. The solution to this problem seems to be a better memory and especially a long-term memory for the network. Long Short-Term Memory (LSTM) are RNN designed to solve this problem.

RNN

Description of RNN



$$h_t = \sigma(Ux_t + Vh_{(t-1)} + b_h),$$
$$o_t = \text{softmax}(Wh_t + b_o)$$

Where U, V and W are weights matrix and the vectors b are bias parameters.

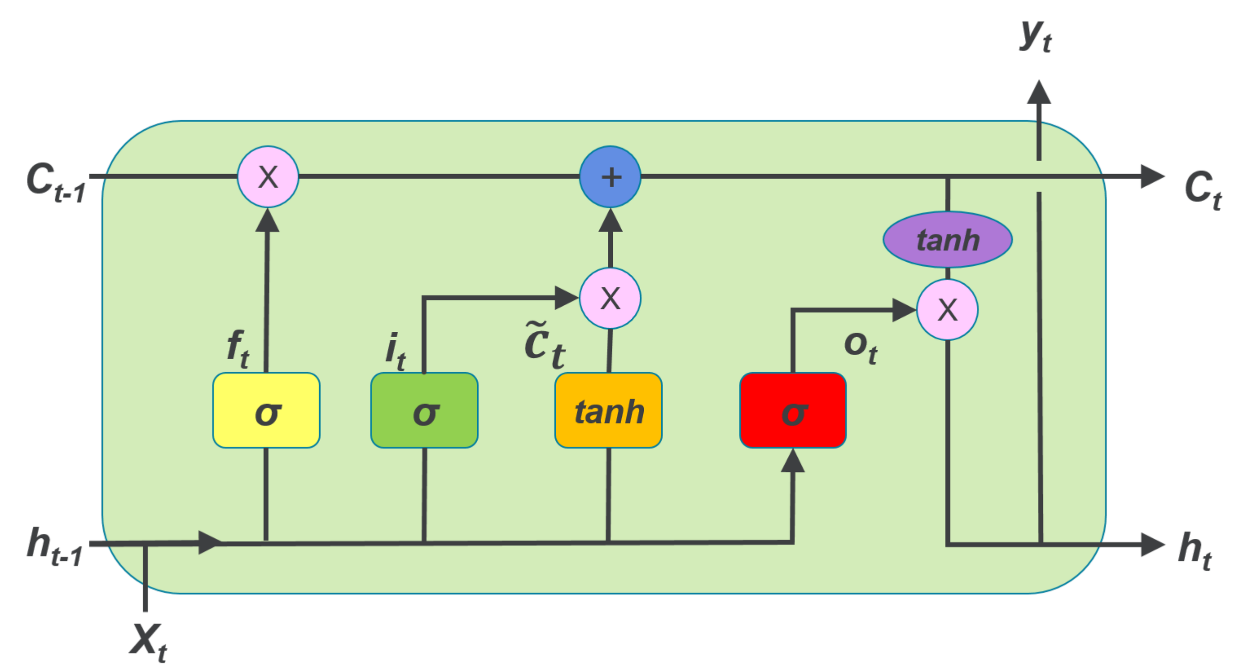
One of the major drawbacks of naive RNN is the numerical instability of the gradient.Consider the gradient with respect to h_t of $o_{t+\delta}$. How does it vary with δ? Following the graph above and applying the chain rule we can see that

$$\nabla_{h_t} o_{t+\delta} = \left(\prod_{k=t+1}^{t+\delta} V^T \text{diag}(1 - h_k^2) \right) \nabla_{h_{t+\delta}} o_{t+\delta}.$$

Thus, as δ grows, the gradient grows exponentially with V. If V is small or large than the gradient will either vanish or explode. This problem is well known. Solutions exist, which brings us to present the LSTM.

LSTM: architecture and its application to sequence generation

The LSTM model has been introduced primarily to solve the vanishing and exploding gradients problem. This model is a RNN in which we replaced every hidden nodes by a *memory cell*.



Intuitively, RNN have *long-term memory* in the form of matrix weights, they change during the training encoding general knowledge about the data. RNN also have *short-term memory* in the form of activation passing from each node to successive ones. The memory cell introduced in the LSTM model provides storage for those memories. We now describe components of the cell following.[3]

- Gates (f_t, i_t, o_t): They are sigmoidal units that takes activation from the input x_t and the output of the hidden layer from previous state h_{t-1}. Note that f_t multiply the value of the previous cell c_{t-1}. The term *gate* stands for the literal meaning in the sense that if f_t is close to 0, then the gate is *closed* and the flow from the previous cell is cut off. If f_t is closed to 1 then all flow is passed through. The output to the hidden layer is $h_t = o_t \odot \tanh(c_t)$ where $\odot$ denote the pointwise multiplication.
- Cell state ($c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$): Cell state maintains information on the input. Also refered as the internal state, c_t has a self-connected edges with a fixed unit weight. This constant weight implies that the error can flow across time without vanishing or exploding.

Our implementation

For the implementation section of this project, we gathered text data that we found around the internet to train a sequence classifier and generate text sequences. The training of the classifier consist of identifying from which corpus between Harry Potter, Lord of the rings, some random quotes and Shakespeare, the sequence corresponds to. Further to this, we trained one model per sequence type for the text generation. Finally, we verified that our generated sequences were well classified by our classifier. The parameters we used for the models are shown in the table below.

Before doing any sort or training, we had to do a bit of preprocessing on the data. First, we tokenized each corpus in sequences of 50 tokens. Then, we removed every capital letters to standardize the text. We built up a dictionary of every tokens ($\approx 60k$) in the datasets, this is our input space. Next, encode each token in a 256 dimensions vector with en embedding layer. We then feed the encoded vectors to our LSTMs with a hidden/cell state of 512 dimensions. The output dimension is 4 for the classifier and $\approx 60k$ for the text genetation.

For the generation of sequences, we initialize our LSTM with a random word drawn from our dictionary. Next, we feed back the prediction in the network until we reach the desired sequence length. We generated 1000 sequences (250 per models) and used our classifier on them. A few examples of generated sequences are shown above. With no surprise, we scored 98% on the classification task, which confirms that our sequences are at least probable.

Results

For the training of our classifier, we used the many-to-one architecture . We used the last output of the LSTM as our input for the classifier, disregarding all the other outputs. The last hidden state contains information about about all the sequence through the memory cell.

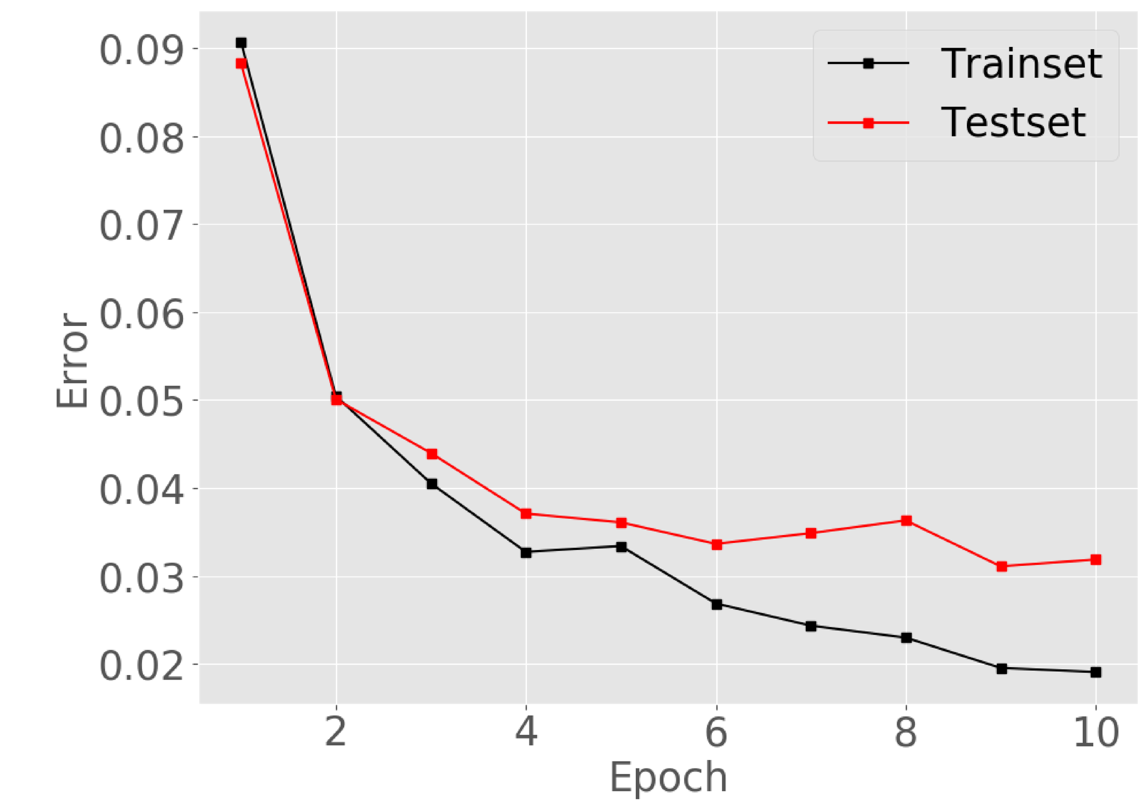


Figure: Classification error on train/test set.

Training trough minimization of the cross-entropy loss, which is equivalent to *perplexity*. The standard metric for language modeling[2].

Dataset	BPC	Perplexity
Harry Potter	1.00	33
LOTR	1.02	35
Random quotes	1.10	45
Shakespeare	0.94	26

- “ well , we ’ ll do it with a wand , ” said hermione . “ really ? ” said harry , looking at each other .
- what looked about this way , the black citadel , was far from the darkness , the ring was heard , but the sea big was big , and a great ring was in his battle .
- failure is a beginning of love and a family which comes from god .
- ” that now my mind shall screens his music , ” ” and i to give thee my vulgar heaven , ” ” i am your brother .

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618, 9780262035613.

[2] Alex Graves. “Generating Sequences With Recurrent Neural Networks”. In: *CoRR* abs/1308.0850 (2013). arXiv: `1308.0850`. URL: `http://arxiv.org/abs/1308.0850`.

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