**Introduction**

The goal of statistical language modelling (SLM) is to define a probability distribution over a sequence of words (Zhai & Massung, 2016, p. 50) or phonemes/characters. Statistical language models (SLMs) are of central importance to any natural language application involving prediction, for example automatic speech recognition, machine translation, and text generation. Perhaps the best-known model is the *N*-gram, but the largest progress in SLM in recent years can be attributed to artificial neural networks (ANNs).

Specifically, recurrent neural networks (RNNs) are at the heart of this progress. Because their output is not only fed forward, as it is in traditional ANNs, but can be looped back onto itself, RNNs have a kind of ‘memory’ that traditional ANNs lack. This is crucial to SLM, because the next word in a sentence usually depends on the words that came before it. The superiority of RNNs over *N*-gram models was demonstrated by Mikolov, Karafiát, Burget, Černocký and Khudanpur (2010), whose best mixture of three RNN models achieved a reduction of 50% in perplexity as compared to a state-of-the-art 5-gram back-off model.

However, conventional RNNs are difficult to train, and their memory capacity only suffices for short-term dependencies (Bengio, Simard & Frasconi, 1994). For example, consider this sentence: “We visited Berlin, the capital of Germany”. Here, to predict “Germany”, it is essential to still remember “Berlin”. Should the distance between “Berlin” and “Germany” increase, however, then it becomes more difficult for conventional RNNs to make the right prediction. A crucial innovation in this respect was the invention of long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997): a specific type of RNN in which each node consists of four interacting layers. Within a node, three gates regulate which information is let through, and which is forgotten. In 2012, Sundermeyer, Schlüter and Ney showed that LSTM networks yielded a perplexity about 8% lower than conventional RNNs.

Despite being state-of-the-art, text generated by LSTM RNNs can currently still fairly easily be distinguished from human-generated text, usually because the computer-generated texts are not very coherent. For example, Karpathy (2015) trained a 3-layer LSTM RNN with 512 nodes per layer on a corpus of all of Shakespeare’s work, with characters as input. This is a fragment of the text that was generated:

*PANDARUS:*

*Alas, I think he shall be come approached and the day*

*When little srain would be attain'd into being never fed,*

*And who is but a chain and subjects of his death,*

*I should not sleep.*

Thus, the quest to define an artificial SLM that has the same language capacity as humans do, is still ongoing. Greff, Srivastava, Koutník, Steunebrink and Schmidhuber (2015) compared a ‘vanilla’ LSTM architecture (as originally proposed in Graves & Schmidhuber, 2005) to eight possible modifications (for example, leaving out one of the gates). They found that none of the modifications significantly improved the vanilla architecture with regard to speech recognition, handwriting recognition, and polyphonic music modelling.

In addition to a model’s architecture, another domain of potential improvement are a model’s hyperparameters. These include the number of hidden layers and the number of nodes within each layer. Sak, Senior and Beaufays (2014) explored different numbers of hidden layers (1, 2, 5, 7) for LSTM RNN models, while keeping the total number of parameters constant at 13 million. The best performance, in terms of word error rate in speech recognition, was found for the model with five hidden layers. For this model, increasing the number of nodes per layer from 440 to 840 (now for a total of 37 million parameters) did not affect performance.

The aim of the current study is to build upon the work of Sak et al. (2014), and explore various hyperparameter settings in order to build the best-possible SLM, this time for generating text rather than speech recognition. Specifically, our models will be trained on song lyrics in three genres: country, metal and pop. Model goodness will be operationalised in terms of the categorical cross-entropy that the model achieves during training (objective), and the quality of a lyric generated by this model (subjective). Based on Greff et al. (2015), we will be using Graves and Schmidhuber’s (2005) vanilla LSTM RNNs [did we do this indeed?]. The following four hyperparameters will be explored:

1. Vocabulary size [or not?]

In the context of RNNs, the vocabulary size concerns the number of input and output nodes. Seemingly trivial, a decision to both include upper- and lower-case letters implies doubling the number of nodes, making correct classification more difficult. We will compare the goodness of a model trained on the full vocabulary of the training set (uppercase letters, lowercase letters, numbers, punctuation) and on a reduced vocabulary (lowercase letters, white space and two tokens marking the beginning and end of a lyric).

1. Number of layers (1, 2, 3)

Using deeper RNNs, in other words RNNs with more hidden layers, has been found to benefit the generalisation of training to test data (Sak et al., 2014), although it may make training harder and convergence slower. To this end, we will compare networks with 1, 2 and 3 hidden layers.

1. Number of nodes (256, 512)

Although Sak et al. (2014) did not find an effect of increasing the number of nodes per hidden layer from 440 to 840, Greff et al. (2015) detected with a random search that this hyperparameter explained 10% of the variance in their speech recognition task (although it was only 2% in handwriting recognition). This does suggest that the number of nodes is worth exploring further. We will compare hidden layers with 256 and 512 nodes.

1. Drop-out rate (20%, 50%)

Like the number of layers, another hyperparameter concerning generalisation from training to test data is the drop-out rate. At the presentation of each training case, a pre-defined percentage of randomly selected nodes is dropped from the model, thus preventing “complex co-adaptations” between several neurons (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012). Hinton et al. (2012) found that a drop-out rate of 50% resulted in big improvements on a variety of speech and image recognition tasks in feedforward ANNs. We will explore the effects of drop-out rates of 20% and 50% in our LSTM RNN for song lyric modelling.

Ideas for the discussion

We didn’t explore learning rate, although Greff et al. (2015) found that to be the hyperparameter that explains most variance (67% in speech recognition and 89% in handwriting recognition). What were our learning rate settings?

Bergstra & Bengio (2012): random search might be better than manual search (which is what we did). “Grid search and manual search are the most widely used strategies for hyper-parameter optimization. This paper shows empirically and theoretically that randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid.”

Vocabulary size should have been systematically manipulated (30 for all genres).

It would be interesting to explore zero drop-out.

References

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6), 602–610. http://doi.org/10.1016/j.neunet.2005.06.042

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2015). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*. http://doi.org/10.1109/TNNLS.2016.2582924

Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2012). *Improving neural networks by preventing co-adaptation of feature detectors*. Retrieved from http://arxiv.org/abs/1207.0580

Hochreiter, S., & Schmidhuber, J. J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780. http://doi.org/10.1162/neco.1997.9.8.1735

Karpathy, A. (2015). *The Unreasonable Effectiveness of Recurrent Neural Networks*. Accessed at <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>, 07-06-2017.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech* (pp. 1045–1048).

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Interspeech* (pp. 338–342). http://doi.org/arXiv:1402.1128

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. *Interspeech*, 194–197.

Zhai, C. X., & Massung, S. (2016). *Text Data Management and Analysis*. New York, NY: Morgan & Claypool Publishers.