Report: extension
ELEC-E5510 - Speech Recognition

Aditya Surikuchi
Student No: 662862

Outline

Premise

This was taken up in extension to *Neural network language models for ASR*[1]. Following were the tasks:

1. Build simple feed-forward-neural-network based language models
2. Analyse and compare with recurrent neural networks used in the baseline project
3. Apply bi-directionality on recurrent neural networks and compare results

Data

English gigaword dataset was used. These are archives of newswire text data and was used in the baseline publication[2] provided by the course personnel. Preliminary data stats:

	number of docs	number of tokens	dictionary size
training split	300000	8658023	59035
validation split	12000	341317	17727

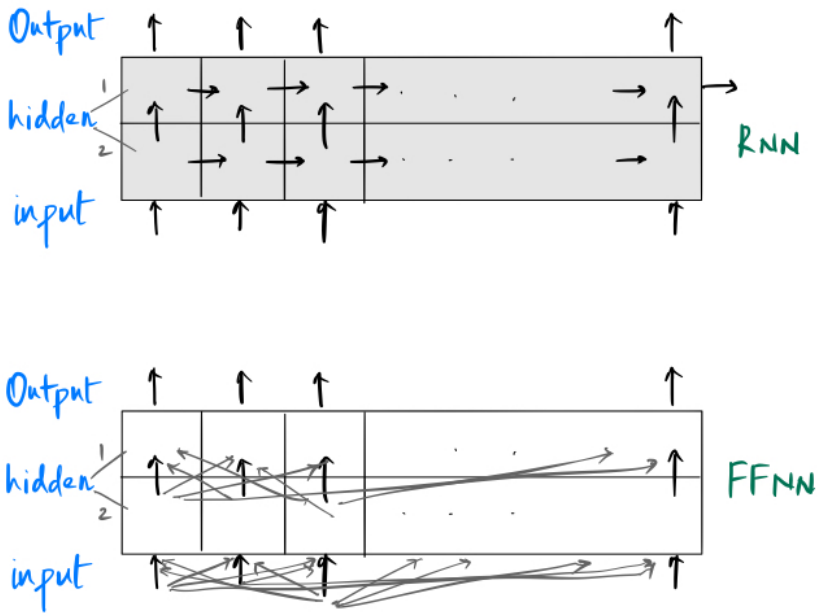
Complete exploratory data analysis available in this notebook.

Experiments

FFNN Task

Recurrent neural networks (including all its variants: LSTM, GRU) are stateful models; meaning their neurons share/propagate/influence consecutive neighbours through memory sharing

(context). On the other hand a simple feed-forward neural network has an architecture that has connections which are fed-forward to neurons in other (typically next) layers rather than ones in the same layer. This also means that their backpropagation mechanisms, training times and thereby their performances vary. Rudimentary difference:



As a part of the course project, I implemented a language-modelling-framework based on recurrent neural networks. To extend/justify the project, implemented a feed-forward-network with similar structure; without varying the number of hidden-layers, embedding size and maintaining the same implementation details (`epochs, vocabulary threshold, learning-rate, same clipping norm value, dropout and batch-size`). The following were taken into account:

- Input to the network (one-hot embedded on vocabulary)
- `Categorical cross-entropy` was used for loss calculation
- Different batch-sizes were used during validation phase to account for smaller data
- Models were saved between epochs based on minimum validation loss criteria

Bi-directionality Task

In regular recurrent neural networks, context is passed from previous hidden states to the current state. Though English is a left-context based language, making the neural networks bidirectional might help the language model predict the next words better.

RNNs (and it's variations can be made bidirectional by making use of `bidirectional=True` flag in PyTorch[3])

Results

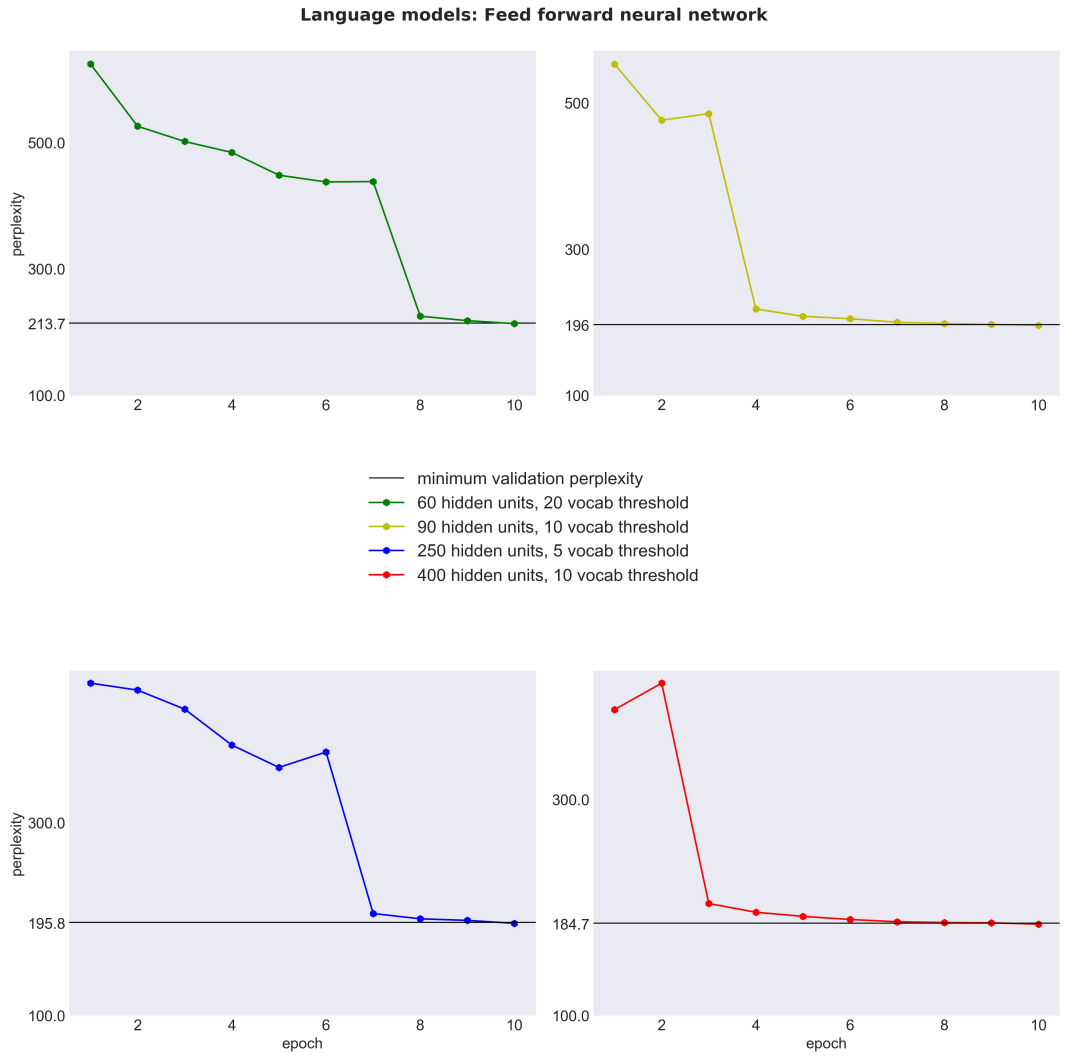
Perplexity which is an intrinsic evaluation technique serves well for model comparison. A well-learned language model should generalize well to unseen text data; held-out/validation dataset.

It is basically the inverse probability of text sequence normalized by number of tokens. *Lower perplexity, better the model:*

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}}$$

FFNN Task

Perplexity of feed-forward neural network models on validation dataset - `figure below`



Bi-directionality Task

Perplexity of bidirectional recurrent neural network models on validation dataset - `figure below`

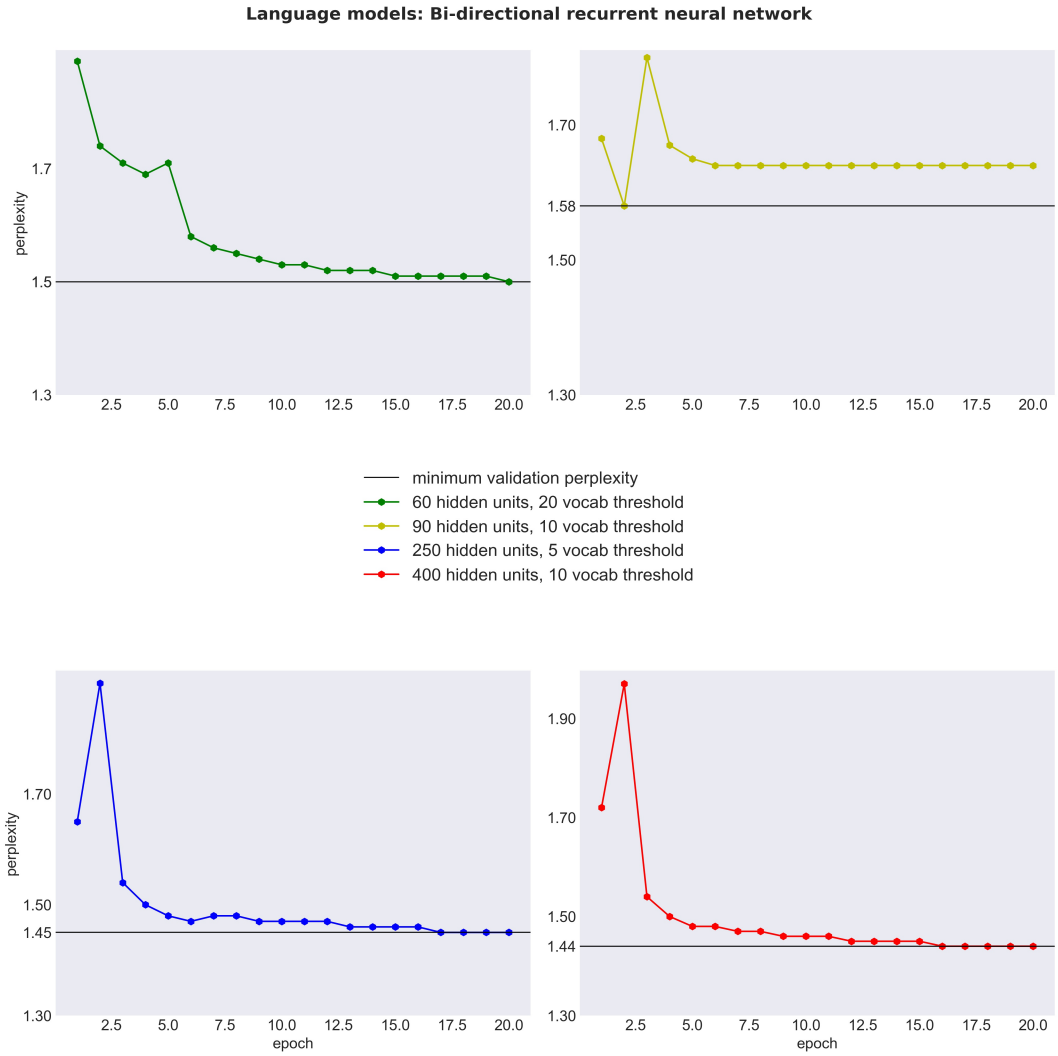


Figure 1:

Discussion

- Perplexities for FFNNs were way higher ($\sim 3X$) compared to those reported for RNNs
- The training & validations time taken was less compared to RNNs; though not dramatically (7 hours for RNNs, 5 hours for FFNNs)
- Because context is not shared explicitly in FFNNs, after certain epochs, models overfit to most occurring words in the vocabulary, yielding the same probability distributions
- Though making the networks bi-directional clearly shows lower perplexities compared to the one's without, the models are an overfit to the overall vocabulary structure (right

from the initial epochs). Nonetheless, this cannot be affirmatively demonstrated without making use of the trained language model in a specific usecase (ex: machine translation, image captioning), but it can be inferred from the drastic difference in perplexity plots

- The bidirectional models take drastically more time ($\sim 3X$) compared to the one's without

Appendix

https://github.com/akskuchi/speech-recognition/

References

[1] Neural network language models for ASR

[2] Mikolov, T. et al (2010)

[3] PyTorch