School of Computing and Information Systems
The University of Melbourne
COMP90042
NATURAL LANGUAGE PROCESSING (Semester 1, 2020)

Workshop exercises: Week 5

Discussion

1. How does a neural network language model (feedforward or recurrent) handle a large vocabulary, and how does it deal with sparsity (i.e. unseen sequences of words)?
2. Why do we say most parameters of a feedforward neural network language model is in their input and output word embeddings?
3. What advantage does an RNN language model have over N-gram language model?
4. What is the vanishing gradient problem in RNN, and what causes it? How do we tackle vanishing gradient for RNN?

Programming

1. In the iPython notebook `07-deep-learning`:
   - Can you find other word pairs that have low/high similarity? Try to look at more nouns and verbs, and see if you can find similarity values that are counter-intuitive.
   - We can give the neural models more learning capacity if we increase the dimension of word embeddings or hidden layer. Try it out and see if it gives a better performance. One thing that we need to be careful when we increase the number of model parameters is that it has a greater tendency to "overfit". We can tackle this by introducing dropout to the layers (`keras.layers.Dropout`), which essentially set random units to zero during training. Give this a try, and see if it helps reduce overfitting.
   - Improve the bag-of-words feed-forward model with more features, e.g. bag-of-N-grams, polarity words (based on a lexicon), occurrence of certain symbols (!).
   - Can you incorporate these additional features to a recurrent model? How?

Get ahead

- While `keras` is a great library for learning how to build basic deep learning models, it is often not as flexible as `pytorch`, due to its high level of abstraction. Follow the pytorch tutorial (`https://pytorch.org/tutorials/`) and learn how to build a word level language model in one of its examples (`https://github.com/pytorch/examples/tree/master/word_language_model`).