Ronak Mehta **CSE 517: Natural Language Processing**
Instructor: Noah A. Smith Winter 2021
Assignment A3 (coding) **Due:** February 5, 2021

For each of the parts, I used the 50-dimensional GloVe embeddings pretrained on the Wikipedia 2014 and Gigaword 5 datasets [Pennington et al., 2014]. "Similarity" refers to cosine similarity.

Part 1: Most Similar Word

Here, we retrieve the most similar word to a query word in the embedding space.

Query Word	Most Similar Word	Similarity
dog	cat	0.922
whale	whales	0.899
before	after	0.951
however	although	0.980
fabricate	fabricating	0.759

Part 2: Analogy Completion

In this part, I removed any word from the list that was included in original list, and applied lower case to the proper nouns. "First", "Second", and "Third" denote the top three candidate words in that order.

___ is to	___, as	___ is to	First	Similarity	Second	Similarity	Third	Similarity
dog	puppy	cat	puppies	0.763	scaredy	0.744	kitten	0.741
speak	speaker	sing	sang	0.623	nateq	0.622	lyricist	0.602
france	french	england	scottish	0.868	english	0.837	welsh	0.806
france	wine	england	orachard	0.662	tasting	0.633	tea	0.616

Part 3: Sentiment Classification

To build a sentiment classification model, I used the Large Movie Review Dataset, as in the TA example [Maas et al., 2011]. I did not truncate any of the sequences, and was left with $23,000$ training, 2000 dev, and $25,000$ test instances, respectively. The vocabulary size contained $30,595$ types, of which $18,741$ were found in the GloVe vocabulary and had associated pretrained word embeddings.

The network architecture consisted of embedding layer taking the $30,595$ dimensional vocabulary to 50-dimensional word vectors, followed by 2 bi-directional GRU-based RNN layers with 128-hidden dimensions each, and a final linear layer which consumed the hidden states of the two layers of each direction, ($2 \cdot 2 \cdot 128 = 512$ units) and outputed a softmax probability distribution on the binary outputs for positive and negative sentiment.

The training hyperparameters included a learning rate 0.001 for an Adam optimizer, 64 for the batch size, and 3 epochs of training before accuracy generally saturated.

Finally, the code for the network and optimization was heavily dependent on the sentiment classification example given be the TA's which can be found here.

Part 4: Results

When freezing the columns of the embedding matrix for words seen in the GloVe vocabulary, the results are as follows. The macro F1 calculation is done via `scikit-learn` [Pedregosa et al., 2011].

Test Accuracy	Test Macro F1 Score	Training Time
0.859	0.850	∼3 minutes

Part 5: Fine-tuning

In this setting, we let the pretrained embeddings be updated with the rest of the embedding weights, and see a performance increase.

Test Accuracy	Test Macro F1 Score	Training Time
0.884	0.878	∼3 minutes

References

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.