Assignment 2

CSE 517: Natural Language Processing – University of Washington

Winter 2020 – Due: February 2, 2020, 11:59 pm

From the Textbook

Complete these exercises from the Eisenstein book. They are ordered roughly the same way the content is ordered in lectures.

- Language modeling: 6.2 (p. 135). You will probably find this question easier if you solve 6.1 first.
- Word vectors: 14.4, 14.5, 14.6, and 14.7 (pp. 331–2). In problem 14.4, "phrase" refers simply to a (typically) short sequence of words. In problem 14.5, you shouldn't need more than four dimensions. Hint: Can you relate 14.4 and 14.5 to the XOR problem we solved in exercise 3.4?
- Extra credit: 14.2 (p. 331). This is a challenging problem, and we may not cover the skip-gram model in much detail in the lecture. Reading the chapter carefully, along with Levy and Goldberg [1], will help.

Implementation: LSTM Language Model

1. Download `ptb_lm_small.zip` from the course folder. Use train/dev/test as your training, development, and test data respectively. Build the vocabulary using the training data. What is your vocabulary size?
2. Using the PyTorch library,[1] (or whichever auto-differentiation toolkit you prefer) implement an LSTM language model. How do you divide the training data into mini batches? Also, describe your model's architecture, e.g., how many layers do you use, what are the dimensionalties of the word embeddings and LSTMs.
3. Train your language model on the training data of ptb_lm_small. After each epoch of training, compute its perplexity on the development data. Stop training when the perplexity stops improving. Plot the development perplexity against # of epochs.
4. Compute and report the perplexity on test data.

Useful resources:

1. LSTM tutorial: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[1] https://pytorch.org

2. PyTorch tutorial: https://pytorch.org/docs/stable/index.html

3. Tutorial on RNN language model: https://medium.com/the-artificial-impostor/notes-neural-language-model-with-pytorch-a8369ba80a5c

References

[1] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NeurIPS*, 2014.