Assignment 3

CSE 447 and 517: Natural Language Processing - University of Washington

Winter 2022

Please consult the course website for current information on the due date, the late policy, and any data you need to download for this assignment. This assignment is designed to advance your understanding of language models. ★ **problems are for CSE 517 students only.** Other problems should be completed by everyone.

Data: The data you need for this assignment is available at `https://nasmith.github.io/NLP-winter22/assets/data/A3.tgz`.

Submit: You will submit your writeup (a pdf) via Gradescope. Instructions can be found here. You will not submit code for this assignment.

1 n-gram Language Model

Your final writeup for this problem should be no more than three pages long.

1.1 Dataset

We provide you with three data files (a subset of the One Billion Word Language Modeling Benchmark). Each line in each file contains a whitespace-tokenized sentence.

- `1b_benchmark.train.tokens`: data for training your language models.
- `1b_benchmark.dev.tokens`: data for debugging and choosing the best hyperparameters.
- `1b_benchmark.test.tokens`: data for evaluating your language models.

A word of caution: You will primarily use the development/validation dataset as the previously unseen data while (i) developing and testing your code, (ii) trying out different model and training design decisions, (iii) tuning the hyperparameters, and (iv) performing error analysis (not applicable in this assignment, but a key portion of future ones). For scientific integrity, it is extremely important that you use the test data only once, just before you report all the final results. Otherwise, you will start overfitting on the test set indirectly. Please don't be tempted to run the same experiment more than once on the test data.

1.2 n-gram Language Modeling

You will build and evaluate unigram, bigram, and trigram language models. To handle out-of-vocabulary (OOV) words, convert tokens that occur **less than three times in the training data** into a special UNK token during training. If you did this correctly, your language model's vocabulary (including the UNK token and STOP, but excluding START) should have 26,602 types.

Your submission will not be evaluated for efficiency (you're not turning in source code), but we recommend keeping such issues in mind to better streamline the experiments.

Deliverables In the writeup, be sure to fully describe your models and experimental procedure. Provide graphs, tables, charts or other summary evidence to support any claims you make.

- Report the perplexity scores of the unigram, bigram, and trigram language models for your training, development, and test sets. Briefly discuss the experimental results.

1.3 Smoothing

To make your language model work better, you will implement linear interpolation smoothing between unigram, bigram, and trigram models:

$$\theta'_{x_j|x_{j-2},x_{j-1}} = \lambda_1 \theta_{x_j} + \lambda_2 \theta_{x_j|x_{j-1}} + \lambda_3 \theta_{x_j|x_{j-2},x_{j-1}}$$

where θ' represents the smoothed parameters, and the hyperparameters $\lambda_1, \lambda_2, \lambda_3$ are weights on the unigram, bigram, and trigram language models, respectively. $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

You should use the development data to choose the best values for the hyperparameters. Hyperparameter optimization is an active area of research; for this homework, you can simply try a few combinations to find reasonable values.

Deliverables In the writeup, be sure to fully describe your models and experimental procedure. Provide graphs, tables, charts or other summary evidence to support any claims you make.

1. Report perplexity scores on training and development sets for various values of $\lambda_1, \lambda_2, \lambda_3$. Report no more than 5 different sets of λ's. In addition to this, report the training and development perplexity for the values $\lambda_1 = 0.1, \lambda_2 = 0.3, \lambda_3 = 0.6$.
2. Putting it all together, report perplexity on the test set, using the hyperparameters that you chose from the development set. Specify those hyperparameters.
3. If you use half of the training data, would it increase or decrease the perplexity on previously unseen data? Why? Provide empirical experimental evidence if necessary.
4. If you convert all tokens that appeared less than 5 times to `<unk>` (a special symbol for out-of-vocabulary tokens), would it increase or decrease the perplexity on the previously unseen data compared to an approach that converts only a fraction of words that appeared just once to `<unk>`? Why? Provide empirical experimental evidence if necessary.

2 ★ Neural Language Modeling, based on Eisenstein 6.10 (p. 136)

Using the PyTorch library, train a neural language model of your choice[1] from the WikiText-2 training corpus. After each epoch of training, compute its perplexity on the WikiText-2 validation corpus. Stop training when the perplexity stops improving.

Deliverables

1. Fully describe your model architecture, hyperparameters, and experimental procedure.
2. After each epoch of training, compute your LM's perplexity on the development data. Plot the development perplexity against # of epochs. Additionally, compute and report the perplexity on test data.
3. Compare experimental results such as perplexity and training time between your n-gram and neural models (both on the Wikitext-2 corpus, for fair comparison). Provide graphs that demonstrate your results.

3 What Can Language Models Do?

Read Bender and Koller [1] and answer one of these two questions in a short paragraph.

- If you are more or less convinced by the paper's argument, tell us how the paper's conclusions affect your thinking about research and/or practice of NLP. What do you believe about NLP now that you didn't before reading the paper?
- If you are unconvinced by the paper's argument, tell us where you think the argument was weakest, and try to summarize your objection.

References

[1] Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proc. of ACL*, 2020. URL `https://aclanthology.org/2020.acl-main.463`.

[1] The course staff recommend vanilla RNN, LSTM, or GRU architecture.