Assignment 1
CSE 447/547M: Natural Language Processing

University of Washington

Due: January 24, 2019, 6pm

Your final writeup for this assignment, including the problem and the report of the experimental findings of the programming part, should be no more than four pages long. You should submit it as a PDF. We *strongly* recommend typesetting your scientific writing using LaTeX. Some free tools that might help: TexStudio (Windows), MacTex (Mac), TexMaker (cross-platform), and Overleaf (online).

Dataset

For this assignment, you are provided with three data files (a subset of the One Billion Word Language Modeling Benchmark [2]). 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.

We've set up an informal language modeling leaderboard for this assignment. [1] **Your performance on the leaderboard (or whether you submit at all) does not affect your grade, it's just for fun / motivation.**

1 Programming: n-gram language modeling (35%)

In this programming problem, 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** 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 26602 unique tokens (types).

You can develop and run your code on the course server, `aziak.cs.washington.edu`. You will turn in your source code along with the PDF writeup. Your submission will not be evaluated for efficiency, but we recommend keeping such issues in mind to better streamline the experiments.

[1] submit at https://goo.gl/forms/5ukY922QQCj4mMON2, results at https://docs.google.com/spreadsheets/d/1uLwIpKUF3uGln93-JH3-94PhmON3oSijvcVn5Ge3Gu4/edit?usp=sharing

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

2 Programming: smoothing (35%)

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 do a "grid search", which means you will simply try a few combinations of reasonable values.[2]

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 then 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.

3 Experimentation: neural language modeling (30%)

In this final section, you will train a neural language model with AllenNLP [3]. AllenNLP will be used extensively in future assignments, and this exercise serves as an introduction.

To help you ensure that your neural language model is directly and fairly comparable with your n-gram language models, we've provided a starter configuration `a1_1b_benchmark_language_model.jsonnet`. You may also find the AllenNLP tutorials[3] helpful, particularly "A Walk Through AllenNLP"[4] and "Creating your Configuration File."[5] Training neural models is computationally expensive, especially without

[2] In practice, "random search" often (perhaps unintuitively) outperforms grid search. See Bergstra and Bengio [1] for more information.

[3] https://github.com/allenai/allennlp/blob/088f0bb/tutorials/README.md

[4] https://github.com/allenai/allennlp/blob/088f0bb/tutorials/README.md#a-walk-through-allennlp

[5] https://github.com/allenai/allennlp/blob/088f0bb/tutorials/how_to/create_a_configuration.md

specialized hardware (GPUs) — start early if you can! If you'd like to train your models faster, using GPUs will help. CSE Support maintains a limited number of GPU-enabled machines for student use.[6]

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 your neural language model's training, development, and test set perplexity. Discuss your experimental results, especially in comparison to your n-gram models.

4 Bonus: Character-level language models (+10%)

Much recent work in NLP operates at the *character level*, rather than the word level. For the bonus portion of this assignment, we'd like to build and evaluate character-level n-gram and neural language models.

Describe your models, report their performance, and discuss the possible benefits and drawbacks of using character-level models in place of word-level models. Can you directly compare your character-level models with your previously-implemented word-level models? Why or why not?

Submission Instructions

Submit a single gzipped tarfile (`A1.tgz`) on Canvas, with the following:

- **Code**: You will submit your code together with a neatly written README file to instruct how to run your code with different settings. We assume that you always follow good practice of coding (commenting, structuring), and these factors are not central to your grade. You may implement the language models in the programming language of your choice. However, please provide well commented code if you want partial credit. If you have multiple files, please provide a short description in the preamble of each file.
- **Report**: As noted above, your writeup should be four pages long, or less, in PDF (one-inch margins, reasonable font sizes). Part of the training we aim to give you in this class includes practice with technical writing. Organize your report as neatly as possible, and articulate your thoughts as clearly as possible. We prefer quality over quantity. Do not flood the report with tangential information such as low-level documentation of your code that belongs in code comments or the README. Similarly, when discussing the experimental results, do not copy and paste the entire system output directly to the report. Instead, create tables and figures to organize the experimental results.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[2] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. In *Proc. of INTERSPEECH*, 2014.

[3] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proc. of NLP-OSS*, 2018.

[6] https://www.cs.washington.edu/lab/facilities/instructional-gpu-resources