Assignment 2
CSE 143: Intro to Natural Language Processing

University of California Santa Cruz

First submission due date: May 10, 2021, **9:00pm**
Second submission due date: May 17, 2021, 11:59pm

This assignment is to be done in Python 3 without use of any NLP libraries (such as NLTK).

You may work in groups of up to four.

Your final writeup for this assignment, including the problem and the report of the experimental findings of the programming part. Most of your grade will depend on the quality of the report. You should submit it as a PDF. We *strongly* recommend typesetting your scientific writing using LaTeX. Some free tools that might help: Overleaf (online), TexLive (cross-platform), MacTex (Mac), and TexStudio (Windows).

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.

1 Programming: n-gram language modeling (25%)

In this part, you will build and evaluate unigram, bigram, and trigram language models. **For this part, use MLE (without any smoothing)**. 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 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.

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.

   To help debug your code, it may be helpful to evaluate on a file consisting of this line:

```
HDTV .
```

(Two tokens, with a space between them). If you use MLE, and you've implemented the models correctly, the unigram, bigram, and trigram perplexities will be 658, 63.7, 39.5, respectively. Some things to watch out for: M in the perplexity calculation is the total number of tokens in the file including <STOP> but not including <START>. For the probability of the token immediately following <START> in the trigram model, you use the bigram probability. So in this example, you use the bigram $p(\text{HDTV}|\ \text{<START>})$ in the trigram model for the probability of HDTV.

2 Programming: additive smoothing (25%)

Implement additive smoothing for the unigram, bigram, and trigram models.

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. For additive smoothing with $\alpha = 1$, report the perplexity scores of the unigram, bigram, and trigram language models for your training, and development sets.
2. Repeat for two other values of $\alpha > 0$ of your choosing.
3. Report the perplexity scores for best values of the hyperparameters for the unigram, bigram, and trigram language models on the test set. Remember to pick the best values of the hyperparameters for each model using the dev set, not the test set. Briefly discuss your results.

3 Programming: smoothing with linear interpolation (25%)

To make your language model work better, you will implement linear interpolation smoothing between the MLE 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 unsmoothed MLE unigram, bigram, and trigram language models, respectively. $\lambda_1 + \lambda_2 + \lambda_3 = 1$. **Do not use smoothing for the unigram, bigram, and trigram language models that you are interpolating.**

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 of reasonable values.[1]

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

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.

To help you debug your code, for $\lambda_1 = .1, \lambda_2 = .3, \lambda_3 = .6$ and the evaluation set

```
HDTV .
```

the perplexity should be 48.1.

Submission Instructions

Submit a zip file (`A2.zip`) on Canvas, containing the following:

- **Code**: You code should be implemented in Python 3, and needs to be runnable. 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. However, please provide well commented code if you want partial credit. If you have multiple files, provide a short description in the preamble of each file.
- **Report**: As noted above, your writeup should be in PDF. Include your name at the top of the report. 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.