Assignment 2
NLP 201: Natural Language Processing I

University of California Santa Cruz

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

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 (33%)

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: smoothing with linear interpolation (33%)

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]

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 $\lambda_1 = 0.3, \lambda_2 = 0.3, \lambda_3 = 0.4$.
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.

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

3 Experiments with GPT-3 (33%)

GPT-3 (Brown et al 2019 - Language Models are Few-Shot Learners) is a large language model trained on 45 TB of text data. Language modeling is one of the three supertasks in NLP, and large language models can do remarkable things. In this part of the homework, you will use the web interface for GPT-3 to experiment with what it can do. In particular, we will use **in-context learning**, where you prompt a language model some examples of a task, and hope that it can perform the task you wanted.

Deliverables

1. Sign up for an account to use GPT-3 at `https://beta.openai.com/signup`. You will be given some free credits to use the models. (It should be fine for our purposes, but if you use the API for larger experiments you may run out).
2. Login and click on examples, and **select "English to other languages."** Change the prompt the following and click "submit." (This is an example from the https://rajpurkar.github.io/SQuAD-explorer/ development set).

```
Read the passage and answer the question.

When the Hollis Professor of Divinity David Tappan died in 1803 and the
president of Harvard Joseph Willard died a year later, in 1804, a struggle
broke out over their replacements. Henry Ware was elected to the chair in 1805,
and the liberal Samuel Webber was appointed to the presidency of Harvard two
years later.

Question: Who succeeded Joseph Willard as president?

Answer:
```

   (The correct answer is Samuel Webber.)
   This is **zero-shot** prompting, because we haven't given it any examples of what we want it to do, we are just expecting it to already know the task. What answer do you get? Try some other questions and report your answers. How many does it get correct? Note you can delete text so that you can prompt GPT-3 again with new questions. You may not get the same answer each time since the generator samples according to the language model probability distribution. You can increase the variation in the samples by increasing the temperature under "settings."
3. Now we will try **few-shot** prompting, also known as **in-context learning**, where we give it some examples in the prompt. Change the prompt the following and click "submit."

```
Read the passage and answer the following questions.

In the late 17th century, Robert Boyle proved that air is necessary for
combustion. English chemist John Mayow (1641–1679) refined this work by showing
that fire requires only a part of air that he called spiritus nitroaereus or
just nitroaereus.
```

```
Question: Who proved that air is necessary for combustion?

Answer: Robert Boyle

Question: John Mayow died in what year?

Answer: 1679

Read the passage and answer the following questions.

When the Hollis Professor of Divinity David Tappan died in 1803 and the
president of Harvard Joseph Willard died a year later, in 1804, a struggle
broke out over their replacements. Henry Ware was elected to the chair in 1805,
and the liberal Samuel Webber was appointed to the presidency of Harvard two
years later.

Question: Who succeeded Joseph Willard as president?

Answer:
```

What answer do you get? Try some other passages and questions and report your answers. How many does it get correct?

4. The newer models, such as default model "text-davinci-002," are based on GPT-Instruct which has been further trained as described in the paper Ouyang et al 2022 - Training Language Models to Follow Instructions with Human Feedback. The original GPT-3 model is called "davinci," which can be selected in the Model drop-down box on the right. How does the original GPT-3 model compare to the newer text-davinci-002 model for zero-shot and few-shot prompting?
5. Try another task, such as summarization, sentiment analysis, or question answering in some other domain, etc. Try to come up with your own prompts for zero-shot and few-shot prompting, rather than using the provided examples. How well does it do the task?

Submission Instructions

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

- **Code**: Your 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.