1. **Derivation: MAP estimate as additive smoothing**

Solution

$\hat{\theta} = argmax(p(\theta \mid D))$

dropping term without θ

$= \text{argmax}(\prod_{j=1}^{k} \theta_j^{x_j} \prod_{j=1}^{k} \theta_j^{\alpha_j - 1})$

$= \text{argmax}(\prod_{j=1}^{k} \theta_j^{x_j + \alpha_j})$

taking ln

$\hat{\theta} = argmax(\ln(\prod_{j=1}^{k} \theta_j^{x_j + \alpha_j}))$

$\hat{\theta}= \text{argmax}(\sum_{j=1}^{k}(\text{x}_j + \alpha_j)\ \ln(\theta_j))$

we need to maximise the above term with the constraint using Lagrange $\sum_{j=1}^{k} \theta_j = 1$

$J(\theta, \lambda) = \sum_{j=1}^{k}(\text{x}_j + \alpha_j)\ \ln(\theta_j)$ - $\lambda(\sum_{j=1}^{k} \theta_j)$

We can search the stationary points of the Lagrangian, i.e pairs (θ, λ) satisfying $\Delta_\theta J(\theta, \lambda) = 0$ and $\Delta_\lambda J(\theta, \lambda) = 0$.

$\frac{\partial J(\theta,\lambda)}{\partial \lambda} = 0$

$$\sum_{j=1}^{k} \theta_j = 1 \tag{1}$$

and by $\frac{\partial J(\theta,\lambda)}{\partial \theta_j} = 0$

$$\frac{x_i + \alpha_i}{\theta_i} = \lambda, \forall i \in \{1, 2, .., k\} \tag{2}$$

using equation 1 and 2

$\sum_{j=1}^{k}(x_j + \alpha_j) = \lambda$

$\hat{\theta}_i = \frac{\sum_{i=1}^{N}(x_i)+\alpha_i}{\sum_{j=1}^{k}(x_j+\alpha_j)}$(*by putting* λ *in* 2)

$\hat{\theta}_i = \frac{count(x_i)+\alpha_i}{N+\sum_{j=1}^{k}(\alpha_j)}$(*where* $N = \sum_{j=1}^{k}(x_j)$)

2. **n-gram language modeling**

Solution

2.1 *Introduction*

In this part, we build and evaluate unigram, bigram, and trigram language models. The One Billion Word Language Modeling Benchmark dataset is used to train our n-gram models (n=1, 2, and 3). The perplexity of each model is determined on the training, validation, and testing subsets after the n-gram models have been trained.

2.2 *Test Preprocessing and Tokenization*

Every language model has a lexicon from which it takes its tokens.[**?**] If the vocabulary for English word-level language modeling contains 100K words, the model must decide which one (out of the 100K) to forecast as the next word given any starting collection of words.

For tokenization and vocabulary, split method was used and the vocabulary used was also limited to the provided dataset. We don't use any pretrained embeddings or tokenization functions such as spacy, nltk, torchtext etc.

We build a dictionary with each token's frequency of occurrence in order to handle Out-of-Vocabulary (OOV) terms. All tokens that appeared three or more times in the dictionary of token occurrences were grouped together into sets, giving us the full vocabulary of our model. Words that appear less than three times in the dataset and are not <START> or <STOP> signals are substituted with the symbol <UNK>.

2.3 *Model Implementation*

a *Unigram Model:*
The unigram model may be trained in two steps and is comparatively easy to build. The independence of each token's likelihood from prior tokens is what makes this model so straightforward. Counting the total number of tokens in the data as well as the instances of each unique token are the first steps. Then, using 'm' as the total number of words, we can obtain the log joint probability for perplexity by appending the likelihood of each word in the corpus. The model will yield the probability of the frequency of the supplied word divided by the entire number of words because we are not applying any smoothing techniques.

b *Bigram/Trigram Models:*
We will employ a two-step procedure to effectively train the bigram and trigram models. The first stage entails building a dictionary using the context, or the previous n-1 words, as its keys. Each of these contexts is mapped to a different dictionary, which contains key-value pairs made up of terms that appear in the training data after the context in question and their frequency. The next step is to use the counts from the previous phase to generate a comparable data structure that represents the likelihood that a word will appear in a specific context (Markov assumption).

$$P\left(w_n \mid w_{1:n-1}\right) \approx P\left(w_n \mid w_{n-N+1:n-1}\right) \tag{3}$$

We decide on the value of 'n' accordingly while deciding on the n-gram model used. Using these two procedures, we create a model that provides us with the likelihood that a word will appear in any given context while still being quick to train and use after training.

2.4 *Results*

In order to evaluate our model, we use maximum likelihood estimate to find the n-gram probabilities. We get the MLE estimate for the parameters of an n-gram model by getting counts from a corpus, and normalizing the counts so that they lie between 0 and 1. For the general case, MLE probability of an n-gram model is:

$$P\left(w_n \mid w_{n-N+1:n-1}\right) = \frac{C\left(w_{n-N+1:n-1} w_n\right)}{C\left(w_{n-N+1:n-1}\right)} \tag{4}$$

To check whether our model is performing as it is supposed to, we ran a debugging test over a 2-word sentence, provided to us in the assignment as our first checkpoint. We are given the sentence: [HDTV .] (2 tokens with a space between them). The resulting perplexities found using MLE are given in Table 1.

Unigram	Bigram	Trigram
648.044	63.707	39.478

Table 1: n-gram perplexities for [HDTV .]

Dataset	Unigram	Bigram	Trigram
Training	976.543	77.073	7.872
Dev	892.246	3443.04	2607451.9
Test	896.499	3436.48	2543381.8

Table 2: n-gram perplexities for Language Modelling Benchmark dataset

After reaching the checkpoint, we move to finding perplexities of training, dev and test sets. Now, for bigram and trigram models, getting perplexities would be a challenge for words having frequency less than 3, as they would give us a probability of 0 and in turn throw a domain error. To avoid this, we replace the 0 probabilities with an infinitely small value to help with our calculations and give us some number. Hence, the perplexities found in our analysis are given in Table 2.

As can be seen in the table, the bigram and trigram models both had exceptionally high scores for the development and testing set since they were given a word in a context they had never encountered before and were instructed to give it a probability of zero. Moving on we will try to use smoothing techniques to combat this issue in the following question.

3. **Smoothing with linear interpolation**

Solution

3.1 *Introduction*

After seeing the shortcomings of the n-gram models with unknown words, we will try to tweak up our configuration to overcome them. For the language model to work better, we will implement linear interpolation smoothing between the MLE unigram, bigram, and trigram models. As we already did implement our model in the previous problem, here we will talk about interpolation and using weighted probabilities to make our language model predict better.

3.2 *Applying Interpolation*

In simple linear interpolation, we combine different order n-grams by linearly interpolating them. Thus, we estimate the trigram probability $P(w_n \mid w_{n-2}w_{n-1})$ by mixing together the unigram, bigram, and trigram probabilities, each weighted by a λ:

$$\begin{aligned}\hat{P}(w_n \mid w_{n-2}w_{n-1}) =& \lambda_1 P(w_n) \\ &+ \lambda_2 P(w_n \mid w_{n-1}) \\ &+ \lambda_3 P(w_n \mid w_{n-2}w_{n-1})\end{aligned} \tag{5}$$

The λs must sum to 1.

Since we are applying three weights for this problem, we will only calculate the trigram probabilities with linear interpolation.

Hyperparameters	Training Perplexity	Development Perplexity
$\lambda1 = 0.1$ $\lambda2 = 0.3$ $\lambda3 = 0.6$	11.151	352.233
$\lambda1 = 0.3$ $\lambda2 = 0.3$ $\lambda3 = 0.4$	15.300	286.634
$\lambda1 = 0.1$ $\lambda2 = 0.1$ $\lambda3 = 0.8$	9.329	482.844
$\lambda1 = 0.2$ $\lambda2 = 0.4$ $\lambda3 = 0.4$	14.697	286.361
$\lambda1 = 0.33$ **$\lambda2 = 0.33$** **$\lambda3 = 0.34$**	**17.039**	**278.157**
$\lambda1 = 0.7$ $\lambda2 = 0.2$ $\lambda3 = 0.1$	42.013	322.426

Table 3: Interpolated Trigram's perplexity scores for different hyperparameter values

Hyperparameters	Testing Perplexity
$\lambda1 = 0.33$ $\lambda2 = 0.33$ $\lambda3 = 0.34$	277.802

Table 4: Interpolated Trigram's perplexity scores on test set.

3.3 *Experiments*

After we have deployed our model along with the interpolation function, we will move to perform required experiments with hyperparameters ($\lambda1$, $\lambda2$, $\lambda3$) and dataset. To begin with, we first check if our model is performing as its supposed to. We ran a debugging test over a two word sentence, provided to us in the assignment as our first checkpoint for the parameter configuration of $\lambda1 = 0.1$, $\lambda2 = 0.3$, $\lambda3 = 0.6$. We are given the sentence: [HDTV .] (2 tokens with a space between them). The resulting perplexity found using MLE for our interpolated trigram was 48.113. After reaching the checkpoint, we move to our experiments.

a *Experiment 1:* Our first experiment is finding perplexity scores on training and development sets for various values of $\lambda1$, $\lambda2$, $\lambda3$. The scores after trying 6 different combinations are given in Table 3 with the best score highlighted in bold.

b *Experiment 2:* After running through various iterations of hyperparameters, we saw that for the values $\lambda1 = 0.33$, $\lambda2 = 0.33$, $\lambda3 = 0.34$, we were getting the best perplexity score on the development set. We ran the same configurations against the test set, the results of which are given in Table 4.

c *Experiment 3:* For this experiment, we will use half of the training set for training the model with the best hyperparameter values we got in experiment 1. After training, the perplexity scores on unseen data are provided in table 5.

Hyperparameters	Dev Perplexity	Test Perplexity
$\lambda 1 = 0.33$ $\lambda 2 = 0.33$ $\lambda 3 = 0.34$	265.714	266.678

Table 5: Perplexity scores on unseen data, after using half of training set

Hyperparameters	Dev Perplexity	Test Perplexity
$\lambda 1 = 0.33$ $\lambda 2 = 0.33$ $\lambda 3 = 0.34$	233.650	233.381

Table 6: Perplexity scores on unseen data, after changing the frequency constraint

As we can see, after using half of the training set, scores were slightly improved. Though theoretically more training data gives us better scores, the aberration of our results can be explained by possible noise in our dataset. It could also be because of low contextual data available in our training set. Another reason could be that our model was initially overfitting and was able to avoid it with less data.

d *Experiment 4:* In this experiment, we convert all tokens that appeared less than 5 times to <UNK>, and find the perplexity score on unseen data with the same hyperparameter configuration, results of which are provided in Table 6.
Again we see a small improvement in our perplexity scores. Our theory behind this improvement remains the same as in the previous experiment since the dataset has remained unchanged. Perhaps on using less noisy data, or training with required measures to overcome overfitting, we could be able to come up with another result or a different reasoning that supports this drop of perplexity score.

4. **Experiments with GPT-3**

Solution

4.1 *Experiment 1:Perform zero-shot prompting*

Passage pasted on GPT3:

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:

After feeding this into the GPT3 model, we got the following passage as the output:

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:Samuel Webber.

4.2 *Experiment 2:Perform few-shot prompting*

Passage pasted on GPT3:

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 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:

After hitting submit, we got the similar answer as we did in experiment 4.1, which was 'Samuel Webber.'

To further test the efficiency of the model, we tried different passages from the following site:

https://www.superteacherworksheets.com/reading-comp/2nd-planetrip_TZZTZ.pdf

On running 10 iterations, GPT-3 got correct answers 8 out of 10 times.

4.3 *Experiment 3:Perform few-shot prompting with older versions*

So far we were working on the latest trained version of GPT-3, i.e, "text-davinci-002." Now, we will be using the pilot version, "davinci" on the same passage used in the last experiment and see how it performs.

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 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 predicted passage followed by the above:
Question: Who succeeded David Tappan as Hollis Professor of Divinity? Answer: Read the passage and answer the following questions. The first American to be elected to the Royal Society was Benjamin Franklin in 1756. The second was John Winthrop, Jr., in 1769. The third was Thomas Jefferson in 1780. Question: Who was the first American to be elected to the Royal Society? Answer: Benjamin Franklin Question: Who was

As we can see, the first version is clearly far off against the latest GPT-3. Not only were answers long, they were vague and carried a lower amount of context as compared to "text-davinci-002."

4.4 *Experiment 4:Summarization*

We ran multiple excerpts from books of famous 19th century authors, and tried to summarize them for 2nd graders. One of the excerpt from Karl Marx's Communist Manifesto is given below in Figure 1, followed by the summarization done by GPT-3.

A SPECTRE is haunting Europe--the spectre of Communism. All the Powers of old Europe have entered into a holy alliance to exorcise this spectre: Pope and Czar, Metternich and Guizot, French Radicals and German police-spies.

Where is the party in opposition that has not been decried as Communistic by its opponents in power? Where the Opposition that has not hurled back the branding reproach of Communism, against the more advanced opposition parties, as well as against its reactionary adversaries?

Two things result from this fact.

I. Communism is already acknowledged by all European Powers to be itself a Power.

II. It is high time that Communists should openly, in the face of the whole world, publish their views, their aims, their tendencies, and meet this nursery tale of the Spectre of Communism with a Manifesto of the party itself.

To this end, Communists of various nationalities have assembled in London, and sketched the following Manifesto, to be published in the English, French, German, Italian, Flemish and Danish languages.

Figure 1: Book Excerpt

Summarization of text in figure 1:
There is a party in Europe that is being accused of being Communist by all of the other parties. This party is fighting back by saying that all of the other parties are Communist. The party has decided to write a document called a Manifesto to explain their views to the world.

We also ran a handful of zero shot and few shot learning tasks, excerpts of which wouldn't be possible to add here. However, we were able to witness the power of GPT-3 and how

accurate it has been in providing correct answers, summarizations, and classification. Going further, we would look to explore more on GPT-3 and find its applications to use in upcoming Capstone project