1 Word-Level Neural Bi-gram Language Model

1.a

Cross-Entropy Loss

The cross-entropy loss is defined as:

$$CE(y, \hat{y}) = -\sum_i y_i \cdot \log(\hat{y}_i)$$

where $\hat{y}_i = \text{softmax}(\theta_i)$. Expanding $\hat{y}_i$ using the softmax function:

$$\hat{y}_i = \frac{\exp(\theta_i)}{\sum_j \exp(\theta_j)}$$

Substituting this into the cross-entropy loss:

$$CE(y, \hat{y}) = -\sum_i y_i \cdot \log\left(\frac{\exp(\theta_i)}{\sum_j \exp(\theta_j)}\right)$$

Simplify the logarithmic term:

$$CE(y, \hat{y}) = -\sum_i y_i \cdot \left[\log(\exp(\theta_i)) - \log\left(\sum_j \exp(\theta_j)\right)\right]$$

$$CE(y, \hat{y}) = -\sum_i y_i \cdot \theta_i + \log\left(\sum_j \exp(\theta_j)\right) \cdot \sum_i y_i$$

Since y is a one-hot vector, $\sum_i y_i = 1$, so:

$$CE(y, \hat{y}) = -\sum_i y_i \cdot \theta_i + \log\left(\sum_j \exp(\theta_j)\right)$$

Gradient of Cross-Entropy Loss with Respect to θ

To compute $\frac{\partial CE(y,\hat{y})}{\partial \theta_i}$, we take the derivative of the expression:

$$CE(y, \hat{y}) = -\sum_i y_i \cdot \theta_i + \log\left(\sum_j \exp(\theta_j)\right)$$

Step 1: Derivative of the First Term The derivative of the first term, $-\sum_i y_i \cdot \theta_i$, is:

$$\frac{\partial}{\partial \theta_i}\left(-\sum_i y_i \cdot \theta_i\right) = -y_i$$

Step 2: Derivative of the Second Term The derivative of the second term, $\log\left(\sum_j \exp(\theta_j)\right)$, is:

$$\frac{\partial}{\partial \theta_i} \log\left(\sum_j \exp(\theta_j)\right) = \frac{\exp(\theta_i)}{\sum_j \exp(\theta_j)} = \text{softmax}(\theta_i)$$

Final Expression

$$\boxed{\frac{\partial CE(y,\hat{y})}{\partial \theta_i} = -y_i + \text{softmax}(\theta_i)}$$

Note, y_i is the true label, which is a one-hot vector. For the correct class, $y_i = 1$, and for all other classes, $y_i = 0$. softmaxθ_i) is the predicted probability for the i-th class.

1.b

To compute the gradient $\frac{\partial J}{\partial x}$ for a one-hidden-layer neural network, we proceed step-by-step through the backpropagation process.

Neural Network Definitions

The neural network is defined as follows:

- **Hidden Layer:**
$$h = \sigma(xW_1 + b_1)$$
where σ is the sigmoid activation function:
$$\sigma(z) = \frac{1}{1+e^{-z}}.$$
- **Output Layer:**
$$\hat{y} = \text{softmax}(hW_2 + b_2)$$
- **Loss Function (Cross Entropy Loss):**
$$J = CE(y,\hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$
where y is the one-hot encoded label vector.

Gradient Derivation

1. Gradient of J with respect to $\hat{y}$: From the cross-entropy loss:
$$\frac{\partial J}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$
For one-hot encoding, this simplifies to:
$$\frac{\partial J}{\partial \hat{y}} = \hat{y} - y$$

2. Gradient of J with respect to $z := hW_2 + b_2$: Using the chain rule:
$$\frac{\partial J}{\partial z} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z}$$
From the softmax function properties:
$$\frac{\partial \hat{y}}{\partial z} = \hat{y} \cdot (1 - \hat{y})$$
Thus:
$$\frac{\partial J}{\partial z} = \hat{y} - y$$

3. Gradient of z with respect to h: The linear transformation at the output layer is:

$$z = hW_2 + b_2$$

Thus:

$$\frac{\partial z_i}{\partial h_j} = (W_2)_{ji}$$

So:

$$\frac{\partial J}{\partial h_j} = \sum_i \frac{\partial J}{\partial z_i}\frac{\partial z_i}{\partial h_j} = \sum_i (\hat{y}_i - y_i)(W_2)_{ji}$$

In matrix form:

$$\frac{\partial J}{\partial h} = (\hat{y} - y)W_2^T$$

4. Gradient of h with respect to $a := xW_1 + b_1$: The activation at the hidden layer is:

$$h = \sigma(a), \quad a = xW_1 + b_1$$

The derivative of the sigmoid function is:

$$\sigma'(a_j) = h_j(1 - h_j)$$

Thus:

$$\frac{\partial J}{\partial a_j} = \frac{\partial J}{\partial h_j} \cdot \sigma'(a_j) = \frac{\partial J}{\partial h_j} \cdot h_j(1 - h_j)$$

In matrix form:

$$\frac{\partial J}{\partial a} = \frac{\partial J}{\partial h} \cdot h \cdot (1 - h)$$

5. Gradient of a with respect to x: The linear transformation at the hidden layer is:

$$a = xW_1 + b_1$$

Thus:

$$\frac{\partial a_j}{\partial x_k} = (W_1)_{kj}$$

So:

$$\frac{\partial J}{\partial x_k} = \sum_j \frac{\partial J}{\partial a_j}\frac{\partial a_j}{\partial x_k} = \sum_j \frac{\partial J}{\partial a_j}(W_1)_{kj}$$

In matrix form:

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial a}W_1^T$$

6. Putting Everything Together: Combining the results from the previous steps:

$$\frac{\partial J}{\partial x} = \left(\frac{\partial J}{\partial h} \cdot h \cdot (1 - h)\right) W_1^T$$

Substituting $\frac{\partial J}{\partial h} = (\hat{y} - y)W_2^T$:

$$\boxed{\frac{\partial J}{\partial x} = \left((\hat{y} - y)W_2^T \cdot h \cdot (1 - h)\right) W_1^T}$$

1.d

The dev perplexity is: ***113.31***

2 Character-level Language Model

2.a

When comparing character-based and word-based language models, each has its own strengths depending on the task and type of data. Here is a detailed explanation:

Advantages of a Character-Based Language Model

- **Handles Out-of-Vocabulary (OOV) Words:** Character-based models work at the level of individual characters, so they can handle words that the model has never seen before, such as slang, typos, or newly created words. This makes them particularly robust for dynamic and noisy text data, like social media or usernames.
- **Effective for Morphologically Rich Languages:** In languages with complex word structures (e.g., Turkish or Finnish), character-based models can naturally learn the relationships between prefixes, suffixes, and root words, making them highly effective.
- **Compact Vocabulary:** Since the model uses individual characters instead of words, the vocabulary size is much smaller. This reduces memory requirements and simplifies the model.
- **Flexible with Input Lengths:** Character-based models can handle unusually long or short words, such as scientific terms, URLs, or rare names, without issues.

Advantages of a Word-Based Language Model

- **Faster Training and Inference:** Word-based models process entire words, which reduces the number of steps required to analyze text compared to character-level models. This makes them faster and more computationally efficient.
- **Better Semantic Understanding:** Word embeddings used in word-based models often encode rich semantic and syntactic information, allowing these models to capture the meaning and relationships between words more effectively.
- **Efficient Sequence Lengths:** A word-based model requires fewer tokens to represent a sentence compared to a character-based model. This makes processing long texts more efficient.
- **High Performance on Standard Text:** For tasks involving formal or standard text, such as news articles or essays, word-based models often perform better because they can directly leverage the meaning of entire words.

In conclusion Character-based models are more flexible and robust for handling unknown or unusual words, making them great for diverse and noisy datasets. On the other hand, word-based models excel in speed and semantic understanding for tasks involving common or structured language. The choice between the two depends on the nature of the text and the specific requirements of the task.

3 Perplexity

3.a

To show that perplexity calculated using ln (natural logarithm) is equivalent to perplexity calculated using $\log_2$ (base-2 logarithm), we start with the formula in base-2:

$$2^{-\frac{1}{M}\sum_{i=1}^{M}\log_2(p(s_i|s_1,\ldots,s_{i-1}))}$$

Using the property $2^{\log_2(x)} = x$, we can rewrite this as:

$$\prod_{i=1}^{M} p(s_i \mid s_1, \ldots, s_{i-1})^{-\frac{1}{M}}$$

Now, replace $\log_2$ with ln, we get:

$$e^{-\frac{1}{M}\sum_{i=1}^{M}\ln(p(s_i|s_1,\ldots,s_{i-1}))}$$

This demonstrates that the two perplexity formulas are mathematically equivalent:

$$2^{-\frac{1}{M}\sum_{i=1}^{M}\log_2(p)} = e^{-\frac{1}{M}\sum_{i=1}^{M}\ln(p)}$$

3.b

The perplexity of bigram model is computed in q3b.py file that is provided with our code, and perplexity of CRNN model is computed in the notebook of the model.

```
Perplexity of bigram model for shakespreare passage: 5.431735332594533
Perplexity of bigram model for wikipedia passage: 24.694984109600412
```

```
RNN Shakespeare Perplexity: 7.115490829497217
RNN Wikipedia Perplexity: 18.972436053296256
```

3.c

1. Shakespeare Text

- **Bigram Model:** Perplexity = 5.43 The bigram model performs slightly better than the RNN on Shakespeare text. This is because Shakespeare's language has simpler, more repetitive patterns that the bigram model (which only considers pairs of words) can capture effectively.
- **RNN Model:** Perplexity = 7.11 The RNN also performs well but is slightly less efficient for this text. Its ability to capture long-range dependencies is not as critical for the predictable, local word relationships in Shakespeare's style.

2. Wikipedia Text

- **Bigram Model:** Perplexity = 24.69 The bigram model struggles with Wikipedia text because it has more diverse vocabulary and complex sentence structures. Bigram models, which only consider direct word-to-word relationships, are insufficient for such text.
- **RNN Model:** Perplexity = 18.97 The RNN performs significantly better on Wikipedia text due to its ability to capture long-range dependencies and adapt to the diverse and complex nature of this type of content.

In summary, the bigram model works well with simpler, predictable text like Shakespeare, where short-term word patterns dominate. However, the RNN model excels at handling more complex and diverse text like Wikipedia, where long-range dependencies are important. Overall, the RNN is more versatile, but the bigram model can outperform it in cases with simpler, repetitive language patterns.

3.d

The perplexities after the pre-processing addition are:

```
RNN Shakespeare Perplexity preprocessed: 7.088679451635923
RNN Wikipedia Perplexity preprocessed: 16.33134999521697
```

Preprocessing for Shakespearean Text

The Shakespearean text presents unique challenges due to its archaic vocabulary, poetic structure, and stylistic nuances. To improve the perplexity results of the language model on this text, we implemented a preprocessing step designed to preserve the Shakespearean style while making the text more interpretable for the model. Below are the key preprocessing steps and their expected impact:

- **Normalize Archaic Contractions and Words:** Shakespearean language often uses contractions and vocabulary that may not be well-represented in the model's training data. We replaced common archaic terms with their modern equivalents while retaining the essence of the Shakespearean style. This improves token consistency and ensures the text remains true to its original meaning.
- **Remove Non-Standard Characters:** Shakespearean texts frequently contain special characters and diacritics that do not contribute significantly to the model's predictions. Using the `unidecode` library, we converted these characters to their ASCII equivalents, enhancing compatibility with the language model.

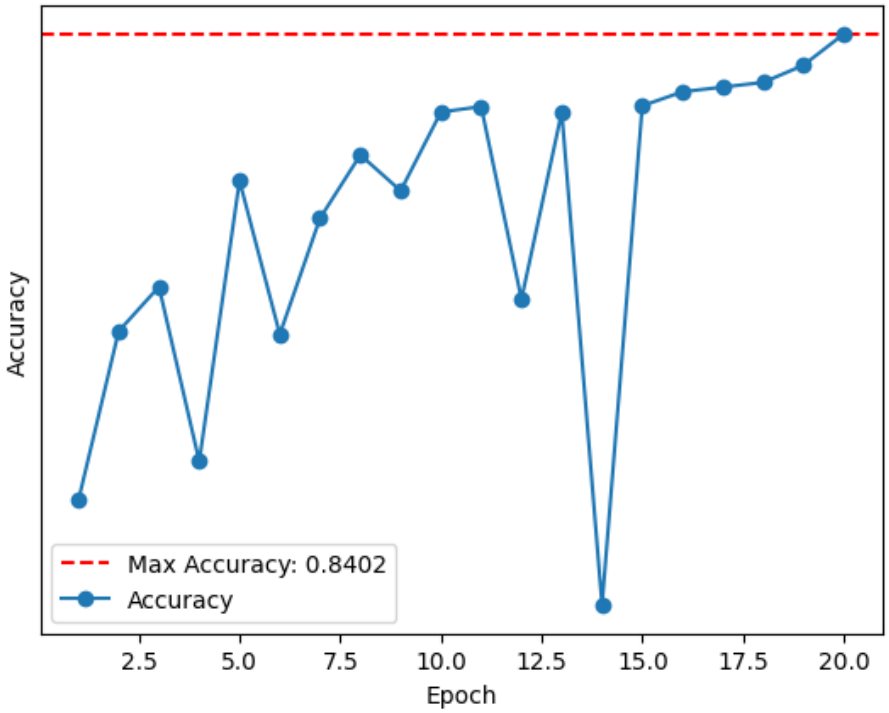
Preprocessing for Wikipedia Text

The Wikipedia text dataset presents a different set of challenges compared to Shakespearean text. Unlike the poetic and archaic structure of Shakespearean language, Wikipedia content is formal, contemporary, and often includes structured paragraphs and technical terminology. The preprocessing step for Wikipedia text focused on cleaning and simplifying the data while preserving its modern and factual nature. Key preprocessing steps include:

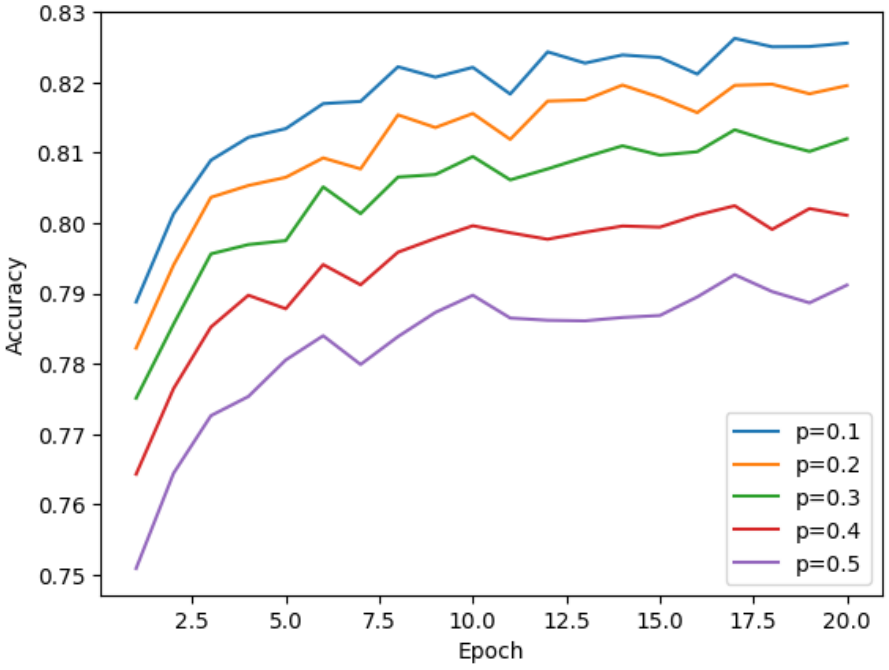
- **Removing Special Characters:** We removed non-alphanumeric characters that did not add meaningful context to the text, ensuring the input remains clean and standardized.
- **Normalize Punctuation:** Irregular punctuation patterns were normalized to improve consistency across the text, making it easier for the model to process.
- **Lowercasing the Text:** All text was converted to lowercase to reduce token variability caused by capitalization, ensuring that identical words are treated uniformly by the model.
- **Removing Extra Whitespace:** Unnecessary spaces and line breaks were removed to standardize the text structure and reduce input noise.

4 Deep Averaging Networks

4.a



4.b



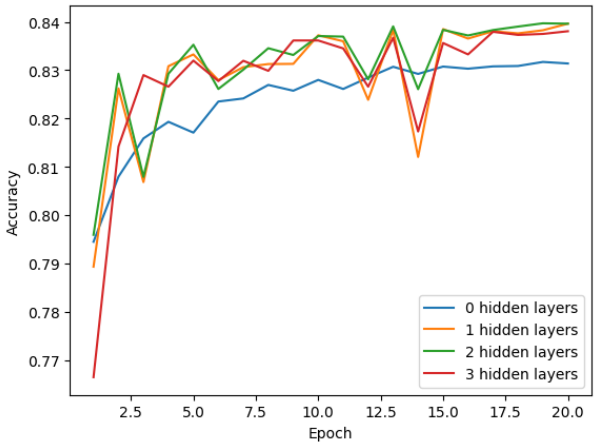
4.c

Here is a summary of the observations:

- The linear model (0 hidden layers) had the lowest accuracy, stabilizing around 82%.
- Adding 1 hidden layer made a significant improvement, with accuracy increasing to around 83.5%.
- Models with 2 and 3 hidden layers performed slightly better, reaching approximately 84% accuracy. However, the difference between these two models was very small.

Adding more hidden layers improves accuracy, but the improvements diminish after the first layer. The jump from 0 to 1 layer is significant, but the difference between 2 and 3 layers is almost negligible.

And indeed our expectations were met, the deeper models performed better than the linear model as expected. However, the additional layers beyond 2 did not provide substantial benefits.



4.d

Analysis of Activation Functions

The experiment compares the performance of three activation functions—ReLU, Sigmoid, and Mish—in terms of evaluation accuracy over 20 epochs. Below are the key observations:

1. Overall Performance

By the final epochs, all three activation functions achieve comparable evaluation accuracy. However, slight differences are observed:

- **ReLU:** Demonstrates stable and consistent improvements across epochs, making it a reliable choice.
- **Sigmoid:** Shows larger fluctuations in accuracy, indicating challenges during optimization. This could be attributed to gradient saturation.
- **Mish:** Performs competitively with minor fluctuations and converges well by the end, slightly outperforming Sigmoid in terms of stability.

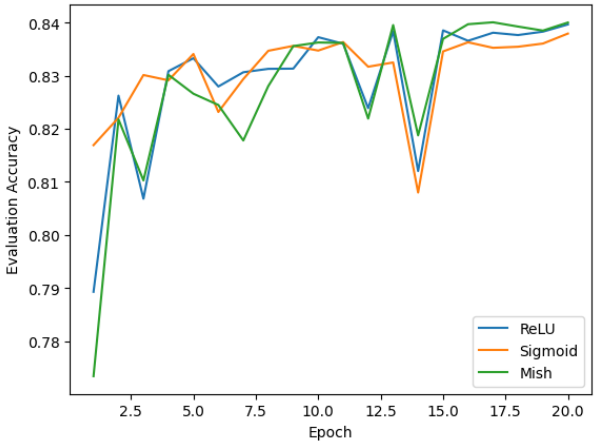
2. Stability and Convergence

- **ReLU:** Smooth and stable curve, indicating strong convergence.
- **Sigmoid:** Initially rises quickly but exhibits instability, making it less consistent in later epochs.
- **Mish:** Shows occasional fluctuations but converges effectively, achieving high accuracy comparable to ReLU.

3. Recommendations

- **Preferred Activation Function:** ReLU or Mish are recommended due to their stability and high accuracy.
- **Further Experiments:**
  - Investigate combinations of activation functions (e.g., using Mish in some layers and ReLU in others).
  - Explore additional activation functions such as Leaky ReLU or Swish.

In conclusion, ReLU and Mish stand out as the most effective activation functions for this experiment. Sigmoid, while viable, suffers from stability issues. For practical applications, ReLU is a simple and reliable choice, whereas Mish offers competitive performance and may yield additional benefits in specific scenarios.



4.e

Example 0

Example: This is just another one of those "American finds romance with charming foreigner in exotic European locale." This genre has been cinematically bankrupt since the 1950s, yet they continue to churn them out. I let my girlfriend talk me into seeing this—bad idea, we both hated it. If all you want to see is Tuscany, try watching *Stealing Beauty* with Liv Tyler—a marginally better movie. This type of genre movie is very Hallmark Greeting Card-ish.

Prediction: Positive **Label:** Negative

Explanation: The model likely classified this example as positive due to phrases like "Tuscany" and "a marginally better movie," which may carry positive connotations. However, the overall sentiment is strongly negative, as reflected by expressions such as "bad idea, we both hated it" and "cinematically bankrupt." The sarcastic and critical tone, particularly in the comparison to "Hallmark Greeting Card-ish" movies, likely confused the model into focusing on isolated positive phrases rather than the broader negative sentiment.

Example 1

Example: Racing enthusiast Fabian (as Tommy Callahan) smokes, drinks, and suffers blackouts while juggling feelings for alluring brunette Annette Funicello (as Francie Madsen) and blonde mainstay Diane McBain (as Annie Blaine). Complicating matters are Ms. Funicello's boozy race car boyfriend Warren Berlinger (as Eddie Sands) and her father Jan Murray (as Pete Madsen), who encourages the reckless drivers. Funicello's cow-eyed performance is sometimes enjoyable; however, her drunken driving scene is unnerving. *Thunder Alley* provides marginally more NASCAR excitement than its predecessor, *Fireball 500* (1966); be warned, it isn't much. A wild party scene, featuring some mild strip tease, is the film's low highlight.

Prediction: Positive **Label:** Negative

Explanation: The model misclassified this example as positive likely because of phrases like "sometimes enjoyable" and "provides marginally more NASCAR excitement." These scattered positive descriptors overshadow the overall negative sentiment, highlighted by phrases like "be warned, it isn't much" and "low highlight." The nuanced tone, blending slight praise with strong criticism, likely caused confusion for the model.

Example 2

Example: Irwin Allen's first venture into all-star spectacle was one all-star disaster. *The Story of Mankind* contains some of the most incredible casting decisions of all time: Virginia Mayo as the blonde Cleopatra, Dennis Hopper chewing the scenery as Napoleon, Peter Lorre dining on the scenery as Nero, and Marie Wilson as Marie Antoinette, a roadshow Marilyn Monroe—that's just the start.

This film is also notable for being the last to feature all three Marx Brothers, though they play separate roles. Chico plays a monk who is Christopher Columbus's confidante, Groucho convinces the Indians to sell Manhattan as Peter Minuit, and Harpo portrays Sir Isaac Newton, discovering gravity when an apple hits his head.

The story centers on mankind being judged as a super H-Bomb threatens to destroy the world. Ronald Colman represents humanity's good side, while Vincent Price, as Old Scratch, argues for humanity's destruction. Despite having some of the finest speaking voices in cinema history, the film's individual stories are so poorly executed that it becomes painful to watch.

Prediction: Positive **Label:** Negative

Explanation: The model likely misclassified this example as positive due to phrases like "some of the finest speaking voices" and "a pleasure to listen to." However, the overall sentiment is negative, as conveyed by statements such as "one all-star disaster" and "you want to scream in agony." The model may

have struggled to recognize the overall critique amidst detailed descriptions of casting and performances.

Example 3

Example: This has to be the ultimate chick flick ever. We taped it off TV years ago, and I've watched it about 30 times over the years. I hadn't seen it for about 12 years and just recently watched it again. I'm not lying; I cried from the opening credits to the ending credits. This movie truly tears your heart out, even if you don't have children.

Prediction: Negative **Label:** Positive

Explanation: The model likely misclassified this example as negative because of the phrase "ultimate chick flick," which can carry a dismissive or sarcastic tone. However, the overwhelmingly positive sentiment—expressed through phrases like "cried from the opening credits to the ending credits" and "tears your heart out"—indicates a strongly positive tone. The model may have been biased by stereotypes or clichés associated with the phrase "chick flick."

Example 4

Example: I've never been a huge fan of Almodóvar, but, generally, I've always found something to enjoy in his films. Unfortunately, I had more trouble finding something to enjoy in *Broken Embraces* than I normally would.

The biggest failure in *Broken Embraces* is its characters, who lack depth. The film is essentially a tragic love story that tries to engage the viewer but ultimately falls flat. One scene I did enjoy involves producer Martel watching video footage obsessively, confirming his worst fears through lip reading. Lena later confronts Martel while matching her speech to the video footage, creating a fascinating but contrived moment.

Unfortunately, the rest of the film left me bored. I couldn't care about the characters or their situations, and no amount of cleverness could compensate for the lack of depth.

Prediction: Positive **Label:** Negative

Explanation: The model likely focused on positive phrases like "one scene I did enjoy," "fascinating," and "brilliant moments," leading to a misclassification. However, the overall sentiment is negative, emphasizing a "lack of depth" and feelings of boredom. The mix of occasional praise within a predominantly critical review likely confused the model.

5 Attention Exploration

5.1

(a) α can be interpreted as a categorical probability distribution because it is defined as:

$$\alpha_i = \frac{e^{k_i^\top q}}{\sum_{j=1}^{n} e^{k_j^\top q}},$$

where $\alpha_i \geq 0$ for all i, and $\sum_{i=1}^{n} \alpha_i = 1$. This ensures that α satisfies the properties of a probability distribution, assigning non-negative weights to each key and normalizing them to sum to 1.

(b) The distribution α puts almost all its weight on α_j when the corresponding dot product $k_j^\top q$ is significantly larger than all other $k_i^\top q$ values. This occurs when the query vector q is nearly aligned (or has a strong similarity) with the key vector k_j.

(c) Under these conditions, the output c becomes approximately equal to the value vector v_j corresponding to the key k_j, as most of the weight α_j is concentrated on v_j, and the contributions from other value vectors are negligible:

$$c \approx v_j.$$

(d) Intuitively, this means that the attention mechanism effectively "copies" the value vector v_j associated with the key k_j that is most similar to the query q. The mechanism prioritizes the value corresponding to the most relevant key, enabling efficient and context-specific information retrieval.

5.2

(a) There exist scalars $c_1, \ldots, c_m$ and $d_1, \ldots, d_p$ such that:

$$v_a = \sum_{i=1}^{m} c_i a_i \quad \text{and} \quad v_b = \sum_{i=1}^{p} d_i b_i.$$

Let us define:

$$M = a_1 a_1^\top + \cdots + a_m a_m^\top.$$

For each $j \in \{1, \ldots, m\}$, we get:

$$Ma_j = \sum_{i=1}^{m} a_i (a_i^\top a_j) = a_j \underbrace{(a_j^\top a_j)}_{=1} + \sum_{i=1, i\neq j}^{m} \underbrace{a_i (a_i^\top a_j)}_{=0} = a_j.$$

For each $j \in \{1, \ldots, p\}$, we get:

$$Mb_j = \sum_{i=1}^{p} \underbrace{a_i (a_i^\top b_j)}_{=0} = \vec{0}.$$

Therefore, we derive:

$$Ms = Mv_a + Mv_b = M\left(\sum_{i=1}^{m} c_i a_i\right) + M\left(\sum_{i=1}^{p} d_i b_i\right) = \sum_{i=1}^{m} c_i Ma_i + \sum_{i=1}^{p} d_i Mb_i.$$

Substituting the results:

$$= \sum_{i=1}^{m} c_i a_i + \sum_{i=1}^{p} \vec{0} = \sum_{i=1}^{m} c_i a_i = v_a.$$

Thus, we have constructed a matrix M such that for all $v_a \in A$ and $v_b \in B$:

$$M(v_a + v_b) = v_a.$$

(b) To maximize the contributions of k_a and k_b to the output c while maintaining balance, we choose a large scalar λ and set $q = \lambda(k_a + k_b)$. This ensures $\alpha_a = \alpha_b \approx \frac{1}{2}$.

Explanation:

Notice that:

$$k_a^\top q = \lambda, \quad k_b^\top q = \lambda, \quad \forall i \notin \{a, b\},\ k_i^\top q = 0.$$

Thus, we derive:

$$\alpha_a = \frac{e^\lambda}{\sum_{i=1}^{n} e^{k_i^\top q}} = \frac{e^\lambda}{(n-2) + 2e^\lambda} \approx \frac{1}{2} \quad \text{(for large } \lambda\text{)}.$$

Similarly, we find $\alpha_b \approx \frac{1}{2}$. Therefore:

$$c = \sum_{i=1}^{n} \alpha_i v_i \approx \frac{1}{2}(v_a + v_b).$$

5.3 Drawbacks of Single-Headed Attention

(a) Knowing that the covariance matrices are vanishingly small, we conclude that $\forall i, k_i \approx \mu_i$. Given that the means are orthogonal, we find that the keys are approximately orthogonal. Specifically:

$$\forall i \neq j,\ k_i^\top k_j \approx 0.$$

This brings us back to the situation described in 5.1.2(b), and the same result $\lambda(k_a + k_b)$ holds.

(b) Since the means are orthonormal, it follows that $\mu_a \mu_a^\top = I$. Thus, the covariance matrix for a can be written as:

$$\Sigma_a = \alpha I + 0.5I \approx 0.5I \quad (\text{for } \alpha \approx 0).$$

This implies:

$$k_a \approx x\mu_a \quad \text{for some } x \sim \mathcal{N}(1, 0.5), \quad \text{and} \quad k_i \approx \mu_i \quad (\forall i \neq a).$$

Given $q = \beta(\mu_a + \mu_b)$, and considering the orthogonality of μ_i for $i \notin \{a, b\}$, the dot product $k_i^\top q$ reduces to zero because $\mu_i^\top(\mu_a + \mu_b) = 0$. For a and b, the dot products are:

$$k_a^\top q \approx x\mu_a^\top \cdot \lambda(\mu_a + \mu_b) = x\lambda,$$
$$k_b^\top q \approx \mu_b^\top \cdot \lambda(\mu_a + \mu_b) = \lambda.$$

Thus, ignoring smaller contributions from other terms, we derive:

$$\alpha_a \approx \frac{1}{1 + e^{x\lambda - \lambda}},$$

$$\alpha_b \approx \frac{1}{1+e^{\lambda x-\lambda}}.$$

For $x = 0.5$, we find $\alpha_a \approx 0$ and $\alpha_b \approx 1$, meaning c aligns closely with v_b. For $x = 1.5$, we find $\alpha_a \approx 1$ and $\alpha_b \approx 0$, meaning c aligns closely with v_a.

On average, given that x alternates between these values, the resulting output c averages out to approximately:

$$c \approx \frac{1}{2}(v_a + v_b),$$

though specific samples may deviate based on the random scaling factor x.

5.4 Benefits of Multi-Headed Attention

(a) If we set $q_1 = \lambda\mu_a$ and $q_2 = \lambda\mu_b$ (choosing the same value of λ as before), we obtain:

$$\alpha_{a,1} \approx 1 \quad \text{and} \quad c_1 \approx v_a,$$

putting almost all the weight on μ_a and μ_b. Therefore, the output is:

$$c \approx \frac{1}{2}(v_a + v_b).$$

(b) Multi-headed attention allows us to become independent of the covariance matrices. As before, taking $k_a \approx x\mu_a$ for some $x \sim \mathcal{N}(1, 0.5)$ and $k_b \approx \mu_b$, we derive:

$$k_a^\top q_1 \approx x\mu_a^\top \mu_a \lambda = x\lambda,$$

$$k_b^\top q_2 \approx \mu_b^\top \mu_b \lambda = \lambda.$$

Thus:

$$\alpha_{a,1} \approx \alpha_{b,2} \approx 1, \quad \text{and the rest of the indices satisfy } \alpha_{i,j} \approx 0.$$

The resulting output vectors are:

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} v_a \\ v_b \end{pmatrix},$$

leading to:

$$c \approx \frac{1}{2}(v_a + v_b).$$