COMP3361 Assignment 3 Submission

Name: SHI Rui
University Number: 3036278067

Spring 2024

1 Written Problems (50%)

1.1 Multi-Choice (20%)

Question Number	Selected Option
1	C
2	D
3	C,D
4	B
5	B
6	D
7	C
8	C
9	B
10	B

Table 1: Selected Options for Multi-Choice Questions

1. Select the answer that includes all the skip-gram (word, context) training pairs for the sentence the cat ran away, for a window size k = 2 from target.

   A) [the, cat], [the, ran], [cat, ran], [cat, away], [ran, away]

   B) [the], [cat], [ran], [away]

   C) [the, cat], [the, ran], [cat, the], [cat, ran], [cat, away], [ran, the], [ran, cat], [ran, away], [away, cat], [away, ran]

   D) [the, ran], [ran, cat], [cat, away]

   E) [the, cat ran], [cat, the], [cat, ran away], [ran, the cat], [ran, away], [away, cat ran]

2. What if gradients become too large or small?

   A) If too large, the model will become difficult to converge

   B) If too small, the model can't capture long-term dependencies

   C) If too small, the model may capture a wrong recent dependency

   D) All of the above

3. Which of the following methods do not involve updating the model parameters? Select all that apply.

   A) Fine-tuning

   B) Transfer learning

   C) In-context learning

   D) Prompting

4. What range of values can cross entropy loss take?

   A) 0 to 1

   B) 0 to ∞

   C) -1 to 1

   D) $-\infty$ to 0

5. Can we use bidirectional RNNs in the following tasks? (1) text classification, (2) code generation, (3) text generation

   A) Yes, Yes, Yes

   B) Yes, No, No

   C) Yes, Yes, No

   D) No, Yes, No

6. Select the models that are capable of generating dynamic word embeddings, which can change depending on the surroundings of a word in a sentence.

   A) Bag of Words (BoW)

   B) Word2Vec

   C) GloVe

   D) T5

7. What makes Span Corruption a unique pretraining objective compared to traditional MLM?

   A) It involves masking and predicting individual tokens rather than spans of tokens.

   B) It only uses punctuation marks as indicators for span boundaries.

   C) It masks contiguous spans of text and trains the model to predict the masked spans, encouraging understanding of longer context.

   D) It requires the model to correct grammatical errors within the span.

8. In the transformer model architecture, positional encodings are added to the input embeddings to provide the model with information about the position of tokens in the sequence. Given a sequence length of L and a model dimension of D, which of the following PyTorch code snippets correctly implements the calculation of sinusoidal positional encodings?

   A)
   ```
   def positional_encoding(L, D):
   ```

```
    position = torch.arange(L).unsqueeze(1)
    div_term = torch.exp(torch.arange(0, D, 2) * -(np.log(10000.0) / D))
    pe = torch.zeros(L, D)
    pe[:, 0::2] = torch.sin(position * div_term)
    pe[:, 1::2] = torch.cos(position * div_term)
    return pe
```

B)

```
def positional_encoding(L, D):
    position = torch.arange(L).unsqueeze(1)
    div_term = torch.exp(torch.arange(0, D, 2) * -(np.log(10000.0) / D))
    pe = torch.zeros(L, D)
    pe[:, 0::2] = torch.sin(position / div_term)
    pe[:, 1::2] = torch.cos(position / div_term)
    return pe
```

C)

```
def positional_encoding(L, D):
    position = torch.arange(L, dtype=torch.float).unsqueeze(1)
    div_term = 1 / (10000 ** (2 * torch.arange(D // 2) / D))
    pe = torch.zeros(L, D)
    pe[:, 0::2] = torch.sin(position * div_term)
    pe[:, 1::2] = torch.cos(position * div_term)
    return pe
```

D)

```
def positional_encoding(L, D):
    position = torch.arange(L, dtype=torch.float).unsqueeze(1)
    div_term = torch.exp(torch.arange(0, D, 2) * -(np.log(10000.0) / L))
    pe = torch.zeros(L, D)
    pe[:, 0::2] = torch.sin(position * div_term)
    pe[:, 1::2] = torch.cos(position / div_term)
    return pe
```

9. In instruction tuning, what is the primary benefit of using natural language instructions for model fine-tuning?

   A) Reduces the need for labeled data in supervised learning tasks.

   B) Enables the model to perform zero-shot or few-shot learning on tasks it was not explicitly trained for.

   C) Significantly decreases the computational resources needed for training large models.

   D) Allows the model to improve its performance on specific tasks without fine-tuning.

10. How does Low-Rank Adaptation (LoRA) efficiently fine-tune large pre-trained models for specific tasks?

    A) By freezing the original parameters and training a small set of new parameters introduced as adapters at certain layers, significantly reducing computational needs.

    B) By applying low-rank matrices to adjust the attention weights, enabling task-specific tuning without extensively retraining the original parameters.

    C) By pruning less important neurons based on initial assessments, simplifying the model for specific tasks with minimal performance impact.

    D) By adding task-specific tokens to the model's vocabulary and fine-tuning their embeddings only, leveraging the existing model for seamless integration.

1.2 Short Answer (30%)

Question 3.1: A trigram language model is also often referred to as a second-order Markov language model. It has the following form:

$$P\left(X_1 = x_1, \ldots, X_n = x_n\right) = \prod_{i=1}^{n} P\left(X_i = x_i \mid X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1}\right)$$

Question 3.1a: Could you briefly explain the advantages and disadvantages of a high-order Markov language model compared to the second-order one?

Question 3.1b: Could you give some examples in English where English grammar suggests that the second-order Markov assumption is clearly violated?

——Solution——

3.1 a

Advantages:

- Capture more information: High-order Markov language model pays more attention to context and takes more previous information into account. Thus, high-order Markov language model could capture long-term dependency more accurate and precise, ending up with better performance in some extent.

Disadvantages:

- Long time to train: Owing to it has to consider more previous information, the dimension when computing probability is extreme higher than second-order Markov LM. In other words, long computing time could be inevitable.

- Hard to train: The probability and gradient of high-order Markov LM could be smaller in general. (We used to let concurrence to represent joint probability on linear structure model, such as RNN). On top of that, we also need to multiple several probability to get the ultimate probability. That means we would high likely encounter gradient vanishing problem.

- Not well-defined: It is hard to define $P(X_i = x_i | X_{i-n} = x_{i-n}, ..., X_{i-1} = x_{i-1})$, because we usually use words' concurrence (N-gram) to represent the conditional probability. The equation is showed below, and we will find $c(X_{i-n} = x_{i-n}, ..., X_{i-1} = x_{i-1})$ [dividend] is high likely to be 0 when n is ascending.

$$\begin{aligned} p(X_i = x_i | X_{i-n} = x_{i-n}, ..., X_{i-1} = x_{i-1}) &= \frac{p(X_{i-n} = x_{i-n}, ..., X_{i-1} = x_{i-1}, X_i = x_i)}{p(X_{i-n} = x_{i-n}, ..., X_{i-1} = x_{i-1})} \\ &= \frac{c(X_{i-n} = x_{i-n}, ..., X_{i-1} = x_{i-1}, X_i = x_i)}{c(X_{i-n} = x_{i-n}, ..., X_{i-1} = x_{i-1})} \end{aligned}$$

3.1 b

[1] "It is impossible for him, a CEO of multi-national enterprise, to participate our meeting."

The subject of "to participate" is "him", not "enterprise". But second-order Markov could only recognize "enterprise" to be the subject of "to participate" when predicting the next token of "enterprise".

[2] "I punched him, so he was attacked by me."

Second-order Markov LM could not recognized the verb conducted by "he" should be passive voice.

Question 3.2 We'd like to define a language model with $V = \{\text{the, a, dog}\}$, and $p(x_1 \ldots x_n) = \gamma \times 0.5^n$ for any $x_1 \ldots x_n$, such that $x_i \in V$ for $i = 1 \ldots (n-1)$, and $x_n =$ STOP, where γ is some expression (which may be a function of n).

Which of the following definitions for γ give a valid language model? Please choose the answer and prove it.

(Hint: recall that $\sum_{n=1}^{\infty} 0.5^n = 1$)

1. $\gamma = 3^{n-1}$
2. $\gamma = 3^n$
3. $\gamma = 1$
4. $\gamma = \frac{1}{3^n}$
5. $\gamma = \frac{1}{3^{n-1}}$

——**Solution**——

Owing to the definition of Language Model, the two rules showed below should be obeyed.

$V^\dagger$ represents all complete sentences, which only consists tokens in V and ends up with STOP with arbitrary length.

$$\begin{aligned} &1. \quad 0 \le p(x_1, ..., x_n) \le 1 \\ &2. \sum_{<x_1,...,x_n> \in V^\dagger} p(x_1, ..., x_n) = 1 \end{aligned}$$

Result: Only when $\gamma = \frac{1}{3^{n-1}}$ is satisfied, it could give a valid language model.

TODO: We need to traverse all possible lengths of sentences ($\sum_{n=1}^{\infty}$) and consider the probability in each length ($p = \gamma * (0.5)^n$).

1. $\gamma = 3^{n-1}$ **[Not valid]**

$$\text{rule 1: } p(x_1, ..., x_n) = 3^{n-1} * (0.5)^n = \frac{1}{2}(\frac{3}{2})^{n-1}$$

when n=3, $p(x_1, ..., x_n) = \frac{9}{8} > 1$. Thus this model could be a valid language model.

2. $\gamma = 3^n$**[Not valid]**

$$\text{rule 1: } p(x_1, ..., x_n) = 3^n * (0.5)^n = (\frac{3}{2})^n$$

when n=1, $p(x_1, ..., x_n) = \frac{3}{2} > 1$. Thus this model could be a valid language model.

3. $\gamma = 1$**[Not Valid]**

$$\text{rule 1: } 0 < p(x_1, ..., x_n) = 1 * (0.5)^n = (\frac{1}{2})^n < 1\text{[ok]}$$

$$\text{rule 2: } \sum_{<x_1,...,x_n> \in V^\dagger} p(x_1, ..., x_n) = \sum_{n=1}^{\infty} 3^{n-1}(0.5)^n \neq 1[violate]$$

4. $\gamma = \frac{1}{3^n}$**[Not valid]**

$$\text{rule 1: } 0 < p(x_1, ..., x_n) = (\frac{1}{6})^n < 1\text{[ok]}$$
$$\text{rule 2: } \sum_{<x_1,...,x_n> \in V^\dagger} p(x_1, ..., x_n) = \sum_{n=1}^{\infty} 3^{n-1} * (\frac{1}{6})^n \neq 1[violate]$$

5. $\gamma = \frac{1}{3^{n-1}}$**[Valid]**

$$\text{rule 1: } 0 < p(x_1, ..., x_n) = 3 * (\frac{1}{6})^n < 1\text{[ok]}$$
$$\text{rule 2: } \sum_{<x_1,...,x_n> \in V^\dagger} p(x_1, ..., x_n) = \sum_{n=1}^{\infty} 3^{n-1} * \frac{1}{3^{n-1}} * (0.5)^n = \sum_{n=1}^{\infty} (0.5)^n = 1[ok]$$

To sum up, only when γ satisfies $\gamma = \frac{1}{3^{n-1}}$, the model could align with the two rules of language model.

Question 3.3 Given a small document corpus D consisting of two sentences: {"i hug pugs", "hugging pugs is fun"} and a desired vocabulary size of N=15, apply the Byte Pair Encoding (BPE) algorithm to tokenize the documents.

Assume the initial vocabulary includes individual characters and spaces as separate tokens. The BPE algorithm should merge the most frequent adjacent pairs of tokens iteratively until the vocabulary size reaches N=15.

Question 3.3a What is the final list of the desired vocabulary tokens?

Question 3.3b What is the final list of document tokens after reaching the desired vocabulary size?

——**Preprocessing**——

I will firstly give the whole running process of BPE algorithm to get a desired vocabulary size of N=15. And then I will extract the answers from the running process of the two questions.

State 0

V_0 = {' ', 'i', 'h', 'u', 'g', 'p', 's', 'n', 'f'}, $|V_0| = 9$

D_0 = {['i'], [' ', 'h', 'u', 'g', 's'], [' ', 'p', 'u', 'g'],[' ','h', 'u', 'g', 'g', 'i', 'n', 'g'], [' ', 'p', 'u', 'g', 's'], [' ', 'i', 's'], [' ', 'f', 'u', 'n']}

TODO: max(bigram(token 1,token 2)) = N('u', 'g') = 4, merge token 'u' and 'g'

State 1

V_1 = {' ', 'i', 'h', 'u', 'g', 'p', 's', 'n', 'f', 'ug'}, $|V_1| = 10$

D_1 = {['i'], [' ', 'h', 'ug'], [' ', 'p', 'ug','s'],[' ','h', 'ug', 'g', 'i', 'n', 'g'], [' ', 'p', 'ug', 's'], [' ', 'i', 's'], [' ', 'f', 'u', 'n']}

TODO: max(bigram(token 1,token 2)) = N(' ', 'h') = 2, merge token ' ' and 'h'

State 2

V_2 = {' ', 'i', 'h', 'u', 'g', 'p', 's', 'n', 'f', 'ug', ' h'}, $|V_2| = 11$

D_2 = {['i'], [' h', 'ug'], [' ', 'p', 'ug','s'],[' h', 'ug', 'g', 'i', 'n', 'g'], [' ', 'p', 'ug', 's'], [' ', 'i', 's'], [' ', 'f', 'u', 'n']}

TODO: max(bigram(token 1,token 2)) = N(' h', 'ug') = 2, merge token ' h' and 'ug'

State 3

V_3 = {' ', 'i', 'h', 'u', 'g', 'p', 's', 'n', 'f', 'ug', ' h', ' hug'}, $|V_3| = 12$

D_3 = {['i'], [' hug'], [' ', 'p', 'ug','s'],[' hug', 'g', 'i', 'n', 'g'], [' ', 'p', 'ug', 's'], [' ', 'i', 's'], [' ', 'f', 'u', 'n']}

TODO: max(bigram(token 1,token 2)) = N('p', 'ug') = 2, merge token 'p' and 'ug'

State 4

V_4 = {' ', 'i', 'h', 'u', 'g', 'p', 's', 'n', 'f', 'ug', ' h', ' hug', 'pug'}, $|V_3| = 13$

D_4 = {['i'], [' hug'], [' ', $\underline{\text{'pug','s'}}$],[' hug', 'g', 'i', 'n', 'g'], [' ', $\underline{\text{'pug', 's'}}$], [' ', 'i', 's'], [' ', 'f', 'u', 'n']}

TODO: max(bigram(token 1,token 2)) = N('pug','s') = 2, merge token 'pug' and 's'

State 5

V_5 = {' ', 'i', 'h', 'u', 'g', 'p', 's', 'n', 'f', 'ug', ' h', ' hug', 'pug', 'pugs'}, $|V_5| = 14$

D_5 = {['i'], [' hug'], [$\underline{\text{' ', 'pugs'}}$],[' hug', 'g', 'i', 'n', 'g'], [$\underline{\text{' ', 'pugs'}}$], [' ', 'i', 's'], [' ', 'f', 'u', 'n']}

TODO: max(bigram(token 1,token 2)) = N(' ','pugs') = 2, merge token ' ' and 'pugs'

State 6

V_6 = {' ', 'i', 'h', 'u', 'g', 'p', 's', 'n', 'f', 'ug', ' h', ' hug', 'pug', 'pugs', ' pugs'}, $|V_6| = 15$

D_6 = {['i'], [' hug'], [' pugs'],[' hug', 'g', 'i', 'n', 'g'], [' pugs'], [' ', 'i', 's'], [' ', 'f', 'u', 'n']}

——**Solution**——

3.3 a The final list of the desired vocabulary tokens is showed below.

V = {' ', 'i', 'h', 'u', 'g', 'p', 's', 'n', 'f', 'ug', ' h', ' hug', 'pug', 'pugs', ' pugs'}

3.3 b The final list of the document tokens is showed below.

D = {['i'], [' hug'], [' pugs'],[' hug', 'g', 'i', 'n', 'g'], [' pugs'], [' ', 'i', 's'], [' ', 'f', 'u', 'n']}

Question 3.4 Let $\mathbf{Q} \in \mathbb{R}^{N \times d}$ denote a set of N query vectors, which attend to M key and value vectors, denoted by matrices $\mathbf{K} \in \mathbb{R}^{M \times d}$ and $\mathbf{V} \in \mathbb{R}^{M \times c}$ respectively. For a query vector at position n, the softmax attention function computes the following quantity:

$$\operatorname{Attn}\left(\mathbf{q}_n, \mathbf{K}, \mathbf{V}\right) = \sum_{m=1}^{M} \frac{\exp\left(\mathbf{q}_n^\top \mathbf{k}_m\right)}{\sum_{m'=1}^{M} \exp\left(\mathbf{q}_n^\top \mathbf{k}_{m'}\right)} \mathbf{v}_m^\top := \mathbf{V}^\top \operatorname{softmax}\left(\mathbf{K}\mathbf{q}_n^\top\right)$$

which is an average of the set of value vectors $\mathbf{V}$ weighted by the normalized similarity between different queries and keys.

Please briefly explain what is the time and space complexity for the attention computation from query $\mathbf{Q}$ to $\mathbf{K}$, $\mathbf{V}$, using the big O notation.

——**Solution**——

There are three steps to get the new representation matrix of initial vectors. I will show these process in matrix expression, which will simplify our analysis of time and space complexity.

tip: @ annotates matrix multiplication.

$$\begin{aligned} &step1. \quad A = K@Q^T \in \mathcal{R}^{M*N} \\ &step2. \quad A' = softmax(A) \in \mathcal{R}^{M*N} \\ &step3. \quad O = V^T@A' \in \mathcal{R}^{c*N} \end{aligned}$$

tip: "Space complexity with optimizing" means we could store intermediate computation result in a single space, and we could cover it when we get a new result. "Space complexity (without optimizing)"

step 1: get initial attention map

Time complexity: $O(d*M*N)$

Space complexity (wit optimizing): $O(N*d+M*d+M*N)$

step 2: softmax

Time complexity: $O(M*N)$

Space complexity (wit optimizing): $O(1)$

step 3: get new representation matrx

Time complexity: $O(c*M*N)$

Space complexity (wit optimizing): $O(M*c)$

Thus, the total time and space complexity should be

$$\begin{aligned} \text{time complexity :}&O((c+d)*M*N) \\ \text{space complexity :}&O(N*d+M*d+M*N+M*c) \end{aligned}$$