Prof. Antoine Bosselut

Modern Natural Language Processing – CS-552
17.04.2025 from 13h30 to 14h30
Duration : 60 minutes

1

Make-Up Midterm 2025 Solutions

SCIPER : **111111**

Do not turn the page before the start of the exam. This document is double-sided, has 12 pages, the last ones possibly blank. Do not unstaple.

- This is a closed book exam. Non-programmable calculators are allowed. No other electronic devices of any kind are allowed.
- Place on your desk: your student ID, writing utensils, one double-sided A4 page cheat sheet if you have one; place all other personal items below your desk.
- You each have a different exam.
- This exam has **multiple-choice** questions of varying difficulty. Each question is worth **one point**.
- Each question has **exactly one** correct answer. For each question, mark the box corresponding to the correct answer. You are not expected to get every question right even for the best grade.
- Only answers in this booklet count. No extra loose answer sheets. You can use the blank pages at the end as scrap paper.
- Use a **black or dark blue ballpen** and clearly erase with **correction fluid** if necessary.
- If a question turns out to be wrong or ambiguous, we may decide to nullify it.

Respectez les consignes suivantes | Observe this guidelines | Beachten Sie bitte die unten stehenden Richtlinien

choisir une réponse | select an answer Antwort auswählen	ne PAS choisir une réponse | NOT select an answer NICHT Antwort auswählen	Corriger une réponse | Correct an answer Antwort korrigieren
☒ ☑ ■	□	□

ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte

▨ ☒ □ ⊡ ▨ □

Question 1 Consider the following statements with respect to in-context learning (ICL):
A: ICL is an emergent property of large language models.
B: ICL is similar to supervised learning in that the model learns by updating its parameters.
C: It is necessary to give some examples of the desired task when using the ICL paradigm.
D: Zero-shot learning is also ICL.
E: ICL is robust to the ordering of in-context examples within the prompt.
F: A unifying theory of ICL is that it always works by performing gradient descent in the forward pass.
G: Prompt engineering can make ICL much more effective.

Which of the above statements are **TRUE**:

□ A and G
□ A, D and G
□ A, D, E and G
□ A, E and G
□ A and E
□ G only
□ A only
□ A, B, C and G
■ A, C and G
□ A, D and E

Solution: Statements A, C and G are true as ICL is an emergent property, requires examples in the prompt, and can be made effective using prompt engineering. B is false as there are no parameter updates during ICL. Zero-shot learning is not ICL. Hence D is false. ICL is affected by the order of in-context examples. Hence E is false. F is false as despite there being theories on how ICL works (gradient descent being one of them), they fail to explain ICL in all settings.

Question 2 Consider a Transformer model with a total hidden dimension $d_{\text{model}} = 768$ and $h = 12$ parallel attention heads. Each attention head independently computes scaled dot-product attention.
What is the dimensionality d_{head} of each individual attention head?

□ $d_{\text{head}} = 12$
□ $d_{\text{head}} = 1024$
□ $d_{\text{head}} = 768$
■ $d_{\text{head}} = 64$
□ $d_{\text{head}} = 128$

Solution: Each attention head in a Transformer splits the total hidden dimension $d_{\text{model}} = 768$ evenly across $h = 12$ heads. So, the dimensionality per head is $d_{\text{head}} = 768/12 = 64$.

Question 3 DistilBERT is a compressed version of BERT obtained via knowledge distillation. Let $P_{\text{BERT}}(y_t^* \mid \dots)$ represent the teacher BERT's output distribution and $P_{\text{dbert}}(y_t^* \mid \dots)$ represent the student DistilBERT's distribution. Suppose $[M]$ is a masked token and $\{y_s^*\}_{s\neq t}$ are additional tokens (e.g., context). Which of the following statements **best** describes the knowledge distillation objective used to train DistilBERT?

☐ DistilBERT is trained to copy only the teacher's argmax token predictions, ignoring softened teacher distributions.

■ The objective is a combined loss:

$$\mathcal{L}_{\text{distil}} = -\,P_{\text{BERT}}\big(y_t^* \,\big|\, [M], \{y_s^*\}_{s\neq t}\big)\ \log\ P_{\text{dbert}}\big(y_t^* \,\big|\, [M], \{y_s^*\}_{s\neq t}\big),$$

$$\mathcal{L}_{\text{mlm}} = -\log\ P_{\text{dbert}}\big(y_t^* \,\big|\, [M], \{y_s^*\}_{s\neq t}\big),$$

$$\mathcal{L} = \gamma_1\,\mathcal{L}_{\text{distil}}\ +\ \gamma_2\,\mathcal{L}_{\text{mlm}},$$

where γ_1, γ_2 are weighting coefficients.

☐ The only requirement is to match the teacher's hidden representations layer by layer, with no cross-entropy or distillation-specific terms involved.

☐ It only uses the standard masked language modeling (MLM) loss from DistilBERT,

$$\mathcal{L}_{\text{mlm}} = -\log\big(P_{\text{dbert}}(y_t^* \mid [M], \{y_s^*\}_{s\neq t})\big),$$

☐ The training process relies exclusively on matching the teacher's predicted distribution. Specifically:

$$\mathcal{L}_{\text{distil}} = -\,P_{\text{BERT}}\big(y_t^* \,\big|\, [M], \{y_s^*\}_{s\neq t}\big)\ \log\ P_{\text{dbert}}\big(y_t^* \,\big|\, [M], \{y_s^*\}_{s\neq t}\big).$$

Solution: DistilBERT is trained using a combined loss that balances two objectives: matching the teacher BERT's output distribution (a distillation loss) and predicting the correct token via standard masked language modeling (MLM). This encourages the student model to both mimic the teacher's behavior and learn from ground truth labels, with weighting coefficients controlling the contribution of each term.

Question 4 Which of the following statements best describes how pretraining and finetuning differ among ELMo, GPT, and BERT?

☐ ELMo relies on a purely masked language modeling objective, requiring full-model fine-tuning. GPT is never fine-tuned, and BERT is only used as a left-to-right LM with no end-to-end fine-tuning.

☐ ELMo, GPT, and BERT share identical pretraining strategies and always follow the same fine-tuning procedure.

☐ ELMo is based on a denoising autoencoder objective, GPT focuses on next-sentence prediction, and BERT is purely a left-to-right language model.

☐ ELMo does not involve language modeling pretraining at all. GPT is a bidirectional masked language model, and BERT is always used as a fixed feature extractor.

■ ELMo uses a bidirectional LSTM language model to learn context-dependent embeddings. GPT uses a left-to-right Transformer and is finetuned end-to-end. BERT uses masked language modeling (and next-sentence prediction) during pretraining and is also fine-tuned end-to-end.

Solution: ELMo uses a bidirectional LSTM trained as a language model to produce contextualized embeddings, but it's typically used as a feature extractor. GPT uses a left-to-right Transformer and is fine-tuned end-to-end for downstream tasks. BERT is pretrained using masked language modeling and next-sentence prediction, and is also fine-tuned end-to-end.

Question 5 In an encoder-decoder Transformer model, what precisely is the input to the cross-attention module in the decoder?

□ Positional embeddings as queries and decoder hidden states as keys.

□ Decoder's token embeddings as keys and encoder outputs as queries.

□ The output of intermediate encoder layers as keys and decoder hidden states as queries.

□ Decoder token embeddings as both keys and queries.

■ The output of the final encoder layer as keys and decoder hidden states as queries.

Solution: In the cross-attention module of a Transformer decoder, the decoder's hidden states serve as queries, while the keys and values come from the output of the final encoder layer. This allows the decoder to attend to relevant parts of the encoded input sequence while generating each output token.

Question 6 Consider a recurrent neural network (RNN) being trained with backpropagation through time (BPTT). The hidden state at time step t is computed as:

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

Which of the following statements **best** describes the vanishing gradient problem that arises in this scenario?

■ Gradients computed at earlier timesteps become extremely small, approaching zero, effectively preventing the network from learning long-range dependencies.

□ Gradients vanish because the input x_t at later timesteps dominates, making gradients with respect to previous hidden states irrelevant.

□ Gradients vanish because they are always exactly zero at initialization, thus the network must rely on random perturbations to begin learning.

□ Gradients computed in earlier layers become extremely large (explode), causing numerical instability and divergence during training.

Solution: The vanishing gradient problem in RNNs occurs when gradients at earlier timesteps become extremely small during backpropagation, making it difficult for the network to learn long-range dependencies. This happens because of repeated multiplication through time.

Question 7 Which of the following statements is **FALSE**?

□ Dense embeddings learn semantic relationships between tokens while sparse one-hot embeddings cannot.

□ CBOW and Skip-gram algorithms do not take into account the order of words in the context.

□ Supervised learning with a task-specific objective cannot yield task-independent embeddings.

■ A model with only sum-pool aggregation layers can be used for sequence labeling task.

Solution: A model with only sum-pool aggregation over tokens cannot compute token-specific representations needed for sequence labeling tasks. All other statements are true.

Question 8 Which of the following statements is **FALSE** regarding the attention mechanism commonly used in encoder-decoder architectures (e.g., LSTM or GRU models with attention)?

□ Attention mechanisms are agnostic to the specific recurrent network architecture used in encoder-decoder models, meaning they can be equally applied to LSTMs, GRUs, or even non-recurrent models.

□ The attention mechanism can be computed using various scoring functions, each with different parameterizations and computational implications.

□ Attention mechanisms alleviate the vanishing gradient problem by creating direct, differentiable pathways between encoder and decoder states, enabling more effective gradient flow.

□ Attention mechanisms provide interpretability by explicitly representing alignment weights between input and output sequences.

■ Attention mechanisms significantly increase computational complexity from linear to exponential with respect to input sequence length, making them impractical for long sequences.

Solution: The false statement is that attention mechanisms increase computational complexity from linear to exponential with respect to input length. In reality, attention introduces quadratic complexity—not exponential—since each token attends to all others.

Question 9 What is the benefit of the backpropagation algorithm?

□ It eliminates the need for activation functions

□ The gradient can be more accurately computed

□ It reduces the memory requirements of training

■ It enables the computation of partial derivatives through multiple layers

□ It allows for parallel computation across multiple GPUs

Solution:

- Backpropagation is a gradient-computation method, thus it cannot eliminate activation functions.
- Backpropagation does not affect the gradient computation accuracy.
- Backpropagation improves the time efficiency but increase the memory requirements since it stores the computation map for every partial gradients.
- Backpropagation does not affect the performance from multi-GPU parallelism.
- **Correct choice:** Backpropagation enables the computation of partial derivatives through multiple layers.

Question 10 Consider the following statements regarding text generation:
A: Greedy decoding is prone to generating repetitive sequences because the negative log-likelihood of these sequences keeps on increasing with sequence length.
B: Top-k sampling considers the k most probable tokens for sampling, while top-p (nucleus) sampling considers the tokens corresponding to top p cumulative probability mass for sampling.
C: Top-p (nucleus) sampling can be seen as a variant of top-k sampling with a dynamically varying k based on the probability distribution.
D: Top-k sampling can be seen as a variant of top-p (nucleus) sampling with a dynamically varying p based on the first k tokens whose cumulative probabilities equal or exceed p.
E: It is reasonable to use a higher temperature to generate more deterministic text, and a smaller temperature to generate more creative text.

Which of the above statements are **FALSE**:

□ A, C and D
□ None of the other options can be chosen
□ D only
□ D and E
■ A and E
□ E only
□ A and D

Solution: A is false because negative log-likelihood decreases with sequence length in case of greedy decoding. E is false as smaller temperature leads to more deterministic output and higher temperature leads to more creative output. B, C and D are true by the definition of top-k and top-p sampling algorithms.

Question 11 A language model generates the following probability distribution over the next token choices:

[`"greetings"` (0.1), `"hi"` (0.3), `"hello"` (0.4), `"hey"` (0.15), `"yo"` (0.05)]

If we use **top-k sampling with** $k = 3$ and **top-p sampling with** $p = 0.75$, which tokens will be considered in each case?

■ Top-k: `["hello", "hi", "hey"]`, Top-p: `["hello", "hi", "hey"]`
□ Top-k: `["hello", "hi", "greetings"]`, Top-p: `["hello", "hi", "hey"]`
□ Top-k: `["hello", "hi", "yo"]`, Top-p: `["hello", "hi", "hey"]`
□ Top-k: `["hello", "hi", "hey"]`, Top-p: `["hello", "hi"]`

Solution: Top-k with $k = 3$ considers the tokens ["hello", "hi", "hey"] corresponding to probabilities $0.4, 0.3, 0.15$. Top-p with $p = 0.75$ considers the tokens ["hello", "hi", "hey"] as this is the smallest set of top probability tokens with cumulative probability greater than or equal to 0.75. Their cumulative probability is $0.4 + 0.3 + 0.15 = 0.85$.

Text Generation

The following tree of tokens in Figure 1 is relevant for the next few questions. The boxes represent tokens. Each arrow $a \xrightarrow{q} b$ represents the choice of next token being b given the token prefix ending at a. The probability of b given prefix ending at a is q. Only the highest probability tokens for each prefix are shown.

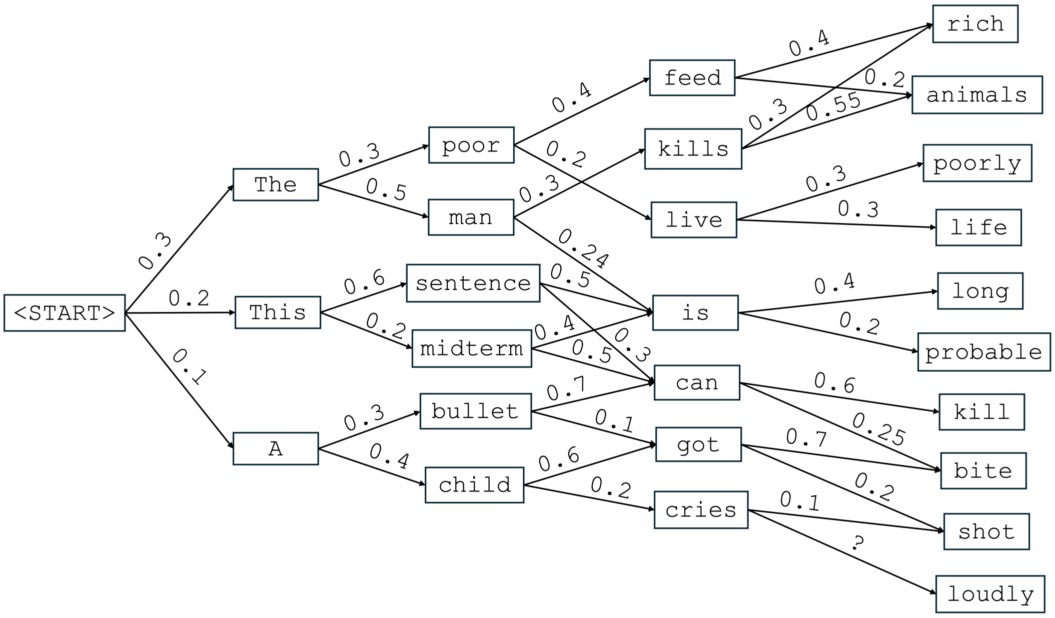

Figure 1. Tree of tokens and probabilities.

Question 12 Assume that greedy argmax decoding is used. What is the generated sequence and what is its probability?

□ *Sequence:* `A bullet can kill` ; *Probability:* 0.0126

☒ *Sequence:* `The man kills animals` ; *Probability:* 0.02475

□ *Sequence:* `This sentence is long` ; *Probability:* 0.024

□ *Sequence:* `This midterm is long` ; *Probability:* 0.0364

Solution: Taking the highest probability transition at each step yields the sequence `The man kills animals` with probability $0.3 * 0.5 * 0.3 * 0.55 = 0.02475$.

Question 13 The statement *"When using Beam Search with beam size of 2, the sequence A has a higher probability of being generated than when using Beam Search with beam size of 3"* is true for which of the following sequence *A*:

☒ None of the other options are correct

□ `A bullet can kill`

□ `This midterm is probable`

□ `The man is long`

□ `The poor live life`

□ `This sentence can bite`

Solution: This question requires understanding the difference between the probability assigned to a sequence by the model and the probability of a sequence being generated by Beam Search. Note that Beam Search is a deterministic search algorithm. A sequence is generated by it when the sequence has the most probability assigned by the model among the candidate beams. Otherwise, the sequence is not generated. So, the probability of a sequence being generated by Beam Search is either 1 or 0. For this particular case, Beam Search generates `The man kills animals` for both beam sizes 2 and 3. So, the probability of this sequence being generated is 1 by Beam Search of beam size 2 as well as 3. The probability of any other sequence being generated is 0. So, there does not exist any sequence which has a higher probability of being generated by Beam Search with beam size 2 than Beam Search with beam size 3.

Code Comprehension: Sampling

You are given the following top-p (nucleus) sampling function:

```
1 def top_p_sampling(probs, p=0.9):
2     sorted_indices = np.argsort(probs)[::-1]
3     sorted_probs = probs[sorted_indices]
4     cumulative_probs = np.cumsum(sorted_probs)
5     cutoff_idx = np.searchsorted(cumulative_probs, p)
6     top_p_indices = sorted_indices[:cutoff_idx]
7     top_p_probs = sorted_probs[:cutoff_idx]
8     top_p_probs /= np.sum(top_p_probs)
9     return np.random.choice(top_p_indices, p=top_p_probs)
```

Here are short descriptions of the NumPy functions involved:

- `np.argsort(probs)`: Returns the indices that would sort the array in ascending order. Example: `np.argsort([0.2, 0.8, 0.5])` returns `[0, 2, 1]`.
- `np.cumsum(sorted_probs)`: Computes the cumulative sum of an array. Example: `np.cumsum([0.8, 0.15, 0.05])` returns `[0.8, 0.95, 1.0]`.
- `np.searchsorted(cumulative_probs, p)`: Finds the smallest index where cumulative probability reaches or exceeds p. Example: `np.searchsorted([0.6, 0.85, 1.0], 0.75)` returns `1`.
- `np.random.choice(top_p_indices, p=top_p_probs)`: Samples an element from `top_p_indices` with probabilities given by `top_p_probs`.
  Example: `np.random.choice([2, 0], p=[0.8, 0.2])` randomly returns `2` or `0` with `2` being more likely.

Question 14
There is a **bug** in this code that makes it not serve the intended purpose. Which of the following best describes the issue?

☒ The computation of `cutoff_idx` is incorrect.

☐ The function does not normalize `top_p_probs` before sampling.

☐ The function does not apply the softmax transformation to `probs` before sampling.

☐ The function incorrectly sorts the probabilities in descending order instead of ascending order.

Solution: `np.searchsorted(cumulative_probs, p)` finds the smallest index where the cumulative probability reaches or exceeds p. As shown in the example, it would return 1 for input `([0.6, 0.85, 1.0], 0.75)`. For this example, in line 6 we would therefore end up cutting off at the first token itself whereas we should have included the second token too. So, `cutoff_idx` is off by 1. There is no other bug and hence other options are incorrect.

N-gram Language Modeling

Given a training set containing the following sequences:

```
"the cat sat on the mat"
"the dog chased after the cat"
"a cat and a dog ran"
"she sat on the sofa"
"they bought a new mat"
"the sofa was comfortable"
```

The training set includes 18 unique tokens and 32 tokens in total. We consider the words split by whitespace as the tokens in the vocabulary. We don't consider adding any special tokens (`<UNK>` or `<stop>`) without specifically claiming. Please answer the following question in the context of N-gram language modeling:

Question 15 What is the probability of the sequence "the cat sat on the sofa" using a unigram (1-gram) language model?

□ None of the choices is correct.

□ $P(\text{the cat sat on the sofa}) = \frac{0}{6} = 0$.

□ $\frac{864}{18^6}$.

□ $\frac{21}{32}$.

■ $\frac{27}{32^5}$.

Solution: $\frac{27}{32^5}$ is correct.
The total number of tokens appear in the training dataset is 32. $P(\text{the}) \times P(\text{cat}) \times P(\text{sat}) \times P(\text{on}) \times P(\text{the}) \times P(\text{sofa}) = \frac{6}{32} \times \frac{3}{32} \times \frac{2}{32} \times \frac{2}{32} \times \frac{6}{32} \times \frac{2}{32} = \frac{27}{32^5}$.

Classification and Dataset Bias

Consider a training set for sentiment classification containing the following sequences:

```
"I liked this love movie because it was exciting" [LABEL=+]
"action movies are always terrible" [LABEL=-]
"this comedy made me laugh" [LABEL=+]
"I disliked the drama because it was too slow" [LABEL=-]
```

We are now conducting sentiment classification by **Naive Bayes** algorithm.

Naive Bayes finds the probability of a label given the probability of the sequences occurred in the training set. Mathematically, it can be defined as:

$$P(c|X) = \frac{P(X|c)P(c)}{P(X)}, \qquad P(X) = \sum_{c \in \{+,-\}} P(X|c)P(c)$$

where c is one of the labels and X is the corresponding sequence.
The training dataset contains 28 tokens in total, including 23 unique tokens. We add one extra token "`<UNK>`" into vocabulary to handle the unseen out-of-vocabulary words. We always apply add-one Laplace smoothing (smoothing factor $\alpha = 1$) to handle zero probabilities. The conditional probability of a single word w given label c can be computed as follows:

$$P(w|c) = \frac{\text{count}(w, c) + \alpha}{\sum_{w_i \in V} \text{count}(w_i, c) + \alpha \cdot |V|}$$

Question 16 Calculate the likelihood ratio $P(\text{action}|+)/P(\text{action}|\text{-})$ for the word "action"

■ 0.50

□ 0.91

□ 0.12

□ 0.40

□ 0.18

□ 0.29

Solution: 0.50 is the correct answer.

$$P(\text{action}|+)/P(\text{action}|\text{-}) = (\text{count}(\text{action}, +) + 1)/(\text{count}(\text{action}, \text{-}) + 1) = \frac{1}{2}$$

Question 17 Using **Naive Bayes** with a **Unigram language model**, what is the probability that the sequence `"I liked this action movie"` is classified as positive?

□ 0.76

□ 0.41

□ 0.23

■ 0.86

□ 0.43

Solution: We need to calculate $P(+|\text{"I liked this action movie"})$ using Naive Bayes and unigram language modeling:

$$\begin{aligned}P(+|X) &= P(X|+)P(+)/P(X)\\ &= P(\text{"I"}|+)P(\text{"liked"}|+,\text{"I"})\dots P(\text{"movie"}|+,\text{"I"},\text{"liked"},\text{"this"},\text{"action"})P(+)/P(X)\\ &= P(\text{"I"}|+)P(\text{"liked"}|+)P(\text{"this"}|+)P(\text{"action"}|+)P(\text{"movie"}|+)P(+)/P(X)\\ &= \frac{1+1}{14+24}\times\frac{1+1}{14+24}\times\frac{2+1}{14+24}\times\frac{0+1}{14+24}\times\frac{1+1}{14+24}\times\frac{1}{2}\times\frac{1}{P(X)}\\ &= \frac{24}{14+24}\times\frac{1}{2}\times\frac{1}{P(X)}\end{aligned}$$

we can then derive $P(X)$ as:

$$\begin{aligned}P(X) &= \frac{1+1}{14+24}\times\frac{1+1}{14+24}\times\frac{2+1}{14+24}\times\frac{0+1}{14+24}\times\frac{1+1}{14+24}\times\frac{1}{2}\\ &+\frac{1+1}{14+24}\times\frac{0+1}{14+24}\times\frac{0+1}{14+24}\times\frac{1+1}{14+24}\times\frac{0+1}{14+24}\times\frac{1}{2}\\ &= \frac{28}{14+24}\times\frac{1}{2}\end{aligned}$$

Then we finally get:

$$\begin{aligned}P(+|X) &= (\frac{24}{14+24}\times\frac{1}{2})/(\frac{28}{14+24}\times\frac{1}{2})\\ &= 24/28\\ &= 6/7\approx 0.86\end{aligned}$$

Language Model Architectures

Please answer the following questions according to Figure 2.

Question 18 Which language model algorithm is described by the computational graph shown in Figure 2?

□ N-gram Language Model

■ Fixed-Window Language Model

□ Transformer with Attention

□ Recurrent Neural Network (RNN)

Solution: The architecture is Fixed-Window Language Model, according to Lecture 2 (EarlyLMs+RNNs-Start) page 56.

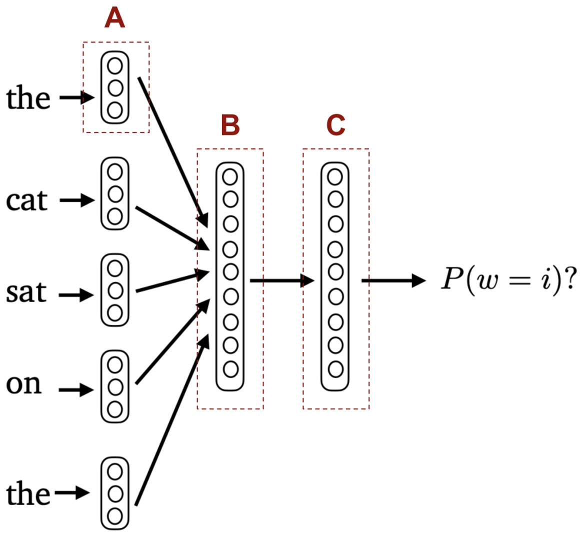

Figure 2. Some Language Model Architecture.

Question 19 Which of the following descriptions are **TRUE** about the language modeling algorithm shown in Figure 2?

(a) It can encode long sequences with long dependencies.

(b) The embedding matrix can be shared between different tokens.

(c) It needs an extra positional embedding layer.

(d) The predicted token probabilities are non-zero.

□ Only (a).
□ None of the other options is True.
□ (a), (c) and (d).
□ Only (b).
□ (a), (b) and (c).
□ Only (c).
□ (a) and (d).
□ (b) and (c).
■ Only (d).
□ (a), (b), (c) and (d).
□ (b) and (d).
□ (c) and (d).

Solution: Only (d) is correct.
(a): the Fixed window language model is still too small to encode long-range dependencies (Lecture 2, EarlyLMs+RNNs-Start, page 60).
(b): the embedding matrix cannot be shared between different tokens in the window (Lecture 2, EarlyLMs+RNNs-Start, page 60).
(c): the Fixed window language model does not need positional embeddings.

Question 20 In the corresponding language model, given that the number of input tokens is 5, the embedding dimension of each individual token is d, and the hidden layer dimension is h, what are the dimensions of the vectors A, B, and C shown in Figure 2?

□ R^{5d}, R^{5d}, R^{5d}

■ R^{d}, R^{5d}, R^{h}

□ R^{d}, R^{3d}, R^{3d}

□ R^{d}, R^{h}, R^{h}

□ R^{d}, R^{5d}, R^{5d}

□ R^{d}, R^{3d}, R^{h}

Solution: The correct choice is R^{d}, R^{5d}, R^{h}, according to Lecture 2 ($EarlyLMs + RNNs - Start$) page 56.