Instruções: Cada questão possui exatamente uma alternativa correta.

Questões

I) Na tarefa de *Named Entity Recognition* (NER) com esquema **BIO**, as categorias são: **B**egin (início de entidade), **I**nside (continuação) e **O**utside. Para a frase

`Lucas trabalha na PUCRS em Porto Alegre`

considerando as entidades *Lucas* (pessoa), *PUCRS* (organização) e *Porto Alegre* (local, dois *tokens*), qual sequência de *tags* é correta?

a) `B-PER, O, O, B-ORG, O, B-LOC, I-LOC`

b) `B-PER, O, O, B-ORG, O, I-LOC, I-LOC`

c) `B-PER, O, O, I-ORG, O, B-LOC, I-LOC`

d) `I-PER, O, O, B-ORG, O, B-LOC, I-LOC`

II) O BERT-Base tem $L = 12$ camadas, dimensão oculta $d = 768$ e $H = 12$ cabeças de atenção com $d_k = d/H = 64$ por cabeça. Em uma única camada de *self-attention*, as projeções são:

$$W^Q,\ W^K,\ W^V \in \mathbb{R}^{d\times(H\cdot d_k)}, \qquad W^O \in \mathbb{R}^{(H\cdot d_k)\times d}.$$

Ignorando vieses, quantos parâmetros treináveis existem nessas **quatro** matrizes em uma única camada?

a) 589 824: apenas W^Q.

b) 1 769 472: W^Q, W^K, W^V sem a projeção de saída W^O.

c) 2 359 296: as quatro matrizes W^Q, W^K, W^V, W^O.

d) 4 718 592: dupla contagem ao considerar dimensões de entrada e saída separadamente para cada projeção.

III) No pré-treino do BERT, a estratégia MLM seleciona aleatoriamente 15% dos *tokens*. Um estudante propõe simplificar a regra 80/10/10 substituindo **100**% dos *tokens* selecionados pelo *token* especial `[MASK]` (em vez de 80% por `[MASK]`, 10% por palavra aleatória e 10% mantidos inalterados). Qual é a principal desvantagem dessa simplificação?

a) O modelo nunca aprende a representar *tokens* reais durante o pré-treino; ao fazer *fine-tuning*, jamais encontra `[MASK]`, criando um desalinhamento entre as duas fases.

b) A perda MLM fica indefinida porque `[MASK]` não faz parte do vocabulário padrão do modelo.

c) Os gradientes da posição `[MASK]` se anulam quando todas as posições selecionadas recebem o mesmo *token*.

d) O modelo aprende a ignorar a posição dos *tokens*, pois `[MASK]` é sempre o mesmo símbolo, independentemente da posição.

IV) A representação de entrada do BERT é $\mathbf{x}_i = E_{\text{token}}(w_i) + E_{\text{segmento}}(s_i) + E_{\text{posição}}(i)$, onde E_{segmento} assume o valor E_A para tokens da primeira sentença e E_B para tokens da segunda. Um pesquisador remove o componente E_{segmento} para simplificar o modelo. Em qual das seguintes tarefas essa remoção causaria a **maior degradação** de desempenho?

a) *POS tagging* em uma única sentença.

b) Classificação de sentimento em uma única sentença (SST-2).

c) *Extractive QA* (SQuAD): identificar o trecho que responde a uma pergunta dado um parágrafo de contexto.

d) *Named Entity Recognition* (NER) em uma única sentença.

V) O ELECTRA (Clark et al., 2020) substitui o objetivo MLM do BERT pelo *Replaced Token Detection* (RTD): um gerador pequeno cria *tokens* substitutos para $\approx 15\%$ das posições, e o discriminador ELECTRA deve classificar **cada posição** da sequência como "real" ou "substituído". Comparado ao BERT com MLM, qual é a principal vantagem do ELECTRA em eficiência de treinamento?

a) O ELECTRA não precisa de uma camada de vocabulário na saída (custo $d \times |V|$), pois a classificação RTD é binária.

b) O ELECTRA gera sinal de gradiente em **todas** as posições da sequência (não apenas nos $\approx 15\%$ mascarados do MLM), obtendo desempenho equivalente com $\approx 25\%$ do *compute*.

c) O ELECTRA não precisa de *fine-tuning*, pois o RTD já otimiza diretamente tarefas discriminativas como NER.

d) O ELECTRA usa *batches* menores que o BERT por precisar do gerador auxiliar, reduzindo o consumo de memória e acelerando o treino por passo.

VI) Um engenheiro extrai $\mathbf{h}^{(L)}_{\texttt{[CLS]}}$ (estado oculto final do `[CLS]`) de um BERT pré-treinado **sem fine-tuning** para medir a similaridade semântica entre sentenças via cosseno. No experimento de Reimers & Gurevych (2019), esse método obtém 20.29 de correlação de Spearman no STS-Benchmark, abaixo do GloVe (58.02). Qual raciocínio explica corretamente esse fenômeno?

a) O `[CLS]` foi treinado no NSP, um objetivo binário e ruidoso que não incentiva a organização do espaço para similaridade cosseno; o espaço resultante é anisotrópico (vetores concentrados em um cone estreito), tornando as distâncias cosseno pouco discriminativas.

b) O BERT usa *position embeddings* aprendidos, logo o `[CLS]` captura apenas informação posicional e perde a semântica do restante da sentença.

c) O `[CLS]` usa apenas os *tokens* da última camada, enquanto o GloVe usa toda a hierarquia de n-grams (mais informação leva ao melhor desempenho).

d) O BERT foi pré-treinado apenas em inglês, e o STS-Benchmark contém sentenças em múltiplos idiomas, favorecendo o GloVe multilíngue.

VII) No SBERT, o *mean pooling* mascarado calcula o vetor de sentença excluindo posições `[PAD]`:

$$\mathbf{u} = \frac{\sum_{i=1}^{L} m_i \, \mathbf{h}_i}{\sum_{i=1}^{L} m_i}, \qquad m_i \in \{0, 1\}.$$

Uma sentença tem $L = 3$ tokens com os seguintes *embeddings* $\mathbf{h}_i \in \mathbb{R}^2$ e máscara de atenção:

Token	$\mathbf{h}_i$	m_i
w_1	$[2,\ 0]$	1
w_2	$[0,\ 2]$	1
w_3 (`[PAD]`)	$[1,\ 3]$	0

Qual é o vetor de sentença $\mathbf{u}$ produzido pelo *mean pooling* mascarado?

a) $[1,\ 1]$: média dos tokens reais w_1 e w_2, excluindo o `[PAD]`.

b) $\left[1,\ \frac{5}{3}\right] \approx [1.00,\ 1.67]$: inclui $\mathbf{h}_3$ do `[PAD]` no numerador e usa $L = 3$ no denominador.

c) $[2,\ 2]$: soma correta dos tokens mascarados, sem dividir por $\sum m_i$.

d) $\left[\frac{2}{3},\ \frac{2}{3}\right] \approx [0.67,\ 0.67]$: exclui o `[PAD]` do numerador, mas divide por $L = 3$ em vez de $\sum m_i = 2$.

VIII) Um sistema de busca semântica precisa classificar $n = 100\,000$ documentos em relação a uma consulta. Um **cross-encoder** requer 1 *forward pass* por par (consulta, documento); um **bi-encoder** pré-computa os documentos *offline* e, no momento da consulta, realiza apenas 1 *forward pass* (para a consulta) seguido de busca vetorial. Cada *forward pass* demora $t = 10$ ms. Qual é a redução de latência no momento da consulta ao usar o bi-encoder?

a) $\approx 100\times$: o bi-encoder ainda precisa de 1 *forward pass* por documento durante a consulta.

b) $\approx 1\,000\times$: eliminando o processamento par-a-par para os primeiros 1 000 documentos filtrados.

c) $\approx 100\,000\times$: a consulta requer apenas 10 ms (um único *forward pass*) em vez de $100\,000 \times 10$ ms $= 1\,000$ s.

d) Equivalente; a busca vetorial requer um produto escalar por documento, igualando o custo do *forward pass*.

IX) Na perda *InfoNCE* com *in-batch negatives*, um *batch* de $N = 4$ pares (x_i, x_i^+) fornece $N - 1 = 3$ negativos por âncora. Um engenheiro aumenta o *batch* para $N = 1\,024$. Qual é o efeito esperado desse aumento?

a) O treino fica mais lento por *token*, mas os gradientes são mais informativos: cada âncora deve distinguir seu positivo de 1 023 competidores, melhorando a qualidade dos *embeddings*.

b) O treino produz resultado equivalente ao de $N = 4$: a perda InfoNCE é normalizada pelo número de negativos e compensa automaticamente o aumento.

c) O treino diverge com N grande porque o denominador do *softmax* fica muito grande, zerando os gradientes.

d) O benefício de N grande é apenas computacional (melhor utilização de GPU), sem impacto na qualidade final dos *embeddings*.

X) ☠️ Considere três *embeddings* (já normalizados) e a *triplet loss* com margem $\alpha = 0.5$:

$$f(x) = [1.0,\ 0.0], \quad f(x^+) = [0.6,\ 0.8], \quad f(x^-) = [-0.6,\ 0.8],$$

$$\mathcal{L}_{\text{triplet}} = \max\big(0,\ \|f(x) - f(x^+)\|_2^2 - \|f(x) - f(x^-)\|_2^2 + \alpha\big).$$

Qual é o valor de $\mathcal{L}_{\text{triplet}}$?

a) 0 (a margem já é satisfeita: o positivo está suficientemente mais próximo que o negativo).

b) 0.50 (a perda iguala exatamente a margem α quando as duas distâncias são iguais).

c) 1.90 (diferença das distâncias quadráticas, sem aplicar o $\max(0, \cdot)$).

d) -1.90 (resultado antes do truncamento por zero).

XI) ☠️ O BERT pré-treina com a perda MLM:

$$\mathcal{L}_{\text{MLM}} = -\frac{1}{|M|} \sum_{i \in M} \log p(w_i \mid \mathbf{w}_{\setminus M}).$$

Em um passo de treino, $|M| = 3$ posições são mascaradas. O modelo atribui as seguintes probabilidades ao *token* correto em cada posição: $p_1 = 0.50$, $p_2 = 0.25$, $p_3 = 0.125$. Qual é o valor de $\mathcal{L}_{\text{MLM}}$?

a) ≈ 0.693

b) ≈ 1.386

c) ≈ 2.079

d) ≈ 4.159

Soluções

I — Resposta: (a)

O esquema BIO exige que o **primeiro token** de qualquer entidade receba a *tag* **B-** (Begin), e os tokens subsequentes da mesma entidade recebam **I-** (Inside). Tokens fora de entidades recebem **O**. Na frase:

- "Lucas" → `B-PER` (inicia a entidade pessoa).
- "PUCRS" → `B-ORG` (inicia a entidade organização).
- "Porto" → `B-LOC` (inicia a entidade local de dois tokens).
- "Alegre" → `I-LOC` (continua a mesma entidade local).

A opção (b) usa `I-LOC` para "Porto" (erro: nunca se inicia uma entidade com `I-`). A opção (c) usa `I-ORG` para "PUCRS" (mesmo erro para organização). A opção (d) usa `I-PER` para "Lucas" (mesmo erro para pessoa).

II — Resposta: (c)

Como $H \cdot d_k = 12 \times 64 = 768 = d$, todas as quatro matrizes são $\mathbb{R}^{768 \times 768}$. O número de parâmetros em cada uma é $768 \times 768 = 589\,824$, e no total:

$$4 \times 589\,824 = 2\,359\,296.$$

A opção (a) conta apenas W^Q. A opção (b) omite a projeção de saída W^O, que transforma a concatenação das cabeças de volta à dimensão d e é tão grande quanto as demais. A opção (d) duplica o resultado correto (erro de dupla contagem).

III — Resposta: (a)

O `[MASK]` é um token **exclusivo do pré-treino**: ele nunca aparece na fase de *fine-tuning* nem na inferência. Se todos os tokens selecionados fossem substituídos por `[MASK]`, o modelo **nunca aprenderia a representar tokens reais**; ao encontrá-los no *fine-tuning*, suas representações seriam subótimas. A regra 80/10/10 mitiga esse desalinhamento ao forçar o modelo a lidar com tokens reais (intactos ou aleatórios) mesmo durante o pré-treino. A opção (b) é falsa: `[MASK]` é um token do vocabulário do BERT. A opção (c) é falsa: posições com o mesmo *token* ainda produzem gradiente distinto, pois os estados ocultos dependem do contexto. A opção (d) confunde mascaramento com ignorância de posição; a *position embedding* é somada antes de qualquer mascaramento.

IV — Resposta: (c)

No formato SQuAD, a entrada é `[CLS] pergunta [SEP] contexto [SEP]`, com os tokens da pergunta recebendo E_A e os do contexto recebendo E_B. Sem *segment embeddings*, o modelo **não distingue** quais tokens são a pergunta e quais são o contexto, degradando drasticamente a tarefa de identificação de *spans*. As tarefas (a), (b) e (d) operam em uma única sentença, portanto todos os tokens compartilhariam E_A: a remoção de E_{segmento} não altera o comportamento nessas tarefas.

V — Resposta: (b)

No MLM do BERT, o gradiente só flui pelas $\approx$ 15% das posições mascaradas. No ELECTRA, o discriminador classifica **todas** as L posições ("real" ou "substituído"), gerando gradiente **denso** a cada passo. Esse sinal mais rico permite que o ELECTRA alcance o mesmo desempenho do BERT com $\approx$ 25% do *compute* de pré-treino. A opção (a) é parcialmente verdadeira (a cabeça de saída é binária), mas a camada de vocabulário do gerador ainda existe; a vantagem principal é o sinal denso. A opção (c) é falsa: o ELECTRA ainda requer *fine-tuning*. A opção (d) inverte o raciocínio: o gerador aumenta o custo por passo, mas o ganho em eficiência vem do sinal mais informativo.

VI — Resposta: (a)

O `[CLS]` é treinado no NSP (classificação binária "é ou não é a próxima sentença"), sem nenhum incentivo para estruturar o espaço de *embeddings* de forma que **frases similares fiquem próximas**. O resultado é um espaço **anisotrópico**: todos os vetores se concentram em um cone estreito do hiperespaço, tornando as distâncias cosseno pouco discriminativas (frases totalmente diferentes podem ter cosseno > 0.9). O GloVe, embora estático, captura co-ocorrências e produz um espaço mais isotrópico, o que o torna superior para STS *vanilla*. A opção (b) confunde *position embeddings* com o objetivo do `[CLS]`. A opção (c) inventa um mecanismo inexistente. A opção (d) é falsa: o BERT original foi pré-treinado em inglês, assim como o GloVe mais comum, e o STS-Benchmark é predominantemente em inglês.

VII — Resposta: (a)

A máscara zera a contribuição de w_3; o numerador é $m_1\mathbf{h}_1 + m_2\mathbf{h}_2 + m_3\mathbf{h}_3 = [2,\ 0] + [0,\ 2] + \mathbf{0} = [2,\ 2]$ e o denominador é $\sum m_i = 1 + 1 + 0 = 2$:

$$\mathbf{u} = \frac{[2,\ 2]}{2} = [1,\ 1].$$

A opção (b) inclui $\mathbf{h}_3 = [1,\ 3]$ do `[PAD]` no numerador e usa $L = 3$: $([2, 0] + [0, 2] + [1, 3])/3 = [3, 5]/3 = [1,\ \frac{5}{3}]$: resultado distorcido pela posição de *padding*. A opção (c) aplica corretamente a máscara ao numerador, mas omite a divisão por $\sum m_i$, produzindo a soma bruta $[2,\ 2]$. A opção (d) exclui o `[PAD]` do numerador (correto), mas divide por $L = 3$ em vez de $\sum m_i = 2$, gerando $[2,\ 2]/3 = [\frac{2}{3},\ \frac{2}{3}]$: erro de normalização.

VIII — Resposta: (c)

O cross-encoder requer 1 *forward pass* por par, logo $n = 100\,000$ passes no momento da consulta:

$$100\,000 \times 10 \text{ ms} = 1\,000\,000 \text{ ms} = 1\,000 \text{ s}.$$

O bi-encoder pré-computa todos os documentos *offline*. Na consulta, basta 1 *forward pass* para a *query* (10 ms) seguido de busca vetorial (produto escalar em hardware SIMD, $\ll 1$ ms para 10^5 vetores). Redução de latência:

$$\frac{1\,000\,000 \text{ ms}}{10 \text{ ms}} = 100\,000\times.$$

A opção (a) erra ao supor que o bi-encoder ainda faz um *forward pass* por documento em tempo de consulta (documentos estão pré-computados). A opção (b) chega a 1 000× por confundir $n = 100\,000$ com $n = 1\,000$. A opção (d) subestima a diferença de velocidade entre um produto escalar e um *forward pass* completo por um transformer (fator de $\sim 10^4$–10^6).

IX — Resposta: (a)

Na perda InfoNCE, para cada âncora x_i o positivo x_i^+ deve ser identificado entre todos os x_j^+ do *batch*:

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{\text{sim}(f(x_i),\, f(x_i^+))/\tau}}{\sum_{j=1}^{N} e^{\text{sim}(f(x_i),\, f(x_j^+))/\tau}}.$$

Com $N = 1\,024$, cada âncora compete com 1 023 negativos simultâneos, gerando gradiente mais rico e discriminativo. Empiricamente, o desempenho escala com $\log N$, motivando *batches* de 1 024–16 384 em modelos SOTA. A opção (b) erra: a normalização pelo N no denominador da média **não** é equivalente a neutralizar o efeito de mais negativos: o denominador do *softmax* cresce com N, tornando o problema de classificação mais difícil. A opção (c) erra: o *softmax* com denominador grande concentra gradiente nos negativos mais difíceis, não o zera. A opção (d) erra: mais negativos melhoram a qualidade, não apenas a eficiência.

X — Resposta: (a)

Cálculo das distâncias quadráticas:

$$\|f(x) - f(x^+)\|_2^2 = (1.0 - 0.6)^2 + (0.0 - 0.8)^2 = 0.16 + 0.64 = 0.80,$$
$$\|f(x) - f(x^-)\|_2^2 = (1.0 - (-0.6))^2 + (0.0 - 0.8)^2 = 2.56 + 0.64 = 3.20.$$

Aplicando a fórmula:

$$\mathcal{L}_{\text{triplet}} = \max(0,\ 0.80 - 3.20 + 0.50) = \max(0,\ -1.90) = 0.$$

O positivo ($d^2 = 0.80$) já está 2.40 unidades mais próximo que o negativo ($d^2 = 3.20$), excedendo a margem $\alpha = 0.50$: o par não viola a restrição e não gera gradiente. A opção (b) ocorreria se as duas distâncias fossem iguais ($d^+ = d^-$); nesse caso $\mathcal{L} = \alpha = 0.50$. A opção (c) é o argumento antes do $\max(0, \cdot)$. A opção (d) é o mesmo sem o valor absoluto.

XI — Resposta: (b)

As probabilidades dos tokens corretos são $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{4}$, $p_3 = \frac{1}{8}$. Aplicando a fórmula com $|M| = 3$:

$$\begin{aligned}\mathcal{L}_{\text{MLM}} &= -\frac{1}{3}\left(\ln\tfrac{1}{2} + \ln\tfrac{1}{4} + \ln\tfrac{1}{8}\right) \\ &= -\frac{1}{3}\left(-\ln 2 - 2\ln 2 - 3\ln 2\right) \\ &= \frac{6\ln 2}{3} = 2\ln 2 \approx 1.386.\end{aligned}$$

Análise das alternativas incorretas:

- *(a)* ≈ 0.693: considera apenas $-\ln(0.50) = \ln 2$, ou seja, somente a primeira posição, sem a média.
- *(c)* ≈ 2.079: usa $|M| = 2$ no denominador ($6\ln 2/2 = 3\ln 2$), como se apenas duas posições fossem mascaradas.
- *(d)* ≈ 4.159: omite a normalização $1/|M|$ por completo ($-(\ln\frac{1}{2} + \ln\frac{1}{4} + \ln\frac{1}{8}) = 6\ln 2$).