# TA4 Actividades de introducción a los LLMs

**Índice**

# 1 Introducción

☐ Los foundation models (FMs) son modelos pre-entrenados con datos masivos para poder aplicarlos en muchos casos de uso. En relación con ellos, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. Ejemplos típicos de FMs son los large language models (LLMs) que responden a prompts en lenguaje natural generando texto, imágenes, audio, vídeo, código, etc.
2. Solo grandes labs como Google, Meta, Alibaba y OpenAI disponen de recursos suficientes para entrenar FMs avanzados.
3. Los FMs frontera son modelos avanzados con capacidades peligrosas para la seguridad pública.
4. Las tres opciones anteriores son correctas.

**Solución:** La 4

---

# 2 Antecedentes

☐ Sea un LM de $n$-gramas de palabras mínimamente perplejo. En relación con la perplejidad y entropía del mismo, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. Solo una única palabra puede seguir a $n-1$ previas (con probabilidad $1$)
2. El modelo concentra toda la probabilidad en una sola frase
3. La perplejidad del modelo es $1$ y su entropía $0$
4. Las tres opciones anteriores son correctas

**Solución:** La 4

---

☐ Sea un LM de $n$-gramas de palabras máximamente perplejo. En relación con la perplejidad y entropía del mismo, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. Cualquier palabra (salvo $<\mathrm{s}>$) puede seguir a $n-1$ previas con probabilidad $1/V$ ($V$ no incluye $<\mathrm{s}>$)
2. La probabilidad asignada a una frase $w_{1:N}$ arbitraria, $N \geq 0$, es $1/(V-1)^N \cdot 1/V \approxeq V^{-N}$
3. La perplejidad del modelo es $V$ (aprox.) y su entropía $\log V$
4. Las tres opciones anteriores son correctas

**Solución:** La 4

---

☐ Sea el LM bigrama:

$$p(\texttt{a} \mid \texttt{<s>}) = 1 \qquad p(\texttt{a} \mid \texttt{a}) = 0.9,\, p(\texttt{b} \mid \texttt{a}) = 0.1 \qquad p(\texttt{b} \mid \texttt{b}) = 0.9,\, p(\texttt{</s>} \mid \texttt{b}) = 0.1$$

La probabilidad $P$ de `ab` según el mismo es:

1. $P < 10^{-5}$
2. $10^{-5} \leq P < 10^{-4}$
3. $10^{-4} \leq P < 10^{-3}$
4. $10^{-3} \leq P$

**Solución:** La 4; $P = p(\texttt{ab}) = p(\texttt{a} \mid \texttt{<s>})\, p(\texttt{b} \mid \texttt{a})\, p(\texttt{</s>} \mid \texttt{b}) = 1 \cdot 0.1 \cdot 0.1 = 0.01$

---

☐ Sea el LM bigrama:

$$p(\texttt{a} \mid \texttt{<s>}) = 1 \qquad p(\texttt{a} \mid \texttt{a}) = 0.9,\, p(\texttt{b} \mid \texttt{a}) = 0.1 \qquad p(\texttt{b} \mid \texttt{b}) = 0.9,\, p(\texttt{</s>} \mid \texttt{b}) = 0.1$$

Sea $P = \mathrm{Perplexity}(\texttt{ab})$ la perplejidad del mismo hallada a partir de `ab`. Se cumple:

1. $P < 1$
2. $1 \leq P < 2$
3. $2 \leq P < 3$
4. $3 \leq P$

**Solución:** La 4; $P = p(\texttt{ab})^{-1/3} = 0.01^{-1/3} = 4.64$

---

**Problema.** Sea $w^{(1)}_{1:N_1}$ una secuencia de $N_1$ palabras de un vocabulario de talla $V$, $\mathcal{V} = \{1, \ldots, V\}$. Se quiere aprender un LM de $n$-gramas de palabras, $\hat{p}$, mediante minimización de la entropía cruzada del modelo $\hat{p}$ relativa a la referencia dada por $w^{(1)}_{1:N_1}$, $p^{(1)}$ :

$$\begin{aligned}\mathbb{H}(p^{(1)}, \hat{p}) &= -\sum_{i=1}^{N_1} p(w_i^{(1)} \mid w^{(1)}_{i-(n-1):i-1}) \log \hat{p}(w_i^{(1)} \mid w^{(1)}_{i-(n-1):i-1}) \\ &= -\sum_{w_{1:n}} \sum_{i=1}^{N_1} \mathbb{I}(w_{1:n} = w^{(1)}_{i-(n-1):i}) \log \hat{p}(w_n \mid w_{1:n-1}) \\ &= -\sum_{w_{1:n}} N(w_{1:n} \in w^{(1)}_{1:N_1}) \log \hat{p}(w_n \mid w_{1:n-1})\end{aligned}$$

La minimización se sujeta a que, para cada historia de $n-1$ palabras observada (en $w^{(1)}_{1:N_1}$), el modelo debe definir una distribución de probabilidad propiamente dicha sobre el vocabulario:

$$\sum_{w \in \mathcal{V}} \hat{p}(w \mid w_{1:n-1}) = 1 \qquad \text{para todo} \quad w_{1:n-1} \in w^{(1)}_{1:N_1}$$

Obviamente, las historias terminadas en el símbolo de fin de frase no se tienen en cuenta. Demuestra que, para cada historia observada $w_{1:n-1}$, la distribución de mínima entropía se obtiene por conteo simple como:

$$\hat{p}(w \mid w_{1:n-1}) = \frac{N(w_{1:n-1} w \in w^{(1)}_{1:N_1})}{\sum_{\tilde{w} \in \mathcal{V}} N(w_{1:n-1} \tilde{w} \in w^{(1)}_{1:N_1})}$$

**Solución:** introducimos un multiplicador de Lagrange $\lambda_{w_{1:n-1}}$ por cada historia observada

$$\mathcal{L}(\hat{p}, \boldsymbol{\lambda}) = \mathbb{H}(p^{(1)}, \hat{p}) - \sum_{w_{1:n-1} \in w^{(1)}_{1:N_1}} \lambda_{w_{1:n-1}} \left(1 - \sum_{w \in \mathcal{V}} \hat{p}(w \mid w_{1:n-1})\right)$$

$$\frac{\partial \mathcal{L}}{\partial \hat{p}(w \mid w_{1:n-1})} = \frac{N(w_{1:n-1} w \in w^{(1)}_{1:N_1})}{\hat{p}(w \mid w_{1:n-1})} - \lambda_{w_{1:n-1}} = 0 \;\rightarrow\; \hat{p}(w \mid w_{1:n-1}) = \frac{N(w_{1:n-1} w \in w^{(1)}_{1:N_1})}{\lambda_{w_{1:n-1}}}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{w_{1:n-1}}} = 1 - \sum_{w \in \mathcal{V}} \hat{p}(w \mid w_{1:n-1}) = 0 \;\rightarrow\; \lambda_{w_{1:n-1}} = \sum_{w \in \mathcal{V}} N(w_{1:n-1} w \in w^{(1)}_{1:N_1})$$

$$\hat{p}(w \mid w_{1:n-1}) = \frac{N(w_{1:n-1} w \in w^{(1)}_{1:N_1})}{\sum_{\tilde{w} \in \mathcal{V}} N(w_{1:n-1} \tilde{w} \in w^{(1)}_{1:N_1})}$$

**Problema.** Obtén el LM bigrama $\hat{p}$ de mínima entropía cruzada relativa a la referencia $p^{(1)}$ dada por $w^{(1)}_{1:N_1} = \texttt{aaaaaaaaaabbbbbbbbbb}$

**Solución:**

$$\hat{p}(\texttt{a} \mid \texttt{<s>}) = \frac{N(\texttt{<s>a} \in w^{(1)}_{1:N_1})}{N(\texttt{<s></s>} \in w^{(1)}_{1:N_1}) + N(\texttt{<s>a} \in w^{(1)}_{1:N_1}) + N(\texttt{<s>b} \in w^{(1)}_{1:N_1})} = 1$$

$$\hat{p}(\texttt{b} \mid \texttt{<s>}) = 0$$
$$\hat{p}(\texttt{</s>} \mid \texttt{<s>}) = 0$$
$$\hat{p}(\texttt{a} \mid \texttt{a}) = 0.9$$
$$\hat{p}(\texttt{b} \mid \texttt{a}) = 0.1$$
$$\hat{p}(\texttt{</s>} \mid \texttt{a}) = 0$$
$$\hat{p}(\texttt{b} \mid \texttt{b}) = 0.9$$
$$\hat{p}(\texttt{</s>} \mid \texttt{b}) = 0.1$$

In [ ]:
```
import nltk; from itertools import tee
s = 'aaaaaaaaaabbbbbbbbbb'
bg = nltk.bigrams(s, pad_right=True, pad_left=True, left_pad_symbol="<s>", right_pad_symbol="</s>")
bg, copy = tee(bg); print(f"Bigrams: {list(copy)}")
voc = nltk.lm.Vocabulary(['a', ' ', 'b', '<s>', '</s>']); print(f"Voc: {list(voc)}")
lm = nltk.lm.MLE(2); lm.fit([bg], voc)
```

```
Bigrams: [('<s>', 'a'), ('a', 'a'), ('a', 'a'), ('a', 'a'), ('a', 'a'), ('a', 'a'), ('a', 'a'), ('a', 'a'),
('a', 'a'), ('a', 'a'), ('a', 'b'), ('b', 'b'), ('b', 'b'), ('b', 'b'), ('b', 'b'), ('b', 'b'), ('b', 'b'),
('b', 'b'), ('b', 'b'), ('b', 'b'), ('b', '</s>')]
Voc: ['a', ' ', 'b', '<s>', '</s>', '<UNK>']
```

In [ ]:
```
lm.counts[['a']]
```

Out[ ]:
```
FreqDist({'a': 9, 'b': 1})
```

In [ ]:
```
paGs = lm.score("a", ["<s>"]); print(paGs)
paGa = lm.score("a", ["a"]); print(paGa)
pbGa = lm.score("b", ["a"]); print(pbGa)
psGb = lm.score("</s>", ["b"]); print(psGb)
paab = paGs * paGa * pbGa * psGb; print(f"{paab:.4f}")
```

```
1.0
0.9
0.1
0.1
0.0090
```

**Problema.** Sea $\mathcal{D} = \{w^{(1)}, \ldots, w^{(N)}\}$ con $w^{(k)} = w^{(k)}_{1:N_k}$ una colección de $N$ secuencias de palabras de un vocabulario de talla $V$, $\mathcal{V} = \{1, \ldots, V\}$. Se quiere aprender un LM de $n$-gramas de palabras, $\hat{p}$, mediante minimización de la entropía cruzada del modelo $\hat{p}$ relativa a la referencia dada por $\mathcal{D}, p$:

$$\begin{aligned}\mathbb{H}(p, \hat{p}) &= \sum_{k=1}^{N} \mathbb{H}(p^{(k)}, \hat{p}) \\ &= -\sum_{w_{1:n}} \sum_{k=1}^{N} N(w_{1:n} \in w^{(k)}_{1:N_k}) \log \hat{p}(w_n \mid w_{1:n-1}) \\ &= -\sum_{w_{1:n}} N(w_{1:n} \in \mathcal{D}) \log \hat{p}(w_n \mid w_{1:n-1})\end{aligned}$$

La minimización se sujeta a que, para cada historia de $n-1$ palabras observada (en $\mathcal{D}$), el modelo debe definir una distribución de probabilidad propiamente dicha sobre el vocabulario:

$$\sum_{w \in \mathcal{V}} \hat{p}(w \mid w_{1:n-1}) = 1 \qquad \text{para todo} \quad w_{1:n-1} \in \mathcal{D}$$

Obviamente, las historias terminadas en el símbolo de fin de frase no se tienen en cuenta. Demuestra que, para cada historia observada $w_{1:n-1}$, la distribución de mínima entropía se obtiene por conteo simple como:

$$\hat{p}(w \mid w_{1:n-1}) = \frac{N(w_{1:n-1} w \in \mathcal{D})}{\sum_{\tilde{w} \in \mathcal{V}} N(w_{1:n-1} \tilde{w} \in \mathcal{D})}$$

**Solución:** generalización inmediata del caso con una única secuencia de aprendizaje; esto es, introducimos un multiplicador de Lagrange $\lambda_{w_{1:n-1}}$ por cada historia observada

$$\mathcal{L}(\hat{p}, \boldsymbol{\lambda}) = \mathbb{H}(p, \hat{p}) - \sum_{w_{1:n-1} \in \mathcal{D}} \lambda_{w_{1:n-1}} \left(1 - \sum_{w \in \mathcal{V}} \hat{p}(w \mid w_{1:n-1})\right)$$

$$\frac{\partial \mathcal{L}}{\partial \hat{p}(w \mid w_{1:n-1})} = \frac{N(w_{1:n-1} w \in \mathcal{D})}{\hat{p}(w \mid w_{1:n-1})} - \lambda_{w_{1:n-1}} = 0 \;\rightarrow\; \hat{p}(w \mid w_{1:n-1}) = \frac{N(w_{1:n-1} w \in \mathcal{D})}{\lambda_{w_{1:n-1}}}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{w_{1:n-1}}} = 1 - \sum_{w \in \mathcal{V}} \hat{p}(w \mid w_{1:n-1}) = 0 \;\rightarrow\; \lambda_{w_{1:n-1}} = \sum_{w \in \mathcal{V}} N(w_{1:n-1} w \in \mathcal{D})$$

$$\hat{p}(w \mid w_{1:n-1}) = \frac{N(w_{1:n-1} w \in \mathcal{D})}{\sum_{\tilde{w} \in \mathcal{V}} N(w_{1:n-1} \tilde{w} \in \mathcal{D})}$$

*Nota:* con el fin de regularizar el modelo, se suelen usar técnicas de suavizado que evitan probabilidades nulas; p. ej. **addone** (a cada cuenta)

$$\hat{p}(w \mid w_{1:n-1}) = \frac{1 + N(w_{1:n-1} w \in \mathcal{D})}{V + \sum_{\tilde{w} \in \mathcal{V}} N(w_{1:n-1} \tilde{w} \in \mathcal{D})}$$

# 3 GPT, BERT, T5 y ChatGPT

☐ En relación con la arquitectura de los LLMs pioneros, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. GPT (1, 2 y 3) es un decoder-only transformer
2. BERT es un encoder-only transformer
3. T5 es un encoder-decoder transformer
4. Las tres opciones anteriores son correctas

**Solución:** La 4

---

☐ ChatGPT es una versión refinada de InstructGPT, un LLM desarrollado por OpenAI en marzo de 2022 mediante fine-tuning de GPT-3 para alinearlo con la intención del usuario. Se aplican tres pasos: fine-tuning supervisado (SFT), modelo de recompensa (RM) y aprendizaje por refuerzo (RL) con PPO(-ptx). En relación los mismos, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. SFT requiere datos de demostración; esto es, respuestas deseadas por el usuario a prompts de un dataset
2. RM require datos de preferencia; esto es, rankings de preferencia sobre posibles respuestas a prompts
3. RL emplea el RM para ajustar GPT-3 tras SFT
4. Las tres opciones anteriores son correctas

**Solución:** La 4

---

# 4 Pre-entrenamiento

☐ En relación con el preproceso para el pre-entrenamiento de LLMs, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. El preproceso preliminar consiste en adquirir y filtrar (limipiar) texto, p. ej. duplicado y tóxico
2. El preproceso básico consiste en tokenizar el texto con base en un vocabulario de tokens de tamaño adecuado
3. La construcción de un vocabulario de tokenización se suele hacer con técnicas incrementales estándar, por ejemplo byte-pair encoding (BPE), hasta alcanzar un tamaño dado
4. Las tres opciones anteriores son correctas

**Solución:** La 4

---

☐ Se está aplicando byte-pair encoding (BPE) para construir un vocabulario a partir de `aaaccaaabdc`. La primera fusión del vocabulario base, `a b c d`, es:

1. `aa`
2. `ab`
3. `ac`
4. Ninguna de las anteriores

**Solución:** La 2; `aa` aparece $4$ veces, más que cualquier otro par de tokens consecutivos.

In [ ]:
```
from tokenizers import Tokenizer, models, trainers, pre_tokenizers
corpus = ["aaaccaaabdc"]
tokenizer = Tokenizer(models.BPE())
trainer = trainers.BpeTrainer(vocab_size=5)
tokenizer.train_from_iterator(corpus, trainer)
vocab = tokenizer.get_vocab(); sorted_vocab = sorted(vocab.items(), key=lambda x: x[1])
print(sorted_vocab); print(tokenizer.model)
```

```
[('a', 0), ('b', 1), ('c', 2), ('d', 3), ('aa', 4)]
BPE(dropout=None, unk_token=None, continuing_subword_prefix=None, end_of_word_suffix=None, fuse_unk=False, byte
_fallback=False, ignore_merges=False, vocab={"a":0, "b":1, "c":2, "d":3, "aa":4}, merges=[("a", "a")])
```

---

☐ La tokenización de `aaaccaaabdc` con el vocabulario BPE `a b c d aa` es:

1. `a a a c c a a a b d c`
2. `aa a c c aa a b d c`
3. `aaa c c aaa b d c`
4. Ninguna de las anteriores

**Solución:** La 2. Tras tokenizar el texto con el vocabulario base, obtenemos la opción 1. Luego aplicamos la regla de fusión que da lugar al token `aa` y obtenemos la opción 2.

In [ ]:
```
print(tokenizer.encode("aaaccaaabdc").tokens)
```

```
['aa', 'a', 'c', 'c', 'aa', 'a', 'b', 'd', 'c']
```

---

☐ Se está aplicando WordPiece para construir un vocabulario a partir de `aabb`. La primera fusión del vocabulario base, `a b ##a ##b`, es:

1. `aa = a ##a`
2. `##ab = ##a ##b`
3. `##bb = ##b ##b`
4. Ninguna de las anteriores

**Solución:** La 1

In [ ]:
```
from tokenizers import Tokenizer, models, trainers, pre_tokenizers
corpus = ["aabb"]
tokenizer = Tokenizer(models.WordPiece())
trainer = trainers.WordPieceTrainer(vocab_size=5)
tokenizer.train_from_iterator(corpus, trainer)
vocab = tokenizer.get_vocab(); sorted_vocab = sorted(vocab.items(), key=lambda x: x[1])
print(sorted_vocab); print(tokenizer.model)
```

```
[('a', 0), ('b', 1), ('##a', 2), ('##b', 3), ('aa', 4)]
WordPiece(unk_token="[UNK]", continuing_subword_prefix="##", max_input_chars_per_word=100, vocab={"a":0, "b":1,
"##a":2, "##b":3, "aa":4})
```

---

☐ La tokenización de `aabaa` con el vocabulario WordPiece `a b ##a ##b aa` es:

1. `a ##a ##b ##a ##a`
2. `aa ##b ##a ##a`
3. `aa ##b aa`
4. Ninguna de las anteriores

**Solución:** La 2, tras tokenizar el prefijo inicial `aa`, el sufijo restante se tokeniza carácter a carácter.

In [ ]:
```
print(tokenizer.encode("aabaa").tokens)
```

```
['aa', '##b', '##a', '##a']
```

---

☐ En relación con el pre-entrenamiento de LLMs recientes, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. Se emplean decenas de trillones de datos y la tendencia es creciente
2. Contienen entre el billón y trillón de parámetros; la tendencia es estable
3. Cuesta entre entre $10^{19}$ y $10^{27}$ FLOPs entrenarlos y la tendencia es creciente
4. Las tres opciones anteriores son correctas

**Solución:** La 4

---

# 5 Post-entrenamiento

☐ El post-entrenamiento de LLMs tiene por objetivo el fine-tuning de un modelo pre-entrenado para adaptarlo según convenga. Con el fin de adaptarlos a conversar en un dominio específico, una aproximación usual consiste en aplicar los pasos descritos en las opciones 1 a 3, en el orden dado. En relación con estos pasos, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. Supervised fine-tuning (SFT): Entrenamiento muy similar al del pre-entrenamiento, aunque con muchos menos datos. Se trata de datos de demostración con respuestas deseadas por el usuario a prompts de un dataset de prompts.
2. Reward model (RM): El modelo SFT se muestrea para obtener dos o más respuestas por prompt y se establece un ranking entre las mismas según preferencias del usuario. Los datos de preferencia obtenidos se emplean para entrenar un modelo de recompensa construido mediante sustitución de la cabeza del modelo SFT por otra para clasificar según preferencias de usuario.
3. Reinforcement learning from human feedback (RLHF): El modelo SFT se ajusta mediante aprendizaje por refuerzo a partir de recompensas dadas por el modelo de recompensa a sus respuestas.
4. Las tres opciones anteriores son correctas.

**Solución:** La 4

---

☐ La aproximación básica al post-entrenamiento de LLMs consta de tres pasos básicos: 1) supervised fine-tuning (SFT) del model pre-entrenado con datos de demostración; 2) entrenamiento de un modelo de recompensa con datos de preferencia; y 3) reinforcement learning from human feedback (RLHF) para ajustar el modelo SFT a partir de recompensas dadas por el modelo de recompensa a sus respuestas. Ahora bien, se han propuesto numerosos refinamientos de esta aproximación con el fin de reducir su coste computacional y mejorar el rendimiento de los modelos. En relación con estos refinamientos, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. Direct preference optimisation (DPO): versión simplificada de RLHF que reduce los pasos 2 y 3 a uno solo.
2. Parameter-efficient fine-tuning (PEFT): entrenamiento eficiente de parámetros; unos "pocos" en lugar de todos.
3. Compresión, cuantización, poda y knowledge distillation: reducción del tamaño del modelo sujeta a mínima degradación.
4. Las tres opciones anteriores son correctas.

**Solución:** La 4

---

# 6 Evaluación

☐ El boom de la IA de los últimos años ha despertado gran interés por la evaluación y comparación de los LLMs más avazados, típicamente desarrollados por grandes empresas tecnológicas como Google, OpenAI, Anthropic, Meta, DeepSeek, etc. De hecho, el creciente impacto socio-económico de estos modelos ha estimulado el desarrollo y popularización de sitios web de evaluación y comparación como Epoch AI, Artificial Analysis, LMArena y OpenCompass. En relación con estos sitios web, indica la opción incorrecta o la última opción si las tres primeras son correctas:

1. Epoch AI es un instituto de investigación sin ánimo de lucro; publica datos e informes exhaustivos que incluyen resultados de benchmarking propios, tendencias y predicciones
2. Artificial Analysis es una consultora: escoges modelo apropiado para tu caso de uso y luego proveedor de inferencia
3. LMArena y OpenCompass son plataformas orientadas a la evaluación colectiva y abierta
4. Las tres opciones anteriores son correctas

**Solución:** La 4

---