Clase 139 — Generación de texto char-RNN

Parte: 2 — Deep Learning · Fuente: Géron, cap. 16 § Generating Shakespearean Text Using a Character RNN. Duración estimada: 70 min.

Clase 139 — Generación de texto char-RNN

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 16 § Generating Shakespearean Text Using a Character RNN. Duración estimada: 70 min.*

Objetivo

Construir un modelo de lenguaje autoregresivo a nivel carácter — el ejercicio canónico de Karpathy 2015 sobre Shakespeare. Entender next-token prediction como tarea de pre-training (la base de todo LLM moderno), sampling con temperatura, y por qué char-RNN fue importante históricamente aunque hoy se hace con tokens BPE y Transformers.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Construir un vocabulario de caracteres (stoi, itos dicts).
- Generar samples (window, target) donde el target es el siguiente carácter.
- Entrenar un modelo Embedding → LSTM → Dense(vocab_size) con cross-entropy.
- Implementar sampling autoregresivo: softmax → multinomial → next char → feed back.
- Aplicar temperatura: logits / T antes de softmax — bajo T = más determinista, alto T = más random.

Temas

- Tokenización a nivel carácter vs word vs BPE.
- Stateful vs stateless RNN: stateful=True para mantener h entre batches.
- Loss: cross-entropy sobre vocab_size.
- Sampling: greedy vs categorical multinomial vs top-k vs nucleus.
- Temperatura como control de creatividad.

Definiciones y características

- Next-token prediction: predecir x_{t+1} dado x_1, ..., x_t. Self-supervised.
- Vocabulario: lista única de caracteres del corpus.
- Stateful RNN: hidden state persiste entre batches consecutivos. Útil para corpus largo.
- Sampling temperatura: p_i = softmax(logits / T). T < 1 más confiado; T > 1 más diverso.
- Top-k sampling: solo considerar los k tokens más probables. Default moderno con k=40-50.

Dataset / recursos

- Tiny Shakespeare (~1 MB de obras): https://raw.githubusercontent.com/karpathy/char-rnn/master/data/tinyshakespeare/input.txt
- Librerías: tensorflow, keras, numpy.

Ejercicios

1. Vocab + encoding: tokenizar el texto a ints. Reportar vocab_size.
2. Modelo: Embedding(vocab_size, 64) → GRU(128, return_sequences=True) → Dense(vocab_size).
3. Train: pasar batches de longitud 100; loss = sparse_categorical_crossentropy.
4. Sample: implementar función que genera N caracteres con T=1.0, T=0.5, T=1.5. Comparar outputs.
5. Top-k: implementar np.argpartition(logits, -k) antes de muestrear.

Homework verificable

Generar Shakespeare:

1. Entrenar Embedding(64) → GRU(256, return_sequences=True, stateful=True) → Dense(vocab_size) por 10 épocas.
2. Generar 1000 caracteres comenzando con "ROMEO:" a temperaturas 0.3, 0.7, 1.2.
3. Reportar val_loss y muestras de output.

Criterio de aceptación: val_loss < 1.5 (vs ~3.5 random); las muestras a T=0.7 tienen palabras reconocibles y estructura de obra teatral.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Output es gibberish total	Modelo no entrena (loss alta) o T demasiad
Output es siempre la misma frase	T = 0 o muy bajo. Fix: T = 0.5 - 1.0.
OOM al generar 10 000 chars	Stateful + sin truncar window. Fix: usar w
argmax greedy → loops ("the the the the")	Greedy sin diversidad. Fix: sampling con t
Vocabulario incluye chars raros que aparec	Aumenta vocab innecesariamente. Fix: filtr

Preguntas frecuentes

¿char-RNN vs word-RNN vs BPE?

Char es simple, no tiene problema de OOV pero los outputs son largos. Word es eficiente pero pierde con palabras nuevas. BPE / SentencePiece (clase 127) es el estándar moderno.

¿Esto es un "LLM"?

Conceptualmente sí: next-token prediction. La diferencia con GPT es escala (parámetros, datos) y arquitectura (Transformer vs RNN). Mismo objetivo.

¿Sampling estrategias avanzadas?

Top-k, nucleus / top-p (Holtzman et al. 2020), beam search. Default moderno: top-p con p=0.9 + temperatura.

¿Cómo evaluar generación?

Perplexity (exp(loss)) automática. Para calidad subjetiva, evaluadores humanos o métricas como BLEU, ROUGE, BERTScore.

¿Char-RNN sirve para algo en 2026?

Sí: NER muy específico, dominios con vocabulario muy chico, o como baseline. Para generación de texto serio → Transformer + BPE.

Referencias

- Géron, cap. 16 — Generating Shakespearean Text.
- Karpathy (2015), The Unreasonable Effectiveness of Recurrent Neural Networks (blog).
- Holtzman et al. (2020), The Curious Case of Neural Text Degeneration (nucleus sampling), ICLR.

Siguiente clase

Clase 140 — Análisis de sentimiento

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb