Generación de Cuentos Cortos con LSTM

Impacto de Parámetros y Corpus

Gabriel Muñoz Marcelo Callisaya

29 de octubre de 2025

Enfoque del proyecto

- Inicialmente: crear un generador interactivo simple (`cuentos.py`) con un modelo LSTM **bueno**.
- Observación clave: pequeñas variaciones en hiperparámetros producían cambios drásticos en coherencia y calidad.
- Decisión: pivotar hacia un **análisis sistemático** del impacto de parámetros y corpus.
- Resultado: 36 configuraciones, automatización y evaluación comparativa.

Resumen

- Sistema LSTM para generar cuentos cortos a partir de una semilla.
- 4 tamaños de corpus (144 a 578 cuentos sintéticos).
- 36 modelos: variación de dropout, temperatura, épocas, longitud de secuencia.
- Métricas: perplejidad, TTR (diversidad), longitud, calidad subjetiva (1–5).
  - Poca perplejidad: Mayor precisión.
  - Mayor TTR: Mayor diversidad de palabras, no necesariamente bueno
  - Longitud: Longitud del cuento generado.
  - Calidad: Calificada manualmente según la escala de Likert.

Problema y Objetivos

Problema: ¿Cómo varía la coherencia, diversidad y completitud narrativa al modificar:

- dropout, temperatura, longitud de secuencia, épocas y tamaño del corpus?

Objetivo general:

- Analizar el impacto de hiperparámetros y corpus en la calidad de cuentos generados por LSTM.

Objetivos específicos:

- Implementar generador interactivo.
- Automatizar entrenamiento y evaluación.
- Cuantificar calidad con métricas automáticas y subjetivas.
- Identificar combinaciones óptimas.

Metodología: Datos utilizados

- Cuentos sintéticos generados con Grok y Gemini.
- Estructura simple: inicio, desarrollo y cierre con palabra clave.
- 4 tamaños de corpus:
  - `corto`: 144 cuentos
  - `mediano`: 288 cuentos
  - `largo`: 432 cuentos
  - `muy_largo`: 578 cuentos

Ejemplo:

```
Un lobo blanco aulló y la luna lo guió a casa.

Un herrero forjó un martillo que construyó un puente mágico.
```

Metodología: Preprocesamiento

1. **Conversión a minúsculas** → normalización.
2. **Tokenización** → `Tokenizer` de Keras.
3. **Secuencias n-gram** → entrada: palabras previas, salida: siguiente.

No se usó stopword removal ni lematización: se preservó estructura narrativa.

Tampoco se usó earlyStop, ya que el modelo no es lo suficientemente grande.

Metodología: Herramientas

- **Python** + librerías:
  - `tensorflow` → modelo LSTM
  - `numpy` → arreglos
  - `keras.preprocessing` → tokenización y padding
  - `os`, `json` → manejo de archivos
  - `sklearn` → train/test split
  - `pandas`, `matplotlib`, `seaborn` → análisis y gráficos

Implementación: Fase Interactiva

Archivo: `cuentos.py`

Flujo:

1. Carga corpus → minúsculas → líneas limpias
2. Preprocesamiento → secuencias + padding
3. Modelo:
   - `Embedding(100)`
   - `LSTM(150)`
   - `Dropout`
   - `Dense(softmax)`
4. Entrenamiento: `adam` + `sparse_categorical_crossentropy`
5. Generación con **temperatura** y parada en `palabrasCierre`

Implementación: Automatización

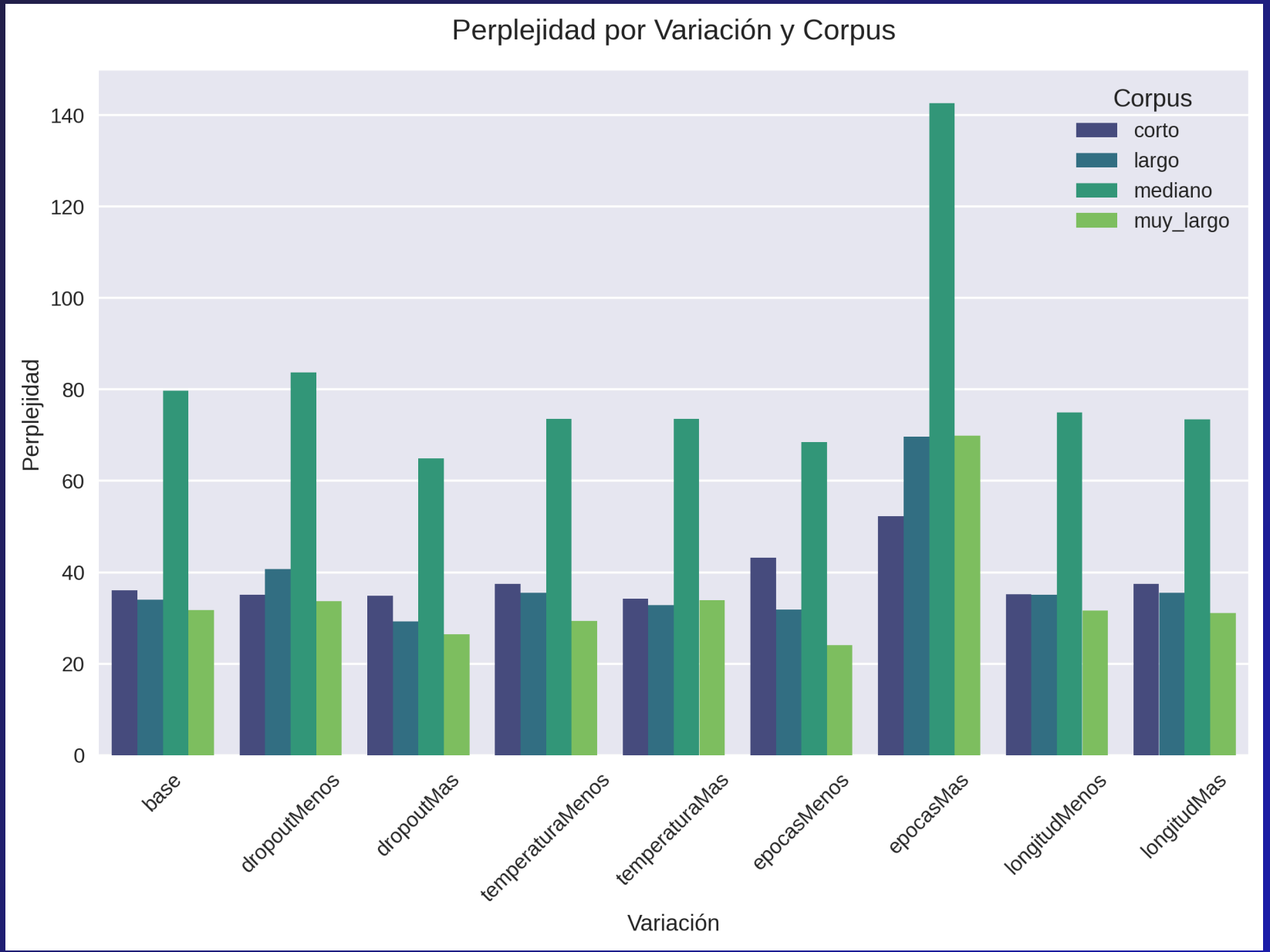
- 36 configuraciones: 4 corpus × 9 variaciones
- Scripts: `cuentosCorto.py`, `...MuyLargo.py`
- `scriptAutomatización.py` ejecuta:
  - Entrena o carga modelo (`modelo.h5`)
  - Genera 7 cuentos con semillas fijas
  - Calcula: perplejidad, TTR, longitud, % cierre
- División 80/20 → perplejidad en validación

Implementación: Análisis

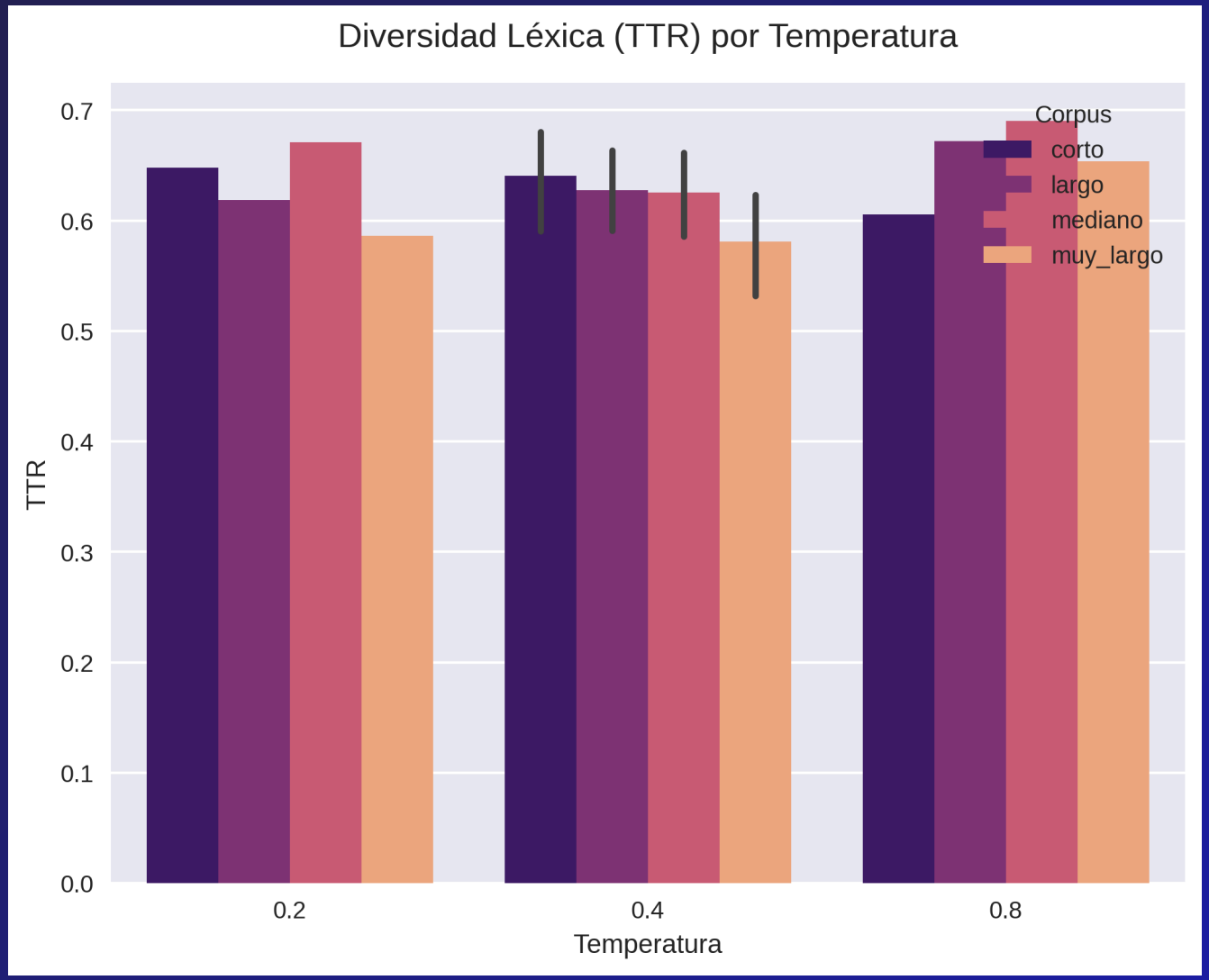
- `analisis.py`:
  - Recopila métricas → `resumen_metricas.csv`
  - Genera 7 gráficos comparativos
  - Permite ingresar calidad subjetiva (1–5)
  - `palabrasCierre` → cierre garantizado

Resultados: Hallazgos clave

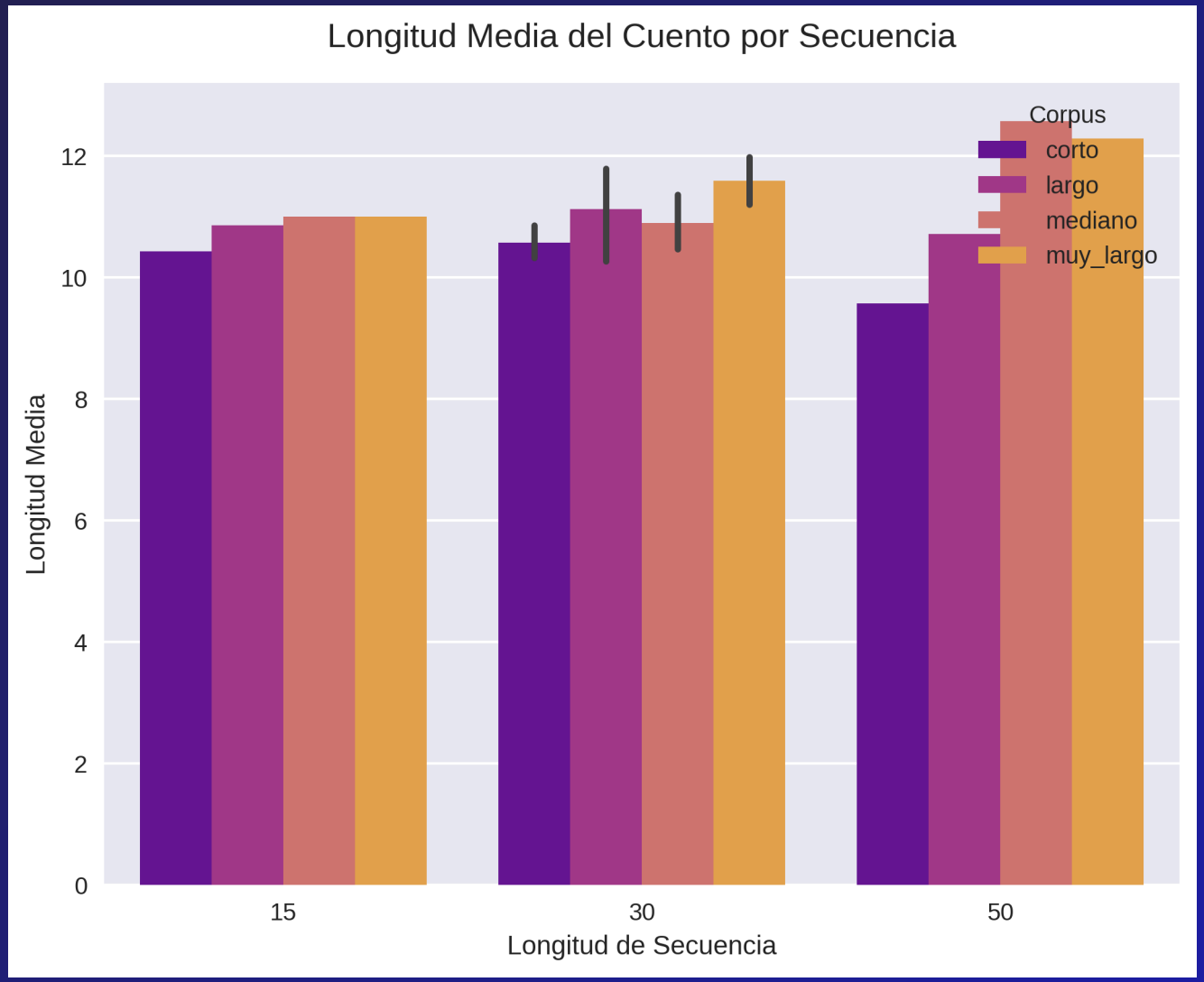
- **Corpus grande** → reduce la perplejidad, mejora la coherencia.
- **Dropout = 0.0** → máxima calidad subjetiva (5).
- **Temperatura 0.2–0.4** → equilibrio coherencia/creatividad.
- **30–60 épocas** → óptimas; 15 → incoherente.
- **Longitud secuencia = 30** → ideal en todos los tamaños.
- **100 % de cierre** en todos los modelos.



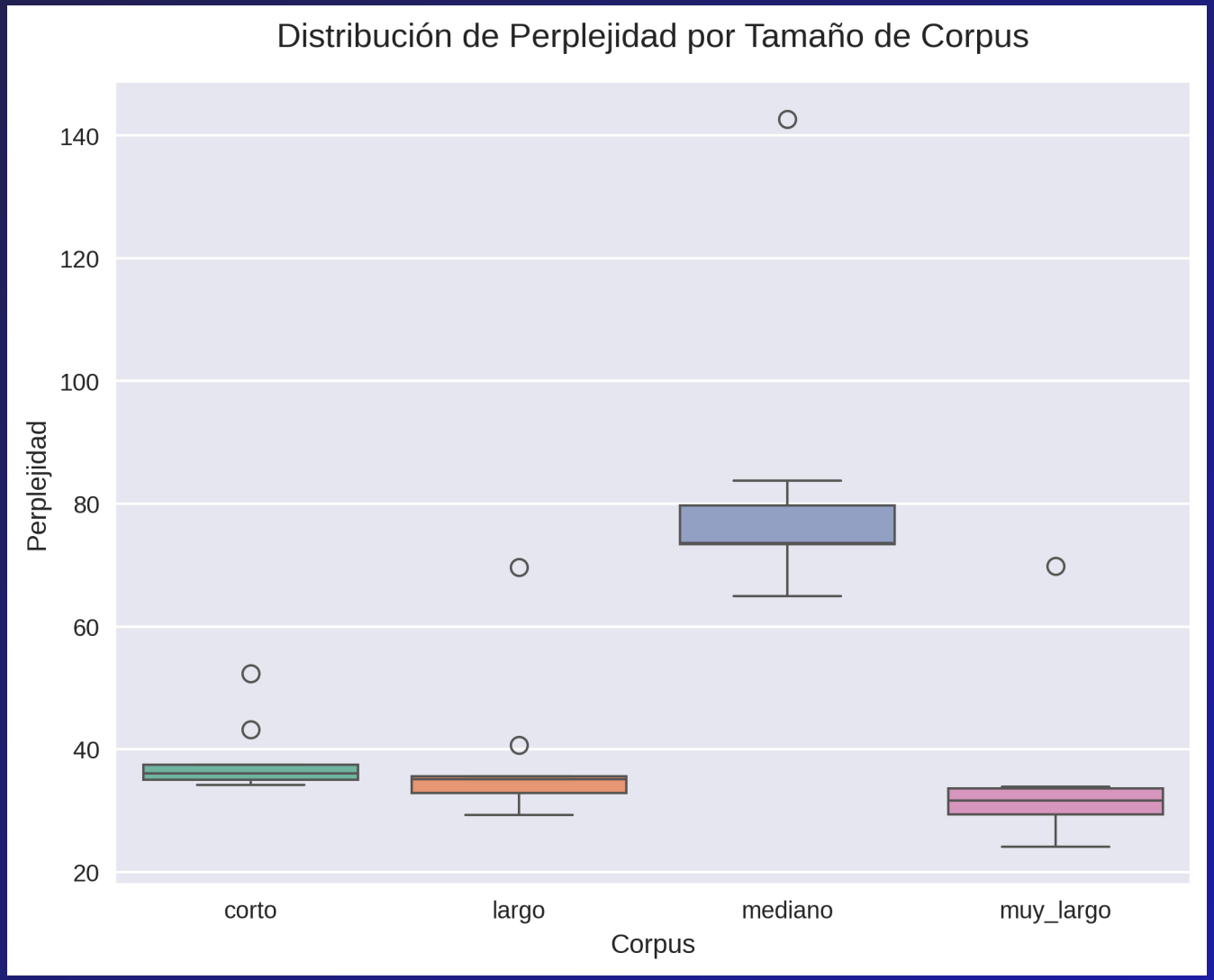
Con más cuentos, la perplejidad baja mucho.



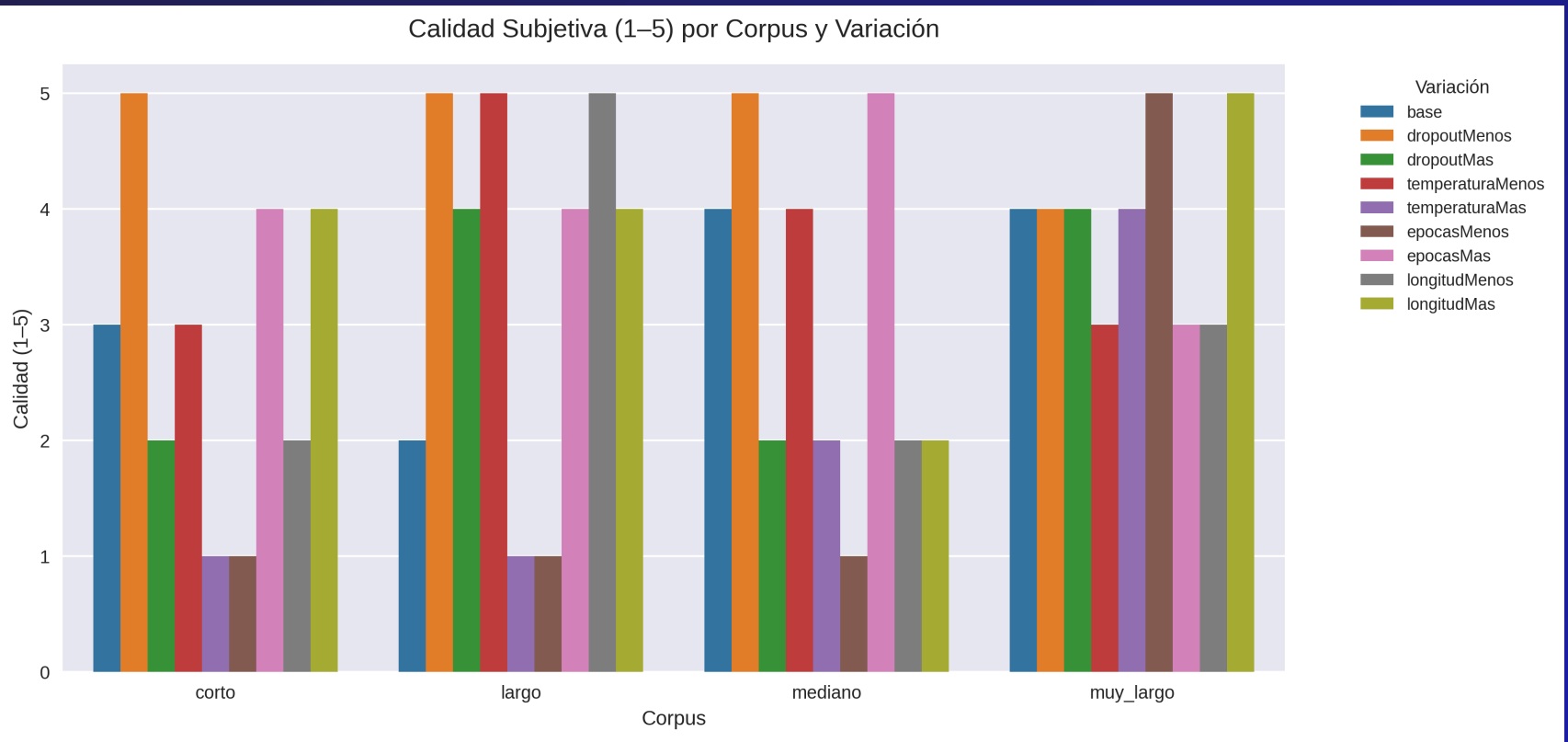
Alta temperatura
= más variedad.



Con 50 palabras de memoria, los cuentos son más largos.



Cuanto más grande el corpus, más estable y baja la perplejidad.



Calificación 5 solo con dropout 0 y configuraciones balanceadas. Corpus pequeño nunca llega a 5.

Ejemplos de cuentos generados

Configuración `corto_longitudMenos` (calificación 2/5)

```
Semilla: un lobo
un lobo de libro que narró sus los contaba de encantados
```

Repetición, incoherente.

Configuración `mediano_dropoutMenos` (calificación 5/5)

```
Semilla: un niño
un niño encontró un martillo que construyó un puente de plata
```

Estructura clara, efecto mágico coherente.

Configuración `muy_largo_temperaturaMas` (calificación 4/5)

```
Semilla: en un pueblo
en un pueblo nevado un sastre cosió abrigos que volaron
```

Alta creatividad, cierre abrupto.

Configuración `largo_epocasMenos` (calificación 1/5)

```
Semilla: un lobo
un lobo fugaz un pan que al usarlo daba la suerte
```

Sin conexión semántica.

Conclusiones

- LSTM simple genera narrativa funcional con datos estructurados.
- **Corpus grande** es el factor dominante.
- **Dropout bajo**, **temp. moderada**, **30–60 épocas**, **seq. 30** → óptimo.
- Cierre garantizado con `palabrasCierre`, pero calidad depende de coherencia interna.
- Limitaciones: corpus sintético, evaluación subjetiva, sin métricas avanzadas.

Gracias