Universidad Mayor de San Andrés
Facultad de Ciencias Puras y Naturales

# Generación de Cuentos con LSTM: Impacto de Parámetros y Corpus

Dat 261 - Procesamiento del Lenguaje Natural

**Docente:**

M. Sc. Rosa Flores Morales

**Estudiante:**

- Gabriel Muñoz Marcelo Callisaya

**Fecha de entrega:**

29 de octubre del 2025

**La Paz - Bolivia**

# Índice

## 2. Resumen

Este proyecto desarrolla un sistema basado en redes LSTM para generar cuentos cortos a partir de una semilla textual, analizando cómo varía la calidad del texto según hiperparámetros y tamaño del corpus. El objetivo fue evaluar el impacto de dropout, temperatura, longitud de secuencia, épocas y número de cuentos (144 a 578) en coherencia, diversidad y completitud narrativa.

La metodología incluyó: (1) preprocesamiento con tokenización y secuencias n--gram, (2) entrenamiento de 36 modelos LSTM mediante un script automatizado, (3) generación de 7 cuentos por configuración con semillas fijas, y (4) evaluación mediante perplejidad, TTR, longitud media, porcentaje de cierre y calidad subjetiva (1–5).

Los resultados muestran que corpus grandes reducen perplejidad y mejoran coherencia; dropout nulo maximiza calidad; temperatura moderada (0.2–0.4) equilibra creatividad y sentido; 30–60 épocas son óptimas; y longitud de secuencia 30 es ideal. Todos los modelos cierran correctamente gracias a un conjunto de palabras clave. Se logra generación narrativa funcional con LSTM simple, aunque limitada por corpus sintético y evaluación subjetiva.

## 3. Introducción

La generación automática de texto ha evolucionado significativamente con el uso de redes neuronales recurrentes, destacando las LSTM (**Long Short-Term Memory**) por su capacidad para modelar dependencias a largo plazo en secuencias. Este tipo de arquitectura es especialmente adecuada para tareas narrativas, donde el contexto previo (semilla) debe guiar la producción coherente de texto. En este trabajo, se emplea una LSTM para generar cuentos cortos a partir de una palabra o frase inicial (**prompt**), explorando cómo parámetros de entrenamiento y diseño afectan la calidad del resultado.

El problema consiste en determinar cómo varía la coherencia, diversidad léxica y completitud narrativa de los cuentos generados al modificar hiperparámetros (dropout, temperatura, longitud de secuencia, épocas) y el volumen del corpus de entrenamiento. Se plantea como caso de estudio el uso de cuentos sintéticos de 10–15 palabras, generados por modelos como **Grok** y **Gemini**, para entrenar y evaluar 36 configuraciones sistemáticas.

**Objetivo general**: Analizar el impacto de los hiperparámetros y el tamaño del corpus en la calidad de cuentos cortos generados por un modelo LSTM.

**Objetivos específicos**:
- Implementar un generador interactivo basado en LSTM.
- Automatizar el entrenamiento y evaluación de múltiples configuraciones.
- Cuantificar la calidad mediante métricas automáticas y subjetivas.
- Identificar combinaciones óptimas de parámetros para narrativa coherente.

# 4. Metodología

## Datos utilizados

Los datos consisten en una colección de cuentos cortos generados artificialmente mediante los modelos **Grok** y **Gemini**, con una longitud promedio de entre 10 y 15 palabras por cuento. Cada cuento sigue una estructura narrativa simple y coherente, con un inicio, desarrollo y cierre marcado por una palabra clave de finalización.

Se utilizaron cuatro corpus de diferentes tamaños:

- `cuentosCorto.txt`: 144 cuentos.
- `cuentosMediano.txt`: 288 cuentos.
- `cuentosLargo.txt`: 432 cuentos.
- `cuentosMuyLargo.txt`: 578 cuentos.

### Ejemplo de cuentos en el corpus:

```
Un músico tocó un violín que hacía levitar los objetos más pesados.

En una herrería antigua, una fragua cantó historias de héroes olvidados.

Un niño solitario adoptó un perro que ladraba y protegía su aldea del peligro.

Una sirena varada dio una perla que brilló y salvó a la tripulación.
```

## Preprocesamiento

El preprocesamiento se realiza en la función `preprocesarTexto`, que incluye los siguientes pasos:

1. **Conversión a minúsculas**: Todo el texto se convierte a minúsculas para normalizar el vocabulario.
2. **Tokenización**: Se emplea `Tokenizer` de `tensorflow.keras.preprocessing.text` para convertir palabras en índices numéricos.
3. **Generación de secuencias n-gram**: Para cada cuento, se crean secuencias de longitud creciente (desde 2 hasta la longitud completa del cuento), usando como entrada todas las palabras excepto la última, y como salida la palabra siguiente.
4. **Relleno (**padding**) previo**: Las secuencias se rellenan con ceros al inicio hasta alcanzar la `longitudSecuencia` especificada, utilizando `pad_sequences` con `padding='pre'`.
5. **Separación de datos**: Los datos de entrada (`datosEntrada`) corresponden a las secuencias sin la última palabra, y los de salida (`datosSalida`) a la palabra objetivo.

No se aplicaron técnicas adicionales como eliminación de **stopwords** ni lematización, dado que el objetivo es preservar la estructura narrativa completa y el vocabulario original de los cuentos.

### Herramientas y librerías utilizadas

El desarrollo se realizó en **Python** utilizando las siguientes librerías:

- `tensorflow`: Framework principal para la construcción, entrenamiento y predicción del modelo LSTM.
- `numpy`: Manipulación eficiente de arreglos numéricos.
- `tensorflow.keras.preprocessing.text.Tokenizer`: Tokenización del texto.
- `tensorflow.keras.preprocessing.sequence.pad_sequences`: Relleno de secuencias.
- `tensorflow.keras.layers`: Capas del modelo (`Embedding`, `LSTM`, `Dense`, `Dropout`).
- `os`: Manejo de rutas y archivos.
- `json`: Lectura y escritura de métricas en formato JSON.
- `sklearn.model_selection.train_test_split`: División de datos en entrenamiento y validación (en el script de automatización).
- `pandas`, `matplotlib`, `seaborn`: Análisis y visualización de resultados en `analisis.py`.

## 5. Implementación

El desarrollo se inició con la creación de un programa interactivo en `cuentos.py` para entrenar un modelo LSTM y generar cuentos cortos a partir de una semilla proporcionada por el usuario. Posteriormente, se extendió el alcance del proyecto para analizar sistemáticamente el impacto de los hiperparámetros y el tamaño del corpus en la calidad de los cuentos generados, dando lugar a una arquitectura modular con automatización y evaluación comparativa.

### Fase inicial: Generador interactivo

El programa original (`cuentos.py`) permite al usuario especificar cuatro hiperparámetros clave: `longitudSecuencia`, `epocas`, `temperatura` y `dropout`. El flujo de ejecución es el siguiente:

1. **Carga del corpus**: La función `cargarCorpus` lee el archivo `cuentos.txt`, convierte el texto a minúsculas y filtra líneas vacías.

```
def cargarCorpus(ubicacionArchivo):
    with open(ubicacionArchivo, 'r', encoding='utf-8') as archivo:
        cuentos = archivo.read().lower().split('\n')
    return [cuento.strip() for cuento in cuentos if cuento.strip()]
```

2. **Preprocesamiento**: `preprocesarTexto` tokeniza el corpus con `Tokenizer`, genera secuencias n-gram de longitud creciente y aplica relleno previo (`padding='pre'`) hasta `longitudSecuencia`.

```
secuenciasRellenas = pad_sequences(secuenciasTokenizadas, maxlen=longitudSecuencia,
padding='pre')
datosEntrada = secuenciasRellenas[:, :-1]
datosSalida = secuenciasRellenas[:, -1]
```

3. **Construcción del modelo**: Se define una arquitectura secuencial con una capa de `Embedding` (dimensión 100), una capa `LSTM` de 150 unidades, `Dropout` y una capa `Dense` con activación `softmax`.

```
modelo = tf.keras.Sequential([
    Embedding(palabrasTotales, 100, input_length=longitudSecuencia-1),
    LSTM(150, return_sequences=False),
    Dropout(dropout),
    Dense(palabrasTotales, activation='softmax')
])
```

4. **Entrenamiento y generación**: El modelo se entrena con pérdida `sparse_categorical_crossentropy` y optimizador `adam`. La generación utiliza muestreo con temperatura para controlar la aleatoriedad.

```
prediccion = np.log(prediccion + 1e-10) / temperatura
prediccion = np.exp(prediccion) / np.sum(np.exp(prediccion))
indicePredicho = np.random.choice(len(prediccion[0]), p=prediccion[0])
```

La generación finaliza cuando se predice una palabra del conjunto `palabrasCierre` y el cuento tiene al menos 6 palabras.

## Observación de sensibilidad a parámetros

Durante las pruebas iniciales con `cuentos.py`, se observó que pequeñas variaciones en los hiperparámetros producían cambios significativos en coherencia, diversidad léxica y finalización adecuada del cuento. Esto motivó la reformulación del proyecto hacia un análisis comparativo sistemático.

## Fase de escalamiento: Automatización y evaluación

Para estudiar el efecto de cada parámetro de forma aislada, se diseñaron 36 configuraciones: 4 tamaños de corpus × 9 variaciones (base + 8 modificaciones: ±dropout, ±temperatura, ±épocas, ±longitudSecuencia). Se crearon cuatro versiones del programa base (`cuentosCorto.py`, `cuentosMediano.py`, `cuentosLargo.py`, `cuentosMuyLargo.py`), cada una asociada a su corpus correspondiente.

El script `scriptAutomatización.py` ejecuta automáticamente:

- Entrenamiento o carga de modelos previamente guardados (`modelo.h5`).
- Generación de 7 cuentos por configuración usando semillas fijas.
- Cálculo de métricas automáticas: perplejidad, TTR, longitud media, porcentaje de cierre.
- Almacenamiento estructurado en directorios `resultados/<corpus>_<variacion>/`.

```
if os.path.exists(rutaModelo):
    modelo = load_model(rutaModelo)
else:
    # Entrenar y guardar
    modelo.save(rutaModelo)
```

Se incorporó división entrenamiento/validación (80/20) y evaluación de perplejidad sobre el conjunto de validación para una métrica objetiva de rendimiento.

## Análisis de resultados

El script `analisis.py` recopila las métricas de todos los modelos, genera un resumen en CSV y produce 6 gráficos comparativos utilizando `matplotlib` y `seaborn`.

Además, permite la incorporación manual de una métrica subjetiva de calidad (escala Likert 1–5).

### Justificación de decisiones técnicas

- **Relleno previo (`padding='pre'`)**: Preserva el orden temporal de las palabras y mejora la eficiencia del procesamiento en lote.
- **Muestreo con temperatura**: Permite controlar el balance entre coherencia (baja temperatura) y creatividad (alta temperatura).
- **Conjunto `palabrasCierre`**: Garantiza que los cuentos generados tengan un cierre narrativo explícito, facilitando la evaluación de completitud.
- **Métricas automáticas y subjetivas**: La combinación de perplejidad (rendimiento predictivo), TTR (diversidad) y evaluación humana proporciona una visión integral de la calidad.

## 6. Resultados

El análisis se basa en 36 configuraciones (4 tamaños de corpus × 9 variaciones de parámetros), evaluadas mediante 7 cuentos generados por modelo con semillas fijas. Las métricas automáticas incluyen perplejidad (sobre conjunto de validación), Type-Token Ratio (TTR), longitud media del cuento y porcentaje de cuentos que finalizan con una palabra del conjunto `palabrasCierre`. La calidad subjetiva se evaluó en escala Likert de 1 a 5.

### Análisis cuantitativo

Los resultados se resumen en `resumen_metricas.csv`.

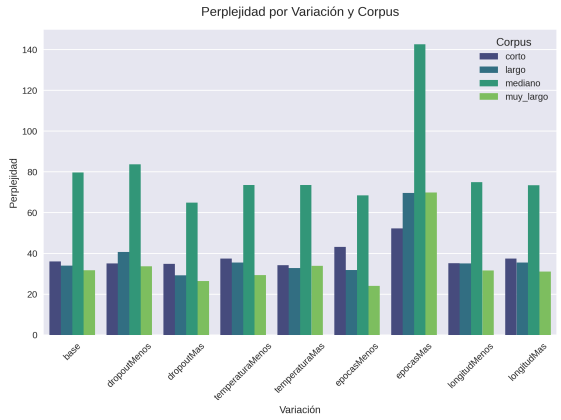
. A continuación se destacan los principales hallazgos:

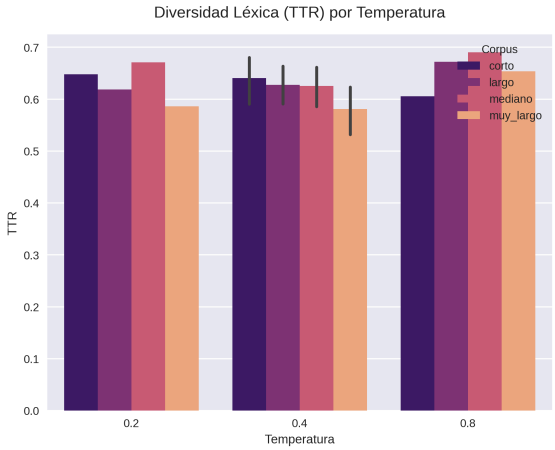
- **Tamaño del corpus**: La perplejidad disminuye al aumentar el número de cuentos (de 36.08 en `corto` a 31.76 en `muy_largo` en configuración base), indicando mejor capacidad predictiva con más datos.
- **Dropout**: Valores bajos (0.0) mejoran la calidad subjetiva (calificación 5 en varios casos), mientras que valores altos (0.5) tienden a reducirla (calificación 2).
- **Temperatura**: Valores altos (0.8) generan mayor diversidad (TTR > 0.65) pero menor coherencia (calificación ≤ 2). Valores bajos (0.2) favorecen coherencia pero reducen creatividad.
- **Épocas**: Entrenamientos cortos (15 épocas) producen resultados incoherentes (calificación 1), mientras que 60 épocas mejoran calidad en corpus medianos y largos.
- **Longitud de secuencia**: Valores extremos (15 o 50) afectan negativamente la coherencia en corpus pequeños, pero mejoran la longitud media en corpus grandes.

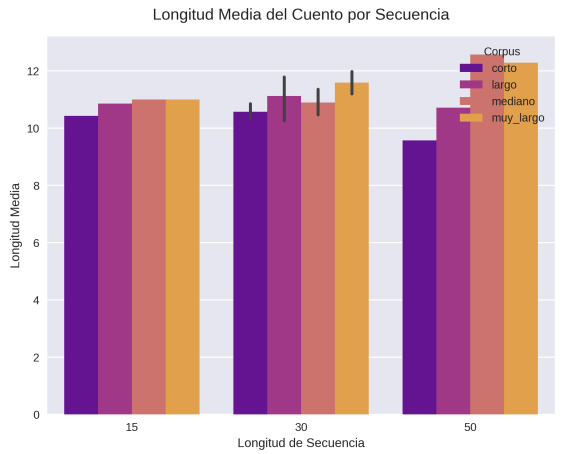
La Figura 1 (`graficos_completos_6.png`) presenta los 7 gráficos generados por `analisis.py`:

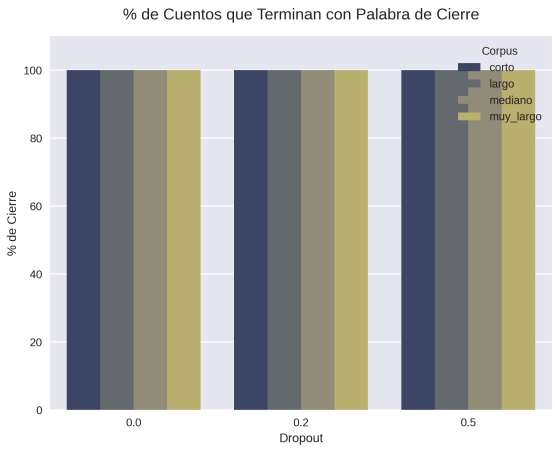
1. Perplejidad por variación y corpus.
2. Diversidad léxica (TTR) por temperatura.
3. Longitud media por longitud de secuencia.

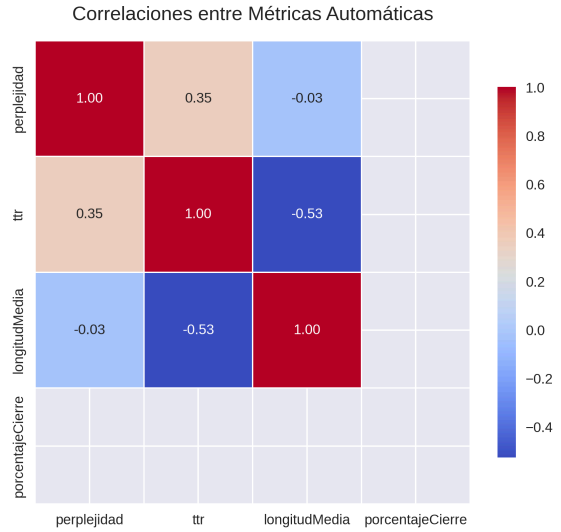
4. Porcentaje de cierre por dropout (100 % en todos los casos).
5. Correlaciones entre métricas automáticas.
6. Distribución de perplejidad por tamaño de corpus.
7. La calidad subjetiva medida por la escala de Likert

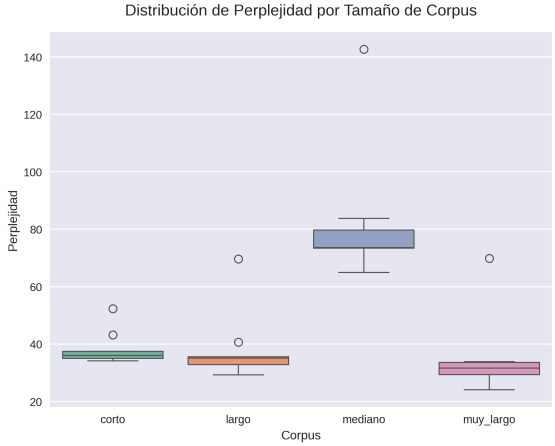


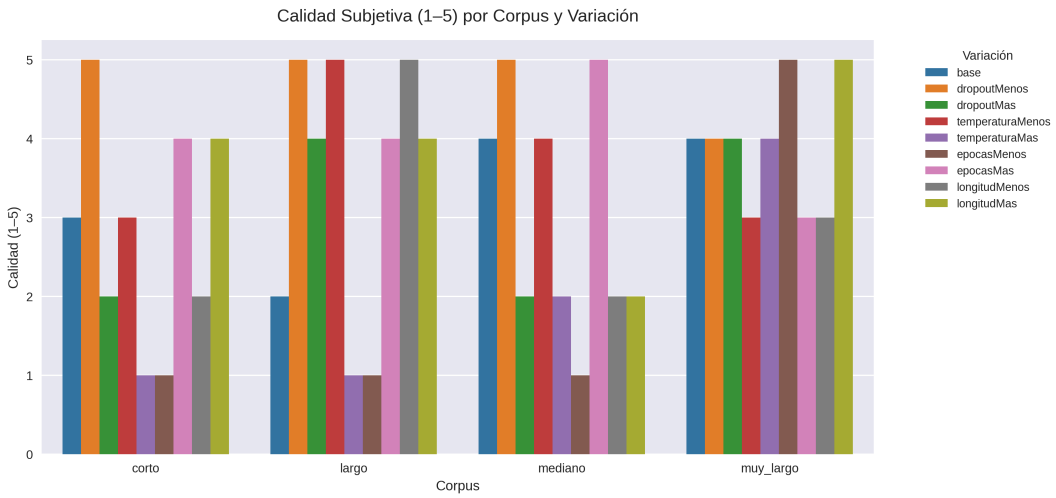






## Ejemplos de predicciones del modelo

A continuación se muestran ejemplos representativos de cuentos generados, junto con su configuración y calificación subjetiva:

- **Configuración `corto_longitudMenos` (longitudSecuencia=15, calificación 2)**:

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

  *Observación*: Repetición y falta de coherencia narrativa.

- **Configuración `mediano_dropoutMenos` (dropout=0.0, calificación 5)**:

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

  *Observación*: Estructura clara, objeto mágico con efecto coherente.

- **Configuración `muy_largo_temperaturaMas` (temperatura=0.8, calificación 4)**:

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

  *Observación*: Creatividad alta, pero cierre abrupto.

- **Configuración `largo_epocasMenos` (épocas=15, calificación 1)**:

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

  *Observación*: Falta de conexión semántica, entrenamiento insuficiente.

## Análisis de aciertos y errores

**Aspectos positivos**:

- Todos los modelos logran 100 % de cierre con palabras del conjunto `palabrasCierre`, gracias al mecanismo de parada explícito.
- Corpus grandes (`largo`, `muy_largo`) generan cuentos más coherentes y variados incluso con parámetros subóptimos.

- Dropout bajo mejora significativamente la calidad subjetiva al reducir sobreajuste en corpus pequeños.

**Limitaciones y errores**:
- Temperaturas altas introducen ruido semántico y repeticiones.
- Longitudes de secuencia extremas en corpus pequeños provocan fragmentación narrativa.
- Entrenamientos cortos resultan en predicciones aleatorias y sin estructura.
- En corpus pequeños, el modelo tiende a repetir patrones del corpus de entrenamiento sin generalización.

En general, la calidad óptima se obtiene con corpus grandes, dropout bajo, temperatura moderada (0.2–0.4), 30–60 épocas y longitud de secuencia de 30.

# 7. Discusión

## Comparación con expectativas iniciales

El objetivo inicial era generar cuentos cortos coherentes mediante un modelo LSTM, con parámetros ajustables por el usuario. Esta funcionalidad se cumplió satisfactoriamente en la versión interactiva (`cuentos.py`). Sin embargo, la variabilidad observada en la calidad de los cuentos generados al modificar hiperparámetros y tamaño del corpus excedió las expectativas iniciales, motivando un análisis sistemático más profundo.

Se esperaba que:
- Un mayor número de épocas mejorara la coherencia: **Confirmado**, especialmente en corpus medianos y grandes (calificación 4–5 con 60 épocas vs. 1 con 15).
- La temperatura controlara la creatividad: **Parcialmente confirmado**. Valores altos aumentan diversidad (TTR), pero a costa de coherencia (calificación ≤ 2).
- El dropout previniera sobreajuste: **Confirmado de forma inversa**. Valores nulos (0.0) mejoran calidad en todos los tamaños de corpus, sugiriendo que el sobreajuste no es el principal problema en este dominio.
- Un corpus más grande mejorara el rendimiento: **Totalmente confirmado**. La perplejidad disminuye y la calidad subjetiva aumenta con el tamaño del corpus.

## Limitaciones encontradas

A pesar de los resultados positivos, se identificaron varias limitaciones:

1. **Dependencia del conjunto `palabrasCierre`**: Aunque garantiza cierre narrativo, puede forzar terminaciones abruptas o poco naturales cuando el modelo no ha desarrollado una narrativa coherente.
2. **Corpus sintético**: Los cuentos fueron generados por **Grok** y **Gemini**, lo que introduce un sesgo estilístico y limita la diversidad real de narrativas humanas. Esto puede haber facilitado el aprendizaje de patrones repetitivos.
3. **Evaluación subjetiva limitada**: La calidad se midió por una sola persona en escala Likert, sin inter-evaluadores ni criterios explícitos de coherencia, creatividad o gramática.

4. **Ausencia de métricas avanzadas**: No se implementaron BLEU, ROUGE ni evaluación de coherencia semántica (e.g., BERTScore), lo que limita la comparación con trabajos previos.
5. **Escalabilidad computacional**: El entrenamiento secuencial de 36 modelos tomó 20 minutos en CPU, lo que restringe la exploración de configuraciones más amplias (e.g., optimización bayesiana).

En resumen, aunque el sistema cumple su propósito y revela tendencias claras en el impacto de los parámetros, los resultados deben interpretarse en el contexto de un corpus controlado y una evaluación subjetiva limitada.

# 8. Conclusiones

El trabajo desarrollado ha permitido construir un sistema funcional para la generación de cuentos cortos mediante redes LSTM, evolucionando desde una herramienta interactiva hasta un marco experimental completo que evalúa sistemáticamente el impacto de hiperparámetros y tamaño del corpus en la calidad del texto generado.

Los resultados confirman que:

- El tamaño del corpus es el factor dominante en la mejora de la perplejidad y coherencia narrativa. Corpus con 432 o 578 cuentos (`largo` y `muy_largo`) logran resultados estables incluso con configuraciones subóptimas.
- El dropout bajo (0.0) mejora significativamente la calidad subjetiva, sugiriendo que la regularización no es necesaria en este dominio con datos estructurados y limitados.
- La temperatura debe mantenerse en rangos moderados (0.2–0.4) para equilibrar coherencia y creatividad. Valores extremos deterioran la narrativa.
- El número de épocas óptimo se sitúa entre 30 y 60; entrenamientos insuficientes (15 épocas) producen texto aleatorio, mientras que excesivos (60 en corpus pequeños) pueden inducir sobreajuste sin beneficio claro.
- La longitud de secuencia de 30 ofrece un compromiso adecuado entre contexto y eficiencia; valores extremos afectan negativamente en corpus pequeños.

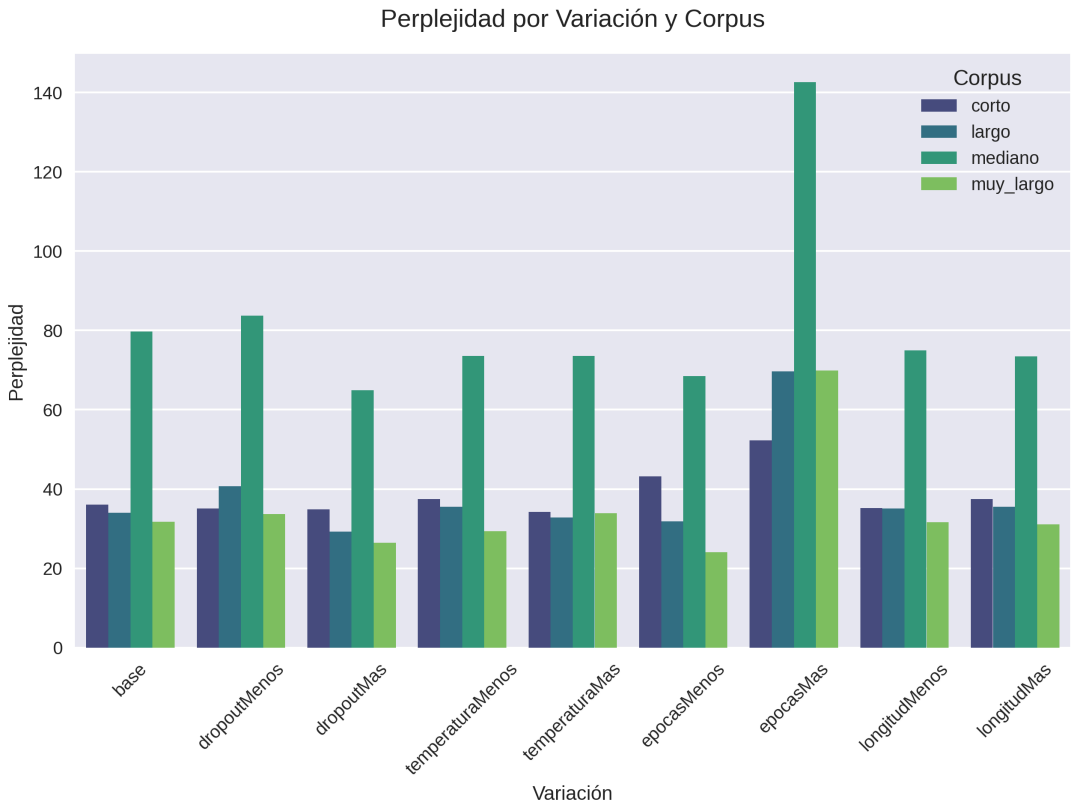
El mecanismo de finalización con `palabrasCierre` garantiza estructura narrativa, pero revela una limitación: la calidad del cuento depende más de la coherencia interna que del cierre forzado.

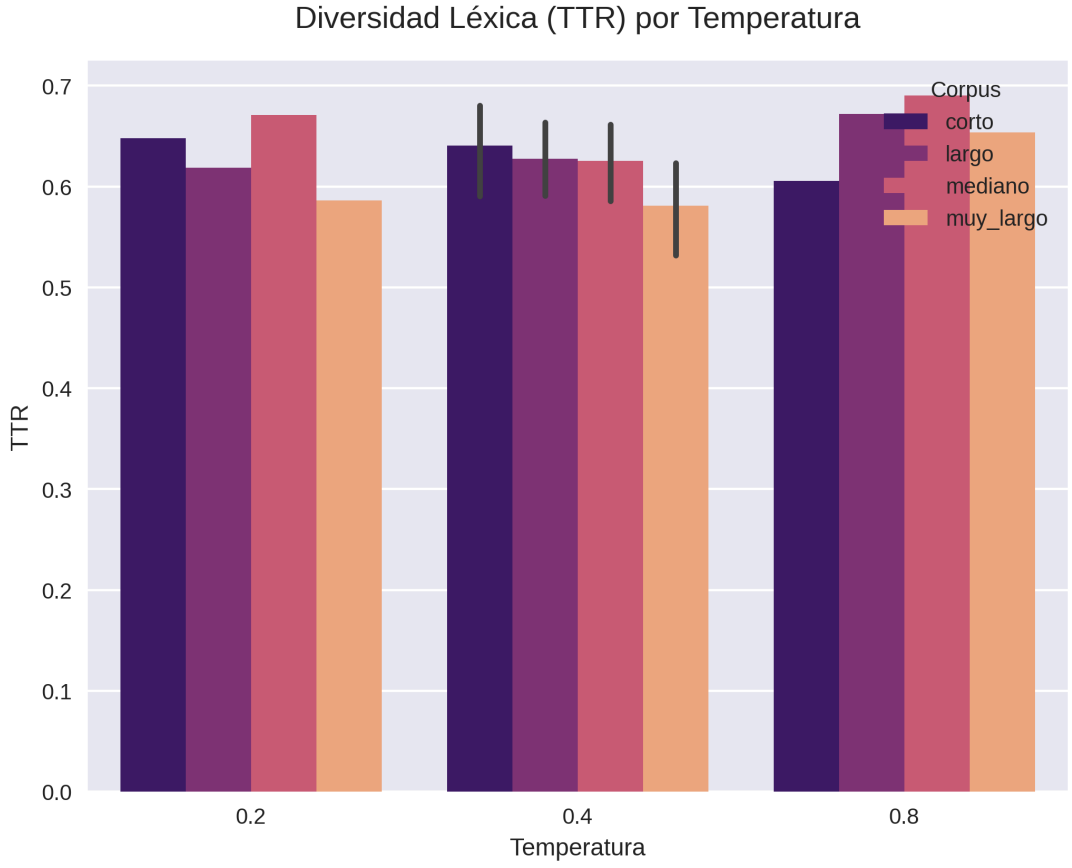
En reflexión final, este proyecto demuestra que modelos LSTM simples pueden generar texto narrativo aceptable con datos estructurados y bien diseñados, incluso sin arquitecturas avanzadas como Transformers. Sin embargo, la calidad sigue limitada por la diversidad y autenticidad del corpus, así como por la evaluación subjetiva. Futuras mejoras podrían incluir corpus de origen humano, métricas automáticas de coherencia semántica, búsqueda automática de hiperparámetros y generación condicional más flexible.

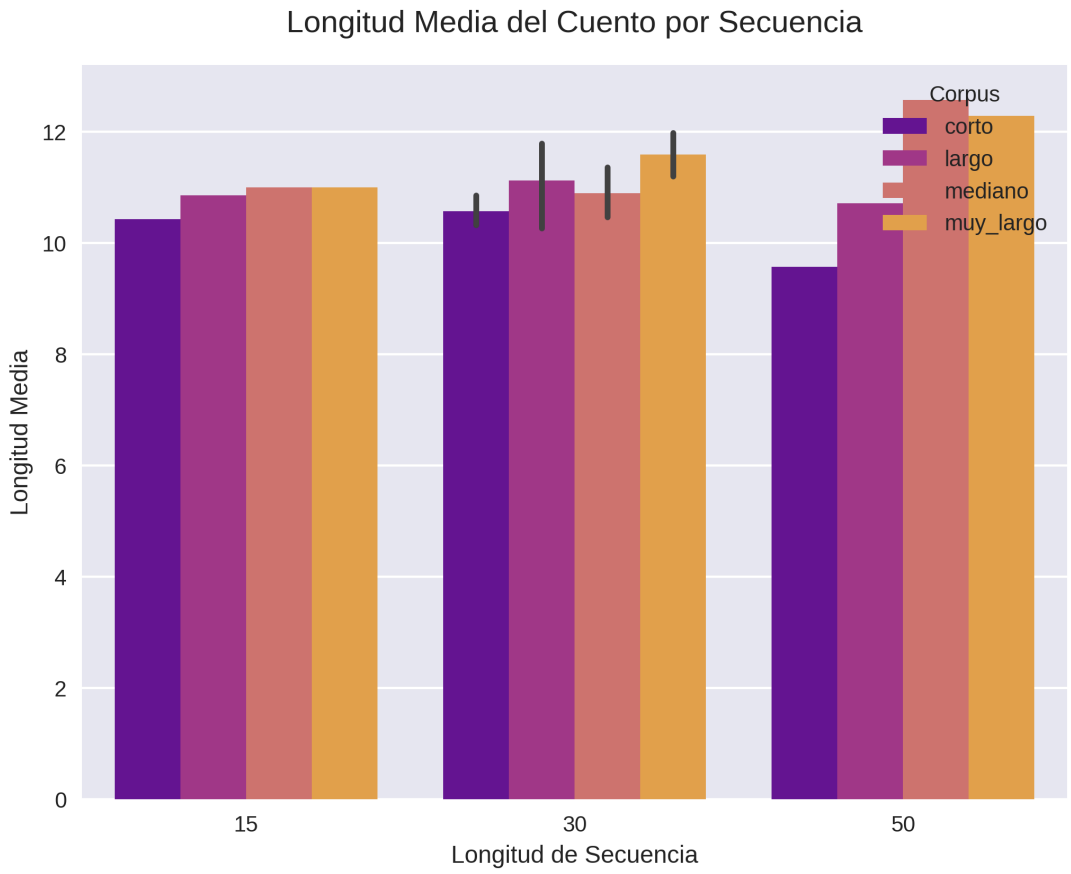
El análisis sistemático realizado establece una base sólida para comprender cómo cada parámetro influye en la generación de texto creativo, sentando las bases para aplicaciones más robustas en narrativa automática.

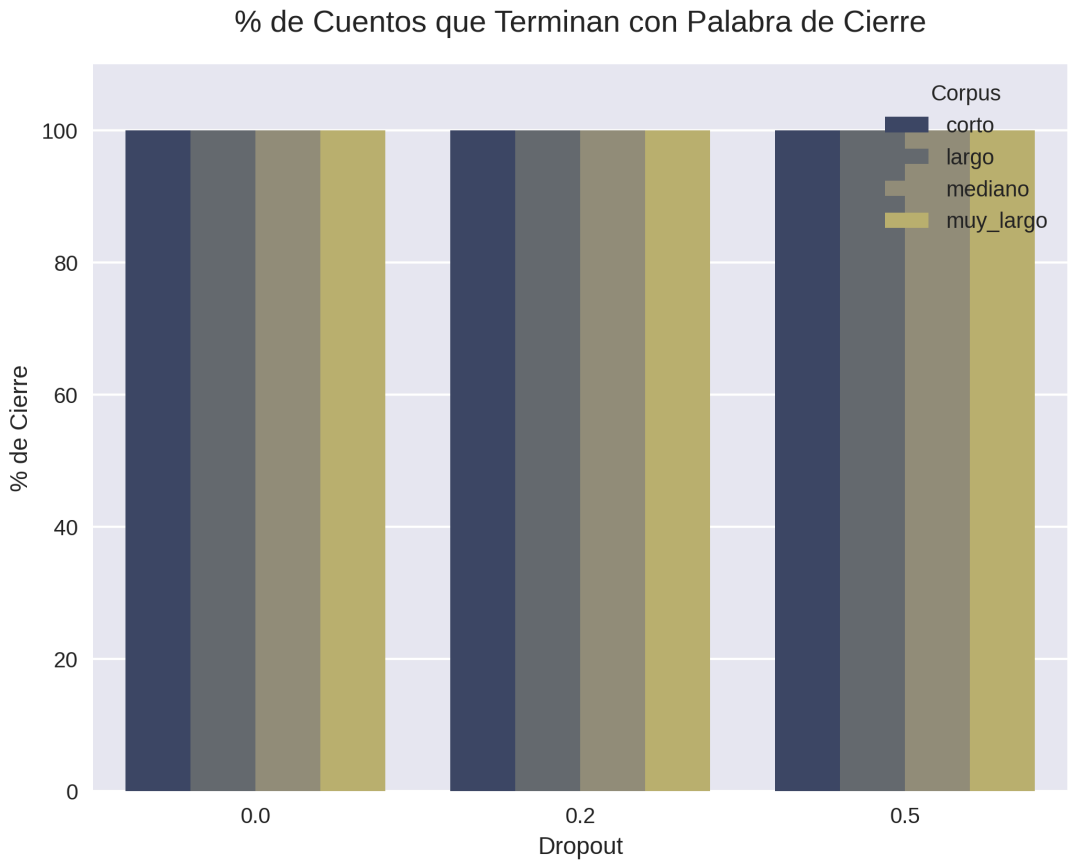
# 9. Anexos.

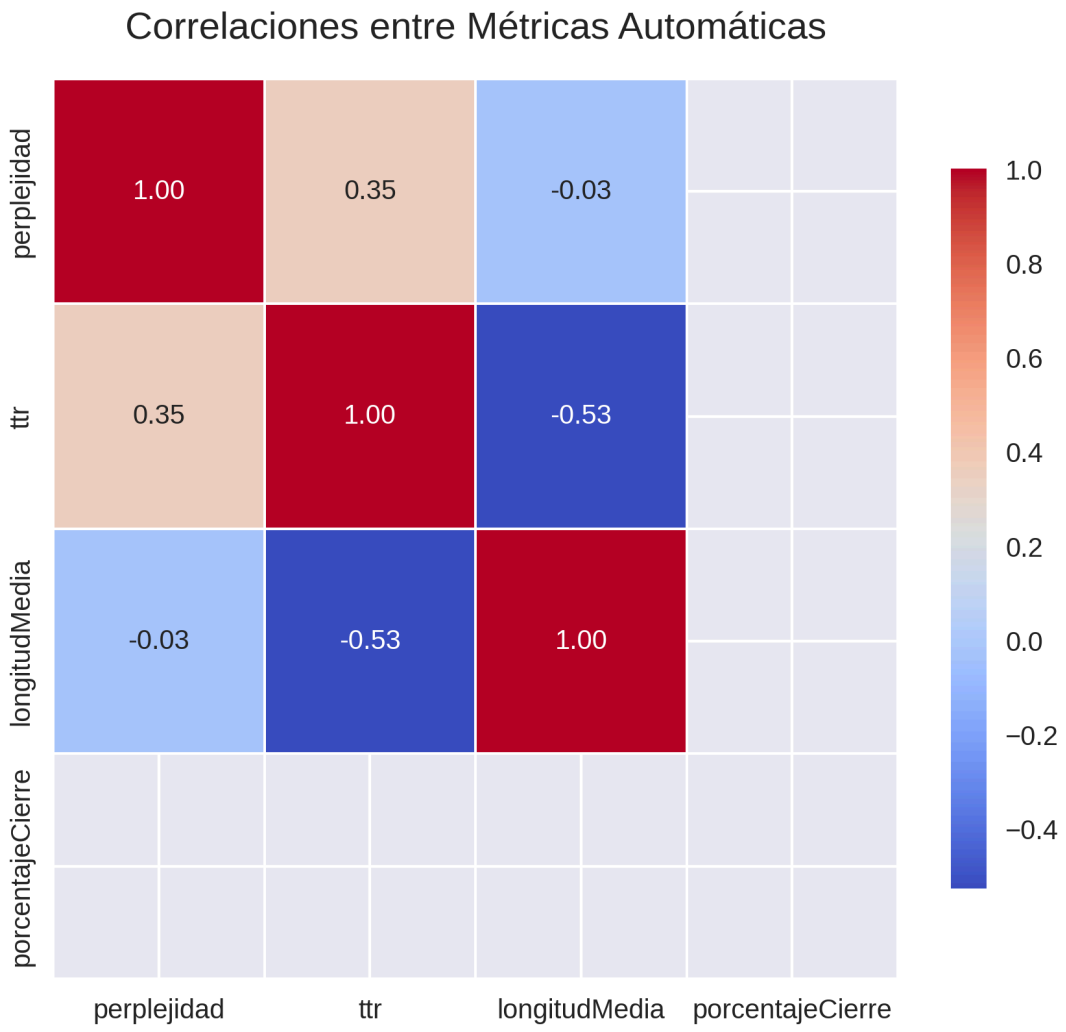
Gráficas obtenidas:

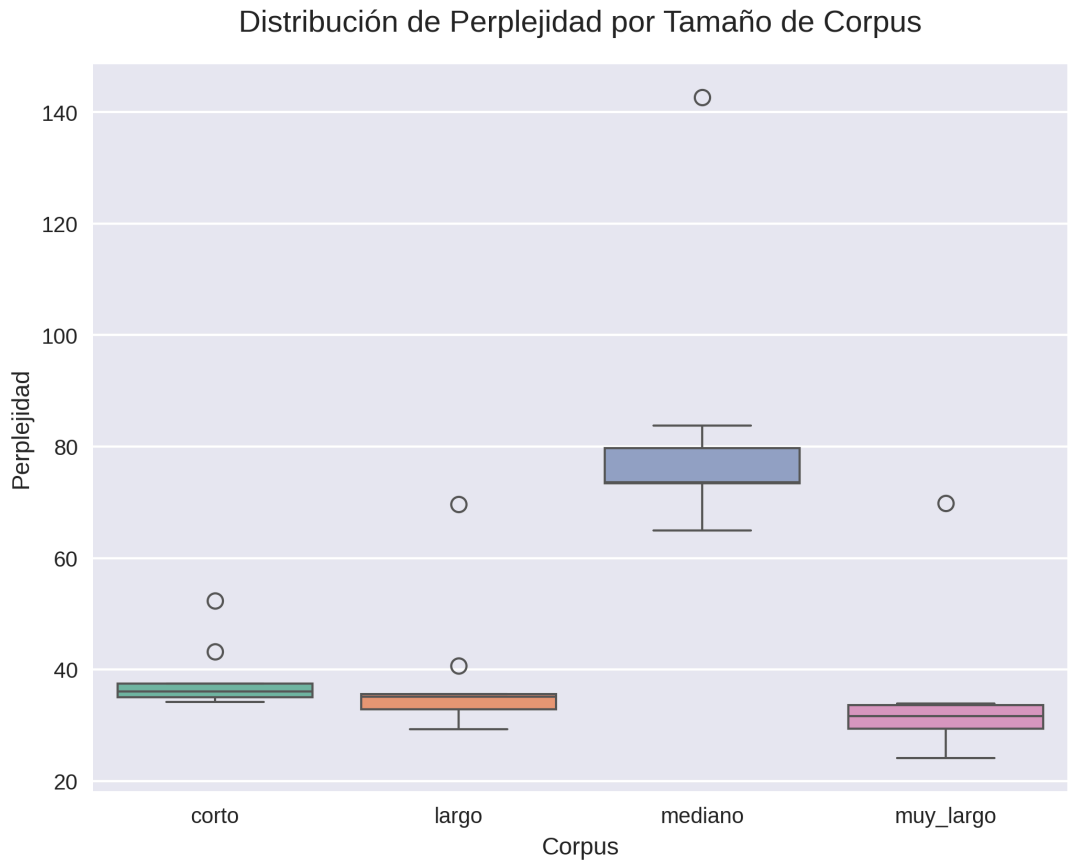


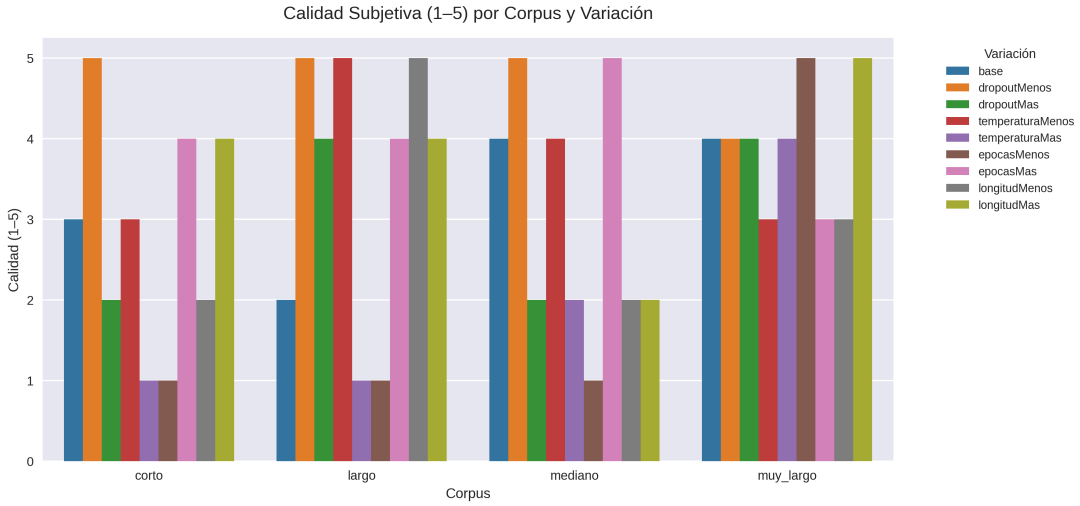




Longitud Media del Cuento por Secuencia
Corpus
corto
largo
mediano
muy_largo
Longitud Media
0
2
4
6
8
10
12
15
30
50
Longitud de Secuencia


% de Cuentos que Terminan con Palabra de Cierre
Corpus
corto
largo
mediano
muy_largo
% de Cierre
0
20
40
60
80
100
0.0
0.2
0.5
Dropout


Correlaciones entre Métricas Automáticas
perplejidad
ttr
longitudMedia
porcentajeCierre
1.00
0.35
-0.03
0.35
1.00
-0.53
-0.03
-0.53
1.00
perplejidad
ttr
longitudMedia
porcentajeCierre
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4




## Métricas:

```
corpus,variacion,longitudSecuencia,epocas,temperatura,dropout,perplejidad,ttr,longitudMedia,porcent
corto,base,30,30,0.4,0.2,36.084549510784726,0.6944444444444444,10.285714285714286,100.0,3
corto,dropoutMas,30,30,0.4,0.5,34.91592368710154,0.6486486486486487,10.571428571428571,100.0,2
corto,dropoutMenos,30,30,0.4,0.0,35.07109680535561,0.6923076923076923,11.142857142857142,100.0,5
corto,temperaturaMas,30,30,0.8,0.2,34.191348584887365,0.6056338028169014,10.142857142857142,100.0,1
corto,temperaturaMenos,30,30,0.2,0.2,37.516153303989725,0.647887323943662,10.142857142857142,100.0,
corto,epocasMas,30,60,0.4,0.2,52.32226607547586,0.6666666666666666,10.714285714285714,100.0,4
corto,epocasMenos,30,15,0.4,0.2,43.191494817798066,0.5064935064935064,11.0,100.0,1
corto,longitudMas,50,30,0.4,0.2,37.48448508854814,0.6716417910447762,9.571428571428571,100.0,4
corto,longitudMenos,15,30,0.4,0.2,35.227547461181345,0.6027397260273972,10.428571428571429,100.0,2
```

```
mediano,base,30,30,0.4,0.2,79.74828650350932,0.6190476190476191,12.0,100.0,4
mediano,dropoutMas,30,30,0.4,0.5,64.9354517405237,0.6455696202531646,11.285714285714286,100.0,2
mediano,dropoutMenos,30,30,0.4,0.0,83.71975537119876,0.6944444444444444,10.285714285714286,100.0,5
mediano,temperaturaMas,30,30,0.8,0.2,73.5842083911022,0.6901408450704225,10.142857142857142,100.0,2
mediano,temperaturaMenos,30,30,0.2,0.2,73.54036185366445,0.6710526315789473,10.857142857142858,100.
mediano,epocasMas,30,60,0.4,0.2,142.6380723164534,0.6835443037974683,11.285714285714286,100.0,5
mediano,epocasMenos,30,15,0.4,0.2,68.49735946254913,0.6027397260273972,10.428571428571429,100.0,1
mediano,longitudMas,50,30,0.4,0.2,73.41184933448923,0.5340909090909091,12.571428571428571,100.0,2
mediano,longitudMenos,15,30,0.4,0.2,74.91436728450117,0.5974025974025974,11.0,100.0,2
largo,base,30,30,0.4,0.2,34.02579526685314,0.6265060240963856,11.857142857142858,100.0,2
largo,dropoutMas,30,30,0.4,0.5,29.26171684456109,0.5632183908045977,12.428571428571429,100.0,4
largo,dropoutMenos,30,30,0.4,0.0,40.708372061059166,0.5975609756097561,11.714285714285714,100.0,5
largo,temperaturaMas,30,30,0.8,0.2,32.83578656072344,0.6721311475409836,8.714285714285714,100.0,1
largo,temperaturaMenos,30,30,0.2,0.2,35.5106976702876,0.618421052631579,10.857142857142858,100.0,5
largo,epocasMas,30,60,0.4,0.2,69.65254965157423,0.7272727272727273,11.0,100.0,4
largo,epocasMenos,30,15,0.4,0.2,31.81498273641937,0.6075949367088608,11.285714285714286,100.0,1
largo,longitudMas,50,30,0.4,0.2,35.55465705886029,0.5866666666666667,10.714285714285714,100.0,4
largo,longitudMenos,15,30,0.4,0.2,35.10779833973816,0.6842105263157895,10.857142857142858,100.0,5
muy_largo,base,30,30,0.4,0.2,31.76450483850353,0.5180722891566265,11.857142857142858,100.0,4
muy_largo,dropoutMas,30,30,0.4,0.5,26.455499516551438,0.5822784810126582,11.285714285714286,100.0,4
muy_largo,dropoutMenos,30,30,0.4,0.0,33.658139095173446,0.5647058823529412,12.142857142857142,100.0
muy_largo,temperaturaMas,30,30,0.8,0.2,33.94147436258113,0.6538461538461539,11.142857142857142,100.
muy_largo,temperaturaMenos,30,30,0.2,0.2,29.411852052648396,0.5862068965517241,12.428571428571429,1
muy_largo,epocasMas,30,60,0.4,0.2,69.8674359340675,0.68,10.714285714285714,100.0,3
muy_largo,epocasMenos,30,15,0.4,0.2,24.099667594398472,0.48148148148148145,11.571428571428571,100.0
muy_largo,longitudMas,50,30,0.4,0.2,31.12886811753964,0.6046511627906976,12.285714285714286,100.0,5
muy_largo,longitudMenos,15,30,0.4,0.2,31.653999336596538,0.6363636363636364,11.0,100.0,3
```