UCA
Pontificia Universidad Católica Argentina

Facultad de Ingeniería y Ciencias Agrarias

Análisis de Tópicos Latentes en Noticias

Utilizando Apache Spark y LDA

Trabajo Práctico Final

Materia: Programación Funcional

Alumno:
Juan Cruz Di Meglio

Carrera:
Licenciatura en Ciencia de Datos

Fecha:
17 de Diciembre 2025

Índice

1. Introducción **2**

2. Metodología **2**
2.1. Recopilación y Exploración de Datos 2
2.2. Preprocesamiento del Texto (Spark NLP) 2
2.3. Vectorización: Selección de Características 3
2.4. Modelado y Evaluación 3

3. Resultados **3**
3.1. Métricas del Modelo y Selección 3
3.2. Tópicos Identificados 3

4. Descubrimiento **4**

5. Conclusiones y Trabajo Futuro **4**

A. Anexo: Fragmentos de Código **5**
A.1. Ingesta Robusta desde GitHub 5
A.2. Pipeline de NLP (Limpieza Refinada) 5

1. Introducción

El análisis automatizado de grandes volúmenes de texto no estructurado es uno de los desafíos centrales en el campo del Big Data. En este trabajo, se propone un enfoque escalable para procesar un corpus de noticias de la BBC, con el objetivo de identificar estructuras temáticas latentes sin necesidad de etiquetado manual previo.

Siguiendo los lineamientos de la cátedra y tomando como referencia técnica el **Capítulo 6 del libro "Advanced Analytics with PySpark"**, se desarrolló una solución basada en el ecosistema de **Apache Spark**. Específicamente, se implementó un pipeline ETL (*Extract, Transform, Load*) integrado con **Spark NLP** para la limpieza lingüística avanzada y la librería **MLlib** para el modelado no supervisado mediante *Latent Dirichlet Allocation* (LDA).

2. Metodología

2.1. Recopilación y Exploración de Datos

Se utilizó el dataset público **BBC News Archive**, que contiene 2225 artículos distribuidos en cinco categorías temáticas. Para garantizar la reproducibilidad y robustez del flujo de datos, se automatizó la ingesta desde un repositorio remoto (GitHub), simulando un entorno de producción.

- **Ingesta:** Se utilizó la librería `urllib` para la descarga dinámica del archivo `csv`.

- **Lectura Robusta:** Dado que los textos periodísticos suelen contener caracteres complejos y saltos de línea internos, se configuró el `SparkReader` con las opciones `multiLine=True` y `escape='`. Esto aseguró la integridad de los datos.

2.2. Preprocesamiento del Texto (Spark NLP)

Se diseñó un *pipeline* de transformación secuencial e inmutable utilizando **Spark NLP**. Las etapas clave fueron:

1. **Limpieza Básica:** Normalización de caracteres y conversión a minúsculas (*Normalizer*).
2. **Filtrado de Stopwords:** Se utilizó una lista estándar de palabras vacías en inglés, enriquecida manualmente con ruido de dominio (términos frecuentes como *'mr', 'said', 'would'*).
3. **Corrección de Pronombres y Verbos:** Se detectó que el lematizador transformaba erróneamente el término "us" (de *United States*) en el pronombre "we". Además, verbos genéricos como "go" (y sus conjugaciones *went/goes*) ensuciaban los tópicos. Ambos casos fueron corregidos agregando estos términos a la lista de exclusión.
4. **Lematización vs. Stemming:** Se optó por la **Lematización** en lugar del Stemming. Esta decisión prioriza la interpretabilidad semántica de los tópicos (ej: *'better'* $\rightarrow$ *'good'*), fundamental para el análisis cualitativo.

2.3. Vectorización: Selección de Características

Para transformar el texto limpio en una representación matemática, se experimentó con dos estrategias:

- **Conteo Simple (*CountVectorizer*):** Representa los documentos basándose en la frecuencia absoluta de los términos.
- **TF-IDF:** Ponderación estadística que penaliza las palabras que aparecen en demasiados documentos.

2.4. Modelado y Evaluación

Se aplicó el algoritmo LDA configurado con $k = 5$ tópicos y una semilla aleatoria (`seed=1234`) para reproducibilidad. La selección del modelo se basó en la métrica de **Log Perplexity** (Perplejidad Logarítmica), donde un valor menor indica una mejor capacidad predictiva.

3. Resultados

3.1. Métricas del Modelo y Selección

Se comparó el desempeño cuantitativo de las dos estrategias de vectorización.

Estrategia de Vectorización	Log Likelihood	Log Perplexity
Conteo Simple (Elegido)	-1,784,641	**6.525**
TF-IDF	-4,218,207	6.612

Tabla 1: Comparativa de modelos. El modelo de Conteo presentó menor perplejidad.

El modelo basado en **Conteo Simple** presentó una mejor perplejidad (6.52). Es notable mencionar que este valor supera al benchmark de referencia citado en la bibliografía de la cátedra para datasets similares (6.77 en Wikipedia según los autores), lo que valida la calidad del preprocesamiento.

3.2. Tópicos Identificados

El modelo ganador logró segmentar el corpus en cinco tópicos semánticamente coherentes:

Tópico 0 (Economía y Mercados): *bank, rise, market, growth, rate, fall, party, economy.*

Tópico 1 (Sociedad y Entretenimiento): *game, show, time, good, take, well, make, people.*

Tópico 2 (Tecnología y Negocios): *people, firm, use, company, phone, music, mobile, technology.*

Tópico 3 (Cine y Cultura): *film, good, award, star, win, england, year, world.*

Tópico 4 (Política y Gobierno): *government, minister, plan, country, company, tell, firm, uk.*

4. Descubrimiento

El hallazgo más relevante del análisis es la superioridad del modelo de **Conteo Simple** sobre TF-IDF en este contexto específico.

Si bien TF-IDF es teóricamente robusto, en este dominio periodístico las palabras de alta frecuencia (como *"government"* en política o *"market"* en economía) actúan como discriminadores fuertes de la categoría. La penalización impuesta por TF-IDF redujo la claridad de estas señales, resultando en una perplejidad ligeramente mayor (6.55 vs 6.52).

Asimismo, la corrección manual de los pronombres y la eliminación de algunos verbos y sus variantes (como go y went) fueron factores determinantes para obtener tópicos limpios, demostrando que la calidad de los datos es más crítica que la complejidad del algoritmo.

5. Conclusiones y Trabajo Futuro

El proyecto demostró exitosamente que **PySpark** y **Spark NLP** ofrecen un entorno robusto y escalable para transformar datos crudos en conocimiento estructurado.

Conclusiones Principales:

1. El modelo LDA es eficaz para recuperar la estructura latente de noticias sin supervisión humana.
2. La **Lematización** es preferible al Stemming cuando la interpretabilidad de los tópicos es prioritaria.
3. La validación mediante métricas objetivas alineadas con la bibliografía (Perplejidad) es esencial para fundamentar decisiones técnicas.

Trabajo Futuro:

- Implementar una búsqueda de hiperparámetros (*Grid Search*) para optimizar automáticamente el valor de k.
- Explorar métricas de Coherencia de Tópicos (C_v) como complemento a la perplejidad.
- Incorporar modelos de *Word Embeddings* (como BERT) en el pipeline de Spark NLP.

A. Anexo: Fragmentos de Código

A.1. Ingesta Robusta desde GitHub

```
import urllib.request
# Descarga automatica y manejo de multilineas
url = "https://github.com/juandimeglio25/TP_Prog/raw/refs/heads/
    main/bbc-news-data.csv"
urllib.request.urlretrieve(url, "bbc_news.csv")

df_raw = spark.read \
    .option("header", True) \
    .option("delimiter", ",") \
    .option("multiLine", True) \
    .option("escape", '"') \
    .csv("bbc_news.csv")
```

A.2. Pipeline de NLP (Limpieza Refinada)

```
# Inclusion de stopwords personalizados y correccion US/WE/GO
palabras_basura = StopWordsCleaner.loadDefaultStopWords("english")
palabras_basura.extend([
    "us", "we", "our",          # Pronombres problematicos
    "mr", "said", "would",      # Ruido de noticias
    "go", "going", "went"       # Verbos genericos
])

stopwords_cleaner = StopWordsCleaner() \
    .setInputCols(["normalized"]) \
    .setOutputCol("clean_tokens") \
    .setStopWords(palabras_basura)

lemmatizer = LemmatizerModel.pretrained("lemma_antbnc", "en") \
    .setInputCols(["clean_tokens"]) \
    .setOutputCol("lemma")
```