

# Text Feature Engineering on Azure

---

## Lab 4

### I. Introduction

In previous labs, you prepared and cleaned the Goodreads dataset — ingesting, transforming, and standardizing reviews, books, and authors into a structured format. You now have a curated dataset stored in the **Gold layer**, where each record represents a cleaned book review with its metadata.

This lab extends that workflow by introducing the crucial step of **text feature extraction**, which transforms raw text into machine-understandable numerical features. While traditional attributes like rating, n\_votes, and date\_added are already numerical, the **review text** contains rich semantic information about user opinions and sentiment that models can learn from — but only after being converted into quantitative representations.

In this exercise, you will create and analyze various text-based features using **Python and Databricks**. The lab connects the conceptual knowledge from **Lecture 5** with practical implementation. You will use libraries such as **scikit-learn**, **NLTK**, and **spaCy** to engineer linguistic and statistical features (TF-IDF, N-grams, sentiment polarity, review length, etc.) that capture different dimensions of text meaning.

By the end of this lab, you will have:

- A cleaned and vectorized text feature matrix ready for model training.
- An understanding of how preprocessing decisions affect downstream model performance.
- A new dataset in the **Gold layer**, which will be used in the upcoming lab for **feature selection** and **modeling**.

This marks the transition from **data preparation** to **model readiness** — turning your curated reviews into high-quality, learnable features.

### II. Splitting the dataset

To avoid data leakage, split the data first, before fitting text vectorizers and encoders only on the training portion.

- Create three splits:
  - **Train** ( $\approx 70\%$ ): used to fit TF-IDF, sentiment thresholds, and any scalers.
  - **Validation** ( $\approx 15\%$ ): used to tune models and feature sets.
  - **Test** ( $\approx 15\%$ ): kept untouched for final evaluation.
- Save the data into the gold layer (feature\_v2) with three subfolders.

## III. Text Feature Extraction in Databricks

After cleaning and curating the dataset in the **Gold layer**, in this stage, you will transform unstructured review text into structured numerical features that capture meaning, sentiment, and linguistic patterns.

### III.1. What you will achieve

This lab focuses on **text feature engineering**, as covered in Lecture 5 on *Feature Engineering in the Modern Machine Learning Lifecycle*. You will apply the core principles of text representation:

- Cleaning and normalizing raw text.
  - Tokenizing words or characters.
  - Converting text into numerical vectors using **TF-IDF**, **N-grams**, and embeddings.
  - Creating document-level features (e.g., review length, sentiment).
- These representations will later feed machine-learning models for predicting rating, detecting sentiment, or summarizing book feedback.

### III.2. Create a new Databricks notebook named `goodreads_text_features`.

- Load the last version of the curated Gold dataset (`feature_v2/train`) from your storage.
- You will work directly with the `review_text` column.

### III.3. Perform text cleaning and normalization

- Convert all text to lowercase.
- Remove punctuation and extra spaces.
- Replace URLs, numbers, or emojis with placeholders if they appear in text.

---

You should use the `re` library for regular expressions to detect and replace URLs and numbers, and the `emoji` library to handle emojis if present in the text.

---

- Trim text and filter out empty or very short reviews (<10 characters).

### III.4. Extract text-based features

#### a. Basic Text Features

- `review_length_words` – number of words in each review.
- `review_length_chars` – number of characters in each review.

These quantify verbosity and writing style differences between reviewers. **Suggested libraries:** nltk, re, or Python string methods (`split`, `len`).

## b. Sentiment Features

Using **VADER SentimentIntensityAnalyzer** from nltk.sentiment or **TextBlob**:

- **sentiment\_pos** – proportion of positive sentiment words.
- **sentiment\_neg** – proportion of negative sentiment words.
- **sentiment\_neu** – proportion of neutral sentiment words.
- **sentiment\_compound** – overall normalized polarity score (-1 = very negative, +1 = very positive).

These features capture emotional tone and opinion polarity. **Suggested libraries:** nltk.sentiment.vader, textblob.

## c. TF-IDF Features

Using **scikit-learn's TfidfVectorizer** or **CountVectorizer**:

- Creates a **high-dimensional matrix** representing the importance of each word or phrase.
- Supports **unigrams, bigrams, or trigrams**.

Recommended settings:

- `max_features`: limit vocabulary size to top N words.
- `stop_words='english'`: remove common filler words.
- `ngram_range=(1,2)`: capture short context like “not good”.

These features represent word frequency and relevance.

## d. Semantic Embedding Features

Using **transformer-based models** (e.g., **BERT**, **DistilBERT**, or **Sentence-BERT**) to extract dense semantic embeddings:

- **bert\_embedding** – contextual vector representation for each review capturing meaning beyond surface words.
- These embeddings reflect relationships between words and phrases, improving downstream model accuracy.

**Suggested libraries:** transformers (Hugging Face), sentence-transformers.

## e. Additional features

---

Any additional or informative feature is a bonus.

---

# IV. Combined Feature Set and Output

After extraction:

- Combine all numerical features (sentiment, length, embeddings) with TF-IDF vectors into one feature matrix.
- Merge the matrix with metadata columns (review\_id, book\_id, rating).
- Save the final feature dataset to the **Gold layer** under **features\_v2** for modeling.

The resulting dataset combines **statistical, semantic, and emotional dimensions** of text, enabling robust predictive modeling in the next lab.

---

As usual, the submission is a the link to your GitHub

- You use the same gir as previously. Create a new branch `text_feature_extraction`.
  - The grade depends on the feature extraction.
  - Saving the training Dataset with all features.
    - The **clarity and consistency of the resulting columns** (proper naming, appropriate types, meaningful derived features).
  - Your explanations.
-