Python Feature Engineering - Capítulo 11: "Extracting Features from Text Variables"

Soledad Galli

October 29, 2025

Análisis Detallado del Capítulo 11: Extrayendo Características de Variables de Texto (Versión Optimizada para PDF)

Este capítulo es la puerta de entrada para que las máquinas "entiendan" el lenguaje humano. Su objetivo principal es convertir el texto, que es caótico y no estructurado, en una tabla de números limpios y ordenados que un algoritmo de machine learning pueda procesar.

Receta 1: Conteo de Características Estadísticas

Concepto Clave: Antes de intentar entender *qué dice* un texto, podemos obtener muchísima información analizando *cómo está escrito*. Esta técnica es como un primer diagnóstico médico: no requiere una cirugía compleja, solo tomar medidas superficiales para tener una idea general del estado del "paciente" (el texto). Se centra en la **verbosidad**, la **riqueza léxica**, la **repetitividad** y la **complejidad** del lenguaje.

Ejemplo Sencillo y Visual:

Imagina que tenemos dos reseñas de un producto. En lugar de una tabla, analizaremos cada una por separado.

Texto A: "Bueno. Me gusta. Es un buen producto, muy bueno."

- **Palabras Totales:** 9
- **Vocabulario (Palabras Únicas):** 5 (`bueno`, `me`, `gusta`, `es`, `un`, `producto`, `muy`)
- **Longitud Media de Palabra:** 3.8 caracteres

Texto B: "El producto superó mis expectativas. La calidad del material es excepcional y su funcionamiento resulta bastante intuitivo."

- **Palabras Totales:** 18
- **Vocabulario (Palabras Únicas):** 18 (todas son únicas)
- **Longitud Media de Palabra:** 5.9 caracteres

Conclusión del Ejemplo: Al comparar las métricas, sin siquiera leer en detalle, los números nos indican que el **Texto B es más largo, utiliza un vocabulario más rico y emplea palabras más complejas**. Esto sugiere que es una reseña mucho más informativa y detallada que el Texto A, que es breve y repetitivo.

Receta 2: Conteo de Oraciones

Concepto Clave: El número de oraciones nos da una idea de la **estructura del texto**. Un texto compuesto por muchas oraciones cortas tiene un ritmo y estilo muy diferente a uno con pocas oraciones largas y complejas. La "tokenización de oraciones" es el proceso de identificar los límites de estas unidades de pensamiento (generalmente marcados por puntos, signos de exclamación, etc.).

Analogía: Piensa que el texto es una conversación. El número de palabras es cuánto tiempo habló alguien. El número de oraciones es cuántas ideas distintas o puntos separados expresó en esa conversación.

Ejemplo Sencillo y Visual:

- **Input (un único bloque de texto):**

  `"Me encanta este producto. Funciona a la perfección! Lo recomiendo al 100%."`

- **Proceso (Tokenización de Oraciones):**

  El sistema busca los delimitadores (`.` y `!`) que indican el final de una idea.

- **Output (una lista de oraciones):**

  1. `"Me encanta este producto."`
  2. `"Funciona a la perfección!"`
  3. `"Lo recomiendo al 100%."`

- **Resultado Final (conteo):**

  El texto contiene **3** oraciones.

Importante: Este paso **debe realizarse antes de eliminar la puntuación**, ya que depende de ella para funcionar.

Receta 3: Bag-of-Words (BoW) y N-gramas

Concepto Clave: Esta es la primera técnica que empieza a mirar *el contenido* del texto.

- **Bag-of-Words (BoW):** Ignora la gramática y el orden, y simplemente cuenta la frecuencia de cada palabra. Es como meter todas las palabras de un libro en una bolsa y contarlas.

- **N-gramas:** Resuelve parcialmente la pérdida de contexto de BoW al contar secuencias de palabras (`"no me gusta"` vs. `"me gusta"`).

Ejemplo Sencillo y Visual:

Oración: `"El perro persigue al gato y el gato persigue al ratón."`

1. Vocabulario (palabras únicas):

`['el', 'perro', 'persigue', 'al', 'gato', 'y', 'ratón']`

2. Representación Bag-of-Words (1-gramas):

Presentamos el conteo de cada palabra del vocabulario en formato de lista:

- `el`: 2
- `perro`: 1
- `persigue`: 2
- `al`: 2
- `gato`: 2
- `y`: 1
- `ratón`: 1

3. Representación con 2-gramas (bigramas):

Ahora las características son pares de palabras.

- `"el perro"`: 1
- `"perro persigue"`: 1
- `"persigue al"`: 1

- `"al gato"`: 1
- `"gato y"`: 1
- `"y el"`: 1
- `"gato persigue"`: 1
- `"al ratón"`: 1

Receta 4: TF-IDF (Frecuencia de Término–Frecuencia Inversa de Documento)

Concepto Clave: TF-IDF es una versión "inteligente" de Bag-of-Words. Asigna un **peso de importancia** a cada palabra. La idea central es que las palabras más importantes son aquellas que aparecen **muchas veces en un documento**, pero **pocas veces en el resto de los documentos**.

Analogía: Eres un detective revisando documentos. La palabra "informe" (TF alto, IDF bajo) no te dice nada. La palabra "coartada" (TF alto en un doc, IDF alto) es una pista clave. TF-IDF resalta estas "pistas".

Ejemplo Sencillo y Visual:

Corpus de 2 documentos:

- **Doc 1 (Deportes):** "El equipo ganó el partido de fútbol."
- **Doc 2 (Política):** "El equipo del ministro ganó el debate."

Analicemos cada palabra clave:

- **Palabra: "el"**
  - **IDF:** Muy Bajo (aparece en todos los documentos, es común).
  - **Resultado TF-IDF:** Tanto en Doc 1 como en Doc 2, su peso será **casi 0**.
- **Palabra: "equipo"**

- **IDF:** Muy Bajo (aparece en todos los documentos).
- **Resultado TF-IDF:** En ambos documentos, su peso será **casi 0**.

- **Palabra: "fútbol"**
  - **TF (Doc 1):** 1 | **TF (Doc 2):** 0
  - **IDF:** Muy Alto (solo aparece en el Doc 1, es un término distintivo).
  - **Resultado TF-IDF (Doc 1):** 1 * Muy Alto = **ALTO**.
  - **Resultado TF-IDF (Doc 2):** 0 * Muy Alto = **0**.

- **Palabra: "ministro"**
  - **TF (Doc 1):** 0 | **TF (Doc 2):** 1
  - **IDF:** Muy Alto (solo aparece en el Doc 2).
  - **Resultado TF-IDF (Doc 1):** 0 * Muy Alto = **0**.
  - **Resultado TF-IDF (Doc 2):** 1 * Muy Alto = **ALTO**.

TF-IDF identifica automáticamente "fútbol" y "ministro" como los términos más relevantes de cada documento, respectivamente.

Receta 5: Limpieza y Derivación (Stemming) de Variables de Texto

Concepto Clave: Este es el proceso de "preparar los ingredientes" antes de aplicar BoW o TF-IDF. El objetivo es la **normalización**: reducir las variaciones de las palabras para que el modelo no se confunda.

Analogía: Es como organizar una biblioteca: quitas etiquetas (limpieza), ignoras libros titulados "El" (stop words) y agrupas todos los libros de "Aventuras" y "Aventureros" en una sola sección "Aventura" (stemming).

Ejemplo Sencillo y Visual (Pipeline de Limpieza):

1. **Texto Original:**

   `"Los ingenieros están desarrollando nuevas tecnologías!!!"`

2. → **Convertir a Minúsculas:**

   `"los ingenieros están desarrollando nuevas tecnologías!!!"`

3. → **Eliminar Puntuación/Números:**

   `"los ingenieros están desarrollando nuevas tecnologías"`

4. → **Eliminar Stop Words (`los`, `están`):**

   `"ingenieros desarrollando nuevas tecnologías"`

5. → **Aplicar Stemming (reducción a la raíz):**

   `"ingenier desarroll nuev tecnolog"`