Preprocesamiento de Texto

Fabián Villena

Introducción

Dada la naturaleza no estructurada del texto debemos realizar un preprocesamiento del mismo para generar una sistematización inicial del texto para que los métodos que posteriormente se aplicarán en los textos funcionen de mejor manera.

También se busca disminuir la dimensionalidad del texto para mejorar el rendimiento de los modelos que se ajustarán posteriormente.

Glosario

- Token: Unidad mínima de análisis la cual la mayoría de las veces se refiere a una palabra.
- Vocabulario: Conjunto de todos los tokens distintos de un corpus.
- Documento: Conjunto de palabras que representan un mismo tema.
- *Corpus*: Conjunto de documentos.
- *Corpora*: Plural de *corpus*.

Normalización del texto

Antes de la realizar la mayoría de las tareas de procesamiento de lenguaje natural se debe normalizar el texto, este procedimiento cuenta con al menos tres pasos:

1. Tokenización (segmentación) de palabras
2. Estandarización de los formatos de las palabras
3. Segmentación de oraciones

Tokenización

La tokenización es el proceso de transformación de cadenas de texto en elementos más pequeños como palabras o *tokens*.

Normalmente podríamos simplemente dividir cadenas de caracteres por espacios, pero típicamente existen elementos que queremos tratar como distintos segmentos y estos no están separados por espacios:

¡Hola!, hoy es 2025-11-13.

`¡ Hola ! , hoy es 2025-11-13`

Algoritmo de tokenización

El **algoritmo que se utilice para tokenizar debe ser consistente con el lenguaje que estamos analizando**; distintos lenguajes expresan de distinta manera la utilización de distintos símbolos.

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"'?():-_`]  # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

Tokenizadores modernos

Existen métodos de tokenización más modernos que construyen el vocabulario desde la distribución de los símbolos presentes en un conjunto de entrenamiento.

Estos **tokenizadores modernos además de dividir el texto en palabras, también se generan subpalabras** que pueden ser concatenadas para generar palabras que no existían en el conjunto de entrenamiento.

La canción y la actuación, cantando y actuando ando.

```
La can# #ción y la actua# #ción , cant# #ando y actu# #ando
ando .
```

Byte-pair encoding

Este algoritmo parte con un vocabulario que consiste en todos los caracteres presentes en el texto e **iterativamente va agregando subpalabras al vocabulario al unir los pares adyacentes** de símbolos más frecuentes.

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens til k times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

Tokenizadores y modelos

En el modelo de BERT en español desarrollado por el DCC se utilizó un tokenizador basado en el algoritmo SentencePiece que aprende el vocabulario desde el conjunto de entrenamiento y genera subpalabras.

Este tokenizador se distribuye al igual que el modelo en sí, para poder obtener resultados reproducibles.

La biblioteca `tokenizers` contiene todas las implementaciones de tokenizadores modernos.

https://huggingface.co/dccuchile/bert-base-spanish-wwm-uncased

- ##viste
- tienda
- ##ple
- ##nud
- mó
- rode
- pris
- presentó
- quedó
- ##telo
- ##gados
- lev
- ##arle
- dipu
- europeas
- ##rue

Tokens, Types y Word tokens

Los tokens son segmentos de texto que representan una unidad de análisis. Estos tokens pueden ser palabras o Word tokens o pueden ser tokens que no representan palabras.

A cada una de los tokens distintos se les llama types y estos forman el vocabulario; la lista de todas las palabras distintas que existen en un *corpus*.

Me gustan los lenguajes de programación y los lenguajes naturales.

```
tokens = [Me, gustan, los, lenguajes, de, programación, y,
          los, lenguajes, naturales, .]
```

```
vocabulary = {Me, gustan, los, lenguajes, de, programación,
              y, naturales, .}
```

Lematización

Esta estrategia de normalización de texto se basa en la **transformación de una palabra en su forma flexionada hacia su lema**, el cual es la forma canónica de un conjunto de palabras relacionadas a través de una inflexión que representan el mismo significado.

cantando -> cantar
canté -> cantar
correrá -> correr
pensado -> pensar
estrellado [verbo] -> estrellar
estrellado [adjetivo] -> estrellado
gatos -> gato
gata -> gato

Modelos para lematización

Para la correcta lematización de una palabra se deben tener modelos que identifiquen la categoría gramatical de la palabra, el significado de la palabra y el contexto en donde ocurre la palabra.

Tomando en cuenta todos estos parámetros, se puede llevar la palabra a su forma canónica de manera correcta.

Stemming

Los algoritmos de lematización hacen uso de complejos analizadores morfológicos para realizar la normalización. Por esta razón es que existen algoritmos más simples que buscan llegar a resultados similares.

Estos algoritmos más simples principalmente consisten en el **corte directo de los sufijos de las palabras**.

cantando -> cant
canté -> cant
correrá -> corr
pensado -> pens
estrellado -> estrell
gatos -> gat

Algoritmos de stemming

Los algoritmos de stemming implementan simples heurísticas para modificar las palabras e intentar llegar a su raíz.

Algunos algoritmos importantes son:

- Porter Stemmer
- Snowball Stemmer

La idea general de los stemmers es cortar las colas de las palabras de la manera más sistemática posible.

Utilización de stemming y lematización

Si queremos reducir el tamaño del vocabulario, por ejemplo para visualización o para mejorar la generalización del modelo, podemos realizar stemming y lematización.

Uno de los **problemas más frecuentes en el despliegue de modelos en producción es la aparición de palabras fuera de vocabulario**, pero si sólo estamos utilizando en nuestro modelo, raíces de palabras, es menos probable que esto pase.

Remoción de stopwords

Hay ciertas palabras que no aportan significado a las oraciones que componen. A este tipo de palabras se les denomina stopwords y típicamente se eliminan de los documentos mediante la utilización de un filtro.

Existen listas de palabras denominadas como stopwords para lenguajes y dominios específicos.

```
a, actualmente, adelante,
además, afirmó, agregó,
ahora, ahí, al, algo,
alguna, algunas, alguno,
algunos, algún, alrededor,
ambos, ante, anterior,
antes, apenas,
aproximadamente, aquel,
aquellas, aquellos, aqui,
aquí, arriba, aseguró,
así, atras, aunque, ayer,
añadió, aún, bajo,
bastante, bien, buen,
buena, buenas, bueno,
buenos, cada, casi, cerca.
```