

Basics

Tokenization

La tokenización es el proceso de dividir el texto en unidades mínimas significativas como palabras, signos de puntuación, símbolos, etc.

**Word-based
tokenizer**

**Character-based
tokenizer**

**Subword-based
tokenizer**

La tokenización suele ser el primer paso en todo proceso de NLP.

Este paso es necesario porque la máquina necesita descomponer los datos del lenguaje natural en los elementos más básicos (*tokens*) para poder analizar cada elemento en el contexto de los demás elementos.



Tokenization

Word-based tokenizer

```
['Nació', 'con', 'el', 'don', 'de', 'la',  
'risa', 'y', 'con', 'la', 'intuición', 'de',  
'que', 'el', 'mundo', 'estaba', 'loco.', 'Y',  
'ese', 'era', 'todo', 'su',  
'patrimonio.']
```

Character-based tokenizer

```
['N', 'a', 'c', 'i', 'ó', ' ', 'c', 'o', 'n', ' ',  
'e', 'l', ' ', 'd', 'o', 'n', ' ', 'd', 'e', ' ',  
'l', 'a', ' ', 'r', 'i', 's', 'a', ' ', 'y', ' ',  
'c', 'o', 'n', ' ', 'l', 'a', ' ', 'i', 'n', 't',  
'u', 'i', 'c', 'i', 'ó', 'n', ' ', 'd', 'e', ' ',  
'q', 'u', 'e', ' ', 'e', 'l', ' ', 'm', 'u', 'n',  
'd', 'o', ' ', 'e', 's', 't', 'a', 'b', 'a', ' ',  
'l', 'o', 'c', 'o', '.', ' ', 'Y', ' ', 'e', 's',  
'e', ' ', 'e', 'r', 'a', ' ', 't', 'o', 'd', 'o',  
' ', 's', 'u', ' ', 'p', 'a', 't', 'r', 'i', 'm',  
'o', 'n', 'i', 'o', '.']
```

"Nació con el don de la risa y con la intuición de que el mundo estaba loco. Y ese era todo su patrimonio."

Subword-based tokenizer

```
['na', '##cio', 'con', 'el', 'don', 'de', 'la', 'ri',  
'##sa', 'y', 'con', 'la', 'int', '##ui', '##cion', 'de',  
'que', 'el', 'mundo', 'est', '##aba', 'loco', '.', 'y',  
'es', '##e', 'era', 'tod', '##o', 'su', 'pat', '##rim',  
'##oni', '##o', '.']
```

Los modelos más avanzados de NLP
tienen sus propios tokenizadores

<https://platform.openai.com/tokenizer>



Tokenizer

El primer paso para poder trabajar un algoritmo mediante cómputo matemático es contar con un **dataset numérico**.

La tokenización consiste en **traducir caracteres a números, palabras a números** o **partes de palabras a números**.

Una secuencia de palabras se transforma a una **secuencia de números**




Word Tokenizer

Se crean **diccionarios** a partir de un conjunto de frases o palabras que asocia a cada "palabra" un número.

A partir de un diccionario, podemos traducir cualquier texto a una **secuencia** de números.

Tensorflow tiene su propio Toolkit para procesamiento NLP, como el Tokenizer. Pero hay otras muchas librerías que proporcionan estos elementos: **Spacy**, **NLTK**, **sklearn**, **Gensim**...

```
from tensorflow.keras.preprocessing.text import Tokenizer  
  
tokenizer = Tokenizer(num_words = 100)
```




```
sentences = [  
    'Me gusta la tortilla de patatas',  
    'La tortilla de champiñones es mucho más jugosa'  
]  
  
tokenizer.fit_on_texts(sentences)
```

diccionario

```
{'la': 1,  
 'tortilla': 2,  
 'de': 3,  
 'me': 4,  
 'gusta': 5,  
 'patatas': 6,  
 'champiñones': 7,  
 'es': 8,  
 'mucho': 9,  
 'más': 10,  
 'jugosa': 11}
```

```
tokenizer.texts_to_sequences(sentences)
```



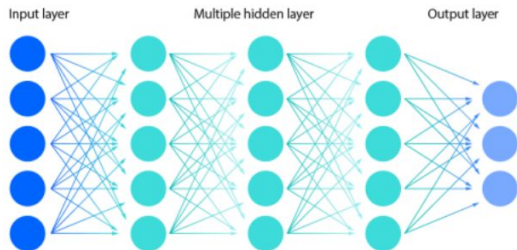
```
[[4, 5, 1, 2, 3, 6], [1, 2, 3, 7, 8, 9, 10, 11]]
```



Padding

Padding es la operación con la que ajustamos las secuencias de varios vectores a una longitud común.

Una de las ventajas es que, cuando vamos a introducir estos vectores en Redes Neuronales, el padding nos permite estandarizar las dimensiones de los *inputs*.



```
from tensorflow.keras.preprocessing.sequence import pad_sequences

sentences = [
    'Me gusta la tortilla de patatas',
    'La tortilla de champiñones es mucho más jugosa'
]

tokenized_sequences = tokenizer.texts_to_sequences(sentences)
padded_sequences = pad_sequences(tokenized_sequences)
```

padding

```
[ 0, 0, 4, 5, 1, 2, 3, 6]
[ 1, 2, 3, 7, 8, 9, 10, 11]
```

El "[pad]" es simplemente un token de relleno que no tiene significado. El modelo aprenderá a ignorar estos tokens durante el entrenamiento.

Stemming & Lemmatizing

Stemming

Proceso heurístico que corta los extremos de las palabras con la intención de llegar a una forma base común, llamada raíz (*stem*).

Lemmatization

Proceso de reducir una palabra a su forma base, conocida como lema.

La lematización tiene en cuenta el análisis morfológico de las palabras, por lo que el lema pertenece siempre a un lenguaje y tiene sentido por sí mismo.

Stemming

```
Me -> me
gusta -> gust
la -> la
tortilla -> tortill
de -> de
patatas -> patat
La -> la
tortilla -> tortill
de -> de
champiñones -> champiñon
es -> es
mucho -> much
más -> mas
jugosa -> jugos
```

Lemmatization

```
Me -> yo
gusta -> gustar
la -> el
tortilla -> tortilla
de -> de
patatas -> patata
La -> el
tortilla -> tortilla
de -> de
champiñones -> champiñón
es -> ser
mucho -> mucho
más -> más
jugosa -> jugoso
```

Stopwords

Es un listado de palabras que son muy comunes y que, por sí solas, **aportan poco valor semántico** al significado de un texto. Algunas de estas palabras suelen ser:

- **Artículos:** el, la, los, las, un, una, unos, unas.
- **Preposiciones:** a, ante, bajo, con, contra, de, desde, en, entre, hacia, hasta, por, según, sin, sobre, tras.
- **Pronombres personales:** yo, tú, él, ella, nosotros, vosotros, ellos, ellas.
- **Conectores y conjunciones:** y, e, ni, o, u, pero, aunque, sino, porque, como.
- **Adverbios comunes:** muy, bien, mal, antes, después, tarde, pronto, aquí, allí, ahora, antes.
- **Verbos auxiliares y modales:** ser, estar, haber, tener, poder, deber.

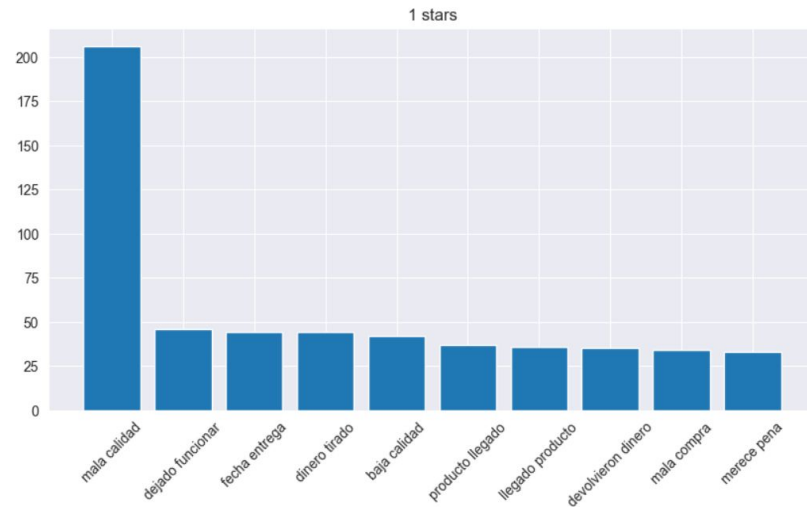
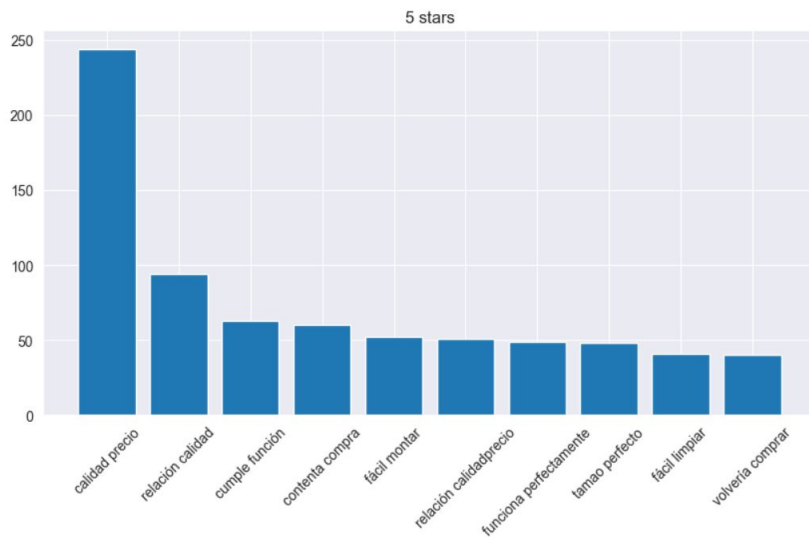
Estas palabras suelen filtrarse en el análisis de texto para enfocarse en palabras que aportan significados más específicos y relevantes al contenido que se está examinando.



N-Grams

Son **combinaciones contiguas de n palabras** de una secuencia dada de texto:

- **Bigramas (2-grams):** “inteligencia artificial”, “café caliente”.
- **Trigramas (3-grams):** “exploratory data analysis”, “taza de café”.
- ...



TF-IDF algorithm

Term-Frequency Inverse-Document-Frequency es un algoritmo utilizado en NLP donde se recupera información para reflejar la importancia de una palabra en un documento dentro de una colección de documentos o corpus.

Este método es ampliamente utilizado para la extracción de características en tareas como la clasificación de textos, el análisis de sentimientos y la recomendación de sistemas. Permite a los sistemas de búsqueda y otros sistemas de procesamiento de texto priorizar las palabras en los documentos en función de dos fórmulas: TF y IDF.

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

Esta puntuación es alta para términos que son frecuentes en un documento particular pero no en muchos documentos del corpus, lo que significa que estos términos son potencialmente más relevantes para ese documento.

El TF se calcula para cada documento, mientras que el IDF se calcula de forma global al corpus.



TF-IDF algorithm

TF (Term-Frequency): Refleja la frecuencia con la que un término (palabra) aparece en un documento. Es simplemente el número de veces que aparece un término en un documento, dividido por el número total de términos en ese documento. Esto ayuda a ajustar por el hecho de que algunos documentos son más largos que otros.

1

$$TF(t, d) = \frac{\text{Número de veces que el término } t \text{ aparece en el documento } d}{\text{Total de términos en el documento } d}$$

IDF (Inverse-Document-Frequency): Evalúa la importancia de un término en todo el corpus. La idea es que si una palabra aparece en muchos documentos, es menos significativa para identificar un documento en particular. Se calcula tomando el logaritmo de la división del número total de documentos por el número de documentos que contienen el término.

2

$$IDF(t, D) = \log \left(\frac{\text{Número total de documentos en el corpus } D}{\text{Número de documentos donde aparece el término } t} \right)$$



Ejemplo

Corpus:

d_1 : "El gato juega en el jardín" \rightarrow [el, gato, juega, en, el, jardín]

d_2 : "El perro juega en el parque" \rightarrow [el, perro, juega, en, el, parque]

$$D = \{d_1, d_2\}$$

TF se calcula dividiendo el número de veces que un término aparece en un documento por el número total de términos en ese documento

$$TF("gato", d_1) = 1/6 \approx 0.167$$

$$TF(t, d) = \frac{\text{Número de veces que el término } t \text{ aparece en el documento } d}{\text{Total de términos en el documento } d}$$

IDF se calcula tomando el logaritmo de la división del número total de documentos por el número de documentos que contienen el término.

$$IDF("gato") = \log(2/1) = \log(2) \approx 0.693$$

$$IDF(t, D) = \log \left(\frac{\text{Número total de documentos en el corpus } D}{\text{Número de documentos donde aparece el término } t} \right)$$

$$TF-IDF("gato", d_1) = TF("gato", d_1) \times IDF("gato") \approx 0.167 \times 0.693 \approx \mathbf{0.116}$$

Ejemplo

Term	TF (d_1)	TF (d_2)	IDF	TF-IDF (d_1)	TF-IDF (d_2)
juega	0.167	0.167	0.000	0.000	0.000
en	0.167	0.167	0.000	0.000	0.000
gato	0.167	0.000	0.693	0.116	0.000
jardín	0.167	0.000	0.693	0.116	0.000
el	0.333	0.333	0.000	0.000	0.000
perro	0.000	0.167	0.693	0.000	0.116
parque	0.000	0.167	0.693	0.000	0.116

Estas serían las palabras “más importantes” de cada texto.



Further Readings

Este tema se puede completar con el algoritmo **Latent Dirichlet Allocation** (LDA, 2003). Este algoritmo se utiliza para describir colecciones de documentos con palabras que se generan a partir de una mezcla de temas. Así, cada tema se caracteriza por una distribución de palabras (*topic modeling*).

Un uso muy habitual es el de clasificar una colección de textos no supervisados en diferentes grupos. El caso de uso es muy similar al visto de Tf-IDF + KMeans de clase. Sin embargo, el enfoque matemático es distinto.

Se propone al estudiante que investigue más sobre este algoritmo.



Further Readings

Este tema se puede completar con el algoritmo **BM25** (2003). Este algoritmo se basa en el Tf-IDF, pero ampliando la fórmula original mejorando especialmente el ajuste de la importancia de las palabras repetidas y es más justo con documentos de diferentes longitudes

$$\text{Score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgl}})}$$

Se propone al estudiante que investigue más sobre este algoritmo.

