



Grado en Inteligencia Artificial – Fundamentos de Procesamiento de Lenguaje Natural

Curso 2024-2025

Práctica I: Tokenización

El objetivo de esta práctica es familiarizarse con diferentes métodos de tokenización, es decir, métodos para segmentar oraciones en lenguaje natural en unidades léxicas o subléxicas, tradicionalmente conocidas como *tokens*. La tokenización es un paso fundamental en la mayoría de las tareas de Procesamiento de Lenguaje Natural (PLN), ya que define cómo se representan y manipulan los textos en etapas posteriores para la mayoría de tareas (por ejemplo, traducción automática, extracción de información, agentes conversacionales, etc).

En esta práctica, exploraremos varias aproximaciones de tokenización, tanto clásicas como modernas:

Tokenización por espacios: Un modelo básico que separa las palabras de una oración utilizando espacios en blanco como único delimitador.

Tokenización por signos de puntuación: Un modelo que segmenta las oraciones considerando como tokens separados palabras, signos de puntuación y otros símbolos especiales (como por ejemplo, emojis).

Tokenización por n-gramas: Un modelo que genera secuencias de palabras contiguas de longitud n. Requiere un método de pretokenización (por ejemplo, la tokenización por espacios).

Tokenización Wordpiece o basada en subpalabras: Este método subléxico divide las palabras en prefijos, raíces o sufijos más pequeños, basándose en su frecuencia en un corpus. Usa un token especial [UNK] para representar palabras desconocidas que quedan fuera del vocabulario del tokenizador. Requiere también un método de pretokenización. Esta es la tokenización usada por (masked) language models como BERT o XLNET.

Tokenización BPE (Byte-Pair Encoding, aunque en la práctica lo haremos basada en caracteres) o basada emparejamiento de pares: Este método comienza dividiendo cada palabra en caracteres individuales y, mediante un proceso iterativo, fusiona los pares de caracteres más frecuentes para formar también subpalabras más grandes, *creando unas reglas de fusión*. Palabras frecuentes pueden mantenerse como unidades completas si aparecen de manera muy frecuente en el texto, mientras que palabras menos comunes se descomponen en componentes más pequeños. Requiere también un método de pretokenización. Esta es la tokenización usada por modelos como GPT-2 o GPT-3.

En la Tabla 1 se muestra un ejemplo de una posible tokenización válida según las distintas aproximaciones para la oración: “The spaceship is preparing for liftoff 🚀 ✨.” No necesariamente coincidirá con la que implementemos, dado que el resultado de la tokenización para algunos modelos depende del conjunto de oraciones usadas durante el entrenamiento.

Método	Tokens generados
Espacios	["The", "spaceship", "is", "preparing", "for", "liftoff", "🚀", "✨", "."]

Signos de puntuación	["The", "spaceship", "is", "preparing", "for", "liftoff", "🚀", "🌟", "."]
N-gramas	["The spaceship", "spaceship is", "is preparing", "preparing for", "for liftoff", "liftoff 🚀🌟."]
Subpalabras	["The", "space", "##ship", "is", "prep", "##aring", "for", "lift", "##off", "[UNK]", "."]
Emparejamiento de pares	["The", "spaceship", "is", "pr", "ep", "ar", "ing", "for", "liftoff", "🚀", "🌟", "."]

Tabla 1: Ejemplo de tokenización de la oración "The spaceship is preparing for liftoff 🚀🌟." utilizando diferentes métodos

Se pide:

- Implementación de los métodos:** Implementar los cinco métodos de tokenización de acuerdo con las especificaciones indicadas.
- Evaluar cada método** en el conjunto de oraciones de prueba proporcionado en el archivo `test_sentences.txt`, incluido en los materiales de la práctica. Para los métodos WordPiece y BPE, entrenar el modelo con el archivo `training_sentences.txt`, probar con tamaños de vocabulario de 100, 150 y 200, e imprimir el vocabulario por pantalla en cada caso. Imprimir la salida de la tokenización tanto en el conjunto de entrenamiento (`training_sentences.txt`) como en el de prueba (`test_sentences.txt`), siguiendo el formato indicado en el Anexo II, y para todos los métodos implementados. Finalmente, realizar una discusión sobre las diferencias observadas.
- Análisis del vocabulario:** Crear una representación visual que muestre cómo evoluciona el tamaño del vocabulario (número de tokens únicos) a medida que crece el número de oraciones procesadas. El gráfico debe incluir una comparación clara entre los métodos implementados. Se debe utilizar el archivo `majesty_speeches.txt`, que se incluye con los materiales de la práctica, para este propósito. En los métodos de tokenización WordPiece y BPE, podéis considerar establecer un tamaño máximo de vocabulario de 3000. Sin embargo, tenéis libertad para explorar otros valores y analizar su efecto.

Entrega

Las prácticas deberán realizarse en Python y deberá entregarse un programa que pueda ejecutarse desde la línea de comandos. Para esta entrega en particular, no se aceptarán notebooks ni otros formatos interactivos al no requerirse el acceso a GPUs para la ejecución.

Se deberá incluir un manual breve que explique cómo ejecutar el código, un análisis de los resultados obtenidos y una discusión sobre las diferencias observadas entre los distintos métodos. Además, será necesario discutir sus ventajas e inconvenientes. La memoria no podrá exceder las 3 páginas, sin contar la portada.

La práctica se realizará en grupos de tres personas.

La práctica deberá subirse al campus virtual, en el apartado habilitado para ello, como muy tarde el 1 de marzo de 2025 a las 23:59 horas. Únicamente un miembro del grupo deberá realizar la entrega.

La defensa de la práctica, de carácter obligatorio, tendrá lugar durante la semana siguiente. Los estudiantes que no asistan a dicha defensa obtendrán una calificación de 0 puntos en la práctica.

ANEXO I - Especificación

Tokenización por espacios

Este método debe dividir una oración en palabras usando únicamente los espacios en blanco como delimitadores.

Tokenización por signos de puntuación

Este método debe dividir las oraciones en palabras y signos de puntuación como unidades independientes. También debe considerar emojis y símbolos especiales como unidades separadas.

Consejo: Utilizar la librería regex de python (librería para expresiones regulares)

Patrón a usar: r"\w+|[\^\\w\\s]|\p{So}"

Ejemplo:

```
import regex as re  
  
pattern = r"\w+|[\^\\w\\s]|\p{So}"  
  
text = "¡Hola, mundo! 😊"  
  
tokens = re.findall(pattern, text, flags=re.UNICODE)
```

Tokenización por n-gramas

Este método debe dividir una oración en secuencias consecutivas de n palabras (n-gramas). El valor de n debe ser configurable y soportar valores mayores o iguales que 1. Como método preliminar para separar las palabras en unidades individuales, usaremos la tokenización por espacios.

Tokenización WordPiece o basada en subpalabras

Este método consta de dos fases principales: (1) la construcción del vocabulario, hasta alcanzar un tamaño máximo, y (2) el propio método de tokenización en base a ese vocabulario.

FASE 1: Construcción del vocabulario dado un conjunto de oraciones de entrenamiento

1. Preprocesar el corpus e inicializar el vocabulario:

- a. Pre-tokenizar cada oración del corpus mediante una tokenización simple basada en espacios.
- b. Calcular la frecuencia de cada palabra en el corpus. Este cálculo será clave para determinar qué subpalabras fusionar de manera eficiente.

- c. Inicializar el vocabulario con un token especial “[UNK]” (Unknown), que se usará para representar palabras fuera del vocabulario durante la etapa de tokenización.
- d. Segmentar cada palabra en caracteres individuales y añadir estos caracteres al vocabulario inicial. Los caracteres que no son iniciales se identificarán con un prefijo ‘##’ para indicar que forman parte de una subpalabra que continúa desde un carácter anterior. Por ejemplo, la palabra “gato” se segmentaría como “g”, “##a”, “##t”, “##o”. Cada uno de estos elementos se añadiría al vocabulario.
- e. *Guardar la segmentación inicial de cada palabra en un formato que se usará en pasos posteriores. Por ejemplo: {“gato”: [“g”, “##a”, “##t”, “##o”]}. Esta segmentación inicial servirá para rastrear las fusiones de subpalabras de manera eficiente.*

2. Fusionar el par de subpalabras más frecuentes en todo el corpus aun no fusionados:

- a. Encontrar el par de subunidades consecutivas con mayor frecuencia en todo el corpus y fusionarlas en una nueva unidad atómica.

Si el par más frecuente involucra caracteres que no son iniciales, se mantiene el prefijo ‘##’. Por ejemplo, si (“##a”, “##t”) es el par más frecuente, se fusionará como “##at”.

Si el par más frecuente incluye el primer carácter de la palabra, no se añade el prefijo ‘##’. Por ejemplo, al fusionar “g” con “##at”, se formaría “gat”.

- b. Añadir el nuevo par fusionado al vocabulario.

- c. Actualizar la segmentación de todas las palabras para reflejar las nuevas unidades atómicas. Por ejemplo, después de fusionar (“##a”, “##t”), todas las palabras del corpus que contengan este par consecutivo se actualizarán para reflejar la nueva unidad atómica. Por ejemplo, la segmentación de “gato” cambiaría de {“gato”: [“g”, “##a”, “##t”, “##o”]} a {“gato”: [“g”, “##at”, “##o”]}. De manera similar, la segmentación de “trato” pasaría de {“trato”: [“t”, “r”, “##a”, “##t”, “##o”]} a {“trato”: [“t”, “r”, “##at”, “##o”]}. Este cambio también afectará a cualquier otra palabra que incluya el par fusionado. Por ejemplo, palabras como “patrón” o “batalla” se transformarían de {“patrón”: [“p”, “##a”, “##t”, “##r”, “##ó”, “##n”]} y {“batalla”: [“b”, “##a”, “##t”, “##a”, “##ll”, “##a”]} a {“patrón”: [“p”, “##at”, “##r”, “##ó”, “##n”]} y {“batalla”: [“b”, “##at”, “##a”, “##ll”, “##a”]}. Este proceso garantiza que las segmentaciones de todas las palabras en el corpus reflejen las unidades atómicas dinámicas resultantes de las fusiones realizadas.

A partir de la primera fusión, las palabras ya no estarán descompuestas exclusivamente en caracteres individuales, sino en unidades atómicas dinámicas (subpalabras), que evolucionarán en función de su frecuencia en el corpus y las fusiones realizadas.

3. Repetir el proceso desde el paso 2:

- a. Recalcular las frecuencias de los pares de subpalabras dadas las nuevas segmentaciones.
- b. Repetir iterativamente el proceso de fusión hasta alcanzar el tamaño deseado del vocabulario o hasta que no haya pares de subpalabras por fusionar.

FASE 2: Tokenización de nuevas oraciones

El proceso de tokenización de nuevas oraciones se basa en dividir cada palabra en las subpalabras más largas posibles que estén presentes en el vocabulario entrenado.

1. Segmentación inicial: Se segmenta la oración utilizando un tokenizador basado en espacios, dividiendo la oración en palabras individuales.
2. Búsqueda del primer segmento válido: Para cada palabra, se busca el segmento más largo desde el principio de la palabra que coincida completamente con una entrada en el vocabulario. El tamaño del primer segmento debe ser mínimo de dos caracteres. Esto se realiza eliminando caracteres desde el final hasta encontrar una coincidencia válida en el vocabulario. Idealmente se busca la palabra completa.
3. Tokenización del resto de la palabra: Una vez identificado el primer segmento, se continúa buscando subpalabras dentro de la parte restante, pero esta vez de manera incremental desde el principio hacia el final de la porción restante, en lugar de buscar desde el final hacia el principio.
4. Caso especial, palabra desconocida: Si no se encuentra ninguna coincidencia en el vocabulario para un primer segmento, la palabra completa se representa con el token especial [UNK].

Por ejemplo, dado el texto "Hola mundo !" y un vocabulario entrenado que incluye {"Hola", "mundo", "mun", "do", "#un", "#do"}, el resultado sería ["Hola", "mun", "#do", "[UNK]"].

Tokenización BPE o basada en emparejamiento de pares

De manera similar a la tokenización por subpalabras, este método consta de dos fases principales con cierta similitud: (1) la construcción del vocabulario, hasta alcanzar un tamaño máximo, y (2) el propio método de tokenización en base a ese vocabulario.

FASE 1: Construcción del vocabulario dado un conjunto de oraciones de entrenamiento

1. Preprocesar el corpus e inicializar el vocabulario:
 - a. Pre-tokenizar el texto mediante una tokenización simple basada en espacios.
 - b. *Calcular la frecuencia de cada palabra en el corpus. Este cálculo será clave para determinar qué subpalabras fusionar de manera eficiente.*
 - c. Segmentar cada palabra en caracteres individuales y añadir esos caracteres al vocabulario inicial. Por ejemplo, para "gato" la segmentación sería ["g", "a", "t", "o"].
 - d. *Guardar la segmentación inicial de cada palabra. Por ejemplo, {"gato": ["g", "a", "t", "o"]}. Esto también nos ayudará a calcular posteriormente qué subunidades fusionar de manera eficiente.*
2. Fusionar el par de elementos más frecuentes en todo el corpus aun no fusionado:
 - a. Identificar el par de subunidades consecutivas más frecuente en todo el corpus y fusionarlas para crear una nueva unidad atómica. Por ejemplo, si el par ("a", "t") es el más frecuente, se fusiona en "at".
 - b. Añadir el nuevo par fusionado al vocabulario y crear una nueva regla de fusión (por ejemplo {("a", "t"): "at"}
 - c. Actualizar la segmentación de cada palabra con sus nuevas unidades atómicas. Se sustituyen las apariciones del par fusionado en todas las palabras por la nueva unidad atómica. Por

ejemplo, si se fusiona el par ("a", "t") en "at", la palabra "gato", que inicialmente estaba segmentada como ["g", "a", "t", "o"], se transforma en ["g", "at", "o"]. De forma similar, "trato", segmentada previamente como ["t", "r", "a", "t", "o"], pasa a ser ["t", "r", "at", "o"].

3. Repetir el proceso desde el paso 2:

- a. Recalcular las frecuencias de los pares de subunidades en las nuevas representaciones.
- b. Repetir iterativamente el proceso de fusión hasta alcanzar el tamaño deseado del vocabulario o hasta que no haya pares de subpalabras por fusionar.

FASE 2: Tokenización de nuevas oraciones

El proceso de tokenización de nuevas oraciones se basa en dividir cada palabra en pares posibles que estén presentes en el vocabulario entrenado.

Segmentación inicial: Se segmenta la oración utilizando un tokenizador basado en espacios, dividiendo la oración en palabras individuales.

División en caracteres: Cada una de esas palabras se divide, a su vez, en caracteres individuales.

Aplicación de reglas de fusión: Para cada regla de fusión aprendida (en el mismo orden en que fueron aprendidas), se verifica si puede ser aplicada a cada palabra de la oración, y si es el caso, se realiza la fusión. Este proceso se repite hasta que todas las reglas de fusión han sido visitadas.

Por ejemplo, dado el texto "Hola caracola" y las siguientes reglas de fusión, representadas en un diccionario:

```
{"("o", "l"): "ol",
("ol", "a"): "ola",
("a", "c"): "ac",
("H", "ola"): "Hola"}
```

El resultado después de aplicar la primera regla de fusión sería: ["H", "ol", "a"] y ["c", "a", "r", "a", "c", "ol", "a"]

Tras la segunda regla de fusión: ["H", "ola"] y ["c", "a", "r", "a", "c", "ola"]

Tras la tercera regla de fusión: ["H", "ola"] y ["c", "ac", "r", "a", "c", "ola"]

Tras la cuarta regla de fusión: ["Hola"] y ["c", "ac", "r", "a", "c", "ola"]

ANEXO II – Ejemplos de salidas estimadas sobre test_sentences.txt

Ejemplo de una posible salida para el método WordPiece con un tamaño de vocabulario de 150 aplicado a las oraciones del archivo test_sentences.txt (puede haber ligeras variaciones si ocurre un empate en la selección del par más frecuente a fusionar).

Input: 'El perro pequeño juega .' -> Tokens: ['El', 'perro', 'pequeño', 'juega', '[UNK]']

Input: 'El ratón pequeño corre rápido 🐭.' -> Tokens: ['El', 'ratón', 'pequeño', 'corr', '##e', '[UNK]', '[UNK]']

Input: 'El gato duerme tranquilo 🐱.' -> Tokens: ['El', 'gato', 'duerm', '##e', 't', '##ran', '##q', '##u', '##i', '##lo', '[UNK]']

Input: 'Una rata pequeña persigue un ratón 🐀.' -> Tokens: ['Una', 'rat', '##a', 'pequeñ', '##a', 'persigue', '[UNK]', 'ratón', '[UNK]']

Input: 'El perro persigue un gato grande.' -> Tokens: ['El', 'perro', 'persigue', '[UNK]', 'gato', 'grande', '##.']

Input: 'El juguete del gato cuesta \$15.' -> Tokens: ['El', 'ju', '##g', '##ue', '##te', 'd', '##e', '##l', 'gato', 'cuesta', '\$1', '##5', '##.']

Input: 'Los gatos se esconden bajo la mesa.' -> Tokens: ['Los', 'gato', '##s', 's', '##e', '[UNK]', '[UNK]', 'la', 'm', '##e', '##s', '##a', '##.']

Input: 'El queso cuesta \$5 por pieza.' -> Tokens: ['El', 'queso', 'cuesta', '\$', '##5', 'p', '##o', '##r', 'p', '##ie', '##z', '##a', '##.']

Input: 'El perro ladra al cartero en la puerta 🚪.' -> Tokens: ['El', 'perro', 'ladra', 'al', 'c', '##a', '##r', '##te', '##r', '##o', 'en', 'la', 'p', '##uer', '##ta', '[UNK]']

Input: 'El sol ilumina el parque y los árboles 🌳.' -> Tokens: ['El', 's', '##o', '##l', '[UNK]', 'el', 'parque', '[UNK]', 'l', '##os', '[UNK]', '[UNK]']

Input: 'Una bicicleta nueva cuesta \$200.' -> Tokens: ['Una', '[UNK]', '[UNK]', 'cuesta', '\$', '##2', '##0', '##0', '##.']

Input: 'El niño juega con su perro felizmente.' -> Tokens: ['El', 'n', '##i', '##ñ', '##o', 'juega', 'co', '##n', 's', '##u', 'perro', 'f', '##e', '##l', '##i', '##z', '##m', '##e', '##n', '##te', '##.']

Ejemplo de una posible salida para el método BPE con un tamaño de vocabulario de 150 aplicado a las oraciones del archivo test_sentences.txt (puede haber ligeras variaciones si ocurre un empate en la selección del par más frecuente a fusionar).

Input: 'El perro pequeño juega 🐶.' -> Tokens: ['El', 'perro', 'pequeño', 'juega', '##o', '##.']

Input: 'El ratón pequeño corre rápido 🐭.' -> Tokens: ['El', 'ratón', 'pequeño', 'corr', 'e', 'r', 'á', 'p', 'i', 'd', 'o', '##', '\u200d', '##', '##.']

Input: 'El gato duerme tranquilo 🐱.' -> Tokens: ['El', 'gato', 'duerm', 'e', 't', 'ran', 'q', 'u', 'i', 'l', 'o', '##', '##.']

Input: 'Una rata pequeña persigue un ratón 🐀.' -> Tokens: ['Una', 'rat', 'a', 'pequeñ', 'a', 'persigue', 'u', 'n', 'ratón', '##o', '##.']

Input: 'El perro persigue un gato grande.' -> Tokens: ['El', 'perro', 'persigue', 'u', 'n', 'gato', 'grand', 'e.']

Input: 'El juguete del gato cuesta \$15.' -> Tokens: ['El', 'ju', 'g', 'ue', 'te', 'd', 'el', 'gato', 'cuesta', '\$1', '5', '.']

Input: 'Los gatos se esconden bajo la mesa.' -> Tokens: ['Los', 'gat', 'os', 's', 'e', 'e', 's', 'co', 'n', 'd', 'en', 'b', 'a', 'j', 'o', 'la', 'm', 'e', 's', 'a', '.']

Input: 'El queso cuesta \$5 por pieza.' -> Tokens: ['El', 'queso', 'cuesta', '\$', '5', 'p', 'o', 'r', 'p', 'i', 'e', 'z', 'a', '.']

Input: 'El perro ladra al cartero en la puerta  .' -> Tokens: ['El', 'perro', 'ladra', 'al', 'c', 'a', 'r', 'te', 'r', 'o', 'en', 'la', 'p', 'uer', 'ta', ' ', , '.']

Input: 'El sol ilumina el parque y los árboles  .' -> Tokens: ['El', 's', 'o', 'l', 'i', 'l', 'u', 'm', 'i', 'n', 'a', 'el', 'parque', 'y', 'l', 'os', 'á', 'r', 'b', 'o', 'l', 'e', 's', ' ', , '.']

Input: 'Una bicicleta nueva cuesta \$200.' -> Tokens: ['Una', 'b', 'i', 'c', 'i', 'c', 'l', 'e', 'ta', 'n', 'ue', 'v', 'a', 'cuesta', '\$2', '0', '0', '.']

Input: 'El niño juega con su perro felizmente.' -> Tokens: ['El', 'niñ', 'o', 'juega', 'co', 'n', 's', 'u', 'perro', 'f', 'el', 'i', 'z', 'm', 'en', 't', 'e.', '']