Procesamiento de Lenguaje Natural - NLP

Casos de uso

Definición

El procesamiento del lenguaje natural (NLP, por sus siglas en inglés) se refiere a la rama de la inteligencia artificial o IA, que se ocupa de brindar a las computadoras la capacidad de comprender textos y palabras habladas, de la misma manera que lo hacen los seres humanos.

Retos del NLP

Amplia variedad de lenguajes:

- Existen muchos lenguajes
- Gran cantidad de reglas gramaticales
- Variaciones regionales de un mismo lenguaje

Retos del NLP

Ambigüedad: El lenguaje natural es ambiguo. Aquí algunas frases curiosas:

- Vi las montañas volando hacia Mendoza.
- Después de la muerte, los mineros se niegan a trabajar.
- En México, una mujer da a luz cada 15 minutos.
- El oficial disparó al hombre con la navaja.

Retos del NLP

Sinonimia: Expresar la misma idea con diferentes términos. Por ejemplo:

- Los términos «alto» y «grande» pueden ser sinónimos para describir un objeto o edificio, pero no son intercambiables en todos los contextos: «grande» puede significar de mayor edad.

Retos del NLP

Correferencia: Encontrar todas las expresiones que hacen referencia a una misma entidad.

- Se aplica, por ejemplo, en la comprensión de texto para resumir documentos.

Retos del NLP

Estilo de escritura: Dependiendo de la personalidad, las intenciones y las emociones del autor, la misma idea se puede expresar de diferentes maneras.

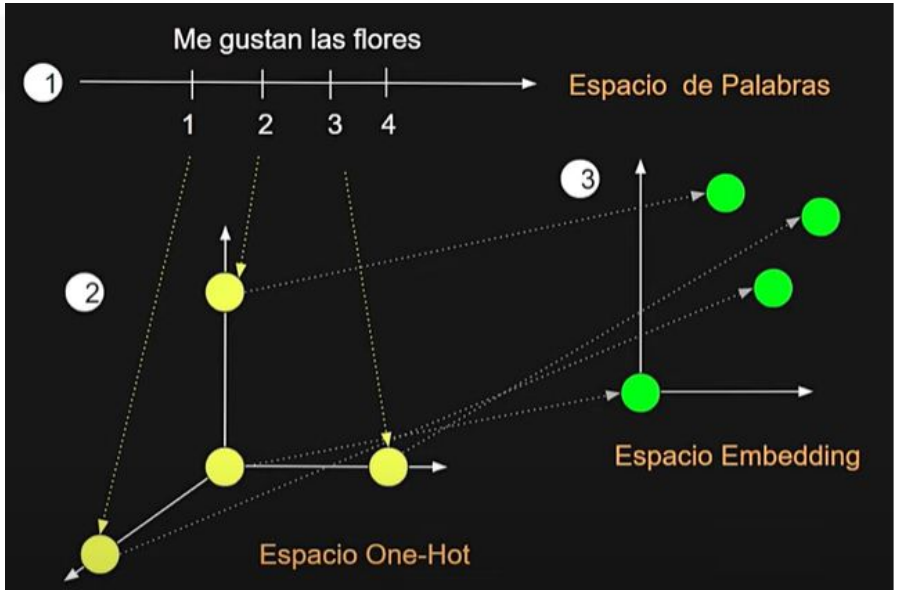
- Ironía: ¡Claro que me encanta estar acá trabajando de 8 a 22!
- Ironía: Gracias por la multa, es justo lo que necesitaba
- Ironía: Seguí durmiendo, que tu cuarto se ordena solo
- Sarcasmo: Encuentro a la televisión muy educativa.
- Sarcasmo: Es todo un personaje

Enfoques en NLP

- Métodos basados en reglas:
  - Creación de reglas específicas para un área de conocimiento. Se usan **expresiones regulares**.
- Métodos estadísticos:
  - Desarrollo de modelos predictivos sobre grandes cantidades de textos llamados corpus. Se usan como ejemplos de entrada a algoritmos de machine learning y las técnicas clásicas abarcan modelos como Naive Bayes,regresión logística, entre otros.
- Modelos de deep learning:
  - Uso de redes neuronales (recurrentes). Las capas de neuronas pueden verse como extractores automáticos de características. Se usan **word embeddings**, en la que las palabras son representadas como vectores de números reales.

Redes neuronales de embeddings

Una red neuronal simplificada que mapea una variable discreta (categórica) con índices enteros (que representan palabras), a un vector de números continuos llamados embeddings.



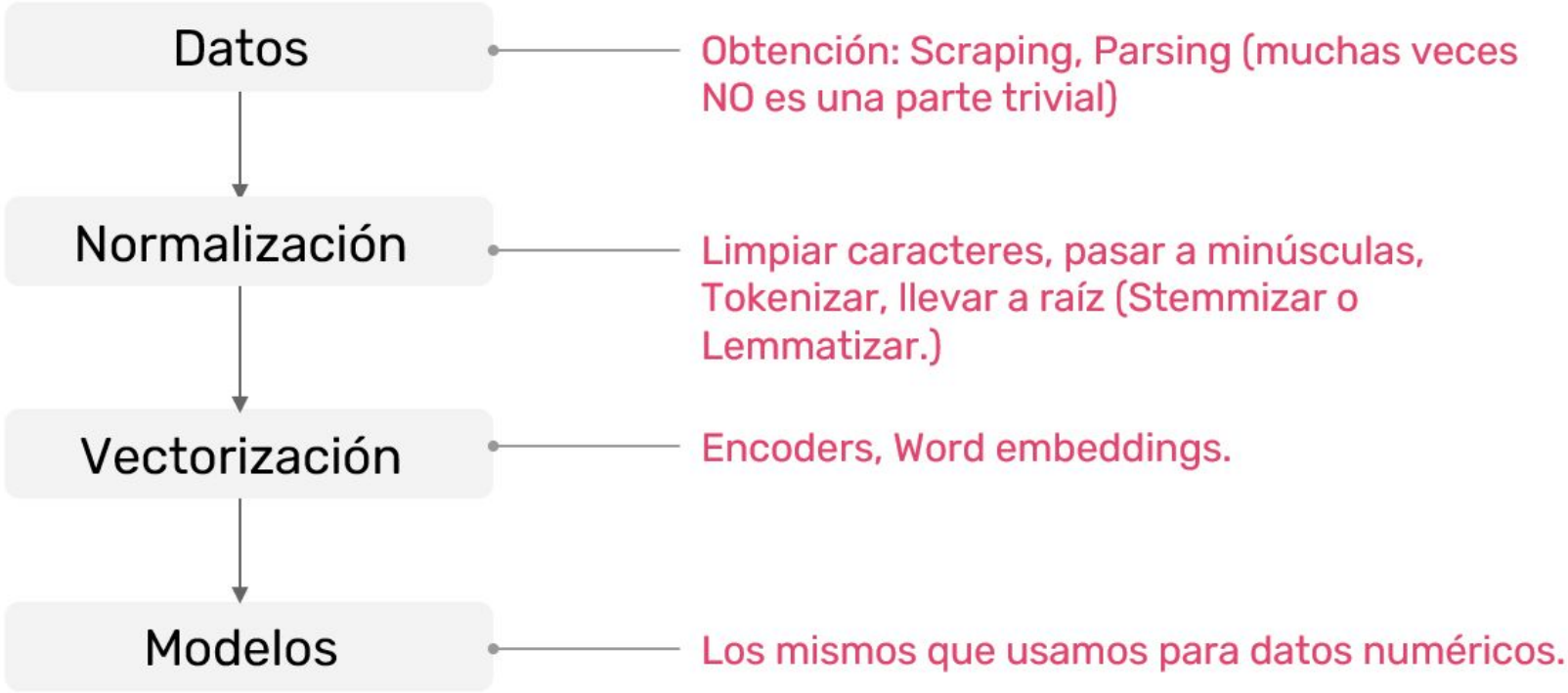
Aplicaciones comerciales

- **Análisis de sentimiento**. Identificación del estado de ánimo u opiniones subjetivas en grandes cantidades de texto, incluyendo minería de sentimiento y opiniones promedio.
- **Chatbots y asistentes virtuales**: Los usuarios pueden tener conversaciones con el sistema. También pueden guiar a los usuarios por flujos de trabajo complicados o ayudarlos a navegar por un sitio o una solución.
- **Búsqueda semántica**: Se usa en comercio electrónico para generar recomendaciones de productos. Decodifica el contexto de las palabras clave analizando motores de búsqueda y usando búsqueda basada en el conocimiento. Interpreta la intención del usuario para proporcionar recomendaciones más relevantes.

Aplicaciones comerciales

- **Reconocimiento de entidad nombrada (NER)**: implica la identificación y clasificación de entidades nombradas en texto no estructurado, como personas, organizaciones, ubicaciones, fechas y otra información relevante. NER se utiliza en varias aplicaciones de PNL, como extracción de información, análisis de sentimientos, respuesta a preguntas y sistemas de recomendación.
- **Resumen de texto**: resumir rápidamente grandes documentos en un texto. La industria de las finanzas aprovecha esto para analizar las noticias y las redes sociales a fin de ayudar a predecir las tendencias del mercado. En el gobierno y el sector jurídico, se usa para extraer información clave de los documentos.

NLP - Flujo de trabajo

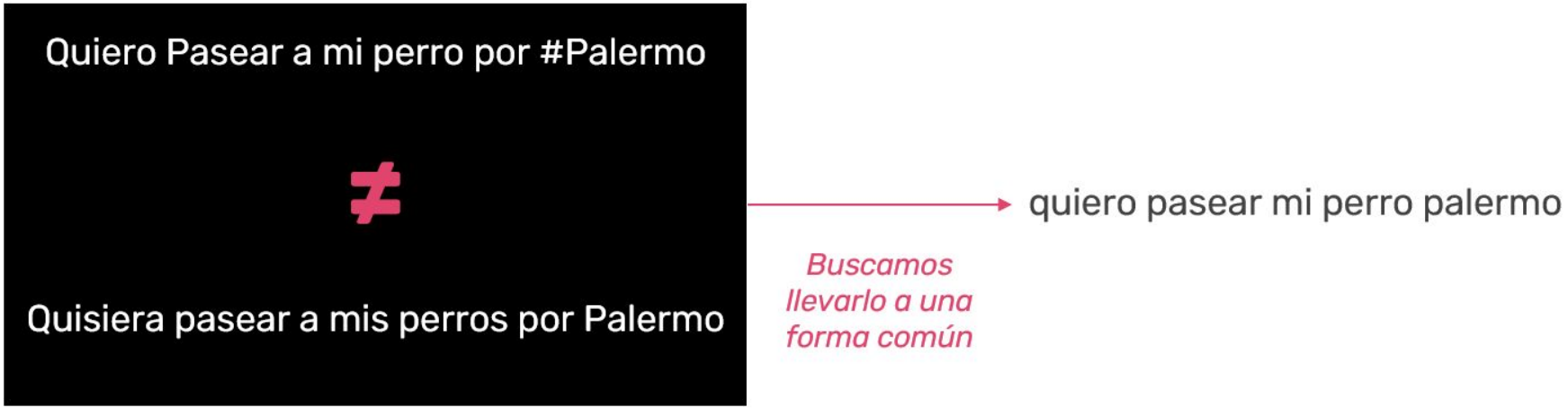


NLP - Flujo de trabajo

Para esta vamos a usar la librería NLTK de Python

Normalización (idea inicial)

llevar todo el texto a un formato común, donde palabras escrita de manera distinta o con significados similares, se representen de la misma manera.



Normalización

Técnicas que se utilizan para estandarizar o simplificar el texto de manera que el análisis y la comparación de los datos sean más efectivos:

- Pasar a minúsculas
- Tokenizar
- Limpiar caracteres
- Eliminar palabras no significativas
- Llevar a raíz

Normalización

- **Pasar a minúsculas:** pasar todas los caracteres de un texto a su forma minúscula para homogeneizar.

"Esto es un texto. Tiene varias oraciones. Todas son distintas, ninguna es igual."

texto.lower()

"esto es un texto. tiene varias oraciones. todas son distintas, ninguna es igual."

Normalización

- **Tokenizar oraciones:** pasar de un único string de texto a una lista de strings de oraciones.

"esto es un texto. tiene varias oraciones. todas son distintas, ninguna es igual."

["esto es un texto.", "tiene varias oraciones.", "todas son distintas, ninguna es igual."]

```
nltk.tokenize.sent_tokenize(texto)
```

Normalización

- **Tokenizar palabras:** pasar de un único string de una oración a una lista de strings de Tokens (palabras, puntuaciones, símbolos).

"esto es un #hastag." → ["esto", "es", "un", "#", "hashtag", "."]

nltk.tokenize.word_tokenize(texto)

Normalización

- **Limpiar caracteres:** nos quedamos sólo con los caracteres de interés. Esto dependerá de nuestro problema en particular. En nuestro caso vamos a utilizar la librería 're', que nos permite modificar texto.


```
import re
re.sub("[^a-zA-Z\s]", "", str(texto))
```

Normalización

- **Llevar a raíz:** buscamos llevar palabras distintas con significados similares a una forma común.

  - **Opción 1: Stemmizer:** Logra esto recortando las palabras mediante un proceso heurístico. Es rápido y fácil de usar, pero a veces no es certero.

["starting", "wants", "repartitions", "america's"] → ["start", "want", "repar", "america"]

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
stemmer.stem(palabra)
```

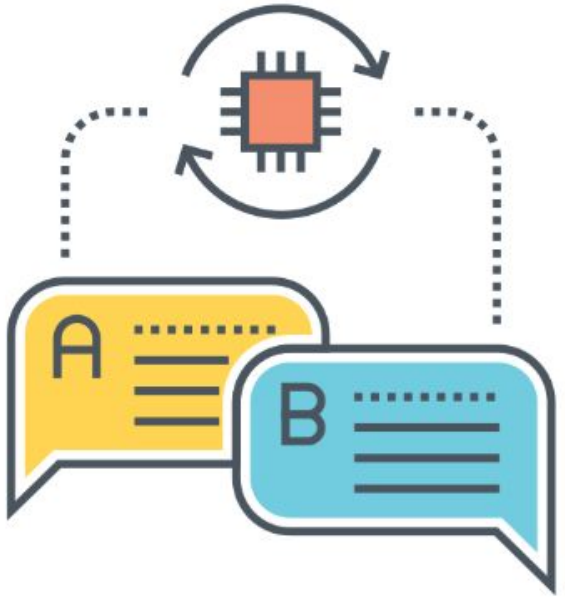
Normalización

Stemming (derivación): es el proceso de eliminar sufijos o prefijos de una palabra para obtener su forma raíz o stem.

- El objetivo principal del stemming es reducir las palabras relacionadas a una forma común.
- Por lo general, las formas resultantes pueden no ser palabras reales, pero son útiles para agrupar palabras similares.

Ejemplo:

- Palabra original: "Correr"
- Forma derivada: "Corr"



Normalización

Stemmizer:

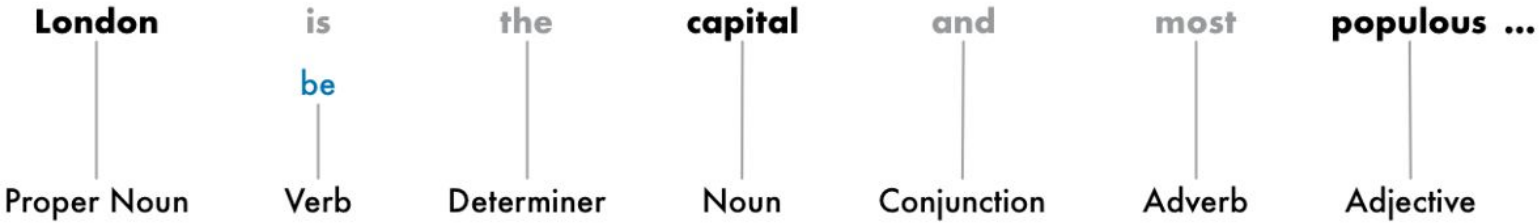
Ejemplos de raíz y desinencia en los verbos regulares

1. **Cantar**. Raíz: ***cant-*** / Desinencias: *cant**aron**, cant**aremos**, cant**é**.*
2. **Beber**. Raíz: ***beb-*** / Desinencias: *beb**ió**, beb**emos**, beb**erás**.*
3. **Amamantar**. Raíz: ***amamant-*** / Desinencias: *amamant**aron**, amamant**ó**, amamant**aríamos**, amamant**arías**.*
4. **Llorar**. Raíz: ***llor-*** / Desinencias: *llor**ó**, llor**aré**, llor**aron**, llor**arán**, han llor**ado**.*
5. **Amanece**. Raíz: ***amanec-*** / Desinencias: *amanec**ió**, amanec**erá**, amanec**ían**, han amanec**ido**.*
6. **Acompañar**. Raíz: ***acompañ-*** / Desinencias: *acompañ**aría**, acompañ**aré**, acompañ**aron**, acompañ**arían**.*
7. **Partir**. Raíz: ***part-*** / Desinencias: *part**ió**, part**iré**, part**irían**, part**irán**.*
8. **Sufrir**. Raíz: ***sufr-*** / Desinencias: *sufr**ió**, sufr**irán**, sufr**iríamos**, sufr**ieron**.*
9. **Doler**. Raíz: ***dol-*** / Desinencias: *dol**ió**, dol**ido**, dol**erá**, dol**ería**, habían dol**ido**.*
10. **Calmar**. Raíz: ***calm-*** / Desinencias: *calm**a**, calm**arían**, calm**aron**, calm**arán**.*
11. **Jugar**. Raíz: ***jug-*** / Desinencias: *jug**aría**, jug**aré**, jug**ó**, jug**aríamos**, hubiesen jug**ado**.*
12. **Cantar**. Raíz: ***cant-*** / Desinencias: *cant**o**, cant**ando**, cant**aré**, cant**aríamos**, habíamos cant**ado**.*

Normalización

- **Llevar a raíz:** buscamos llevar palabras distintas con significados similares a una forma común.

- **Opción 2: Lemmatizer:** Logra esto utilizando un vocabulario y realizando un análisis morfológico de las palabras. Precisa que además de la palabra se le informe cual es la función de la palabra en el texto

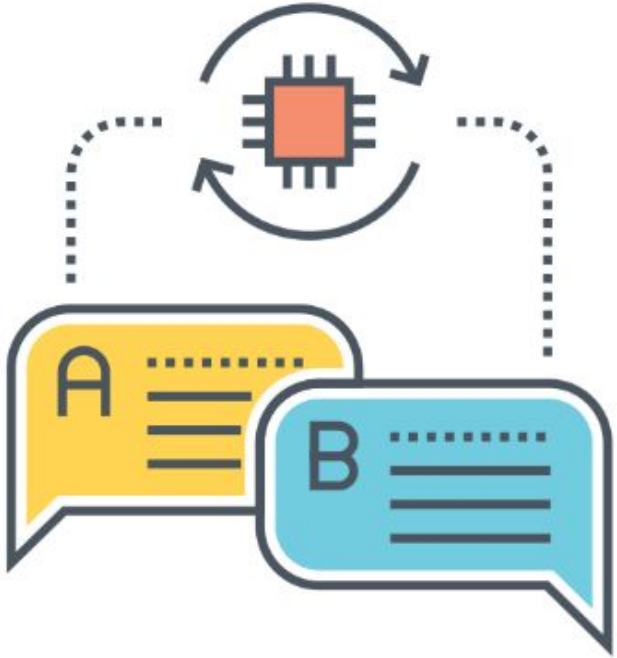


Normalización

Lematización: La lematización es un proceso más sofisticado que busca reducir las palabras a su forma base, pero asegurándose de que la forma base resultante sea una palabra real que exista en el idioma. Para lograr esto, la lematización utiliza un diccionario o un conjunto de reglas gramaticales para realizar la reducción.

Ejemplo

- Palabra original: "Corriendo"
- Forma base (lemma): "Correr"

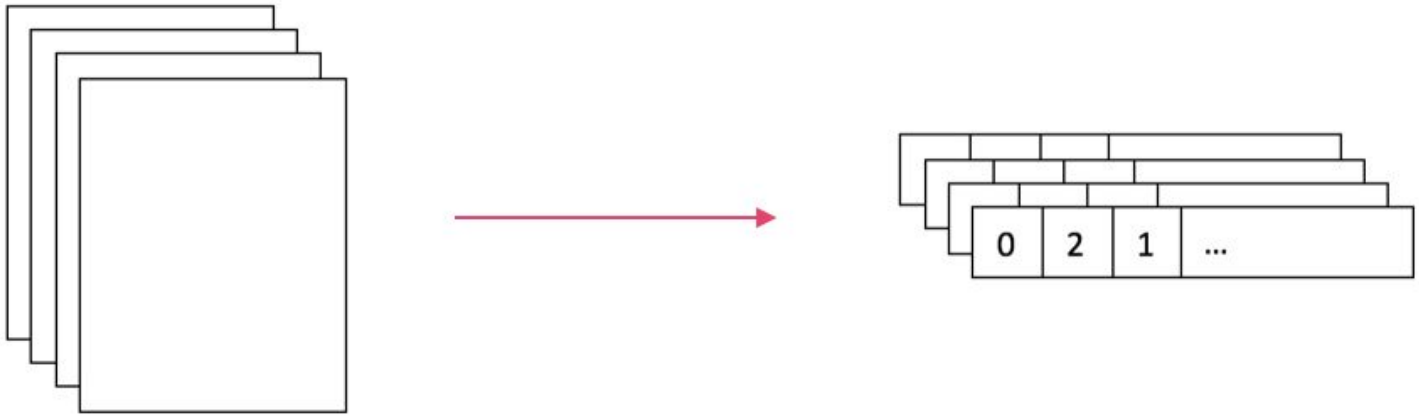


Normalización


```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
wordnet_lemmatizer.lemmatize(palabra,
get_wordnet_pos(palabra)
```

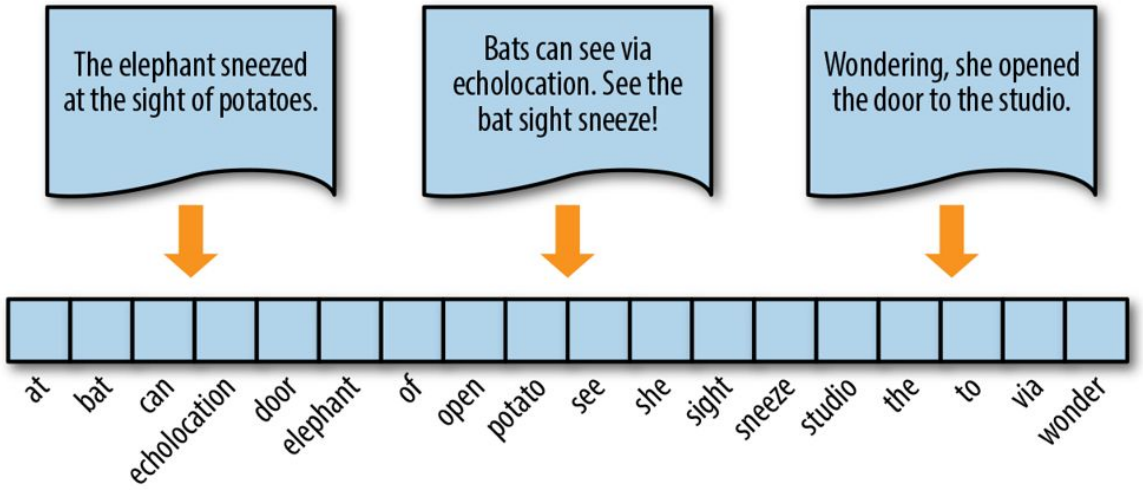
Vectorización

Objetivo: Representar cada texto (instancia de la base de datos) como un vector que podamos usar como vector de features para entrenar una de los modelos



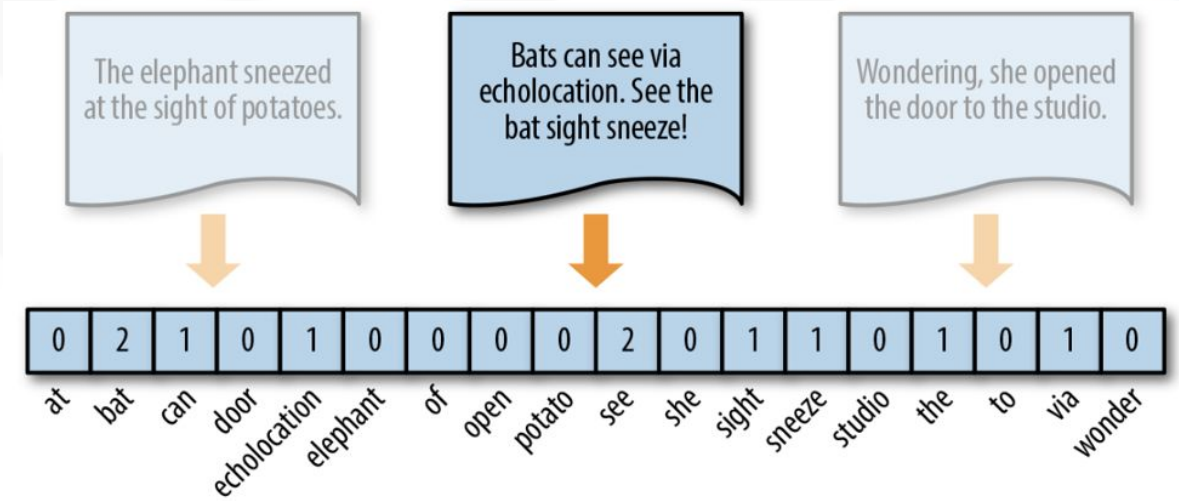
Vectorización - Bag of Words

Idea: Generar un vector que represente todas las palabras del corpus. Representar cada instancia como un vector con la cantidad de veces que aparecen las palabras.



Vectorización - Bag of Words

Idea: Generar un vector que represente todas las palabras del corpus. Representar cada instancia como un vector con la cantidad de veces que aparecen las palabras.

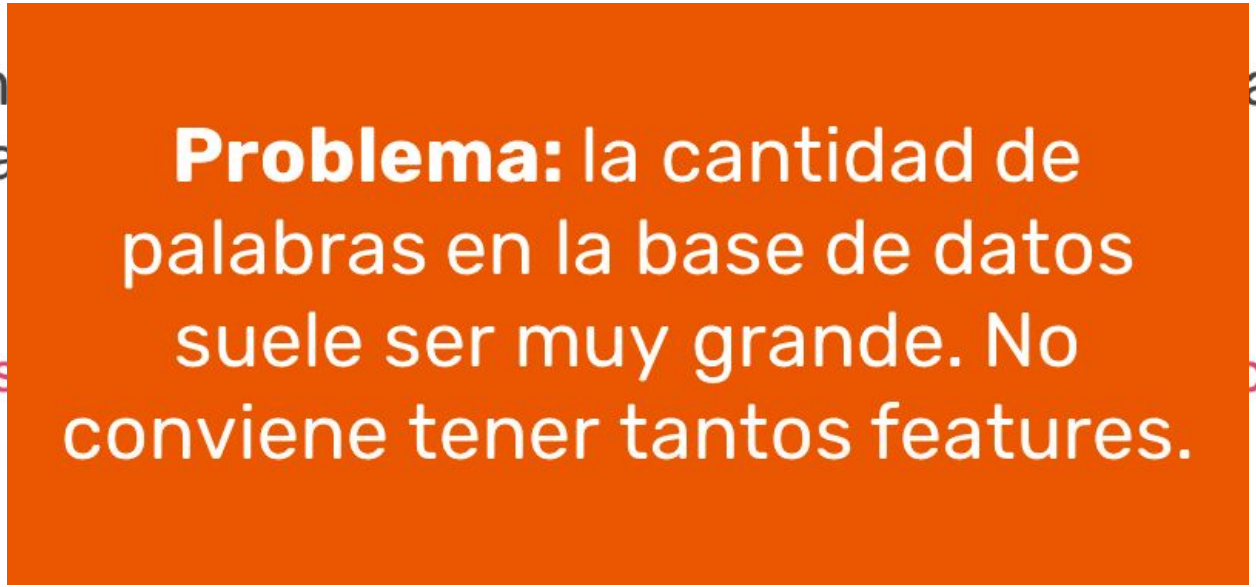


Vectorización - Bag of Words

Para implementarlo utilizamos una función de sklearn llamada CountVectorizer:

```
from sklearn.feature_extraction.text import CountVectorizer
```

Vectorización - Bag of Words



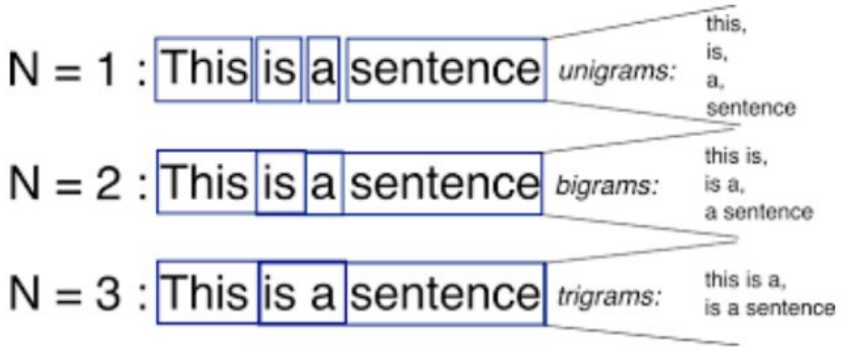
Vectorización - Bag of Words con N-gramas

Problema: hay palabras que cobran sentido cuando se las agrupa con otras, ejemplos: “Plaza Italia” y “Control Remoto” .

Vectorización - Bag of Words con N-gramas

N-Gramas

subsecuencia de n elementos consecutivos en una secuencia dada.



Vectorización - Bag of Words con N-gramas

Solución

Además de cada palabra por separado, agregamos los grupos de 2 (ó N) palabras contiguas a nuestro vector de Features.

Para implementar esto usando CountVectorizer:

```
CountVectorizer(analyzer='word', stop_words="english" , ngram_range=(1, 2))
```

Ojo con la cantidad de Features

Vectorización - TF-IDF

Observación: si buscamos diferenciar cada documento por las palabras que lo componen, las palabras que están en todos ellos no aportan información.

Idea: hay que medir no sólo cuanto aparece una palabra en una instancia (documento), sino también qué tan frecuente es esa palabra en todo el corpus.

Vectorización - TF-IDF

Term Frequency

Frecuencia de una palabra (***term***) en una instancia o documento (***doc***).

$$\text{TF}(term, doc) = \frac{\#\text{ de veces que el } term \text{ aparece en el } doc}{\#\text{ de } terms \text{ diferentes en el } doc}$$

$$\underbrace{\text{Hello,}}_{0.125}\ \text{my}\ \underbrace{\text{name}}_{0.125}\ \text{is}\ \underbrace{\text{Brandon.}}_{0.375}\ \text{Brandon Brandon.}\ \text{The}\ \underbrace{\text{elephant}}_{0.125}\ \underbrace{\text{jumps}}_{0.125}\ \text{over the}\ \underbrace{\text{moon.}}_{0.125}$$

Vectorización - TF-IDF

Document Frequency

Fracción de todos los documentos en nuestro corpus que contienen el término.

$$\mathbf{DF}(term, corpus) = \frac{\#\text{ de } docs \text{ que contienen } term}{\#\text{ total de } docs}$$

Vectorización - TF-IDF

Inverse Document Frequency

Logaritmo inversa de DF.

$$\mathbf{DF}(\textit{term},\textit{corpus}) = \text{Log}\left(\frac{\#\text{ total de } \textit{docs}}{\#\text{ de } \textit{docs} \text{ que contienen } \textit{term}}\right)$$

Ejemplo: si está en todos los docs log(N/N) = log(1) = 0

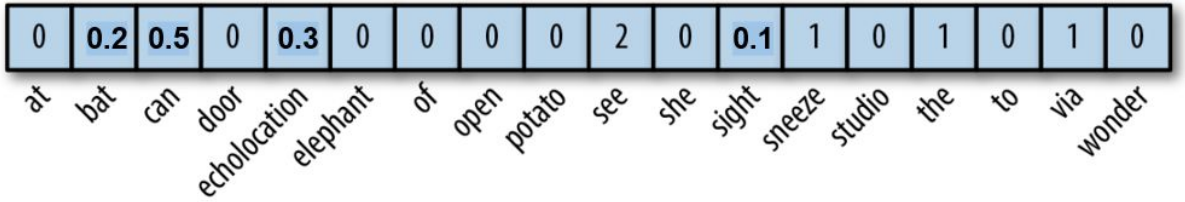
Vectorización - TF-IDF

Inverse Document Frequency

Producto del valor de TF por el de IDF.

TF-IDF(*term,corpus,doc*) = **TF**(*term,doc*) X **IDF**(*term,corpus*)

Cada palabra tiene un valor asociado en cada documento, con esto formamos nuestro vector (no necesariamente serán valores enteros):



Vectorización - TF-IDF

Considera un documento que contiene 100 palabras en donde la palabra “teléfono” aparece 5 veces.

El término frecuencia (es decir, tf) para teléfono es entonces (5/100) = 0.05.

Ahora, supongamos que tenemos 10 millones de documentos y la palabra teléfono aparece en mil de estos. Luego, la frecuencia del documento inverso (es decir, IDF) se calcula como log (10,000,000 / 1,000) = 4.

Por lo tanto, el peso de Tf-IDF es el producto de estas cantidades: 0.05 * 4 = 0.20.

Tf-IDF puede implementarse desde sklearn.feature_extraction.text import TfidfVectorizer

Preguntas…