Procesamiento de Lenguaje Natural

Procesamiento Básico

Mauricio Toledo-Acosta
mauricio.toledo@unison.mx

Departamento de Matemáticas
Universidad de Sonora

Section 1

Expresiones regulares

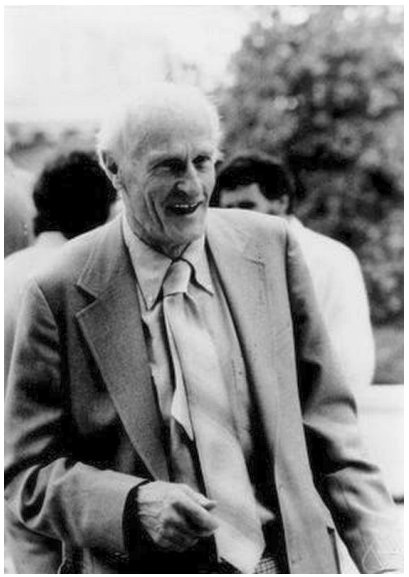
Expresiones regulares

Una **expresión regular** (regex, expresión racional) es una secuencia de caracteres que especifica un patrón de coincidencia en un texto. Los algoritmos de búsqueda de cadenas suelen utilizar este tipo de patrones para realizar operaciones de "búsqueda" o "búsqueda y sustitución" de cadenas, o para validar entradas.

Las expresiones regulares constan de constantes (denotan conjuntos de cadenas) y símbolos de operaciones (denotan operaciones sobre estos conjuntos).

Un poco de historia

Se originaron en 1951, por Stephen Cole Kleene. Usualmente se usa el standard `IEEE POSIX`. Kleene es uno de los fundadores de las ciencias computacionales teóricas.



Utilidad de las expresiones regulares

- Validación de datos.
- Búsqueda, extracción y reemplazo de texto.
- División de Texto
- Transformación de Texto.
- Tareas de PLN (eliminación de stopwords).

Referencias adicionales

- `https://regex101.com/`
- Tutorial 1.
- Tutorial 2.
- Tutorial 3.

Tutorial: Metacaracteres

Los metacaracteres son caracteres que un motor RegEx interpreta de forma especial

```
[] . ^ $ * + ? {} () \ |
```

[ ]	Cualquier caracter dentro de los corchetes
.	Cualquier caracter (excepto cambios de línea)
^	Buscar si el caracter siguiente está al inicio de una línea
[^]	Negación de cualquier caracter dentro de los corchetes
$	Buscar si el caracter anterior está al final de una línea

Si queremos buscar los metacaracteres como caracteres se anteceden de un \.

Tutorial: Metacaracteres

[ ] . ^ $ * + ? { } ( ) \ |

*	Busca si el caracter anterior ocurre 0 o más veces
+	Busca si el caracter anterior ocurre 1 o más veces
?	Busca si el caracter anterior ocurre 0 o 1 vez
$\{m, n\}$	Busca si el caracter anterior ocurre al menos m veces y máximo n veces
|	*OR*, busca el caracter antes o después del |
()	Agrupar patrones (expresiones).
\1	*Backreference*, captura el patrón anterior repetido consecutivamente.

Agrupación de patrones

Los paréntesis **()** *capturan* partes del patrón para poder acceder a ellas por separado.

```python
import re

texto = "La fecha es 25/12/2023"
patron = r'\d{2}/\d{2}/\d{4}'  # SIN paréntesis
match = re.search(patron, texto)

print(match.group(0))  # "25/12/2023"
```

```python
import re

texto = "La fecha es 25/12/2023"
patron = r'(\d{2})/(\d{2})/(\d{4})'  # CON paréntesis
match = re.search(patron, texto)

print(match.group(0))  # "25/12/2023" (lo mismo)
print(match.group(1))  # "25" (ahora puedo extraer el día)
print(match.group(2))  # "12" (ahora puedo extraer el mes)
print(match.group(3))  # "2023" (ahora puedo extraer el año)
```

Tutorial: Sequencias especiales

\A	Inicio de la string
\b	Frontera de palabra
\w	Cualquier *word character*, es equivalente a `[a-zA-Z0-9_]`
\W	Cualquier *non word character*
\d	Cualquier dígito
\D	Cualquier no dígito

Implementación en Python

El módulo `re` de Python proporciona soporte completo para expresiones regulares:

Función	**Descripción**
`search()`	Busca la primera coincidencia en cualquier parte de la cadena
`match()`	Busca coincidencia solo al inicio de la cadena
`findall()`	Devuelve lista con todas las coincidencias
`sub()`	Sustituye coincidencias por otro texto

Limitaciones y alternativas

Cuándo NO usar regex:

- Parsing de estructuras anidadas (HTML, XML, JSON)
- Validación compleja de sintaxis
- Cuando el rendimiento es crítico en textos grandes
- Patrones que cambian frecuentemente

Alternativas especializadas:

Tarea	**Herramienta**
HTML/XML	BeautifulSoup, lxml
JSON	`json` module
CSV	pandas, csv module
URLs	urllib.parse
Fechas	dateutil, datetime
Emails	email.utils

Ejemplo: Cuándo evitar regex

Incorrecto - Parsing HTML con regex: Frágil y propenso a errores

```
html = '<div class="data">Valor</div>'
regex_pattern = r'<div class="data">(.*?)</div>'
```

Correcto - Usando herramientas apropiadas:

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'html.parser')
valor = soup.find('div', class_='data').text
```

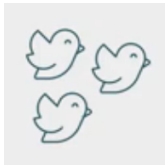
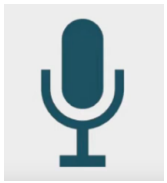
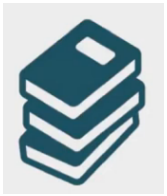
Principio clave: Las regex son excelentes para patrones de texto plano, pero no para estructuras complejas que requieren contexto semántico.

Section 2

Procesamiento básico de texto

Corpus

Un corpus es una colección de textos que se utilizará para alguna tarea de NLP.

Algunos ejemplos de corpus:

- 20newsgroups
- IMDB
- Project Gutenberg
- OntoNotes 5
- Penn Treebank

Técnicas de preprocesamiento

En cualquier aplicación de NLP el preprocesamiento de texto es el primer paso para cualquier técnica de modelado.

- Tokenización
- Lematización
- Stop words removal
- Etiquetado POS
- Etiquetado NER
- Análisis de dependecias

El Objetivo del Preprocesamiento

- **Reducción de ruido**: Eliminar información irrelevante que no ayuda al modelo (stop words, puntuación).
- **Normalización**: Reducir la dimensionalidad. Corro, corres, corriendo son la misma idea (correr). Hace que los modelos sean más eficientes y robustos.
- **Estructuración**: Convertir texto no estructurado en datos estructurados (tokens, etiquetas) que un algoritmo de Machine Learning pueda entender.
- **Enriquecimiento**: Añadir información lingüística (como las etiquetas POS) que ayuda a los modelos a entender mejor el contexto.

Subsection 1

Stopwords

Stop Words

Las **stop words** son palabras muy comunes en un idioma que, por sí solas, aportan poco significado léxico o temático a un texto.

Ejemplos Típicos en Español

Artículos (*el*, *la*, *los*, *las*), preposiciones (*de*, *en*, *por*, *para*), conjunciones (*y*, *o*, *pero*, *porque*), pronombres (*yo*, *tú*, *él*, *ello*, *me*, *te*, *se*) y algunos verbos auxiliares (*es*, *ha*, *tener*).

Stop Words: Cuándo **no** eliminarlas

Eliminar stop words es una operación agresiva.

- **Negación**: La palabra "no" es a menudo considerada una stop word. Eliminarla invierte el significado.

  "el producto **no** es bueno" $\rightarrow$ "producto bueno"

- **Expresiones Idiomáticas y Frases Hechas**: "poco **a** poco", "cara **a** cara", "**de** acuerdo".
- **Lenguaje Formal y Específico de Dominio**: En textos legales, "**por tanto**", "**en** virtud **de**" son cruciales para la estructura lógica.
- **Análisis Sintáctico o de Estilo**: Son importantes al estudiar la estructura de la lengua o la autoría.

Stop Words

Conclusión

- **No** es un paso obligatorio en todo pipeline de NLP.
- **Sí** es un hiperparámetro que debe probarse. Entrena modelos **con** y **sin** stop words y compara.
- **Personaliza tu lista**: Las listas predefinidas (NLTK, spaCy) son un buen punto de partida, pero revísalas y adáptalas. ¿"No" está en la lista? ¿Quieres eliminar verbos como "ser" y "haber"?

Subsection 2

Tokenización

Tokenización (Tokenization)

La **tokenización** es el proceso de dividir un texto en unidades más pequeñas llamadas tokens. Estos tokens pueden ser palabras, caracteres, símbolos o frases. La tokenización es un paso fundamental en el procesamiento del texto.

Dificultades en la tokenización

Finland's capital	→	Finland Finlands Finland's ?
what're, I'm, isn't	→	What are, I am, is not
Hewlett-Packard	→	Hewlett Packard ?
state-of-the-art	→	state of the art ?
Lowercase	→	lower-case lowercase lower case ?
San Francisco	→	one token or two?
m.p.h., PhD.	→	??

Dificultades en la tokenización

French

- ***L'ensemble*** → one token or two?
  - ***L*** ? ***L'*** ? ***Le*** ?
  - Want ***l'ensemble*** to match with ***un ensemble***

German noun compounds are not segmented

- ***Lebensversicherungsgesellschaftsangestellter***
- 'life insurance company employee'
- German information retrieval needs **compound splitter**

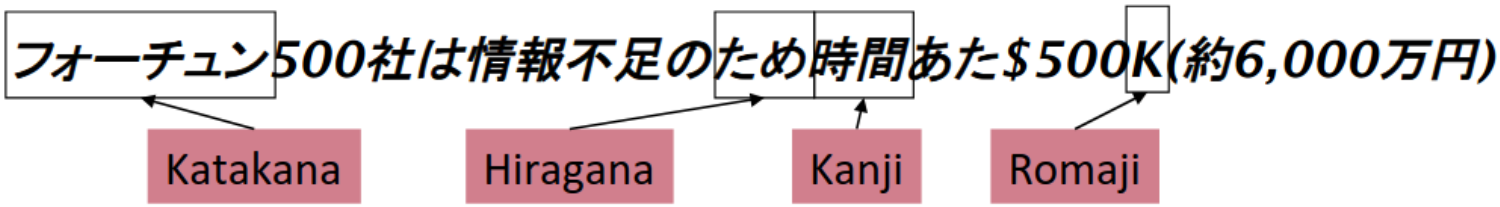
Dificultades en la tokenización

Chinese and Japanese no spaces between words:

- 莎拉波娃现在居住在美国东南部的佛罗里达。
- 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Sharapova now lives in US southeastern Florida

Further complicated in Japanese, with multiple alphabets intermingled

- Dates/amounts in multiple formats



Subsection 3

Lematización

Lematización y Stemming

La **lematización** es el proceso de reducir una palabra a su forma base (lema). Se utiliza para:

- Reducir la dimensionalidad del espacio de características, al mapear palabras relacionadas a un solo lema.
- Mejorar la precisión de los modelos de lenguaje, al tratar palabras con el mismo significado como una sola entidad.
- Facilitar la comparación y el análisis de textos, al estandarizar la forma de las palabras.

$$\text{Correr, corre, corriendo, corredor} \longrightarrow \texttt{correr}$$

$$\text{Feliz, felicidad, felices} \longrightarrow \texttt{feliz}$$

Stemming

El **stemming** es el proceso de reducir las palabras a su raíz o tronco, eliminando sufijos y prefijos. El objetivo es identificar la forma base de una palabra, independientemente de su conjugación, número o género. Se utiliza para:

- Reducir la dimensionalidad del espacio de características en tareas de clasificación de texto.
- Mejorar la eficiencia en la indexación de texto.
- Facilitar la búsqueda de información.

$$\text{Correr, corre, corriendo, corredor} \longrightarrow \texttt{corri}$$

El stemming puede ser más rápido, aunque menos preciso, que la lematización. Además de producir palabras posiblemente no validas.

Diferencias entre Stemming y Lematización

Difference between stemming Vs. lemmatization

Stemming	**Lemmatization**
Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.	**Lemmatization** considers the context and converts the word to its meaningful base form, which is called Lemma.
For instance, stemming the word '**Caring**' would return '**Car**'.	For instance, lemmatizing the word '**Caring**' would return '**Care**'.
Stemming is used in case of large dataset where performance is an issue.	Lemmatization is computationally expensive since it involves look-up tables and what not.

Diferencias entre Stemming y Lematización

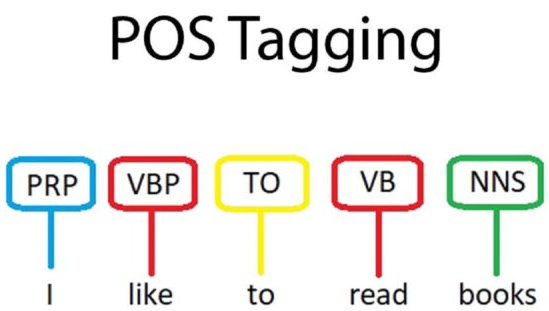
- **Stemming:**
  - "corriendo" → "corr"
  - "corrió" → "corr"
  - "gatos" → "gat"
- **Lematización:**
  - "corriendo" → "correr"
  - "corrió" → "correr"
  - "gatos" → "gato"

Subsection 4

POS tagging

Etiquetado POS

El **POS Tagging (Part-of-Speech Tagging)** es el proceso de identificar la categoría gramatical de cada palabra en un texto, como: Sustantivo (NOUN), verbo (VERB), adjetivo (ADJ), etc. El objetivo del POS Tagging es etiquetar cada palabra con su correspondiente categoría gramatical, lo que permite comprender mejor el significado y la estructura del texto.

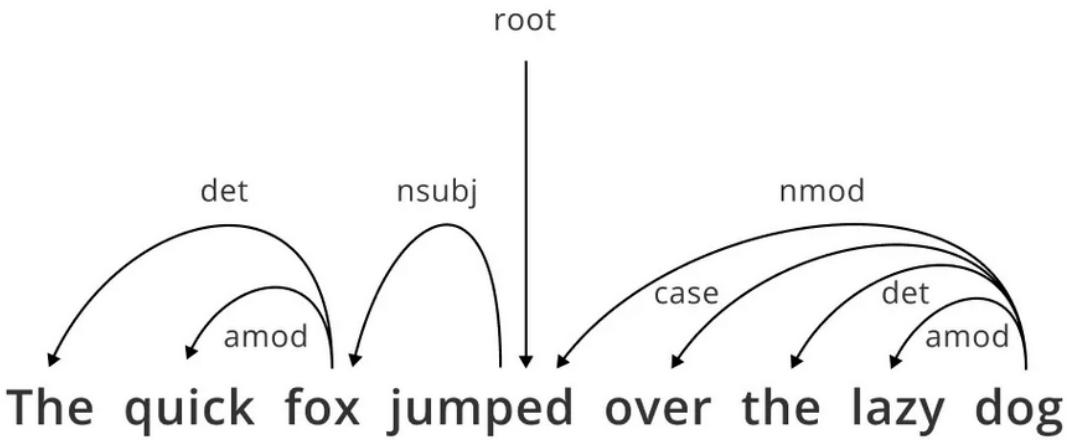


Subsection 5

Parsing

Parsing

El **parsing**, también conocido como análisis sintáctico, es el proceso de analizar una secuencia de tokens para determinar su estructura gramatical. En otras palabras, es el proceso de identificar las relaciones entre las palabras o símbolos en una secuencia para entender su significado.



Aplicaciones

- Extracción de Conocimiento e información
- Generación de Resúmenes con Coherencia
- Detección de Plagio
- Detección de Errores Gramaticales o Sintácticos

Subsection 6

Name Entity Recognition (NER)

Reconocimiento de Entidades Nombradas

NER (Named Entity Recognition) es el proceso de identificar y clasificar entidades nombradas en un texto en categorías predefinidas como:

- Nombres de personas (PER)
- Nombres de lugares (LOC)
- Nombres de organizaciones (ORG)
- Fechas (DATE)
- Monedas (MONEY)
- ...

El objetivo de NER es extraer información relevante de un texto y clasificarla en categorías significativas para su posterior análisis o procesamiento.

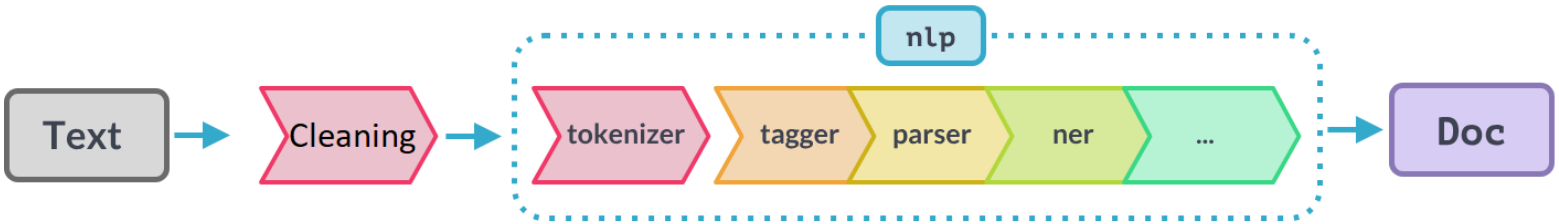
Reconocimiento de Entidades Nombradas

Ejemplo

Oración Original: "Dr. Smith, from Apple Inc., flew to Paris in January 2024 and bought 3 amazing books for $200. He's running late now, but he's feeling happier!"

Tokenización	['Dr.', 'Smith', ',', 'from', 'Apple', 'Inc.', ',', 'flew', 'to', 'Paris', 'in', 'January', '2024', 'and', 'bought', '3', 'amazing', 'books', 'for', '$', '200', '.', 'He', "'s", 'running', 'late', 'now', ',', 'but', 'he', "'s", 'feeling', 'happier', '!']
Stop Words	['Dr.', 'Smith', ',', 'Apple', 'Inc.', ',', 'flew', 'Paris', 'January', '2024', 'bought', '3', 'amazing', 'books', '$', '200', '.', "'s", 'running', 'late', ',', "'s", 'feeling', 'happier', '!']
Lematización	['Dr.', 'Smith', ',', 'from', 'Apple', 'Inc.', ',', 'fly', 'to', 'Paris', 'in', 'January', '2024', 'and', 'buy', '3', 'amazing', 'book', 'for', '$', '200', '.', 'He', 'be', 'run', 'late', 'now', ',', 'but', 'he', 'be', 'feel', 'happy', '!']
POS Tagging	[PROPN, PROPN, PUNCT, ADP, PROPN, PROPN, PUNCT, VERB, ADP, PROPN, ADP, PROPN, NUM, CCONJ, VERB, NUM, ADJ, NOUN, ADP, SYM, NUM, PUNCT, PRON, AUX, VERB, ADJ, ADV, PUNCT, CCONJ, PRON, AUX, VERB, ADJ, PUNCT]
NER	[Dr. Smith]_PER, from [Apple Inc.]_ORG, flew to [Paris]_GPE in [January 2024]_DATE and bought 3 amazing books for $200. He's running late now, but he's feeling happier!

NLP Pipeline



- NLTK - Natural Language Toolkit
- Spacy