

Diplomado de especialización de desarrollo de aplicaciones con Inteligencia Artificia

# Clase 01: Preprocesamiento de Textos

**CURSO:** 

Análisis de sentimiento en redes sociales.

PROFESOR:

MAG. ERASMO G. MONTOYA

# Contenido

- 1. Text Preprocessing
  - Text Preprocessing
  - Tokenization
  - Tokenization Issues
- 2. Activity

#### **Text Preprocessing**

#### **Premisa:**

Los textos (como cualquier otro dato) son adquiridos muy probablemente con "ruido" o "suciedad"

¿Qué es la "suciedad" en los textos?

#### Limpieza de:

- Signos de puntuación
- Símbolos
- Términos con caracteres no alfanuméricos
- Términos con caracteres no alfabéticos
- Palabras vacías (stopwords) y con longitud menor a 3

¿Qué debemos tener en cuenta para limpiar la información?

- ¿La fuente de información produce los textos de forma estandarizada siguiendo reglas de ortografía y gramática? (publicaciones académicas, documentos legales, artículos periodísticos, entre otros).
- ¿La fuente de información textual es sujeta a edición o revisión por expertos? (documentos de trabajo).
- ¿La fuente de información textual es **producida en masa** por cualquier persona sin ningún tipo de validación? (publicaciones en la web, redes sociales, correos electrónicos, chats).

¿Qué debemos tener en cuenta para limpiar la información?

#### **CONOCER NUESTROS DATOS**

¿De dónde provienen o quién los genera?

¿Cómo se almacenan? (encoding issues)

¿Por qué procesos/transformaciones pasan?

¿Qué debemos tener en cuenta para limpiar la información?

#### **CONOCER NUESTRO OBJETIVO DE ANÁLISIS**

¿Necesito toda la información?

¿Qué proceso de limpieza es el más adecuado?

#### ¿Qué debemos tener en cuenta para limpiar la información?

In practice, Mikolov et al. (2013b) obtained better results on the word translation task using a simple linear mapping, and did not observe any improvement when using more advanced strategies like multilayer neural networks. Xing et al. (2015) showed that these results are improved by enforcing an orthogonality constraint on W. In that case, the equation (I) boils down to the Procrustes problem, which advantageously offers a closed form solution obtained from the singular value decomposition (SVD) of  $YX^T$ :

$$W^* = \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\mathcal{F}} = UV^T, \text{ with } U\Sigma V^T = \text{SVD}(YX^T). \tag{2}$$



Keep Facebook from tracking you across the rest of the web: mzl.la/2GetoR1

#### Limpieza de:

- Signos de puntuación ¿No se va a dividir oraciones/frases para un análisis más granular?
- Símbolos ¿No hay elementos como #hashtags o @menciones?
- Términos con caracteres no alfanuméricos
- Términos con caracteres no alfabéticos
- Palabras vacías (stopwords) y con longitud menor a 3 ¿va a depender de la lengua procesada?

# Data cleaning: Python

```
Librerías útiles para limpieza:
string
regex (re)
nltk
spacy
```

# Data cleaning: Python

#### Ejemplos:

Remover puntuación: <a href="https://chrisalbon.com/machine\_learning/">https://chrisalbon.com/machine\_learning/</a>
<a href="preprocessing">preprocessing text/remove punctuation/</a>

Remover palabras vacías (u otro conjunto): <a href="https://chrisalbon.com/machine-learning/preprocessing-text/remove-stop-words/">https://chrisalbon.com/machine-learning/preprocessing-text/remove-stop-words/</a>

Reemplazar o limpiar caracteres o símbolos: <a href="https://chrisalbon.com/">https://chrisalbon.com/</a> machine learning/preprocessing text/replace characters/

# Preprocesamiento: Tokens y Vocabulario

```
text = "caballo caballo caballito
caballote"
tokens = text.split()
num_tokens = len(tokens)
vocabulary_size = len(list(set(tokens)))
from collections import Counter
c_tokens = Counter(tokens) #count per v. entry
```

# **Tokenization**

No solo se trata de "separar" **palabras** o **términos**, si no también se puede referir a nivel de **frases** y **oraciones**.

#vocab <= #tokens.</pre>

La separación implica **CONOCER** cómo se marca la **delimitación** de una unidad (palabra) en un lenguaje o contexto dado.

P.e. no es lo mismo tokenizar palabras en un idioma indoeuropeo (inglés) que en chino:

Data analytics = 數據分析

# **Tokenization**

**NLTK**: (Ejemplo para palabras y oraciones en inglés)

https://chrisalbon.com/machine\_learning/preprocessing\_text/tok
enize\_text/

### **Word Tokenization**

¿Por qué es un tópico específico?

¿No se refiere a sólo dividir las palabras/tokens en un texto según la posición de los **espacios en blanco** o **separadores**?

# **Tokenization: Issues**

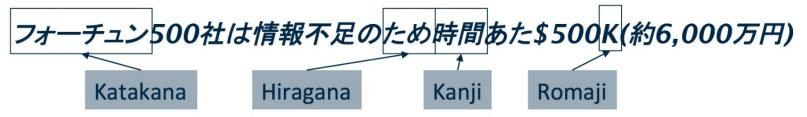
```
Finland's capital → Finland Finlands Finland's ?
what're, I'm, isn't → What are, I am, is not
Hewlett-Packard → Hewlett Packard ?
state-of-the-art → state of the art ?
Lowercase → lower-case lowercase lower case ?
San Francisco → one token or two?
m.p.h., PhD. → ??
```

# Tokenization: Language Issues

- 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

# **Tokenization: Language Issues Chinese -> Word Segmentation**

- 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



End-user can express query entirely in hiragana!

# Word Normalization - Estrategias

```
<u>Ejemplos (procurar normalizar):</u>
```

```
"U.S.A." vs "USA" -> "USA"

"espacio-temporal" vs "espacio temporal" vs

"espaciotemporal" -> "espacio temporal"

"Huamanga" vs "Guamanga" -> "Huamanga"

Ejemplos (procurar NO normalizar):

"CAT" (marca) vs "cat" (gato en inglés)

"río" (entidad geográfica) vs "rió" (reír en pasado, 3ra per.)
```

# **Word Normalization**

¿Y los herrores ortográficos?

Este paso también puede involucrar el uso de: **correctores ortográficos** (basados en reglas y/o corpus, y algoritmos de distancia de edición de textos)

# Morfología

La morfología es el estudio de la construcción de las palabras:

- Estructuras
- Variaciones
- Similitudes

El análisis morfológico permite descomponer una palabra y extraer:

- Raíz o lexema: Parte invariable de la palabra (raíz + morfema
   palabra)
- Lema: Unidad autónoma del léxico
- Categoría Gramatical (morfo-sintáctica)

# Morfología

#### Flexión:

```
Gat (o) (a) (os) (as)
Gat (it) (o) (a) (os) (as)
```

#### Composición:

```
agua + fiestas -> aguafiestas
```

#### **Derivación:**

```
Prefijos: telé- fono / megá- fono
```

Sufijos: trauma -tólogo / trauma -tología

Combinados: anti- constitucional

# Morfología: Lemmatization

Obtención de la forma canónica (lema) a partir de una palabra:

- Verbo: forma infinitiva.
  - Ejemplo: partirás -> partir
- Nombre, Adjetivo, Artículo, etc.: forma masculina singular.
  - Ejemplo: liso, lisa, lisos... -> liso

Se requiere un procesamiento lingüístico (herramienta o función ya implementada):

- Puede ser lento/tedioso
- Es dependiente de la lengua

# Morfología: Stemming

Obtención de una forma truncada común a todas las variables morfológicas:

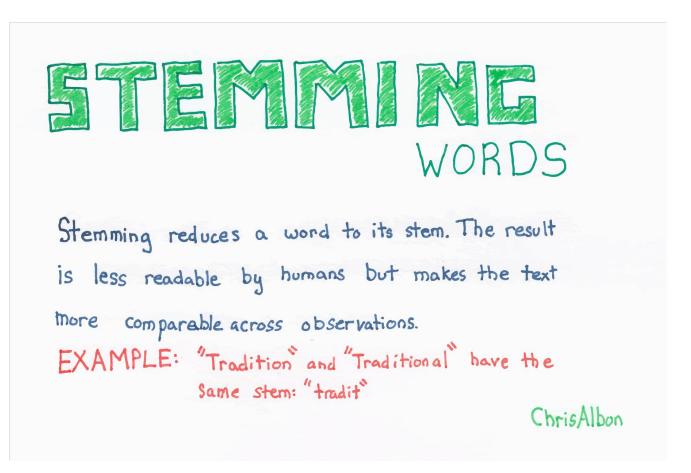
- Supresión de las flexiones
- Supresión de los sufijos
- Ejemplo: caballo, caballero, caballos, caballones, ... ->cabo (cabonis)

Stemming: se basa en reglas (ver algoritmo de Porter, en inglés)

- Es rápido
- Es dependiente de la lengua

"Aglutina" o agrupa muchos más términos que la lematización

# **Stemming Words**



https://chrisalbon.com/machine\_learning/preprocessing\_text/stemming\_words/

# Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a "."
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

# Meta-Información Lingüística

```
POS-tagging (Part-of-Speech: Categoría Gramatical)

Morphological Features (Accidentes morfológicos: género, número, tiempo, ...)
```

**Syntax Dependency** relationships (Relaciones sintácticas de dependencia entre términos en una oración: nsubj, dobj, iobj, ...)

**Python**: NLTK o SpaCy <a href="https://spacy.io/usage/linguistic-features">https://spacy.io/usage/linguistic-features</a>

#### **Descanso 15 min**

Recomendación musical: <u>Globuldub</u>



# **Actividad**

Buscar un libro/novela en inglés (libre) en la web y usarla para la siguiente tarea (también pueden usar sus datos de la TA si están en inglés):

- 1. Crear un archivo Jupyter Notebook que cargue el/los archivos
- Procesar la limpieza de los textos, con el objetivo de identificar: el tamaño de vocabulario y los términos relevantes más frecuentes en la lista de Tokens
- 3. Aplicar stemming (Porter) a sus textos, y volver a ejecutar el paso (2). Comparar tamaños.
- 4. Aplicar POS-tagging a sus textos (sin stemming), y obtener la distribución de los tipos de palabras en su colección. ¿Qué tipos son los más comunes?