

# Redes Neuronales

## Procesamiento de datos

Facultad de Ingeniería. Universidad del Valle

Octubre de 2022

# Contenido

## 1 Procesamiento de texto

- Introducción
- Limpieza del texto
- Vectorización

## 2 Procesamiento de Imágenes

- Introducción
- Procesado

# Contenido

## 1 Procesamiento de texto

- Introducción
- Limpieza del texto
- Vectorización

## 2 Procesamiento de Imágenes

- Introducción
- Procesado

# Procesamiento de texto

## Introducción

Para trabajar con textos en modelos de redes neuronales es necesario realizar los siguientes pasos:

- Limpiar el texto
- Convertir el texto a vectores numéricos

# Procesamiento de texto

## Limpieza de texto

Los textos deben limpiarse teniendo en cuenta

- Caracteres no ASCII
- Puntuaciones: signos de interrogación, tildes, apostrofes o comillas
- Normalizar
- Eliminar palabras comunes
- Stemming

# Procesamiento de texto

## Limpieza de texto

Inicialmente vamos a limpiar la puntuación:

```
import re
import string
#texto contiene el texto cargado
array_palabras = [d.split() for d in texto]
re_punct = re.compile('[%s]' % re.escape(string.punctuation))
palabras_limpias = [list(map(lambda w: re_punct.sub('', w),
                             p)) for p in array_palabras]
```

# Procesamiento de texto

## Limpieza de texto

Eliminamos los caracteres no alfanumericos

```
palabras_limpias = [list(map(lambda w: ''.join(filter(str.isalnum,w)),p)) for p in palabras_limpias]
```

# Procesamiento de texto

## Limpieza de texto

Vamos a convertir las palabras de minúsculas

```
palabras_limpias_min = [list(map(lambda w: ''.join(filter(
    str.isalnum,w)),p)) for p in palabras_limpias]
```



# Procesamiento de texto

## Eliminar palabras comunes

El proceso de eliminar palabras comunes consiste en eliminar palabras de uso común como: de, para, el, la, los, etc.

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopword_en = nltk.corpus.stopwords.words('english')
stopword_es = nltk.corpus.stopwords.words('spanish')
stopword = stopword_en + stopword_es
```

# Procesamiento de texto

## Eliminar palabras comunes

```
palabras_limpias_stop = [list(filter(lambda x: not(x in  
stopword), p)) for p in palabras_limpias_min]
```

# Procesamiento de texto

## Normalización

Existen palabras que significan lo mismo pero se escriben diferente, tales como caminar, camino, caminas, las podrían reducirse a una sola palabras que describa la acción que en este caso es caminar, este proceso se llama **stemming**, el cual consiste en convertir palabras en raíces.

```
import nltk
from nltk import SnowballStemmer
spanishstemmer=SnowballStemmer('spanish')
stems = [list(map(lambda x: spanishstemmer.stem(x),w)) for w
          in palabras_limpias_stop]
```

# Procesamiento de texto

## Normalización

Para poder entrenar un modelo de red neuronal es necesario convertir el texto en una bolsa de palabras (en inglés Bow-of-words model BoW). Los objetivos de este proceso son:

- Centrarse en la aparición de las palabras
- Transformar el texto en un vector numérico que lo describe

[ [Perro, gato, mamá ]  
 [gato, lobo, araña] ]

Perro gato mamá lobo araña  
 [ ]

[ "perro gato mamá" ] [ 1 1 1 0 0 ]  
 "gato lobo araña" [ 0 1 0 1 1 ]

Unigram

perro-perro  
 perro-gato  
 perro-mamá  
 ;

perro-araña  
 gato-perro  
 ;

araña-araña

[ 0 1 ... ]

# Procesamiento de texto

## Countvectorizer

Este modelo permite generar un vector numerico a partir del conteo del número de veces que aparece una palabra en los datos de entrada. Estos vectores tendrán muchos ceros, por ello usaremos vectores dispersos para representarlos.

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
#Casos de n_gramas ngram_range
texto_pro = [" ".join(w) for w in stems]
vectorizer.fit(texto_pro)
print(vectorizer.vocabulary_)
vector = vectorizer.transform(texto_pro)
print(vector.toarray())
```

# Procesamiento de texto

## TfidfVectorizer

Este modelo trabaja a partir de puntuaciones de palabras que tratan de resaltar las palabras que son más interesantes, más frecuentes en un documento en específico, pero no en todos los documentos.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
#Casos de n_gramas ngram_range
texto_pro = [" ".join(w) for w in stems]
vectorizer.fit(texto_pro)
print(vectorizer.vocabulary_)
vector = vectorizer.transform(texto_pro)
print(vector.toarray())
```

# Procesamiento de texto

## HashingVectorizer

El problema del conteo es que los vocabularios pueden ser muy grandes y requieran vectores muy ampplios. La idea es convertir las apalabras a hash de número enteros, lo que evitará tener vocabulario y se tendrá un vector de tamaño fijo. Sin embargo, puede representar pérdida de información.

```
from sklearn.feature_extraction.text import
    HashingVectorizer
vectorizer = HashingVectorizer(n_features=20)#Casos de
    n_gramas ngram_range
#Casos de n_gramas ngram_range
texto_pro = [" ".join(w) for w in stems]
vectorizer.fit(texto_pro)
vector = vectorizer.transform(texto_pro)
print(vector.toarray())
```



# Contenido

## 1 Procesamiento de texto

- Introducción
- Limpieza del texto
- Vectorización

## 2 Procesamiento de Imágenes

- Introducción
- Procesado

# Procesamiento de imágenes

## Introducción

Para trabajar con imágenes en modelos de redes neuronales es necesario realizar los siguientes pasos:

- Las imágenes deben tener el mismo tamaño
- Reducir información, transformando de color a blanco y negro (depende del problema)
- Transformar a un vector

# Procesamiento de imágenes

## Fuentes de datos

Sklearn cuenta con datasets de imágenes listos para trabajar, revisar:  
https:

[//scikit-learn.org/stable/datasets/toy\\_dataset.html](https://scikit-learn.org/stable/datasets/toy_dataset.html)

# Procesamiento de imágenes

## Escalado de imágenes

Para establecer el tamaño de una imagen usaremos las

```
from sklearn import datasets
from skimage import transform

digits = datasets.load_digits().images
digits_scaled = [transform.resize(image, output_shape
                                   =(10,10))
                 for image in digits]
```

# Procesamiento de imágenes

## Escala de grises

También podemos eliminar el color y sólo pasar a imágenes en blanco y negro. Este paso depende del problema que se busque solucionar, ya que representa pérdida de información.

```
from sklearn import datasets
from skimage.color import rgb2gray

images = datasets.load_sample_images().images
images_gray = [rgb2gray(image)
                for image in images]
```

# Procesamiento de imágenes

## Vectorización

Una imagen es un vector 2D (blanco y negro) o 3D (si es a color), para que la red Neuronal acepte como entrada una imagen debemos transformarla en un vector 1D, para esto usaremos reshape y flatten de Numpy.

```
from sklearn import datasets
import numpy as np
digits = np.array(datasets.load_digits().images)
images = np.array(datasets.load_sample_images().images)
vector_digits = digits.reshape(-1,digits.shape[1]*digits.
    shape[2])
vector_images = images.reshape(-1,images.shape[1]*images.
    shape[2]*images.shape[3])
print(vector_digits.shape,vector_images.shape)
```

# ¿Preguntas?

Próximo tema: Aprendizaje profundo