

Taller de Aprendizaje Automático

Actividades Taller 2

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería



UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Montevideo, 2025

Tabla de contenido

- 1 Objetivos del Taller
- 2 El problema
- 3 Presentación de Comet

Objetivos del Taller 2

- Entender la importancia del preprocesamiento de los datos
- Un primer acercamiento a técnicas clásicas de Procesamiento de Lenguaje Natural
 - Expresiones regulares
 - Tokenization, Lemmatization
 - Modelo *bag-of-words*
- Profundizar en la utilización de *pipelines*
- Una herramienta para gestionar experimentos: [Comet](#)

Clasificación de sentimientos

- Se cuenta con críticas de películas y se quiere determinar si la crítica es positiva o negativa
- Tareas involucradas:
 - ① Levantar los datos
 - ② Limpiar el texto
 - ③ Generar un vector de características
 - ④ Entrenar un clasificador
 - ⑤ Evaluarlo

Los datos

- Se dispone de 50000 críticas de cine para desarrollar el modelo, algunas de ellas positivas y otras negativas



Figure: Nubes de puntos iniciales

Limpieza de texto

- En el texto aparecen *tags html* y otros caracteres que no aportan mucha información
- Se utilizan **expresiones regulares** para limpiar el texto

```
def simple_preprocessor(text):  
    # se eliminan los tags html  
    text = re.sub('<.*?>', '', text)  
  
    # se eliminan caracteres 'non-words'  
    # Words characters are a letter or digit or underbar [a-zA-Z0-9_].)  
    text = re.sub('[\W]+', ' ', text)  
  
    text = text.lower()  
    return text
```

Tokenizer

- Separación del texto en palabras

```
def simple_tokenizer(text):  
    return text.split()
```

- Opciones más sofisticadas
 - Stemming
 - **Lemmatization**

Modelo Bag of Words

- Construye un vector de características a partir de una secuencia de palabras
- Para ello
 - Se crea un diccionario de palabras
 - Se construye un vector de características de largo igual al tamaño del diccionario. Cada elemento del vector representa las veces que aparece en el documento la palabra del diccionario.

```
from sklearn.feature_extraction.text import CountVectorizer
corpus = ['This is the first document.',
          'This document is the second document.',
          'And this is the third one.',
          'Is this the first document?']
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names())
```

```
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

```
print(X.toarray())
```

```
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```


Variantes en la generación del modelo

1 Stop Words

- palabras muy frecuentes no son utilizadas

2 tf-idf: term frequency - inverse document frequency

$$tf-idf(t, d) = tf(t, d) \times idf(t, D)$$

- Utilizar *TfidfTransformer*

3 N-gramas

- considera como un elemento a toda secuencia de *n-palabras*

Comet