PROY-NB01-PREPROCESAMIENTO

December 9, 2018

1 Proyecto final. Datamining.

1.0.1 Análisis tweets UEFA Champions League Final 2018

1.0.2 Participantes:

Gonzalo de las Heras de Matías - Jorge de la Fuente Tagarro - Alejandro Amarillas Cámara - Sergio Sampio Balmaseda.

1.0.3 Notebook (1/4). Preprocesamiento y transformación del dataset.

1.0.4 Objetivo del notebook:

Este notebook se centra en preprocesar el dataset, limpiando las columnas existentes y generando aquellas nuevas que sean necesarias.

Librerias

In [ ]:
```
import pandas as pd
import numpy as np
import string
import math
import re
import nltk
import sys
import goslate
from matplotlib import *
from pylab import *
from datetime import datetime, timedelta
from wordcloud import WordCloud, STOPWORDS
from collections import Counter
from nltk.tokenize import TweetTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn
from nltk.corpus import sentiwordnet as swn
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from google.oauth2 import service_account
from google.cloud import translate
```

title

```
from sklearn.feature_extraction.text import TfidfVectorizer

credentials = service_account.Credentials.from_service_account_file("/Users/gonzalo/Do
```

Descarga de datasets necesarios de nltk

In [ ]:
```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('sentiwordnet')
nltk.download('omw')
```

Funciones

In [ ]:
```
def ProcesarTweet(Tweet):
    """
    Función para eliminar del texto del tweet: enlaces, tildes, # etc.
    """
    Tweet = re.sub(r"\&\w*;", "", Tweet)
    Tweet = re.sub("@[^\s]+","",Tweet)
    Tweet = re.sub("\[[^\s]+\]","",Tweet)
    Tweet = re.sub(r"\$\w*", "", Tweet)
    Tweet = Tweet.lower()
    Tweet = re.sub(r"https?:\/\/.*\/\w*", "", Tweet)
    Tweet = re.sub(r"https...", "", Tweet)
    Tweet = re.sub(r"á", "a", Tweet)
    Tweet = re.sub(r"é", "e", Tweet)
    Tweet = re.sub(r"í", "i", Tweet)
    Tweet = re.sub(r"ó", "o", Tweet)
    Tweet = re.sub(r"ú", "u", Tweet)
```

```
        Tweet = re.sub(r"#\w*", "", Tweet)
        Tweet = re.sub(r"[" + string.punctuation.replace("@", "") + "]+", " ", Tweet)
        Tweet = re.sub(r"\b\w{1,2}\b", "", Tweet)
        Tweet = re.sub(r"\s\s+", " ", Tweet)
        Tweet = Tweet.lstrip(" ")
        Tweet = "".join(c for c in Tweet if c <= "\uFFFF")
        return Tweet

    def PreprocesarTexto(Texto):
        """
        Función para eliminar palabras que no afectan al análisis de sentimiento y signos
        """
        nopunc = [char for char in list(Texto) if char not in string.punctuation]
        nopunc = "".join(nopunc)
        return [word for word in nopunc.lower().split() if word.lower() not in stopwords.w

    def CodificarPalabra(Palabra):
        """
        Función para codificar una palanra como ASCII.
        """
        return Palabra.encode("ascii", errors="ignore").decode()
```

1.1 1. Carga de datos

```
In [ ]: DIR_DATASET_JSON = "/Users/gonzalo/Documents/Datasets/tweets-ucl-final-2018/data.json"
        if "json" not in globals():
            json = pd.read_json(DIR_DATASET_JSON, lines=True)
```

1.3 Construcción dataset final

```
In [ ]: data = json.copy()
        Dataset_Final = pd.DataFrame()
```

1.2 2. Limpieza de datos

Primera fase del proceso de KDD en el que limpiamos los datos

1.2.1 2.1 Eliminación de columnas

2.1.1 Coordinates

```
In [ ]: if len(data[data['coordinates'] == None].index) == 0:
            print("La columna 'coordinates' está toda a NaN, borrando columna.")
            del data['coordinates']
```

2.1.2 Contributors

```
In [ ]: if len(data[data['contributors'] == data['contributors']].index) == 0:
            print("La columna 'contributors' está toda a NaN, borrando columna.")
```

```
        del data['contributors']
```

2.1.3 Withheld_in_countries

```
In [ ]: withheld_in_countries = len(data[data['withheld_in_countries'] == data['withheld_in_cou
        if  withheld_in_countries== 0:
            print("La columna 'withheld_in_countries' está toda a NaN, borrando columna.")
            del data['withheld_in_countries']
        else:
             print('Hay ', withheld_in_countries, "ocurrencias")
```

Hay muy pocas ocurrencias que no sean nulas, eliminamos la columna entera al no ser de utilidad.

```
In [ ]: del data['withheld_in_countries']
```

2.1.4 favorite_count

```
In [ ]: favorite_count = len(data[data['favorite_count'] != 0.0].index)
        if  favorite_count == 0:
            print("La columna 'favorite_count' está toda a 0.0")
            del data['favorite_count']
        else:
            print('Hay ', favorite_count, "ocurrencias")
```

2.1.5 geo

```
In [ ]: geo_count = len(data[data['geo'] == None].index)
        if geo_count == 0:
            print("La columna 'geo' está toda a NaN, borrando columna.")
            del data['geo']
        else:
            print('Hay ', geo_count, "ocurrencias")
```

2.1.6 filter_level

```
In [ ]: filter_level_count = len(data['filter_level'].unique().tolist())
        if filter_level_count == 1:
            print("La columna 'filter_level' es siempre igual.")
            del data['filter_level']
        else:
            print('Hay ', filter_level_count, "valores distintos")
```

2.1.6 place (se recogerá la información en la parte de usuarios) Después de haber probado unos cuentos, nos quedamos con la localizacion de los usurios, casi siempre es la misma.

```
In [ ]: del data['place']
```

2.1.8 Campos que no aportan información

```
In [ ]: del data['display_text_range']
        del data['truncated']
```

1.2.2 2.2 Eliminación filas

2.2.1 Eliminación filas nulas

```
In [ ]: data = data.drop(data[data['id'] != data['id']].id.index)
        data = data.reset_index(drop=True)

        Dataset_Final['id'] = data['id'].astype('int64')
        del data['id']
        del data['id_str']
```

1.2.3 2.3 Volcado al dataset final

2.3.1 Conversión de fecha completa a solo tiempo (todos los tweets son del mismo día)

```
In [ ]: data['created_at'] += pd.DateOffset(hours=2)
        Dataset_Final['hora'] = data['created_at']
```

2.3.2 Formateo de fuente (plataforma desde el que se lanza el tweet)

```
In [ ]: # Extraemos la fuente que se encuentra como el literal de un hipervínculo.
        Dataset_Final['fuente'] = data['source'].replace('<a[^>]*>([^<]+)<\/a>', r'\1', regex=

        # Categorizamos las funtes más importantes.
        Dataset_Final.loc[Dataset_Final['fuente'] == 'Twitter for Android', 'fuente'] = 'Andro
        Dataset_Final.loc[Dataset_Final['fuente'] == 'Twitter for iPhone', 'fuente']  = 'iPhon
        Dataset_Final.loc[Dataset_Final['fuente'] == 'Twitter for iPad', 'fuente']  = 'iPad'
        Dataset_Final.loc[Dataset_Final['fuente'] == 'Twitter Web Client', 'fuente']  = 'Twitt
        del data['source']
```

2.3.3 Idioma

```
In [ ]: Dataset_Final['idioma'] = data['lang']
        del data['lang']
```

2.3.4 Id tweet respuesta

```
In [ ]: data['in_reply_to_status_id'].fillna(-1)
        Dataset_Final['tweet_respuesta_id'] = data['in_reply_to_status_id']
        del data['in_reply_to_status_id']
        del data['in_reply_to_status_id_str']
```

2.3.5 Id usuario tweet respuesta

```
In [ ]: data['in_reply_to_user_id'].fillna(-1)
        Dataset_Final['tweet_respuesta_usuario_id'] = data['in_reply_to_user_id']
        del data['in_reply_to_user_id']
        del data['in_reply_to_user_id_str']
```

2.3.6 Nombre usuario tweet respuesta

```
In [ ]: data['in_reply_to_screen_name'].fillna(-1)
        Dataset_Final['tweet_respuesta_nombre_twitter'] = data['in_reply_to_screen_name']
        del data['in_reply_to_screen_name']
```

2.3.7 Información anidada

```
In [ ]: id_usuarios = []
        name = []
        screen_name = []
        description = []
        verified = []
        location = []
        followers_count = []
        listed_count = []
        favourites_count = []
        created_at = []
        usuario_lang = []
        statuses_count = []
        text = []
        hashtag = []
        user_mentions_screen_name = []
        user_mentions_id = []
        user_mentions_name = []

        # Iteración sobre todos los tweets.
        for index, row in data.iterrows():

            # Extraemos los campos necesarios.
            datos_usuario = row["user"]
            extended_tweet = row["extended_tweet"]
            entities = row["entities"]
            statuses_count.append(datos_usuario["statuses_count"])
            location.append(datos_usuario["location"])
            id_usuarios.append(datos_usuario["id"])
            name.append(datos_usuario["name"])
            screen_name.append(datos_usuario["screen_name"])
            description.append(datos_usuario["description"])
            verified.append(datos_usuario["verified"])
            followers_count.append(datos_usuario["followers_count"])
```

```
favourites_count.append(datos_usuario["favourites_count"])
created_at.append(datos_usuario["created_at"])
usuario_lang.append(datos_usuario["lang"])

# Menciones en un tweet.
user_mentions_screen_name_temp = []
user_mentions_id_temp = []
user_mentions_name_temp = []

# Menciones.
for user in entities["user_mentions"]:
    user_mentions_screen_name_temp.append(user["screen_name"])
    user_mentions_id_temp.append(user["id"])
    user_mentions_name_temp.append(user["name"])

# Screename mención.
if len(user_mentions_screen_name_temp) > 0:
    user_mentions_screen_name.append(user_mentions_screen_name_temp)
else:
    user_mentions_screen_name.append(-1)

# Id mención.
if len(user_mentions_id_temp) > 0:
    user_mentions_id.append(user_mentions_id_temp)
else:
    user_mentions_id.append(-1)

# Nombre mención.
if len(user_mentions_name_temp) > 0:
    user_mentions_name.append(user_mentions_name_temp)
else:
    user_mentions_name.append(-1)

# Existe la entidad extendida.
if extended_tweet is not "nan":
    text.append(row["text"])
    hashtags_temp = []
    for i in range(0, len(entities["hashtags"])):
        temp_hashtag = CodificarPalabra(entities["hashtags"][i]["text"].lower())
        if len(temp_hashtag) > 0:
            hashtags_temp.append(temp_hashtag)
    if len(hashtags_temp) > 0:
        hashtag.append(hashtags_temp)
    else:
        hashtag.append(-1)
else:
    text.append(extended_tweet["full_text"])
    hashtags_temp = []
```

```
            for i in range(0, len(extended_tweet["entities"]["hashtags"])):
                hashtags_temp.append(extended_tweet["entities"]["hashtags"][i]["text"].
            if len(hashtags_temp) > 0:
                hashtag.append(hashtags_temp)
            else:
                hashtag.append(-1)
```

Colocamos cada lista en una columna del dataset

```
In [ ]: Dataset_Final['usuario_id'] = id_usuarios
        Dataset_Final['usuario_nombre'] = name
        Dataset_Final['usuario_nombre_twitter'] = screen_name
        Dataset_Final['usuario_localizacion'] = location
        Dataset_Final['usuario_idioma'] = usuario_lang
        Dataset_Final['usuario_verificado'] = verified
        Dataset_Final['usuario_numero_seguidores'] = followers_count
        Dataset_Final['usuario_numero_favoritos_hechos'] = favourites_count
        Dataset_Final['usuario_numero_tweets'] = statuses_count
        Dataset_Final['usuario_numero_creacion'] = created_at
        Dataset_Final['hashtag'] = hashtag
        Dataset_Final['texto_original'] = text
        Dataset_Final['mencion_usuario_id'] = user_mentions_id
        Dataset_Final['mencion_usuario_nombre'] = user_mentions_name
        Dataset_Final['mencion_usuario_nombre_twitter'] = user_mentions_screen_name

        del data['user']
        del data['entities']
```

Rellenamos campos nulos y convertimos las columnas numéricas a int

```
In [ ]: Dataset_Final.fillna(-1, inplace=True)
        Dataset_Final.tweet_respuesta_id = Dataset_Final.tweet_respuesta_id.astype(int)
        Dataset_Final.tweet_respuesta_usuario_id = Dataset_Final.tweet_respuesta_usuario_id.as
```

1.3 3. Transformación y generación

1.3.1 3.1 Limpiamos el texto de los tweets

```
In [ ]: Dataset_Final['texto_limpio'] = Dataset_Final['texto_original'].apply(ProcesarTweet)
        Dataset_Final['texto_traducido'] = "-1"
```

1.3.2 3.2 Traducimos los textos de los tweets

```
In [ ]: translate_client = translate.Client(credentials=credentials)
        terminado = False
        print("Hay", len(Dataset_Final), "filas")
        for index, row in Dataset_Final.iterrows():
            if index % 100 == 0:
                print(index, "filas traducidas.")
```

```
        if row["idioma"] != "en" and row["texto_traducido"] == "-1":
            # Traducción al inglés.
            target = 'en'
            # Traducción.
            translation = translate_client.translate(row["texto_limpio"], target_language=t
            Dataset_Final.loc[index, 'texto_traducido'] = translation['translatedText']
        else:
            Dataset_Final.loc[index, 'texto_traducido'] = row['texto_limpio']
```

1.3.3 3.3 Analizamos el sentimiento del tweet

```
In [ ]: # Vectorizamos el tweet en un array de características.
        vectorizador = TfidfVectorizer(max_df=0.5, max_features=10000, min_df=7, stop_words='en
        vectorizador.fit_transform(Dataset_Final['texto_traducido'].str.upper())

        # Objeto analizador de sentimientos.
        analizador = SentimentIntensityAnalyzer()

        # Analizamos el sentimiento.
        Dataset_Final['sentimiento_compound_polarity'] = Dataset_Final["texto_traducido"] \
            .apply(lambda x:analizador.polarity_scores(x)['compound'])
        Dataset_Final['sentimiento_neutral'] = Dataset_Final["texto_traducido"] \
            .apply(lambda x:analizador.polarity_scores(x)['neu'])
        Dataset_Final['sentimiento_negativo'] = Dataset_Final["texto_traducido"] \
            .apply(lambda x:analizador.polarity_scores(x)['neg'])
        Dataset_Final['sentimiento_positivo'] = Dataset_Final["texto_traducido"] \
            .apply(lambda x:analizador.polarity_scores(x)['pos'])

        # Clasificación del resultado.
        Dataset_Final['sentimiento_tipo'] = ""
        Dataset_Final.loc[Dataset_Final['sentimiento_compound_polarity'] > 0 ,'sentimiento_tipo
        Dataset_Final.loc[Dataset_Final['sentimiento_compound_polarity'] == 0,'sentimiento_tipo
        Dataset_Final.loc[Dataset_Final['sentimiento_compound_polarity'] < 0 ,'sentimiento_tipo
```

1.3.4 3.4 Separamos el texto del tweet en palabras

```
In [ ]: Dataset_Final['tokens'] = Dataset_Final['texto_traducido'].apply(PreprocesarTexto)
```

1.3.5 3.5 Eliminamos hashtags no válidos

```
In [ ]: Dataset_Final= Dataset_Final.reset_index(drop=True)
        for index, row in Dataset_Final.iterrows():
            hashtags = []
            print(index)
            if row["hashtag"] != "-1" and row["hashtag"] != -1:
                for hastag in row["hashtag"]:
                    if hastag != "_" and hastag != "__" and hastag != "___" and hastag != "ucl
                        hashtags.append(hastag)
```

```
            if len(hashtags) > 0:
                Dataset_Final.set_value(index, "hashtag", hashtags)
            else:
                Dataset_Final.loc[index, "hashtag"] = "-1"
```

```
In [ ]: Dataset_Final.to_json("_datos_limpios.json")
        Dataset_Final.head()
```

2 Referencias

https://www.kaggle.com/xvivancos/tweets-during-r-madrid-vs-liverpool-ucl-2018
https://github.com/pbugnion/gmaps
https://pandas.pydata.org/
Apuntes de la asignatura