Proyecto Marvick.

**Alumno:** Germán Suárez

**Fecha:** 09 de Agosto 2021

# Introducción

Para este proyecto se utilizaron tweets etiquetados provenientes de países hispanohablantes (Costa Rica, España, México, Perú, Puerto Rico y Uruguay) conseguidos de la página de TASS: ‘Workshop on Semantic Analysis at SEPLN’ -http://tass.sepln.org/-.

En la primera parte se intenta predecir el sentimiento (Positivo o Negativo) de los tweets utilizando una red neuronal LSTM y diversos modelos de Machine Learning.

Para la segunda parte del ejercicio se busca predecir el país de origen de los tweets utilizando una red neuronal LSTM y aplicando SMOTE sampling para los casos que no tengan suficientes muestras.

# Sentiment Analysis

## Armado de Dataframe

## Para armar el dataset primero se pasan los diferentes archivos XML a un dataframe de pandas donde se puede ver la etiqueta del sentimiento asociado y de qué país proviene. En una segunda instancia, se eliminan los tweets que no tienen etiqueta o si la misma es neutral dejando solo los tweets con sentimientos positivos y negativos.

## Limpieza de Datos

Los tweets se limpian de la siguiente manera:

* Se pasa todo a minúscula
* Se quitan puntuaciones y símbolos
* Se quitan las referencias a retweets (rt)
* Se quitan las stopwords
* Se hace stemmering a las palabras, intentando reducir el vocabulario total
* Se quitan los caracteres que no sean alfabéticos

Luego se utiliza CountVectorizer de ScikitLearn para conocer cuántos tweets y palabras hay en el dataset. En total hay unos 7037 tweets que cuentan con 9977 palabras, este último dato se utiliza para la capa de embedding de la red neuronal.

## Armado de WordCloud

Se arman dos WordClouds con los tweets ya limpios y utilizando una máscara de Latinoamérica. Primero se importa la imagen a utilizar, se pasa a blanco y negro para luego utilizarla como máscara en la función que crea el WorldCloud.



*Ilustración 1 - WorldCloud con las palabras de los tweets con sentimiento positivo (fondo blanco) y negativo (fondo negro)*

## Ensamble y entrenamiento de la red neuronal LSTM

Para entrenar la red neuronal primero se transforman los tweets en vectores utilizando la función Tokenizer de Keras, siguiente se separa el dataset en *train*, *test* y *validation*. La red neuronal consta de las siguientes capas:

* Embedding ()
* Dropout (0.5)
* LSTM de 8 unidades
* Densa de 2 unidades y activación sigmoid

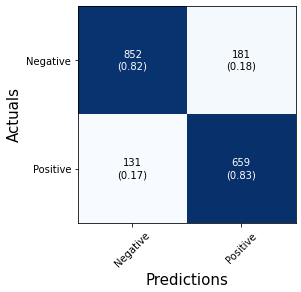
Se compila el modelo utilizando el optimizador *adam*, la función de pérdida *binary\_crossentropy* y *accuracy* para las métricas. Primero se entrena la red de cero sin utilizar *transfer learning*, durante 10 *epochs* y un *batch\_size* de 32 teniendo poco más de 90 mil parámetros entrenables. No se consideró necesario entrenar durante más de 10 *epochs* debido a que convergía rápidamente.

## Resultados

Los resultados para el set de test de este primer entrenamiento a nivel total fueron los siguientes:

* *Score*: 0.45
* *Accuracy*: 83 %

Viendo la matriz de confusión se puede apreciar que el modelo tiene un mejor desempeño para predecir tweets negativos, con una precisión del 87% y un recall de 82%, de todas formas, el desempeño con los tweets positivos también es bueno con una precisión de 78% y un recall de 83%.



*Ilustración 2 - Matriz de confusión, el valor de arriba es la cantidad de tweets y entre paréntesis el recall.*

## Entrenamiento con transfer learning

Para esta parte se utiliza un *embedding* ya entrenado como punto de partida, el mismo se puede bajar del siguiente [link](https://zenodo.org/record/3234051/files/embeddings-s-model.vec?download=1). El modelo viene en formato vectorial por lo que primero a realizar es extraer la matriz de los pesos.

En una primera instancia se entrenó el modelo utilizando como punto de partida el embedding ya entrenado, pero dejando los pesos libres para re-entrenarlos con los datos disponibles mediante el parámetro *trainable = True* en la capa de Embedding. El resto de la capa queda exactamente igual a la parte anterior de manera de poder comparar resultados. Al final tendremos más de 39 millones de parámetros entrenables por lo que el entrenamiento resulta bastante más lento.

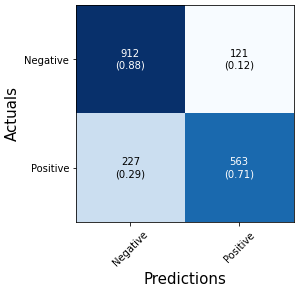
En una segunda instancia se realizó un entrenamiento utilizando los pesos congelados en la capa de *embedding* (*trainable = False)* pero los resultados fueron muy pobres por lo que no se documentan los mismos.

## Resultados

Los resultados para el set de test no varían mucho respecto a la parte anterior:

* *Score*: 0.43
* *Accuracy*: 81 %

La matriz de confusión muestra que el comportamiento es similar respecto a la diferencia de desempeño entre los tweets negativos y positivos. Para los negativos la precisión empeoró 7pp (pasa de 87% a 80%) mientras que el recall mejoró 6pp (pasa de 82% a 88%). En el caso de los positivos la precisión mejoró 4pp (pasó de 78% a 82%) mientras que el recall empeoró bastante bajando 12pp (pasó de 83% a 71%).



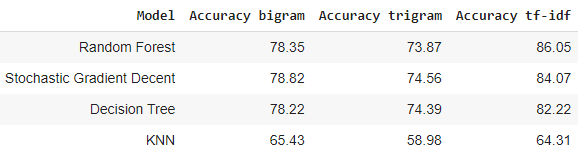
*Ilustración 3 - Matriz de confusión para entrenamiento con transfer learning, el valor de arriba es la cantidad de tweets y entre paréntesis el recall.*

## Predicción utilizando diferentes modelos de Machine Learning

A continuación, se realiza la misma clasificación, pero utilizando modelos de machine learning, específicamente *K-NN*, *Decision Tree*, *Random Forest* y *Stochastic Gradient Descent*.

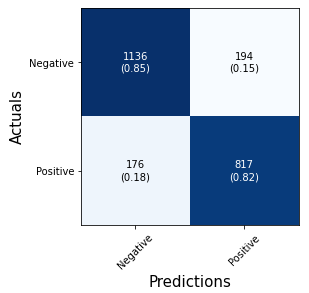
Para entrenar los modelos primero se crean uni-gramas, bi-gramas y tri-gramas, luego se transforman los uni-gramas en una matriz tf-idf con el objetivo de disminuir el impacto de las palabras que tienen mucha ocurrencia.

Todos los modelos mencionados anteriormente fueron entrenados con bi-gramas, tri-gramas y la matriz de tf-idf brindando los siguientes resultados:



*Ilustración 4 - Tabla con comparación entre modelos*

Se incluye la matriz de confusión del modelo *Random Forest* entrenado con tf-idf debido a que fue la de mejor desempeño.



*Ilustración 5 - Matriz de confusión para el modelo Random Forest entrenado con tf-idf*

## Conclusiones

A continuación, se listan las conclusiones que se obtienen del análisis

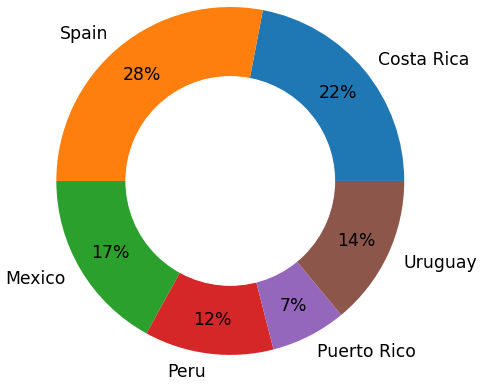
* Utilizar un *embedding* pre-entrenado no resultó en una mejora de la predicción, ya sea entrenando el modelo dejando los pesos libres, donde se obtuvieron resultados muy similares a entrenar arrancando de cero, ni cuando se congelaron los pesos y se re-entrenaron las siguientes capas, brindando predicciones muy malas.
* Para los modelos de machine learning, la configuración con tf-idf fue la que brindó mejores resultados, particularmente para los modelos *Random Forest*, *SGD* y *Decision Tree* los resultados son muy similares a los provistos por la red neuronal LSTM.
* En todos los casos se puede notar un sesgo hacia los tweets negativos, tanto el recall como la precisión resultan mejor que para los positivos, esto puede deberse a que el dataset no está completamente balanceado

# Country Prediction

## Estudio del dataset

Primero se estudia el dataset, calculando cuántos tweets se tienen para cada uno de los países presentes en proporción del total.

Viendo el gráfico circular se puede apreciar que España es el país que cuenta con más tweets, contrario a Puerto Rico que apenas representa un 7% de los tweets totales. Debido a la poca proporción de tweets para este último país, se excluye del análisis.



*Ilustración 6 - Gráfico circular con la proporción de tweets por país respecto al total*

Para armar los sets de datos de entrenamiento, validación y testeo se procede de manera similar a como se realizó en el análisis de sentimiento, la única diferencia es que ahora las etiquetas son los países y no los sentimientos.

## Ensamble y entrenamiento de la red neuronal LSTM

En primer lugar, se intenta predecir de qué país provienen los tweets utilizando una red neuronal LSTM.   
Luego de ciertas pruebas fallidas, se pueden sacar algunas conclusiones para el armado y entrenamiento de la red.

* Se tuvo que aumentar el tamaño del *embedding* respecto al ejercicio de análisis de sentimiento pasando de uno de tamaño 9 a uno de 200. Un *embedding* tan chico no permitía al modelo realizar una buena clasificación, por el contrario, aumentar mucho más la dimensión del mismo no presentaba grandes mejoras.
* Se utilizó un *batch\_size más grande*, utilizando un *batch\_size* de 32 el error de validación aumentaba con el paso de los *epochs*, mientras que utilizando uno de 128 el comportamiento era un poco más estable, mostrando una mejor generalización.

La red está compuesta por las siguientes capas:

* Embedding con un output\_dim de 200
* Densa con 32 unidades y activación *relu*
* Dropout (0.25)
* LSTM de 32 unidades
* Dropout (0.5)
* Densa con 5 unidades (una por país) y activación *softmax*

Se compila el modelo utilizando el optimizador *adam*, la función de pérdida *categorical\_crossentropy* y *accuracy* para las métricas. Se entrena con 10 *epochs* y un *batch\_size* de 128.

## Resultados

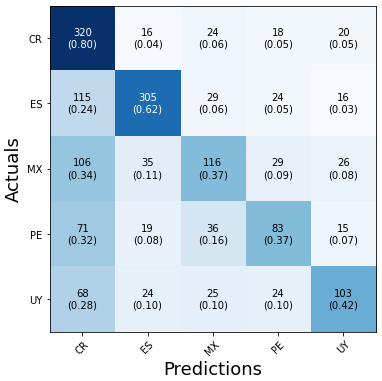
Los resultados generales para el set de test son los siguientes:

* *Score*: 1.64
* *Accuracy*: 56 %

A nivel país se tienen los siguientes resultados:

* Costa Rica 🡪 Recall: 80.4% / Precisión: 47.1%
* España 🡪 Recall: 62.4% / Precisión: 76.4%
* México 🡪 Recall: 37.2% / Precisión: 50.4%
* Perú 🡪 Recall: 37.1% / Precisión: 46.6%
* Uruguay 🡪 Recall: 42.2% / Precisión: 57.2%

Matriz de confusión resultante:



*Ilustración 7 - Matriz de confusión entre países*

Se puede apreciar como el modelo tiene un sesgo hacia Costa Rica pero no tanto hacia España, cuando este último país es el que posee mayor proporción de tweets, Costa Rica está en segundo lugar. En la siguiente parte del ejercicio se intenta corregir ese sesgo y buscar una mayor cantidad de tweets en la diagonal.

## Entrenamiento con SMOTE sampling

Para esta parte se utiliza SMOTE sampling con el objetivo de tener la misma cantidad de muestras para cada país, generando un dataset más equilibrado. Tanto la red neuronal como los hiper-parámetros se mantienen igual de modo de poder comparar resultados.

## Resultados con SMOTE sampling

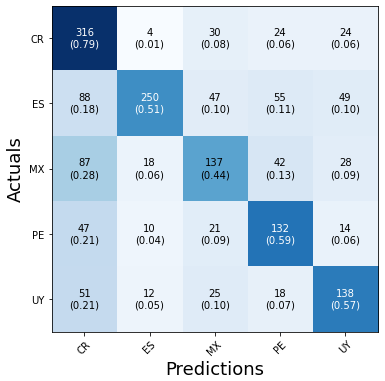
Los resultados generales para el set de test son los siguientes y no presentan grandes variaciones respecto a la parte anterior.

* *Score*: 1.65
* *Accuracy*: 58 %

A nivel país tenemos los siguientes resultados:

* Costa Rica 🡪 Recall: 79.4% / Precisión: 53.7%
* España 🡪 Recall: 51.1% / Precisión: 85.0%
* México 🡪 Recall: 43.9% / Precisión: 52.7%
* Perú 🡪 Recall: 58.9% / Precisión: 48.7%
* Uruguay 🡪 Recall: 56.6% / Precisión: 54.5%

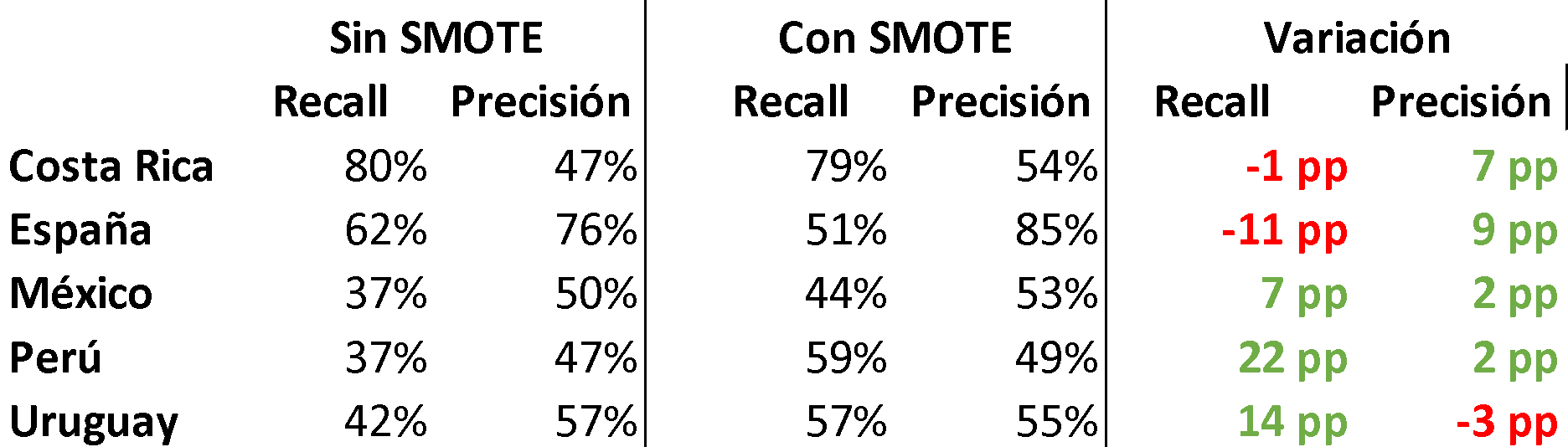
Matriz de confusión resultante:



*Ilustración 8 - Matriz de confusión entrenando con SMOTE sampling*

## Comparación de resultados

En la siguiente tabla se ven los resultados con y sin SMOTE sampling y su variación correspondiente.

**

*Ilustración 9 - Comparación de resultados entrenando con y sin SMOTE sampling*

## Conclusiones

Utilizar SMOTE sampling para balancear el dataset demostró ser una buena estrategia para los países que tenían menos muestras -Uruguay y Perú-, en cambio para España resultó en un empeoramiento del recall, generando más falsos positivos principalmente entre los países que mejoraron más, dicho de otra manera, el modelo confunde más tweets españoles con uruguayos y peruanos.