**Documento Explicativo Predictvia AI Challenge 2020**

Equipo: CMD

Integrantes:

* Mario Catapano
* Massimo Di Gennaro

1. **Preprocesamiento:**

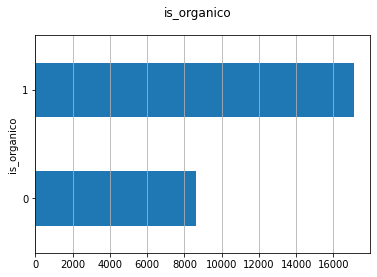
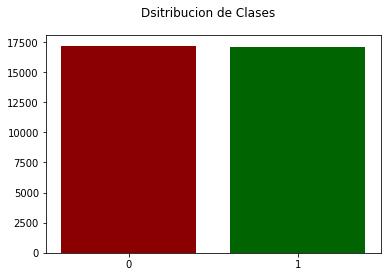
**-Métodos utilizados:**

* Agrupar links de páginas web: todas las direcciones URL fueron sustituidas por una palabra en común (direccionweburl), para agrupar los diferentes enlaces web.
* Agrupar todos los números: Cualquier cifra numérica que apareciera en los tweets fue reemplazada por una palabra en común (valornumerico)
* Agrupar todos los usuarios de Twitter: todos los usuarios que fueran mencionados en tweets (@ejemplodeusuario) fueron sustituidos por una palabra en común (usuariodetwitter)
* Separar letras y caracteres especiales: se separaron todas las palabras de los posibles caracteres especiales (,.?:’”() etc.) que tuvieran adheridos, por ejemplo un tweet como “Hoy, o mañana?” se convertiría en “Hoy , o mañana ?”. Esto se hizo para que no sean consideradas por el modelo como palabras diferentes (este paso no se pudo realizar antes ya que las direcciones URL cuentan con símbolos que, si se separan de las letras, la dirección deja de funcionar o no la reconoce el algoritmo).
* Hashtags en el dataset: no se agrupan los hashtag (#) como se hace con los arroba(@) y URL, porque estos a veces tienen una palabra que da contexto a la frase, como por ejemplo, #música o #arquitectura (o palabras que no están es español pero recurrentes como COVID19), sin embargo para extraer la palabra se separa el ‘#’ del texto y a su vez se sustituye el símbolo ‘#’ por un ‘##’ ya que hay ocasiones de tweets falso donde el # está separado del texto, y si no hacemos este cambio no habría diferencia luego de preprocesar un “#españa” por ejemplo de un “# españa”, con esta manera de preprocesar se evidenciaría la diferencia ya que estos hashtags resultarían en

“## españa’ y “# españa” respectivamente.

* Upsampling/Oversampling: al aplicar esta estrategia, la cual consistió en duplicar los tweets de la clase de minoría en los datos (tweets inorgánicos) logramos subir el F1 score en aproximadamente 0.005

Previo a Oversampling: Luego de Oversampling

* Feature Engineering: Al igual que la técnica de Upsampling esta técnica mejoró nuestro F1 en aproximadamente 0.005, entre los datos que extraemos de los tweets para el Feature Engineering están: número de palabras, número de caracteres, número palabras en mayúscula, número palabras que inician por mayúscula, etc. Y usando la librería Spacy se extrajeron datos como: número de verbos, número de stopwords, número de dígitos/cifras numéricas, número de adjetivos, número de organizaciones, número de palabras desconocidas y conocidas, número de signos de puntuación etc. En total se extraen 34 Features diferentes.
* Tokenización y Padding: por último, se tokenizaron los tweets (a nivel de palabras) y se utilizó padding cero por la derecha, como máxima longitud de tweet se estableció 115 palabras (el tweet real con más palabras luego de preprocesar tenía esa longitud)

**-Métodos descartados:**

A continuación, mencionamos los métodos de preprocesamiento que decidimos no implementar luego de realizar varias pruebas.

* Minúsculas: Se experimentó utilizando el mismo modelo (Embedding y LSTM) y los mismos datos de validación, con la única diferencia siendo que en el preprocesamiento de un intento se pasaron todas las letras a minúsculas y en otro no, y se obtuvieron los siguientes resultados:

|  |  |  |
| --- | --- | --- |
|  | Conservando Mayúsculas | Descartando Mayúsculas |
| LOSS | 0.1292 | 0.1335 |
| ACC | 0.9522 | 0.9481 |
| F1 | 0.9506 | 0.9457 |

Se puede observar que los resultados obtenidos con el uso de mayúsculas son superiores y, por lo tanto, las mayúsculas se dejaron en los tweets.

* Análisis de idioma usando Langdetect: se probó quitando todos los tweets que no estaban en español, sin embargo, se perdían muchos tweets que no se encontraban realmente en otro idioma, por ello, luego se experimentó quitando solo aquellos que estaban en inglés, pero no se utilizó al final dado que disminuye el rendimiento y el set de validación tenía otros idiomas también (algunos tweets en inglés principalmente).
* Lematización: se decidió no utilizar este método, ya que algunas de las palabras no eran convertidas correctamente a su raíz, como, por ejemplo, en el caso de la palabra “serie”, la lematización convertía esta palabra en “seriar”, lo cual no es correcto.
* Quitar stopwords: este método de preprocesamiento se descartó ya que se perdían demasiadas palabras que consideramos importantes para el modelo, ya que las stopwords en español son muchas más palabras que las stopwords en inglés.
* Quitar puntuación: en el caso de la puntuación, este método se descartó ya que consideramos que la puntuación puede proveer de pistas importantes al modelo, a la hora de predecir si un tweet es orgánico o inorgánico, por ejemplo, la cantidad de signos de puntuación puede ser un factor importante dentro del Feature Engineering, y por ello, se decidió no eliminarla.
* Agrupar emojis y hashtags: luego de varios experimentos, decidimos descartar esta opción ya que no mejora los resultados, a pesar de que aumenta la cantidad de palabras sin vector en el embedding pre entrenado más adelante.

1. **Pruebas del modelo:**

Se realizaron varias pruebas al modelo, para encontrar los hiperparámetros óptimos de la red LSTM, estas pruebas fueron guardadas en un documento de Google y posteriormente analizadas.

No se utilizó regularización, ya que se obtuvieron los mejores resultados sin utilizar Dropout.

Para el set de validación se probó utilizar el 30%, 20%, 10%, 5%, 1%, y se obtuvieron los mejores resultados con el 20%.

Se probaron diferentes embeddings, específicamente, no pre entrenados de 50 ,100 y 300 dimensiones, y pre entrenados de 100 dimensiones y 300 dimensiones, sin embargo, para la entrega definitiva se utilizó el embedding de 300 dimensiones, ya que proporcionaba los mejores resultados.

1. **Código y motivos de su uso:**

Para la lectura de los datos, utilizamos un try-except, para que, al no tener acceso a nuestro Google Drive, los datos se carguen directamente, ya sea desde su dirección en Kaggle (../input/aichallenge2020/training.csv) o directamente (training.csv).

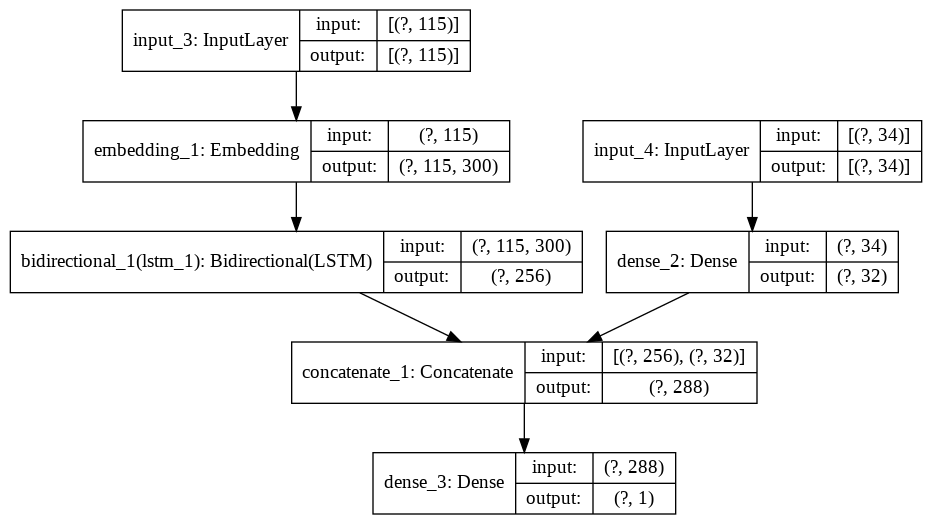
Se analizaron la cantidad de tweets orgánicos e inorgánicos en el dataset, y al encontrar que existe un desbalance, se decidió utilizar Upsampling para balancear la cantidad de tweets inorgánicos (se explicará más adelante). Se buscan las palabras más y menos utilizadas en el dataset como análisis exploratorio de los datos, y además para encontrar aquellas que podrían ser contadas en el Feature Engineering, y que aparezcan mucho en una clase, pero no en la otra, y estas fueron: asteriscos y hashtag (aparecen mucho en tweets inorgánicos) y los guiones (aparecen mucho en tweets orgánicos). Luego se analizaron los tweets reales y falsos más largos, tanto en el set de entrenamiento como en el set de validación, como base para determinar el largo a seleccionar como máxima longitud de tweets (115 fue seleccionado al final).

Luego se vuelven a revisar las palabras más y menos usadas, y se observan los cambios. Para el padding y tokenizado, se utilizaron un total de, aproximadamente, 51.000 palabras diferentes, al contar las palabras diferentes en el dataset. A su vez, se seleccionó un máximo de longitud de 115, luego de verificar que esta es la cantidad de palabras en el tweet orgánico más largo.

Se revisa el correcto funcionamiento de la Tokenización, al convertir los tokens nuevamente en texto y verificar que sea el mismo texto del tweet preprocesado original. Luego, se utiliza un embedding pre-entrenado, descargado de zenodo.org (<https://zenodo.org/record/3234051/files/embeddings-l-model.vec?download=1>), y se leen y obtienen los vectores de 300 dimensiones de cada palabra dentro del embedding que coincida con las palabras que quedan en el dataset luego del preprocesamiento (para este caso si hacemos la comparación con las palabras en minúscula, de tal manera que se le asigne el mismo vector a Casa o casa por ejemplo), con aproximadamente 44000 palabras encontradas y 6000 sin vectores en el embedding (para las palabras sin embedding, se inicializa su vector utilizando una distribución uniforme entre -0,5 y 0,5).

Se realiza upsampling de la clase minoritaria, en este caso, de los tweets inorgánicos, para que ambas categorías posean una cantidad muy similar de datos, y luego se verifica nuevamente la cantidad de tweets orgánicos e inorgánicos.

Luego, se separan los datos en entrenamiento y validación, se calculan las funciones para el puntaje F1, y se crea el modelo con las siguientes características:



El modelo posee dos inputs, uno con las secuencias de entrenamiento (tweets), y otro con las 34 columnas de feature engineering, el primero es pasado por la capa Embedding, la cual a pesar de ser pre entrenada igual la entrenamos porque descubrimos que así obtenemos mejores resultados, y luego por una LSTM bidireccional de 128 neuronas, mientras que el feature engineering es pasado por una capa Dense de 32 neuronas, activación Relu. Estas dos capas luego son concatenadas, y se pasan a una última capa Dense de 1 neurona y activación sigmoide para obtener la probabilidad de que el tweet sea orgánico o inorgánico. Entrenamiento: Optimizador Adam, batch size de 32, 2 epochs, y con función de pérdida binary-crossentropy.

Finalmente, el modelo es guardado, así como la imagen representativa de este (mostrada anteriormente), y se procede a realizar las predicciones usando este modelo sobre el test. Estas predicciones luego son guardadas en un archivo CSV para ser subidas a Kaggle.

1. **Otros Modelos y Arquitecturas probados:**

**4.1. Deep Learning:**

* LSTM Bidireccional:

Este modelo contaba con la capa embedding, seguida de la una capa bidireccional LSTM (con la cual se experimentaron con 64, 128 y 256 neuronas) y la última capa dense (1) sigmoide. Con este modelo la mayor puntuación F1 obtenida fue aproximadamente 0.9

* LSTM con convoluciones:

Igual al modelo anterior, con la diferencia de que, luego de la capa embedding habían un par de capas extras con CONV1D y max pooling, los resultados obtenidos estaban ligeramente por debajo de aquellos obtenidos por el modelo sin convoluciones.

* Transformers:

Este modelo contaba con la estructura Encoder de la arquitectura Transformers, con un solo bloque Transformers de 32 cabezas, la mayor puntuación F1 obtenida con este modelo no llegó al 0.9.

* Convoluciones:

Un modelo que luego de la capa Embedding estaba compuesto por varias capas CONV1D y max pooling, que luego pasarían a capas de Fully Connected. Los resultados fueron bastantes similares al modelo de solo LSTM, y terminamos descartando este por el de LSTM.

**4.2. Machine Learning**

Se decidió realizar pruebas utilizando modelos de machine learning, cuyos resultados se muestran a continuación:

|  |  |
| --- | --- |
| Método | Accuracy |
| Regresión Logística | 0,66 |
| Support Vector Machine (SVM) | 0,67 |
| Random Forest (101 nodos) | 0,78 |

Como se puede observar, los resultados obtenidos se encuentran por debajo de los resultados conseguidos mediante el uso de Deep Learning (LSTM), por lo que estos fueron descartados.