FACULTAD DE INGENIERIA
Universidad de Buenos Aires

75.06/95.58 Organización de Datos
Primer Cuatrimestre de 2020
Trabajo Práctico 2

Grupo:35
Data Hunters

Apellido/s	Nombre/s	Padrón	E-mail
Inneo Veiga	Sebastian Bento	100998	sinneo@fi.uba.ar

Introducción

En el siguiente informe detallaremos los pasos realizados para resolver el problema de predicción de la competencia de Kaggle: “Real or Not? NLP with Disaster Tweets”. Para ello haremos uso de distintos algoritmos y herramientas vistos en la materia con el fin de obtener la puntuación más alta posible.

Link al repositorio de Github: https://github.com/pablbob/9558DataHunters

Limpieza de datos

Esta fue quizás una de las partes más complicadas dado que en el trabajo anterior únicamente habíamos limpiado la columna ‘keyword’ y un poco de ‘location’ sin haber tocado nada ‘text’ por lo que no sabíamos qué podíamos encontrarnos.

Keyword

Lo primero fue quitarle el caracter especial que estaba en algunas de las keyword que equivale al espacio en blanco. A su vez se decidió separar la columna en 2, una en la cual se le aplicó stemming y otra en la que no, para luego obtener distintas features.

Location

Se procedió a quitar los números, limpiar espacios en blanco, quitar comas, guiones y usar la forma larga de algunas abreviaciones de ciudades o países que se encontraron (p. ej: “united states of america” por “usa”).

Text

Lo primero que tratamos fue limpiar todos los caracteres especiales reemplazandolos por espacios vacíos o por su respectiva representación (espacio o letra). En el proceso observamos que habían varios links que decidimos, en un principio, borrar y que luego reemplazamos por la palabra ‘link’ para unificarlos. Dado que estamos analizando desastres nos pareció indiferente el hecho de que una palabra esté en mayúscula o minúscula por lo que decidimos poner todo en minúscula. El siguiente paso fue separar números de letras y reemplazarlas por ‘number’ dado que habían muchos y tenerlos dispersos no agregaban información.
Se decidió separar la columna en 2, una a la cual se le aplicó stemming y otra a la que no para después ser utilizadas por tf-idf y word2vec respectivamente.

Feature Engineering

A continuación se detallarán las distintas herramientas y enfoques utilizadas para crear nuevos features.

Id

Se descartó la columna.

Location

Binary encoding

En un principio se optó por utilizar binary encoding pero dado que quedaba una columna más en el set de entrenamiento que en el de test se descartó y se hicieron pruebas sin esta columna.

TF-IDF

En un siguiente intento por codificar la columna se procedió a aplicar tf-idf sobre la misma y hacer la suma, tomar el máximo y mínimo, el promedio y la mediana de los términos en sus respectivas líneas. Cabe resaltar que con esto se logró una pequeña mejora sobre los resultados generales.

Word2vec

Ante la falta de mejoras sobre el score de 73 se procedió a utilizar la misma idea que con la columna ‘keyword’ de obtener la norma de los vectores que representan a cada palabra (en este caso, ciudades o países). Si bien este nuevo feature no tuvo la importancia que esperábamos, si hizo que avanzáramos a un score de 74.
A su vez, en búsqueda de aumentar los datos se crearon nuevas columnas con la norma de la suma de los n-gramas que se crearon (1, 2, 3, 4-gramas). Esto no alteró el resultado pero se dejó por el momento. Adicionalmente se crearon columnas con la suma de las normas de los distintos n-gramas.

Keyword

Binary Encoding

En un principio se utilizó binary encoding aunque los resultados no fueron muy favorables por lo que no pasó mucho hasta que se cambió.

TF-IDF

Al igual que con ‘location’, se procedió a codificar la columna con tf-idf y aplicar la suma, obtener el máximo y mínimo, el promedio y la mediana de los términos en sus respectivas líneas. Sin embargo, dado que la mayoría de los keyword contenían únicamente 1 palabra, tomar los valores antes mencionados no generaron cambios significativos.

Word2vec

En búsqueda de superar el score de 70 se decidió probar aplicar word2vec a la columna keyword que no tuviera stemming. De esta forma teníamos una representación vectorial única de cada keyword de la cual se obtuvo la norma haciendo que cada keyword fuera representado por un número diferente. Sorprendentemente dio resultados positivos y pudimos avanzar unos 3 puntos en el score.

Text

TF-IDF

En una primera prueba se trató de utilizar todos los términos como columnas con sus respectivos valores dependiendo de la línea en la que aparecieran. Esto fue rápidamente descartado dado que el set de entrenamiento y el de test tenían términos distintos y tratar de limpiar ambos set para que tuvieran los mismo hubiera sido muy trabajoso y quizás no diera los resultados esperados. Sin embargo, no se descartó el uso de esta codificación del todo dado que se procedió a hacer la suma, tomar el máximo y mínimo, el promedio y la mediana de los distintos términos que involucraban cada una de las líneas tratando de buscar valores que representen a los términos en conjunto e individualmente.

Word2vec

Dado que no habían avances con los encodings ya hechos se procedió a probar otro encoding como word2vec utilizando un set ya entrenado (google-news) para obtener los valores de cada término. Se procedió de la misma manera que con tf-idf para obtener nuevas columnas que tuvieran la suma, máximo, mínimo, promedio y mediana pero en este caso se hizo un paso extra, estos valores se tomaron en relación a la distancia coseno de cada uno de los términos que involucraban el texto de la línea en cuestión. Con este encoding sólo no se logró una mejora por encima de lo que ya había pero en conjunto con tf-idf se logró pasar a un score de 70.
Se trató de aplicar una lógica parecida a la que se hizo con keyword y tomar la norma de la suma de los vectores de cada término y la suma de la norma de cada término. Sin embargo, esto no dio resultados favorables en el entrenamiento.
De la misma forma que con 'location', se crearon n-gramas (1, 2, 3 y 4-gramas) de los textos y se tomaron la norma de la suma de cada texto. A simple vista los resultados no mejoraron aunque pareciera ser que se disminuyó el overfitting en los distintos algoritmos aplicados. Además en random forest se visualizó que la columna de 2-gramas es el segundo feature más importante por lo que se deduce que se ganó precisión.
Además de tomar la norma de la suma de cada n-grama también se tomó la suma de las normas de los distintos n-gramas los cuales parecen dar un mejor resultado.

Len text

Al igual que en el TP pasado se hizo uso de la columna len_text que representa el largo del texto. Al principio se usó sólo el largo del texto una vez este estuviera limpio, sin embargo se agregaron 2 columnas más: una columna que representa el largo de la columna original y otro que representa la diferencia entre el largo del texto original y el largo del texto limpio.

Len keyword

Se decidió agregar el largo de cada keyword como columna, si bien la mayoría de los valores eran muy parecidos se creyó que podría agregar precisión al momento de que el modelo prediga.

Len location

De la misma forma que con 'len_keyword', se buscaba agregar features para ayudar a los modelos aunque en este caso los valores eran más variados.

Algoritmos de Machine Learning utilizados

Random Forest

Este algoritmo fue el primero en ser utilizado para dar un pantallazo sobre qué hiper parámetros había que tunear y se le dio uso para saber la importancia de los features que se iban agregando. A pesar de esto, no daba los mejores resultados así que no se utilizó para obtener las predicciones finales.

XGBoost

Fue el segundo algoritmo probado y usado para predecir. Dio mejores resultados que random forest incluso tomando los mejores parámetros del mismo pero no fue el mejor para predecir por sí solo.

LightGBM

Fue el algoritmo que mejor resultados dio por sí sólo al principio pero luego de agregar más columnas se fue quedando al mismo nivel que xgboost. Con este algoritmo se logró llegar a un score de 70.

Catboost

Se probó al principio sin utilizar parámetros y no se lograron mejoras respecto de los demás algoritmos por lo que se dejó de lado un tiempo. Luego de agregar nuevas columnas se volvió a hacer la prueba y parecía que estaba más cerca de los otros algoritmos pero sigue sin ser prioritario probarlo.
Luego de agregar los nuevos features en base a la columna ‘location’ se hizo una prueba con este algoritmo y el resultado fue que score mejoró (de 73 a 74) por lo que se pasó a hacer nuevas pruebas con distintos parámetros en busca de una mejora sobre los resultados.

Bagging

Luego de varias pruebas individuales se probó hacer un ensamble con bagging con los algoritmos antes mencionados con los parámetros que mejor resultado dieron individualmente. Desafortunadamente las pruebas no fueron nada favorables, las predicciones estaban muy lejos de superar a los algoritmos anteriores por sí sólos.

Sequential

Se hicieron algunas pruebas con distintos parámetros pero los resultados fueron pésimos en todos los casos y el tiempo empleado por cada prueba era mayor que el de los otros algoritmos por lo que se decidió no utilizarlo.

Perceptron

Se hicieron unas pruebas pero fue descartado porque no mejoraba los resultados.

Perceptron multicapa

Se hicieron algunas pruebas pero fue rápidamente descartado porque no mejoraba el score.

KNN

Si bien con el set de entrenamiento parecía tener una desarrollo tan bueno como los otros algoritmos sólo ajustando la cantidad de vecinos y la métrica a Manhattan, el resultado en las predicciones fue peor.

Ensamble

Si bien se trató de hacer algo parecido con bagging, en este caso se utilizaron varios de los algoritmos ya mencionados de forma que (aplicando majority voting) llegar a mejorar el score actual (74). Sin embargo, incluso agregando features nuevos no se logró aumentar el score porque la mayoría de los algoritmos (con los parámetros usados) dan resultados muy parecidos.

Búsqueda de hiper-parámetros

Random search

Se utilizó con random forest y xgboost con parámetros que abarquen un cierto rango para así poder identificar si se inclina para un lado o para otro. En ninguno de los casos dio buenos resultados.

Grid search

Se utilizó para lightgbm con parámetros que abarquen un cierto rango para así poder identificar si se inclina para un lado o para otro. No dio buenos resultados.

Manualmente

Se consiguieron mejores resultados haciendo cambios de a poco en cada uno de los hiper-parámetros.

Métrica

Accuracy

Fue la primera métrica utilizada dado que es fácil de entender sin embargo, debido a las diferencias con la métrica utilizada en la competencia (y las diferencias en el score obtenido en el set de entrenamiento con el de test) se dejó de utilizar.

F1-Mean

Se decidió utilizar la misma métrica utilizada en la competencia para obtener una mejor relación de si se estaba teniendo un avance en las pruebas o no.

Algoritmo final usado

El algoritmo que dio más resultado fue LightGBM con un score de 74,532.