COMCOP **hace la entrega del reto de Hackaton 2020**

Equipo conformado por:

Carlos Eduardo Argoti Patiño
Juan Fernando Gonzalez Orrego
Juan Felipe Orozco Cortés

Gracias.

Librerías necesarias:

tensor flow
keras
touch
csv

Reto 1

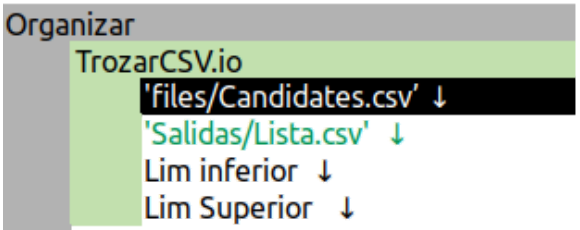
Porcentaje de afinidad: Candidato vs Vacante

Magneto empleos es una plataforma que permite a personas de todos los sectores laborales encontrar empleos a nivel nacional e internacional y actualmente cuenta con una bolsa de candidatos disponibles para que grandes empresas encuentren los mejores perfiles que se ajusten a sus vacantes.

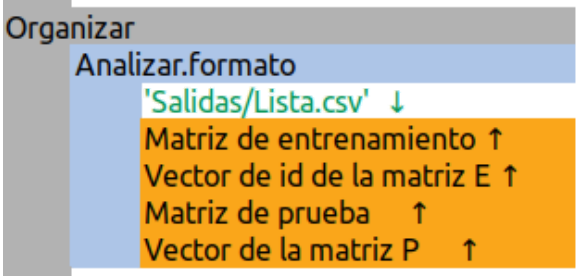
Esta bolsa de candidatos ha ido creciendo de manera exponencial durante los últimos tiempos. Para cada candidato recopilamos desde información personal como: nombres, apellidos, email, teléfonos, entre otros, hasta información relevante para el proceso de reclutamiento como los estudios, habilidades personales, experiencia y perfil laboral. Aunque toda esta información es posible llenarla por cada candidato, es permitido ingresar en Magneto Empleos candidatos con tan solo el nombre y el correo electrónico y de manera fácil importarlos a una vacante específica.

Algoritmo del preprocesamiento de la información

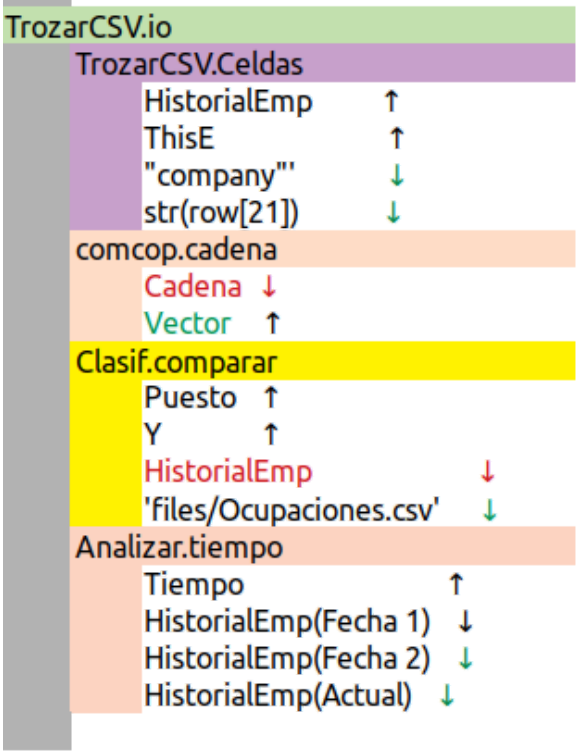
La información a procesar consta de una tablas en formato csv que serán convertidas a matrices de numpy en un formato que pueda ser sometido a las máquinas de boltsman en un siguiente paso; para este pre-procesamiento se programaron los siguientes pasos:



Esta función genera un fichero .csv, en este caso el marcado en verde, con fondo negro está el fichero con la información de los candidatos.



A partir de este fichero generado en la funcion anterior, se da formato de matriz con el fin de introducir la información sintetizada en las máquinas de aprendizaje



Segmenta los archivos tipo JSON deacuerdo al 3º parámetro de esta lista

Convierte una cadena en vector

Compara las ocupaciones con el historial de cada empleado, devolviendo un similar y su tiempo de duraciòn en este

Calcula el tiempo laborado, incluso si está actualmente, si la fecha 1 o 2 està mal, returna 0 días

Clasif.comparar

Clasif.Pareado

Resultados ↑
Data ↑
Identificador ↑
Modificar ↑
Acomparar ↑
Contadorj ↑
Acomparar ↓
Data ↓
Identificador ↓
Index ↓
EmpleoFormacionpsico ↓

Se encarga de comparar el contenido del cada experiencia laboral o académica y retorna su indexación.

La función c.k(Vector, int(identificador)) sirve para hacer un debug controlado
Las funciones dentro del fichero comcop son básicas de càlculos.
Conversiones

↑ Parámetros recibidos de la función

Para predecir el mejor candidato se normalizo de 0 a 1 las características de los usuarios en un matriz de mxn, donde que aplica la función

sc = MinMaxScaler(feature_range = (0, 1)) de sklearn para lograr este resultado.

La inteligencia artificial (IA) que se utilizo son los autoencoder donde se pretende predecir características de una persona por medio de las características que se encuentran en una base de datos. Para nuestro trabajo se utilizó la siguiente matriz de entrada:

0	0.703277	0.703001	0	0.761947	0.160408	0
0	0.703277	0.690211	0	0.761947	0.160408	0
0	0.703277	0.690211	0	0.761947	0.160408	0
0	0.705776	0.69254	0	0.761947	0.160408	0
0	0.707046	0.690211	0	0.761947	0.160408	0
0.190174	0.747522	0.731445	0.190174	0.807218	0.320077	0.190
0	0.703277	0.690211	0	0.761947	0.160408	0
0.101955	0.726997	0.712317	0.101955	0.786217	0.246008	0.101
0	0.703277	0.690211	0	0.761947	0.160408	0
0	0.743097	0.690211	0	0.761947	0.160408	0
0	0.703277	0.693723	0	0.761947	0.174009	0
0	0.703277	0.690211	0	0.761947	0.160408	0

Figura 1: Entrada a los autoencoder

Los elementos en 0 son los valores que la IA le va a asignar al usuario x de acuerdo a los estados que encuentre en la base de datos. Por ejemplo si una persona es experta en Python que características se le asignará a las habilidades en C++, probablemente el autoencoder devuelva un valor real cercano a 0.9 ya que son características que un usuario que programa en black-end pueda desarrollar esas habilidades en poco tiempo.

La estructura que se utilizó es un deep autoencoder con entradas igual a la cantidad de usuarios, con 3 capas profundas una de 10, 5, 10 y al final la cantidad igual al numero de usuarios. Como se observa en la figura 2.

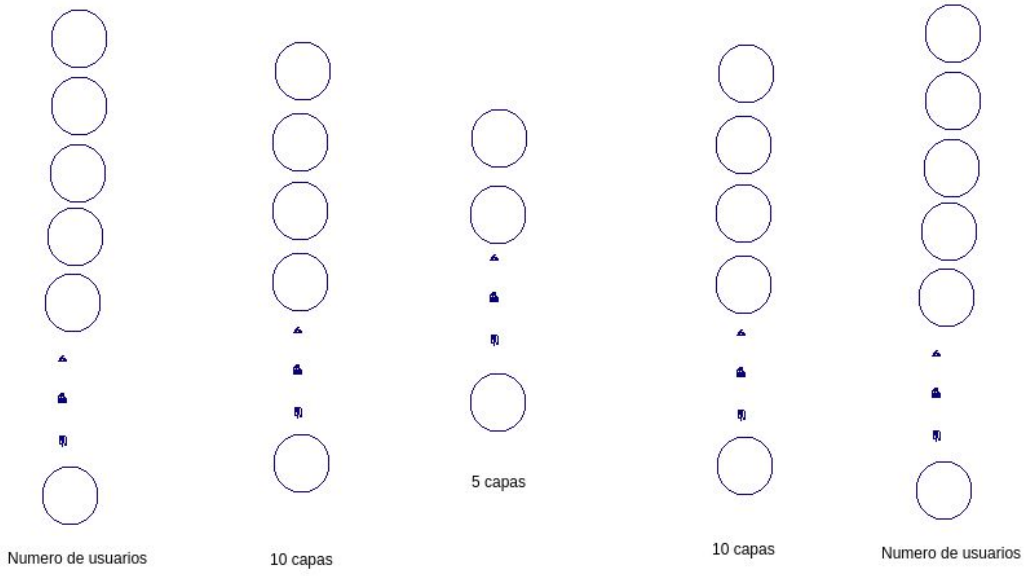


Figura 2: Diagrama de la red neuronal.

Para determinar la convergencia del modelo se utilizó el descenso del gradiente donde se va variando los pesos de la entrada hasta que se distribuyen unas características en la capa final, estas características las encuentra la IA a partir de las entradas, esto hace que no modelo sea de Deep Learning no supervisado. Las neuronas se activan si el elemento de la característica es mayor a 0 más un sesgo b.

Para mayor información de la IA consultar:

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

Para la entrega de los requisitos del reto se ilustra la función utilizada para determinar las características de la entrada a partir de la base de datos entregada.

```
def Organizar(candidatos):
        os.system('rm logs/*')
        desde=0
        hasta=5000000
        print('Inicio del clasificador')
        TrozarCSV.io('files/Candidates.csv','Salidas/Paralelo.csv' , 'Salidas/Lista.csv', desde, hasta,
['"])
        [candidatosTrain, matrizTrain, candidatosTest,
matrizTest]=Analizar.formato('Salidas/Lista.csv')

        return candidatosTrain, matrizTrain, candidatosTest, matrizTest
```

Para predecir la afinidad entre los usuarios y una vacante se comparan utilizó la similitud chi-cuadrado que determina qué tan parecidas son dos entradas. Es importante aclarar que ya se utilizó la predicción del autoencoder y que se tiene un valor real en cada una de las características.

Para mayor información consultar el siguiente link:
https://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/

Para la entrega de los requisitos del reto se ilustra la función utilizada para determinar la similitud entre las vacantes y el empleado. En este caso la vacante es el último usuario.

```
def fit(vacantsPath, applicationsPath  ):
    best_candidate = np.zeros(len(applicationsPath))
    for i in range (len(applicationsPath)):
        probabilityFe = chi2_distance(vacantsPath, applicationsPath[i])
        best_candidate[i]+= probabilityFe

    position_candidate = []
    for i in range (0, 5):
        indice_min = int(np.where(best_candidate==min(best_candidate))[0])

        best_candidate = np.delete(best_candidate, indice_min)
        position_candidate.append(indice_min)

    position_candidate.pop(0)
    mejores = []
    for i in position_candidate:
        mejores.append(candidatosTrain[i])
    return mejores
```

Los mejores candidatos para la vacante es:

0	str	1	201
1	str	1	201
2	str	1	243916
3	str	1	102

Figura 3: Mejores candidatos

Para ampliar un poco el funcionamiento de los Autoencoder se ilustra el funcionamiento de las máquinas de Boltzmann, que trabaja con las la asignación de características.

Para eso se sugiere leer el siguiente artículo:

http://proceedings.mlr.press/v5/salakhutdinov09a/salakhutdinov09a.pdf

Ten en cuenta que entre menor energía mayor probabilidad de que suceda un evento. Esto no se trabajó en el proyecto pero se da una idea de cómo lograr predecir nuevos estados.

Reto 2

Para el reto del trm se trabajó con los históricos de precios donde se dividió en tramos de 60 datos para predecir el dato 61 de la misma base de datos. Se utilizó redes neuronales recurrentes. La estructura de esta red es de 5 capas LTMS y una de salida, con esto se pretende crear la memoria en los datos para predecir el trm en el tiempo futuro.

El resultado es el siguiente:

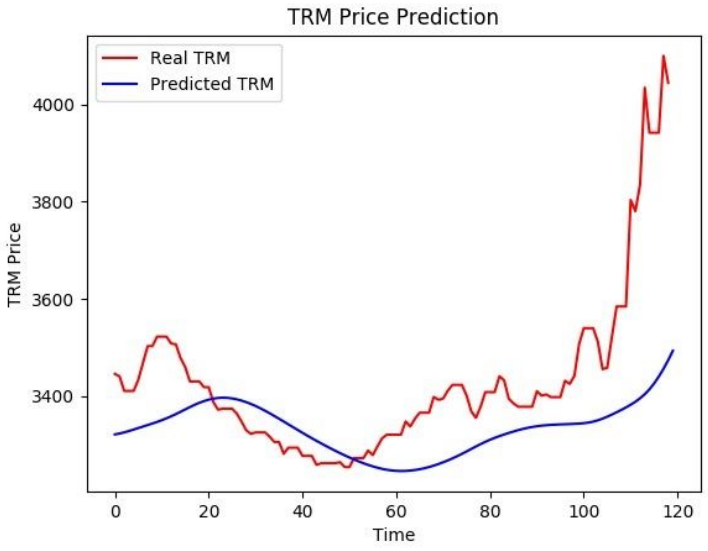


Figura 5: Resultado del trm

Es importante aclarar que este resultado es con una época ya que los recursos computacionales con que cuenta el equipo no cumplen con las restricciones del modelo.

Variar el parámetro epochs = 1 (En la línea 64 del código) para obtener resultados diferentes. Para la prueba se colocar **50**

Para la entrega de los requisitos del reto se ilustra la función utilizada para determinar el valor del trm.

```
def predict_trm(dataset_test):

    dataset_total = pd.concat((dataset_train['TRM'], dataset_test['TRM']), axis = 0)

    inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
    inputs = inputs.reshape(-1,1)
    inputs = sc.transform(inputs)
    X_test = []
    for i in range(60, 180):
        X_test.append(inputs[i-60:i, 0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
    predicted_stock_price = regressor.predict(X_test)
    predicted_stock_price = sc.inverse_transform(predicted_stock_price)

    # Visualising the results
    plt.plot(real_stock_price, color = 'red', label = 'Real TRM')
    plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted TRM')
    plt.title('TRM Price Prediction')
    plt.xlabel('Time')
    plt.ylabel('TRM Price')
    plt.legend()
    plt.show()

    return predicted_stock_price
```

Reto 3

Modelo que permita calcular la probabilidad de rotación de los candidatos
La tercera parte del reto es construir un modelo que calcule la probabilidad de que un candidato rote en la empresa, en menos de 3 meses después de la fecha de ingreso, a partir de los resultados de la pruebas psicométricas.

Para este se emplearon procedimientos similares a los del reto 1, agrupando los resultados de los test en subconjuntos de 3 posiciones cada uno, donde se representa el tipo de pregunta dentro del test, así:

[c1 c2 c3 c4 c5 c6 c7 TT TiempoLaborado]

Donde
c1 hasta c7 son tripletas que denotan el tipo de test (“DP” : “Dentro del perfil”, “PS” : “Preguntas de seguimiento”, “FP” : “Fuera del perfil”), (Ejemplo 26 pts tipo PS se representa [0,26,0])
las letras TT representan el tipo de test al final en una tripleta de unidades binarias [01 01 01] , ejemplo [0,0,1] para una prueba Fuera de perfil.

[0, 0, 48, **27, 0, 0**, 21, 0, 0, 19, 0, 0, 0, 0, 35, 20, 0, 0, 1, 0, 0, **10**]

una persona que laboró 10 meses, el primer test es FP con 48 pts, el segundo **FP con 27pts**…, el resultante es DP (1 0 0)

TiempoLaborado es una variable de tipo int en meses

Posteriormente se hace el análisis de la información con máquinas de boltzmann similarmente al Reto 1, dado que la información se presta para ello.

Conclusión

Esta oportunidad ha servido para mostrar los avances dentro del área de inteligencia artificial al servicio de los sistemas sociales a gran escala, analizando informaciòn para que los reclutadores sean más eficaces durante la selección de personal.