` # Project Report

11.17.2022

Human-Computer Interaction

Walker Daniels - Main Coder & Report

Oscar Salinas - Writer (Report & Slides) & Coder

Robert Taylor - Coder & Report

Zachary Urrutia - Writer (Report & Slides) & Coder

Overview

1. Introduction
2. Provided Data Instance
3. Preparing Data
4. Determining Occurrences
5. Algorithmic Description and Implementation
6. Implementation Results
7. Pros and Con
8. Further Improvements

Introduction

Dr. Huang has tasked our group with completing specific tasks regarding the manipulation of data within a CSV file. As a group, we are to Identify the Gross Pay of an individual on Unigram Language Model by Corpus Cross Entropy, as well as providing additional data, dissecting the Human.csv file into more digestible/readable data. Demonstrating this will give our audience a better understanding of Corpus Cross Entropy in this project. Using our knowledge from Human-Computer Interaction and our coding knowledge as a group, we used the programming language Python to implement all this.

Provided Data Instances

Data instance - Data used in our project

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	Y
2	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K.
3	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K.
4	28	Local-gov	336951	Assoc-acc	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K.
5	44	Private	160323	Some-col	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K.
6	18	?	103497	Some-col	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K.
7	34	Private	198693	10th	6	Never-married	Other-service	Not-in-famil	White	Male	0	0	30	United-States	<=50K.
8	29	?	227026	HS-grad	9	Never-married	?	Unmarried	Black	Male	0	0	40	United-States	<=50K.
9	63	Self-emp-not-inc	104626	Prof-scho	15	Married-civ-spouse	Prof-specialty	Husband	White	Male	3103	0	32	United-States	>50K.
10	24	Private	369667	Some-col	10	Never-married	Other-service	Unmarried	White	Female	0	0	40	United-States	<=50K.
11	55	Private	104996	7th-8th	4	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	10	United-States	<=50K.
12	65	Private	184454	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	6418	0	40	United-States	>50K.
13	36	Federal-gov	212465	Bachelors	13	Married-civ-spouse	Adm-clerical	Husband	White	Male	0	0	40	United-States	<=50K.
14	26	Private	82091	HS-grad	9	Never-married	Adm-clerical	Not-in-famil	White	Female	0	0	39	United-States	<=50K.
15	58	?	299831	HS-grad	9	Married-civ-spouse	?	Husband	White	Male	0	0	35	United-States	<=50K.
16	48	Private	279724	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	3103	0	48	United-States	>50K.
17	43	Private	346189	Masters	14	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	50	United-States	>50K.

The provided data instance is a simple CSV file titled ‘human.csv’ it contains typical information one might expect, including education level, race, sexuality, gender.

Preparing Data

The first task needed to complete this project was to prepare the human.csv dataset. The provided data has a few blank sections in its columns with missing values. These are marked with '?'. To remove them we ran a for loop and wrote into a new csv file titled 'human_edit' for every line that didn't contain a '?'.

CSV Element Replace Code - gets rid of "?" in the data set

```
with open('Human.csv', 'r') as inp, open('Human_edit.csv', 'w') as out:
    writer = csv.writer(out)
    for row in csv.reader(inp):
        if row[1] != " ?":
            writer.writerow(row)
```

The second requirement was to change all of the string info into arbitrary code. This was done by reading our file into an instance variable, using the replace function to switch every unique string with a number, and then mapping them back into the instance.

Arbitrary Code

```
#This is where the occurences for each column are switched to arbitrary values
df = pd.read_csv("Human_edit.csv")
d = {'Private': 0, 'Self-emp-not-inc': 1, 'Local-gov': 2, 'Self-emp-inc': 3, 'Federal-gov': 4, 'Without
df['x2'] = df['x2'].map(d)
d = {'HS-grad': 0, 'Some-college': 1, 'Bachelors': 2, 'Masters': 3, 'Assoc-voc': 4, '11th': 5, 'Assoc-a
df['x4'] = df['x4'].map(d)
d = {'Married-civ-spouse': 0, 'Never-married': 1, 'Divorced': 2, 'Separated': 3, 'Widowed': 4, 'Marriec
df['x6'] = df['x6'].map(d)
d = {'Craft-repair': 0, 'Exec-managerial': 1, 'Prof-specialty': 2, 'Adm-clerical': 3, 'Sales': 4, 'Othe
df['x7'] = df['x7'].map(d)
d = {'Husband': 0, 'Not-in-family': 1, 'Own-child': 2, 'Unmarried': 3, 'Wife': 4, 'Other-relative': 5}
df['x8'] = df['x8'].map(d)
d = {'White': 0, 'Black': 1, 'Asian-Pac-Islander': 2, 'Amer-Indian-Eskimo': 3, 'Other': 4}
df['x9'] = df['x9'].map(d)
d = {'Male': 0, 'Female': 1}
df['x10'] = df['x10'].map(d)
d = {'United-States': 0, 'Mexico': 1, 'Phillipines': 2, 'Puerto-Rico': 3, 'Germany': 4, 'El-Salvador':
df['x14'] = df['x14'].map(d)
d = {'<=50K.': 0, '>50K.': 1}
df['Y'] = df['Y'].map(d)
```

The last requirement needed to prepare the data was to split our csv file into two separate files of 70% and 30%. This was achieved by randomly choosing 70% of a file and then writing that and its composite into two new files.

File Split Code - 70% and 30% respectively

```
#Here is where the edited file is split into two seperate files of seventy and thirty percent using pandas
df = pd.read_csv('Human_edit.csv')
df['split'] = np.random.randn(df.shape[0], 1)

msk = np.random.rand(len(df)) <= 0.7

seventy = df[msk]
thirty = df[~msk]

#And here is where the percentages are written into two new files
seventy.to_csv('seventy.csv', index=False)
thirty.to_csv('thirty.csv', index=False)
```

Determining Occurrences

The second requirement of our project was to determine the number of occurrences for each unique human in every given column. This was achieved with the built-in python function value_count() that reads a given column in a data instance and displays relevant information.

Number of Occurrences Code & Output

```
df['x2'].value_counts()
print("Employment Occurences")
employment = df['x2'].value_counts()
print(employment)

df['x3'].value_counts()
print()
print("Employment Number Occurences")
employmentNum= df['x3'].value_counts()
print(employmentNum)

df['x4'].value_counts()
print()
print("Employment Education Occurences")
employmentEdu = df['x4'].value_counts()
print(employmentEdu)
```

```
Employment Specialty Occurences
 Prof-specialty       1416
 Exec-managerial      1415
 Craft-repair         1414
 Adm-clerical         1307
 Sales                1302
 Other-service        1157
 Machine-op-inspct     727
 Transport-moving      522
 Handlers-cleaners     463
 Tech-support          374
 Farming-fishing       338
 Protective-serv       251
 Priv-house-serv        72
 Armed-Forces            3
 ?                       3
Name: x7, dtype: int64

Employment Filing Occurences
 Husband           4423
 Not-in-family     2870
 Own-child         1539
 Unmarried         1104
 Wife               494
 Other-relative     334
Name: x8, dtype: int64

Employment Race Occurences
 White                 9223
 Black                 1028
 Asian-Pac-Islander     322
 Amer-Indian-Eskimo     110
 Other                   81
Name: x9, dtype: int64

Employment Sex Occurences
 Male      7265
 Female    3499
Name: x10, dtype: int64
```

Algorithmic Description and Implementation

Once the edited file had been split, our first algorithmic implementation was to apply the Maximum Likelihood Estimation to the testing file based around humans sex and annual salary. This was achieved by running the data row by row through a series of if statements. If a particular row met a requirement, the corresponding count would be increased. Once the data instance had finished its run through these statements the counts were then applied to the MLE algorithm as the A and B variables.

```
with open('Seventy.csv', 'r') as infile:
    reader = csv.reader(infile, delimiter=",")
    header = next(reader)
    for row in reader:

        if row[14] != " >50K." :
            numOfLinesLessThan50k+=1
            if row[9] != " Male":
                numOfLinesWithBothF+=1
            if row[9] != " Female":
                numOfLinesWithBothM+=1

        if row[14] !=  " <=50K.":
            numOfLinesMoreThan50k+=1
            if row[9] != " Male":
                numOfLinesWithBothFemale+=1
            if row[9] != " Female":
                numOfLinesWithBothMale+=1
```

MLE Algorithm - Calculation - Using the MLE equation

```
probabilityofAnAttributeSexMale1=(numOfLinesWithBothMale/numOfLinesMoreThan50k)

probabilityofAnAttributeSexFemale1=(numOfLinesWithBothFemale/numOfLinesMoreThan50k)
```

MLE Output

```
70% MLE Calucation of An attribute Gender:

Probability of Sex Male and >50k:
0.6190774630233141
Probability of Sex Female and >50k:
0.3809225369766859
Number of Rows with >50k:
7978
Number of rows with Sex Male and >50k:
4939
Number of rows with Sex Female and >50k:
3039


Probability of Sex Male and <=50k:
0.8555045871559633
Probability of Sex Female and <=50k:
0.1444954128440367
Number of Rows with <=50k:
2616
Number of rows with Sex Male and <=50k:
2238
Number of rows with Sex Female and <=50k:
378
```

The second algorithm required the representation of an entropy calculation. This calculation was performed using pandas, a feature that can be installed in python3. Pandas is used to provide an easy and efficient data structure. In this case the structure stores the occurrences using *value_counts*, however in this case the function calculates entropy using those stored occurences and returns that instead.

```
def pandas_entropy(column, base=None):
    vc = pd.Series(column).value_counts(normalize=True, sort=False)
    base = e if base is None else base
    return -(vc * np.log(vc)/np.log(base)).sum()
```

Entropy Output - 30% dataset

```
Employment Occurences entropy
0.9915940879131591

Employment Number Entropy
8.329027444200655

Employment Education Entropy
2.020793174281795

Employment Education Rank Entropy
2.020793174281795

Employment Marriage Entropy
1.253436112587584

Employment Specialty Entropy
2.3718665931379106

Employment Filing Entropy
1.4890824144682608
```

Implementation Results

The Entropy accuracy was calculated using the *Entropy Output* of the 100% and 30% dataset. The calculation was accomplished by using the entropy output of the 30% dataset and dividing it by the entropy output of the 100% dataset. This is important as it gives the user the accuracy of utilizing a portion of the dataset informing them whether or not the limited information at hand is reliable and efficient.

Entropy Output - 100% dataset

```
----------------------------------------
100% Entropy Calculations:
----------------------------------------


Employment Occurences entropy
1.16329092458208

Employment Number Entropy
9.359221241780364

Employment Education Entropy
2.02999133760112

Employment Education Rank Entropy
2.02999133760112

Employment Marriage Entropy
1.275110231680792

Employment Specialty Entropy
2.448933060512211
```

Entropy Accuracy - Calculated using the 30% dataset

```
-------------------------------------
Entropy Accuracy
-------------------------------------

Employment Occurences Accuracy
0.8437405150958854
Employment Number Accuracy
0.8870849040554625
Employment Education Accuracy
0.9933618591158507
Employment Education Rank Accuracy
0.9933618591158507
Employment Marriage Accuracy
0.9905953157884656
Employment Specialty Accuracy
0.9670320650876086
Employment Filing Accuracy
0.9931122632533457
Employment Race Accuracy
0.9571236756374049
Employment Sex Accuracy
0.9989528217775993
Employment Number 1 Accuracy
0.9759527963408009
```

Pros

- The program runs efficiently. Should in theory work with any given csv file.
- Information displayed is correct.

Cons

- Information is not displayed in a particularly aesthetic manner.
- Furthermore it can be quite difficult to find the particular information you are looking for once it has been displayed in a terminal.

Further Improvement

- Visual output of the information. The product could be neater and more visually appealing to the audience.
- An interactive UI So the information isn't displayed all at once, this ties in with visual output.
- Design an instruction manual or guide for this program, for the user.