In [472]:
```
import numpy as np
import csv
```

In [473]:
```
#Strategy: create company object and predict valuation based on similarity
```

In [ ]:
```
```

In [474]:
```
#parse: retrieve values from each row for instantiation in company class
def parse(row):
    if row[0] != "":
        valuation = int(row[0])
    else:
        valuation = 0

    name = row[1]

    if row[4] != "":
        growth = int(row[4])
    else:
        growth = 0

    if row[5] != "":
        mindshare = int(row[5])
    else:
        mindshare = 0

    if row[6] != "":
        employee_count = int(row[6])
    else:
        employee_count = 0

    employee_bucket = ""
    if employee_count < 100:
        employee_bucket = "Small"
    elif employee_count < 250:
        employee_bucket = "SMB"
    elif employee_count < 500:
        employee_bucket = "Medium"
    else:
        employee_bucket = "Large"

    if row[7] != "":
        monthly_uniques = int(row[7])
    else:
        monthly_uniques = 0


    if row[8] != "":
        m_last_funding = int(row[8])
    else:
        m_last_funding = 0

    if row[9] != "":
        founded = int(row[9])
    else:
        founded = 0

    stage = row[10]

    investors_array = row[11].split("|")
    investor_count = len(investors_array)
    if row[12] != "":
        total_funding = int(row[12])
    else:
        total_funding = 0

    #omit column 13 - funding date - irrelevant to calculations
    if row[14] != '':
        last_funding_amt = row[14]
    else:
        last_funding_amt = 0

    location = row[15]
    revenue_range = row[18]
    #for estimation, assign revenue ranges to midpoint of range
    if revenue_range == "Less than $500K":
        revenue = 250000
    elif revenue_range == "$500K - $1M":
        revenue = 750000
    elif revenue_range == "$1M - $5M":
        revenue = 3500000
    elif revenue_range == "$5M - $10M":
        revenue = 7500000
    elif revenue_range == "$10M - $25M":
        revenue = 17500000
    elif revenue_range == "$25M - $50M":
        revenue = 37500000
    elif revenue_range == "$50M - $100M":
        revenue = 75000000
    elif revenue_range == "$100M - $250M":
        revenue = 175000000
    elif revenue_range == "$250M - $500M":
        revenue = 375000000
    elif revenue_range == "$500M - $1B":
        revenue = 750000000
    elif revenue_range == "$1B - $5B":
        revenue = 3750000000
    elif revenue_range == "Greater than $5B":
        revenue = 7500000000
    else:
        revenue = 0

    business_model = row[19]
    industries = row[20].split("|")
    return [valuation, name, growth, mindshare, employee_bucket, monthly_u
```

In [475]:
```
class Company:
    def __init__(self, valuation, name, growth, mindshare, employee_bucket
        self.valuation = valuation
        self.name = name
        self.growth = growth
        self.mindshare = mindshare
        self.employee_bucket = employee_bucket
        self.monthly_uniques = monthly_uniques
        self.m_last_funding = m_last_funding
        self.founded = founded
        self.stage = stage
        self.investor_count = investor_count
        self.total_funding = total_funding
        self.last_funding_amt = last_funding_amt
        self.location = location
        self.revenue = revenue
        self.business_model = business_model
        self.industries = industries

    def similarity_score(self, other):
        #give factors that are more important more weight in determining s
        similarity = 0
        if abs(self.growth - other.growth) < 100:
            similarity += 1
        if abs(self.mindshare - other.mindshare) < 100:
            similarity += 1
        if abs(self.employee_bucket == other.employee_bucket):
            similarity += 1
        if abs(self.monthly_uniques - other.monthly_uniques) < 100:
            similarity += 2.5
        if abs(self.m_last_funding - other.m_last_funding) < 5:
            similarity += 1
        if abs(self.founded - other.founded) <= 1:
            similarity += 1
        if self.stage == other.stage:
            similarity += 7.5
        if abs(self.investor_count - other.investor_count) < 5:
            similarity += 1
        if abs(self.total_funding - other.total_funding) < 100000000:
            similarity += 1
        #if self.location == company.location:
            #similarity += 1
        if self.revenue == other.revenue:
            similarity += 5
        if self.business_model == other.business_model:
            similarity += 1
        #similar_industries = [industry for industry in self if industry i
        similarity += len(intersect(self.industries, other.industries))
        return similarity

    def predict_valuation(self, other):
        #Combine valuation projections from different factors and weight e
        total_valuation = 0
        total_weight_count = 0
        #growth-based valuation
        if self.growth !=0 and other.growth !=0:
            total_valuation += self.growth * (other.valuation/other.growth
            total_weight_count += 1
        if self.revenue != 0 and other.revenue != 0:
            total_valuation += self.revenue * (other.valuation/other.reven
            total_weight_count += 5
        if self.monthly_uniques != 0 and other.monthly_uniques:
            total_valuation += self.monthly_uniques * (other.valuation/oth
            total_weight_count += 1
        if self.total_funding != 0 and other.total_funding != 0:
            total_valuation += self.total_funding * (other.valuation/other
            total_weight_count += 2
        if total_weight_count > 0:
            return round(total_valuation/total_weight_count)
        return 0
```

In [476]:

```
def overlapping_industries(companyA, companyB):
    ind_arr = [industry for industry in companyA if industry in companyB]
    return ind_arr
```

In [477]:

```
for_parsing = open('InternData_reorg.csv', encoding="ISO-8859-1")
data = csv.reader(for_parsing)
given_valuations = {}
predicted_valuations = {}

num_row = 0
for row in data:
    if num_row > 0 and num_row < 20:
        given_valuations[row[1]] = Company(*parse(row))
    if num_row >= 20:
        predicted_valuations[row[1]] = Company(*parse(row))
    num_row += 1
```

In [ ]:

In [478]:

```
#Find most similar company
for company in predicted_valuations:
    most_similar = ''
    max_similarity_score = 0
    for c in given_valuations:
        s_score = predicted_valuations[company].similarity_score(given_val
        if s_score > max_similarity_score:
            max_similarity_score = s_score
            most_similar = c
    predicted_valuations[company].valuation = predicted_valuations[company
```

In [479]:

```
def convert_to_string(valuation):
    return "$"+ str(valuation) + "MM"
```

In [481]:
```
#print(predicted_valuations.keys())
#print(given_valuations.keys())
#print(given_valuations.keys())

print(predicted_valuations["PayTM"].valuation)

for_parsing = open('InternData_reorg.csv', encoding="ISO-8859-1")
data = csv.reader(for_parsing)

with open("Data_with_Valuations.csv", mode="w") as csvfile:
    datawvaluations = csv.writer(csvfile)
    i = 0
    for row in data:
        if i < 20:
            datawvaluations.writerow(row)
        else:
            row[0] = str(predicted_valuations[row[1]].valuation)
            datawvaluations.writerow(row)
        i += 1

csvfile.close()
```

```
1123
```

In [ ]:

In [ ]: