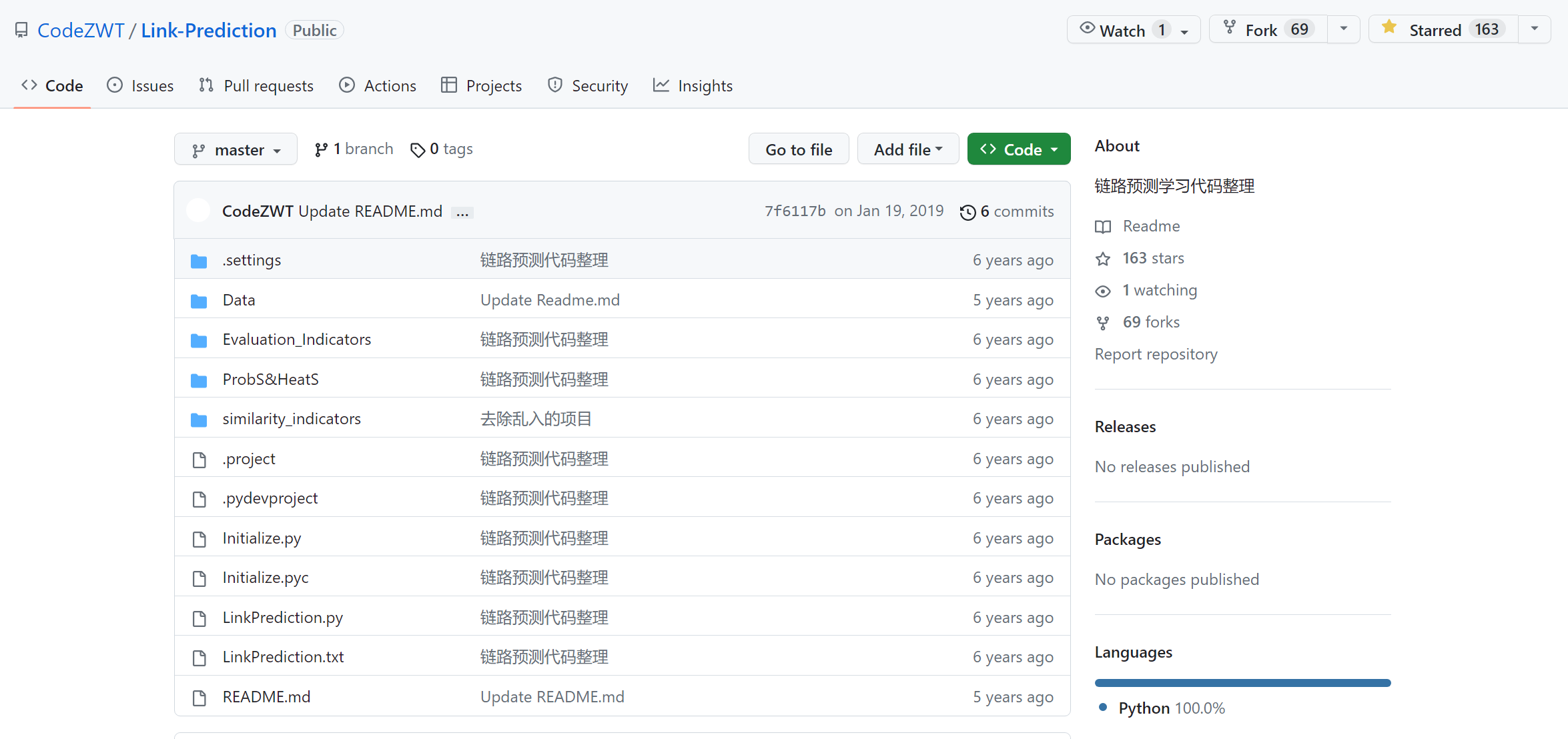
# A02014087 张文滔 复杂网络作业五

对Email数据集，用Katz, CN, AA, RA, LP, ACT，SimRank算法进行链路预测。设置训练集比例为90%，并比较AUC值和Precision (L=100,200)。

## 代码：

借鉴了<https://github.com/CodeZWT/Link-Prediction>的代码



## 主函数：LinkPrediction.py

import time

import os

import Initialize

import Evaluation\_Indicators.AUC

import similarity\_indicators.CommonNeighbor

import similarity\_indicators.AA

import similarity\_indicators.RA

import similarity\_indicators.LP

import similarity\_indicators.Katz

import similarity\_indicators.ACT

startTime = time.time()

NetFile = 'Data/Email.txt'

NetName = 'Email'

print("\nLink Prediction start:\n")

TrainFile\_Path = 'Data\\'+NetName+'\\Train.txt'

if os.path.exists(TrainFile\_Path):

    Train\_File = 'Data\\'+NetName+'\\Train.txt'

    Test\_File = 'Data\\'+NetName+'\\Test.txt'

    MatrixAdjacency\_Train, MatrixAdjacency\_Test, MaxNodeNum = Initialize.Init2(Test\_File, Train\_File)

else:

    MatrixAdjacency\_Net, MaxNodeNum = Initialize.Init(NetFile)

    MatrixAdjacency\_Train, MatrixAdjacency\_Test = Initialize.Divide(NetFile, MatrixAdjacency\_Net, MaxNodeNum, NetName)

similarity\_StartTime = time.time()

for Method in range(6):

    if Method == 0:

        print('----------SIM----------计算相似性矩阵1 7----------SIM----------')

        print('----------Cn----------')

        Matrix\_similarity = similarity\_indicators.CommonNeighbor.Cn(MatrixAdjacency\_Train)

        Evaluation\_Indicators.AUC.Calculation\_AUC(MatrixAdjacency\_Train, MatrixAdjacency\_Test, Matrix\_similarity, MaxNodeNum)

    elif Method == 1:

        print('----------RA----------')

        Matrix\_similarity = similarity\_indicators.RA.RA(MatrixAdjacency\_Train)

        Evaluation\_Indicators.AUC.Calculation\_AUC(MatrixAdjacency\_Train, MatrixAdjacency\_Test, Matrix\_similarity, MaxNodeNum)

    elif Method == 2:

        print('----------Katz----------')

        Matrix\_similarity = similarity\_indicators.Katz.Katz(MatrixAdjacency\_Train)

        Evaluation\_Indicators.AUC.Calculation\_AUC(MatrixAdjacency\_Train, MatrixAdjacency\_Test, Matrix\_similarity, MaxNodeNum)

    elif Method == 3:

        print('----------SIM----------基于随机游走----------SIM----------')

        print('----------ACT----------')

        Matrix\_similarity = similarity\_indicators.ACT.ACT(MatrixAdjacency\_Train)

        Evaluation\_Indicators.AUC.Calculation\_AUC(MatrixAdjacency\_Train, MatrixAdjacency\_Test, Matrix\_similarity, MaxNodeNum)

    elif Method == 4:

        print('----------AA----------')

        Matrix\_similarity = similarity\_indicators.AA.AA(MatrixAdjacency\_Train)

        Evaluation\_Indicators.AUC.Calculation\_AUC(MatrixAdjacency\_Train, MatrixAdjacency\_Test, Matrix\_similarity, MaxNodeNum)

    elif Method == 5:

        print('----------SIM----------基于路径----------SIM----------')

        print('----------LP----------')

        Matrix\_similarity = similarity\_indicators.LP.LP(MatrixAdjacency\_Train)

        Evaluation\_Indicators.AUC.Calculation\_AUC(MatrixAdjacency\_Train, MatrixAdjacency\_Test, Matrix\_similarity, MaxNodeNum)

    else:

        print("Method Error!")

similarity\_EndTime = time.time()

print('----------汇总----------')

print("All SimilarityTime: {} s".format(similarity\_EndTime - similarity\_StartTime))

# Calculate AUC

Evaluation\_Indicators.AUC.Calculation\_AUC(MatrixAdjacency\_Train, MatrixAdjacency\_Test, Matrix\_similarity, MaxNodeNum)

endTime = time.time()

print(f"\nRunTime: {endTime - startTime} s")

## Initialize.py

import numpy as np

import time

def Data\_Shape(Data):

    List\_A = []

    List\_B = []

    for row in range(Data.shape[0]):

        List\_A.append(Data[row][0])

        List\_B.append(Data[row][1])

    List\_A = list(set(List\_A))

    List\_B = list(set(List\_B))

    length\_A = len(List\_A)

    length\_B = len(List\_B)

    print(f'    数据集长度：{str(Data.shape[0])}')

    print(f'    第一列节点长度：({str(length\_A)})')

    print(f'    第二列节点长度：({str(length\_B)})')

    MaxNodeNum =  int(max(max(List\_A),max(List\_B))) + 1

    print(f'    节点数量为：{str(MaxNodeNum)}')

    return MaxNodeNum

def MatrixAdjacency(MaxNodeNum,Data):

    MatrixAdjacency = np.zeros([MaxNodeNum,MaxNodeNum])

    for col in range(Data.shape[0]):

        i = int(Data[col][0])

        j = int(Data[col][1])

        MatrixAdjacency[i,j] = 1

        MatrixAdjacency[j,i] = 1

    return MatrixAdjacency

def spones(Array):

    for index in range(len(Array)):

        if Array[index] != 0:

            Array[index] = 1

    return Array

def writeTrainFile(Matrix,NetName):

    Matrix = np.triu(Matrix)

    index = np.argwhere(Matrix != 0)

    with open(f'Data\\{NetName}\\Train.txt','w') as file:

        np.savetxt(file,index,fmt='%d')

def writeTestFile(Matrix,NetName):

    Matrix = np.triu(Matrix)

    index = np.argwhere(Matrix != 0)

    with open(f'Data\\{NetName}\\Test.txt','w') as file:

        np.savetxt(file,index,fmt='%d')

def Init(NetFile):

    print("DataShape......")

    NetData = np.loadtxt(NetFile)

    MaxNodeNum = Data\_Shape(NetData)

    MatrixAdjacency\_Net = MatrixAdjacency(MaxNodeNum, NetData)

    return MatrixAdjacency\_Net,MaxNodeNum

def Divide(NetFile,MatrixAdjacency\_Net,MaxNodeNum,NetName):

    print("Divide......")

    DivideTime\_Start = time.process\_time()

    DivideNum = 0.9

    NetData = np.loadtxt(NetFile)

    lengthData = len(NetData)

    NumSet = NetData.shape[0]

    NumTest = int(float(1-DivideNum)\*NumSet)

    MatrixAdjacency\_Test =np.zeros([MaxNodeNum,MaxNodeNum])

    while(len(np.nonzero(MatrixAdjacency\_Test)[0]) < NumTest):

        index\_Link = int(np.random.rand(1) \* NetData.shape[0])

        Uid1 = int(NetData[index\_Link,0])

        Uid2 = int(NetData[index\_Link,1])

        MatrixAdjacency\_Net[Uid1,Uid2] = 0

        MatrixAdjacency\_Net[Uid2,Uid1] = 0

        tempVector = MatrixAdjacency\_Net[Uid1]

        sign = 0

        Uid1\_TO\_Uid2 = np.dot(tempVector,MatrixAdjacency\_Net) + tempVector

        if Uid1\_TO\_Uid2[Uid2] > 0:

            sign = 1

        else:

            count = 1

            while (len((spones(Uid1\_TO\_Uid2) - tempVector).nonzero()[0]) != 0):

                tempVector  = spones(Uid1\_TO\_Uid2)

                Uid1\_TO\_Uid2 = np.dot(tempVector,MatrixAdjacency\_Net) + tempVector

                count += 1

                if Uid1\_TO\_Uid2[Uid2] > 0:

                    sign = 1

                    break

                if count >= MatrixAdjacency\_Net.shape[0]:

                    print(f'不可达：{str([Uid1,Uid2])}')

                    sign = 0

        if sign == 1:

            NetData = np.delete(NetData,index\_Link,axis=0)

            MatrixAdjacency\_Test[Uid1,Uid2] = 1

        else:

            NetData = np.delete(NetData,index\_Link,axis=0)

            MatrixAdjacency\_Net[Uid1,Uid2] = 1

            MatrixAdjacency\_Net[Uid2,Uid1] = 1

    MatrixAdjacency\_Train = MatrixAdjacency\_Net

    MatrixAdjacency\_Test = MatrixAdjacency\_Test + MatrixAdjacency\_Test.T

    print(f'    训练集邻接矩阵：{str(MatrixAdjacency\_Train.shape)}')

    print(f'    训练集计划边数：{str(lengthData - NumTest)}')

    print(f'    训练集实际边数：{str((np.nonzero(MatrixAdjacency\_Train)[0].shape[0])//2)}')

    print(f'    测试集邻接矩阵：{str(MatrixAdjacency\_Test.shape)}')

    print(f'    测试集计划边数：{str(NumTest)}')

    print(f'    测试集实际边数：{str((np.nonzero(MatrixAdjacency\_Test)[0].shape[0])//2)}')

    writeTrainFile(MatrixAdjacency\_Train, NetName)

    writeTestFile(MatrixAdjacency\_Test, NetName)

    DivideTime\_End = time.process\_time()

    print(f'DivideTime：{(DivideTime\_End - DivideTime\_Start)} s')

    return MatrixAdjacency\_Train,MatrixAdjacency\_Test

def Init2(Test\_File,Train\_File):

    print("DataShape......")

    TrainData = np.loadtxt(Train\_File)

    MaxNodeNumTrain = Data\_Shape(TrainData)

    TestData = np.loadtxt(Test\_File)

    MaxNodeNumTest = Data\_Shape(TestData)

    MaxNodeNum = max(MaxNodeNumTrain,MaxNodeNumTest)

    MatrixAdjacency\_Train = MatrixAdjacency(MaxNodeNum, TrainData)

    MatrixAdjacency\_Test = MatrixAdjacency(MaxNodeNum, TestData)

    return MatrixAdjacency\_Train,MatrixAdjacency\_Test,MaxNodeNum

## 运行结果：

Link Prediction start:

DataShape......

数据集长度：4906

第一列节点长度：(750)

第二列节点长度：(1123)

节点数量为：1133

数据集长度：545

第一列节点长度：(314)

第二列节点长度：(373)

节点数量为：1105

----------ACT----------

----------Cn----------

SimilarityTime: 0.056853600006434135 s

Calculation AUC......

AUC指标为：0.8854290600832838

AUCTime：1.2904255390167236 s

precision(L=100):0.32

precision(L=200):0.28

----------RA----------

SimilarityTime: 0.109375 s

Calculation AUC......

AUC指标为：0.8874189470553242

AUCTime：1.1307573318481445 s

precision(L=100):0.33

precision(L=200):0.22

----------Katz----------

SimilarityTime: 0.093750 s

Calculation AUC......

AUC指标为：0.9531751933372993

AUCTime：1.126634120941162 s

precision(L=100):0.31

precision(L=200):0.275

----------ACT----------

Calculation AUC......

AUC指标为：0.8350089232599643

AUCTime：1.173448085784912 s

precision(L=100):0.06

precision(L=200):0.055

----------AA----------

SimilarityTime: 0.250000 s

Calculation AUC......

AUC指标为：0.8881246281975015

AUCTime：1.1335554122924805 s

precision(L=100):0.39

precision(L=200):0.275

----------LP----------

SimilarityTime: 0.265625 s

Calculation AUC......

AUC指标为：0.8981640392623439

AUCTime：1.1891725063323975 s

precision(L=100):0.11

precision(L=200):0.11

----------汇总----------

All SimilarityTime: 9.964800119400024 s

RunTime: 9.99631929397583 s

## 整理后的结果：

AUC保留小数点后三位

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 方法  指标 | | Katz(=0.01) | CN | AA | RA | LP(=0.01) | ACT |
| AUC | | 0.953 | 0.885 | 0.888 | 0.887 | 0.898 | 0.835 |
| Precision | L=100 | 0.31 | 0.32 | 0.39 | 0.33 | 0.11 | 0.06 |
| L=200 | 0.275 | 0.28 | 0.275 | 0.22 | 0.11 | 0.055 |