Lab Report

Performance (on the validation dataset)

PART 1

zid	MSR	correlation
1111111	9760285857759024.0	0.0157004189441393

PART 2

zid	average_precision	average_recall	accuracy
1111111	0.7015088129114462	0.8628158844765343	0.6525

JSON formated columns

我使用了Python的json.loads包对于JSON格式的数据进行处理，对于带有JSON格式的数据(例如cast，crew，genres，keywords，production_companies，production_countries)，经过观察后，不妨令此类第i个特征对应数值的取值集合为$D_i = \{d_0, d_1, \dots, d_m\}$，第$j$个数据在第$i$个特征上的取值集合为$D_i^{(j)}$，则有：

$$\begin{aligned} D_i^{(j)} &\subseteq D_i \\ \bigcup_{j=0}^{n} D_i^{(j)} &= D_i \end{aligned} \tag{1}$$

考虑到上述性质，我们可以对特征的取值d_k进行`one-hot`编码，举个例子：

$$\begin{aligned} D_i^{(j)} &= \{d_2, d_3, \dots, d_6\},\ m = 9 \\ &\Rightarrow\ [\,0, 0, 1, 1, 0, 0, 1, 0, 0\,] \end{aligned} \tag{2}$$

注意测试集和验证集要满足one-hot编码下标一致，需要进行预处理.

封装成了函数，代码如下：

```python
def Preprocess(dfList, dfIdx, elementKey):
    global jsonMap
    curDict, idxDict = {}, 0
    for df in dfList:
        for i, v in enumerate(df[dfIdx]):
            curList = json.loads(df[dfIdx][i])
            for k in curList:
                if k[elementKey] not in curDict.keys():
                    curDict[k[elementKey]] = idxDict
                    idxDict += 1
    jsonMap[dfIdx] = curDict

def ParseJson(df, dfIdx, elementKey):
    global jsonMap
    retNp = []
    curDict, idxDict = {}, 0

    if dfIdx not in jsonMap.keys():
        print("ERROR")
    else:
        idxDict = len(jsonMap[dfIdx])
        curDict = jsonMap[dfIdx]

    for i, v in enumerate(df[dfIdx]):
        curList = json.loads(df[dfIdx][i])
        posList = []
        for k in curList:
            posList.append(curDict[k[elementKey]])
        curVec = GenetateOneHotVec(posList, idxDict)
        retNp.append(curVec)

    retNp = np.array(retNp)
    return retNp
```

对于`tagline`，`original_title`等于NLP相关的特征处理...