大连理工大学

信息检索实验报告

项目名称：SMP 微博用户画像

专　　业：计算机科学与技术

班　　级：电计 1504 班

学　　号：201546022

学生姓名：王 玥

联系方式：15524666836

时　　间：2017-12-25

目录

一、 实验内容

用户画像（user profiling）是指对用户的人口统计学特征、行为模式、偏好、观点、目标等进行标签化，是互联网时代实现精准化服务、营销和推荐的必经之路，在网络安全、管理和营运等领域具有重要意义。微博用户画像是指利用微博用户的内容信息（如发表的微博和评论）、行为记录（如浏览、转发、点赞、收藏等）和链接结构（如用户之间的粉丝关系）等，对用户的不同维度进行画像，对完善及扩充微博用户信息、分析微博生态以及支撑微博业务等方面具有非常重要的意义。

参赛队伍利用给定的新浪微博数据（包括用户个人信息、用户微博文本以及用户粉丝列表，详见数据描述部分），进行微博用户画像，具体包括以下三个任务：

任务 1：推断用户的年龄（共 3 个标签：-1979/1980-1989/1990+）

任务 2：推断用户的性别（共 2 个标签：男/女）

任务 3：推断用户的地域（共 8 个标签：东北/华北/华中/华东/西北/西南/华南/境外）

二、 技术路线

1. 数据预处理

对每个人的微博进行合并处理。

根据原始数据合并数据集。

2.特征工程：

1）提取用户的姓名长度，缺失值数量，粉丝数量，微博数量，平均每条微博的评论/转发人数，以及它们的排序特征，最大/最小的评论/转发数，姓名/图片/粉丝/转发/评论是否缺失，搭建省份-地区映射表，对博文中出现的省份进行匹配，并映射成地区，统计博文中出现的地区数量。

2）根据合并的数据集和高频词生成数据特征文件，提取博文的 tfidf 特征并训练 xgboost 进行分类预测。

3. 模型：

主要采用 gbdt 模型（xgboost）。又对性别预测使用二分类，对地点预测使用 softmax 多分类。

模型融合：单独提取文本的 tfidf 特征，并训练 xgboost 进行初次预测，将得到的预测结果作为新特征(stacking 融合)，和其他特征共同训练 xgboost 并进行最终的分类预测，最终合并 3 个分类结果构成最终的提交结果。

三、 主要代码

（完整代码已由电子版报告附件提交）

1. 根据原始数据合并数据集

```
f_I_part_1 = pd.merge(df_part_1_i_b_count,
                      df_part_1_i_b_b4_time,
                      on = ['item_id'], how = 'left')
f_I_part_1 = pd.merge(f_I_part_1,
                      df_part_1_i_u_count,
                      on = ['item_id'], how = 'left')[['item_id',
                                                       'i_u_count_in_6',
                                                       'i_u_count_in_3',
                                                       'i_u_count_in_1',
                                                       'i_b1_count_in_6',
                                                       'i_b2_count_in_6',
                                                       'i_b3_count_in_6',
                                                       'i_b4_count_in_6',
                                                       'i_b_count_in_6',
                                                       'i_b1_count_in_3',
                                                       'i_b2_count_in_3',
                                                       'i_b3_count_in_3',
                                                       'i_b4_count_in_3',
                                                       'i_b_count_in_3',
                                                       'i_b1_count_in_1',
                                                       'i_b2_count_in_1',
                                                       'i_b3_count_in_1',
                                                       'i_b4_count_in_1',
                                                       'i_b_count_in_1',
                                                       'i_b4_rate',
                                                       'i_b4_diff_hours']]
```

2. 根据合并的数据集和高频词生成数据特征文件

```
# ---area feature---
area = ['东北','华北','华中','华东','西北','西南','华南','境外']

loc  = defaultdict(list)
loc[0] = ['东北','辽宁','吉林','黑龙江','长白山','北戴河','松花湖','老虎滩','锡林郭勒草原','沈阳故宫',
          '沈阳','大连','鞍山','抚顺','本溪','丹东','锦州','营口','阜新','辽阳','盘锦','铁岭','朝阳','葫芦岛','兴城','海城',
          '长春','四平','松原','白城','辽源','通化','延边','梅河口','公主岭','长影世纪城','净月潭','伪满','查干湖','松花湖',
          '清昭陵','清福陵','星海广场','千山','鸭绿江','本溪水洞' ,'甘井子','大工','东财','辽师','吉大',
          '太阳岛','亚布力','中央大街','五大连池','扎龙湿地','镜泊湖', '哈工大','哈工程','黑大','哈医大',
          '哈尔滨', '齐齐哈尔','鸡西','鹤岗','双鸭山','大庆','伊春','佳木斯','七台河','牡丹江','黑河','绥化','大兴安岭']

loc[1] = [ '华北','北京','天津','河北','山西','内蒙古','呼和浩特','包头','鄂尔多斯','乌兰察布','冀','黄土高原',
          '故宫','颐和园','五台山','平遥古城','云冈石窟','野三坡','白洋淀','晋','北大','清华','南开','天大',
          '保定','唐山','太原','大慈阁','北戴河','清东陵','白洋淀','避暑山庄','西柏坡','山海关',
         '石家庄','辛集','藁城','晋州','新乐','鹿泉','遵化','迁安','秦皇岛','邯郸','武安','邢台','南宫','沙河','涿州','定州',
          '安国','高碑店','张家口','承德','沧州','泊头','任丘','黄骅','河间','廊坊','霸州','三河','衡水','冀州','深州']

loc[2] = ['华中','河南','湖北','湖南','中原','少林寺','龙门石窟','嵩山','黄鹤楼','神农架','张家界','衡山',
          '郑州','殷墟','天地之中','黄帝故里','白马寺',
          '开封','洛阳','南阳','漯河','许昌','三门峡','平顶山','周口','驻马店','新乡','鹤壁','焦作','濮阳',
          '安阳','商丘','信阳','济源',
         '武汉','黄石','襄樊','十堰','荆州','宜昌','荆门','鄂州','孝感','黄冈','咸宁','随州','恩施',
          '仙桃','潜江','天门','神农架','武昌','东湖','武当山','三峡','隆中','汉口',
         '张家界','岳麓山','岳阳楼','炎帝陵','凤凰古城','韶山','衡山','东江湖',
          '长沙','湘潭','株洲','郴州','衡阳','邵阳','永州','湘西','常德','娄底','益阳','怀化']
```

3. 提取博文的 tfidf 特征并训练 xgboost 进行分类预测

```
df_mge['loc_bin'] = df_mge['loc'].map(bin_loc)
df_mge['age_bin'] = df_mge['age'].map(bin_age)

age_le = LabelEncoder()
y_age = age_le.fit_transform(df_mge.iloc[:3200]['age_bin'])

loc_le = LabelEncoder()
y_loc = loc_le.fit_transform(df_mge.iloc[:3200]['loc_bin'])

sex_le = LabelEncoder()
y_sex = sex_le.fit_transform(df_mge.iloc[:3200]['sex'])

tokenizer = lambda s:s.split(' ')
tfv = TfidfVectorizer(tokenizer=tokenizer,min_df=3,
                      norm='l2',use_idf=True,sublinear_tf=True)
TR = 3200
TE = 1240
X_all_sp = tfv.fit_transform(df_mge.bag_twts)
X_all_sp = X_all_sp.tocsc()
X_sp = X_all_sp[:TR]

prds = []
stacks = []
stacks_name = []
task = ['sub']
print(task)
```

```
                    early_stopping_rounds=25, verbose_eval=20)
if 'sub' in task:
    n_iter = 680
    dtrain = xgb.DMatrix(X_all_sp[:TR], y)
    dtest = xgb.DMatrix(X_all_sp[TR:])

    watchlist = [(dtrain, 'train')]
    bst = xgb.train(params, dtrain, n_iter, evals=watchlist,
                    early_stopping_rounds=25, verbose_eval=100)

    prds.append(bst.predict(dtest))
if 'stack' in task:
    n_iter = 680
    n = 5
    num_class = 3
    stack_tr = np.zeros((TR,num_class))
    stack_te = np.zeros((TE,num_class))
    dtest = xgb.DMatrix(X_all_sp[TR:])
    for i,(tr,va) in enumerate(StratifiedKFold(y,n_folds=n)):
        print('stack:%d/%d'%(i+1,n))
        dtr = xgb.DMatrix(X_sp[tr],y[tr])
        dva = xgb.DMatrix(X_sp[va],y[va])
        bst = xgb.train(params, dtr, n_iter)
        stack_tr[va] = bst.predict(dva)
        stack_te += bst.predict(dtest)
    stack_te /= n
    stack = np.vstack([stack_tr,stack_te])
    stacks.append(stack)
    stacks_name += ['%s_%d'%(label,i) for i in range(num_class)]

```

4. 对 location 的预测

```
def num_missing(x):
    return sum(x.isnull())

merge_data['num_missing'] = merge_data.apply(num_missing, axis=1)

#rank特征
merge_data['rank_sum_content'] = merge_data['sum_content'].rank(method='max')
merge_data['rank_sum_fans'] = merge_data['sum_fans'].rank(method='max')
merge_data['rank_mean_retweet'] = merge_data['mean_retweet'].rank(method='max')
merge_data['rank_mean_review'] = merge_data['mean_review'].rank(method='max')
merge_data['rank_num_missing'] = merge_data['num_missing'].rank(method='max')


# In[134]:

#导入使用tfidf特征训练的模型的预测结果（采用stacking融合，把预测结果作为新特征加进模型）
tfidf_stacking = pd.read_csv('../data/newfeat/stack_new.csv',encoding='UTF-8')
merge_data = pd.concat([merge_data,tfidf_stacking],axis=1)

feat_time_3hour = pd.read_csv('../data/newfeat/feat_time_3hour.csv',encoding='UTF-8')
merge_data = pd.merge(merge_data,feat_time_3hour,on='uid',how='left')

#导入使用word2vec特征训练的模型的预测结果
'''
w2v_stacking = pd.read_csv('./data/newfeat/w2v_prob1.csv')
merge_data = pd.merge(merge_data,w2v_stacking,on='uid',how='left')
'''

'''newmerge_feat1 = pd.read_csv('../data/newfeat/newmerge_feat.csv',encoding='UTF-8')
merge_data = pd.merge(merge_data,newmerge_feat1,on='uid',how='left')'''

```

```
# In[424]:

params = {
    "objective": "multi:softprob",
    "booster": "gbtree",
    "eval_metric": "merror",
    "num_class":8,
    'max_depth':4,
    #'min_child_weight':2.5,
    'subsample':0.65,
    'colsample_bytree':1.0,
    'gamma':2.5,
    "eta": 0.006,
    #"lambda":1,
    #'alpha':0,
    "silent": 1,
    #'seed':1123
}
xgb1=xgb.train(params,dtrain,num_boost_round=25)


# In[425]:

pre=xgb1.predict(dtest,ntree_limit=25)
pre_loc=[loc_le.classes_[idx] for idx in pre.argmax(1)]
sub = pd.DataFrame()
sub['uid'] = merge_data.iloc[TR:]['uid']
sub['province'] = pre_loc
sub.to_csv('../data/location_sub.csv',index=False)

# In[426]:
```

5. 对 sex 的预测

```
                'mean_retweet','sum_content','mean_review','num_missing',
                  'w2v_f_prob','w2v_m_prob','w2v_young_prob','w2v_old_prob','w2v_mid_prob',
                  'max_retweet','min_retweet','max_review','min_review',
                  'rank_sum_content','rank_sum_fans','rank_mean_retweet','rank_mean_review','rank_num_missing',
                  'timePeriod_3hour_0','timePeriod_3hour_1','timePeriod_3hour_2','timePeriod_3hour_3',
                  'timePeriod_3hour_4','timePeriod_3hour_5','timePeriod_3hour_6','timePeriod_3hour_7',
                  'name_isnull','image_isnull','fans_isnull','retweet_isnull','review_isnull',
                  'area_0','area_1','area_2','area_3','area_4','area_5','area_6','area_7'
                  ])
cols = [c for c in merge_data.columns if re.match(cols,c)]

age_le = LabelEncoder()
ys = {}
ys['age'] = age_le.fit_transform(merge_data.iloc[:3200]['age2'])

loc_le = LabelEncoder()
ys['loc'] = loc_le.fit_transform(merge_data.iloc[:3200]['location2'])

sex_le = LabelEncoder()
ys['sex'] = sex_le.fit_transform(merge_data.iloc[:3200]['sex'])


merge_data = merge_data.fillna(0)
task = ['sub']


TR = 3200
TE = 1240
X_all = merge_data[cols]
X = X_all[:TR]
prds = []
```

四、 心得体会

通过本次 SMP CUP 微博用户画像任务，我加深对机器学习相关知识的理解，积累了实践经验。并且对 xgboost，tfidf 等方法有了深入学习和理解。

我通过抽取出 TFIDF 特征向量，并尝试使用逻辑回归模型，xgboost 进行分类训练模型融合。经过比较测试，最后采用单独提取文本的 tfidf 特征，并训练 xgboost 进行初次预测，将得到的预测结果作为新特征(stacking 融合)，并加入其他特征共同训练 xgboost 并进行最终的分类预测。

但由于时间关系，并未成功采用其他辅助方法，例如 PCA 降维、SVD 降维、t-SNE 降维可视化、word2vec 词向量训练尝试加入特征工程，但还未成功，有待进一步的完善。

虽然报告已提交，但仍待后期的进一步完善特征工程和模型测试。虽然这门课程已经结束，但我通过这门课初识用户画像，并对其产生了很强的兴趣，学习的步伐不会停止，我仍会继续学习机器学习等领域知识，不断完善我的用户画像，期待日后我对用户画像的深入理解。