# 运行环境

Win7 64位

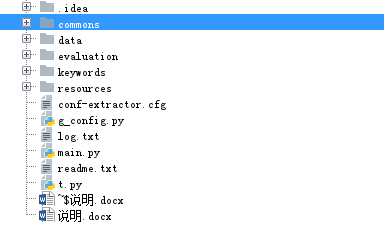
Anaconda pyton3.6

Pycharm

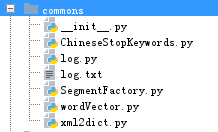


# 工程目录结构说明：

## 注意：每一个py 基本都有测试函数，可以运行下试试。注意输入路径



## 1 commons 共公



chinesesStopKeywords.py,停用词

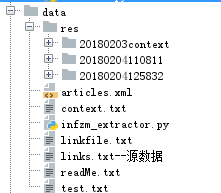
log.py日志

SegmentFactory.py，分词，使用的是jieba

wordVectory.py　词向量工具

xml2dict.py　内容解析，数据格式查看data目录　下的数据文本

## 2 data



Res是程序运行结果，按时间来保存

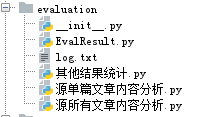
Articles.xml是夏天的数据

Context.txt是我使用infzm\_extractor.py爬的数据

Linkfile.txt是infzm\_extractor.py爬的数据的临时文件，保存的是url

Test.txt是数据格式，就两条记录

## 3 evaluation



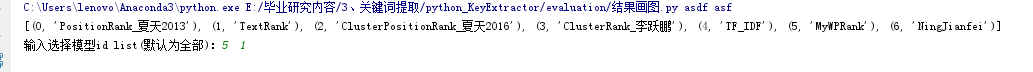
### 结果画图.py

生成PRF3个指标的图

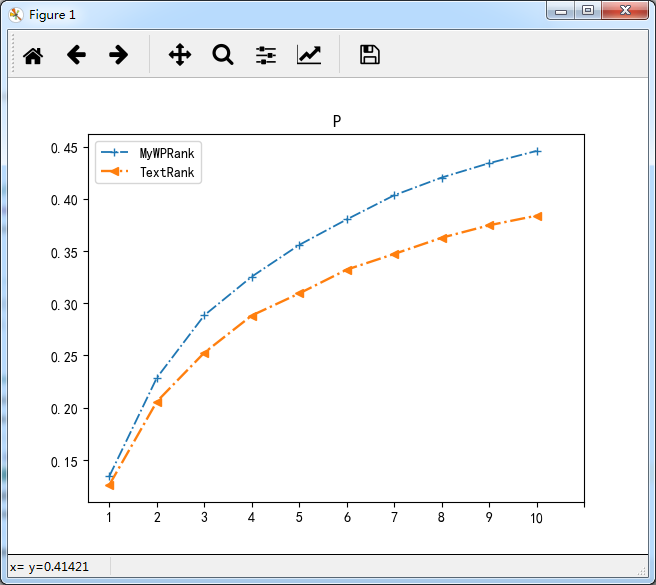
需要修改路径为生成的结果：

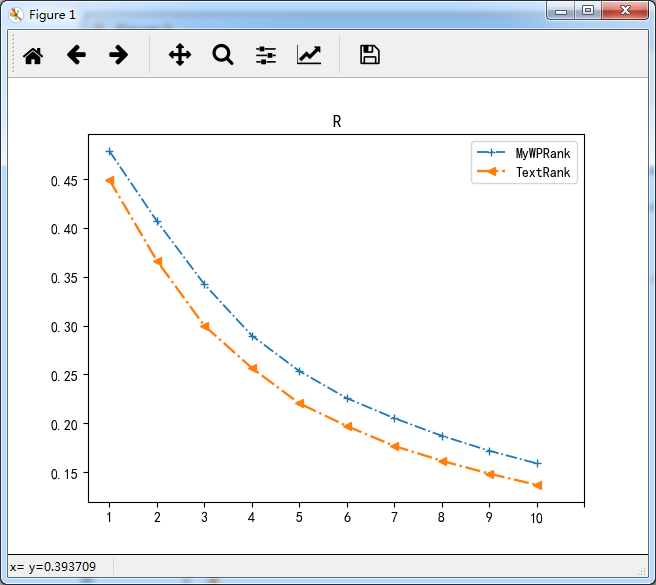
filePath = **'../data/res/20180301141258/allRes.csv'**

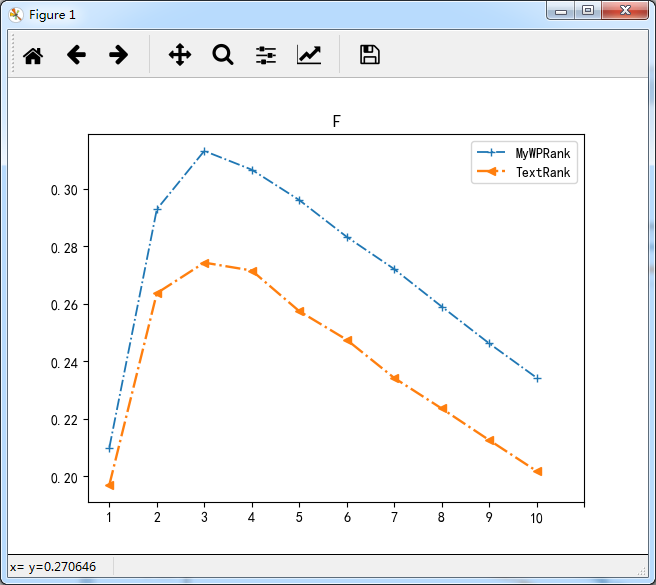
运行输入参数：



运行结果：







### evalResult.py是计算PRF的类

其他结果统计.py是统计以下信息，保存在“”data/res/时间/其他结果统计“”目录

t\_结果 = **''**t\_结果+=KeyExtractor.GraphType(modeId).name+**' '**t\_结果 +=**"topN="**+str(topN)+**' '**t\_结果 += **'命中总数='**+str(totCount)+**' '**t\_结果 +=**"平均每篇文章个数="**+str(totCount / allResList.\_\_len\_\_())+**' '**t\_结果 += **'长词个数(>=4)='**+str(长词个数)+**' '**t\_结果 += **'平均词长='**+str(totLen / totCount)+**' '**t\_结果 +=**"一个关键词没中的文章个数="**+str(tot没中的)

### 源所有文章内容分析　结果内容，

平均长度： 3.187458081824279

================================

长词(2)： 1093 占比： 0.36653252850435947

长词(3)： 663 占比： 0.22233400402414488

长词(4)： 975 占比： 0.32696177062374243

长词(>4)： 248 占比： 0.08316566063044936

================================ 总个数 2982

没在标题中出现的个数： 3407 没在内容中出现的个数： 783 都没出现过： 682

词数出现总次数： 5518

没在标题中出现的占比： 0.6174338528452338 没在内容中出现的占比： 0.1418992388546575 都没出现过占比： 0.12359550561797752

================================

词出现位置统计： {'q': 318, 'z': 195, 'h': 327, 'qz': 772, 'zh': 532, 'qzh': 3245}

权重：q : 1.059009092596029 z : 1.0361848209315272 h : 1.060679161254407 qz : 1.1432547782519948 zh : 1.0987196140285767 qzh : 1.602152532937465

================================

词性统计: [('n ', 1755), ('nr ', 412), ('v ', 337), ('ns ', 330), ('n n ', 313), ('j ', 216), ('vn ', 208), ('', 191), ('nt ', 142), ('n vn ', 134),

### 源单篇文章内容分析结果，对单篇文本进行分词，分布，词距计算

输出合并词后，按词在文本中位置权重{word: [占比，分布，权重，序]}

{'记忆': [0.9085989176187613, 1.6, 1.4537582681900183, 'n', 0], '国人': [0.8959711365003007, 1.6, 1.4335538184004812, 'n', 1], '历史': [0.8725195429945881, 1.6, 1.396031268791341, 'n', 2], '中国': [0.8544798556825015, 1.6, 1.3671677690920025, 'ns', 3], '七七事变': [0.7095610342754058, 1.6, 1.1352976548406495, 'nz', 4],

输出关键词信息

['七七事变', 0.7095610342754058, 1.6, 1.1352976548406495, 'nz', 4, 4]

['抗日战争', 0.0, 1.06, 0.0, 'nz', 233, 233]

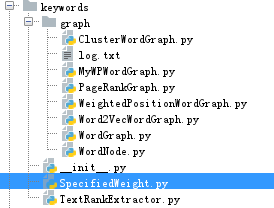
['国家记忆', 0.3036680697534576, 1.06, 0.3218881539386651, 'n', 43, 43]

['中日关系', 0.4533974744437763, 1.06, 0.4806013229104029, 'nz', 27, 27]

['日本战犯', 0.0, 1.06, 0.0, 'ns', 258, 258]

['南京大屠杀', 0.06313890559230306, 1.06, 0.06692723992784125, 'nz', 69, 69]

## 4 keywords



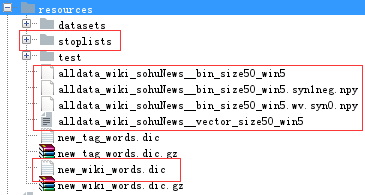
Graph目录，存放的是算法实现，

Specifiedweight.py没有用，

textRankExtractor.py这个是调用接口，主要有以下几种方式

**class** GraphType(Enum):  
 TF\_IDF=0  
 *# // 传统的TextRank方法 夏天. 词语位置加权TextRank的关键词抽取研究. 现代图书情报技术, 2013, 29(9): 30-34.* TextRank=1  
 *# // 词语位置加权 夏天. 词语位置加权 TextRank 的关键词抽取研究 2013, 区别于TextRank的是权值* PositionRank\_夏天2013=2  
 *###################### 主要是以上的，不进行词向量加载#########################  
 # // 融合Word2vec 与 TextRank的关键词抽取研究 NingJianfei 融合 Word2vec 与 TextRank 的关键词抽取研究* NingJianfei=3  
 *# // 词向量聚类加权* ClusterRank\_李跃鹏=4  
 *# // 词向量聚类 + 位置加权 夏天 词向量聚类加权 TextRank 的关键词抽取* ClusterPositionRank\_夏天2016=5  
  
 *# 提出的基于 Word2Vec 的词向量聚类关键词抽取 李跃鹏, 金翠, 及俊川. 基于 Word2vec 的关键词提取算法  
 # 没有实现  
  
 # 我的* MyWPRank = 6

## 5 resources



用的数据主要是画红框的，没有画的没用，关于new\_tag\_words.dic词典，我没用。这个词曲是关键词添加词典后，再进行分词。

## 6 conf-extractor.cfg

看readme.txt

## 7 g\_config.py

配置文件内容读取

## 8 main.py

主程序运行