QINMIN HU

2018年6月30日

LAB 6 Learning to Rank

一. 实验目的

1. 给定topic及某个引擎检索的结果，进行重新排序。
2. 简单的Point-wise实现：考虑单个文档，用传统的机器学习方法对给定查询下的文档的相关度进行学习——分类or回归: McRank，SVM，最大熵，Subset Ranking, SLR
3. 动手编写一些评价指标：MAP、NDCG、P@K

二. 实验步骤

1. 生成训练集

1.1.加载 query 和 document

（加载查询文件中的 title 作为 query，加载相应的 document 的 title 和 abstracttext 作为这个 text 的内容）

```
def loadXmlText(filename,docdir,foldername):
    print(docdir+'/'+foldername+'/'+filename)
    if filename.endswith('.xml'):
        xmldoc=ET.parse(docdir+'/'+foldername+'/'+filename)
    else:
os.rename(docdir+'/'+foldername+'/'+filename,docdir+'/'+foldername+'/'+filena
me+'.xml')
        xmldoc = ET.parse(docdir + '/' + foldername + '/' + filename +
'.xml')

    test=''
    for MedlineCitation in xmldoc.findall('MedlineCitation'):
        for article in MedlineCitation.findall('Article'):

            for journal in article.findall('Journal'):
```

```
                    for title in journal.findall('Title'):

                        test = test + ' ' + title.text
                for a in article.findall('Abstract'):
                    for abstract in a.findall('AbstractText'):

                        if abstract.text==None:
                            for s in abstract.findall('AbstractText'):
                                test = test + ' ' + s.text
                        else:
                            test = test + ' ' + abstract.text
        testList=[]

        i=0
        for word in test.split():

            if re.findall(r'\d+', word) or word  == "":
                continue
            while '_' in word or '`' in word or '.' in word or '-' in word or '!' in
word or ';' in word or ',' in word or '$' in word or '//' in word or '/' in word or '{' in
word or '}' in word or '*' in word or '#' in word or '^' in word or '|' in word or '~' in
word or '=' in word or '\'' in word or '+' in word or ':' in word or '?' in word:
                word = word.replace("_", "").replace('`', "").replace(".",
"").replace('-', "").replace("!", "").replace(
                    ";",
                    "").replace(
                    ",", "").replace("$", "").replace("//", "").replace('/',
"").replace('{', "").replace('}', "").replace(
                    '*',
                    "").replace(
                    '^', "").replace("|", "").replace('~', "").replace('=',
"").replace('\'', "").replace('+', "").replace(
                    ':',
                    "").replace(
                    '?', "").replace("#","")
            testList.append(word)

        text_token=tokenization(testList)
        text_stopword=stopword(text_token)
        text_lemmatize=lemmatization(text_stopword)
        text=stemmed(text_lemmatize)

        return text
```

1.2.预处理

（词条化）

```
def tokenization(str_test):
    word_token=[]
    for i in range(len(str_test)):
        if str_test[i]==None:
            word_token.append("None")
        elif re.findall(r'\S', str_test[i]):

            word_token.append(nltk.word_tokenize(str_test[i])[0])
    return word_token
```

（去除停用词）

```
def stopword(str_test):
    test=[]
    for word in str_test:
        if word in stopwords:
            continue
        else:
            test.append(word)
    return test
```

（词形归并）

```
def lemmatization(str_test):

    test=[]
    lemmatizaer=WordNetLemmatizer()
    for i in range(len(str_test)):
        if str_test[i]=="None":
            continue
        else:
            test.append(lemmatizaer.lemmatize(str_test[i]))
    return test
```

（词干还原）

```
def stemmed(str_test):
    test=[]
    porter_stemmer = PorterStemmer()
    for word in str_test:
        test.append(porter_stemmer.stem(word))
    return test

        text_token=tokenization(testList)
        text_stopword=stopword(text_token)
        text_lemmatize=lemmatization(text_stopword)
        text=stemmed(text_lemmatize)
```

1.3.倒排索引

```
def loadDocument(foldername,docdir,query):
    docLength={}#文档长度
    wordDir={}#倒排索引，word是key
    N=0
    for root,dirs,files in os.walk(docdir+'/'+foldername):
        count=0
        for file in files:
            if file.endswith('.gz'):
                continue
            else:
                count=count+1
        for file in files:
            if file=='.DS_Store':
                continue
            if file.endswith('.gz'):
                continue
            else:
                N=N+1
                if re.findall(r'\d+', file):
                    text=loadXmlText(file,docdir,foldername)
                    docLength[file]=len(text)
                    wordFreq={}
                    for word in text:
                        if word in wordDir:
                            if file in wordDir[word]:
                                wordDir[word]
[file]=wordDir[word][file]
                            else:
                                wordDir[word][file]=1
                        else:
                            test={}
                            test[file]=1
                            wordDir[word]=test
```

1.4.计算 tfidf、bm25

```
def getTfIdf(query,wordDir,N,docLength):
    wordDocFreq={}
    score={}
    for key,value in wordDir.items():
        wordDocFreq[key]=len(value)
    for doc in docLength.keys():
```

```
        s=0
        for word in query:
            if word in wordDocFreq.keys():
                idf=math.log(N/wordDocFreq[word])
            else:
                idf=0
            if word in wordDir.keys():
                if doc in wordDir[word].keys():
                    freq=wordDir[word][doc]/docLength[doc]
                    s=s+idf*freq
                else:
                    s=s+0
        score[doc]=s
    return score

def getBM25(query,wordDir,N,docLength):
    wordDocFreq={}
    score={}
    for key,value in wordDir.items():
        wordDocFreq[key]=len(value)
    count=0
    for doc in docLength.keys():
        count=count+docLength[doc]
    avgLen=count/N

    for doc in docLength.keys():
        s=0
        for word in query:
            if word in wordDocFreq.keys():
                idf=math.log((N-wordDocFreq[word]+0.5)/(0+0.5))
            else:
                idf=math.log((N-0+0.5)/(0+0.5))
            if word in wordDir.keys():
                if doc in wordDir[word].keys():
                    freq=wordDir[word][doc]
                    s=s+idf*freq*2.5/
(freq+1.5*(0.25+0.75*docLength[doc]/avgLen))
        score[doc]=s
    return score
```

1.5.写入训练集文件

（将 query 文件名、documentname、tfidf、tfidf 的 排名、bm25、bm25 的排名、对应的 label 写入 10152130138Traindata 文件）

```
def outputFile(tfidfSort,bm25Sort,queryname):
    print('output')
    f=open('10152130138_丁婉宁_trainnew','a')

    bm25KeyList=[]
    bmDic={}

    for key,value  in bm25Sort:
            bmDic[key]=value
            bm25KeyList.append(key)
    tfidfIndex=0

    for key,value in tfidfSort:
            tfidfIndex=tfidfIndex+1
            keylist=key.replace('.xml',"")
            keyIndex = bm25KeyList.index(key) + 1
            bmS = bmDic[key]

            if keylist in trainQrels[queryname]:
                    print(keylist + '  1\n')
                    writeStr=str(queryname) +' '+str(keylist)+' '+str(value)+' '
+str(tfidfIndex)+' '+str(bmS)+' '+str(keyIndex)+' 1\n'
            else:
                    writeStr = str(queryname) + ' ' + str(keylist) + ' ' + str(value)
+ ' ' + str(tfidfIndex) + ' ' + str(bmS)+ ' ' + str(keyIndex) + ' 0\n'
            f.write(writeStr)
    f.close()
```

2. 生成测试集（同训练集）

   2.1.加载 query 和 document

   2.2.预处理

   2.3.倒排索引

   2.4.计算 tfidf、bm25

   2.5.写入测试集文件： 将每个 query 文件名、documentname、tfidf、tfidf 的 排名、bm25、bm25 的排名写入 10152130138Testdata 文件

3. 建模—测试

3.1.根据生成的 10152130126Traindata 文件建立逻辑回归模型：

3.1.1.调用逻辑回归的库函数

```
def logreRression(features,flag):
    logit=sm.Logit(flag,features)
    model=logit.fit()
    resR=model.predict(testFeatures)
    print(len(testX))
    print(len(resR))
    return resR
```

3.1.2.自己编写逻辑回归

```
def logRegModel(train_x, train_y, opts):
    print('logRegModel')
    numSamples, numFeatures = shape(train_x)
    alpha = opts['alpha']
    maxIter = opts['maxIter']
    weights = ones((numFeatures, 1))
    for k in range(maxIter):
            for i in range(numSamples):
                    print(k,maxIter,i,numSamples)
                    output = sigmoid(train_x[i, :] * weights)
                    error = train_y[i, 0] - output
                    weights = weights + alpha * train_x[i, :].transpose() * error
    return weights
```

预测模型：

```
def predictLogModel(weights,train_x):
    print('predictLogModel')
    resM=[]
    numSamples, numFeatures = shape(train_x)
    for i in range(numSamples):
            p=sigmoid(train_x[i,:]*weights)
            print(p[0])
            resM.append(p[0])
    return resM
```

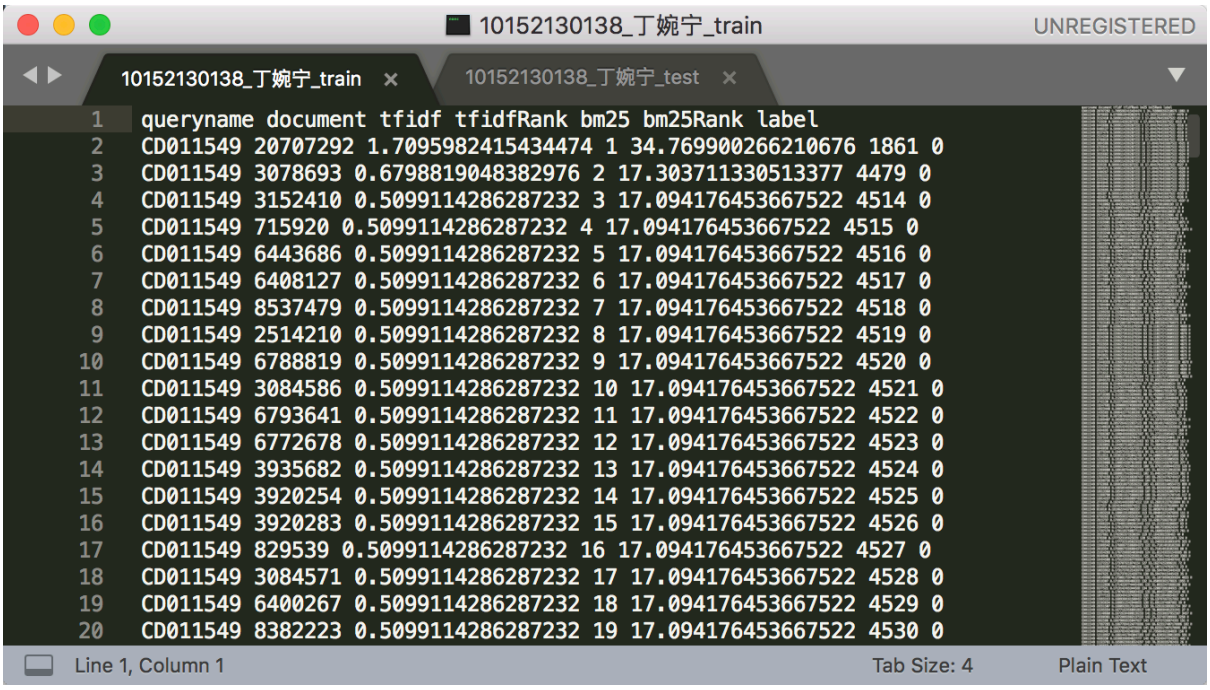
3.2.对生成的 10152130138Testdata 文件进行测试，得到每个 doc 对应 于相应的 query 的概率，并根据这个概率排序。

```
def outputFile(result):
    i=0
```

```
test=''
dicOffi={}
dicTest={}
f=open('machinePredict','w')
for file in testFile:
        if test==file[0]:
                dicTest[file[1]]=result[i]
        else:
                if i!=0:
                        dicOffi[test]=dicTest
                        dicTest={}
                test=file[0]
                dicTest[file[1]]=result[i]
        i=i+1
for key,value in dicOffi.items():
        valueSort =sorted(value.items(),key=lambda d:d[1],reverse=True)
        for key1,value2 in valueSort:
                f.write(str(key)+' '+str(key1)+' '+str(value2))
f.close()
```

三. 实验结果

10152130138_丁婉宁_train：



10152130138_丁婉宁_test：

10152130138_丁婉宁_test UNREGISTERED

10152130138_丁婉宁_test 10152130138_丁婉宁_train

```
CD010653 11368031 0.2907711272145743 10 14.790082352700766 1642
CD010653 15949668 0.2907711272145743 11 14.790082352700766 1643
CD010653 9516667 0.2907711272145743 12 14.790082352700766 1644
CD010653 10374655 0.2907711272145743 13 14.790082352700766 1645
CD010653 15567086 0.2907711272145743 14 14.790082352700766 1646
CD010653 20724112 0.2907711272145743 15 14.790082352700766 1647
CD010653 17261362 0.2907711272145743 16 14.790082352700766 1648
CD010653 15567085 0.2907711272145743 17 14.790082352700766 1649
CD010653 20537866 0.2907711272145743 18 14.790082352700766 1650
CD010653 20832995 0.2907711272145743 19 14.790082352700766 1651
CD010653 12591594 0.2907711272145743 20 14.790082352700766 1652
CD010653 17851040 0.2907711272145743 21 14.790082352700766 1653
CD010653 17306508 0.2907711272145743 22 14.790082352700766 1654
CD010653 15567073 0.2907711272145743 23 14.790082352700766 1655
CD010653 8794505 0.2907711272145743 24 14.790082352700766 1656
CD010653 20006543 0.2907711272145743 25 14.790082352700766 1657
CD010653 17123783 0.2907711272145743 26 14.790082352700766 1658
CD010653 12505146 0.2907711272145743 27 14.790082352700766 1659
CD010653 15653284 0.2907711272145743 28 14.790082352700766 1660
CD010653 20688481 0.2907711272145743 29 14.790082352700766 1661
```

Line 1, Column 1 Tab Size: 4 Plain Text