گزارش تمرین دوم درس بازیابی اطلاعات

مینا فریدی، شماره دانشجویی: ۸۱۰۱۰۰۴۳۰

آماده سازی محیط:

ابتدا سیستم عامل ubuntu را نصب کردم و سپس در آن به ترتیب جاوا، maven و گالاگو را نصب نمودم و فایل متن ها را که به صورت زیپ هست باید از زیپ در بیاوریم تا به فرمت trectext در بیاید.

ایجاد شاخص:

برای ایجاد شاخص به ترتیب مراحل توکن کردن، نرمال کردن کلمات، حذف حروف اضافه و ریشه یابی انجام داده شد.

نمونه انجام مراحل:

- According to Wikipedia, Information Retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources.

- **Tokenization**,
  According, to, Wikipedia, Information, Retrieval, is, the, activity, of, obtaining, information, resources, relevant, to, an, information, need, from, a, collection, of, information, resources.
- **Normalization**,
  according, to, wikipedia, information, retrieval, is, the, activity, of, obtaining, information, resources, relevant, to, an, information, need, from, a, collection, of, information, resources.
- **Stopping**,
  according, -, wikipedia, information, retrieval, -, -, activity, -, obtaining, information, resources, relevant,-, -, information, need, -, -, collection, -, information, resources.

  "to", "is", "the", "of", "an", "a" are treated as stop words. Because they are frequent and uninformative.
- **Stemming**,
  accord, -, wikipedia, inform, retrieve, -, -, act, -, obtain, inform, resource, relate,-, -, inform, need, -, -, collect, -, inform, resource.

برای توکن کردن متن و تبدیل آن به توکن‌های سازنده اش از این تنظیمات استفاده می‌کنیم:

```
"tokenizer" : {

"fields" : ["text","head"],

"formats" : {
    "text"    : "string",
    "head"    : "string"
  }
},
```

الگوریتم های مختلفی برای ریشه یابی وجود دارد که یکی از رایج ترین آن ها porter است که در واقع از آخر کلمات قسمت های پر تکرار مثل ing را حذف می‌کند و در کل عملکرد مناسبی دارد. تنظیم کردن آن به این صورت انجام شد:

```
"stemmer" : ["porter"],
```

برای ایجاد شاخص تنظیمات ذکر شده را ذخیره کرده و با دستور زیر اجرا میکنیم:

Galago/galago-3.16/core/target/appassembler/bin/galago build /home/mina/Desktop/CA1-Resources/indexResult.json

```
{

  "server" : true,

  "indexPath" : "indexDir",
  "inputPath" : "./CA1-Resources/Corpus/corpus/Documents.trectext",

    "stemmer" : ["porter"],

    "tokenizer" : {

    "fields" : ["text","head"],

    "formats" : {
        "text"    : "string",
        "head"    : "string"
      }
    },
    "fileType":"trectext"
}
```

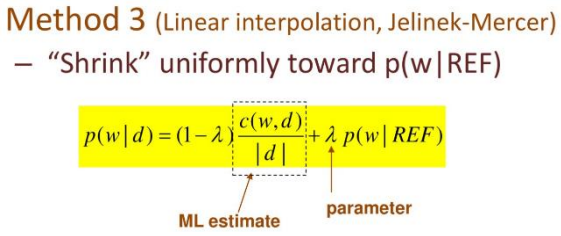
سوال یک:

گالاگو روش‌های متعددی برای امتیازدهی اسناد ارائه می‌کند و در صورت لزوم از الگوریتم‌های هموارسازی استفاده می‌کند. این الگوریتم های هموارسازی به عنوان Scorers نامیده می شوند و به عنوان عملگرهای پرس و جو پیاده سازی می شوند. عملگر هموارسازی به طور خودکار توسط Galago در نقاط مناسب در طول پیمایش ImplicitFeatureCastTraversal به پرس و جو اضافه می شود، بنابراین نیازی به گنجاندن صریح عملگر هموارسازی در پرس و جو وجود ندارد.

روش امتیازدهی پیش فرض در گالاگو روش دیریکله است. برای استفاده از هر امتیازدهی دیگری، امتیازدهنده را می‌توان به‌عنوان یک پارامتر scorer در کامند search تعریف کرد.

روش JM:

امتیازدهی JM، تابع Jelinek Mercer را پیاده سازی می‌کند. این روش از تعداد هر کلمه در داکیومنت و در کل مجموعه متن ها و طول آن ها و پارامتر lambda برای تعیین میزان احتمال هر یک استفاده می‌کند. فرمول روش JM بصورت زیر است:



مقدار پیش فرض لاندا برابر با ۰.۵ است. در این فرمول در واقع می‌خواهیم احتمال بک گراند و فورگراند را همزمان در نظر بگیریم. برای مثال اگر کلمه ای در همه داکیومنت ها وجود داشته باشد احتمال بک گراند آن کم می‌شود. مثل حروف اضافه.

برای استفاده از روش JM به این صورت عمل می‌کنیم:

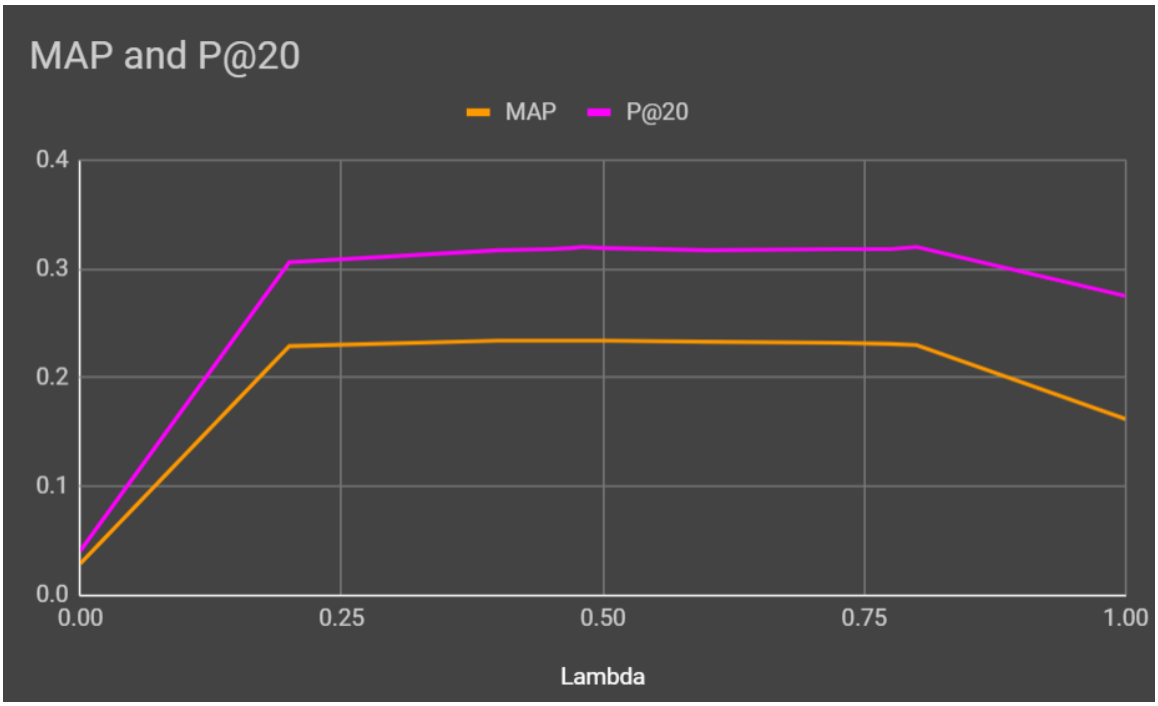
```
./Galago/galago-3.16/core/target/appassembler/bin/galago batch-search
/home/mina/Desktop/CA1-Resources/Queries/topics51-100.json --requested=1000 --
defaultTextPart=postings.porter --index=./indexFile  --scorer=jm  --lambda=0.5 > output.txt
```

در ورودی این دستور مقدار پارامتر لاندا تعیین شده و همچنین فایل‌های کوئری ۵۱ تا ۱۰۰ به عنوان کوئری داده می‌شود و از فایل indexFile که قبلا توسط گالاگو ساختیم را به عنوان شاخص در ورودی می‌دهیم. تعداد نتایج خروجی درخواست شده برابر با ۱۰۰۰ است.

در طول آزمایش به لاندا مقادیر بین صفر تا یک را می‌دهیم زیرا احتمال بین صفر و یک می‌تواند باشد.

روش JM		
P@20	**MAP**	**Lambda**
0.041	0.029	0
0.306	0.229	0.2
0.317	0.234	0.4
0.318	0.234	0.45
0.319	0.234	0.47
0.32	0.234	0.48
0.319	0.234	0.5
0.317	0.233	0.6
0.318	0.232	0.725
0.318	0.231	0.775
0.32	0.23	0.8
0.275	0.162	1

مقدار بهینه بدست آمده برابر با ۰.۴۸ است. برای یافتن این مقدار ابتدا مقادیر با فواصل بزرگ مثل صفر و یک و ۰.۵ را امتحان می‌کنیم سپس گام‌ها را کوچکتر کرده و به تدریج به مقدار ۰.۴۸ می‌رسیم. با زیاد شدن لاندا از صفر تا ۰.۲ مقدار MAP و P@20 با شیب بیشتری افزایش پیدا میکنند و به تدریج سرعت افزایش کاهش می یابد و در ۰.۴۸ به مقدار بهینه می‌رسند. برای مقادیر بیشتر از ۰.۴۸ لاندا، مقدار MAP و P@20 به تدریج کاهش می‌یابند تا به مقدار مینیمم به ترتیب ۰.۱۶۲ و ۰.۲۷۵ برسند.



روش Dirichlet

این روش هموارسازی بصورت پیش فرض در گالاگو استفاده می‌شود. این تابع از طول داکیومنت و پارامتر mu به عنوان ضریب برای بک گراند و فورگراند استفاده می‌شود. در واقع این روش مانند روش JM است با این تفاوت که مقدار لاندا برابر با حاصل تقسیم mu بر mu+d است.

- Method 4 (Dirichlet Prior/Bayesian)
  - Assume pseudo counts μp(w|REF)

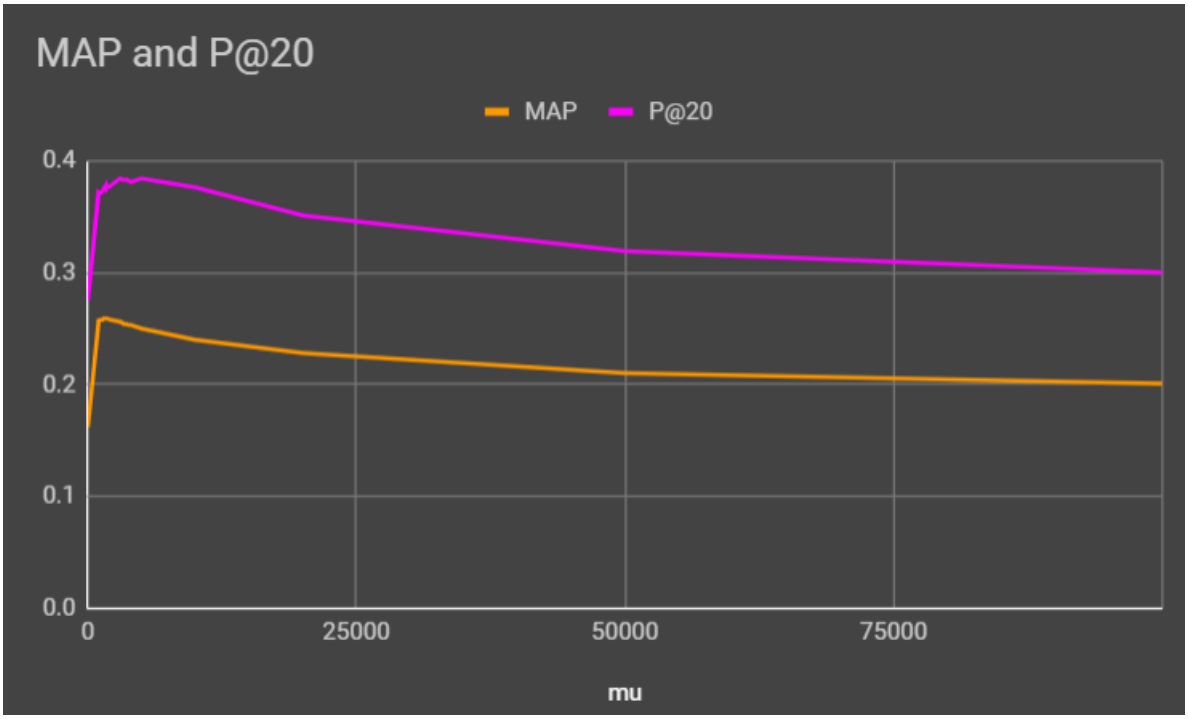
$$p(w|d)=\frac{c(w,d)+\mu\, p(w|REF)}{|d|+\mu}=\frac{|d|}{|d|+\mu}\frac{c(w,d)}{|d|}+\frac{\mu}{|d|+\mu}p(w|REF)$$

parameter

مقدار پیش فرض mu برابر با ۱۵۰۰ است. در ورودی این دستور مقدار پارامتر mu تعیین شده و همچنین فایل‌های کوئری ۵۱ تا ۱۰۰ به عنوان کوئری داده می‌شود و از فایل indexFile که قبلا توسط گالاگو ساختیم را به عنوان شاخص در ورودی می‌دهیم. تعداد نتایج خروجی درخواست شده برابر با ۱۰۰۰ است

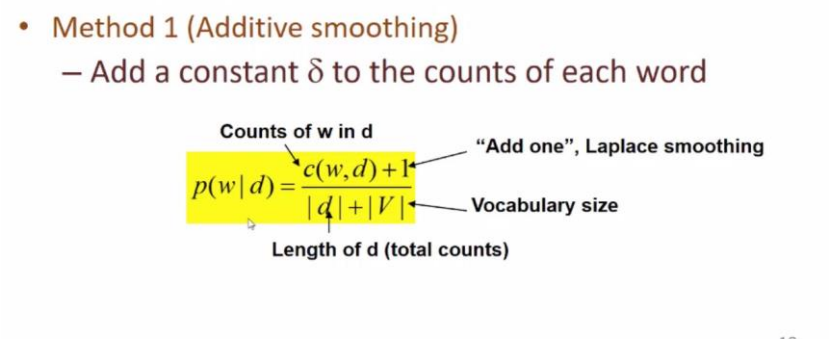
روش Dirichlet		
P@20	MAP	mu
0.275	0.162	0
0.372	0.257	1000
0.371	0.258	1200
0.373	0.258	1400
0.376	0.259	1500
0.377	0.259	1600
0.377	0.259	1650
0.378	0.259	1700
0.376	0.259	1750
0.377	0.259	1800
0.377	0.258	2000
0.384	0.256	3000
0.383	0.255	3200
0.383	0.254	3400
0.383	0.254	3450
0.383	0.254	3500
0.383	0.254	3550
0.383	0.254	3600
0.382	0.253	3800
0.381	0.253	4000
0.384	0.25	5000
0.376	0.24	10000
0.351	0.228	20000
0.319	0.21	50000
0.3	0.201	100000

برای آزمایش ابتدا گام های بلندتر بررسی شد و سپس اعداد بین ۱۰۰۰ و ۴۰۰۰ مقدار بالایی داشتند و بررسی حول آن ها انجام شد. مقادیر مختلف mu آزمایش شد و درنهایت تقریبا در نقطه 3200 به مقدار بهینه معیارها میرسیم.



روش Additve:

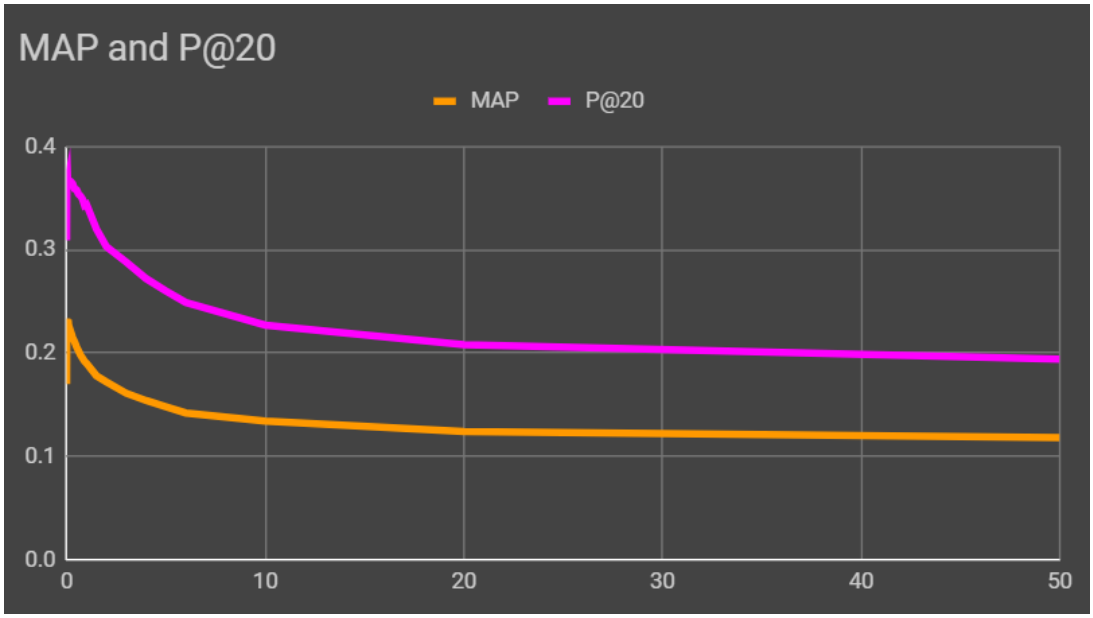
در این روش از تابع زیر برای هموارسازی استفاده می‌کنیم. این تابع اندازه سایز کل کلمات، سایز داکیومنت-ها و تعداد کلمات در داکیومنت‌ها و یک پارامتر تتا می گیرد. این پارامتر را به طور پیش فرض مقدار یک میگذاریم و هنگام آزمایش دستور به آن مقادیر مختلف می‌دهیم.



در ورودی دستور مقدار تتا و همچنین فایل‌های کوئری ۵۱ تا ۱۰۰ به عنوان کوئری داده می‌شود و از فایل indexFile که قبلا توسط گالاگو ساختیم را به عنوان شاخص در ورودی می‌دهیم. تعداد نتایج خروجی درخواست شده برابر با ۱۰۰۰ است.

P@20	MAP	teta
0.309	0.17	0
0.361	0.228	0.01
0.369	0.23	0.03
0.367	0.229	0.05
0.364	0.229	0.055
0.366	0.23	0.06
0.366	0.229	0.065
0.366	0.229	0.07
0.367	0.229	0.09
0.367	0.227	0.1
0.366	0.221	0.2
0.364	0.215	0.3
0.358	0.206	0.5
0.352	0.198	0.7
0.343	0.192	0.9
0.344	0.19	1
0.26	0.148	5

ابتدا گام های بلندتر بررسی شد و سپس اعداد بین ۱ و ۰ مقدار بالایی داشتند و سپس بین ۰.۱ و ۰.۰۱ رسیدیم و بررسی حول آن ها انجام شد. مقادیر مختلف تتا آزمایش شد و در نهایت در تتای ۰.۰۶ به مقدار بهینه MAP که برابر با ۰.۲۳ است میرسیم. همچنین در تتای ۰.۰۳ به مقدار بهینه p@20 که برابر با ۰.۳۶۹ است دست می یابیم.



سوال دوم:

در این سوال روش‌های JM و Dirichlet که در قسمت قبل آزمایش کردیم را با فرمول زیر ترکیب می‌کنیم.

$$P(w|d) = (1-\lambda)\frac{c(w_و d) + \mu p(W|C)}{|d| + \mu} + \lambda p(W|C)$$

در ورودی این دستور مقدار پارامتر لاندا و mu تعیین شده و همچنین فایل‌های کوئری ۵۱ تا ۱۰۰ به عنوان کوئری داده می‌شود و از فایل indexFile که قبلا توسط گالاگو ساختیم را به عنوان شاخص در ورودی می‌دهیم. تعداد نتایج خروجی درخواست شده برابر با ۱۰۰۰ است.

برای آزمایش به پارامترهای mu و lambda مقداردهی می کنیم. به پارامتر mu ابتدا مقادیر ۱۰۰۰ تا ۵۰۰۰ را با گام‌های بلند مقدار دهی می‌کنیم. همچنین به پارامتر lambda مقادیر صفر تا یک را مقدار می دهیم. به تدریج مطابق دو جدول پایین به پارامترها مقداردهی می‌کنیم و از روی رنگ میزان MAP و P@20 را مشاهده می‌کنیم و گام‌ها را کوچکتر می‌کنیم و به سمت mu=1500 و lamda=0.07 در MAP=0.259 همگرا می‌شویم.

MAP								
51-100								2 Step
0.6	0.5	0.4	0.3	0.2	0.1	0.07	0	
0.252	0.257	0.258	0.258	0.258	0.257	0.256	0.257	1000
0.251	0.255	0.257	0.257	0.258	**0.259**	**0.259**	**0.259**	1500
0.25	0.254	0.256	0.258	0.258	**0.259**	**0.259**	**0.259**	1700
0.25	0.252	0.254	0.256	0.258	0.258	0.258	0.258	2000
0.245	0.248	0.249	0.251	0.253	0.254	0.255	0.256	3000
0.24	0.244	0.247	0.248	0.25	0.251	0.252	0.253	4000
0.237	0.24	0.244	0.246	0.248	0.25	0.251	0.252	5000

در مورد P@20 هم در مقادیر lamda=0 و mu= 3000 به مقدار بهینه P@20=0.384 می رسیم. در کل به نظر می آید مقدار معیارهای بازیابی خیلی فرقی نکرده است.

P@20								
51-100								2 Step
0.6	0.5	0.4	0.3	0.2	0.1	0.07	0	
0.368	0.369	0.371	0.374	0.371	0.371	0.372	0.373	1000
0.369	0.375	0.373	0.375	0.377	0.375	0.375	0.376	1500
0.371	0.377	0.377	0.376	0.377	0.377	0.378	0.378	1700
0.375	0.379	0.38	0.379	0.38	0.378	0.378	0.377	2000
0.376	0.381	0.38	0.379	0.381	0.381	0.382	**0.384**	3000
0.371	0.374	0.38	0.38	0.379	0.381	0.38	0.381	4000
0.365	0.372	0.375	0.38	0.381	0.38	0.379	0.38	5000

سوال سوم:

عملگر pseudo Relevance feedback در واقع یک پرس و جو را با اصطلاحات "مرتبط" به طور خودکار تولید می کند. این عبارات را به جستار اصلی اضافه می کند (RelevanceModel3) یا به طور کلی جایگزین آنها می شود (RelevanceModel1). اپراتور آمار و اطلاعات طول را برای هر فیلد مشخص شده به دست می آورد. پرس و جو اصلی به ترکیبی از مجموع وزنی برای هر عبارت پرس و جو در هر یک از فیلدهای مشخص شده، با استفاده از وزن های مشخص شده برای هر فیلد، گسترش می یابد.