

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



دانشکده مهندسی برق و کامپیوتر

بازیابی هوشمند اطلاعات - تمرین دوم

سید مهدی رضوی

استاد : خانم دکتر شاکری

آبان ماه ۱۴۰۲

فهرست مطالب

۳	۱	تمرین اول عملی
۶	۲	تمرین دوم عملی
۷	۳	تمرین اول تشریحی
۸	۴	تمرین دوم تشریحی
۱۰	۵	تمرین سوم تشریحی

فهرست تصاویر

۳	۱	Additive , JM Smoothing
۴	۲	اسکرپتی برای آزمایش روش‌های هموارسازی
۴	۳	Drichlet Smoothing
۵	۴	TwoStep Smoothing
۶	۵	پیاده‌سازی الگوریتم ترکیبی
۶	۶	ارزیابی کد بالا

تمرین اول عملی ۱

```

mahdi@MahdiRazavi: ~/Documents/bazyabi/HW2-RUNS/additiveSmoothing
File Edit View Search Terminal Help
217 0.000 0.000 0.000 0.000 0.001 0.026 0.000 0.001 0.026 0.000 0.001 0.026 0.000 0.001 0.026
00 0.000 0.001 0.026 0.000 0.001 0.027 0.000 0.001 0.026 0.000 0.001 0.026 0.000 0.001 0.026
218 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
219 0.000 0.001 0.028 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
223 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
224 0.000 0.063 0.160 0.100 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.001 0.038 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
225 0.000 0.003 0.048 0.000 0.000 0.001 0.024 0.000 0.001 0.024 0.000 0.001 0.024 0.000 0.001 0.024
00 0.001 0.024 0.000 0.001 0.024 0.000 0.001 0.024 0.000 0.001 0.024 0.000 0.001 0.024 0.000

run-id      map      ndcg      p10
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runAdditive1.txt 0.007 0.048 0.012
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-1.txt 0.008 0.042 0.017
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-2.txt 0.009 0.042 0.016
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-3.txt 0.009 0.042 0.015
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-4.txt 0.009 0.041 0.013
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-5.txt 0.009 0.041 0.016
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-6.txt 0.009 0.041 0.015
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-7.txt 0.009 0.040 0.013
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-8.txt 0.009 0.040 0.012
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runJMS-9.txt 0.008 0.039 0.013

Sig-Test: randomized, threshold set to 0.050000
mahdi@MahdiRazavi:~/Documents/bazyabi/HW2-RUNS/additiveSmoothing$

```

شکل ۱: Additive , JM Smoothing

از آنجایی که Additive Smoothing هیچ وابستگی به متغیرهای لاند و مو ندارد ، آن را فقط یکبار آزمایش خواهیم کرد. درباره مقایسه روش هموارسازی Additive در مقایسه با JM Smoothing نمی‌توان کاملاً به قطعیت نظر داد چون دچار اختلاف بین پارامترها هستیم. برای سهولت امر تست پارامترهای مختلف ، اسکریپت آن را نوشته‌ام. پارامترهای متناظر مو و لاند را با توجه به بازدهای آن‌ها بررسی خواهیم کرد. همانطور که از پارامترها مشاهده می‌شود ، در JM Smoothing تغییر محسوسی در میزان دقت و سایر پارامترهای متناظر ایجاد نشده است ، اما با نگاهی ریزبینانه تر متوجه خواهیم شد که با افزایش میزان متغیر لاند ، میزان دقت تابع امتیازدهی به اسناد رو به کاهش می‌باشد.

```
additive.sh -- /Documents/bazyabi/HW2-RUNS/additiveSmoothing -- Atom

14 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.15 $queryPath > $runPath/runJMS-1.
15 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.25 $queryPath > $runPath/runJMS-2.
16 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.33 $queryPath > $runPath/runJMS-3.
17 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.45 $queryPath > $runPath/runJMS-4.
18 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.56 $queryPath > $runPath/runJMS-5.
19 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.62 $queryPath > $runPath/runJMS-6.
20 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.75 $queryPath > $runPath/runJMS-7.
21 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.853 $queryPath > $runPath/runJMS-8.
22 ./galago batch-search --requested=100 --index=$indexPath --scorer=JMSmoothing --landa=0.953 $queryPath > $runPath/runJMS-9.
23 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=1 $queryPath > $runPath/runDri-1.txt
24 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=10 $queryPath > $runPath/runDri-2.txt
25 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=20 $queryPath > $runPath/runDri-3.txt
26 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=30 $queryPath > $runPath/runDri-4.txt
27 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=45 $queryPath > $runPath/runDri-5.txt
28 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=56 $queryPath > $runPath/runDri-6.txt
29 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=67 $queryPath > $runPath/runDri-7.txt
30 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=78 $queryPath > $runPath/runDri-8.txt
31 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=87 $queryPath > $runPath/runDri-9.txt
32 ./galago batch-search --requested=100 --index=$indexPath --scorer=Dirichlet --mu=100 $queryPath > $runPath/runDri-10.txt
33 ./galago batch-search --requested=100 --index=$indexPath --scorer=TwoStep --mu=100 --landa=0.15 $queryPath > $runPath/runT
34 ./galago batch-search --requested=100 --index=$indexPath --scorer=TwoStep --mu=200 --landa=0.25 $queryPath > $runPath/runT
35 ./galago batch-search --requested=100 --index=$indexPath --scorer=TwoStep --mu=40 --landa=0.33 $queryPath > $runPath/runTv
36 ./galago batch-search --requested=100 --index=$indexPath --scorer=TwoStep --mu=5 --landa=0.45 $queryPath > $runPath/runTwc
37 ./galago batch-search --requested=100 --index=$indexPath --scorer=TwoStep --mu=50 --landa=0.56 $queryPath > $runPath/runTv
38 ./galago batch-search --requested=100 --index=$indexPath --scorer=TwoStep --mu=20 --landa=0.62 $queryPath > $runPath/runTv
```

شکل ۲: اسکریپتی برای آزمایش روش های هموارسازی

```
maledi@MahdiRazavi:~/Documents/bazyabi/HW2-RUNS/DirichletSmoothing

217 0.000 0.000 0.001 0.026 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
218 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
219 0.000 0.000 0.001 0.027 0.000 0.000 0.001 0.023 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
223 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
224 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.002 0.002 0.042 0.000 0.002 0.042 0.000 0.002 0.043 0.040 0.000 0.002 0.002 0.041 0.002 0.043 0.000 0.000 0.000
225 0.002 0.030 0.118 0.200 0.002 0.029 0.132 0.100 0.017 0.105 0.100 0.016 0.016 0.104 0.016 0.102 0.100 0.016 0.102 0.104
00 0.100 0.016 0.102 0.100 0.016 0.102 0.100 0.100 0.016 0.102 0.100 0.016 0.102 0.100 0.016 0.102 0.100 0.016 0.102 0.104
00 0.016 0.102 0.100 0.100 0.016 0.103 0.100 0.100 0.014 0.060 0.021

run-id map ndcg p10
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-1.txt 0.012 0.058 0.020
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-2.txt 0.012 0.058 0.019
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-3.txt 0.012 0.054 0.021
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-4.txt 0.012 0.057 0.021
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-5.txt 0.013 0.057 0.021
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-6.txt 0.013 0.057 0.024
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-7.txt 0.013 0.058 0.024
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-8.txt 0.013 0.058 0.022
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-9.txt 0.013 0.058 0.022
/home/mahdi/Documents/bazyabi/HW2-RUNS/additiveSmoothing/runDri-10.txt 0.014 0.060 0.021

Sig-Test: randomized, threshold set to 0.050000
maledi@MahdiRazavi:~/Documents/bazyabi/HW2-RUNS/DirichletSmoothings$
```

شکل ۳: Dirichlet Smoothing

در این روش هموارسازی ، همانطور که مشخص است بهترین اجرای ما مربوط به اجرای ۶ می باشد. با توجه به این که مقدار متغیر مو در اینجا برابر ۵۶ می باشد و ما در این سری آزمایش ها کلا بازه انتخابی برای متغیر مو را بین ۱ تا ۱۰۰ در نظر گرفته ایم ، مو اینطور به نظر می رسد که در حالت وسط نتیجه متغیرهای مربوط به دقت بازیابی بهتر خواهد بود.

```
File Edit View Search Terminal Help
mahdi@MahdiRazavi: ~/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
217 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.001 0.026 0.000 0.001 0.026 0.000 0.000
00 0.001 0.027 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
218 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
219 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
223 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
224 0.000 0.002 0.042 0.000 0.000 0.001 0.039 0.000 0.000 0.001 0.039 0.000 0.000 0.000 0.000 0.000
00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
225 0.000 0.012 0.093 0.100 0.012 0.105 0.100 0.014 0.097 0.100 0.011 0.089 0.000 0.000 0.000 0.000
00 0.100 0.014 0.123 0.100 0.012 0.119 0.000 0.013 0.120 0.000 0.012 0.117 0.000 0.000 0.000 0.000
00 0.010 0.096 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

run-id map ndcg p10
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-1.txt 0.014 0.060 0.020
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-2.txt 0.015 0.061 0.019
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-3.txt 0.014 0.060 0.022
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-4.txt 0.014 0.062 0.018
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-5.txt 0.015 0.062 0.020
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-6.txt 0.015 0.062 0.019
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-7.txt 0.015 0.062 0.020
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-8.txt 0.015 0.065 0.018
/home/mahdi/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing/runTwo-9.txt 0.013 0.061 0.018

Sig-Test: randomized, threshold set to 0.050000
mahdi@MahdiRazavi:~/Documents/bazyabi/HW2-RUNS/TwoStepSmoothing$
```

شکل ۴: TwoStep Smoothing

با توجه به تصاویر بالا بهترین حالت اجرا مربوط به اجرای هشتم می شود. متغیر لاندای برابر 0.853 و متغیر مو برابر ۳۰ می باشد. این نتایج با توجه به حالت های مختلف به دست آمده است.

۲ تمرین دوم عملی

```

197 df.setMaximumFractionDigits(8);
198
199 int epochs = 20;
200 while (epochs > 0)
201 {
202     for(String s : queryTerms)
203     {
204         p_Zt.put(s ,
205             (lambda * p_ThetaC.get(s))/
206             (lambda * p_ThetaC.get(s) + (1 - lambda) * p_ThetaF.get(s)));
207     }
208     double sigma = 0;
209     for(String s : queryTerms)
210         sigma = sigma + feedback_term_count.get(s) * (1 - p_Zt.get(s));
211
212     for(String s : queryTerms)
213     {
214         p_ThetaF.put(s , (feedback_term_count.get(s) * (1 - p_Zt.get(s) / sigma)));
215     }
216     for(String s : queryTerms)
217     {
218         WeightedTerm w = new wordWeight(s , p_ThetaF.get(s));
219         wt.add(w);
220     }
221     epochs--;
222 }
223
    
```

شکل ۵: پیاده‌سازی الگوریتم ترکیبی

ما الگوریتم ذکرشده در اسلاید درس مبنی بر EM Computation را پیاده‌سازی کرده‌ایم. از ساختمان داده HashMap برای ذخیره احتمال مورد انتظار و بیشینه احتمال به ازای هر کویری ترم می‌پردازیم. همانطور که در تصویر ارزیابی مشخص است ، نتایج این الگوریتم به طرز معناداری بهتر از روش های هموارسازی های تمرین اول است. با اختلاف بیشتر از 0.008 این الگوریتم با حدود ۲۰ بار تکرار به این صورت عمل خواهد کرد. به نظر می رسد هر بار تکرار به بیشینه کردن احتمالات تعلق و یا عدم تعلق کلمات مرتبط و نامرتب کمک خواهد کرد.

```

170 0.000 0.000 0.000
171 0.061 0.231 0.000
173 0.028 0.144 0.000
175 0.000 0.000 0.000
176 0.000 0.000 0.000
183 0.000 0.000 0.000
184 0.000 0.000 0.000
189 0.009 0.062 0.000
196 0.007 0.075 0.000
200 0.000 0.000 0.000
201 0.000 0.000 0.000
203 0.000 0.000 0.000
204 0.000 0.000 0.000
209 0.000 0.000 0.000
210 0.007 0.065 0.000
211 0.027 0.127 0.100
212 0.283 0.570 0.300
213 0.098 0.295 0.200
214 0.015 0.094 0.000
215 0.000 0.000 0.000
217 0.000 0.000 0.000
218 0.000 0.000 0.000
219 0.000 0.000 0.000
223 0.000 0.000 0.000
224 0.000 0.000 0.000
225 0.000 0.000 0.000
number NaN NaN NaN

run-id      map      ndcg      p10
/home/mahdi/Documents/bazyabi/HW2/Resources/doc_ranking.txt 0.020 0.060 0.031

Sig-Test: randomized, threshold set to 0.050000
mahdi@MahdiRazavi:~/Documents/bazyabi/HW2-RUNS/exercise-2$
    
```

شکل ۶: ارزیابی کد بالا



۳ تمرین اول تشریحی

۴ تمرین دوم تشریحی

$$OKAPI - TF = \sum_{t \in q} \frac{(k_1 + 1) * tf_{td}}{k + tf_{td}}$$

$$BM - 25 = \sum_{t \in q} \log \frac{(r_1 + 0.5)/(R - r_i + 0.5)}{(n - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} * \frac{(k_1 + 1)f_i}{K + f_i} * \frac{(k_2 + 1)qf_i}{K_2 + qf_i}$$

Relevance Documents to queries status

D3	D2	D1	Relevance
NO	YES	YES	Q1
NO	YES	NO	Q2
NO	YES	NO	Q3

متغیر b تاثیر طول سند را کنترل می‌کند.

متغیر k تاثیر تکرار عبارت‌ها را کنترل می‌کند.

$$Okapi(Q1, D1) = 1.3422$$

$$Okapi(Q1, D2) = 0.9174$$

$$Okapi(Q1, D3) = 0$$

$$Okapi(Q2, D1) = 1.3422$$

$$Okapi(Q2, D2) = 1.8265$$

$$Okapi(Q2, D3) = 0.88105$$

$$Okapi(Q3, D1) = 2.22333$$

$$Okapi(Q3, D2) = 2.74395$$

$$Okapi(Q3, D3) = 1.76211$$

$$\text{BM-25}(Q1, D1) = -1.4746$$

$$\text{BM-25}(Q1, D2) = -1.0079$$

$$\text{BM-25}(Q1, D3) = 0$$

$$\text{BM-25}(Q2, D1) = 2.9492$$

$$\text{BM-25}(Q2, D2) = 2.006639$$

$$\text{BM-25}(Q2, D3) = 0.967940$$

$$\text{BM-25}(Q3, D1) = 1.474647$$

$$\text{BM-25}(Q3, D2) = 3.01454$$

$$\text{BM-25}(Q3, D3) = 1.9358$$

Rankings Based on BM-25 Scoring

Q1 : (1) D3 , (2) D2 , (3) D1

Q2 : (1) D1 , (2) D2 , (3) D3

Q3 : (1) D2 , (2) D3 , (3) D1

Okapi-TF و BM-25 هر دو الگوریتم‌های رتبه‌بندی و امتیازدهی هستند که در سیستم‌های IR استفاده می‌شوند. الگوریتم Okapi-TF یک الگوریتم رتبه‌بندی است که فراوانی ترم‌ها را در سند بررسی می‌کند و با فرض این که هر چه یک کوئری ترم در سند بیشتر باشد، آن ترم با سند مرتبط‌تر است. BM-25 (Best Matching 25) همچنین یک الگوریتم رتبه‌بندی است که علاوه بر فراوانی کوئری ترم‌ها، از رویکردی احتمالی برای رتبه‌بندی اسناد استفاده می‌کند. (از جمله محاسبه IDF) مزیت الگوریتم BM-25 در آن است که طول سند و آمار کل Collection را در نظر می‌گیرد در حالی که در الگوریتم Okapi-TF اینطور نیست. این امر باعث می‌شود که الگوریتم BM-25 در امتیازدهی موثرتر واقع شود.

۵ تمرین سوم تشریحی

بخش اول

$$P(neural|D1) = 0.7 * \frac{1}{16} + 0.3 * \frac{1}{1000} = 0.04405$$

Dirichlet Prior

$$P(W|d) = \frac{|d|}{|d| + \mu} \frac{c(w, d)}{|d|} + \frac{\mu}{|d| + \mu} P(W|C)$$

$$P(SVM|C) = \frac{1}{33}$$

بخش دوم

$$P(SVM|D2) = \frac{17}{17 + 1500} * \frac{1}{17} + \frac{1500}{17 + 1500} * \frac{1}{33} = 0.03062264$$

بخش سوم

JM Smoothing

$$P(machinelearningmodels|D1) = P(machine|D1)*P(learning|D1)*P(models|D1) = 0.000475085$$

$$P(machinelearningmodels|D2) = P(machine|D2)*P(learning|D2)*P(models|D2) = 0.000354225$$

Dirichlet Smoothing

$$P(machinelearningmodels|D1) = P(machine|D1)*P(learning|D1)*P(models|D1) = 0.000445664$$

$$P(machinelearningmodels|D2) = P(machine|D2)*P(learning|D2)*P(models|D2) = 0.000442364$$

برای هموار ساز Dirichlet سند شماره ۱ با احتمال بالاتری بازگردانده خواهد شد. (البته اختلاف میزان احتمال سند ۱ خیلی قابل توجه به نسبت سند ۲ نیست.)
برای هموار ساز JM سند شماره ۱ با احتمال بالاتری بازگردانده خواهد شد. (این بار با اختلاف محسوس تری این سند بازگردانده خواهد شد.)

از برتری‌های JM نسبت به Dirichlet در مرتب‌سازی اسناد ممکن است شامل موارد زیر باشد : (همواره این اتفاق نمی‌افتد و مورد به مورد ممکن است متفاوت باشد)

۱. تعمیم‌پذیری : یکی از برتری‌های JM نسبت به Dirichlet این است که این روش قابلیت تعمیم‌پذیری بیشتری دارد. به عبارت دیگر JM قادر است بهتر با تغییرات در مجموعه اسناد و یا مجموعه کلمات موجود در اسناد ، سازگاری یابد. احساس می‌کنم بیشتر به خاطر این که متغیر لاندا بین صفر و یک نوسان می‌کند راحت‌تر می‌توان آن را تعمیم داد.

۲. انعطاف پذیری : JM انعطاف بیشتری در تنظیم پارامترهای خود دارد که این امکان را به ما می‌دهد که بهتر بتوانیم آن را به وضعیت خاص داده‌ها و مساله مورد نظرمان تطبیق بدهیم.

۳. استفاده از اطلاعات پیشین

۴. عملکرد بهتر در موارد پراکنده : در شرایطی که داده‌ها پراکنده باشد ، این روش ممکن است عملکرد بهتری داشته باشد و بهبود قابل توجهی نسبت به Dirichlet داشته باشد .