بسم الله الرحمن الرحیم

دانشکده مهندسی برق و کامپیوتر

بازیابی هوشمند اطلاعات - تمرین دوم

سید مهدی رضوی

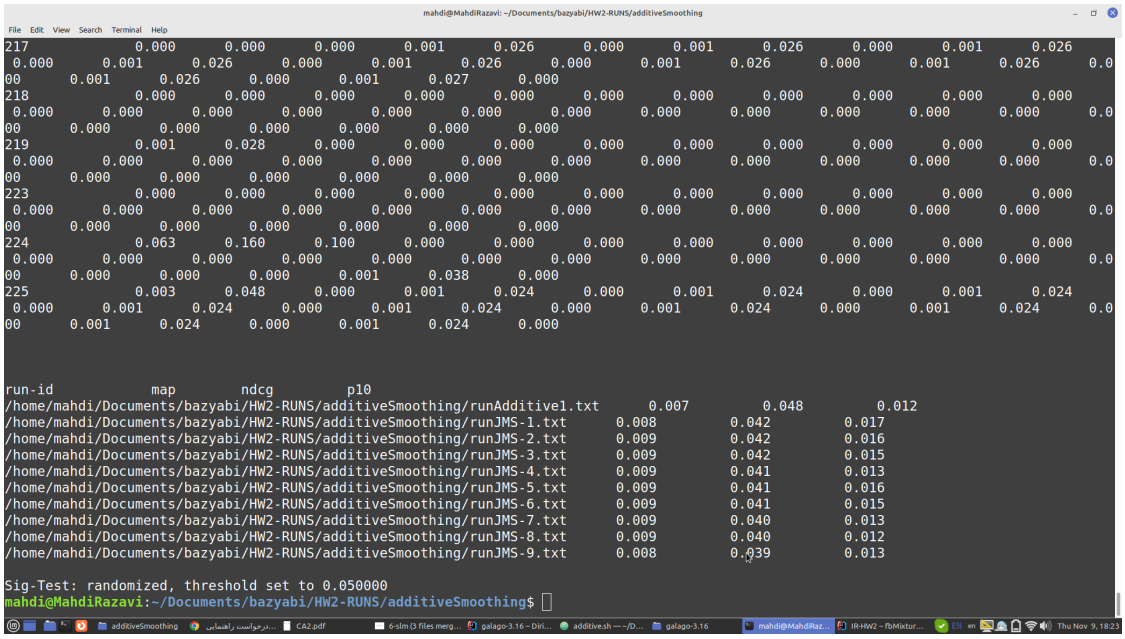
استاد : خانم دکتر شاکری

آبان ماه ۱۴۰۲

فهرست مطالب

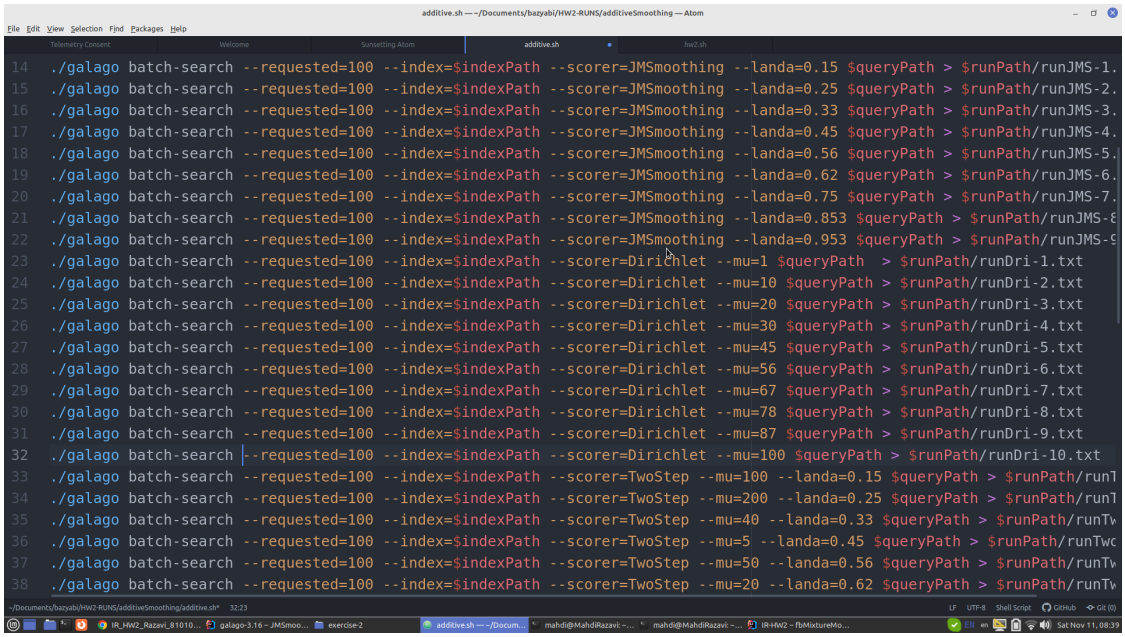
فهرست تصاویر

۱ تمرین اول عملی

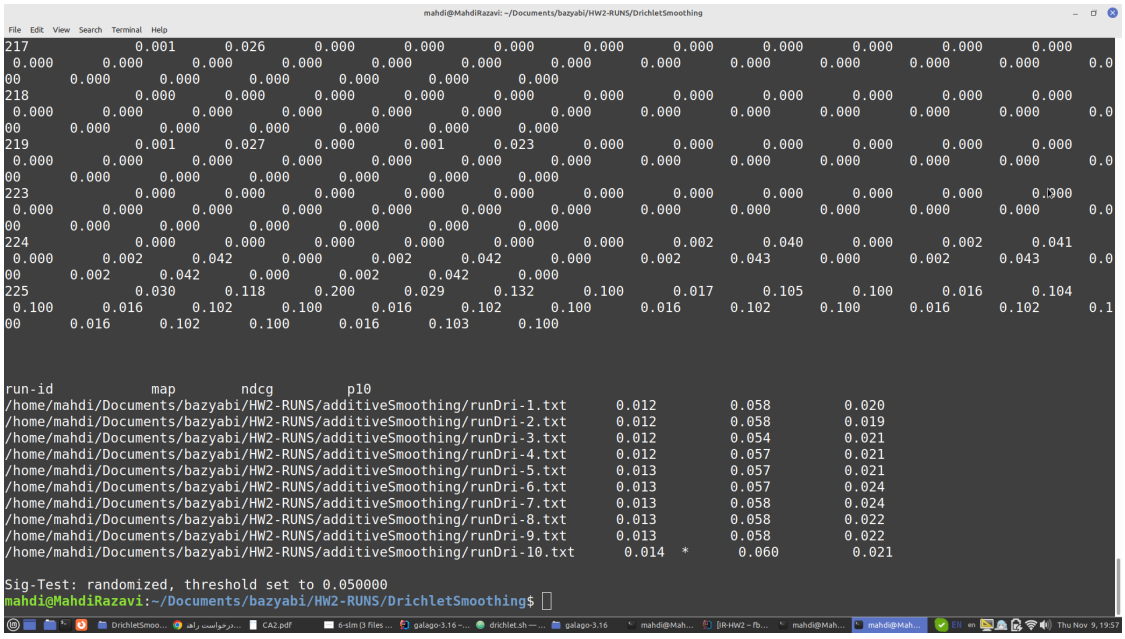


شکل ۱: Additive , JM Smoothing

از آنجایی که Additive Smoothing هیچ وابستگی به متغیرهای لاندا و مو ندارد ، آن را فقط یکبار آزمایش خواهیم کرد.
درباره مقایسه روش هموارسازی Additive در مقایسه با JM Smoothing نمی‌توان کاملا به قطعیت نظرداد چون دچار اختلاف بین پارامترها هستیم.
برای سهولت امر تست پارامترهای مختلف ، اسکریپت آن را نوشته‌ام. پارامترهای متناظر مو و لاندا را با توجه به بازه‌های 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 لاندا ، میزان دقت تابع امتیازدهی به اسناد رو به کاهش می‌باشد.

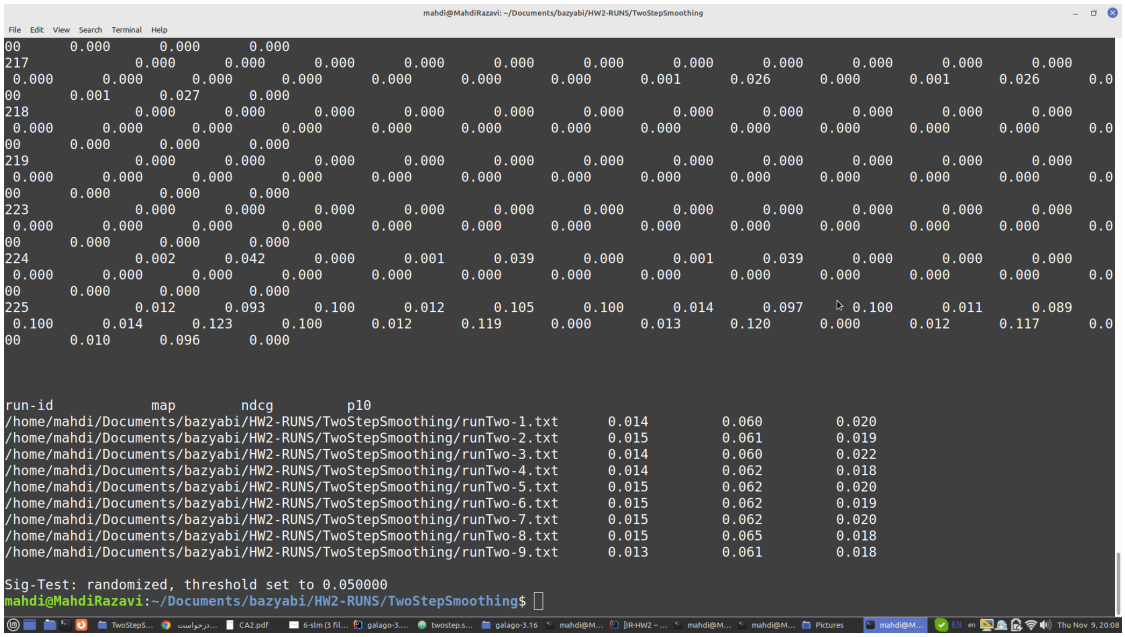


شکل ۲: اسکریپتی برای آزمایش روش‌های هموارسازی



شکل ۳: Drichlet Smoothing

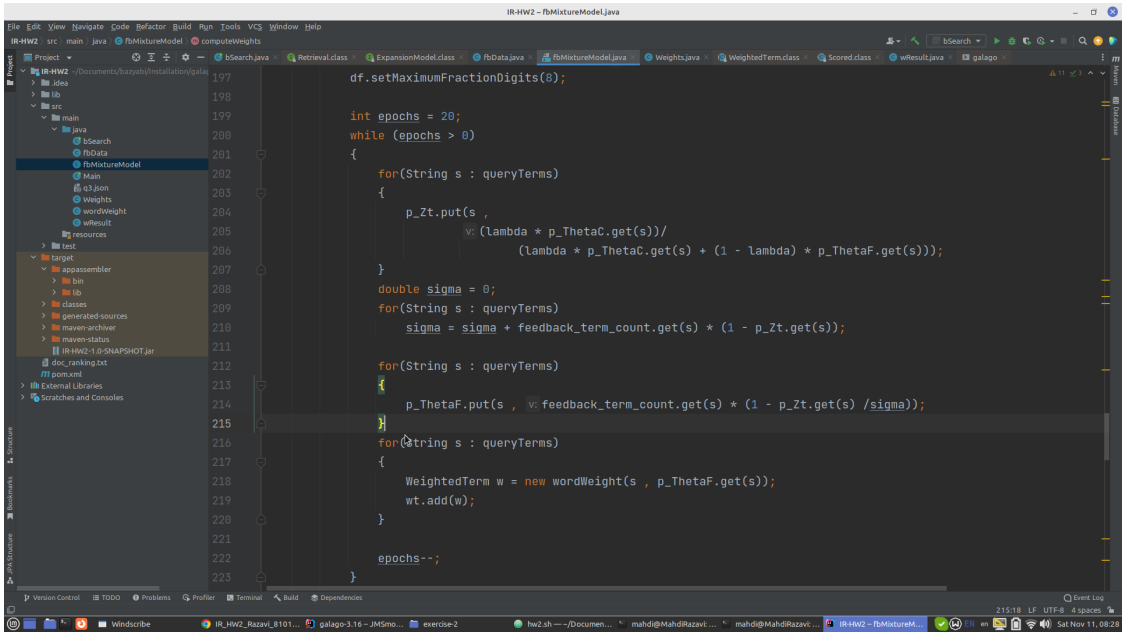
در این روش هموارسازی ، همانطور که مشخص است بهترین اجرای ما مربوط به اجرای ۶ می باشد. با توجه به این که مقدار متغیر مو در اینجا برابر ۵۶ می باشد و ما در این سری آزمایش ها کلا بازه انتخابی برای متغیر مو را بین ۱ تا ۱۰۰ در نظرگرفته ایم ، مو اینطور به نظر می رسد که در حالت وسط نتیجه متغیرهای مربوط به دقت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



شکل ۴: TwoStep Smoothing

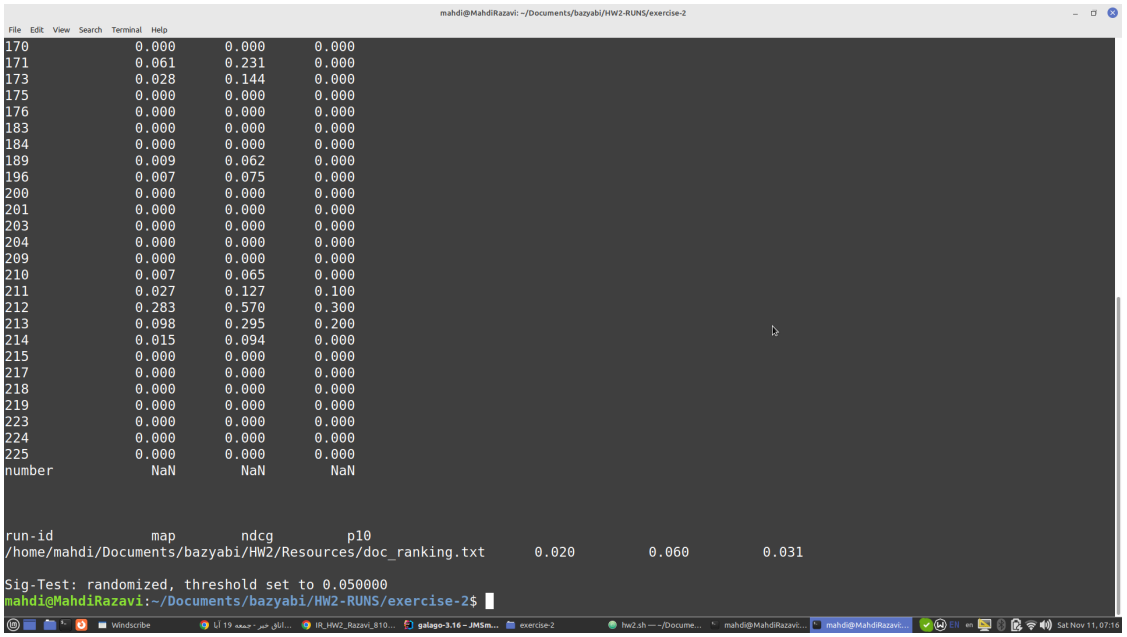
با توجه به تصاویر بالا بهترین حالت اجرا مربوط به اجرای هشتم می شود. متغیر لاندا برابر 0.853 و متغیر مو برابر ۳۰ می باشد. این نتایج با توجه به حالت های مختلف به دست آمده است.

۲ تمرین دوم عملی



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 نامرتبط کمک خواهد کرد.



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.

$$\text{Okapi(Q1 , D1)} = 1.3422$$
$$\text{Okapi(Q1 , D2)} = 0.9174$$
$$\text{Okapi(Q1 , D3)} = 0$$
$$\text{Okapi(Q2 , D1)} = 1.3422$$
$$\text{Okapi(Q2 , D2)} = 1.8265$$
$$\text{Okapi(Q2 , D3)} = 0.88105$$
$$\text{Okapi(Q3 , D1)} = 2.22333$$
$$\text{Okapi(Q3 , D2)} = 2.74395$$
$$\text{Okapi(Q3 , D3)} = 1.76211$$

BM-25(Q1 , D1) = -1.4746
BM-25(Q1 , D2) = -1.0079
BM-25(Q1 , D3) = 0
BM-25(Q2 , D1) = 2.9492
BM-25(Q2 , D2) = 2.006639
BM-25(Q2 , D3) = 0.967940
BM-25(Q3 , D1) = 1.474647
BM-25(Q3 , D2) = 3.01454
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 ترم ها را در سند بررسی می‌کند و با فرض این‌که 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 کوئری‌ترم‌ها ، 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 درنظر 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 سند۱ 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 محسوس‌تری 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 تغمیم 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 تظبیق 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 .