

# بازیابی اطلاعات

دکتر امین گلزاری اسکوئی

[a.golzari@azaruniv.ac.ir](mailto:a.golzari@azaruniv.ac.ir)

[a.golzari@tabrizu.ac.ir](mailto:a.golzari@tabrizu.ac.ir)

<https://github.com/Amin-Golzari-Oskoue>



دانشگاه صنعتی ارومیه

پاییز ۱۴۰۲

# فصل ۷

## محاسبه امتیازها در یک سیستم کامل جستجو

مطالب این فصل

اهمیت رتبه بندی: مطالعات کاربر در گوگل

نرمال سازی طول: نرمال سازی محوری

سیستم جستجوی کامل

## چرا رتبه بندی اهمیت دارد؟

- ❖ مشکلات مربوط به بازیابی بدون رتبه، همانطور که قبلا گفته شد:
  - . کاربران می‌خواهند به نتایج محدودی نگاه کنند، نه به همه‌ی آن‌ها.
  - . نوشتن پرس‌وجوهایی که نتایج کمی دارند بسیار سخت است. (حتی برای جستجوگران متخصص)
  - . رتبه بندی مهم است، زیرا به طور موثر مجموعه بزرگی از نتایج را به نتیجه بسیار کوچک کاهش می‌دهد.

❖ بعدا در مورد جمله "کاربران فقط به چند نتیجه نگاه می‌کنند" خواهیم پرداخت.

❖ واقعیت این است که در اکثر مواقع، آنها فقط حداکثر 3 نتیجه را بررسی می‌کنند.

## بررسی تجربی تاثیر رتبه بندی

❖ چگونه می‌توانیم میزان اهمیت (رتبه بندی را اندازه گیری کنیم؟

❖ توجه کنید که جستجوگران هنگام جستجو در یک محیط کنترل شده (CONTROLLED SETTING) چه کاری انجام می‌دهند:

. از آنها فیلمبرداری کنید. (Videotape them)


. از آنها بخواهید، همین که به فکرشان فطور کرد بگویند. (think loud)

. با آنها مصاحبه کنید.

. آنها را زیر نظر بگیرید.

. Time them

. کلیک های آنها را بشمارید و ذخیره کنید.



❖ Dan Russell مدیر Uber Teach در مورد کیفیت جستجو و رضایت مندی کاربران در گوگل است.

❖ اسلاید های بعدی از سفرانی های Dan Russell هستند.



Interview video

So.. Did you notice the FTD official site?

To be honest, I didn't even look at that.

At first I saw "from \$20" and \$20 is what I was looking for.

To be honest, 1800-flowers is what I'm familiar with and why I went there next even though I kind of assumed they wouldn't have \$20 flowers

And you knew they were expensive?

I knew they were expensive but I thought "hey, maybe they've got some flowers for under \$20 here..."

But you didn't notice the FTD?

No I didn't, actually... that's really funny.

## Rapidly scanning the results

Note scan pattern:

Page 3:  
Result 1  
Result 2  
Result 3  
Result 4  
Result 3  
Result 2  
Result 4  
Result 5  
Result 6 <click>

**Q: Why do this?**

A: What's learned later  
influences judgment  
of earlier content.

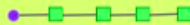


## Kinds of behaviors we see in the data

Short / Nav



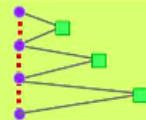
Topic exploration



Topic switch



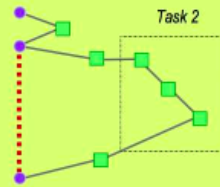
Methodical results exploration



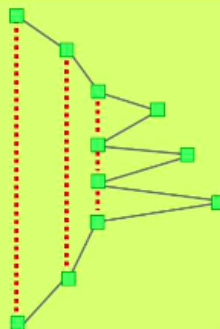
Query reform



Multitasking

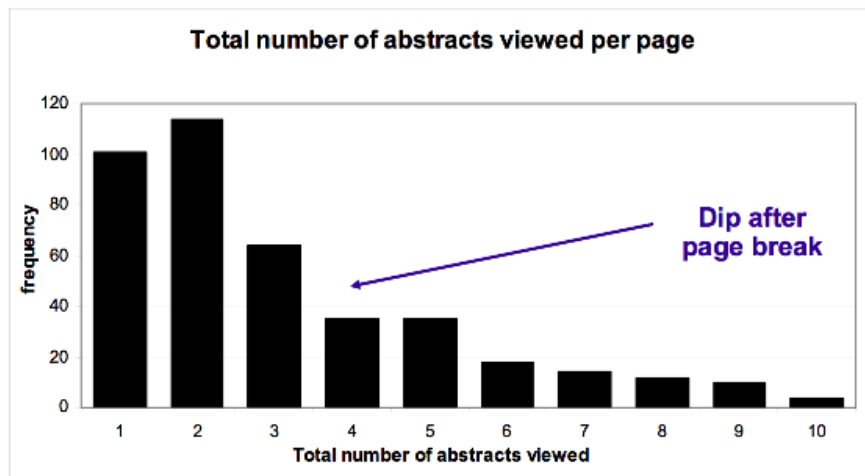


Stacking behavior





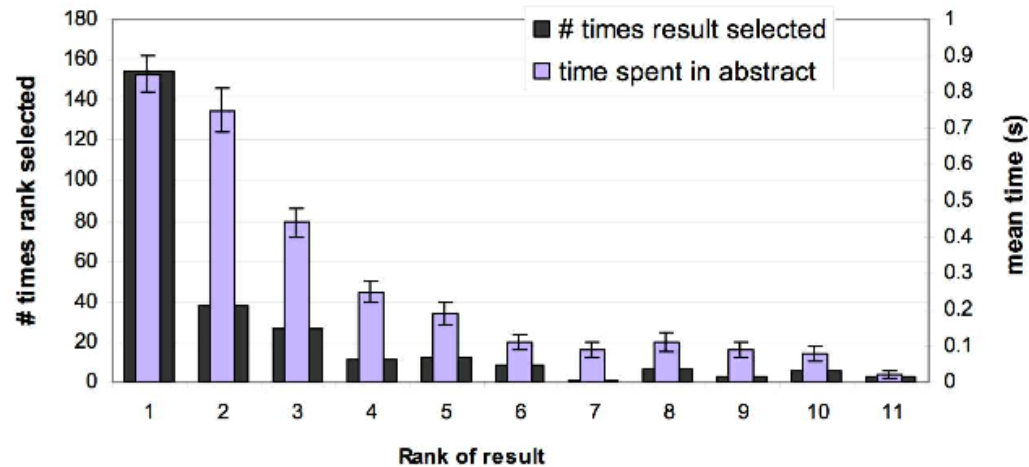
How many links do users view?



Mean: 3.07 Median/Mode: 2.00



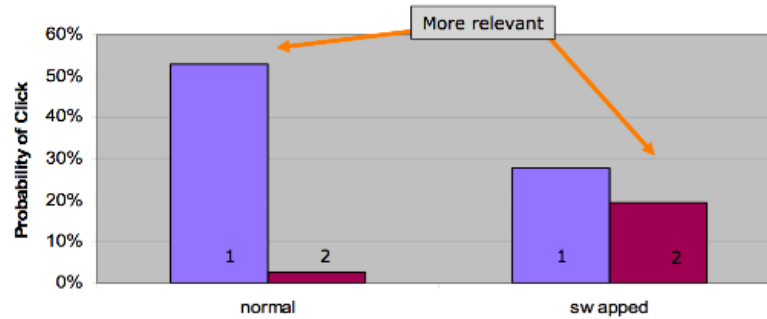
## Looking vs. Clicking



- Users view results one and two more often / thoroughly
- Users click most frequently on result one

## Presentation bias – reversed results

- Order of presentation influences where users look **AND** where they click



## خلاصه:

❖ مشاهده چکیده‌ها: کاربران به احتمال زیاد چکیده صفحات با رتبه برتر (1,2,3,4) را نسبت به چکیده صفحات با رتبه پایین مطالعه می‌کنند.

❖ **Clicking: Distribution is even more skewed for clicking**

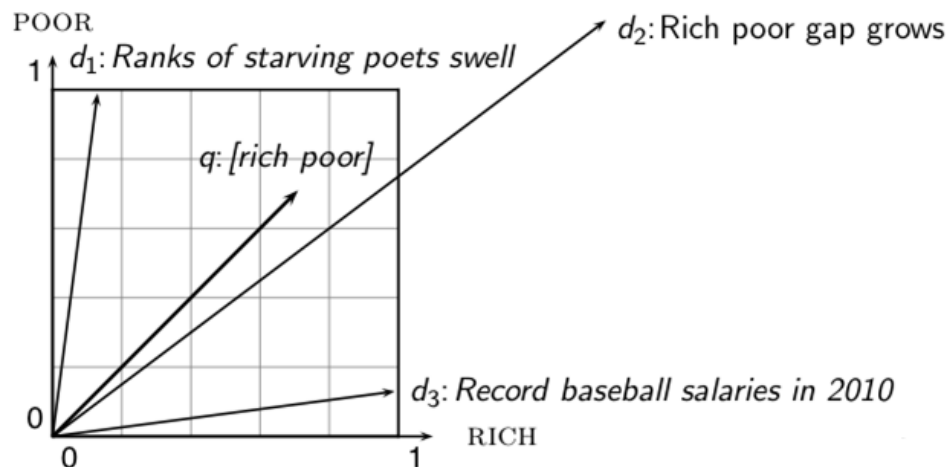
❖ یک مورد از هر دو مورد، کاربران بر روی صفحه با top rank کلیک می‌کنند.

❖ حتی اگر صفحه top rank مرتبط با موضوع نباشد، 30 درصد از کاربران روی آن کلیک می‌کنند.

. بسیار مهم است که رتبه بندی را صحیح انجام دهیم.

. به دست آوردن صحیح صفحه top rank مهم است.

## چرا "فاصله" ایده خوبی نیست؟



❖ فاصله اقلیدسی  $\vec{d}_2$  و  $\vec{q}$  بزرگ است، اگر چه توزیع عبارات در پرس و جو  $q$  و توزیع اصطلاحات در سند  $d_2$  بسیار شبیه هستند. به همین دلیل است که ما نرمال سازی طول انجام می دهیم یا به طور معادل از کسینوس برا محاسبه امتیاز های تطبیق پرس و جو و سند استفاده می کنیم.

## تمرین : یک مسئله برای سازی کسینوسی

❖ Query q: “anti-doping rules Beijing 2008 olympics”

❖ سه سند را با هم مقایسه کنید

D1: یک سند کوتاه در مورد قوانین ضد دوپینگ در المپیک 2008

D2: یک سند طولانی شامل یک کپی از d1 و 5 خبر دیگر، همه در موضوعات متفاوت از المپیک/ضد دوپینگ

D3: یک سند کوتاه در مورد قوانین ضد دوپینگ در المپیک 2004 آتن

❖ در مدل فضای برداری چه رتبه ای را انتظار داریم؟

❖ در این مورد چه کاری می توانیم انجام دهیم؟

# نرمال سازی محوری

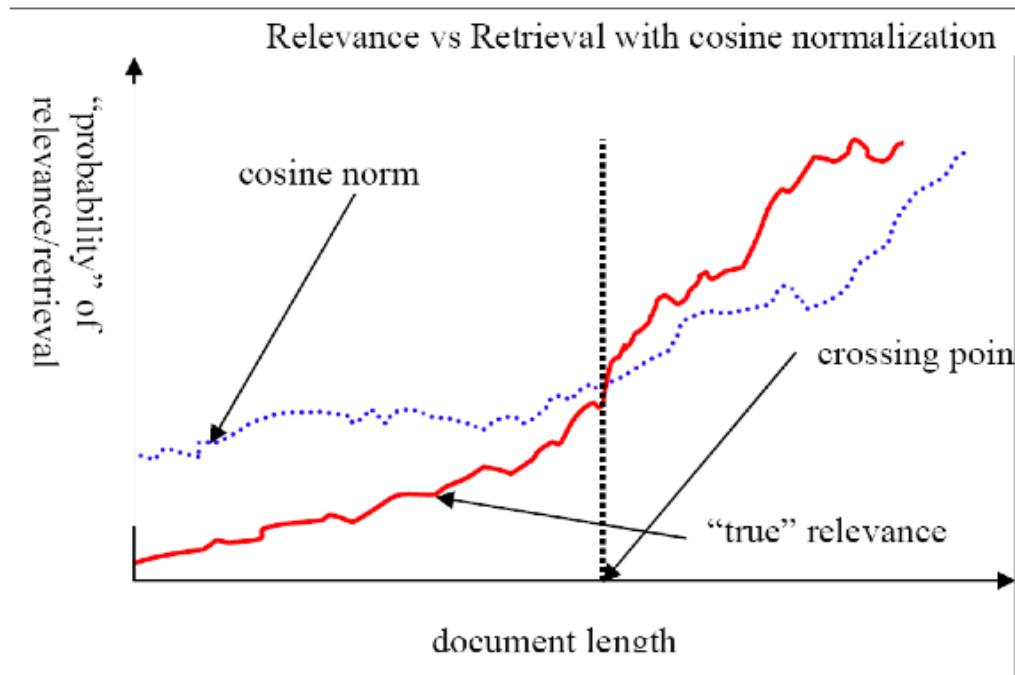
❖ نرمال سازی کسینوسی وزن هایی را تولید می کند که برای اسناد کوتاه بسیار بزرگ و برای اسناد طولانی بسیار کوچک هستند. (به طور میانگین)

❖ با کاهش پرس و جو شباهت اسناد کوتاه و با افزایش پرس و جو شباهت اسناد طولانی بیشتر می شود.

❖ این مزیت نادرست، اسناد کوتاه را از میان برمی دارد.

❖ Adjust cosine normalization by linear adjustment: “turning” the average normalization on the pivot

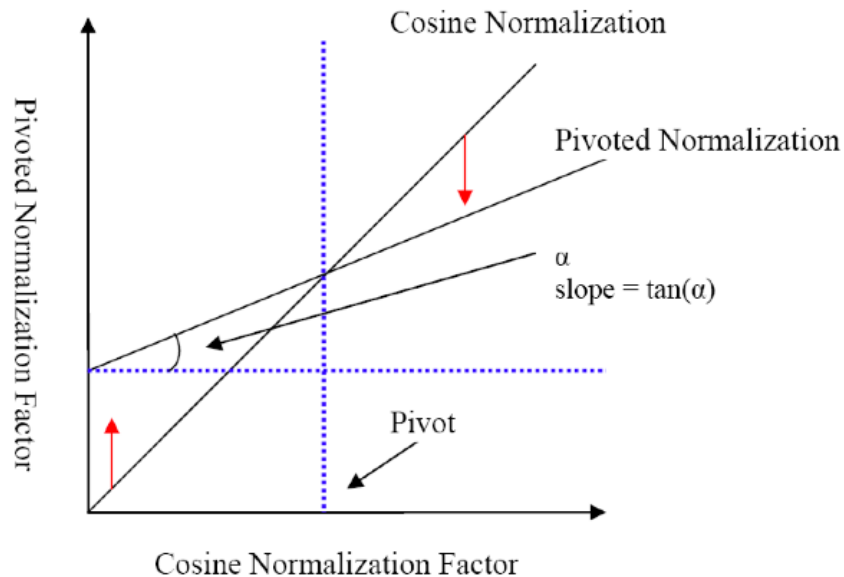
## پیش بینی و احتمال رابطه صحیح



source:  
Lillian Lee



## Pivot normalization



source:

Lillian Lee

## نرمال سازی محوری: Amit Signal's experiments

Cosine	Pivoted Cosine Normalization				
	Slope				
	0.60	0.65	0.70	<b>0.75</b>	0.80
6,526	6,342	6,458	6,574	<b>6,629</b>	6,671
0.2840	0.3024	0.3097	0.3144	<b>0.3171</b>	0.3162
Improvement	+ 6.5%	+ 9.0%	+10.7%	<b>+11.7%</b>	+11.3%

❖ اسناد مربوطه بازیابی شده و (تخیر در) میانگین درستی.

## ما به فراوانی عبارت در شاخص نیاز داریم

BRUTUS	→	1 ,2	7 ,3	83 ,1	87 ,2	...
CAESAR	→	1 ,1	5 ,1	13 ,1	17 ,1	...
CALPURNIA	→	7 ,1	8 ,2	40 ,1	97 ,3	

❖ فراوانی عبارت

ما به موقعیت ها نیز نیاز داریم که اینجا نشان داده نشده‌اند.

# فراوانی عبارت در شاخص معکوس (وارونه)

❖ در هر پست،  $tf_{t,d}$  علاوه بر docID ذخیره می‌کنیم.

❖ به عنوان یک فراوانی اعداد صمیم، نه به عنوان یک عدد حقیقی وزن دار

❖ فشرده‌سازی اعداد حقیقی دشوار است.

❖ کد unary برای رمزگذاری فراوانی اطلاعات موثر است. چرا؟

❖ به طور کلی، فضای اضافی مورد نیاز کم است. کم‌تر از یک بایت برای هر پست با فشرده‌سازی بیتی.

❖ یا یک بایت به ازای هر پست با کد بایت متغیر.

## تمرین: چگونه top k را در رتبه بندی محاسبه کنیم؟ (K سند برتر)

- ❖ در بسیاری از برنامه ها، ما نیازی به رتبه بندی کامل نداریم.
- ❖ ما فقط به top K برای یک k کوچک نیاز داریم (به عنوان مثال،  $k = 100$ )
- ❖ اگر به یک رتبه بندی کامل نیاز نداریم، آیا روش کارآمدی برای محاسبه top K وجود دارد؟
- ❖ به صورت ساده:
  - امتیاز همه N سند را محاسبه کنید.
  - مرتب سازی کنید.
  - top K را برگردانید.

❖ مشکل این راه چیست؟

❖ راه جایگزین چیست؟

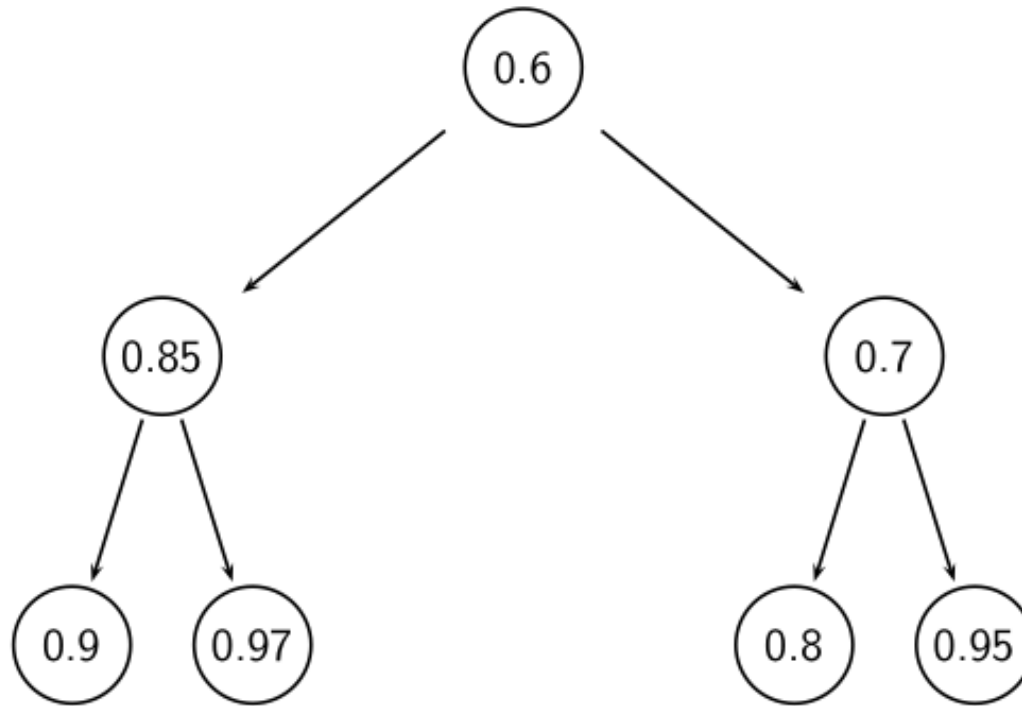
## از درخت دودویی min-heap برای انتخاب top k از N استفاده می‌کنیم.

❖ از یک درخت دودویی min (binary min heap) استفاده کنید.

❖ یک درخت دودویی min heap درخت باینری است که در آن مقدار هر گره کمتر از مقادیر فرزندانش است.

❖  $O(N \log_2^k)$  عملیات را برای سافت انجام می‌دهد. (که در آن n تعداد اسناد است) سپس k winners را در مراحل  $O(k \log_2^k)$  می‌فواند.

## Binary min heap



# انتخاب اسناد امتیازدهی top k در $O(N \log_2^k)$

❖ هدف: k سند برتر که تا کنون دیده شده‌اند را نگه دارید.

❖ از یک min heap استفاده کنید.

❖ برای پردازش یک سند جدید  $d'$  با امتیاز  $s'$

❖ Get current minimum  $h_m$  of heap ( $O(1)$ )

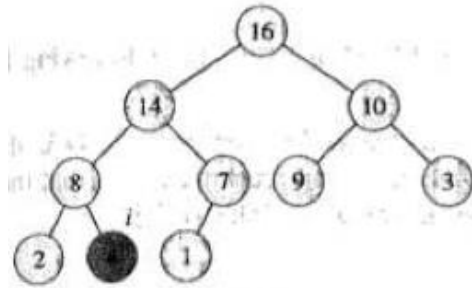
❖ If  $s' < h_m$  skip to next document

❖ If  $s' > h_m$  heap-delete-root ( $O(\log k)$ )

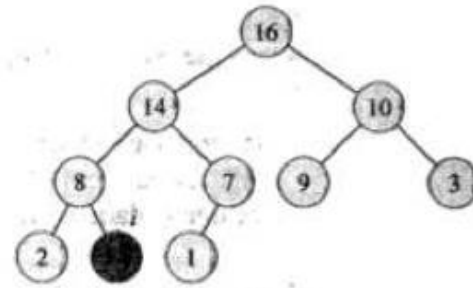
❖ Heap-add  $d'/s'$  ( $O(\log k)$ )



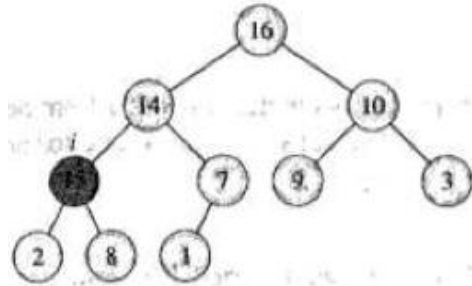
# نمونه صف اولویت



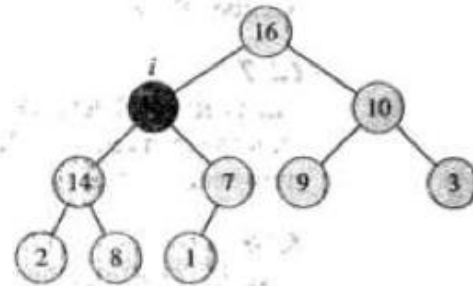
(a)



(b)



(c)



(d)

❖ رتبه بندی دارای پیچیدگی زمانی  $O(N)$  است که  $N$  تعداد اسناد است.

❖ بهینه سازی ها عامل دائمی (constant factor) را کاهش می دهند، اما همچنان برای  $O(N)$ ,  $N > 10^{10}$  است.

❖ آیا الگوریتم های زیرقطی وجود دارند؟

❖ کاری که ما در عمل انجام می دهیم: حل مسئله  $k$ -nearest neighbor (kNN) برای بردار پرس و جو (QUERY POINT) است.

❖ هیچ راه حل کلی برای این مسئله وجود ندارد که زیرقطی باشد.

❖ هنگامی که طبقه بندی kNN را در IIR14 انجام دهیم، این موضوع را دوباره بررسی خواهیم کرد.

❖ ایده یک: لیست پست‌ها را دوباره مرتب کنید.

- به جای مرتب کردن بر اساس docID، بر اساس برفی از معیارهای "expected relevance" مرتب کنید.

❖ ایده دو: اکتشافی برای پاکسازی فضای جستجو

- تضمینی برای درستی وجود ندارد.
- اما به ندرت موفق نمی‌شود.
- در عمل، به زمان ثابت نزدیک است.
- برای این کار، ما به مفاهیم پردازش "اسناد در یک زمان" و "عبارت در یک زمان" نیاز داریم.

## مرتب سازی به صورت non-docID ، لیست پست ها

- ❖ تا کنون لیست پست ها به صورت docID مرتب شده اند.
- ❖ معیاری مستقل از پرس و جو برای (goodness) بودن یک صفحه.
- ❖ Example: PageRank  $g(d)$  of page  $d$ , a measure of how many “good” pages hyperlink to  $d$  (chapter 21)
- ❖ ترتیب اسناد در لیست پست ها با توجه به PageRank:

$$g(d_1) > g(d_2) > g(d_3) > \dots$$

❖ تعریف امتیاز ترکیبی یک سند:

$$\text{net-score}(q, d) = g(d) + \cos(q, d)$$

- ❖ این مدل از فاصله زود هنگام جلوگیری می کند: ما مجبور نیستیم لیست پست ها را به طور کامل پردازش کنیم تا k top را پیدا کنیم.

ادامه...

❖ فرض کنید :

$g \rightarrow [0,1] - 1$  و  $g(d) < 0.1 - 2$  برای سند  $d$  که در حال حاضر در حال پردازش هستیم، 3 - کوچک ترین امتیاز  $k$  top که تاکنون پیدا کرده ایم 1.2 است.

❖ سپس تمام امتیازات بعدی از 1.1 بزرگ تر خواهند بود.

❖ بنابراین ما قبلا  $k$  top را پیدا کرده ایم و می توانیم پردازش بقیه پست لیست ها را متوقف کنیم.

## پردازش اسناد در یک زمان

❖ هم مرتب سازی docID و هم مرتب سازی PageRank یک ترتیب ثابت را بر روی اسناد موجود در لیست پست ها تحمیل می کنند.

❖ محاسبه کسینوس در این مدل سند در یک زمان است.

❖ ما قبل از شده به محاسبه امتیاز شباهت پرس و جو- سند از  $di+1$ ، امتیاز شباهت "پرسش-سند" سند  $di$  را محاسبه می کنیم.

❖ روش جایگزین: پردازش یک عبارت در لحظه

## لیست‌پست‌های مرتب شده بر اساس وزن

❖ ایده: پست‌هایی را که سهم کمی در امتیاز نهایی دارند پردازش نکنید.

❖ اسناد را در لیست‌پست‌ها بر اساس وزن مرتب کنید.

❖ ساده‌ترین حالت: وزن نرمال شده  $tf-idf$  (به ندرت انجام می‌شود: فشرده‌سازی سفت است)

❖ اسنادی که در  $top\ k$  قرار دارند احتمالاً در اوایل این لیست‌های مرتب شده وجود دارند.

❖ بعید است که خاتمه زودهنگام هنگام پردازش لیست‌پست‌ها،  $top\ k$  را تخییر دهد.

❖ ولی:

❖ ما دیگر ترتیب ثابتی از اسناد در لیست‌پست‌ها نداریم.

❖ ما دیگر نمی‌توانیم از پردازش اسناد در یک زمان استفاده کنیم.

## پردازش عبارت (اصطلاح) در یک زمان

❖ ساده ترین حالت: لیست پست‌های اولین عبارت پرس‌وجو را به طور کامل پردازش کنید.

❖ برای هر docID که با آن روبرو می شوید یک انباشتگر ایجاد کنید.

❖ سپس لیست پست‌های عبارت دوم پرس‌وجو را به طور کامل پردازش کنید، و به همین ترتیب...



## پردازش عبارت (اصطلاح) در یک زمان

COSINESCORE( $q$ )

- 1 *float* Scores[ $N$ ] = 0
  - 2 *float* Length[ $N$ ]
  - 3 **for each** query term  $t$
  - 4 **do** calculate  $w_{t,q}$  and fetch postings list for  $t$
  - 5     **for each** pair( $d, tf_{t,d}$ ) in postings list
  - 6         **do** Scores[ $d$ ] +=  $w_{t,d} \times w_{t,q}$
  - 7 Read the array Length
  - 8 **for each**  $d$
  - 9 **do** Scores[ $d$ ] = Scores[ $d$ ] / Length[ $d$ ]
  - 10 **return** Top  $k$  components of Scores[]
- The elements of the array "Scores" are called **accumulators**.

- ❖ برای وب (20 میلیارد سند)، آرایه ای از انباشتگرهای A در حافظه غیرممکن است.
- ❖ بنابراین: فقط برای اسنادی که در لیست پست‌ها وجود دارند، انباشتگر ایجاد کنید.
- ❖ این معادل است با: برای اسناد با امتیاز صفر، انباشتگر ایجاد نکنید. (به عنوان مثال، اسنادی که حاوی هیچ یک از اصطلاحات پرس‌وجو نیستند)

## مثالی از انباشتگر ها

BRUTUS  $\longrightarrow$ 

1 ,2	7 ,3	83 ,1	87 ,2	...
------	------	-------	-------	-----

CAESAR  $\longrightarrow$ 

1 ,1	5 ,1	13 ,1	17 ,1	...
------	------	-------	-------	-----

CALPURNIA  $\longrightarrow$ 

7 ,1	8 ,2	40 ,1	97 ,3
------	------	-------	-------

❖ For query: [Brutus Caesar]:

❖ فقط برای 1، 5، 7، 13، 17، 83، 87 انباشتگر نیاز داریم.

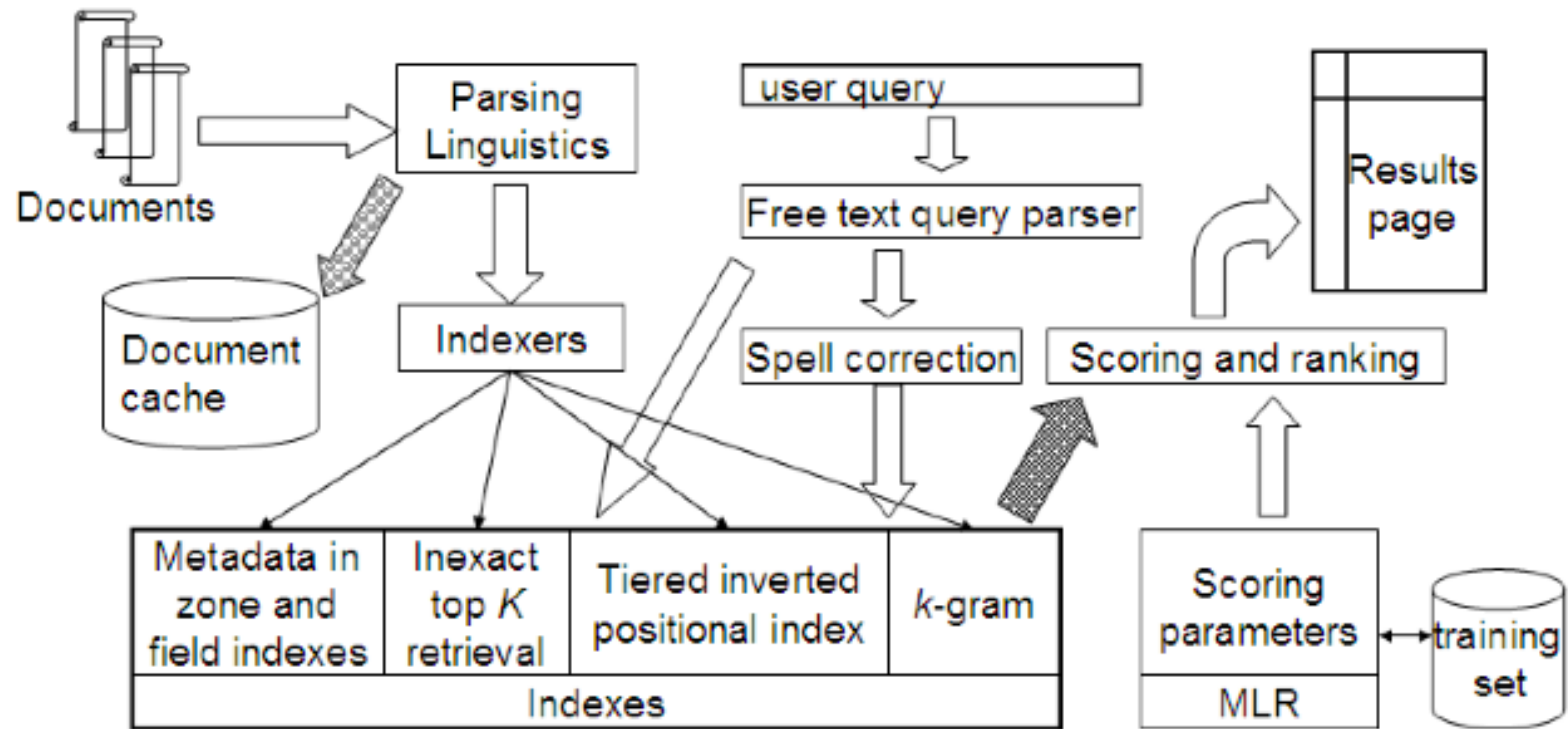
❖ برای 8، 40، 85 به انباشتگر نیاز نداریم.

❖ همانطور که قبلا توضیح داده شد از هیپ یا صف اولویت استفاده کنید.

❖ Can further limit to docs with non-zero cosines on rare (high idf) words

❖ یک جستجو پیوندی (a la Google) را اعمال کنید: non-zero cosines on all words in query

❖ مثال فقط یک انباشتگر برای [Brutus Caesar] در مثال بالا وجود دارد. زیرا فقط  $d1$  حاوی هر دو کلمه است.



❖ ایده پایه:

چندین ردیف از شاخص را مطابق با اهمیت اصطلاحات شاخص بندی ایجاد کنید.

در طول پردازش پرس و جو، با شاخص بالاترین سطح شروع کنید.

اگر شاخص بالاترین سطح حداقل (k به عنوان مثال،  $k = 100$ ) را برمی گرداند: توقف کنید و نتایج را به کاربر برگردانید.

اگر فقط ( $k >$ ) برخورد را پیدا کرده‌ایم: برای index in tier cascade بعدی تکرار کنید.

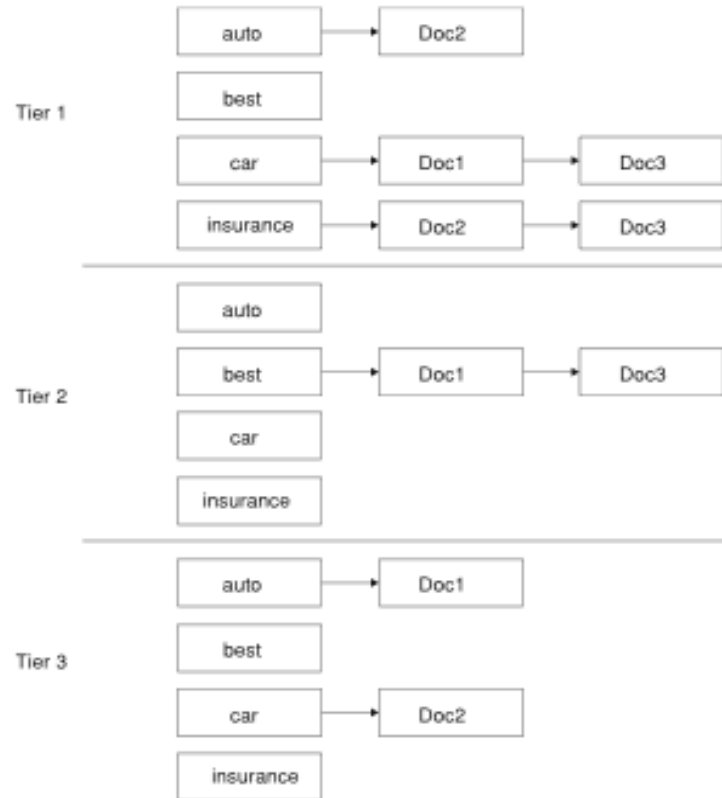
❖ مثال: سیستم دو ردیفی

ردیف 1: شاخص همه عناوین

ردیف 2: شاخص بقیه اسناد

صفحاتی که حاوی کلمات جستجو شده در عنوان هستند، نسبت به صفحاتی که کلمات جستجو شده در بدنه متن را شامل می شوند، برخوردهای (بازیابی) بهتری دارند.

## شاخص لایه‌ای



## شاخص لایه‌ای

❖ اعتقاد بر این است که استفاده از شاخص های لایه‌ای یکی از دلایلی است که کیفیت جستجوی گوگل در ابتدای سال (2000) به طور قابل توجهی بالاتر از کیفیت جستجوی رقبای بود.

❖ (به همراه PageRank، استفاده از anchor text و محدودیت های مجاورتی)

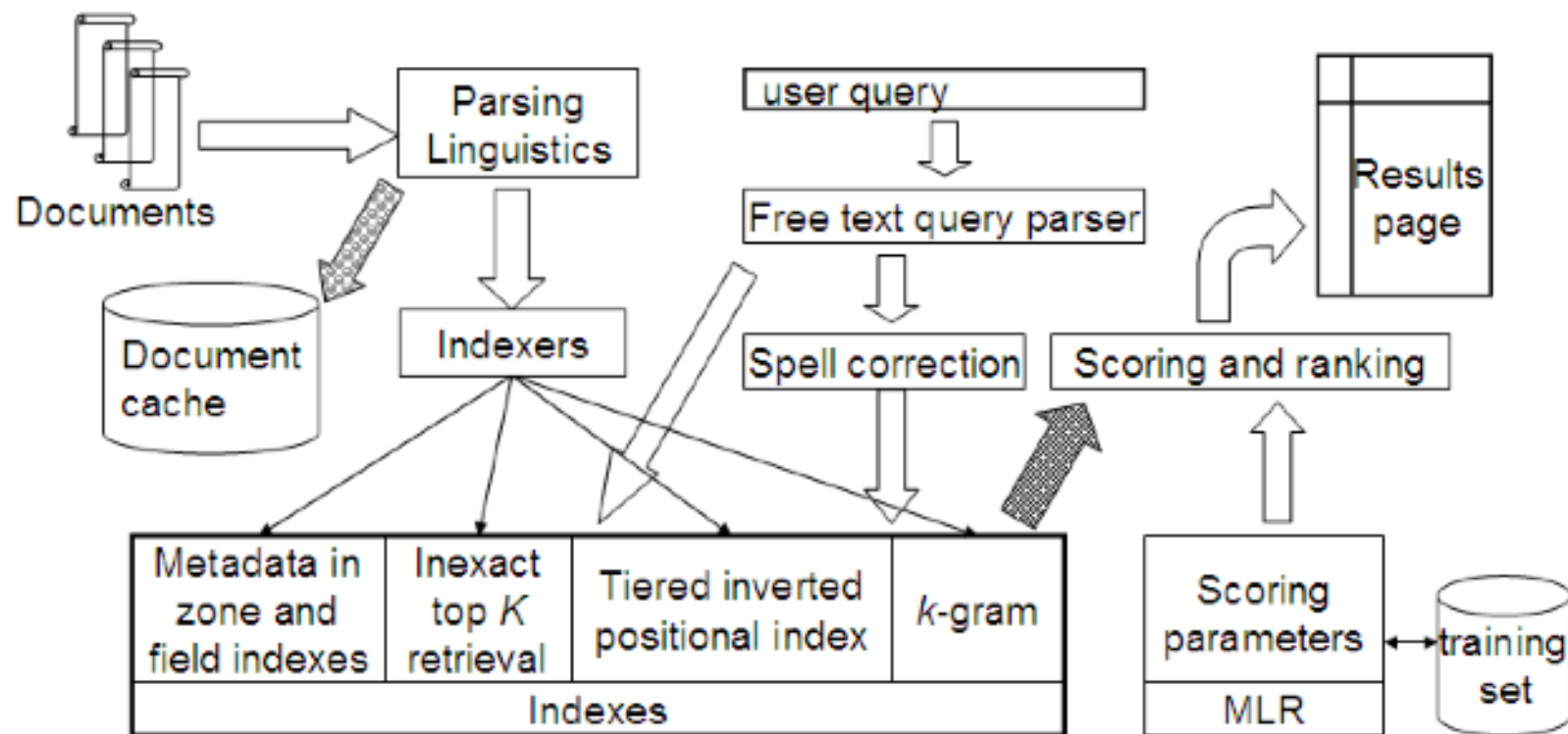


❖ معیارهای طراحی برای سیستم طبقه بندی شده (لایه‌ای):

- هر ردیف باید مرتبه ای کوچکتر از ردیف بعدی باشد.
- 100 برخورد (بازیابی) برتر برای اکثر پرس‌وجوها باید در ردیف 1 باشد، 100 بازیابی برتر برای اکثر پرس‌وجوهای باقیمانده در ردیف 2 و ...
- ما به یک تست ساده نیاز داریم که "آیا می‌توانم در این ردیف توقف کنم یا باید به مرحله بعدی بروم؟"
- هیچ مزیتی برای طبقه بندی وجود ندارد، اگر به هر حال مجبور باشیم برای اکثر پرس‌وجوها بیشتر ردیف‌ها را بازیابی کنیم.

❖ سوال 1: یک سیستم دو لایه‌ای را در نظر بگیرید که در آن ردیف اول عناوین و ردیف دوم همه چیز را نمایه می‌کند. مشکلات احتمالی این نوع لایه بندی چیست؟

❖ سوال 2: آیا می‌توانید راه بهتری برای راه اندازی یک سیستم چند لایه‌ای بیان کنید؟ کدام "منطقه" یک سند باید در سطوح مختلف (عنوان، بدنه سند، سایر موارد) شاقص بندی شود؟ برای گنجاندن یک سند در ردیف 1 از چه معیاری می‌فواهید استفاده کنید؟



❖ چگونه بازیابی عبارت را با بازیابی فضای برداری ترکیب کنیم؟

❖ ما نمی خواهیم فراوانی سند / idf را برای هر عبارت ممکن محاسبه کنیم. چرا؟

❖ چگونه بازیابی بولی را با بازیابی فضای برداری ترکیب کنیم؟

❖ به عنوان مثال: محدودیت های "+" و محدودیت های "-".

❖ Postfiltering ساده است، اما می تواند بسیار ناکارآمد باشد.

❖ چگونه wild cards را با بازیابی فضای برداری ترکیب کنیم؟



# تشکر

## سوال؟

[a.golzari@azaruniv.ac.ir](mailto:a.golzari@azaruniv.ac.ir)

[a.golzari@tabrizu.ac.ir](mailto:a.golzari@tabrizu.ac.ir)

<https://github.com/Amin-Golzari-Oskoue>