Quiz — 4 — Answer
MIR — Spring 2026
Instructor: Dr. Amin Eskandari
Head Teaching Assistants: Hamid Namjoo, Amir Hossein Hemati
Teaching Assistants: Ali Mojahed, Ali Nikvan, Amir Javvi

پاسخ سوال 1

قانون Zipf بیان می‌کند که فراوانی واژه‌ها در متن توزیع یکنواختی ندارد اگر واژه‌ها بر اساس تعداد تکرار مرتب شوند:

1) واژه رتبه اول بیشترین تکرار را دارد
2) واژه رتبه دوم تقریباً نصف آن تکرار می‌شود
3) واژه رتبه سوم کمتر از آن و به همین ترتیب ادامه پیدا می‌کند

رابطه تقریبی قانون Zipf <<< $f(r) \approx C / r$

که در آن $f(r)$ فراوانی واژه با رتبه r است و C یک ثابت است

نتیجه مهم قانون Zipf >>> تعداد کمی واژه بسیار پرتکرار هستن و تعداد بسیار زیادی واژه کم‌تکرار وجود دارد

در موتور جستجو این موضوع باعث می‌شود برخی Posting Listها بسیار بزرگ شوند و تعداد زیادی Posting List کوچک نیز وجود داشته باشد

قانون Heaps رشد تعداد واژه‌های یکتا را با افزایش تعداد اسناد توصیف می‌کند

فرمول >>> $V(N) = kN^{\beta}$

1) $V(N)$ تعداد واژه‌های یکتا است
2) N تعداد کل کلمات یا اسناد است
3) βمعمولاً بین 0.4 تا 0.7 قرار دارد

ویژگی مهم قانون Heaps >>> رشد واژگان زیرخطی است یعنی با افزایش اسناد، تعداد واژه‌های جدید با سرعت کمتری رشد می‌کند

در نتیجه >>> بیشتر واژه‌های جدید قبلاً در سیستم دیده شده‌اند و رشد Dictionary نسبت به رشد اسناد آهسته‌تر خواهد بود

Front Coding

در Dictionary بسیاری از واژه‌ها پیشوند مشترک دارند

مثال:

Compute , computer , computing , computation

در Front Coding پیشوند مشترک فقط یک بار ذخیره می‌شود و بقیه واژه‌ها فقط بخش متفاوت خود را ذخیره می‌کنند

مزیت اصلی >>> کاهش قابل توجه فضای Dictionary

Gap-based Posting List

در Posting List به جای ذخیره مستقیم DocIDها، اختلاف بین DocIDهای متوالی ذخیره می‌شود

مثال:

DocIDهای اصلی >>> 5 , 12 , 20 , 100

Gapها >>> 5 , 7 , 8 , 80

مزیت >>> اعداد کوچک‌تر می‌شوند و فشرده‌سازی بهتر انجام می‌شود

Gamma Encoding روشی برای فشرده‌سازی اعداد صحیح مثبت است

ویژگی اصلی >>> برای اعداد کوچک بسیار کارآمد است و برای اعداد بزرگ کارایی کاهش پیدا می‌کند

پس هرچه Gapها کوچک‌تر باشند Gamma Encoding بهتر عمل می‌کند

اثر ترتیب DocIDها

ترتیب تخصیص DocID تأثیر مستقیم بر اندازه Gapها دارد اگر DocIDها تصادفی باشند >>> Gapها ممکن است بسیار بزرگ شوند

اگر اسناد مشابه نزدیک هم قرار بگیرند >>> Gapها کوچک می‌شوند و فشرده‌سازی بسیار بهتر خواهد شد

بخش دوم تحلیل قسمت الف

رویا ادعا می‌کند که سه مشاهده مطرح‌شده با یکدیگر ناسازگار هستند و بنابراین حداقل یکی از قوانین Zipf یا Heaps برقرار نیست

این ادعا نادرست است ، هر سه مشاهده می‌توانند به صورت همزمان رخ دهند بدون آنکه هیچ‌یک از قوانین نقض شده باشند

تحلیل مشاهده اول >>> «اندازه Dictionary کمتر از انتظار رشد کرده است»

طبق قانون Heaps رشد تعداد واژه‌های یکتا زیرخطی است بنابراین با افزایش تعداد اسناد بیشتر واژه‌ها قبلاً دیده شده‌اند و تعداد کمی واژه جدید وارد سیستم می‌شود در نتیجه رشد Dictionary به صورت طبیعی آهسته خواهد بود

علاوه بر این، Front Coding نیز باعث کاهش بیشتر اندازه Dictionary می‌شود زیرا واژه‌های جدید معمولاً پیشوندهای مشترک دارند و فضای کمی اشغال می‌کنند پس کوچک بودن رشد Dictionary کاملاً طبیعی است و با قانون Heaps سازگار است

تحلیل مشاهده دوم >>> «اندازه Posting File بسیار بیشتر از انتظار رشد کرده است»

طبق قانون Zipf تعداد کمی واژه بسیار پرتکرار وجود دارند با اضافه شدن اسناد جدید همان واژه‌های پرتکرار بارها تکرار می‌شوند و در نتیجه Posting List این واژه‌ها بسیار بزرگ می‌شود بنابراین حتی اگر تعداد واژه‌های جدید کم باشد، تعداد Postingها می‌تواند بسیار زیاد افزایش پیدا کند در نتیجه رشد شدید Posting File کاملاً با قانون Zipf سازگار است

تحلیل مشاهده سوم >>> «Gamma Encoding برای برخی واژه‌های پرتکرار بدتر از واژه‌های کم‌تکرار عمل کرده است»

در نگاه اول انتظار داریم واژه‌های پرتکرار Gapهای کوچک‌تری داشته باشند و بهتر فشرده شوند اما عامل مهم دیگری وجود دارد ترتیب DocIDها ، اگر DocIDها بر اساس زمان ورود اسناد تخصیص داده شوند و اسناد مشابه کنار هم نباشند، Gapها ممکن است بسیار بزرگ شوند

مثال

DocIDها >>> 5 , 1200 , 35000 , 900000

Gapها >>> 5 , 1195 , 33800 , 865000

Gamma Encoding برای اعداد بزرگ کارایی ضعیفی دارد در مقابل، یک واژه کم‌تکرار ممکن است در اسناد نزدیک به هم ظاهر شود

500 , 501 , 503

Gapها >>> 500 , 1 , 2

در این حالت Gamma Encoding بسیار بهتر عمل می‌کند بنابراین ممکن است

1) واژه پرتکرار فشرده‌سازی ضعیفی داشته باشد
2) واژه کم‌تکرار فشرده‌سازی بهتری داشته باشد

این موضوع کاملاً ممکن است و تناقضی با قوانین Zipf و Heaps ندارد

جمع‌بندی قسمت الف

هر سه مشاهده می‌توانند همزمان رخ دهند

1) قانون Heaps باعث رشد آهسته واژگان می‌شود
2) Front Coding اندازه Dictionary را بیشتر کاهش می‌دهد
3) قانون Zipf باعث بزرگ شدن Posting List واژه‌های پرتکرار می‌شود
4) ترتیب نامناسب DocIDها باعث ایجاد Gapهای بزرگ می‌شود

5) Gamma Encoding برای Gapهای بزرگ عملکرد ضعیف‌تری دارد

بنابراین ادعای رویا نادرست است (ایشالا سری بعد 😓)

بخش سوم تحلیل قسمت ب

انتخاب بهترین تغییر معماری سه گزینه برای تغییر وجود دارد

1) تغییر روش مرتب‌سازی و تخصیص DocID
2) تغییر روش فشرده‌سازی Dictionary
3) تغییر روش Encoding مربوط به Posting Listها

بهترین گزینه >>> تغییر روش مرتب‌سازی و تخصیص DocID

دلیل انتخاب بیشترین فضای ایندکس معمولاً مربوط به Posting File است، نه Dictionary

طبق قانون Zipf >>> تعداد کمی واژه بسیار پرتکرار هستند و بخش بزرگی از Postingها را تشکیل می‌دهند

اگر ترتیب DocIDها به گونه‌ای تغییر کند که اسناد مشابه کنار هم قرار بگیرند

1) Gapها کوچک‌تر می‌شوند
2) Gamma Encoding بسیار مؤثرتر عمل می‌کند
3) اندازه Posting File به شدت کاهش پیدا می‌کند

مثال >>> قبل از مرتب‌سازی 5 , 1200 , 35000 , 900000

Gapهای بزرگ >>> فشرده‌سازی ضعیف >>> بعد از مرتب‌سازی 500 , 501 , 502 , 503

Gapهای کوچک >>> فشرده‌سازی بسیار قوی در نتیجه تغییر DocID Ordering بیشترین تأثیر را بر کاهش فضای نهایی ایندکس دارد

Trade-offهای این انتخاب

مزایا

1) کاهش شدید فضای Posting File
2) افزایش کارایی فشرده‌سازی
3) بهبود locality داده‌ها

معایب

1) پیچیده‌تر شدن فرآیند ساخت ایندکس
2) سخت‌تر شدن مدیریت اسناد جدید
3) نیاز به خوشه‌بندی یا مرتب‌سازی اسناد

چرا تغییر Dictionary Compression تأثیر کمتری دارد؟

طبق قانون Heaps رشد واژگان زیرخطی است در نتیجه Dictionary نسبت به Posting File کوچک‌تر است همچنین Front Coding در حال حاضر نیز فشرده‌سازی مؤثری انجام می‌دهد بنابراین حتی اگر روش فشرده‌سازی Dictionary بهتر شود، کاهش فضای کلی ایندکس محدود خواهد بود

چرا تغییر Encoding Posting List تأثیر کمتری دارد؟

مشکل اصلی سیستم، بزرگ بودن Gapها است اگر Gapها بزرگ باقی بمانند حتی روش‌های Encoding بهتر نیز تأثیر محدودی خواهند داشت در مقابل، تغییر DocID Ordering مستقیماً اندازه Gapها را کاهش می‌دهد وقتی Gapها کوچک شوند حتی Gamma Encoding ساده نیز عملکرد بسیار خوبی خواهد داشت بنابراین ریشه اصلی مشکل، ترتیب نامناسب DocIDها است نه خود الگوریتم Encoding!

پاسخ سوال 2

بخش اول درک مسئله:

در این سوال یک سیستم بازیابی اطلاعات داریم که رزومه‌ها را با استفاده از مدل برداری (Vector Space Model) رتبه‌بندی می‌کند

در این مدل هر سند و هر کوئری به صورت یک بردار از واژه‌ها نمایش داده می‌شود

وزن هر واژه معمولاً با فرمول زیر محاسبه می‌شود

tf-idf = tf × idf

tf >>> تعداد دفعات ظاهر شدن واژه در سند است

Idf >>> نشان می‌دهد یک واژه در کل مجموعه اسناد چقدر نادر است

فرمول

$$idf(t) = \log (N / df(t))$$

N = تعداد کل اسناد

df(t) = تعداد اسنادی که واژه در آن‌ها آمده

در این سؤال سیستم از Cosine Similarity برای مقایسه بردار سند و بردار کوئری استفاده می‌کند

فرمول شباهت

$$Similarity(Q,D) = (Q \cdot D) / (|Q| |D|)$$

این فرمول دو اثر مهم دارد

1) اسناد طولانی‌تر نرمال‌سازی می‌شوند

2) جهت بردار مهم است نه اندازه آن

ایندکس ناحیه‌ای (Zone Index)

سند به سه بخش تقسیم شده است

1) title

2) skills

3) experience

و هر ناحیه وزن متفاوت دارد

title = 5

skills = 3

experience = 1

یعنی اگر یک واژه در title بیاید اثر آن ۵ برابر experience است

کوئری

Q = deep learning for lung cancer diagnosis

واژه‌های مهم کوئری

lung , diagnosis , Deep , Learning , cancer

کلمات stopword مثل for اهمیت زیادی ندارند

بررسی اسناد

رزومه R1 : واژه deep learning بسیار زیاد تکرار شده است

رزومه R2 : این رزومه دقیقاً به موضوع تشخیص سرطان ریه مرتبط است

رزومه R3 : تمرکز روی AI عمومی است

سؤال 1) کدام رزومه ممکن است به‌صورت گمراه‌کننده رتبه اول بگیرد؟

پاسخ: احتمالاً R1

علت >>> در سیستم فعلی از tf خام استفاده می‌شود

در R1 واژه deep learning چندین بار در experience تکرار شده پس tf برای این واژه بزرگ می‌شود چون کوئری شامل deep و learning است، این واژه‌ها وزن زیادی می‌گیرند پس وزن واژه‌های مشترک بین Q و R1 زیاد می‌شود

در مدل برداری Dot Product بین Q و R1 بزرگ می‌شود در نتیجه شباهت cosine افزایش پیدا می‌کند پس R1 ممکن است رتبه بالایی بگیرد اما این رتبه تا حدی گمراه‌کننده است چون تمرکز کوئری روی lung cancer diagnosis است، نه فقط deep learning

سؤال 2

کدام رزومه معنایی نزدیک‌تر است ولی ممکن است رتبه پایین‌تری بگیرد؟

پاسخ >>> R2

علت : در R2 واژه‌های مهم کوئری وجود دارند مثل lung و cancer و diagnosis این واژه‌ها دقیقاً موضوع کوئری را بیان می‌کنند اما مشکل اینجاست در R2 این واژه‌ها کم تکرار شده‌اند پس tf آن‌ها کوچک است در نتیجه tf-idf آن‌ها نیز نسبتاً کوچک می‌شود در مقابل در R1 واژه deep learning چندین بار تکرار شده پس وزن آن بزرگ‌تر می‌شود

در نتیجه ممکن است سیستم به اشتباه R1 را بالاتر از R2 قرار دهد

سؤال 3

چگونه ترکیب tf خام، cosine normalization و وزن ناحیه‌ای باعث خطا می‌شود؟؟

مرحله اول tf خام باعث می‌شود که هرچه یک واژه بیشتر تکرار شود، وزن آن به صورت خطی افزایش یابد پس تکرار مصنوعی واژه‌ها باعث افزایش امتیاز می‌شود این همان مشکل keyword stuffing است

مرحله 2 وزن ناحیه‌ای

title = 5

skills = 3

experience = 1

اگر یک واژه مهم در skills یا title باشد، اثر زیادی دارد

در R1 ، deep learning در skills وجود دارد پس وزن آن بیشتر می‌شود

مرحله 3 Cosine Normalization

cosine similarity طول بردار را نرمال می‌کند یعنی

Similarity = dot product / (|Q||D|)

اگر سندی شامل تعداد زیادی واژه نامرتبط نباشد طول بردار آن خیلی بزرگ نمی‌شود در نتیجه بردار سندی که واژه‌های کوئری را زیاد تکرار کرده است ممکن است زاویه کوچک‌تری با Q داشته باشد پس Similarity افزایش می‌یابد

ترکیب این سه عامل باعث می‌شود:

1) tf خام >>> تکرار زیاد وزن را زیاد می‌کند
2) zone weighting >>> بعضی واژه‌ها بیش از حد تقویت می‌شوند
3) cosine normalization >>> اثر طول سند را تعدیل می‌کند

در نتیجه سندی که واژه‌ای را زیاد تکرار کرده است ممکن است امتیاز بالایی بگیرد حتی اگر از نظر معنایی بهترین سند نباشد

سؤال 4

آیا افزایش وزن title ممکن است کیفیت رتبه‌بندی را بدتر کند؟؟ بله

تحلیل >>> اگر weight(title) خیلی زیاد شود ، واژه‌های موجود در title اثر بسیار بزرگی خواهند داشت اما title معمولاً کوتاه است

مثال:

R3

title = AI Engineer

اگر کارفرما کوئری مرتبط با AI بدهد، وجود واژه AI در title ممکن است امتیاز بزرگی ایجاد کند حتی اگر بقیه رزومه ارتباط کمی با کوئری داشته باشد در نتیجه سیستم بیش از حد به title وابسته می‌شود و اطلاعات دقیق موجود در experience یا skills نادیده گرفته می‌شود پس افزایش بیش از حد وزن title ممکن است باعث bias در رتبه‌بندی شود

بخش ب

پیشنهاد 1 >>> استفاده از log(tf)

فرمول:

$$tf' = 1 + \log(tf)$$

اثر این تابع وقتی tf زیاد می‌شود، رشد وزن کند می‌شود

مثال:

$$tf = 1 \rightarrow 1$$

$$tf = 5 \rightarrow 1 + \log(5)$$

افزایش وزن دیگر خطی نیست در نتیجه تکرار زیاد یک واژه دیگر تأثیر بسیار بزرگی ندارد این روش مشکل keyword stuffing را کاهش می‌دهد

پیشنهاد 2 >>> Binary tf در skills

در binary weighting ، اگر واژه وجود داشته باشد tf = 1 و اگر وجود نداشته باشد tf = 0

اثر این روش این است که تکرار یک مهارت چندین بار دیگر امتیاز اضافی ایجاد نمی‌کند

مثال:

machine learning

اگر یک بار یا پنج بار نوشته شود وزن یکسان خواهد داشت ، این روش از تقلب جلوگیری می‌کند اما یک مشکل دارد

قدرت تمایز کاهش می‌یابد

مثلاً فردی که واقعاً تخصص زیادی دارد و واژه‌ای را چند بار به شکل طبیعی استفاده کرده است دیگر امتیاز بیشتری نمی‌گیرد

پیشنهاد 3 >>> حذف Cosine Normalization

اگر cosine حذف شود، شباهت تقریباً برابر می‌شود با Dot Product

یعنی Similarity ≈ Q · D

در این حالت اسناد طولانی‌تر معمولاً امتیاز بیشتری می‌گیرند زیرا واژه‌های بیشتری دارند در نتیجه رزومه‌های طولانی صرفاً به دلیل طول بیشتر امتیاز بالاتری می‌گیرند این یک bias به سمت اسناد بلند ایجاد می‌کند

سؤال 3 بخش ب

آیا ممکن است دو رزومه با شباهت معنایی متفاوت امتیاز یکسان بگیرند؟؟ بله

مثلا اگر از binary tf استفاده شود

فرض کنید دو رزومه:

رزومه A >>>

deep learning

lung

cancer

رزومه B >>>

deep learning

lung

cancer

اما رزومه A توضیح بسیار دقیق‌تری دارد چون binary tf فقط وجود واژه را بررسی می‌کند، هر دو سند وزن یکسانی برای این واژه‌ها می‌گیرند در نتیجه امتیاز آن‌ها تقریباً برابر می‌شود

سؤال 4 بخش ب

آیا حذف normalization می‌تواند وزن ناحیه‌ای را بی‌اثر یا اغراق‌آمیز کند؟؟ بله

اگر normalization حذف شود طول بردار سند دیگر کنترل نمی‌شود در نتیجه اگر یک سند طولانی باشد و چند واژه در title داشته باشد اثر weight(title) بسیار بزرگ می‌شود چون مقدار dot product بدون محدودیت افزایش می‌یابد

پس ترکیب zone weight + عدم normalization می‌تواند باعث اغراق در امتیاز بعضی اسناد شود

سؤال 5

اگر فقط یک تغییر مجاز باشد بهترین انتخاب چیست؟

بهترین انتخاب استفاده از log(tf) هست

علت:

1) مشکل اصلی سیستم تکرار زیاد واژه‌ها است
2) log(tf) رشد وزن را کنترل می‌کند
3) در عین حال اطلاعات فراوانی واژه را حفظ می‌کند
4) برخلاف binary tf، قدرت تمایز از بین نمی‌رود
5) برخلاف حذف bias، normalization جدید ایجاد نمی‌کند

بنابراین log(tf) یک راه‌حل متعادل برای کاهش keyword stuffing است

پاسخ سوال 3

بخش اول درک مسئله و هدف طراحی سیستم

در این مسئله هدف طراحی یک سیستم بازیابی اطلاعات است که بتواند اسناد مرتبط را به شکل دقیق و کارا رتبه‌بندی کند. چالش اصلی این است که مجموعه اسناد شامل تعداد زیادی سند بازنشر شده یا نزدیک به بازنشر (Near-Duplicate) است چنین اسنادی معمولاً شامل تمام واژه‌های پرس‌وجو هستند و بنابراین در معیارهای ساده‌ای مانند شباهت برداری (Cosine Similarity) امتیاز بالایی می‌گیرند، اما از نظر معنایی ارزش یکسانی ندارند

در بسیاری از موارد مقاله اصلی واژه‌های مهم پرس‌وجو را در یک بخش مشخص و مرتبط از متن به کار می‌برد، در حالی که نسخه‌های بازنشر شده ممکن است همان واژه‌ها را در بخش‌های مختلف متن پراکنده کرده باشند. بنابراین یک سیستم رتبه‌بندی مناسب باید علاوه بر شباهت واژگانی به نحوه‌ی توزیع مکانی واژه‌ها نیز توجه کند

در عین حال دسترسی به اطلاعات موقعیت واژه‌ها هزینه‌ی محاسباتی بالایی دارد به همین دلیل سیستم از یک ساختار ایندکس چندلایه استفاده می‌کند تا بتواند ابتدا اسناد غیرمرتبط را با هزینه‌ی کم حذف کند و تنها در مراحل بعدی از اطلاعات پیچیده‌تر استفاده نماید.

بخش دوم نقش ایندکس چندلایه در سیستم بازیابی اطلاعات

در این سیستم از سه لایه ایندکس استفاده شده است که هر کدام سطح متفاوتی از اطلاعات را در اختیار سیستم قرار می‌دهند

لایه اول >>> لایه وجود یا عدم وجود واژه‌ها (Boolean Layer)

در این لایه فقط مشخص می‌شود که آیا یک واژه در سند وجود دارد یا خیر این اطلاعات بسیار فشرده و سریع قابل دسترسی است بنابراین از این لایه برای حذف اولیه اسناد کاملاً نامرتبط استفاده می‌شود اگر سندی هیچ‌یک از واژه‌های پرس‌وجو را نداشته باشد، در همین مرحله کنار گذاشته می‌شود

لایه دوم >>> لایه فراوانی واژه‌ها (Term Frequency Layer)

در این لایه تعداد تکرار هر واژه در سند ذخیره می‌شود این اطلاعات برای محاسبه‌ی شباهت برداری بسیار مفید است الگوریتم Fast Cosine Score در این مرحله استفاده می‌شود تا شباهت اولیه بین پرس‌وجو و اسناد محاسبه شود این مرحله هنوز از نظر محاسباتی نسبتاً ارزان است و می‌تواند روی تعداد زیادی سند اعمال شود

لایه سوم >>> لایه موقعیت واژه‌ها (Positional Layer)

در این لایه مکان دقیق هر واژه در سند ذخیره شده این اطلاعات امکان محاسبه‌ی فاصله بین واژه‌ها و در نتیجه محاسبه‌ی معیار Proximity را فراهم می‌کند اما دسترسی به این لایه پرهزینه است و نباید برای همه اسناد استفاده شود بنابراین تنها اسناد امیدوارکننده به این مرحله وارد می‌شوند

بخش سوم جایگاه الگوریتم Fast Cosine Score در سیستم

Fast Cosine Score یک روش سریع برای محاسبه‌ی شباهت برداری بین پرس‌وجو و اسناد است این روش از اطلاعات فراوانی واژه‌ها استفاده می‌کند و طول سند را نیز در نظر می‌گیرد تا اسناد بسیار بلند به صورت ناعادلانه امتیاز بالاتر نگیرند هدف اصلی استفاده از این الگوریتم در سیستم انجام یک غربال اولیه است در این مرحله اسنادی که از نظر واژگانی شباهت بسیار کمی به پرس‌وجو دارند حذف می‌شوند و تنها اسنادی که پتانسیل مرتبط بودن دارند باقی می‌مانند با این حال باید توجه داشت که Cosine Similarity به ترتیب یا فاصله واژه‌ها توجه نمی‌کند بنابراین ممکن است اسناد Near-Duplicate که واژه‌ها را در بخش‌های مختلف متن پخش کرده‌اند، امتیاز بالایی بگیرند به همین دلیل این معیار به تنهایی برای رتبه‌بندی نهایی کافی نیست.

بخش چهارم طراحی جریان پردازش یا Pipeline رتبه‌بندی

یک طراحی مناسب برای pipeline سیستم باید چند مرحله داشته باشد تا هم کارایی و هم دقت سیستم حفظ شود

مرحله اول >>> فیلتر اولیه بر اساس وجود واژه‌ها

در این مرحله سیستم با استفاده از لایه اول ایندکس بررسی می‌کند که کدام اسناد حداقل بخشی از واژه‌های پرس‌وجو را دارند. اسنادی که هیچ‌یک از واژه‌ها را ندارند حذف می‌شوند این مرحله بسیار سریع است و حجم اسناد را کاهش می‌دهد

مرحله دوم >>> محاسبه Fast Cosine Score

در این مرحله با استفاده از اطلاعات لایه دوم ایندکس شباهت برداری بین پرس‌وجو و اسناد محاسبه می‌شود هدف این مرحله ایجاد یک مجموعه از اسناد کاندیدا است

نکته مهم این است که در این مرحله نباید از یک آستانه بسیار سخت استفاده کرد. اگر تنها اسناد با cosine بسیار بالا انتخاب شوند، ممکن است اسنادی که cosine متوسط اما نزدیکی واژه‌های بسیار خوبی دارند حذف شوند

مرحله سوم >>> انتخاب اسناد امیدوارکننده

پس از محاسبه cosine مجموعه‌ای از اسناد که احتمال مرتبط بودن دارند انتخاب می‌شوند این مجموعه ممکن است شامل اسنادی با cosine بالا و همچنین برخی اسناد با cosine متوسط باشد هدف این است که هیچ سند بالقوه مهمی پیش از بررسی proximity حذف نشود

مرحله چهارم >>> محاسبه Proximity

در این مرحله سیستم به لایه سوم ایندکس مراجعه می‌کند و فاصله بین واژه‌های پرس‌وجو در اسناد کاندیدا را بررسی می‌کند. اگر واژه‌ها در یک بخش نزدیک از سند ظاهر شوند امتیاز proximity بالا خواهد بود اگر واژه‌ها در بخش‌های مختلف متن پراکنده باشند، امتیاز proximity پایین خواهد بود این مرحله کمک می‌کند تا اسناد اصلی که واژه‌ها را در یک زمینه معنایی مشترک به کار برده‌اند از اسناد بازنشرشده که واژه‌ها را پراکنده دارند متمایز شوند

مرحله پنجم >>> رتبه‌بندی نهایی

در مرحله نهایی امتیاز شباهت برداری و امتیاز proximity با یکدیگر ترکیب می‌شوند و رتبه نهایی اسناد تعیین می‌شود

بخش پنجم

اگر سیستم به شکل ساده طراحی شود، مشکلات مهمی ایجاد خواهد شد اگر تنها از Cosine Similarity استفاده شود اسناد بازنشر شده که فقط واژه‌های پرس‌وجو را تکرار کرده‌اند ممکن است در رتبه‌های بالا قرار بگیرند اگر proximity برای همه اسناد محاسبه شود هزینه محاسباتی سیستم بسیار بالا خواهد رفت و سیستم از نظر عملی قابل استفاده نخواهد بود اگر تنها اسناد با بالاترین cosine وارد مرحله proximity شوند ممکن است اسنادی که cosine متوسط اما نزدیکی واژه‌های بسیار خوبی دارند حذف شوند بنابراین تنها راه‌حل مناسب استفاده از یک pipeline چندمرحله‌ای است که در آن هر مرحله تعداد اسناد را کاهش داده و اطلاعات دقیق‌تری برای رتبه‌بندی فراهم کند

بخش ششم طراحی تابع تحلیل (Analysis Function)

تابع تحلیل وظیفه استخراج ویژگی‌ها از پرس‌وجو و اسناد را بر عهده دارد این تابع تصمیم‌گیری نمی‌کند بلکه اطلاعات لازم برای مرحله امتیازدهی را فراهم می‌کند در این سیستم تابع تحلیل باید اطلاعات زیر را استخراج کند

از پرس‌وجو >>> واژه‌های تشکیل‌دهنده پرس‌وجو ، اهمیت نسبی هر واژه ، طول پرس‌وجو

از سند >>> تعداد تکرار هر واژه در سند ، طول سند ، موقعیت دقیق واژه‌ها در متن ، الگوی توزیع واژه‌ها در طول سند

این اطلاعات پایه لازم برای محاسبه شباهت برداری و همچنین تحلیل proximity را فراهم می‌کنند

بخش هفتم تشخیص اسناد Near-Duplicate

اسناد Near-Duplicate معمولاً دارای ویژگی‌های زیر هستن

1) آن‌ها اغلب تمام واژه‌های پرس‌وجو را دارند و بنابراین cosine بالایی می‌گیرند
2) اما واژه‌ها در بخش‌های مختلف سند پراکنده هستند و در یک زمینه معنایی مشترک ظاهر نمی‌شوند

در مقابل سند اصلی معمولاً واژه‌های پرس‌وجو را در یک بخش متمرکز از متن به کار می‌برد بنابراین فاصله بین واژه‌ها کم است و امتیاز proximity بالایی دریافت می‌کند استفاده از اطلاعات مکانی واژه‌ها به سیستم کمک می‌کند تا این تفاوت را تشخیص دهد

بخش هشتم طراحی تابع امتیازدهی نهایی (Scoring Function)

تابع امتیازدهی نهایی باید چند عامل را به طور همزمان در نظر بگیرد

1) شباهت برداری بین پرس‌وجو و سند
2) نزدیکی واژه‌های پرس‌وجو در متن سند
3) الگوی توزیع واژه‌ها در سند

در این تابع هیچ‌کدام از این عوامل نباید به تنهایی تعیین‌کننده باشند اگر تنها cosine در نظر گرفته شود اسناد بازنشر شده امتیاز بالایی خواهند گرفت اگر تنها proximity در نظر گرفته شود ممکن است در اسناد بسیار طولانی فاصله‌های کوچک تصادفی باعث امتیاز بالا شوند بنابراین ترکیب متعادل این عوامل باعث می‌شود اسناد واقعاً مرتبط رتبه بالاتری بگیرند

بخش نهم استفاده کنترل‌شده از لایه موقعیتی

از آنجا که دسترسی به اطلاعات موقعیتی پرهزینه است سیستم باید تنها برای اسناد امیدوارکننده به این لایه مراجعه کند این اسناد معمولاً پس از مرحله cosine انتخاب می‌شوند با این روش تنها بخش کوچکی از اسناد نیاز به بررسی دقیق proximity خواهند داشت و هزینه محاسباتی سیستم کنترل می‌شود

پاسخ سوال 4

هدف این مسئله بررسی و ارزیابی عملکرد دو سیستم بازیابی اطلاعات است. در حوزه بازیابی اطلاعات برای سنجش کیفیت سیستم‌ها معمولاً از دو دسته معیار استفاده می‌شود ، دسته اول معیارهای کلاسیک بازیابی بدون رتبه هستند که شامل F1 ، Recall ، Precision و Accuracy می‌شوند این معیارها صرفاً بر اساس تعداد اسناد مرتبط و نامرتبط بازیابی‌شده محاسبه می‌شوند و ترتیب نمایش نتایج را در نظر نمی‌گیرن

دسته دوم معیارهای ارزیابی رتبه‌بندی هستند در سیستم‌های جستجو مانند موتورهای جستجو ترتیب نمایش نتایج بسیار مهم است زیرا کاربران معمولاً فقط چند نتیجه اول را مشاهده می‌کنند. معیارهایی مانند MAP ، R-Precision Precision@k و NDCG برای ارزیابی کیفیت رتبه‌بندی نتایج استفاده می‌شوند

در ادامه ابتدا سیستم A با معیارهای کلاسیک بررسی می‌شود، سپس سیستم B با معیارهای رتبه‌بندی ارزیابی می‌شود و در پایان رابطه بین معیارهای عددی و تجربه واقعی کاربران تحلیل خواهد شد

بخش اول: ارزیابی سیستم A با معیارهای کلاسیک

اطلاعات داده شده برای سیستم A به صورت زیر است

1. تعداد کل اسناد بازیابی شده 30
2. تعداد اسناد مرتبط در میان نتایج بازیابی شده 15
3. تعداد کل اسناد مرتبط واقعی در مجموعه 20

ماتریس خطا (Confusion Matrix): در مسائل طبقه‌بندی و بازیابی اطلاعات برای تحلیل عملکرد سیستم از چهار نوع نتیجه استفاده می‌شود

True Positive: اسناد مرتبطی که سیستم آن‌ها را بازیابی کرده است

False Positive: اسناد نامرتبطی که سیستم به اشتباه بازیابی کرده است

False Negative: اسناد مرتبطی که سیستم نتوانسته آن‌ها را بازیابی کند

True Negative: اسناد نامرتبطی که سیستم آن‌ها را بازیابی نکرده است

محاسبه مقادیر ماتریس خطا

True Positive تعداد اسناد مرتبط بازیابی شده برابر است با TP = 15

False Positive تعداد اسناد نامرتبط بازیابی شده برابر است با تعداد کل اسناد بازیابی شده - تعداد اسناد مرتبط بازیابی شده FP = 30 – 15 = 15

False Negative تعداد اسناد مرتبطی که بازیابی نشده‌اند برابر است با تعداد کل اسناد مرتبط - اسناد مرتبط بازیابی شده FN = 20 - 15 = 5

True Negative برای محاسبه TN باید اندازه کل مجموعه اسناد مشخص باشد چون این مقدار در صورت سؤال داده نشده است مقدار TN قابل محاسبه نیست.

Precision نشان می‌دهد چه نسبتی از اسناد بازیابی شده واقعاً مرتبط هستند. این معیار میزان دقت سیستم در ارائه نتایج مرتبط را اندازه‌گیری می‌کند

Precision = TP / (TP + FP)

جایگذاری مقادیر >>>

Precision = 15 / (15 + 15)

Precision = 15 / 30

Precision = 0.5

بنابراین مقدار Precision برای سیستم A برابر 0.5 است این مقدار نشان می‌دهد که تنها نیمی از اسناد بازیابی شده واقعاً مرتبط بوده‌اند

Recall نشان می‌دهد چه نسبتی از کل اسناد مرتبط موجود در مجموعه توسط سیستم بازیابی شده‌اند. این معیار توانایی سیستم در پیدا کردن اسناد مرتبط را اندازه‌گیری می‌کند

Recall = TP / (TP + FN)

جایگذاری >>>

Recall = 15 / (15 + 5)

Recall = 15 / 20

Recall = 0.75

بنابراین مقدار Recall برای سیستم A برابر 0.75 است این مقدار نشان می‌دهد که سیستم توانسته است 75 درصد از اسناد مرتبط موجود را بازیابی کند

F1 Score >>> در بسیاری از موارد لازم است بین Precision و Recall تعادل برقرار شود برای این منظور از معیار F1 استفاده می‌شود که میانگین هارمونیک Precision و Recall است

فرمول F1 به صورت زیر است

F1 = 2 × (Precision × Recall) / (Precision + Recall)

جایگذاری مقادیر >>>

Precision = 0.5

Recall = 0.75

F1 = 2 × (0.5 × 0.75) / (0.5 + 0.75)

F1 = 2 × 0.375 / 1.25

F1 = 0.75 / 1.25

F1 = 0.6

بنابراین مقدار F1 برای سیستم A برابر 0.6 است

Accuracy نشان می‌دهد چه نسبتی از کل پیش‌بینی‌ها درست بوده‌اند

فرمول Accuracy به صورت زیر است

Accuracy = (TP + TN) / (TP + TN + FP + FN)

اما چون مقدار True Negative و اندازه کل مجموعه اسناد در صورت سؤال مشخص نشده است مقدار Accuracy در این مسئله قابل محاسبه نیست این موضوع در مسائل بازیابی اطلاعات رایج است زیرا تعداد اسناد نامرتبط معمولاً بسیار زیاد است

جمع‌بندی نتایج سیستم A

Precision = 0.5

Recall = 0.75

F1 = 0.6

Accuracy به دلیل نامشخص بودن اندازه کل مجموعه قابل محاسبه نیست

بخش دوم: ارزیابی سیستم B با معیارهای رتبه‌بندی

در سیستم B نتایج به صورت رتبه‌بندی شده ارائه شده‌اند. وضعیت مرتبط بودن 10 نتیجه اول به شکل زیر است

رتبه‌ها >>> 1 ، 2 ، 3 ، 4 ، 5 ، 6 ، 7 ، 8 ، 9 ، 10

مرتبط بودن اسناد >>> 0 ، 1 ، 1 ، 0 ، 1 ، 0 ، 0 ، 1 ، 0 ، 1

بنابراین اسناد مرتبط در رتبه‌های زیر قرار دارند:

رتبه 2

رتبه 3

رتبه 5

رتبه 8

رتبه 10

Precision@5 نشان می‌دهد چه نسبتی از پنج نتیجه اول مرتبط هستند

پنج نتیجه اول عبارتند از 0 ، 1 ، 1 ، 0 ، 1 >>> در این میان سه سند مرتبط وجود دارد

بنابراین

Precision@5 = 3 / 5 >>> Precision@5 = 0.6

این مقدار نشان می‌دهد که 60 درصد از پنج نتیجه اول مرتبط هستند

در معیار R-Precision مقدار R برابر با تعداد کل اسناد مرتبط واقعی در مجموعه است

در این مسئله R = 20 بنابراین باید Precision را در 20 نتیجه اول محاسبه کنیم ، اما در صورت سؤال فقط وضعیت 10 نتیجه اول داده شده است و اطلاعاتی درباره رتبه‌های 11 تا 20 وجود ندارد پس مقدار دقیق R-Precision قابل محاسبه نیست

برای محاسبه Average Precision باید Precision را در تمام رتبه‌هایی که سند مرتبط ظاهر شده است محاسبه کنیم و سپس میانگین آن‌ها را بگیریم

رتبه‌های اسناد مرتبط عبارتند از 2 ، 3 ، 5 ، 8 ، 10 و Precision در رتبه 2 ، تا این رتبه یک سند مرتبط دیده شده است

Precision = 1 / 2 = 0.5

Precision در رتبه 3 ، تا این رتبه دو سند مرتبط دیده شده است

Precision = 2 / 3 ≈ 0.667

Precision در رتبه 5 ، تا این رتبه سه سند مرتبط دیده شده است

Precision = 3 / 5 = 0.6

Precision در رتبه 8 ، تا این رتبه چهار سند مرتبط دیده شده است

Precision = 4 / 8 = 0.5

Precision در رتبه 10 ، تا این رتبه پنج سند مرتبط دیده شده است

Precision = 5 / 10 = 0.5

جمع مقادیر Precision برابر است با:

0.5 + 0.5 + 0.5 + 0.6 + 0.667 =2.767

فرمول Average Precision

AP = مجموع Precisionها / تعداد کل اسناد مرتبط

AP = 2.767 / 20

AP ≈ 0.138

از آنجا که فقط یک پرس‌وجو داریم، مقدار MAP نیز برابر با همین مقدار خواهد بود

معیار NDCG کیفیت رتبه‌بندی را با در نظر گرفتن جایگاه اسناد مرتبط اندازه‌گیری می‌کند در این معیار اسنادی که در رتبه‌های بالاتر قرار دارند ارزش بیشتری دارند ، ابتدا مقدار DCG محاسبه می‌شود. با توجه به رتبه‌های اسناد مرتبط داریم >>>

DCG ≈ 0.631 + 0.5 + 0.387 + 0.315 + 0.289

DCG ≈ 2.122

سپس مقدار ایده‌آل یعنی IDCG محاسبه می‌شود ، در حالت ایده‌آل پنج سند مرتبط باید در پنج رتبه اول قرار بگیرند

IDCG ≈ 1 + 0.631 + 0.5 + 0.431 + 0.387

IDCG ≈ 2.949

در نهایت >>> NDCG = DCG / IDCG

NDCG ≈ 2.122 / 2.949

NDCG ≈ 0.72

بخش سوم تحلیل تفاوت معیارهای آفلاین و رضایت کاربران

ممکن است یک سیستم در معیارهای آفلاین مانند MAP یا NDCG عملکرد نسبتاً خوبی داشته باشد اما کاربران در عمل از آن رضایت کافی نداشته باشند ، یکی از مهم‌ترین دلایل این موضوع رفتار واقعی کاربران است کاربران معمولاً فقط چند نتیجه اول را مشاهده می‌کنند و اغلب روی اولین یا دومین نتیجه کلیک می‌کنند بنابراین اگر نتیجه اول نامرتبط باشد، تجربه کاربری به شدت تحت تأثیر قرار می‌گیرد ، دلیل دیگر این است که معیارهایی مانند MAP کیفیت کل رتبه‌بندی را در نظر می‌گیرند و ممکن است خطاهای مهم در ابتدای لیست نتایج را کمتر منعکس کنند

همچنین snippet یا خلاصه‌ای که زیر هر نتیجه نمایش داده می‌شود می‌تواند باعث ایجاد برداشت نادرست در کاربر شود. اگر snippet شامل کلمات پرس‌وجو باشد کاربر ممکن است تصور کند سند مرتبط است و روی آن کلیک کند در حالی که محتوای واقعی صفحه با نیاز او همخوانی نداشته باشد ، به همین دلیل در ارزیابی سیستم‌های بازیابی اطلاعات علاوه بر معیارهای آفلاین، استفاده از آزمایش‌های کاربری و تحلیل رفتار کاربران نیز اهمیت زیادی دارد.

موفق و موید:(