زهرا گنجی 9531802

گزارش مرحله دوم پروژه بازیابی اطلاعات

- مدل سازی اسناد در فضای برداری

برای بازنمایی اسناد به روش بردارهای وزن tf-idf کلمات، به دو پارامتر tf یا تعداد تکرار هر کلمه در هر سند و df یا تعداد اسنادی که هر کلمه در آن ها ظاهر شده است نیاز داریم.

:Tf

در هنگام پردازش اسناد در تابع create_index، برای هر سند دیکشنری termid_freq={termid:freq,termid2:freq2,...} که تعداد تکرار کلمات موجود در هر سند را نشان می دهد، بدست می آوریم. سپس هر کدام از این دیکشنری ها را به عنوان value در دیکشنری termid_freq_perdoc قرار می دهیم.در این دیکشنری keyها، docid ها هستند قرار می دهیم

termid_freq_perdoc = {docid:{termid:freq,termid2:freq2,...},docid2:{termid:freq,termid2:freq2,...}

پس بدین طریق تعداد تکرار هر کلمه در هر سند را در اختیار داریم.

:Df

در هنگام پردازش اسناد در تابع create_index، در صورت مشاهده شدن هر کلمه در هر سند، به تعداد اسنادی که کلمه در آن ها آمده (document frequency) اضافه می کنیم و در نهایت تعداد اسنادی که هر کلمه در آن ها قرار گرفته را در دیکشنری document_freq در اختیار داریم.

Document_freq={termid: df,termid2:df2,...}

بعد از بدست آوردن دو پارامتر Tf و df برای هر کلمه و سند، دو دیکشنری termid_freq_perdoc و document_freq را به تابع tf-idf برای محاسبه ی وزن هر کلمه در هر سند می دهیم.

در تابع tf_idf ، در یک حلقه به ازای هر docid، تعداد تکرار هر کلمه ی آن را از دیکشنری termid_freq_perdoc استخراج می کنیم و در tf قرار می دهیم، و تعداد اسنادی که هر کلمه در آن ها ظاهر شده است را نیز از document_freq استخراج می کنیم و در df قرار می دهیم و با فرمول زیر وزن هر کلمه را در سند مربوطه بدست می آوریم.

$$w_{t,d} = \log(1+\mathrm{tf}_{t,d}) \times \log_{10}(N/\mathrm{df}_t)$$

سپس این وزن ها را در دیکشنری tf_idf_weights قرار می دهیم.

Tf_idf_weights={docid:{termid:w,termid2: w2,…},docid2:{termid:w,termid2:w2,…},..}

برای آنکه از به کار بردن فضای بیش از حد در بازنمایی اسناد جلوگیری شود، تنها کلماتی در دیکشنری termid_freq_perdoc تعداد تکرارشان برای هر docid آمده است که در آن docid حداقل یک بار تکرار شده باشند. پس در هنگام محاسبه ی tf-idf چون بر اساس این دیکشنری کلمات موجود برای هر docid وزنشان محاسبه می شود، پس کلماتی که در یک doc وجود ندارند وزنشان محاسبه نمی شود و در بازنمایی اسناد نمی آیند. به این تکنیک index elimination گفته می شود.

خروجی: به دلیل زیاد بودن زمان اجرا برای همه ی خبر ها، 300 خبر از هر فایل خواندم و برنامه را روی آن ها اجرا کردم.

5 کلمه ی با بیش ترین و کم ترین وزن در سند شماره 2:

هر چه وزن یک کلمه در سند بیش تر باشد به دو صورت تعبیر می شود:

Tf آن کلمه در آن سند بالا بوده است یعنی کلمه با تعداد تکرار بیش تری در سند ظاهر می شود. هر چه تکرار یک کلمه در یک سند بالاتر باشد وزن آن کلمه در آن سند بالاتر می رود یعنی احتمال این که آن سند به آن کلمه مرتبط باشد بیش تر می شود پس اهمیت کلمه در آن بیش تر است.

Df کلمه کم تر بوده است یا idf آن بیش تر بوده است. یعنی کلمه در اسناد کم تری تکرار شده است که این به معنی خاص تر بودن کلمه است. کلمات رایج که در بیش تر اسناد ظاهر می شوند وزن کلمه در سند را افزایش نمی دهند یعنی احتمال این که این سند به آن کلمه ی خاص مرتبط باشد زیاد نیست. در صورتی که وقتی کلمات خاص در یک سند قرار می گیرند احتمال این که آن سند به آن کلمه مرتبط باشد بیش تر است پس اهمیت کلمه در آن بیش تر است.

- پاسخگویی به پرسمان در فضای برداری

در این بخش ابتدا بردار مخصوص پرسمان ورودی را در تابع tf-idf محاسبه می کنیم. به این صورت که تمام کلمات پرسمان را استخراج کرده و تعداد تکرار هر کلمه در پرسمان را بدست می آوریم و در دیکشنری tf_query قرار می دهیم: tf_query = {termed:tf in query,termid2:tf2 in query,...}

و با استفاده از df آن کلمه که از قبل در دیکشنری document_freq داشتیم، وزن کلمات موجود در پرسمان را بدست می آوریم و در دیکشنری query_weights قرار می دهیم. سایز بردار پرسمان را نیز محاسبه می کنیم و در query_size برای استفاده در محاسبه ی شباهت کسینوسی قرار می دهیم.

Tf_query شامل termid تمام کلمات موجود در پرسمان می باشد. پس با یک حلقه در این دیکشنری ، برای تک تک کلمات موجود در پرسمان، ابتدا وزن کلمه در پرسمان را از query_weight بدست آورده و در w_t_q قرار می دهیم و سپس documentهایی که کلمه مورد نظر در آن ها حداقل یک بار ظاهر شده است را با استخراج postings list آن ها بدست می آوریم. به این تکنیک index elimination می گوییم یعنی تنها اسنادی را شباهتشان را با پرسمان محاسبه می کنیم که حداقل یک کلمه ی مشترک با پرسمان داشته باشند و این باعث کم تر شدن محاسبات می شود.

بعد از بدست آوردن این اسناد، وزن کلمه ی مورد نظر در آن ها را با استفاده از دیکشنری tf_idf_weights استخراج می کنیم و در w_t_d ذخیره می کنیم و w_t_q*w_t_d را بدست می آوریم و در دیکشنری doc_scores ذخیره می کنیم.

Doc_scores = {docid:score,docid2:score2,...}

این کار را برای تمام کلمات در پرسمان تکرار می کنیم و هر بار برای یک سند مشابه ، w_t_d*w_t_q جدید را با doc_scores[docid] قبلی جمع می کنیم. به این روش term at a time می گویند و تا آخر حلقه ی روی کلمات پرسمان امتیاز docها به صورت کامل محاسبه نشده است. بعد از تمام شدن این حلقه ی for، همه ی امتیاز ها را تقسیم بر document_size[docid]*query_size می کنیم تا شباهت کسینوسی نهایی بدست آید.

سپس از دیکشنری doc_scores با استفاده از heap، 10 سند با بیش ترین امتیاز را انتخاب کرده و آن ها را به کاربر نمایش می دهیم.

- افزایش سرعت پردازش پرسمان

برای ساخت champion list، در تابع create_index بعد از پردازش اسناد و ساختن شاخص معکوس و دیکشنری termed_freq_perdoc که تعداد تکرار هر کلمه در هر سند را مشخص می کرد، از این دیکشنری

استفاده می کنیم و champion list را می سازیم. می دانیم champion list لیست اسنادی است که کلمه را با وزن بیش تری در خود دارند. از آن جایی که idf یک کلمه ی خاص در تمام اسناد یکسان است پس دلیل تفاوت وزن یک کلمه در اسناد مختلف تعداد تکرار آن کلمه در اسناد یا tf ان کلمه در آن سند است. پس به ازای هر کلمه از تمام کلمات (یک for در دیکشنری term_termid_match)، ابتدا روی تمام اسناد(docid) موجود در دیکشنری termed_freq_perdoc یک for می زنیم و به ازای هر docid دیکشنری تمام کلماتی که در آن سند موجود هستند و تعداد تکرارشان را استخراج می کنیم. سپس چک می کنیم کلمه ی مورد نظر در دیکشنری مرتبط با آن سند است یا خیر، اگر باشد آن سند و تعداد تکرار کلمه در آن را به دیکشنری champion_list در value مرتبط با آن کلمه اضافه می کنیم. این دیکشنری بدین صورت است:

Champion_list = {termid:{docid:tf,docid2:tf2,…},termid2:{docid:tf,….}}

Docid های موجود در دیکشنری هر کلمه را براساس تعداد تکرار کلمه در آن ها به ترتیب نزولی مرتب می کنیم. سپس بیش ترین تعداد تکرار هر کلمه در اسنادی کلمه در آن ها ظاهر شده است را به عنوان بیش ترین وزن بدست می آوریم و اسنادی که تعداد تکرار کلمه در آن ها از بیش ترین تکرار-10 کم تر است را از دیکشنری هر termed یا به عبارت دیگر از champion_list آن کلمه حذف می کنیم.

سپس در تابع tf_idf برای بدست آورن اسنادی که می خواهیم شباهت آن ها را با پرسمان محاسبه کنیم به جای postings list کلمات در پرسمان از champion_list این کلمات استفاده می کنیم.این کار باعث کاهش محاسبات می شود زیرا تنها شباعت اسنادی را با پرسمان محاسبه می کنیم که احتمال این که در top k قرار گیرند به دلیل این که کلمات پرسمان را با وزن بیش تری در خود دارند بیش تر است.