# بازیابی اطلاعات – پروژه شماره 1


## زهرا رستمی– نرگس کریمیان– الهه رضاپناه


**فاز اول) استخراج اطلاعات ساختار یافته اسناد**

کد ما برای استخراج اطلاعات ساختاریافته از یک سند PDF طراحی شده است و هدف آن سازماندهی اطلاعات به صورت سلسله‌مراتبی از عناوین اصلی، زیرعناوین، آیتم‌های لیست و محتوای متنی معمولی است و خروجی کد، داده‌ها را به شکل سلسله‌مراتبی و در قالب JSON ذخیره می‌کند.

مثلا در سند ما:

- عناوین اصلی مانند "Depressive Disorders" و "Major Depressive Disorder" به عنوان کلیدهای اصلی در JSON قرار می‌گیرند.

- زیرعناوین مانند "Diagnostic Criteria" و "Development and Course" به عنوان کلیدهای داخلی در هر عنوان اصلی قرار می‌گیرند.

- آیتم‌های لیست مانند لیست نشانه‌ها و معیارهای تشخیصی در قالب لیست‌های متنی زیر کلید مربوطه ذخیره می‌شوند.

- متن عادی در قالب رشته‌های متنی در کلیدهای مناسب ذخیره می‌شود.

شرح الگوریتم:

دو تابع اصلی داریم:

۱. extract_text_elements_from_pdf(pdf_path) : این تابع PDF را می‌خواند و همه عناصر متنی و اندازه فونت آنها را استخراج می‌کند. داده‌ها در قالب یک لیست از دیکشنری‌ها ذخیره می‌شوند.

۲. parse_text_elements(elements) : عناصر متنی را تجزیه و تحلیل می‌کند تا محتوا را بر اساس سلسله‌مراتب تعیین‌شده توسط اندازه فونت‌ها سازماندهی کند.

به طور کلی می‌توان گفت :

این برنامه با استفاده از اندازه فونت و علائم شماره‌گذاری موارد زیر را شناسایی می‌کند:

- عناوین اصلی و زیرعناوین: بر اساس اندازه فونت و نوع قالب‌بندی.

- آیتم‌های لیست: با تشخیص شماره‌گذاری‌های معمول مانند "۱."، "الف." و سایر علائم مشابه.

- محتوای معمولی: متنی که به عنوان، عنوان یا آیتم لیست شناسایی نمی‌شود.

روند کد به این صورت است که:

۱. شناسایی عناوین اصلی و زیرعناوین :

- عناوین اصلی توسط اندازه فونتی که به طور قابل‌توجهی بزرگ‌تر از میانگین است شناسایی می‌شوند و زیرعناوین نیز فونتی کمی بزرگ‌تر از میانگین دارند.

- عناوین و زیرعناوین زمانی نهایی می‌شوند که خطی با قالب متفاوت شناسایی شود.

۲. شناسایی آیتم‌های لیست :

- آیتم‌های لیست از طریق شماره‌گذاری (مثلاً "۱." یا "الف.") یا حروف الفبا شناسایی می‌شوند. زمانی که یک آیتم جدید شناسایی شود، آیتم قبلی ذخیره شده و خط فعلی به لیست افزوده می‌شود.

۳. پردازش محتوای عادی:

- متنی که به عنوان عنوان یا زیرعنوان شناسایی نمی‌شود، به عنوان محتوای معمولی مرتبط با بخش در نظر گرفته می‌شود و تا رسیدن به عنوان، زیرعنوان یا آیتم لیست بعدی در بافر جمع‌آوری می‌شود و سپس در داده‌های JSON خلاصه‌سازی می‌شود.

همچنین علت استفاده ما از این الگوریتم در کدمان این است که ، روش سلسله‌مراتبی امکان تجزیه و تحلیل انعطاف‌پذیر اسناد ساختاریافته را فراهم می‌کند که در آن اندازه‌های فونت و شماره‌گذاری نشانگر عناصر ساختاری هستند. این روش برای اسنادی که ساختار عناوین و زیرعناوین دارند کاربردی است.

**فاز دوم) مدل‌سازی بازیابی اطلاعات از یک مجموعه اسناد**

سیستم بازیابی اطلاعات ما بر اساس سه مدل اصلی **Boolean Model**، (TF-IDF) **Vector Space Model** و **Levenshtein Model** است و به صورت مراحل زیر عمل میکند:

**استخراج متن از فایل‌هایPDF**

**پیش‌پردازش متون**

- امکان استفاده از چندین حالت پیش‌پردازش مانند :
  - حذف Stopwords
  - Stemming
  - Lemmatization

**مدل‌های بازیابی اطلاعات**

- **Boolean Model** : با استفاده از ایندکس‌گذاری سندها و انجام جستجوی منطقی روی کلمات کلیدی
- **Vector Space Model** : با استفاده از محاسبه شباهت کسینوسی بین پرس‌وجو و سندها
- **Levenshtein Model** : با محاسبه فاصله Levenshtein بین کلمات پرس‌وجو و سندها (روش هیوریستیک)

**ارزیابی عملکرد مدل‌ها**

- از آنجایی که ارزیابی عملکرد ما در این پروژه به صورت رتبه‌دهی اسناد هست به جای Precision و recall به محاسبه **P@k** و **R@k** برای اندازه‌گیری دقت و فراخوانی در k سند برتر بازیابی شده می‌پردازیم( k توسط کاربر مشخص می‌شود) البته در خصوص مدل Boolean از آنجایی که ترتیب و رتبه‌دهی مهم نیست همان Precision و recall را محاسبه می‌کنیم.

در سیستم ما عملکرد به این صورت هست که مدل های بازیابی روی کوئری داده شده به چندین صورت انجام می‌شود و P@k و R@k هم برای همه ی حالات حساب می‌شوند.

حالات مختلف:

هر ۳ مدل Boolean ، Vector(TF-IDF) ، Huristic(Levenshtein)

- بدون پیش پردازش
- با حذف StopWords
- با Stemming
- با Lemmatization

حالا به بررسی محاسبات در هر حالت می‌پردازیم:

در ابتدای کار از کاربر k رو دریافت می‌کنیم ( تعداد سند برتر اول)

سپس بعد از وارد کردن کوئری مورد نظر، مدل بازیابی خود را مشخص می‌کنیم

حال به بررسی خروجی می‌پردازیم:

**مدل Boolean :**

**کوئری تست شده: "panic AND attack"**

**بدون پیش‌پردازش)**

**Boolean Model Precision: 0.8667, Recall: 0.9286**

**با حذف StopWord ها)**

**Boolean Model Precision: 0.8235, Recall: 1.0000**

**با Stemming)**

**Boolean Model Precision: 0.4000, Recall: 1.0000**

**با Lemmatization)**

**Boolean Model Precision: 0.4118, Recall: 1.0000**

**مدل Vector (TF-IDF) :**

**کوئری تست شده: "Understanding Feeding and Eating Disorders in adolescents"**

**K** مشخص شده : 15

**بدون پیش‌پردازش)**

**Vector Space Model P@15: 0.9333, R@15: 0.7000**

**با حذف StopWord ها)**

**Vector Space Model P@15: 0.9333, R@15: 0.7000**

**با Stemming)**

**Vector Space Model P@15: 0.4667, R@15: 0.3500**

**با Lemmatization)**

**Vector Space Model P@15: 0.7333, R@15: 0.5500**

**مدل Levenshtein(Huristic) :**

**کوئری تست شده: "Is Bipolar I Disorder genetic?"**

**K** مشخص شده : 15

**بدون پیش‌پردازش)**

**Levenshtein Model P@15: 0.5333, R@15: 0.4000**

**با حذف StopWord ها)**

**Levenshtein Model P@15: 0.4667, R@15: 0.3500**

**با Stemming)**

**Levenshtein Model P@15: 0.4000, R@15: 0.3000**

**با Lemmatization)**

**Levenshtein Model P@15: 0.4667, R@15: 0.3500**

## تحلیل خروجی ها:

### مدل Boolean

مدل Boolean بر اساس تطابق دقیق کلمات عمل می‌کند و معمولاً برای جستجوهایی که دقیقاً می‌خواهند اسناد مرتبط را شناسایی کنند، استفاده می‌شود.

بعد از بررسی متوجه شدیم که این مدل حساسیت بالایی به پیش‌پردازش‌ها دارد، زیرا هر تغییر در ساختار کلمات می‌تواند تاثیر زیادی در تطابق نهایی بگذارد.

- **بدون پیش‌پردازش:**
  - مدل در این حالت عملکرد نسبتاً خوبی دارد. به این معنی که تعداد زیادی از نتایج بازگشتی مرتبط هستند و همچنین اکثر اسناد مرتبط شناسایی شده‌اند. به طور کلی،Precision و Recall خوب است.
- **با حذف StopWords :**
  - حذف StopWords باعث بهبود Recall شده است. با حذف این کلمات غیرضروری، مدل توانسته تمام اسناد مرتبط را شناسایی کند ،اما دقت کاهش یافته است، زیرا StopWords معمولاً در تطابق دقیق اختلال ایجاد می‌کنند و حذف آن‌ها ممکن است منجر به برگشت نتایج نامرتبط شود.

- **با Stemming** :
  - پس از اعمال Stemming، دقت به شدت کاهش یافته است. این کاهش به دلیل تغییر ساختاری در کلمات است Stemming . می‌تواند کلمات را به ریشه‌های خود تبدیل کند که باعث می‌شود مدل نتایج نامرتبط بیشتری را بازگرداند، اما تمام اسناد مرتبط همچنان شناسایی شده‌اند.
- **با Lemmatization** :
  - در این حالت، دقت کاهش یافته است، ولی Recall همچنان کامل باقی مانده است، Lemmatization به کلمات شکل پایه‌تر و معنادارتری می‌دهد که تطابق دقیق‌تری ایجاد می‌کند، اما همچنان دقت به دلیل وجود برخی نتایج نامرتبط کاهش می‌یابد.

**نتیجه‌گیری کلی** : مدل Boolean به شدت به پیش‌پردازش وابسته است.

بهترین عملکرد در این مدل زمانی به دست می‌آید که از **Lemmatization** استفاده شود.

حذف StopWords باعث بهبود Recall می‌شود، ولی دقت را کاهش می‌دهد و Stemming تاثیر منفی شدیدی بر دقت دارد.

### مدل Vector Space (TF-IDF)

مدل Vector Space به دلیل ویژگی‌های انعطاف‌پذیر خود، معمولاً عملکرد خوبی در بازیابی اطلاعات دارد. این مدل بر اساس وزن‌دهی کلمات در اسناد مختلف عمل می‌کند و از آنجا که قادر است روابط معنایی را به خوبی درک کند، عملکرد آن معمولاً نسبت به تغییرات ساختاری کلمات حساس است.

- **بدون پیش‌پردازش:**
  - در این حالت، مدل عملکرد عالی در دقت دارد، که نشان‌دهنده توانایی آن در شناسایی اسناد مرتبط است. با این حال، Recall کمی پایین است، به این معنی که ممکن است برخی از اسناد مرتبط را از دست داده باشد.
- **با حذف StopWords** :
  - حذف StopWords در این مدل تأثیری بر دقت یا Recall ندارد، زیرا این کلمات معمولاً در مدل‌های برداری وزن کمی دارند و حذف آن‌ها تاثیر چندانی در نتایج نهایی نمی‌گذارد.
- **با Stemming** :
  - پس از اعمال Stemming ، عملکرد مدل به طور قابل‌توجهی کاهش می‌یابد، Stemming می‌تواند کلمات را به فرم‌های غیرطبیعی یا مبهم تبدیل کند، که باعث کاهش کیفیت تطابق برداری می‌شود. این باعث کاهش چشمگیر دقت و یادآوری می‌شود.

- با **Lemmatization** :
  - در این حالت، Lemmatization باعث بهبود عملکرد نسبت به Stemming شده است. با این حال، هنوز کمی کاهش در دقت و یادآوری مشاهده می‌شود، اما همچنان مدل توانایی خوبی در شناسایی اسناد مرتبط دارد.

**نتیجه‌گیری کلی** : مدل Vector Space به حذف StopWords حساس نیست، اما **Stemming** می‌تواند به شدت عملکرد آن را کاهش دهد.

بهترین عملکرد زمانی مشاهده می‌شود که از **Lemmatization** استفاده شود.

**مدل Levenshtein (Heuristic)**

مدل Levenshtein برای تطابق تقریبی طراحی شده است و معمولاً در شرایطی که کلمات به‌درستی تطبیق نمی‌یابند، عملکرد خوبی دارد.

این مدل می‌تواند زمانی که کلمات به شکل صحیح تطبیق نمی‌یابند نیز نتایج مرتبط را بازیابی کند.

- **بدون پیش‌پردازش:**
  - مدل در این حالت عملکرد متوسطی دارد. این نشان‌دهنده این است که مدل قادر است برخی از اسناد مرتبط را بازگرداند، ولی در تطابق دقیق کاراکترها عملکرد ضعیف‌تری دارد.
- با حذف **StopWords** :
  - حذف StopWords تأثیری در عملکرد این مدل نداشته است، زیرا تمرکز مدل بر تطابق کاراکتر به کاراکتر است و این کلمات اضافی در تطابق کاراکتر به کاراکتر تأثیر چندانی نمی‌گذارند.
- با **Stemming** :
  - پس از اعمال Stemming ، عملکرد مدل کاهش یافته است. تغییرات کلمات به ریشه‌های کوتاه‌تر باعث می‌شود تطابق‌های تقریبی دقیق نباشند.
- با **Lemmatization** :
  - مشابه حالت حذف StopWords، Lemmatization تأثیری در عملکرد مدل نداشته است. اما Lemmatization می‌تواند کلمات را به فرم طبیعی‌تر تبدیل کند که برای این مدل مفید است.

**نتیجه‌گیری کلی** : مدل Levenshtein به پیش‌پردازش حساس است و بهترین عملکرد زمانی دارد که از **Lemmatization** استفاده شود.

حذف StopWords یا Stemming تاثیری چندانی بر بهبود عملکرد این مدل ندارد.

**جمع‌بندی کلی:**

- **مدل Boolean** بهترین عملکرد را زمانی دارد که از **Lemmatization** استفاده شود.
- **مدل Vector Space** برای داده‌های خام یا زمانی که از **Lemmatization** استفاده می‌شود بهترین عملکرد را دارد و حذف StopWords تاثیری ندارد.
- **مدل Levenshtein** نیز به پیش‌پردازش حساس است و بهترین عملکرد با **Lemmatization** مشاهده می‌شود.

همچنین در بررسی های مشابه متوجه شدیم که اگر مقدار k خیلی بزرگ شود، ممکن است مدل، تمام اسناد مرتبط را در بین k سند بازیابی کند و بنابراین P@k و R@k مشابه شوند. ما در اینجا برای بررسی مدل بهتر است از مقادیر کوچکتر برای k استفاده کردیم.

در برخی از تست ها دیدیم که recall برابر ۱ می‌شود اما precision کمتر از ۱ است، که در این صورت متوجه شدیم

Recall = 1 نشان می‌دهد که ما تمام اسناد مرتبط را پیدا کرده‌ایم.

Precision ≠ 1 به این معنی است که ما علاوه بر اسناد مرتبط، اسناد غیرمرتبط هم دریافت کرده‌ایم و دقت مدل در انتخاب اسناد مرتبط کاهش یافته است.

البته به طور کلی به نظر می‌رسد نتایج نهایی ما وابسته به کوئری های وارد شده هم هستند و دقت کلی سیستم بازیابی با میانگین‌گیری روی نتایج کوئری های مختلف با شرایط مختلف قابل اندازه‌گیری است.
به صورتی که از روش MAP استفاده می‌توان کرد (میانگین دقت در k ها، که اسناد مرتبط هستند)

**فاز سوم) ساخت تزاروس و گسترش کوئری**

این بخش را با هدف بررسی نتایج بهتر با دو روش و با استفاده از دو کتابخانه‌ی nltk و spacy پیاده سازی کردیم که هر روش را جداگانه تحلیل و بررسی میکنیم.

**روش اول: ساخت تزاروس مبتنی بر هم‌رخدادی کلمات**

در این روش، از هم‌رخدادی کلمات در یک پنجره متنی ثابت برای ایجاد تزاروس استفاده شده است. مراحل اصلی این روش مشابه مرحله فاز دوم با خواندن اطلاعات از فایل های pdf شروع شده و پس از پیش پردازش های مشخص شده مراحل زیر را دنبال می‌کند:

**ساخت تزاروس:**

- یک پنجره متنی ثابت (مثلاً دو کلمه قبل و بعد از کلمه هدف) تعریف می‌شود.
- با استفاده از این پنجره، ماتریس هم‌رخدادی ایجاد می‌شود. در این ماتریس، هر سطر و ستون نشان‌دهنده یک کلمه است و مقدار سلول‌ها بیانگر تعداد دفعات هم‌رخدادی آن کلمات در پنجره متنی است.
- این ماتریس به‌عنوان پایه‌ای برای استخراج روابط معنایی و ساخت تزاروس مورد استفاده قرار می‌گیرد.

**گسترش کوئری:**

- کلمات موجود در کوئری با استفاده از ماتریس هم‌رخدادی و تزاروس گسترش می‌یابند و کلمات مرتبط به کوئری اضافه می‌شوند.

**مدل‌سازی فضای برداری:**

- اسناد و کوئری با استفاده از **TF-IDF**برداری‌سازی می‌شوند.
- شباهت کسینوسی بین بردارها محاسبه و اسناد بر اساس این شباهت رتبه‌بندی می‌شوند.

**ویژگی‌های اصلی این روش:**

- این روش ساده است و در داده‌های عمومی با حجم بالا سریع‌تر اجرا می‌شود.
- محدودیت اصلی آن دقت پایین‌تر روابط معنایی است، زیرا تنها بر هم‌رخدادی‌های سطحی متکی است.

**روش دوم: ساخت تزاروس مبتنی بر تجزیه نحوی با استفاده ازSpaCy**

در این روش، از قابلیت‌های پیشرفته کتابخانه SpaCy برای تحلیل روابط معنایی و نحوی استفاده کردیم.

**ساخت تزاروس با استفاده از روابط نحوی**

- برای هر کلمه در متن :
  - **تحلیل وابستگی‌ها :(Dependency Parsing)** کلمات فرزند (children) و روابط نحوی آن‌ها بررسی می‌شوند.
  - **کلمات همسایه :**کلماتی که در شعاع کوچک (۲ کلمه قبل و بعد) قرار دارند، به‌عنوان کلمات مرتبط در نظر گرفته می‌شوند.
    - روابط شناسایی‌شده به‌صورت یک مجموعه از کلمات مرتبط ذخیره می‌شوند.

**گسترش کوئری**

- کوئری پردازش‌شده به کمک تزاروس توسعه می‌یابد. هر کلمه در کوئری، کلمات مرتبط از تزاروس را به خود اضافه می‌کند.

**مدل برداری فضای برداری(Vector Space Model)**

- مشابه روش اول، از TF-IDF و شباهت کسینوسی برای رتبه‌بندی اسناد استفاده می‌شود.

**ویژگی‌های اصلی این روش:**

- این روش دقت بیشتری دارد و برای متون تخصصی یا پیچیده مناسب‌تر است.
- اما پیچیدگی پیاده‌سازی و نیاز به منابع محاسباتی بیشتر، از محدودیت‌های این روش است.

**مقایسه دو روش ساخت تزاروس**

| ویژگی ها | روش اول : هم‌رخدادی | روش دوم: تجزیه نحوی با spacy |
|---|---|---|
| سادگی پیاده‌سازی | آسان و سریع | پیچیده و زمان‌بر |
| دقت روابط معنایی | محدود به روابط سطحی | تحلیل روابط دقیق نحوی |
| حساسیت به ترتیب کلمات | کم | زیاد (وابسته به ساختار گرامری) |
| پیچیدگی محاسباتی | پایین | بالا |
| انعطاف‌پذیری | مناسب برای داده های عمومی | ایده‌آل برای داده های تخصصی |
| پیچیدگی محاسباتی | پایین | بالا |

**تحلیل خروجی ها:**

**کوئری تست شده ۱: "Understanding Feeding and Eating Disorders in adolescents" (مشابه تست شده در TF-IDF در فاز قبلی)**

**روش اول: ساخت تزاروس مبتنی بر هم‌رخدادی کلمات**

**Expanded Query:**

**Understanding and metal also Eating Disorders adolescents may heavy in Feeding**

**P@k: 0.4000, R@k: 0.3000, F1-Score@k: 0.3429**

**روش دوم: ساخت تزاروس مبتنی بر تجزیه نحوی با استفاده ازSpaCy**

**P@k: 0.3333, R@k: 0.2500, F1-Score@k: 0.2857**

**و از آنجایی که در روش تزاروس ما از TF-IDF همراه با پیش پردازش Lemmatization استفاده کردیم در تحلیل این مورد هم به خروجی همین بخش برمی‌گردیم:**

**Vector Space Model P@15: 0.7333, R@15: 0.5500**

**کوئری تست شده ۲: "What is Dissociative Identity Disorder and how is it treated?"**

**روش اول: ساخت تزاروس مبتنی بر هم‌رخدادی کلمات**

**Expanded Query:**

**Identity how What is Disorder medication le Dissociative successfully ? it and may treated disorder**

**P@k: 0.2000, R@k: 0.1500, F1-Score@k: 0.1714**

**روش دوم: ساخت تزاروس مبتنی بر تجزیه نحوی با استفاده ازSpaCy**

**P@k: 0.2000, R@k: 0.1500, F1-Score@k: 0.1714**

**خروجی فاز ۲ (TF-IDF with Lemmatization)**

**Vector Space Model P@15: 0.8667, R@15: 0.6500**

**کوئری تست شده ۳: " REM sleep behavior disorder"**

**روش اول: ساخت تزاروس مبتنی بر هم‌رخدادی کلمات**

**Expanded Query:**

**REM use sleep disorder obstructive apnea may sexual repetitive anxiety rem depressive behavior personality**

**P@k: 0.0667, R@k: 0.0500, F1-Score@k: 0.0571**

**روش دوم: ساخت تزاروس مبتنی بر تجزیه نحوی با استفاده ازSpaCy**

**P@k: 0.0667, R@k: 0.0500, F1-Score@k: 0.0571**

**خروجی فاز ۲(TF-IDF with Lemmatization)**

**Vector Space Model P@15: 0.8667, R@15: 0.6500**

**مدل بدون تزاروس(Vector Space Model TF-IDF)**

- هدف روش این بود که به کلمات مهم‌تر در اسناد وزن بیشتری بدهد و حالا با استفاده از **Lemmatization** کلمات را به ریشه‌ی خود کاهش دادیم و به مدل کمک کرد تا تفاوت‌های نحوی و صرفی کلمات را نادیده بگیرد و دقت بازیابی را افزایش دهد.
- این مدل با دیدن نتایج عملکرد تقریبا خوبی داشته

**مدل با تزاروس و گسترش کوئری:**

در این مدل، از **تزاروس مبتنی بر هم‌رخدادی کلمات و همچنین تجزیه نحوی** استفاده کردیم.

هدف این بود که با گسترش کوئری اصلی، از کلمات مشابه یا هم‌معنی استفاده کنیم تا دامنه جستجو وسیع‌تر شود و اسناد بیشتری بازیابی شوند.

تست رو برای بررسی بیشتر و تحلیل دقیق‌تر روی مثال های مختلف روی ۳ کوئری انجام دادیم.

در اینجا، چون کوئری گسترش یافته، ممکن است به دلایل مختلفی باعث کاهش کیفیت نتایج شده باشد :

- **افزودن کلمات غیرمرتبط:** در فرآیند گسترش کوئری، ممکن است کلماتی به کوئری اضافه شوند که با زمینه اصلی کوئری مرتبط نباشند.
  مثلاً کلمات "in", "heavy", "may", "also", "metal" ممکن است از نظر معنایی با موضوع اصلی همخوانی نداشته باشند ( در روش اول این کلمات از wordnet پیدا شدند و در روش دوم توسط تحلیل نحوی انتخاب شدند )
  این باعث می‌شود که مدل، اسناد غیرمرتبط بیشتری را بازیابی کند، که **Precision** را کاهش می‌دهد.

حالا با استفاده از دو روش استفاده شده در تزاروس و گسترش کوئری ما به نتایج زیر رسیدیم:

- **کاهش Precision :** زمانی که کلمات اضافی به کوئری افزوده می‌شوند، اسناد غیرمرتبط بیشتری بازیابی شدند، که همین باعث کاهشPrecision ما شد و این یعنی مدل اسناد مرتبط کمتری بازیابی کرده است.
- **کاهش Recall :** حالا که گسترش کوئری با کلمات غیرمرتبط انجام شده، اسناد مفیدی که به طور طبیعی با کوئری اصلی ارتباط دارند، به دلیل ناتوانی در تشخیص کلمات هم‌معنی مرتبط با زمینه اصلی، از دست رفته‌اند. این مسئله باعث کاهش **Recall** شده.

از آنجا که تزاروس و گسترش کوئری به طور کلی به معنای استفاده از هم‌معنای کلمات و واژه‌های مرتبط است در واقع، مشکلی که مشاهده می‌شود به نحوه انتخاب و استفاده از کلمات گسترش‌یافته برمی‌گردد. در پیاده‌سازی ما مشاهده می‌شود، کلمات افزوده شده از لحاظ معنایی ارتباط زیادی با مفهوم اصلی کوئری ندارند وهمین باعث کاهش کیفیت سرچ شده.

همچنین با بررسی متوجه شدیم که استفاده از spacy یا nltk در پیاده‌سازی های ما تفاوت چندانی نداشت و نتایج تقریبا برابر هم شدند.