یک نمای کلی از آنچه در این کد اتفاق می افتد:

- تابع read() در محتویات چندین سند می خواند و آنها را در لیستی به نام docsContents ذخیره می کند.
- تابع normalize(data, i) با انجام مراحل زیر، داده های هر سند را عادی و پیش پردازش می کند:
- کلاس Normalizer() از کتابخانه NLTK برای عادی سازی متن استفاده می شود.

در پردازش زبان طبیعی، عادی سازی متن‌ها به معنای تبدیل کلمات و عبارات به شکل استاندارد است، تا بتوان آن‌ها را با هم مقایسه کرد. عادی سازی متن‌ها قبل از ساخت شاخص باید انجام شود، زیرا شاخص‌ها معمولا بر پایه فراوانی کلمات در متون ساخته می‌شوند و اگر کلماتی در متون با طول متفاوت یا فراوانی متفاوت به کار برده شوند، ممکن است محاسبه شاخص به درستی انجام نشود. به عنوان مثال، قبل از ساخت شاخص، لازم است تمام حروف کلمات به یک شکل استاندارد تبدیل شده و حروف اضافی مانند حروف ربط و حروف تعریف حذف شوند. همچنین، پسوندهای کلمات برداشته شده و کلمات به شکل ریشه آن‌ها بازنویسی می‌شوند. با انجام این مراحل، متون با هم مقایسه شوند و شاخص‌ها با دقت بالا ساخته شوند.

- تابع word_tokenize() از کتابخانه NLTK برای توکن کردن متن نرمال شده استفاده می شود.

عملیات توکن سازی، به معنای تبدیل یک ورودی از متن به توکن‌های جداگانه با استفاده از بریدن و جدا کردن آن به موارد تشکیل دهنده‌اش می‌باشد.
اگر عمل توکن سازی را در پیش پردازش انجام ندهیم، ممکن است برای پردازش‌های بعدی نیاز به تبدیل دسته‌ای متون به توکن‌های جداگانه باشد. در این صورت، باید به صورت جداگانه ابتدا متن‌ها را توکن سازی کرده و سپس از آن‌ها برای پردازش مورد نظر استفاده کنیم.
به علاوه، با انجام توکن سازی در پیش‌پردازش، می‌توانیم توکن‌های غیرمعنادار مانند حروف ربط، حروف تعریف و حروف اضافه را از متون حذف کرده و تاثیر آن‌ها در شمارش وزن‌ها را کاهش دهیم.

- کلاس Stemmer() از کتابخانه NLTK برای اجرای stemming روی متن توکن شده استفاده می شود.

عملیات stemming یکی از روش‌های پیش پردازش متن است که در آن کلمات ریشه‌یابی می‌شوند و به شکل استانداردی در می‌آیند. این کار، بهبود قابل توجهی در کیفیت و دقت شاخص ایجاد می‌کند.
شاخص‌ها براساس تعداد کلمات و فراوانی آن‌ها در متن ساخته می‌شوند. اگر کلمات به شکل استانداردی نیایند، ممکن است به دلیل تنوع بیشتر، کلماتی مشابه در کنار هم شمرده شوند و این باعث بیشتر شدن میزان خطا و کاهش دقت شاخص می‌شود.

- هر کلمه توقفی که در متن وجود دارد با استفاده از لیستی به نام stopWordsList حذف می شود.
- تابع index() با متن نشانه گذاری شده و یک شاخص i به عنوان پارامترهای ورودی فراخوانی می شود.
- تابع index(tokenizedData, i) دیکشنری positionalIndex را به روز می کند که هدف نهایی این برنامه است. موارد زیر را انجام می دهد:

- برای هر کلمه در داده های نشانه گذاری شده، تعداد کلمات در فرهنگ لغت positionalIndex را 1 افزایش می دهد.
- اگر کلمه قبلاً در فرهنگ لغت positionalIndex وجود داشته باشد، سند i و موقعیت کلمه را به لیست مربوطه از docID و position اضافه می کند.
- اگر کلمه در فرهنگ لغت positionalIndex وجود نداشته باشد، یک ورودی جدید برای کلمه ایجاد می‌شود که تعداد آن 1 است و فهرست‌های docID و موقعیت با سند i و موقعیت کلمه مقداردهی اولیه می‌شوند.

در نهایت، فرهنگ لغت positionalIndex چاپ می شود که شامل فهرست همه اسناد است.