**※以下の文章はメンテナンスされません。テスト用データとしてお使いください。**

**1. 前提となるファイル配置と設定ファイルの準備**

1. **PDFファイルやWordファイルを配置  
   Ragponに取り込みたいPDFファイルやWordファイルを、あらかじめ指定のフォルダ（例：examplesフォルダ）に配置します。**
2. **設定ファイル（YAML形式）を用意  
   Ragponでは、各種挙動を管理するための設定ファイルが必要です。PDF用・データフレーム用など、用途に応じたYAMLファイルを用意しておきます（例：sample\_config\_for\_pdf.ymlとsample\_config\_for\_dataframe.yml）。**
3. **ロギングの設定（任意）  
   開発やデバッグのために、ログレベル（INFO・DEBUG・WARNINGなど）やログのフォーマットを設定することができます。**

**2. 設定の読み込みとサービスの初期化**

1. **Configクラスを使って設定ファイルを読み込む  
   PDF用の設定ファイルを指定し、Configクラスのインスタンスを生成します。**
2. **DocumentProcessingServiceクラスをインスタンス化  
   先ほど生成したConfigインスタンスを引数に渡して、DocumentProcessingServiceクラスを初期化します。  
   これにより、PDFやWordファイルの分割処理やベクトル化、検索などをまとめて行うことが可能になります。**

**3. PDFファイルやWordファイルの処理**

1. **process\_fileメソッドでファイルを処理**
   * **PDFファイルを指定して読み込む**
   * **Wordファイルを指定して読み込む  
     これにより、それぞれの文章がセクションごとに分割・ベクトル化され、検索可能な状態になります。**
2. **データの再投入  
   一度削除したデータを再度読み込みたい場合も、同じprocess\_fileメソッドを使って再格納することができます。**

**4. 検索機能の利用**

1. **searchメソッドでキーワードを検索  
   検索したいキーワード（例：「投資信託のリスク」）を引数として渡すと、関連度の高いテキスト片が返ってきます。**
   * **top\_kパラメータを指定することで、上位何件の検索結果を取得するかを設定可能です。**
   * **結果はリスト形式で受け取ることができ、必要に応じて変数に格納し後続の処理に利用できます。**
2. **delete\_by\_idsメソッドで特定の文書IDを削除**
   * **PDFやWordの内容がセクションごとに分割されており、各セクションに割り振られたIDが存在します。**
   * **不要なセクションをID指定で削除すると、次回の検索結果からそのセクションが除外されます。**
3. **delete\_by\_metadataメソッドでメタデータに応じた削除**
   * **serial\_numberなどのメタデータを指定し、該当するセクションをまとめて削除できます。**

**5. 検索結果の拡張と再ランク付け**

1. **enhance\_search\_resultsメソッド**
   * **検索で得られた結果に対し、前後の文章を付与して文脈を広げることができます。**
   * **この処理によって、検索結果オブジェクトにenhanced\_textという項目が追記され、より詳しい内容を把握できます。**
2. **rerank\_resultsメソッド**
   * **検索結果を再ランク付けし、より関連度の高いものを上位に表示させます。**
   * **独自の再ランク付けモデル（reranker）を用いてスコアを付与し、高いスコア順に結果が並びます。**
   * **search\_result\_text\_keyの引数に、再ランク付けの対象とするテキスト項目（通常はenhanced\_text）を指定します。**
   * **スコアは結果のrerank項目に追記されます。**

**6. データフレームを使った検索**

1. **DataFrameの準備  
   例えば、営業記録などのデータをPandasのDataFrame形式で用意しておきます。**
   * **各行に含まれる文章やメモ（例：「訪問記録」列）をRagponで検索可能にします。**
2. **process\_dataframeメソッドでDataFrameを取り込む**
   * **chunk\_col\_nameに分割対象となる文章の列名を指定します。**
   * **id\_col\_nameには、各行をユニークに識別できる列（例：「index」）を指定します。**
   * **これにより、PDFやWordファイルと同様に文書がベクトル化され、検索対象に加わります。**
3. **searchとrerank\_resultsの利用**
   * **DataFrameから取り込んだデータにも、同じsearchメソッドで検索をかけることができます。**
   * **ただし、分割の考え方がPDF・Wordの場合と異なるため、拡張用メソッド（enhance\_search\_results）は使わないケースが多いです（分割されていないテキストなので前後文脈の追記は不要な場合もあります）。**
   * **必要に応じてrerank\_resultsを使って再ランク付けを行い、検索結果を精度向上させることが可能です。**

**7. まとめ**

* **Ragponは、設定ファイル（YAML）を介して様々なフォーマットの文書を分割・ベクトル化し、高精度な全文検索を行うライブラリです。**
* **PDF、Wordのようなドキュメントファイルから、DataFrame形式のテキストまで柔軟に扱うことができ、削除や再投入、再ランク付けなどの操作を簡単に実行できます。**
* **検索結果をより良い形で活用するために、前後文脈の付与や独自モデルを使った再ランク付けをサポートしており、用途に応じて拡張することが可能です。**

**以上がRagponの基本的な使い方の概要です。日々の業務や情報管理などにご活用ください。**