Engineer

全文検索

文書の単語で探し、関連度の順に並べる検索

どのようなものか

全文検索は、検索条件の単語を含む文書を語形の違いも考慮して探し、よく一致する文書から順に並べられる仕組みです。英語では、ハイフンなしでfull text searchとも書きます。文書とは、アプリが検索の一単位とするものです。メール、記事、サポートチケット、テーブルの一行の題名と本文を合わせたものなどです。

パターンマッチングと異なるのは、比較を始める前に行う処理です。最初に、単語、数値、メールアドレスなどの種類のトークンへ分けます。次に、同じ単語の変化形が同じ形になるように選んだ表現、レキシームへ正規化します。大文字を小文字にし、ステマーが英語の複数形などの語尾を取り除き、よく現れて手がかりにならないストップワードを除きます。各レキシームの位置とともに保存するため、後の順位付けで、検索語同士が近い文書を優先できます。検索語も同じ処理を通り、「payments」の検索と「payment」の文書が、同じ形で出会います。

PostgreSQLは、これをデータベースに組み込みます。`tsvector`が正規化した文書、`tsquery`が検索条件を持ち、`@@`が一致するかを答えます。テキスト検索設定は、どのパーサーが分割し、どの辞書がトークンをレキシームへ変えるかを決めます。標準配布には、`english`、`danish`、`norwegian`、`swedish`などがあり、psqlの`\dF`で、インストール済みのものを一覧にできます。SQLiteも、FTS5拡張で、仮想テーブルに索引を置き、`MATCH`で検索する、同じ考えを提供します。

概念を判断に生かす

「全文検索とは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 年々重くなってきたシステムのクエリと不足するインデックスを調べ、効果のある箇所を改善します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 30日分の電力データを180 msで処理 — 180 ms 1か月分のメーター値を集計

30日分の電力データを180 msで処理 お問い合わせ →

なぜ重要なのか

PostgreSQLの文書は、古い演算子の不足を説明します。`LIKE`、`ILIKE`、正規表現に、言語の支援はありません。全変化形を書かなければ、「satisfy」で「satisfies」を見つけられません。順位付けもないため、千件の一致が返っても、有用な順序には並びません。この検索に普通の索引も使えず、毎回、全文書を読みます。`%cat%`は、「category」「concatenate」にも一致し、見つかるべき文書の増加ではなく、雑音になります。

人は、検索欄へ単語を入れ、その話題の文書を期待します。契約書の保存庫、ヘルプデスク、商品カタログでは、信頼する検索欄と、三回外れたら使わなくなる検索欄の違いになります。順位は、一致と同じくらい重要です。「late payment」なら、上に欲しいのは督促状で、「payment」を一度だけ含む、百枚の請求書ではありません。

運用中のデータベース内にあることにも、実用的な意味があります。索引がデータと同じトランザクションにあるため、確定した瞬間に検索でき、削除も同時に結果から消えます。日付、顧客、権限の条件も、テキスト条件の隣の、普通の`WHERE`句です。別の検索サーバーなら、もっとできますが、二つ目へ投入し、同期する必要があります。ずれは、あるはずなのにない結果や、消えるべきなのに残る結果として現れます。

最初に知るべき限界もあります。照合するのは単語で、意味ではありません。同義語辞書で対応させなければ、「car」は「automobile」を見つけません。意味の数値表現を比べるベクトル検索が、その場合を扱い、よく併用します。

実際の活用例

題名、本文、出所を持つ文書テーブルを考えます。計算結果を保存する生成列が、挿入と更新のたびに検索用の形を更新し、GIN索引が速く探せるようにします。

```
CREATE TABLE documents (
    id         bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    title      text,
```

```
    body       text,
    source     text NOT NULL,
    search_vec tsvector GENERATED ALWAYS AS (
                   setweight(to_tsvector('english', coalesce(title, '')), 'A')
                || setweight(to_tsvector('english', coalesce(body, '')), 'D')
               ) STORED
);

CREATE INDEX documents_search_idx ON documents USING GIN (search_vec);
```

三つが重要です。設定は、セッションの `default_text_search_config` に任せず、 `'english'` と指定します。二引数の `to_tsvector` が、PostgreSQLで保存と索引化に使える形です。 `coalesce` は、ない題名や本文を空の文字列へ変えます。一つのNULLで、文書全体がNULLになるのを防ぎます。 `setweight` は、題名ならA、本文ならDと、各レキシームの出所を示します。別の指定がなければ、順位付け関数は、D、C、B、Aへ{0.1, 0.2, 0.4, 1.0}を使い、題名の検索語は、本文の十倍になります。

文書が推奨するのは、GINです。レキシームごとの項目と、それを含む行の圧縮した一覧を持つ、転置索引です。GiSTも使えますが、索引が元の情報をすべて保持するわけではないため、誤一致を除くには、テーブルの行を確かめます。

利用者が入れた単語を、そのまま受け取る検索です。

```
SELECT id, title, source,
       ts_rank_cd(search_vec, query) AS rank
FROM   documents,
       websearch_to_tsquery('english', 'unpaid invoice -draft') AS query
WHERE  search_vec @@ query
ORDER  BY rank DESC
LIMIT  10;
```

`websearch_to_tsquery` は、ウェブ検索欄のように読みます。普通の単語はすべて、英語のステマーが同一視する、どれかの形で必要です。二重引用符の内容は句として必要で、「or」は選択肢、先頭のマイナスは除外です。そのため、この検索は「draft」を含む全文書を除きます。構文エラーを起こさないため、生の入力に安全な選択です。 `ts_rank_cd` は、Clarke、Cormack、Tudhopeの1999年の論文による、検索語の近さも評価する、被覆密度で並べます。関連する `ts_rank` は、出現頻度を数えます。両方が正規化フラグを受け、例えば1は、評価値を、文書長の対数に一を足した値で割ります。長いだけで勝つことがなくなります。

同じ必要性が、会社のAIエージェント向けの企業データの仕事にもあります。古いPDF、スキャンデータ、メールアーカイブを、出所とともに検索できる本文へ変え、保存するだけでなく見つけられるようにする話です。上の `source` 列も、その役目です。元を開ければ、結果の価値が増します。

SQLiteは、もっと少ない手順で同じ結果へ届きます。 `CREATE VIRTUAL TABLE notes USING fts5(title, body)` で宣言し、 `WHERE notes MATCH 'invoice NOT draft'` で検索します。 `ORDER BY bm25(notes)` で、よく合うものが先です。FTS5は、小さい数ほどよい結果なので、既定の昇順が正解です。既定のトークナイザーは `unicode61` です。 `porter` はステミングを加えます。文書によれば、このトークナイザーは英語だけを対象に設計されています。

問題が起きるところ

検索が、索引を使いません。 `to_tsvector('english', body)` の式索引を使えるのは、同じ式を繰り返す検索だけです。既定に頼る `to_tsvector(body)` では、全テーブルを読みます。上のように生成列へ保存し、その列を検索すれば、ずれません。

一つのNULLが、文書を隠します。 `to_tsvector(NULL)` はNULLを返し、NULLの題名へ本文をつないでもNULLです。題名のない行が、気付かれないまま全検索から消えます。各フィールドを `coalesce` で包みます。

生の入力を `to_tsquery` へ渡します。 `to_tsquery` は独自の演算子構文を期待し、ほかの変換関数より厳しいため、対応しない括弧や、余分なアンパサンドでエラーページになります。 `websearch_to_tsquery` か、演算子が一切不要なら `plainto_tsquery` を通します。

言語が違います。 デンマーク語を英語の設定で索引化すると、英語の規則で語幹処理し、英語のストップワードを除くため、変化形の半分は一致しなくなります。本文の言語に合う設定を選びます。多言語なら、設定名を別の列に持ち、検索も同じ列を指定すれば、 `to_tsvector(config_name, body)` を索引化できます。

すべての一致結果の順位を計算します。 一致した各行の `tsvector` を読み、文書は、I/Oが律速になって遅くなる場合があると警告します。一語で半分のテーブルに合えば、 `LIMIT` が十件へ絞る前に、半分を評価します。日付範囲や顧客など、普通の条件で候補を絞り、行ごとに計算し直さないよう、ベクトルも保存します。

評価値を、割合として見せます。 関数は、集合全体の情報を使わないため、「関連度87%」という妥当な割合へ換算することはできません。正規化フラグ32は、各評価値を0から1の範囲へ収めますが、文書は、見た目だけの変更とします。結果を並べるためだけに使います。

強調表示に、マークアップが付いてきます。 `ts_headline` は、検索語を強調した抜粋を返します。結果ページに欲しいものですが、PostgreSQLは「ウェブページへ直接入れて安全だとは保証しない」と警告します。HTMLの文書が、スクリプトを運び込む場合があります。入力からマークアップを除くか、出力をHTMLサニタイザーへ通します。ベクトルではなく原文で動くため、全一致ではなく、`LIMIT` 後の、そのページの行だけで呼びます。

文書が、一つのベクトルには大きすぎます。 `tsvector` は1 MB未満で、一つのレキシームに保持できる位置は、最大256個です。本一冊分や一年分の議事録を一行に入れると、両方の制限に達します。節やページに分ければ、検索結果から該当箇所をより細かく示せます。

断片や部品番号を探します。 レキシーム全体を比べるため、「inv」は「invoices」を探せず、`AB-1042-X` などは分割される場合があります。完全一致で探す列へ普通の索引を置くか、SQLiteの `trigram` を使います。連続する三文字ずつをトークンにし、部分文字列を探せます。

SQLiteの索引が、テーブルとずれます。 `content=` のFTS5は、別のテーブルから本文を読み、両者の整合性を保つのは利用者の仕事だと、文書は述べます。ずれると、相反するように見える行が返ります。仮想テーブルと同じマイグレーションで、文書の示す、挿入、更新、削除のトリガーを作ります。

参考資料

- 全文検索の紹介 — 文書、トークン、レキシーム、設定の定義と、`LIKE` や正規表現の不足です。
- テーブルと索引 — テーブルの検索、式索引と保存される生成列の比較、設定名が必要な理由です。
- テキスト検索の制御 — 重み、`websearch_to_tsquery`、`ts_rank`、`ts_rank_cd`、正規化フラグ、`ts_headline` の警告です。
- テキスト検索に推奨する索引の種類 — GiSTが行を再確認する理由を含む、GINとの比較です。
- 制限 — レキシーム、ベクトル、位置、検索条件の大きさの限界です。
- SQLite FTS5拡張 — 仮想テーブル、`MATCH`、内蔵トークナイザー、`bm25()`、外部コンテンツテーブルです。

用語集

実際の制約に合わせて選ぶ

代表的なデータセット、そこから答えを得たい問い、現在のボトルネックをお知らせください。品質、処理量、復旧の測り方を合意できます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「全文検索とは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/full-text-search/