

PGroongaを使って 全文検索結果を より良くする方法

堀本 泰弘 株式会社クリアコード

PostgreSQL Conference Japan 2021
2021-11-12





本日の資料

- 本日のスライド
 - <https://slide.rabbit-shocker.org/authors/komainu8/improve-search-result-with-pgroonga.rab>







自己紹介



堀本 泰弘

全文検索エンジン

grnga pgrnga
mrnga rrnga

の開発・サポート



目次

1. 検索の評価指標
2. PGroongaで検索結果の改善
3. 参考資料



検索の評価指標

よく検索結果が
いまいちだ...
という話を
聞きます



検索の評価指標

何を基準に「検索結果がいまいち」と判断しているのでしょうか？





検索の評価指標

- 🙄 検索漏れ
- 🙄 ノイズが多い
- 🙄 有用な情報を探し出せない



検索の評価指標

1. 効率性

低コストで検索できるかどうか

2. 有効性

検索結果の全体 or 一部が
欲しい情報だったかどうか



検索の評価指標

今日は有効性についてのお話です



有効性の指標

1. 適合率
2. 再現率
3. ランキング



適合率と再現率

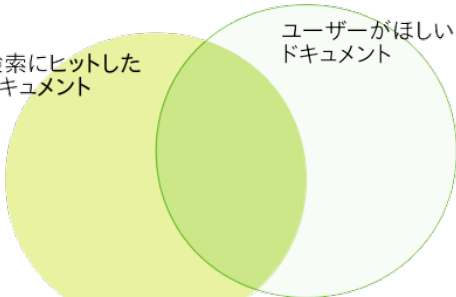
$$\text{再現率} = \frac{\text{検索にヒットしたドキュメント}}{\text{ユーザーがほしいドキュメント}}$$

$$\text{適合率} = \frac{\text{ユーザーがほしいドキュメント}}{\text{検索にヒットしたドキュメント}}$$

検索対象のドキュメント全体

検索にヒットしたドキュメント

ユーザーがほしいドキュメント





適合率と再現率

再現率: 低
適合率: 高

検索対象のドキュメント全体

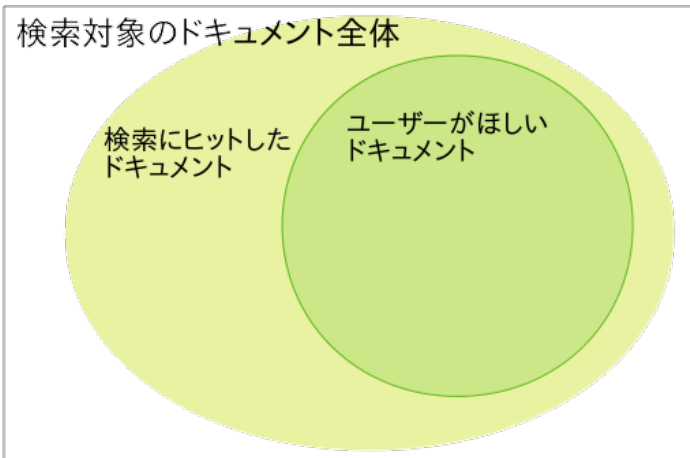
検索にヒットした
ドキュメント

ユーザーがほしい
ドキュメント



適合率と再現率

再現率: 高
適合率: 低





ランキング

欲しい情報が
ランキング上位に
あるか



ランキング

ユーザーは
上位数件
しか見ない



PGroongaで検索結果の改善

- PGroongaで適合率/再現率改善
 - ノーマライザーを使う
 - トークナイザーを使う
 - ステミングを使う
 - fuzzy検索を使う
 - 同義語展開を使う



PGroongaで検索結果の改善

- PGroongaでランキング改善
 - スコアラーを使う



PGroongaのノーマライザー (デフォルト)

```
CREATE DATABASE pgroonga_test;  
CREATE EXTENSION pgroonga;  
CREATE TABLE normalizer_test (  
  id integer,  
  content text  
);  
CREATE INDEX pgroonga_content_index ON normalizer_test USING pgroonga (content);  
  
INSERT INTO normalizer_test VALUES (1, 'キログラム');  
INSERT INTO normalizer_test VALUES (2, 'きろぐらむ');  
INSERT INTO normalizer_test VALUES (3, '𐐪𐐰');  
INSERT INTO normalizer_test VALUES (4, '𐐪𐐰𐐤𐐤');  
INSERT INTO normalizer_test VALUES (5, 'kiroguramu');  
INSERT INTO normalizer_test VALUES (6, 'k i r o g u r a m u');  
  
SELECT * FROM normalizer_test WHERE content &@ 'キログラム';
```



PGroongaのノーマライザー (デフォルト)

id	content
1	キログラム
3	キロ グラム
4	キログラム



PGroongaのノーマライザー (デフォルト)

- 半角/全角を同一視
- 𐤀とキログラムを同一視



PGroongaのノーマライザー (デフォルト)

PGroongaのデフォルトは
NFKCを使った正規化
※対象のテキストのエン
コードがUTF-8の時



ノーマライザーの変更

再現率を上げたい



PGroongaのノーマライザー (NormalizerNFKC130)

```
DROP INDEX pgroonga_content_index;  
CREATE INDEX pgroonga_content_index  
    ON normalizer_test  
    USING pgroonga (content)  
    WITH (normalizers='NormalizerNFKC130("unify_to_romaji", true)');  
SELECT * FROM normalizer_test WHERE content &@ 'キログラム';
```



PGroongaのノーマライザー (NormalizerNFKC130)

id	content
1	キログラム
2	きろぐらむ
3	𑜀𑜂𑜆𑜐𑜫
4	𑜀𑜂𑜆𑜐𑜫 𑜀𑜂𑜆𑜐𑜫



PGroongaのノーマライザー (NormalizerNFKC130)

id		content
5		kiroguramu
6		k i r o g u r a m u



PGroongaのノーマライザー (NormalizerNFKC130)

- Unify_to_romaji
 - ローマ字に正規化
ローマ字で読んだときに同じ語は同一視する
 - (e.g. 「kiroguramu」と「きろぐらむ」を同一視。ローマ字読みが同じだから)



オプションの指定方法

- 'NormalizerNFKC130
("オプション名", true)');



指定可能オプション一覧

- NormalizerNFKC130のオプション一覧
 - https://groonga.org/ja/docs/reference/normalizers/normalizer_nfkc130.html#syntax



PGroongaのトークナイザー (デフォルト)

```
CREATE TABLE tokenizer_test (  
  title text  
);  
CREATE INDEX pgroonga_content_index ON tokenizer_test USING pgroonga (title);  
  
INSERT INTO tokenizer_test VALUES ('京都府 1日目 金閣寺');  
INSERT INTO tokenizer_test VALUES ('京都府 2日目 嵐山');  
INSERT INTO tokenizer_test VALUES ('京都府 3日目 天橋立');  
INSERT INTO tokenizer_test VALUES ('東京都 1日目 スカイツリー');  
INSERT INTO tokenizer_test VALUES ('東京都 2日目 浅草寺');  
INSERT INTO tokenizer_test VALUES ('北海道 1日目 函館');  
INSERT INTO tokenizer_test VALUES ('北海道 2日目 トマム');  
INSERT INTO tokenizer_test VALUES ('北海道 3日目 富良野');  
INSERT INTO tokenizer_test VALUES ('北海道 4日目 美瑛');  
INSERT INTO tokenizer_test VALUES ('北海道 5日目 旭川');  
  
SELECT * FROM tokenizer_test WHERE title &@ '京都';
```



PGroongaのトークナイザー (デフォルト)

title		
京都府	1日目	金閣寺
京都府	2日目	嵐山
京都府	3日目	天橋立
東京都	1日目	スカイツリー
東京都	2日目	浅草寺



トークナイザーの変更

適合率を上げたい



PGroongaのトークナイザー (TokenMecab)

```
DROP INDEX pgroonga_content_index;  
CREATE INDEX pgroonga_content_index  
    ON tokenizer_test  
    USING pgroonga (title)  
    WITH (tokenizer='TokenMecab');  
  
SELECT * FROM tokenizer_test WHERE title &@ '京都';
```




PGroongaのトークナイザー (TokenMecab)

title		
京都府	1日目	金閣寺
京都府	2日目	嵐山
京都府	3日目	天橋立



トークナイザーの指定方法

- `tokenizer='トークナイザー名'`



指定可能トークナイザー一覧

- 使用可能なトークナイザー
 - <https://groonga.org/ja/docs/reference/tokenizers.html>



ステミング(語幹処理)

語形変化
意味は同じだが
語の形が変わる



ステミング(語幹処理)

例えば

- develop(原形)
- developped(過去形)
- developing(進行形)

意味は同じだが語形は異なる



ステミング(語幹処理)

語幹：単語の変化
しない部分



ステミング(語幹処理)

develop
developped
developing



ステミング(語幹処理)

語幹で検索
->語形変化後の語
も検索できる



PGroongaのステミング (未使用)

```
CREATE TABLE stemming_test (  
  title text  
);  
CREATE INDEX pgroonga_content_index ON stemming_test USING pgroonga (title);  
  
INSERT INTO tokenizer_test VALUES ('I develop Groonga');  
INSERT INTO tokenizer_test VALUES ('I am developing Groonga');  
INSERT INTO tokenizer_test VALUES ('I developed Groonga');  
  
SELECT * FROM tokenizer_test WHERE title &@ 'develop';
```



PGroongaのステミング (未使用)

title
I develop Groonga



PGroongaのステミング

```
CREATE INDEX pgroonga_content_index
  ON stemming_test
  USING pgroonga (title)
  WITH (plugins='token_filters/stem',
        token_filters='TokenFilterStem');
```



PGroongaのステミング

title
I develop Groonga
I am developing Groonga
I developed Groonga



同義語

同義語：同じ意味
を持つ別の語



同義語

例えば
「ミルク」と
「牛乳」



同義語

意味が同じものは
ヒットしてほしい



同義語展開

ミルク ->
ミルク OR 牛乳



PGroongaの同義語展開

```
CREATE TABLE synonyms (  
  term text PRIMARY KEY,  
  synonyms text[]  
);  
  
CREATE INDEX synonyms_search ON synonyms USING pgroonga (term pgroonga.text_term_search_ops_v2);  
  
INSERT INTO synonyms (term, synonyms) VALUES ('ミルク', ARRAY['ミルク', '牛乳']);  
INSERT INTO synonyms (term, synonyms) VALUES ('牛乳', ARRAY['牛乳', 'ミルク']);  
  
CREATE TABLE memos (  
  id integer,  
  content text  
);  
  
INSERT INTO memos VALUES (1, '牛乳石鹸');  
INSERT INTO memos VALUES (2, 'ミルクジャム');  
INSERT INTO memos VALUES (3, 'ストロベリー');  
  
CREATE INDEX pgroonga_content_index ON memos USING pgroonga (content);  
  
SELECT * FROM memos  
WHERE  
  content &@~  
    pgroonga_query_expand('synonyms', 'term', 'synonyms', '牛乳');
```



同義語展開

id	content
1	牛乳石鹸
2	ミルクジャム



typo対策



曖昧検索

- 似たような語ならヒットする
(完全一致じゃなくてもヒットする)



曖昧検索

「テノクロジー」
で
「テクノロジー」
がヒット



PGroongaのfuzzy検索

```
CREATE TABLE tags (  
  name text  
);  
  
CREATE INDEX tags_search ON tags USING pgroonga(name) WITH (tokenizer='');  
INSERT INTO tags VALUES ('テクノロジー');  
INSERT INTO tags VALUES ('テクニカル');  
  
SELECT name FROM tags  
WHERE  
  name &  
    ('fuzzy_search(name, ' || pgroonga_escape('テクノロジー') || ',  
      {"with_transposition": true,  
        "max_distance": 1})');
```



曖昧検索

name
テクノロジー



何を基準に
ランキングを
決めるのか



PGroongaのスコアリング

- TF(PGroongaのデフォルト)
- TF-IDF



PGroongaのスコアリング TF(デフォルト)

単語の出現数
が大事



PGroongaのスコアリング TF(デフォルト)

- 検索キーワードが文書内に多く含まれる文書のスコアが高くなる



PGroongaのスコアリング TF-IDF

単語のレア度
が大事



PGroongaのスコアリング TF-IDF

- 文書に出てくる頻度が高い
(レア度低い)
- 文書に出てくる頻度が低い
(レア度高い)



PGroongaのスコアリング TF-IDF

```
CREATE TABLE memos (  
  title text,  
  content text  
);  
  
CREATE INDEX pgroonga_memos_index  
  ON memos  
  USING pgroonga (content);  
INSERT INTO memos VALUES ('PostgreSQL', 'PostgreSQLはリレーショナル・データベース管理システムです。');  
INSERT INTO memos VALUES ('Groonga', 'Groongaは日本語対応の高速な全文検索エンジンです。');  
INSERT INTO memos VALUES ('PGroonga', 'PGroongaはインデックスとしてGroongaを使うためのPostgreSQLの拡張機能です。');  
INSERT INTO memos VALUES ('コマンドライン', 'groongaコマンドがあります。');  
  
SELECT *, pgroonga_score(tableoid, ctid) AS score  
FROM memos  
WHERE content &@~  
  ('PostgreSQL OR 検索',  
   ARRAY[1],  
   ARRAY['scorer_tf_idf($index)'],  
   'pgroonga_memos_index')::pgroonga_full_text_search_condition_with_scorers  
ORDER BY score DESC;
```



PGroongaのスコアリング TF-IDF

title	score
Groonga	1.386294364929 1992
PostgreSQL	1
PGroonga	1



参考資料

- PGroonga自体の解説
 - <https://www.slideshare.net/kou/postgresql-conference-japan-2017>