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

自己紹介

堀本　泰弘

全文検索エンジン

groonga　pgroonga

mroonga　rroonga

の開発・サポート

今日のテーマ

検索結果の改善

目次

1. 検索結果の評価指標
2. PGrooongaで検索結果の改善

検索結果の評価指標

よく検索結果が
いまいちだ...
という話を
聞きます

検索結果の評価指標

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？

いまいちな検索結果

- ☹検索漏れ
- ☹ノイズが多い
- ☹有用な情報を探し出せない

検索結果の評価指標

1. 適合率
2. 再現率
3. ランキング

適合率

$$\text{適合率} = \frac{\text{検索にヒットしたドキュメントのうち情報要求にマッチしたドキュメント数}}{\text{検索にヒットしたドキュメント数(HIT数)}}$$

再現率

$$\text{再現率} = \frac{\text{検索ヒットしたドキュメントのうち情報要求にマッチしたドキュメント数}}{\text{検索対象のドキュメント全体のうちユーザの情報要求にマッチしたドキュメント数}}$$

適合率と再現率

具体例

適合率と再現率

Wikipediaで
"キログラムの
定義"を調べる

適合率と再現率

```
SELECT title FROM wikipedia where text &@~ 'キログラム 定義';
                          title
-----------------------------------------------------
 水
 力
 国際単位系
   .
   .
   .
キログラム
   .
   .
   .
(229 rows)
```

適合率と再現率

適合率

2/229=0.87%

適合率と再現率

- ヒットしなかったけど欲しかった記事
  - SI基本単位の再定義
  - 国際キログラム原器

適合率と再現率

再現率

2/4=50.0%

適合率と再現率の重要度

Web検索

適合率 > 再現率

適合率と再現率の重要度

特許検索

適合率 < 再現率

ランキング

検索結果の順序

ランキング

ユーザーは

上位数件

しか見ない

ランキングの指標(参考)

- Precision@n
- Map(Mean Average Precision)
- nDCG(Normalizerd Discounted Cumulative Gain)

PGroongaで検索結果の改善

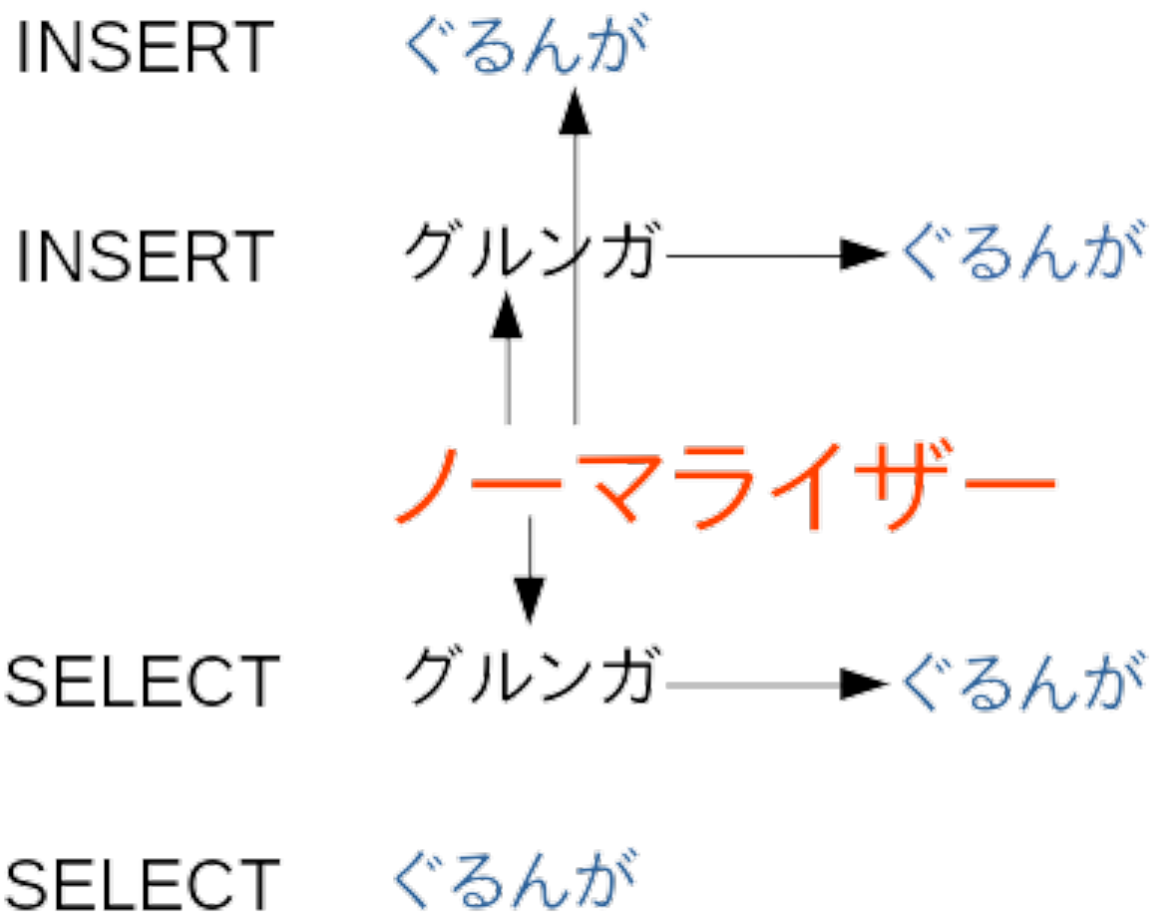
- PGroongaで適合率/再現率改善
  - ノーマライザーを使う
  - トークナイザーを使う
  - ステミングを使う
  - fuzzy検索を使う
  - 同義語展開を使う

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

ノーマライズ(正規化)とは？

ノーマライズ


INSERT
ぐるんが
INSERT
グルンガ
ぐるんが
ノーマライザー
SELECT
グルンガ
ぐるんが
SELECT
ぐるんが

ノーマライズ

例）カタカナをひらがなに正規化

- きろぐらむ -> きろぐらむ
- キログラム -> きろぐらむ

PGroongaのノーマライザー（デフォルト）

```
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, '㌔');
INSERT INTO normalizer_test VALUES (4, 'ｷﾛｸﾞﾗﾑ');
INSERT INTO normalizer_test VALUES (5, 'kiroguramu');
INSERT INTO normalizer_test VALUES (6, 'ｋｉｒｏｇｕｒａｍｕ');

SELECT * FROM normalizer_test WHERE content &@ 'kiroguramu';
```

PGroongaのノーマライザー（デフォルト）

```
SELECT * FROM normalizer_test WHERE content &@ 'kiroguramu';
 id |        content
----+----------------------
  5 | kiroguramu
  6 | ｋｉｒｏｇｕｒａｍｕ
(2 rows)
```

PGroongaのノーマライザー(デフォルト)

- ｋｉｒｏｇｕｒａｍｕ

↓

- kiroguramu

PGroongaのノーマライザー（デフォルト）

- NFKCを使った正規化
  - テキストエンコードがUTF-8の場合

PostgreSQLのノーマライザー（デフォルト）

```
SELECT * FROM normalizer_test
  WHERE content = (SELECT normalize('ｋｉｒｏｇｕｒａｍｕ', NFKC));
 id |  content
----+------------
  5 | kiroguramu
(1 row)
```

ノーマライザーの変更

適合率/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 &@ 'kiroguramu';
```

PGroongaのノーマライザー (NormalizerNFKC130)

```
SELECT * FROM normalizer_test WHERE content &@ 'kiroguramu';
 id |        content
----+-----------------------
  1 | キログラム
  2 | きろぐらむ
  3 | ㌔
  4 | ｷﾛｸﾞﾗﾑ
  5 | kiroguramu
  6 | ｋｉｒｏｇｕｒａｍｕ
(6 rows)
```

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

```
CREATE INDEX pgroonga_content_index
          ON normalizer_test
       USING pgroonga (content)
        WITH (normalizers='NormalizerNFKC130("unify_to_romaji", true)');
```

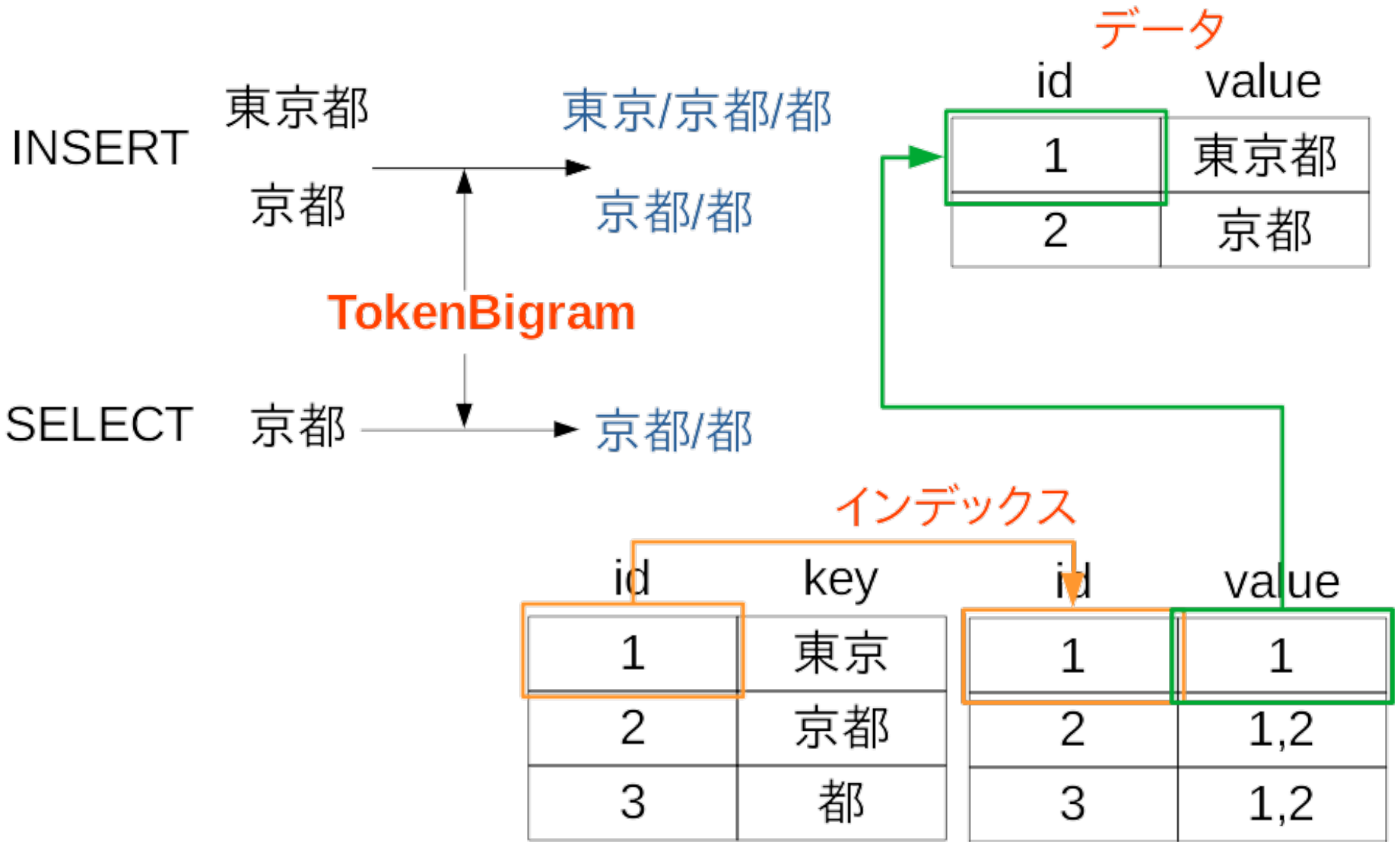
複数オプションの指定方法

```
CREATE INDEX pgroonga_content_index
          ON normalizer_test
       USING pgroonga (content)
        WITH (normalizers='NormalizerNFKC130("unify_to_romaji", true,
                                             "unify_hyphen", 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

トークナイズ


INSERT
東京都
京都
東京/京都/都
京都/都
TokenBigram
SELECT
京都
京都/都
データ
id
value
1
東京都
2
京都
インデックス
id
key
1
東京
2
京都
3
都
id
value
1
1
2
1,2
3
1,2

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）

```
SELECT * FROM tokenizer_test WHERE title &@~ '京都';
                 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
(5 rows)
```

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)

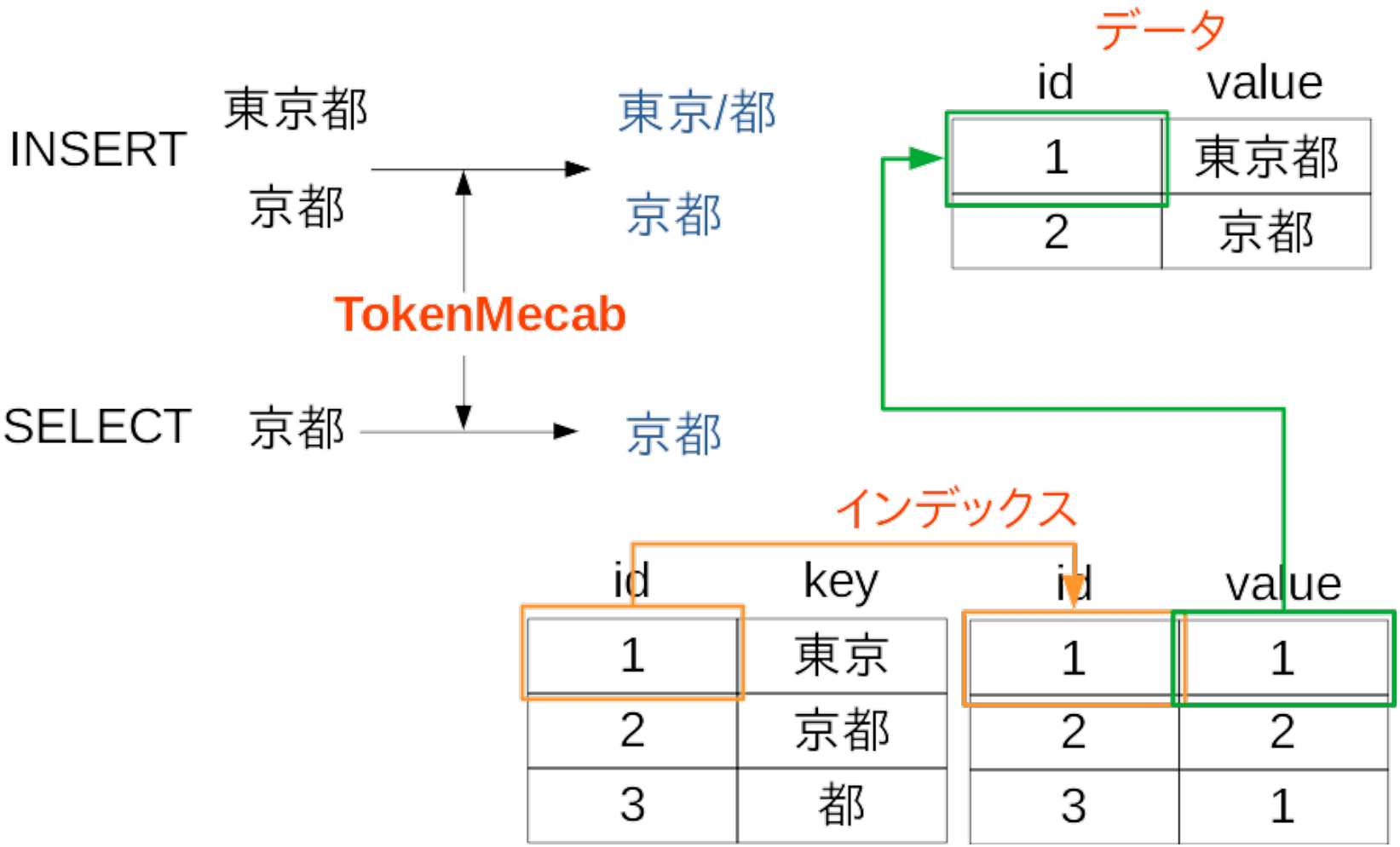
```
CREATE INDEX pgroonga_content_index
          ON tokenizer_test
       USING pgroonga (title)
        WITH (tokenizer='TokenMecab');

SELECT * FROM tokenizer_test WHERE title &@~ '京都';
```

PGroongaのトークナイザー (TokenMecab)

```
SELECT * FROM tokenizer_test WHERE title &@~ '京都';
                 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
(3 rows)
```

PGroongaのトークナイザー (TokenMecab)



トークナイザーの指定方法

```
CREATE INDEX pgroonga_content_index
          ON tokenizer_test
       USING pgroonga (title)
        WITH (tokenizer='TokenMecab');
```

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)

意味は同じだが
語の形が変わる
(語形変化)

ステミング(語幹処理)

例えば

- develop(原形)
- developed(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
developed
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 steming_test (
  title text
);
CREATE INDEX pgroonga_content_index ON steming_test USING pgroonga (title);

INSERT INTO steming_test VALUES ('I develop Groonga');
INSERT INTO steming_test VALUES ('I am developing Groonga');
INSERT INTO steming_test VALUES ('I developed Groonga');

SELECT * FROM steming_test WHERE title &@~ 'develop';
```

PGroongaのステミング（未使用）

```
SELECT * FROM steming_test WHERE title &@~ 'develop';
        title
--------------------
 I develop Groonga
(1 row)
```

PGroongaのステミング

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

INSERT INTO steming_test VALUES ('I develop Groonga');
INSERT INTO steming_test VALUES ('I am developing Groonga');
INSERT INTO steming_test VALUES ('I developed Groonga');

SELECT * FROM steming_test WHERE title &@~ 'develop';
```

PGroongaのステミング

```
SELECT * FROM steming_test WHERE title &@~ 'develop';
           title
------------------------
 I develop Groonga
 I am developing Groonga
 I developed Groonga
(3 rows)
```

高度な話題

処理順序

1. ノーマライズ
2. トークナイズ
3. トークンフィルター

高度な話題

処理順序が問題になることがある

高度な話題

TokenMecab と unify_kana

高度な話題

- 「ワールドカップ」->
  「わーるどかっぷ」

高度な話題

- 「ワールドカップ」 ->
  「ワールドカップ」
- 「わーるどかっぷ」 ->
  「わ/ー/る/ど/かっぷ」

高度な話題

処理順序

1. ノーマライズ
2. トークナイズ
3. トークンフィルター

高度な話題

TokenFilterNFKC100を使う

高度な話題

- TokenFilterNFKC100
  - 働きは、NormalizerNFKC100と同じ
  - トークナイズ後にノーマライズしたいときに使う

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

```
SELECT * FROM memos
  WHERE
    content &@~
      pgroonga_query_expand('synonyms', 'term', 'synonyms', '牛乳');

 id |       content
----+--------------------
  1 | 牛乳石鹸
  2 | ミルクジャム
(2 rows)
```

fuzzy検索

typo対策

fuzzy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）

fuzzy検索

「テノクロジー」
で
「テクノロジー」
がヒット

fuzzy検索
編集距離

- Aを何回操作するとBになるか
- 操作とは？
  - 文字の挿入・削除・置換・隣接文字交換
- 操作回数を距離とする

fuzzy検索
編集距離

- A：テノクロジー
- 隣接文字交換：ク⇔ノ
- B：テクノロジー

編集距離：1

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('テノクロジー') || ',
                     {"with_transposition": true,
                      "max_distance": 1})');
```

fuzzy検索

```
SELECT name FROM tags
  WHERE
    name &`
      ('fuzzy_search(name, ' || pgroonga_escape('テノクロジー') || ',
                     {"with_transposition": true,
                      "max_distance": 1})');
        name
--------------------
 テクノロジー
(1 row)
```

PGroongaでランキング改善

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(デフォルト)
- TF-IDF
- TF at Most

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ーが高くなる

PGroongaのスコアリング TF(デフォルト)

```
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 ('PGroonga1', 'PGroongaは全文検索エンジンGroongaを使っています。');
INSERT INTO memos VALUES ('PGroonga2', 'Groonga、Groonga、Groonga、Groonga、Groonga');
```

PGroongaのスコアリング TF(デフォルト)

```
SELECT *, pgroonga_score(tableoid, ctid) AS score
  FROM memos
 WHERE content &@~ 'Groonga'
 ORDER BY score DESC;

   title   |                                  content                                  | score
-----------+---------------------------------------------------------------------------+-------
 PGroonga2 | Groonga、Groonga、Groonga、Groonga、Groonga                                |     5
 Groonga   | Groongaは日本語対応の高速な全文検索エンジンです。                         |     1
 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。 |     1
 PGroonga1 | PGroongaは全文検索エンジンGroongaを使っています。                         |     1
(4 rows)
```

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

```
SELECT *, pgroonga_score(tableoid, ctid) AS score
  FROM memos
 WHERE content &@~
   ('Groonga OR 全文検索',
    ARRAY[1],
    ARRAY['scorer_tf_idf($index)'],
    'pgroonga_memos_index')::pgroonga_full_text_search_condition_with_scorers
 ORDER BY score DESC;

   title   |                                    content                                    | score
-----------+-------------------------------------------------------------------------------+-------
 Groonga   | Groongaは日本語対応の高速な全文検索エンジンです。                                |     2
 PGroonga1 | PGroongaは全文検索エンジンGroongaを使っています。                               |     2
 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。 |     1
 PGroonga2 | Groonga、Groonga、Groonga、Groonga、Groonga                                   |     1
(4 rows)
```

PGroongaのスコアリング
TF at Most

スコアーの
最大値を制限

PGroongaのスコアリング TF at Most

```
SELECT *, pgroonga_score(tableoid, ctid) AS score
  FROM memos
WHERE content &@~ 'Groonga OR PostgreSQL'
ORDER BY score DESC;

   title    |                                content                                 | score
------------+------------------------------------------------------------------------+-------
 PGroonga2  | Groonga、Groonga、Groonga、Groonga、Groonga                             |     5
 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。 |     2
 Groonga    | Groongaは日本語対応の高速な全文検索エンジンです。                      |     1
 PGroonga1  | PGroongaは全文検索エンジンGroongaを使っています。                      |     1
 PostgreSQL | PostgreSQLはリレーショナル・データベース管理システムです。             |     1
(5 rows)
```

PGroongaのスコアリング TF at Most

```
SELECT *, pgroonga_score(tableoid, ctid) AS score
  FROM memos
 WHERE content &@~
   ('Groonga OR 全文検索',
    ARRAY[1],
    ARRAY['scorer_tf_at_most($index, 1)'],
    'pgroonga_memos_index')::pgroonga_full_text_search_condition_with_scorers
 ORDER BY score DESC;

   title   |                                    content                                    | score
-----------+-------------------------------------------------------------------------------+-------
 Groonga   | Groongaは日本語対応の高速な全文検索エンジンです。                                |     2
 PGroonga1 | PGroongaは全文検索エンジンGroongaを使っています。                               |     2
 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。 |     1
 PGroonga2 | Groonga、Groonga、Groonga、Groonga、Groonga                                   |     1
(4 rows)
```

参考資料

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

参考資料

- ノーマライザーのオプション一覧
  - https://groonga.org/ja/docs/reference/normalizers/normalizer_nfkc130.html#parameters

参考資料

- 使用可能なトークナイザー一覧
  - https://groonga.org/ja/docs/reference/tokenizers.html

参考資料

- ステミングの使用方法
  - https://pgroonga.github.io/ja/reference/create-index-using-pgroonga.html

参考資料

- 同義語検索の方法
  - https://pgroonga.github.io/ja/how-to/synonyms.html

参考資料

- 使用可能なスコアラー一覧
  - https://groonga.org/ja/docs/reference/scorer.html#built-in-scorers

参考資料

- スコアラーの設定方法
  - https://pgroonga.github.io/ja/reference/operators/query-v2.html