ADC
Applaudo
Developers
Conference
2020

Powered by ApplaudoStudios

What did you say?

Making sense of user input
with PostgreSQL

Kevin Aleman

Tecnologías

@kaleman15

@kevin-alemán

@KevLehman

@kaleman15

Agenda

1. Qué es fuzzy search?
2. Cómo buscamos?
3. Cómo buscar mejor?
3.1 Trigrams en la vida real
3.2. Trigrams en PostgreSQL
4. Ejemplo

Qué es fuzzy search?

Fuzzy (english - adj): dificultad para percibir o explicar algo a simple vista. Difuso, indistinto

Qué es fuzzy search?

Aunque no sean iguales

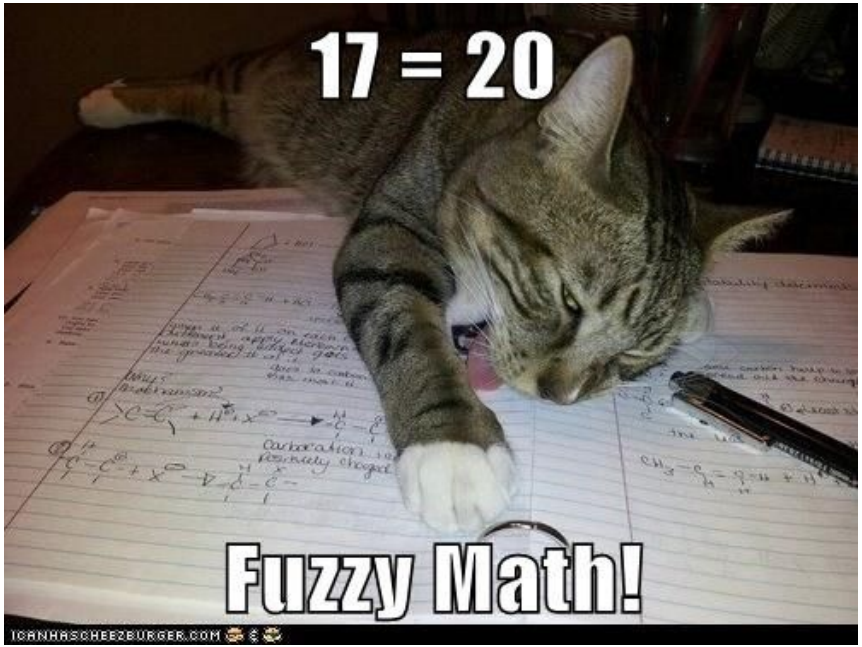
Es un tipo de búsqueda donde se retornan resultados que no son precisamente “iguales” al valor recibido, pero que tienen cierto grado de similaridad.

Eg: lavadora vs lvadora

Usa algoritmos “fuzzy”

Un algoritmo “fuzzy” busca esa similaridad entre ambos valores, y retorna los valores cuya similaridad sea más alta.

Usan “fuzzy logic”. Esta se basa en “rangos de verdad” en lugar de la típica lógica booleana de “verdadero” o “falso”


17 = 20
Fuzzy Math!
ICANHASCHEEZBURGER.COM

Cómo buscamos?

Técnicas de búsqueda comunes

Match exacto

Es un tipo de búsqueda donde se retornan resultados que son iguales (o contienen) el término de búsqueda del usuario

Eg: Término de búsqueda lava

Nuestra data: ILIKE %lava%

Match “casi” exacto

En este tipo de búsqueda, dividimos el valor entrado por el user y comparamos las palabras de forma individual contra la base, buscando algún match.

Eg: Término de búsqueda: mario ernesto perejil

Query generada: OR mario OR ernesto OR perejil

Técnicas de búsqueda comunes (2)

“Fuzzy” search

Este tipo de búsqueda trata de encontrar similitudes entre el término de búsqueda del usuario y la data que tenemos en nuestra base de datos.

Eg: Término de búsqueda: telesor

Query generada: ???

Búsqueda de texto completo

Este tipo de búsqueda es ideal para buscar en textos grandes, donde también nos importan factores como la cantidad de veces que una palabra se usa, palabras que son plural o derivadas de otras (eg: gobierno y gobernación)

Match exacto

Comparar de forma estricta ambas palabras. Generalmente, usando el valor de entrada como una “substring”

```
SELECT *
FROM users
WHERE full name ilike '%query param%';
```

Match exacto

VENTAJAS	DESVENTAJAS
Simple de usar	Casos de uso limitados
Compatible	Díficil de actualizar para usar queries complejas
Directa	No retorna valores si el término de búsqueda está mal escrito
Rápida	

Match “casi” exacto

Comparar los términos de búsqueda del usuario palabra por palabra contra lo que tengamos en la base

```
-- término: “mario perejil”

SELECT *
FROM users
WHERE (
  full name ilike ‘%mario%’
  OR full name ilike ‘%perejil%’
)
```

Match “casi” exacto

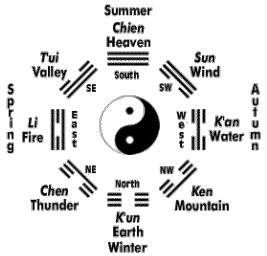
VENTAJAS	DESVENTAJAS
Simple de usar	Considerablemente más lento
Compatible	La query puede volverse muy compleja
Soporta más opciones de búsqueda	No retorna valores si el término de búsqueda está mal escrito
Retorna valores aunque los términos estén desordenados	

Cómo buscar mejor?

Trigrams

O cómo segmentar nuestras palabras

Qué es un trigram?



Lo sabía?

Los símbolos que rodean al logo Taoísta I-ching también son trigrams!

Una forma de dividir nuestras palabras

El concepto de trigram viene de un “n-gram”, que es una forma de dividir cualquier segmento de texto en grupos de igual longitud. Un trigram es un uso especial donde estos segmentos serán de 3 caracteres siempre

Se usan para analizar textos

Los n-grams se usan para analizar la frecuencia con la que grupos de caracteres aparecen en textos. Por ejemplo, según Wikipedia, el trigram más común es “the”.

También, se usan en criptografía para analizar patrones en el texto cifrado!

Trigrams en la vida real

```
-- Término: “applaudo studios”
-- Caracteres por grupo: 3

-- Resultado
app, ppl, pla, lau, aud, udo, do_
_st, stu, tud, udi, dio, ios
```

Nota: los _ representan espacios

Cómo usamos los trigrams para buscar?

Pasos

1. Dividimos el término de búsqueda en segmentos de 3 (trigrams)
2. Dividimos cada uno de los valores en nuestra data en grupos de 3 (trigrams)
3. Comparamos los trigrams del término de búsqueda con los de cada valor en nuestra data.
4. Calculamos la “similaridad”. Definimos “similaridad” como la cantidad de trigrams compartidos entre el término y el valor en nuestra data
    a. Para ello, usamos la siguiente fórmula

$$similaridad = \frac{trigramsencomun}{trigramsdelaprimerapalabra}$$

5. Ordenamos los resultados del más al menos similar

Trigrams

Ejemplo

Ejemplo - Comparemos “applaudo” con “applaudir”

Paso 1: Dividir “Applaudo” en segmentos

Applaudo = app, ppl, pla, lau, aud, udo, do_

Paso 2: Dividir “Applaudir” en segmentos

Applaudir = app, ppl, pla, lau, aud, udi, dir, ir_

Paso 3: Cuántos trigrams tienen en común?

Respuesta rápida: 5 trigrams en común

Paso 4: Usa la técnica, calamardo

$$similaridad = \frac{5}{7}$$

$$similaridad = 0.71$$

Paso 5: Ordenamos por similaridad!

71%

Similaridad entre ambas palabras

Muy bonito y todo, pero, debe haber una forma más fácil.

– Paulo Coelho, filósofo salvadoreño

Trigrams en PostgreSQL

```
select show_trgm('applaudo');
                  show_trgm
-------------------------------------------
 {"  a","  ap",app,aud,"do ",lau,pla,ppl,udo}


select show_trgm('applaudir');
                  show_trgm
-----------------------------------------------
 {"  a","  ap",app,aud,dir,"ir ",lau,pla,ppl,udi}
```

Cómo usamos los trigrams para buscar en PostgreSQL?

Pasos

1. Usamos la función “word_similarity” para calcular la “similaridad” entre ambas palabras.
2. Y ya, no hay más pasos…

Ejemplo - Comparemos “applaudo” con “applaudir”

Paso 1: Usemos la función y ya…

```
select    word_similarity(“applaudo”,  “applaudir”);
 word_similarity
-----------------
        0.777778
```

77%

Similaridad entre ambas palabras

La similaridad subió, por qué?

Subió porque la cantidad de trigrams generados por PostgreSQL para la comparación es mayor. PostgreSQL agrega espacios al inicio y al final de las palabras, para balancear el match

“Fuzzy” search

Comparar los términos de búsqueda del usuario palabra por palabra contra lo que tengamos en la base

```
-- término: “mario”

SELECT *
FROM users
WHERE
  word_similarity(name, ‘mario’) > 0.5
ORDER BY
  word_similarity(name, ‘mario’);
```

Nota: 0.5 es un valor arbitrario, puede ser cualquier otro valor.

“Fuzzy” Search

VENTAJAS	DESVENTAJAS
Matches más completos	Se necesita instalar extensiones extra
Proceso más robusto	Se necesita conocimiento extra para debuggear
Retorna valores aunque los términos estén desordenados o mal escritos	Puede ser más lenta o más rápida dependiendo de varios factores

Ejemplo:
Fuzzy search en acción

Links útiles

API

Website

Fuzzy Search - Blog

Sección de preguntas

Applaudo
Developers
Conference
2020

Powered by ApplaudoStudios