Information Retrievel with PostgreSQL

Alexander Hebel

Heidelberg University
Institute of Computer Science
Database Systems Research Group
vx228@uni-heidelberg.de

Mai 6, 2020

Outline

1. Introduction
2. Approach and realizations
3. Custom C-functions in PostgreSQL
4. Rating sections vs. rating pages
5. Conclusion

Outline

1. Introduction
2. Approach and realizations
3. Custom C-functions in PostgreSQL
4. Rating sections vs. rating pages
5. Conclusion

Task definition

- How looks and performs an IRS made of a relational database
- Similar to Apache Solr
- Finding different database models
- Python api for the database creation and communication
- Crawl Wikipages to gather some text data
- Special type in PostgreSQL named tsvector (full text search)

First goal

Support some boolean search querys like AND

Outline

1. Introduction
2. Approach and realizations
3. Custom C-functions in PostgreSQL
4. Rating sections vs. rating pages
5. Conclusion

Realization

Wiki crawler

- Based on package wikipedia version 1.4.0
- Takes number of pages and category as input
- Also searches in subcategories
- Variable level of subcategories

Database pipeline

- Used package psycopg2 version 2.8.5
- custom converter for tsvector

3 database model approaches

tsvector

- all done with tsvector
- full text search with tsvector
- rating of tsvector
- weighting of tsvector
- tokenization and lemmatization
- tsquery

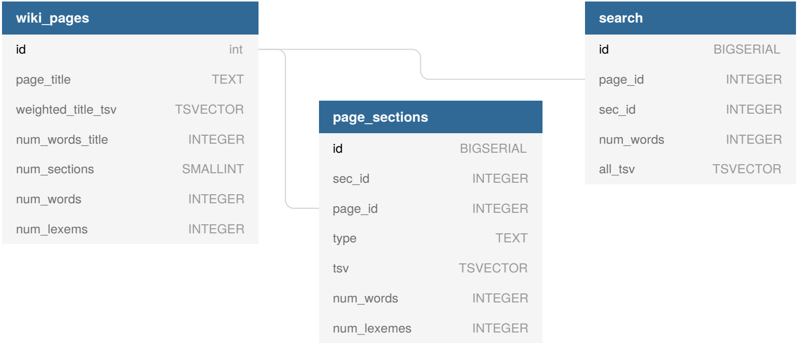
Mix

- raw tsvecot + word-matrix for each doc
- needs a lot of memory
- customizable

word-matrix

- one big word-matrix
- probably slow

Database model



wiki_pages	
id	int
page_title	TEXT
weighted_title_tsv	TSVECTOR
num_words_title	INTEGER
num_sections	SMALLINT
num_words	INTEGER
num_lexems	INTEGER

page_sections	
id	BIGSERIAL
sec_id	INTEGER
page_id	INTEGER
type	TEXT
tsv	TSVECTOR
num_words	INTEGER
num_lexemes	INTEGER

search	
id	BIGSERIAL
page_id	INTEGER
sec_id	INTEGER
num_words	INTEGER
all_tsv	TSVECTOR

Tsvector

Possibilities

- Full text search
- GIN-Index
- Automatic tokenization and lemmatization
- Adding weights
- Predefined rating function

Limitations

- The number of lexemes must be less than 2$\hat{6}$4
- Max position value: 16383
- No more than 256 positions per lexeme
- Relative small set of manipulation methods
- Limited rating

Example

{'a':1,6,10 'and':8 'cat':3 'fat':2,11 'mat':7 'on':5 'rat':12 'sat':4}

Outline

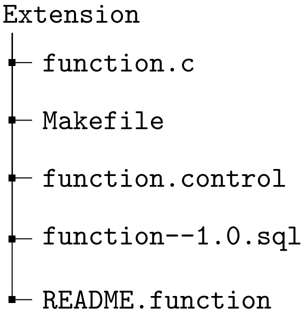
1. Introduction
2. Approach and realizations
3. Custom C-functions in PostgreSQL
4. Rating sections vs. rating pages
5. Conclusion

Adding your custom C-functions to PostgreSQL

Prerequisites

- Developer version of PostgreSQL
- Installation of make
- Root privilege on database

Folder structure



```
Extension
├─ function.c
├─ Makefile
├─ function.control
├─ function--1.0.sql
└─ README.function
```

Steps

(1) make install

(2) CREATE EXTENSION "extension"

Example

```c
#include "postgres.h"
#include "fmgr.h"
#include "utils/geo_decls.h"

#ifdef PG_MODULE_MAGIC
        PG_MODULE_MAGIC;
#endif

PG_FUNCTION_INFO_V1(add_one);

Datum
add_one(PG_FUNCTION_ARGS)
{
        int32   arg = PG_GETARG_INT32(0);
        PG_RETURN_INT32(arg + 1);
}
```

Outline

1 Introduction

2 Approach and realizations

3 Custom C-functions in PostgreSQL

4 Rating sections vs. rating pages

5 Conclusion

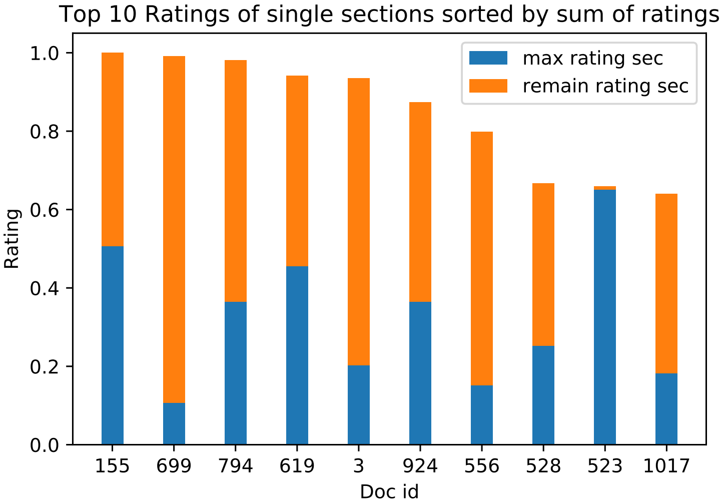
Idea

- Originates from a misunderstanding
- Thought the task is to rank whole wiki pages
- User wants the best section and not the "best" document
- So how is the relationship between page and section ranking

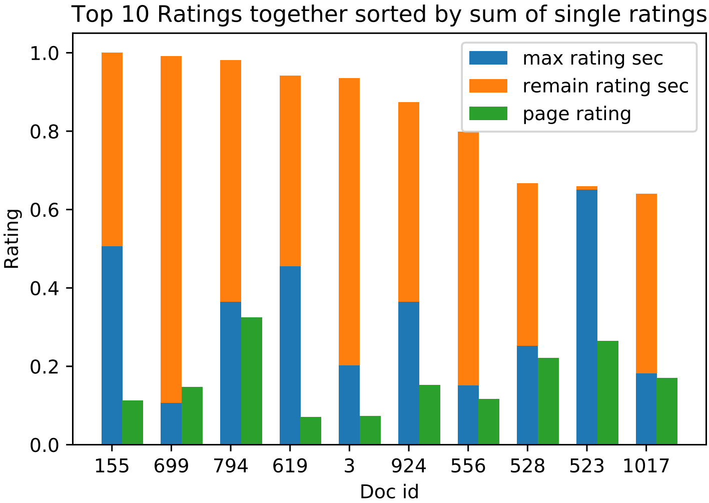
Calculation of Rating

- **section**: rating / num_words_of_section
- **page**: sum_of_ratings / num_words_of_page

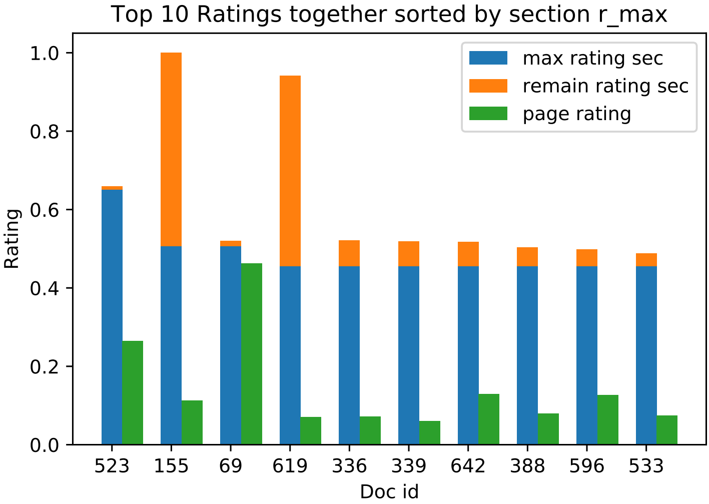
Query:"game", sorted by sum of section rankings



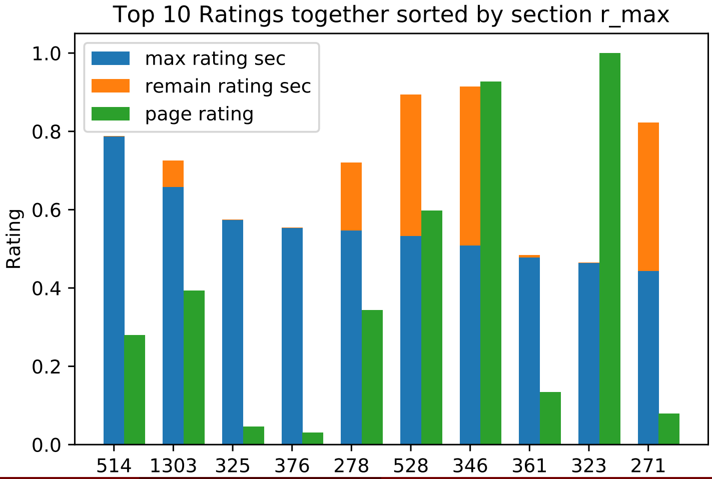
Query:"game", adding the rank for the whole page



Query:"game", ordered by max section rating



Query:"game AND team AND ball", ordered by max section rating



distance between ratings

Calculating Distance

- To calculate the difference between pageranking and sectionranking
- Distance first of max section rank to first of max page rank, distance of second max section ranking to second of max page rank and so on...

Query "game"

- result pages: 1176
- Top 10 - 289.5 avg(dist)
- Top 20 - 250.9 avg(dist)
- Top 30 - 183.2 avg(dist)
- Top 40 - 155.5 avg(dist)

Query "game & team & ball"

- result pages: 274
- Top 10 - 36.0 avg(dist)
- Top 20 - 40.6 avg(dist)
- Top 30 - 36.6 avg(dist)
- Top 40 - 34.0 avg(dist)

Outline

1. Introduction
2. Approach and realizations
3. Custom C-functions in PostgreSQL
4. Rating sections vs. rating pages
5. Conclusion

Conclusion and future work

Conclusion

- Ratings for sections and page return total different results
- Tsvector has a lot of potential
- PostgreSQL is easy customizable
- TODO comparing performance and ranking to solr

Future work

- Improve the rating algorithm with tf idf information (ts_stat)
- Tests on big datasets

Questions

Questions