

# Information Retrieval with PostgreSQL

Alexander Hebel

Heidelberg University  
Institute of Computer Science  
Database Systems Research Group  
[vx228@uni-heidelberg.de](mailto:vx228@uni-heidelberg.de)

Mai 6, 2020

# Outline

- 1 Introduction
- 2 Approach and Realizations
- 3 Adding Functionality to Tsvectors
- 4 Evaluation Objectives
- 5 Conclusion

# Outline

- 1 Introduction
- 2 Approach and Realizations
- 3 Adding Functionality to Tsvectors
- 4 Evaluation Objectives
- 5 Conclusion

# Information Retrieval System (IRS)

## Information Retrieval (IR)

- Information retrieval is the science of searching for information in a document
- An IR system is a software system that provides access to books, journals and other documents; stores and manages those documents.
- Web search engines are the most visible IR applications.

## Full-Text-Search

The activity of searching through a collection of natural-language documents to locate those that best match a query

# Objectives

IRS already exist.

But to our best knowledge there is no IRS made of a relational database.

- Finding different suitable database models
- Python api for the database creation and communication
- Crawl Wikipages to gather text data
- Store and manage those text documents
- Support search queries with the boolean operator AND
- Create a ranking function for the query results
- Compare the ranking of whole wiki pages and their sections

# Tsvector and Tsquery

- PostgreSQL provides **two** data types that are designed to support full text search
- The **tsvector** type represents a document in a form optimized for text search
- The **tsquery** type similarly represents a text query

## Example Tsvector

```
SELECT to_tsvector('english', 'The Fat Rats');  
{'fat':2 'rat':3}
```

## Example Tsquery

```
SELECT to_tsquery('Fat:ab & Cats');  
{'fat':AB & 'cat'}
```

# Outline

- 1 Introduction
- 2 Approach and Realizations**
- 3 Adding Functionality to Tsvectors
- 4 Evaluation Objectives
- 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

## Document Definition

A document in my case is either a page title, section title or section text



# Text Data Statistics

## Wikipedia Category "Sports"

- Number of Wiki pages: 2000
  - Number of sections (captions also count as section): 45756
  - Total number of words: 17587832
  - Total number of lexemes: 1969427
- 
- Max number of words of a page: 124136
  - Max number of lexemes of a page: 17787
- 
- Max number of words of a section: 24163
  - Max number of lexemes of a section: 2785
  - Average number of words per section: 734.7

# Text Data Statistics 2

## Top 10 Frequent Terms

(1) "game"	TF: 14044	DocF: 5370
(2) "world"	TF: 13568	DocF: 6144
(3) "championship"	TF: 12106	DocF: 5475
(4) "team"	TF: 11955	DocF: 4362
(5) "sport"	TF: 10658	DocF: 5014
(6) "1"	TF: 10237	DocF: 4029
(7) "first"	TF: 10199	DocF: 5073
(8) "player"	TF: 9577	DocF: 3243
(9) "2"	TF: 9090	DocF: 3821
(10) "play"	TF: 9082	DocF: 3827

TF = term frequency

DocF = document frequency

# Three Database Model Approaches

## Tsvector

- full text search with tsvector
- ranking function of tsvector
- weighting of tsvector
- tokenization and lemmatization
- tsquery

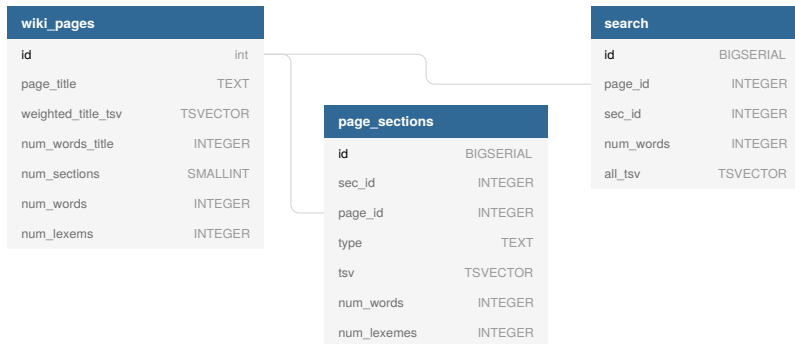
## Mix

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

## Word-matrix

- one big word-matrix
- probably slow

# Database model



# Tsvector

## Possibilities

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

## Limitations

- The number of lexemes must be less than  $2^{64}$
- Max position value: 16383
- No more than 256 positions per lexeme
- Relative small set of manipulation methods
- Ranking function must be customized

# Outline

- 1 Introduction
- 2 Approach and Realizations
- 3 Adding Functionality to Tsvectors**
- 4 Evaluation Objectives
- 5 Conclusion

# Reason to Extend PostgreSQL

Only a few manipulation methods for tsvector exist

For example it is not possible to:

- count the occurrence of **all** lexemes in a tsvector (not only the number of distinct lexemes)
- create a tsvector with an offset for the position index
- get the max index of a tsvector
- concatenate two tsvecotrs without changing the position values

## Solution

Write your own function either in plpgsql language or in C

# Custom C-Functions in 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"



# Outline

- 1 Introduction
- 2 Approach and Realizations
- 3 Adding Functionality to Tsvectors
- 4 Evaluation Objectives**
- 5 Conclusion

# Evaluation Objectives

- Three different database models have been implemented
- The model mainly using tsvector has proven most suitable
- + GIN index for performance
- + ts\_rank() function for ranking
- + ts\_query even supports phraseto\_tsquery()

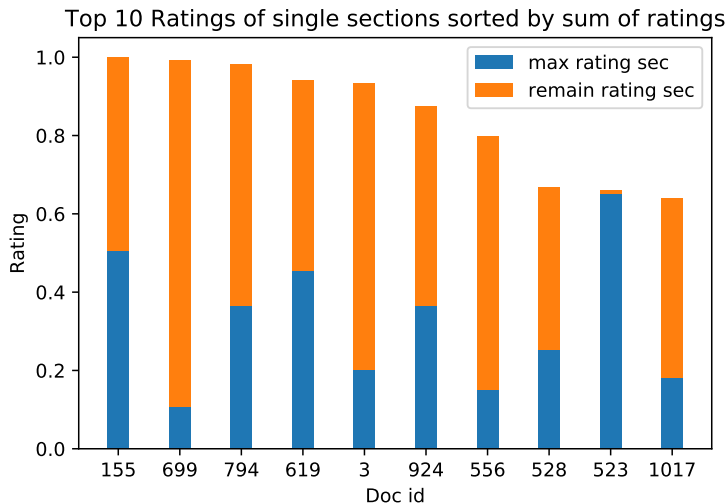
## Calculation of Ranking

- Either with ts\_rank() or Cover Density Ranking (CDR)
- Considers word occurrences and distance between words

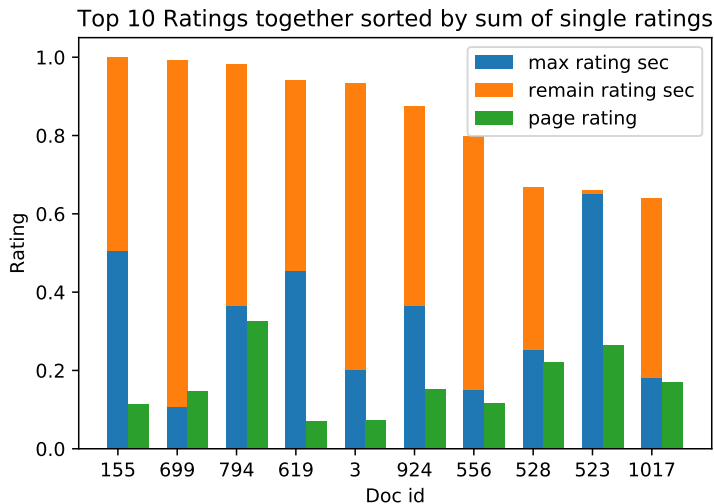
## Relationship Between Page and Section Ranking

- **section ranking:**  $\text{ranking} / \text{num\_words\_of\_section}$
- **page ranking:**  $\text{sum\_of\_rankings} / \text{num\_words\_of\_page}$

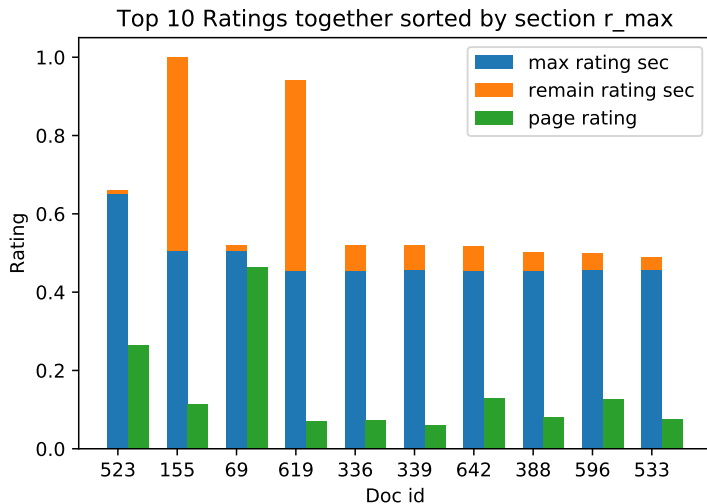
# Query:"game", Sorted by Sum of Section Rankings



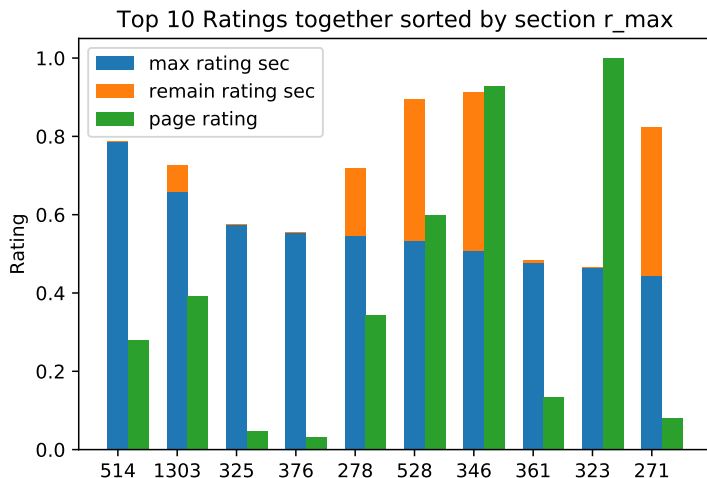
# Query:"game", Adding The Rank for the Whole Page



# Query:"game", Ordered by Max Section Ranking



# Query:"game AND team AND ball", Ordered by Max Section Ranking



# Distance Between Rankings

## 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 Adding Functionality to Tsvectors
- 4 Evaluation Objectives
- 5 Conclusion**



# Conclusion and Future Work

## Conclusion

- Rankings 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 ranking algorithm with tf idf information (ts\_stat)
- Tests on big datasets

# Questions

## Questions