TECH
VAULT

FULL TEXT SEARCH

AMR ELHELW

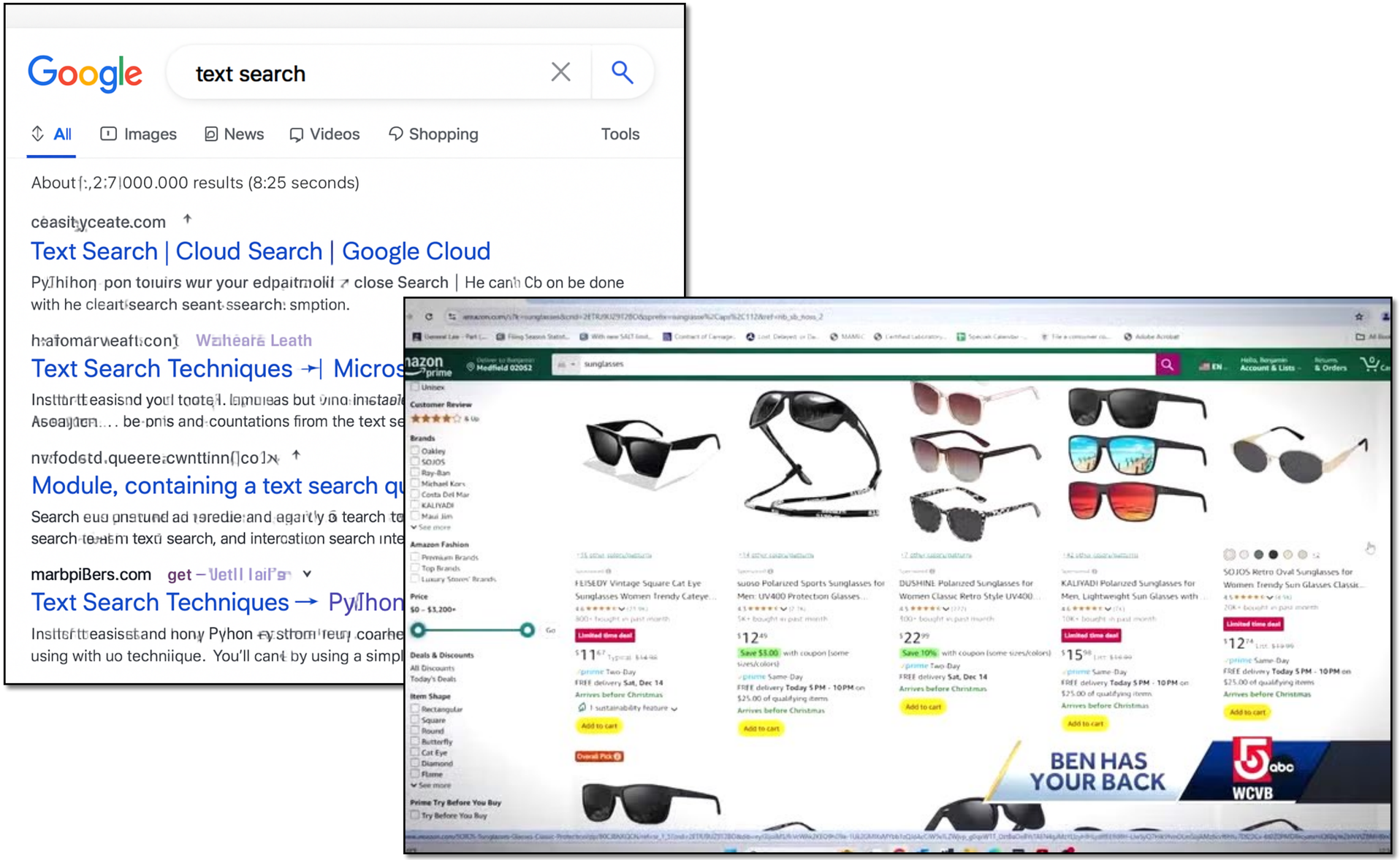
Finding words within large text fields

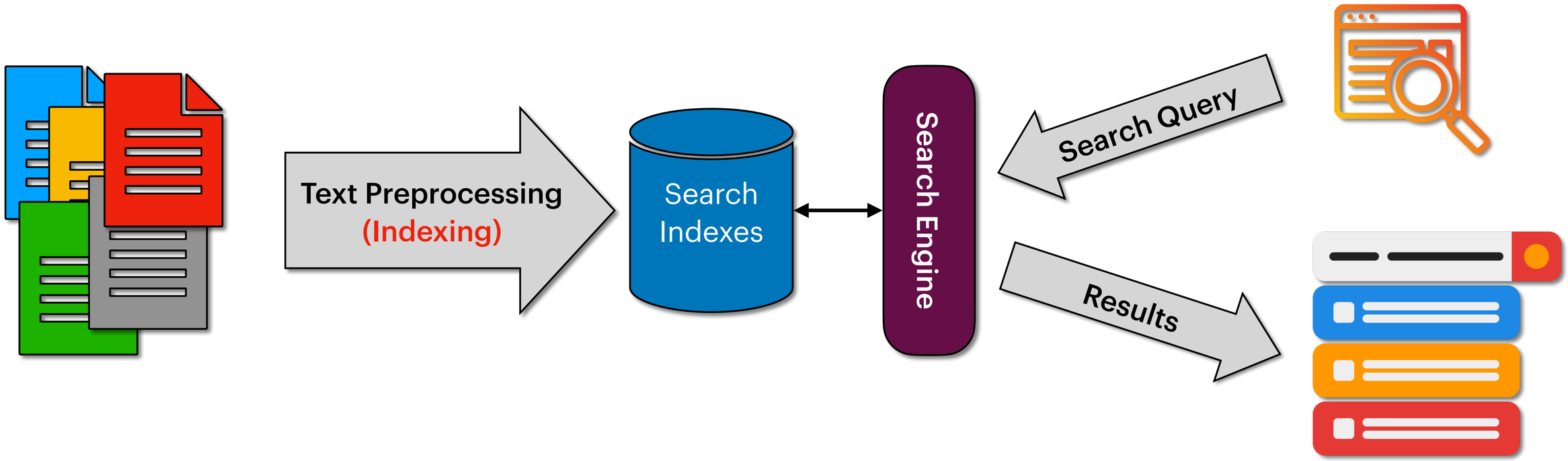
Matching partial words or variations of a term

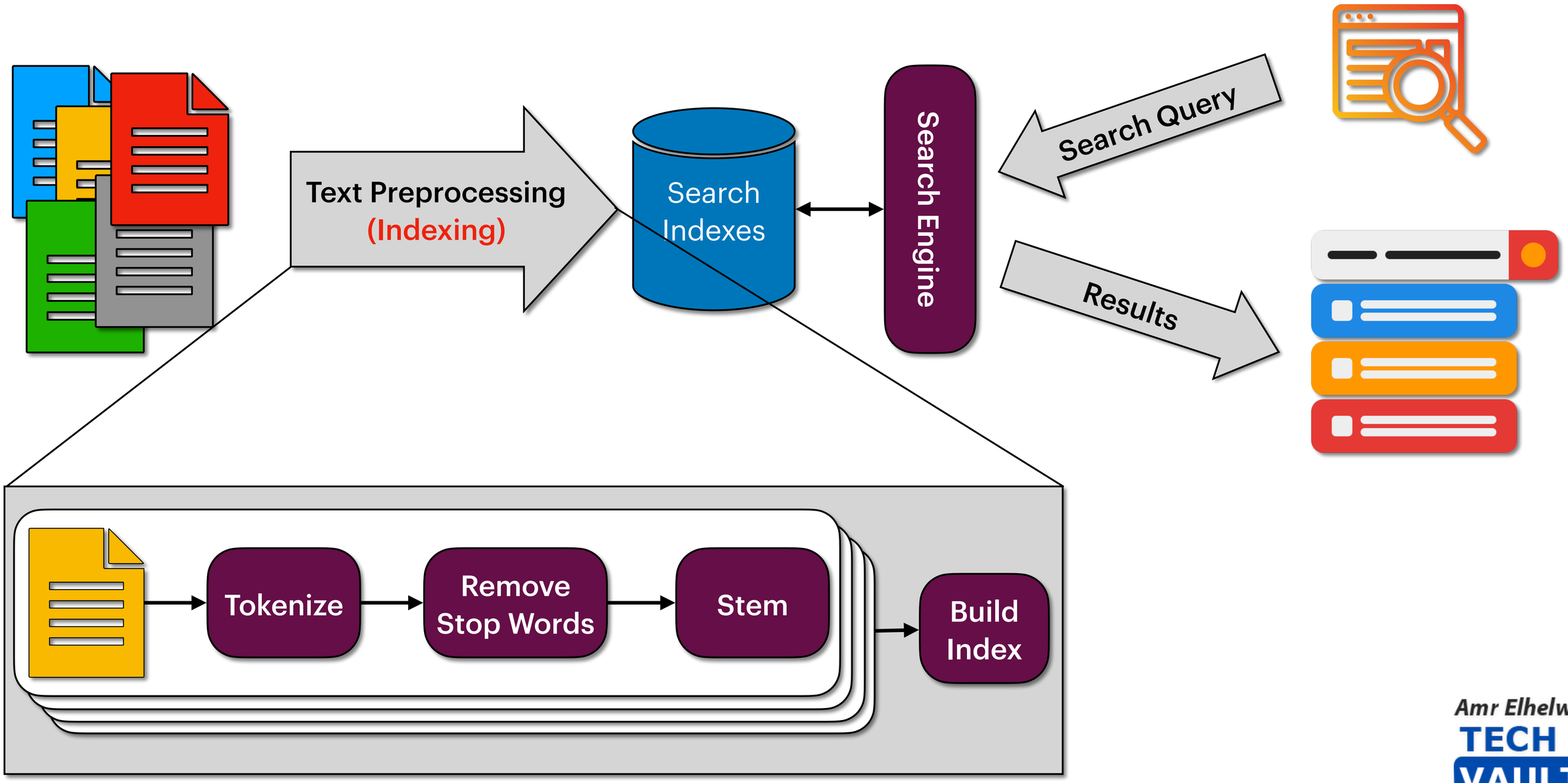
Ranking results based on relevance

Simple keyword/tag matching

Comparing structured fields or exact values







Tokenization

- Breaking down text into individual units (tokens)

- **Example**
  - Input: *"The quick brown fox jumps over the lazy dog"*
  - Tokenized:

    ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]

Stop word Removal

- Removing common words that are not particularly meaningful in search
- Examples: “the”, “a", “is”, “on”, “for”

- **Example**
  - Input: ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", “dog"]
  - Ouput: ["quick", "brown", "fox", "jumps", "lazy", “dog"]

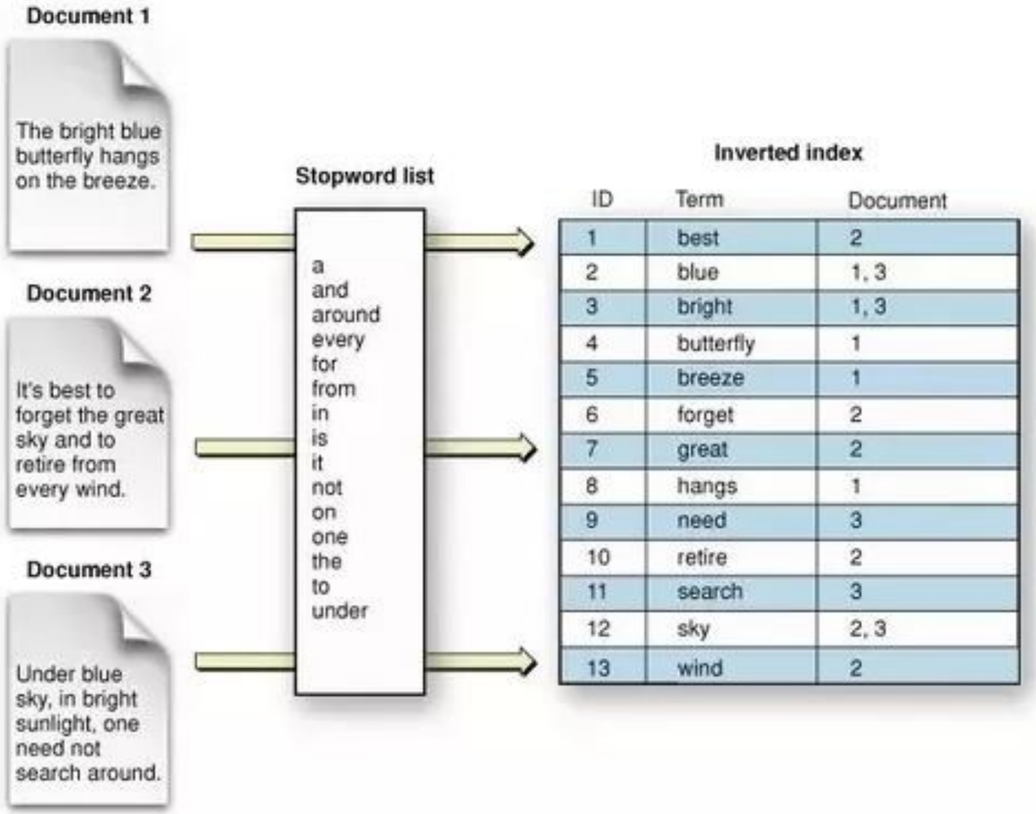
Stemming

- Reducing words to their root form
- Example: "running" → "run", "cats" → "cat"

- **Example**
  - Input: ["quick", "brown", "fox", "jumps", "lazy", “dog"]
  - Ouput: ["quick", "brown", "fox", "jump", "lazy", “dog"]

Inverted Index

- A data structure that maps terms to the documents they appear in.



Database queries

```
WHERE dept = 'support'
```

'customer support' 

```
WHERE order_status IN ['delivered', 'cancelled']
```

'delivered_' 

Text Search Features in PostgreSQL

Data Types

- *tsvector* – represents preprocessed text for full-text search
- *tsquery* – represents a structured form of a search query

Functions and operations

- *to_tsvector(<text>)* — converts a text input to a *tsvector*.
  This returns data that can then be “searched”.
- *to_tsquery(<text>)* — converts search terms to a *tsquery*.
- *<tsvector> @@ <tsquery>* — check whether a *tsvector* satisfies a *tsquery*