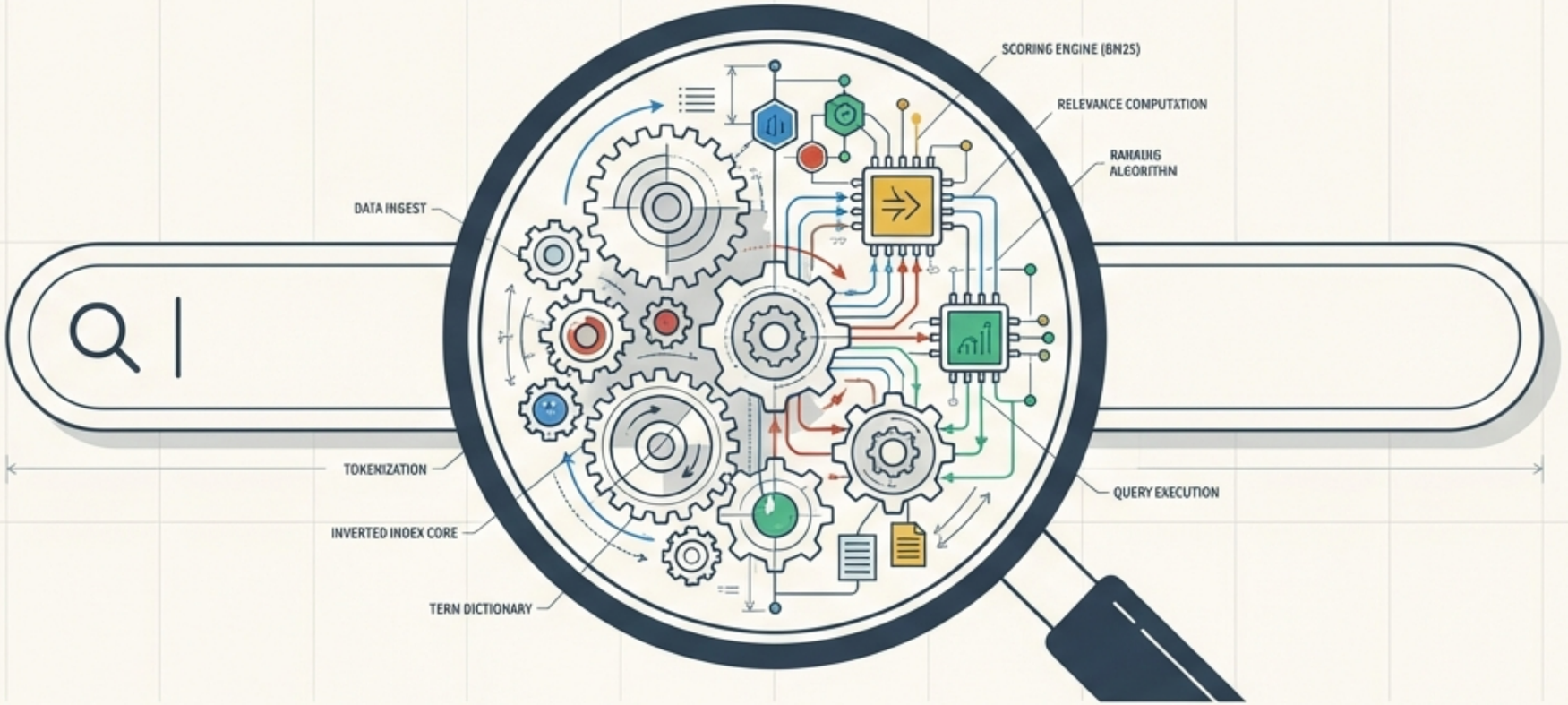


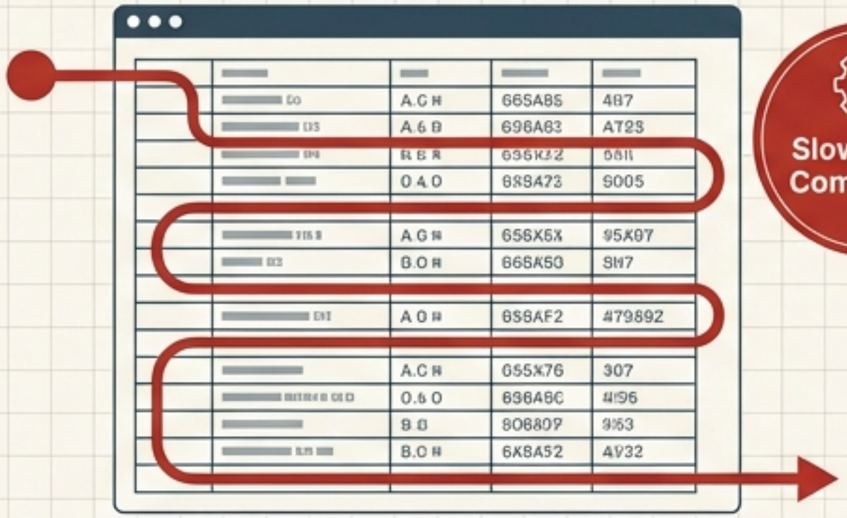
Full-Text Search: From Fundamentals to BM25

A comprehensive guide to indexing mechanics, ranking algorithms, and practical MySQL implementation.

THE CONTEXT: We type. We wait milliseconds. We get answers. This deck peels back the layers of that interaction, moving from the inverted index (speed) to probabilistic ranking (relevance).

The Problem: Scanning vs. Seeking

The Old Way: SQL LIKE



Slow / O(n) Complexity

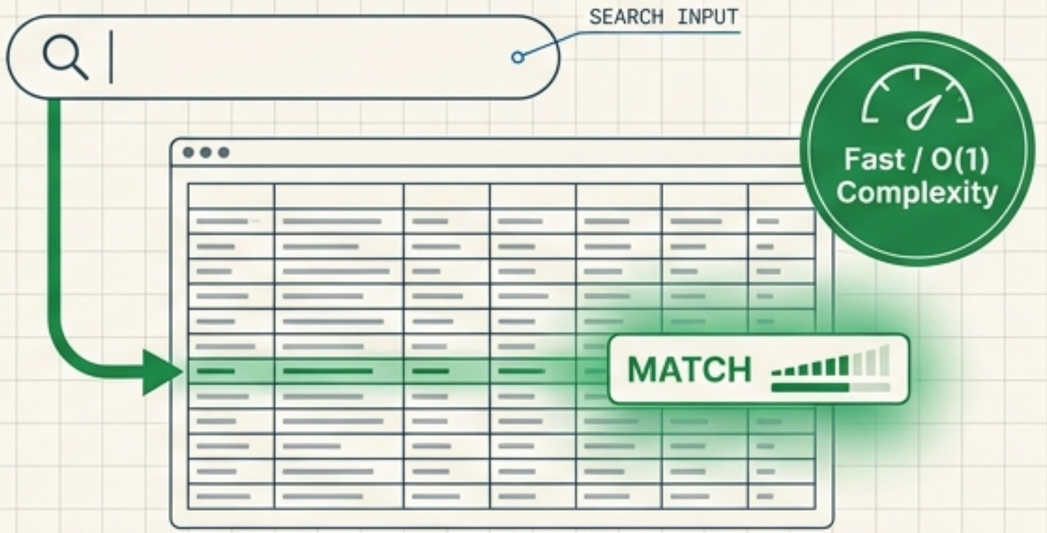
Efficiency: Scans every row in the table.
Intelligence: Binary match only. No ranking.
Language: Fails on synonyms ('car' vs 'auto').

CODE

```
SELECT * FROM docs WHERE body LIKE '%pattern%'
```

Linear Scan Operation
Linear Scan Creration
Binary Pattern Match

The FTS Way: Inverted Index



Fast / O(1) Complexity

Efficiency: Uses index for direct lookup.
Intelligence: Calculates Relevance Ranking.
Language: Understands stems and nuances.

CODE

```
MATCH(body) AGAINST('pattern')
```

Inverted Index Lookup
Relevance Scoring & Linguistic Analysis

PROJECT: SEARCH OPTIMIZATION	DATE: OCT 26, 2023
DRAWN BY: SYSTEM ARCHITECT	SHEET NO: 01 OF 01

The Engine: The Inverted Index

The secret to search speed is inverting the data structure. We map words to documents, rather than documents to words.

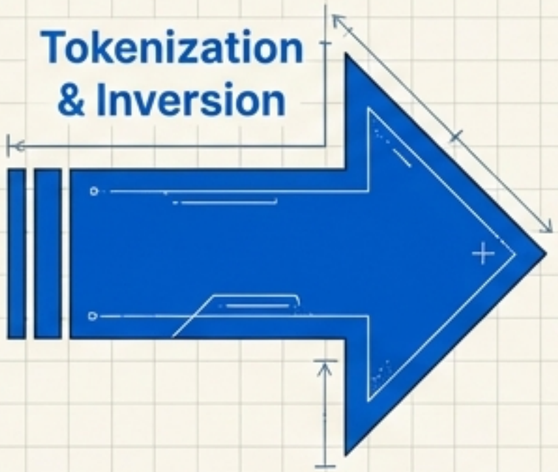
Documents

ID: 1 | Content: "Python is great"

ID: 2 | Content: "Java is verbose"

ID: 3 | Content: "Python implies coding"

The Inversion



Dictionary

Term	Postings List (Doc IDs)
python	[1, 3]
is	[1, 2]
great	[1]
java	[2]
coding	[3]

Technical Detail: Real-world indexes also store Metadata.
1. **Position:** To enable phrase search (are words neighbors?).
2. **Frequency:** To enable relevance scoring (how often does it appear?).

PROJECT: SEARCH OPTIMIZATION	DATE: OCT 28, 2023
DRAWN BY: SYSTEM ARCHITECT	SHEET NO: 01 OF 01

Building the Index: The Tokenization Pipeline

Before we index, we must normalize. This pipeline transforms unstructured human language into structured mathematical tokens.

Key Insight: Normalization ensures that searches for `'python'`, `'Python'`, and `'PYTHON'` all map to the same mathematical entity.

PROJECT: SEARCH OPTIMIZATION	DATE: OCT 28, 2023
DRAWN BY: SYSTEM ARCHITECT	SHEET NO: 01 OF 01

The Mathematics of Relevance: TF & IDF

Search Relevance = Frequency (TF) × Rarity (IDF)

Term Frequency (TF)

Definition: How loud is this word in this specific document?

Intuition: If “Python” appears 5 times in Document A and 1 time in Document B, Document A is likely more relevant.

Mathematical Implication: A local counter per document.

Inverse Document Frequency (IDF)

Definition: How rare is this word globally?

Intuition: “The” appears everywhere (Low Value). “Microservices” appears rarely (High Value).

Mathematical Implication: A global weight penalizing common words.

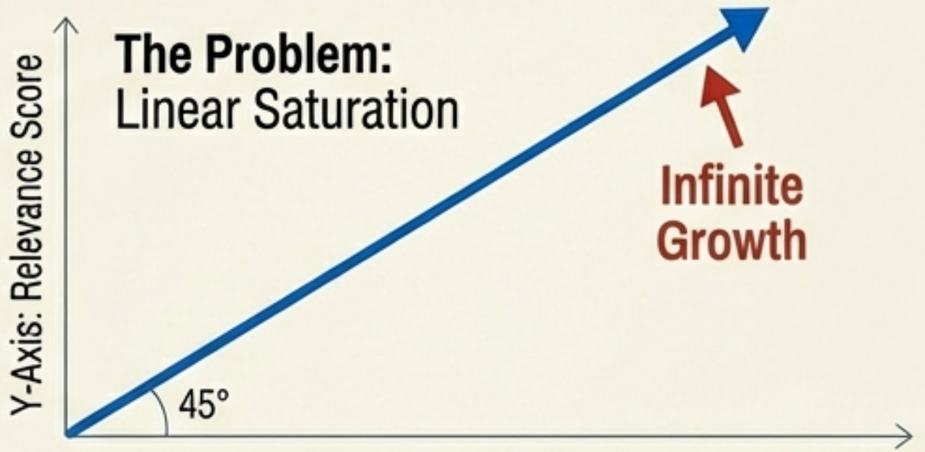
PROJECT: SEARCH OPTIMIZATION	DATE: OCT 28, 2023
DRAWN BY: SYSTEM ARCHITECT	SHEET NO: 01 OF 01

The Classic Standard: TF-IDF

$$Score = TF(term) * IDF(term)$$

A **Bonus**/Penalty System. **Bonus** for local repetition, **Penalty** for global commonality.

The Flaw (Visualized)



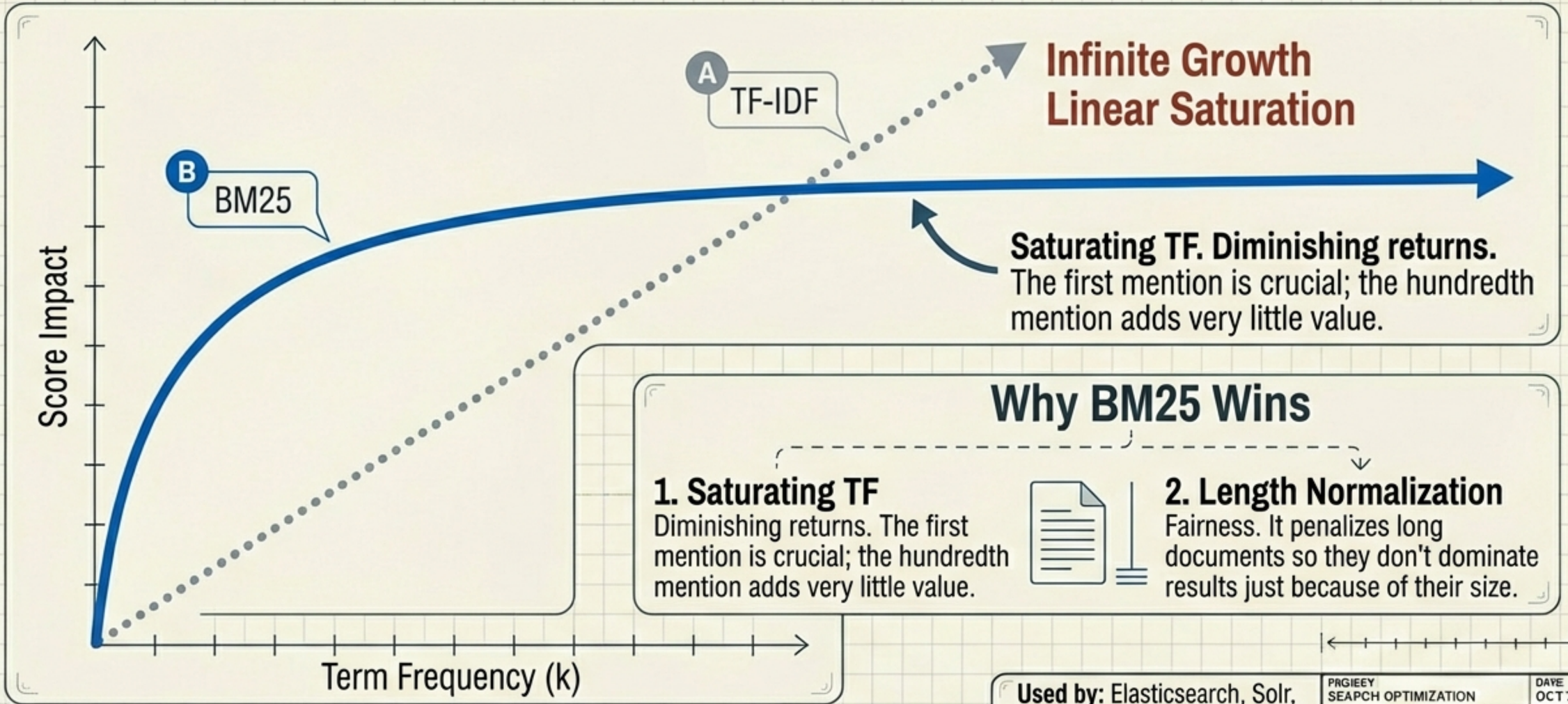
Problem 1: Linear Saturation. The 10th occurrence of a word counts as much as the 1st. In reality, relevance should plateau.

Problem 2: No Length Normalization. Longer documents unfairly get higher scores just by having more words.

PROJECT: SEARCH OPTIMIZATION	DATE: OCT 28, 2023
DRAWN BY: SYSTEM ARCHITECT	SHEET NO: 01 OF 01

The Modern Standard: BM25

Fixing TF-IDF with Saturation and Normalization.



Why BM25 Wins

1. Saturating TF
Diminishing returns. The first mention is crucial; the hundredth mention adds very little value.

2. Length Normalization
Fairness. It penalizes long documents so they don't dominate results just because of their size.

Used by: Elasticsearch, Solr, Lucene, and Modern MySQL.

PROJECT SEARCH OPTIMIZATION	DATE OCT 28, 2023
DRAWN BY SYSTEM ARCHITECT	SHEET NO 01 OF 01

Deconstructing the BM25 Formula

$$\text{Score} = \text{IDF} * \left(\frac{((\text{TF} * (k_1 + 1))}{(\text{TF} + k_1 * (1 - b + b * (|\text{D}| \ / \ \text{avgd}|)))} \right)$$

The Saturation Knob (k_1)

Controls how quickly the score plateaus.

Higher k_1 = Takes more repetitions to reach max score.

Typical value: 1.2 to 2.0.

The Length Knob (b)

Controls how much document length matters.

If $b = 1$: Full length penalty.
If $b = 0$: Length is ignored completely.

Typical value: 0.75.

BM25 isn't just a static formula; it is a configurable algorithm that balances frequency against document size.

PROJECT SEARCH OPTIMIZATION	DATE OCT 28, 2023
SPAWNED BY SYSTEM ARCHITECT	SHEET NO. 01 OF 01

Worked Example: The Math in Action

Query: “python programming”

Doc A
Python is a great programming language

Short, Perfect Match

Doc B
Java is also a programming language

Partial Match

IDF Calculation

`python`
Appears in 1 doc (Rare) → IDF = **High** (e.g., 0.41)

`programming`
Appears in 2 docs (Common) → IDF = **Low** (e.g., 0.1)

Scoring Logic

Total Score = Score(‘python’) + Score(‘programming’)

Doc A Calculation

High Score (for ‘python’) + **Low Score** (for ‘programming’) = **WINNER**

Contains ‘python’ (High IDF) and ‘programming’ (Low IDF). Score dominated by rare term.

Doc B Calculation

0 (No ‘python’) + **Low Score** (for ‘programming’) = **LOSER**

Missing ‘python’. Only scores for common term ‘programming’.

The Result

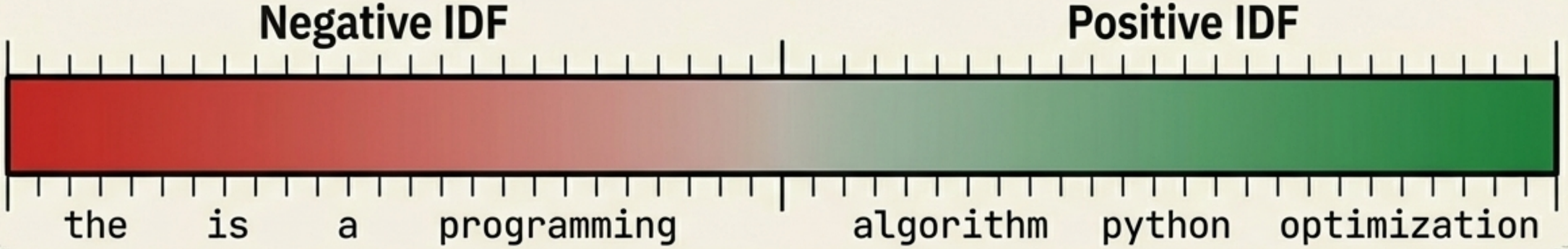
Conclusion
Doc A wins because it contains the rare term (‘python’), even though both contain ‘programming’.

Used by: Elasticsearch, Solr, Lucene, and Modern MySQL.

PAGIEEY SEAPCH OPTIMIZATION	DAVE OCT 28, 2023
SPAVNI ST SVSTEM ARCHITECT	NICETVCI 01 OF 01

The Mystery of Negative Scores

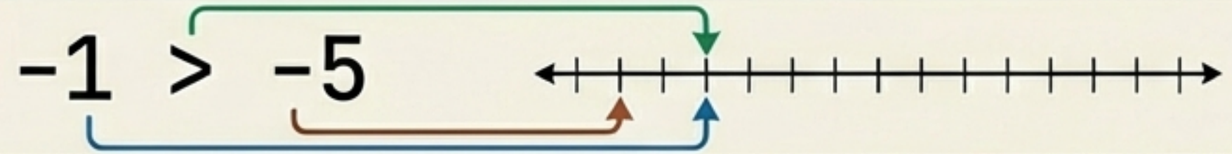
Why do some words generate negative values? And does it break the system?



IDF becomes negative when a term appears in more than 50% of the documents. These are treated as “Stop Words” or common noise.

Does a negative score matter?

Ranking is Relative.



A score of -1 is still mathematically better than a score of -5. The relative order of relevance is preserved, so the best match still rises to the top.

PAGIEEY SEAPCH OPTIMIZATION	DAVE OCT 28, 2023
SPAVNIST SVSTEM ARCHITECT	NICETVCI 01 OF 01

Implementation: MySQL FTS Modes

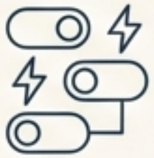
Natural Language

The default “Human” search.

- Interprets query as a natural phrase.
- Uses BM25/TF-IDF automatically.
- Best for: Standard search bars.

DEFAULT

Boolean Mode



Precise Control.

- Supports operators (+, -, >).
- Ignores the 50% threshold rule.
- Best for: Advanced filters and expert users.

POWER USER

Query Expansion

The “Widener”.

- Two-pass search: Finds results, then finds related words in those results to search again.
- Best for: Vague queries or “Did you mean...” features.

DISCOVERY

PROJECT: SEARCH OPTIMIZATION
ENGINEER: SYSTEN ARCHITECT

Precision Control: Boolean Operators

Overriding the algorithm in Boolean Mode.

Cheat Sheet

+	MUST contain	+MySQL
-	MUST NOT contain	-Oracle
>	BOOST relevance	>performance
<	DEMOTE relevance	<latency
" "	Exact Phrase	"inverted index"

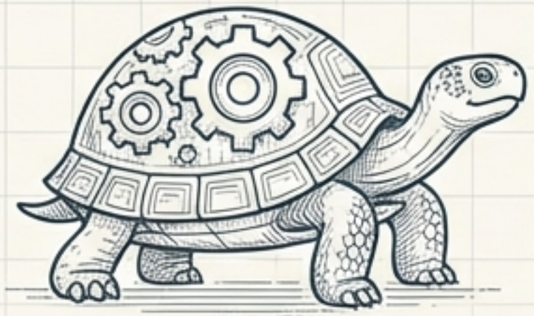
Note: The wildcard `*` is NOT supported in MySQL Boolean mode.

```
MATCH(content) AGAINST('+MySQL -Oracle >performance' IN BOOLEAN MODE)
```

PAGIEEY SEAPCH OPTIMIZATION	DAYE OCT 28, 2023
SPAVNI ST SVSTEM ARCHITECT	NIEETVCI 03 OF 03

Practical Application: The WordPress Ecosystem

Default Search



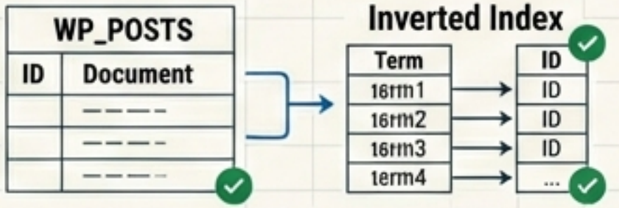
- Uses SQL `LIKE %...%`.
- Scans every post.
- Sorts by date, not relevance.

INEFFICIENT

FTS Upgrades

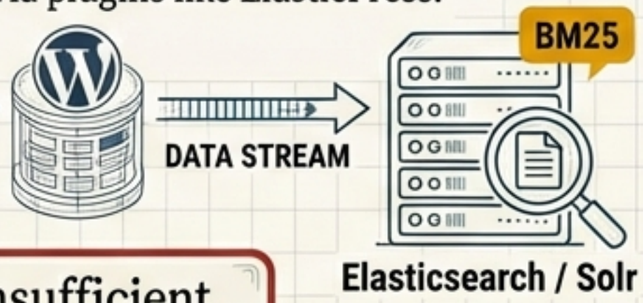
Native MySQL FullText

Enabled via plugins like Relevanssi.
Adds inverted index structure to WP tables.



External Engines

Offload to Elasticsearch/Solr (BM25)
via plugins like ElasticPress.



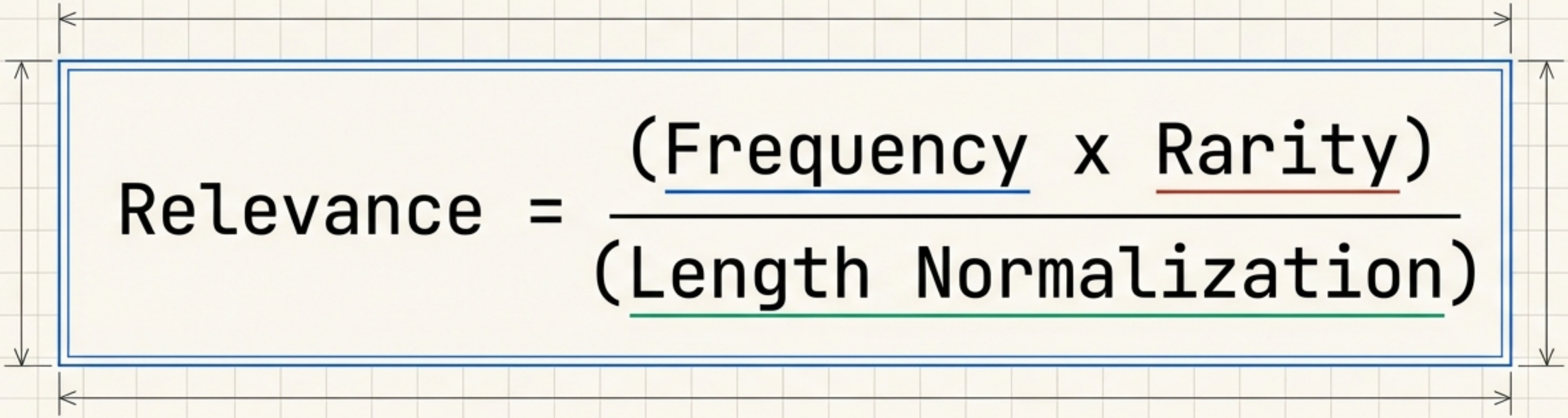
For serious content sites, the default database search is insufficient. Offloading to an Inverted Index is mandatory for scale.

Summary: Choosing the Right Engine

Your Scenario	Recommended Tool
Simple Prototype / Educational	TF-IDF / Natural Language Mode
Power User / Specific Filters	MySQL Boolean Mode
Vague User Intent / Discovery	Query Expansion
High Scale / Production Quality	BM25 (Elasticsearch / Solr)

Rule of Thumb: Start with Natural Language. Upgrade to Boolean for control. Upgrade to BM25/Elastic for scale and quality.

The Search Equation



Search is the balance between the mechanical speed of the Index and the mathematical intelligence of the Ranking. It is not just about finding data; it is about finding value.

PROJECT: SEARCH OPTIMIZATION	DATE: OCT 28, 2023
AUTHOR: SYSTEM ARCHITECT	SHEET NO: 08 OF 08