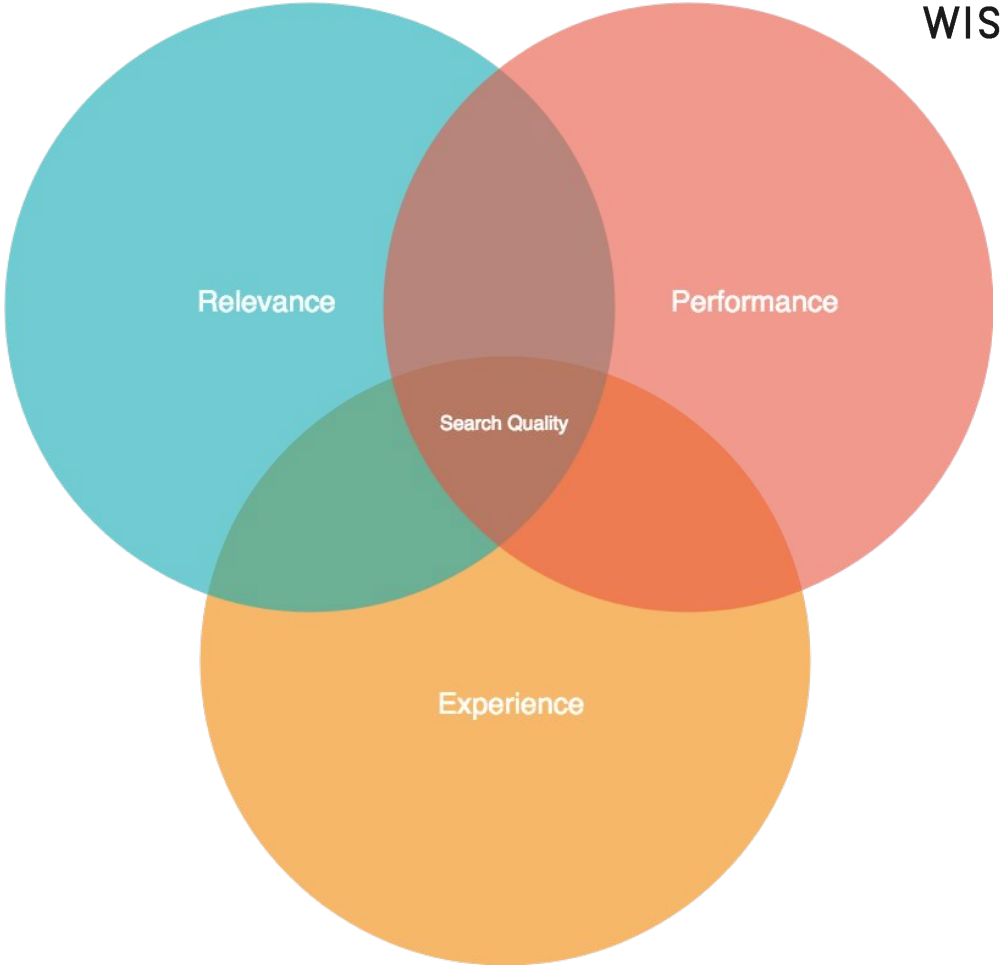
WISE ENGINEERING

Elasticsearch: Controlling relevance

Vitalii Melnychuk, Software Engineer

WISE ENGINEERING



WISE ENGINEERING

elasticsearch

Store,

Index,

Analize

Terminology

- Cluster
- Node
- Shards
- Replicas

- Index -> Table
- Document -> row
- Mapping -> schema
- Field -> column

What's next

- Document indexing
- Boolean model to find documents
- Lucene's Practical scoring

Index time
Vs
Query time

Index meanings

- Index (noun)

- Index (noun)
- **Index (verb)**

Index meanings

Index meanings

- Index (noun)
- Index (verb)
- **Inverted index**

- Index (noun)
- Index (verb)
- **Inverted index**

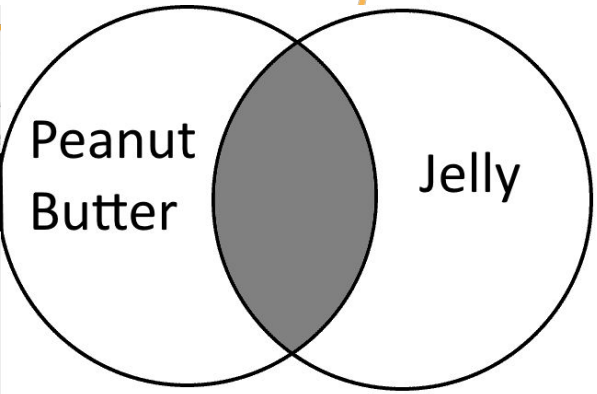
Term	Doc_1	Doc_2
Quick		X
The	X	
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	
summer		X
the	X	

- Index (noun)
- Index (verb)
- **Inverted index**

Term	Doc_1	Doc_2
Quick		X
The	X	
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	
summer		X
the	X	

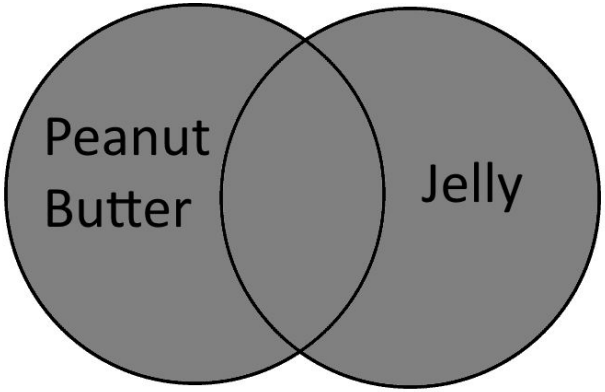
Moving on

- Document indexing
- **Boolean model to find documents**
- Lucene's Practical scoring



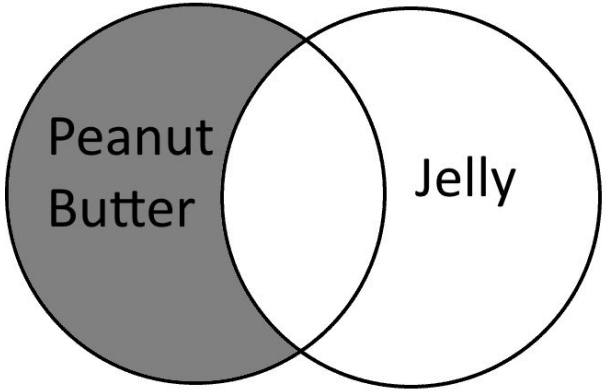
AND

Using AND, this search would only retrieve results with Peanut Butter and Jelly.



OR

Using OR, this search would retrieve results with peanut butter, with jelly, and with both.



NOT

Using NOT, this search would retrieve results with peanut butter, and exclude those with jelly or PB with jelly.

Moving on

- Document indexing
- Boolean model to find documents
- **Lucene's Practical scoring**

Relevance

- Term frequency

$$tf(t \text{ in } d) = \sqrt{frequency}$$

Relevance

- Term frequency
- **Inverse document frequency**

$$idf(t) = 1 + \log (numDocs / (docFreq + 1))$$

Relevance

- Term frequency
- Inverse document frequency
- **Field-length norm**

$$norm(d) = 1 / \sqrt{numTerms}$$

Lucene's Practical scoring

score (q, d) =

queryNorm(q)

$$queryNorm = 1 / \sqrt{sumOfSquaredWeights}$$

Lucene's Practical scoring

score (q, d) =

queryNorm(q)

coord(q,d)

- Document with fox → score: **1.5 * 1 / 3 = 0.5**
- Document with quick fox → score: **3.0 * 2 / 3 = 2.0**
- Document with quick brown fox → **score: 4.5 * 3 / 3 = 4.5**

Lucene's Practical scoring

score (q, d) =

queryNorm(q)

coord(q,d)

Σ (

tf(t in d)

idf(t)^2

norm(t, d)

) (t in q)

tf(t in d) = √frequency

idf(t) = 1 + log (numDocs / (docFreq + 1))

norm(d) = 1 / √numTerms

Lucene's Practical scoring

score (q, d) =

queryNorm(q)

coord(q,d)

∑ (

tf(t in d)

idf(t)^2

t.getBoost()

norm(t, d)

) (t in q)

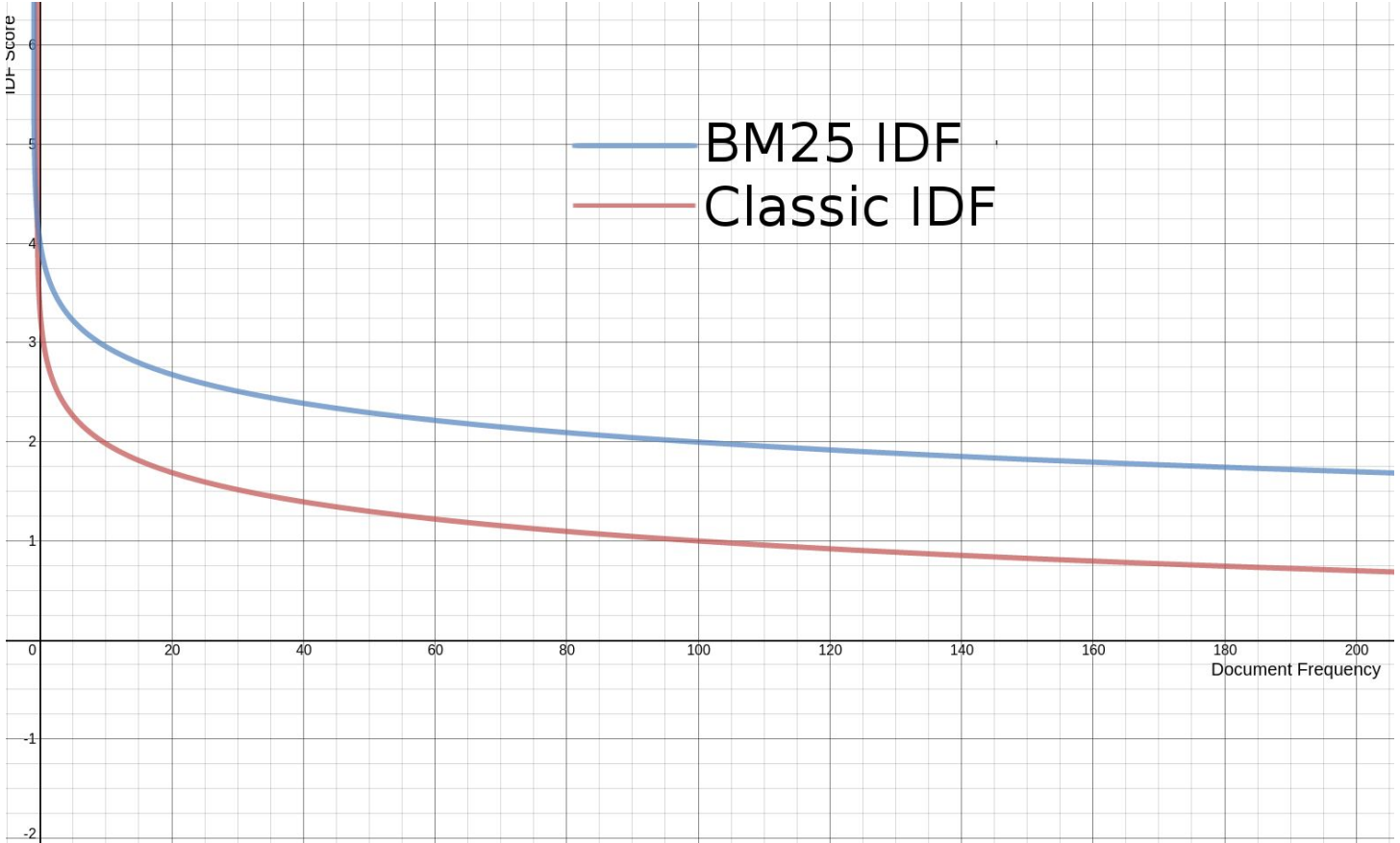
```
GET /docs_2014_*/_search
{
  "indices_boost": {
    "docs_2014_10": 3,
    "docs_2014_09": 2
  },
  "query": {
    "match": {
      "title": {
        "query": "quick brown fox",
            "boost": 2
      }
    }
  }
}
```

BM25 Algorithm

$$
\sum \Big(\text{tf}(t \text{ in } d) * \text{idf}(t) * (k+1) \Big/ \Big(\text{tf}(t \text{ in } d) + k * \big(1 - b + b * \text{termsCount} / \text{avgTermCount}\big) \Big) * \textbf{t.getBoost()} \Big) \ (t \text{ in } q)
$$

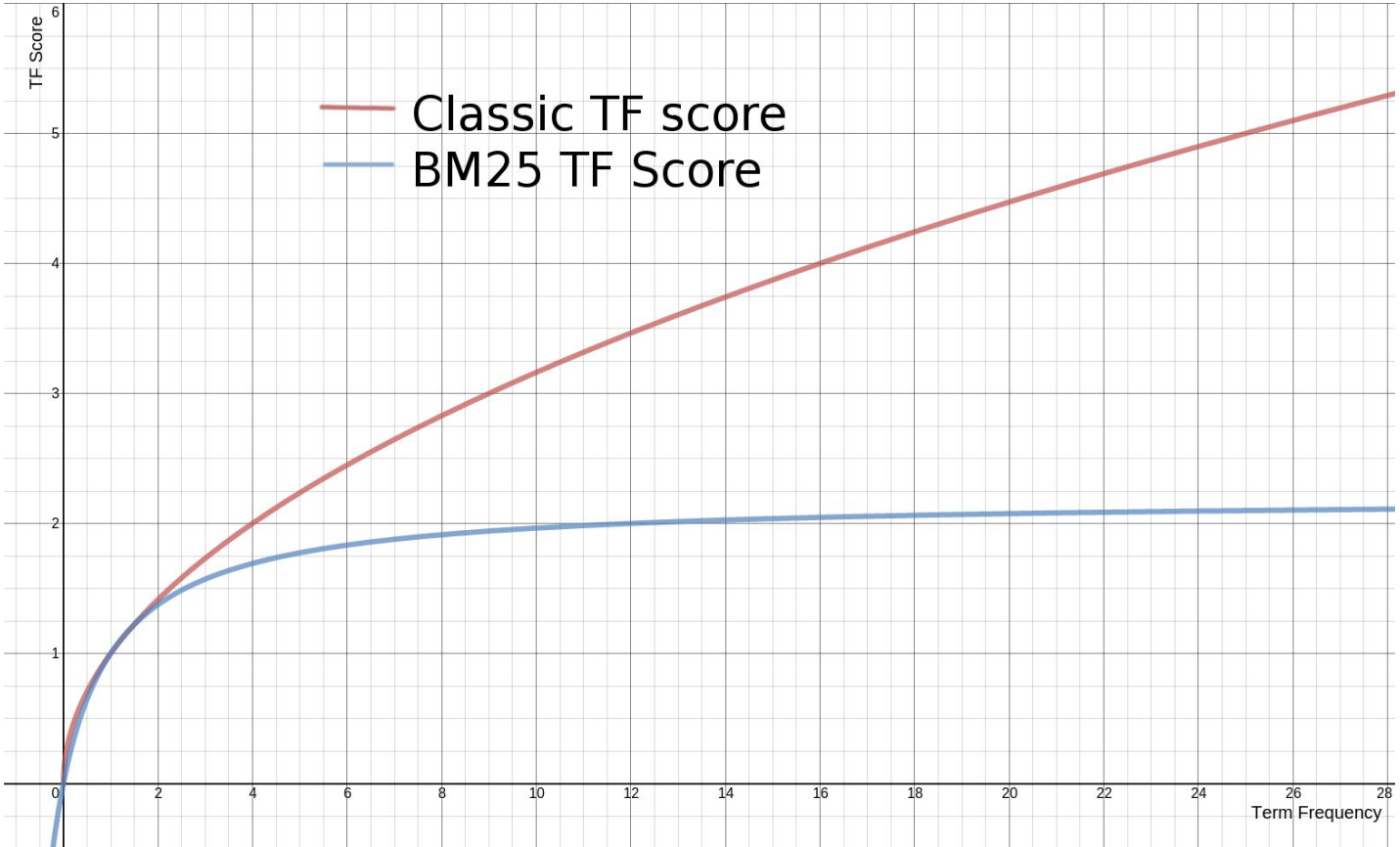
WISE ENGINEERING

BM25's take on IDF



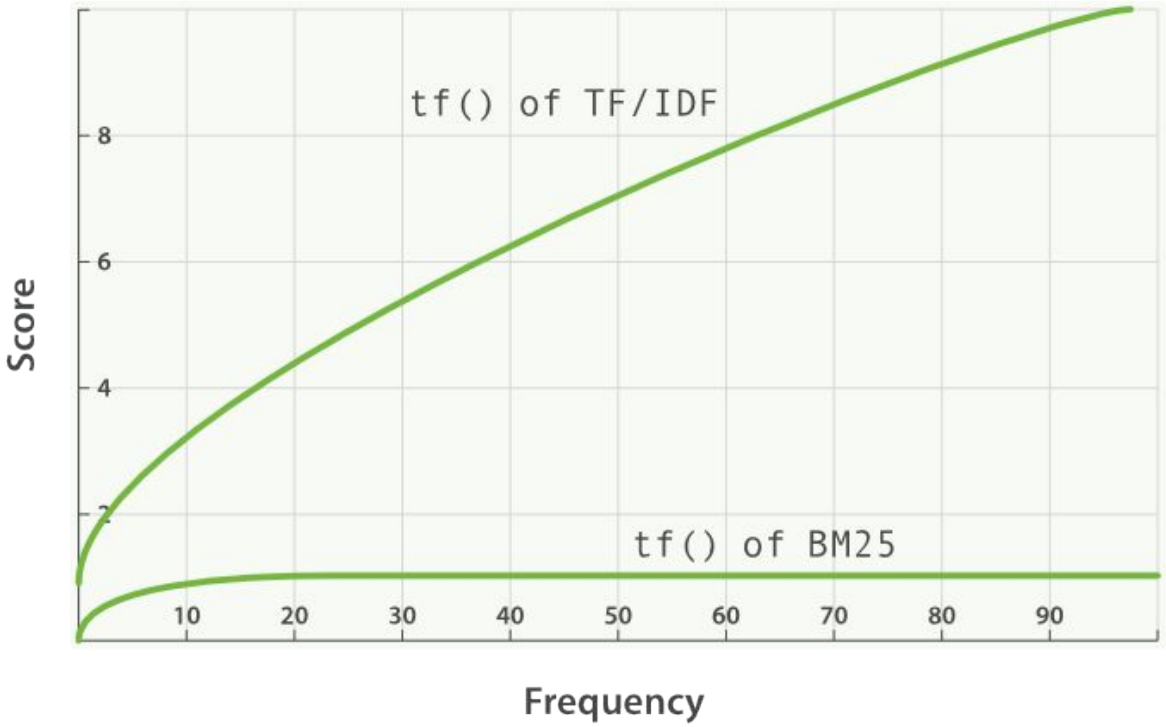
WISE ENGINEERING

BM25's take on TF



WISE ENGINEERING

TF/IDF and BM25



BM25 Algorithm

$$\sum \left(tf(t \text{ in } d) * idf(t) * (\mathbf{k} + 1) \right) / \left(tf(t \text{ in } d) + \mathbf{k} * (1 - \mathbf{b} + \mathbf{b} * termsCount / avgTermCount) \right) (t \text{ in } q)$$

b=0.75

k=1.2

work pretty well

Scoring with scripts

```
price = doc["price"].value;

discount = doc["margin"].value;

score = doc["_score"];


return (price * (1 - discount))+ 1 + score;
```

Relevance tuning is the last 10%

Thank you!