

Topic 4: The Probabilistic Model and BM25

COMP3009J: Information Retrieval

Dr. David Lillis (david.lillis@ucd.ie)

UCD School of Computer Science
Beijing Dublin International College

The Probabilistic Model

- Along with the Boolean and Vector Space models, the other “classic” model is the **Probabilistic Model**.
 - Also known as the “binary independence retrieval” model.
- Originally proposed by Robertson and Sparck Jones in 1976.
- Pages 79-86 of Modern Information Retrieval (2nd Edition)

Basics

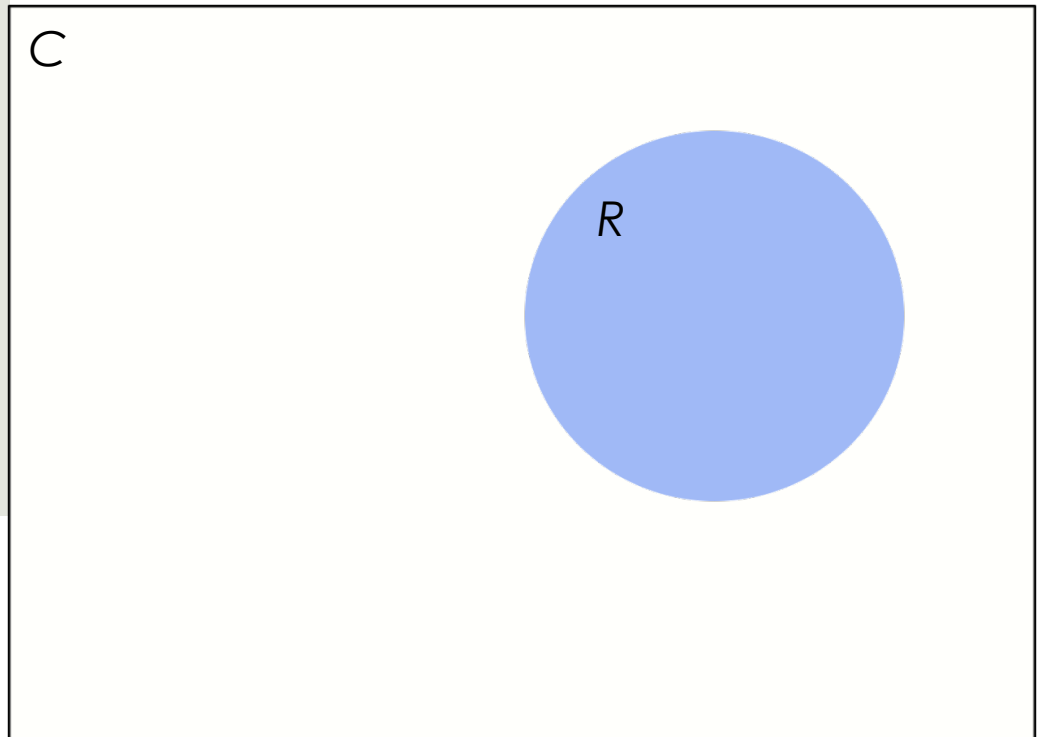
- The premise underlying the probabilistic model is that there exists a set of documents that contains all of the documents that are relevant to the user's information need and no others.
- This is referred to as the **ideal answer set**, or R .
- If we have a full description of this ideal answer set, we would have no problems retrieving its documents.
- We can think of the query process as the process of trying to specify what the properties of R actually are.

R : the ideal answer set

C : the **corpus** (i.e. the set of all documents in the index).

R : the **ideal answer set** (i.e. the set of documents that are relevant to the user's information need).

All documents outside R are not relevant. This can also be described as \bar{R}



Basics

- We know that there are index **terms** available that describe the documents in some way.
- These terms should be used to describe the properties of the ideal answer set.
- At the beginning of the retrieval process, we don't know what the properties of the ideal answer set are, so we must attempt an **initial guess** so as to return an **reasonable initial set** of documents to the user.
- The user can then interact with the system to help describe what the ideal answer set should be.
- This **user interaction** is a key difference between the Probabilistic Model and the others we have seen so far.

Basics

- The user looks at the documents that have been retrieved and marks which ones are **relevant**.
- **Remember** only the user knows what the information need is: a query is just an attempt to express this.
- The belief behind the Probabilistic Model is that the information need can be better defined by having users expressly state which documents help to satisfy their information need, rather than reducing it to a few keywords.

Probabilistic Model Process

1. User has an **information need**.



Probabilistic Model Process

2. User **sends a query** to the IR system.



Probabilistic Model Process

3. System sends back an
initial set of results.

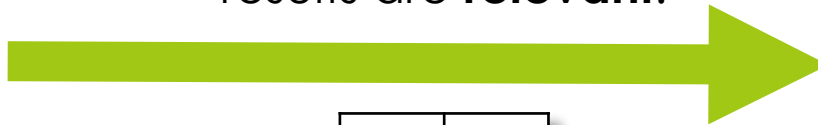


d5
d1
d45
d13
d93
d66
d12
d39
d11
d73



Probabilistic Model Process

4. User **tells** the system which of these results are **relevant**.



✓	d5
	d1
✓	d45
	d13
	d93
✓	d66
	d12
✓	d39
✓	d11
	d73



Probabilistic Model Process

5. System uses this information to **refine** the results
(i.e. improve the quality of the results)

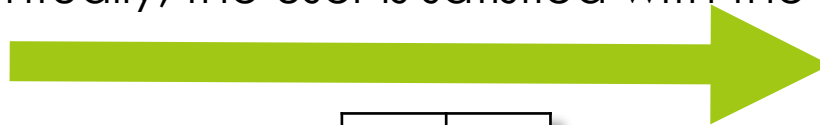


✓	d5
✓	d45
✓	d66
✓	d39
✓	d11
	d14
	d8
	d31
	d92
	d70



Probabilistic Model Process

5. This process may be **repeated** many times. Eventually, the user is satisfied with the results.



✓	d5
✓	d45
✓	d66
✓	d39
✓	d11
✓	d14
✓	d8
✓	d31
✓	d92
✓	d70



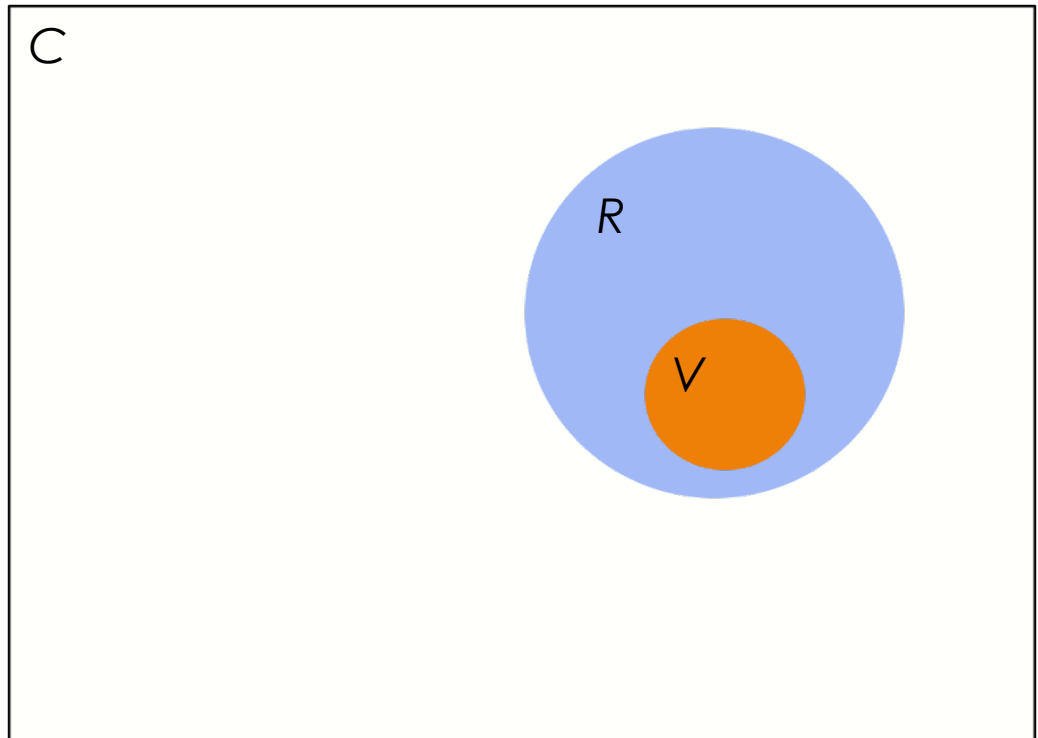
R : the ideal answer set

C : the **corpus** (i.e. the set of all documents in the index).

R : the **ideal answer set** (i.e. the set of documents that are relevant to the user's query).

V : the set of documents that the **user** has **verified** as being **relevant**.

V must be a subset of R (user will only mark documents relevant that are actually relevant).

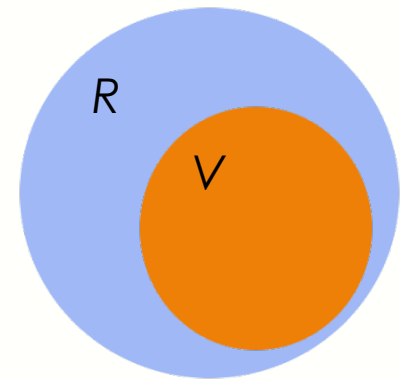


R : the ideal answer set

As more interaction occur,
the user marks more
documents as relevant:

V grows to become closer to R .

C



R : the ideal answer set

As more interaction occur,
the user marks more
documents as relevant:

V grows to become closer to R .

Ideally, V and R will be equal at
the end of the process.

C



$R = V$

Basics

- The system uses this user feedback to refine the description of the ideal answer set.
- By repeating this process many times, it is expected that the description will evolve and become closer to the real description of the ideal answer set.
- A conscious effort is made to use **probabilities** to create this description.

Assumptions

- As with the other IR models we have seen, we speak of query q and document d_j .
- The model tries to estimate the probability that document d_j is relevant.
- It is assumed that this relevance **depends only on the query and on the document representation**.
- It is also assumed that there is a **subset of all documents** that the user prefers as the answer for query q (this is the **ideal answer set R**).
- Documents in R are predicted to be **relevant** and those not in R are predicted to be **non-relevant**.

Notation

- q : the query supplied by the user
- d_j : a document in the index
- R : a set of documents containing all the relevant documents and no non-relevant ones (ideal answer set).
- \bar{R} : a set of documents containing all the non-relevant documents and no relevant ones.

Similarity Scores

- As with the other models we have seen, the Probabilistic Model calculates a **similarity score** that is used to rank the documents when presented to the user.
- In this model, this score is defined as follows:

$$\frac{P(d_j \text{ relevant-to-} q)}{P(d_j \text{ non-relevant-to-} q)}$$

- This gives us the **odds** of document d_j being relevant to a particular query q .
- As with the Vector Space Model for example, a similarity score must be calculated for **every document in the collection** and this is ultimately used to rank the documents.

Similarity Scores

- Mathematical analysis (including the use of Bayes Rule) allows us to break these probabilities down so as to deal with individual index terms: k_i (further details on p.81 of Modern Information Retrieval (2nd Edition)).

- This gives the following formula:

$$\text{sim}(d_j, q) \sim \sum_{i=1}^t w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\bar{R})}{P(k_i|\bar{R})} \right)$$

- To calculate this we need:

- t : Total number of terms in the collection (very easy)
- $w_{i,q}$: Weight of term k_i in query q (easy)
- $w_{i,j}$: Weight of term k_i in document d_j (easy)
- $P(k_i|R)$: Probability that a relevant document contains k_i (difficult)
- $P(k_i|\bar{R})$: Probability that a non-relevant document contains k_i (difficult)

Similarity Scores

- How do we calculate $P(k_i|R)$ (the probability that a relevant document contains the term k_i)?

- If we knew which documents were in R (i.e. all the relevant documents), it would be very simple:

$$P(k_i|R) = \frac{\text{number of relevant documents containing } k_i}{\text{number of relevant documents}}$$

- But if we know R then we don't need to do any retrieval at all!
 - We need to **estimate** this somehow.

Estimating Probabilities

- If a user has identified some relevant documents for us, we can use these to estimate the probabilities relating to all relevant documents.
- However, at the start, no retrieval has occurred so no feedback has been given.
- In this case, we must **guess reasonable values** for our first retrieval run, which will improve with user interaction later.
- We initially **assume** that all terms in the query have the **same probability** of being contained in **relevant documents**. A value typically used is 0.5, i.e.: $P(k_i|R) = 0.5$ (so we guess that half of the relevant documents will contain this term).

Estimating Probabilities

- The same problem applies to $P(k_i|\bar{R})$: the probability that k_i is contained in a non-relevant document.

- Again, if we knew \bar{R} , it would be as simple as:

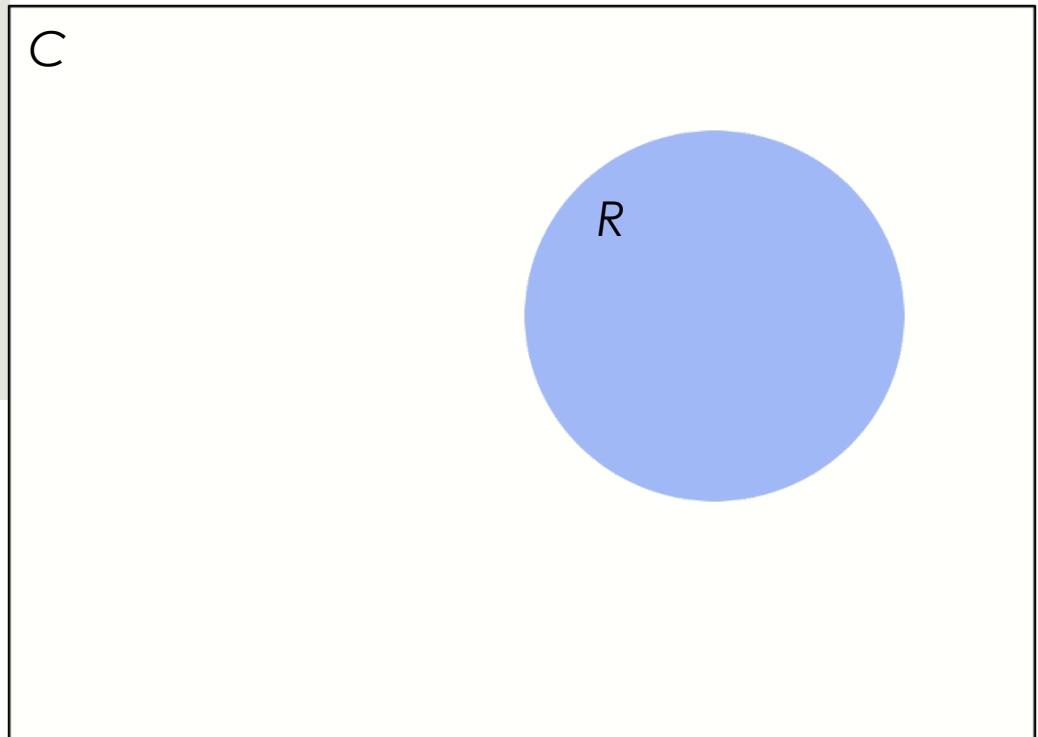
$$P(k_i|\bar{R}) = \frac{\text{number of non-relevant documents containing } k_i}{\text{number of non-relevant documents}}$$

- However, we don't know it, so we must find another value that is approximately equal.

Observation: This is not realistic!

To illustrate the concept of R , we used this diagram earlier.

In reality, R will be much smaller: there are far more non-relevant documents for most queries than there are relevant ones.

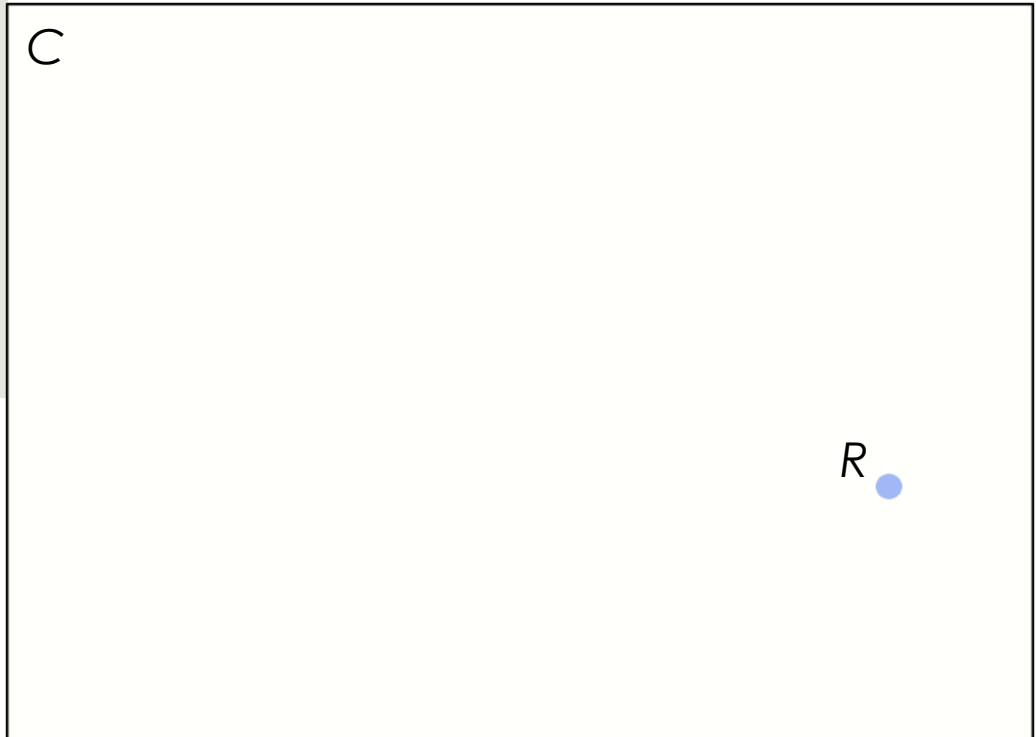


Observation: This more like it!

To illustrate the concept of R , we used this diagram earlier.

In reality, R will be much smaller: there are far more non-relevant documents for most queries than there are relevant ones.

Therefore, \bar{R} is very similar to C .



Estimating Probabilities

- This means that the set of non-relevant documents will be **very similar** to the set of **all** documents.
- If we assume initially that the set of non-relevant documents \bar{R} has the same characteristics as the set of all documents, we can express $P(k_i|\bar{R})$ as follows:

$$P(k_i|\bar{R}) = \frac{\text{number of documents containing } k_i}{\text{number of documents}}$$

Estimating Probabilities

- More formally: $P(k_i|\bar{R}) = \frac{n_i}{N}$
 - Here, n_i is the total number of documents that contain term k_i .
 - N is the total number of documents in the collection.
- Of course, these initial assumptions are not fully correct, but they do give reasonable results in practice, allowing us to improve our probability estimates afterwards, when the user(s) give feedback.

Estimating Probabilities

- Revisiting our formula:

$$\text{sim}(d_j, q) \sim \sum_{i=1}^t w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\bar{R})}{P(k_i|\bar{R})} \right)$$

- We still need to consider the term weights $w_{i,q}$ and $w_{i,j}$.
- Like the Boolean Model, the Probabilistic Model uses a simple binary weighting system: i.e.
 - $w_{i,j} = 1$ if document d_j contains term k_i .
 - $w_{i,j} = 0$ if document d_j does not contain term k_i .
- This means that we need only perform the calculation whenever a term is contained in both the query and the document, otherwise the value of the expression will be 0.

Improving Probabilities

- We can now calculate a similarity score for each document, rank the documents as normal and present the list to the user.
- The user can respond by indicating which documents are relevant to the query.
- We can then use this information to improve our estimates of probability and so improve the quality of the retrieval result.
- This is different to the other models we have seen, where one query is provided and results are returned on a one-off basis.
 - With these models, a user who wants to change the results must submit a new query.

Improving Probabilities

- To do the same thing more formally, remember that V is the set of documents the user has indicated are relevant, we can now change how we calculate our probabilities:
 - $P(k_i|R) = \frac{V_i}{V}$
 - $P(k_i|\bar{R}) = \frac{n_i - V_i}{N - V}$
- Here, V_i is the number of documents we **know** to be relevant that contain term k_i .
- If we run the query again with these probabilities, this results in a new set of documents being presented to the user, who can then indicate further relevant documents and repeat the process.
- The system should get closer to the ideal answer set each time, as V grows to become more similar to R .

Improving Probabilities

- In practice, the previous formula for calculating the probabilities causes difficulties for some small values of V and V_i (e.g. $V = 1$ and $V_i = 0$).
- To address this problem, an adjustment factor is often added, to give:

- $$P(k_i|R) = \frac{V_i + \frac{n_i}{N}}{V+1}$$

- $$P(k_i|\bar{R}) = \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1}$$

Automatic Probabilistic Retrieval

- In the above analysis, we have relied on the **user** of our system to provide **feedback** so that we know which documents are relevant.
- Most modern general-purpose IR systems tend to avoid this kind of user interaction, preferring to return just a single set of results in response to the query received (as with the Boolean and Vector Space models).
- To allow this behavior in the Probabilistic Model, we introduce one further **assumption** that allows us to create V : the set of documents we know to be relevant.
- After every run, we choose the top r documents and **assume** these are relevant. We then recalculate the probabilities based on these documents and run the retrieval again.

Automatic Probabilistic Retrieval

- This type of recursive retrieval allows the Probabilistic Model to improve its probabilities in practice, without involving a human to give feedback on what is relevant.
- Most modern implementations of the Probabilistic Model (and its variants) work in this way; repeated user interaction is not very popular. Users who cannot quickly find what they want generally prefer to change their query and try again.

Summary

■ Advantages:

- Documents are ranked in decreasing order of their probability of being relevant.
- Users may add feedback to the system in order to improve retrieval performance.

■ Disadvantages:

- We need to guess our initial probabilities.
- All term weights are binary, so term frequency is ignored.
- Terms are assumed to be independent (although as with the Vector Space Model, it is not clear whether this is actually a disadvantage in practice).

The BM25 Model

Okapi BM25

- The BM25 Model (“BM” stands for “Best Match”) came about from a series of experiments that were carried out to extend the classic probabilistic model.
- First implemented in the Okapi system that was created in London City University in the 1980s and 1990s.
- Many variants have been proposed: we will look only at BM25 itself.
- Today, BM25 is considered to be a state-of-the-art retrieval method that operates using the same principles as TF-IDF but generally performs better than the classic version we have already studied.
- Pages 104-107 of Modern Information Retrieval (2nd Edition)

BM25: Reasoning

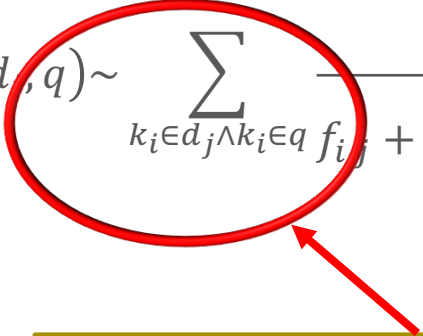
- Based on the belief that good term weighting comes from three principles:
 - Inverse document frequency: (terms that are rare across a collection should carry more weight).
 - Term frequency: (terms that are common within a document should carry more weight).
 - Document length normalisation: (so that longer documents do not get an unfair advantage if they contain query terms often simply because of their length).
- The formula evolved over time in response to many experiments being conducted.

BM25 Formula

$$sim_{BM25}(d_j, q) \sim \sum_{k_i \in d_j \wedge k_i \in q} \frac{f_{i,j} \times (1 + k)}{f_{i,j} + k \left((1 - b) + \frac{b \times len(d_j)}{avg_doclen} \right)} \times \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right)$$

- Again, we calculate a similarity score for each document d_j .
- The similarity score increases whenever a term k_i is in document d_j and also in the query.

BM25 Formula

$$sim_{BM25}(d, q) \sim \sum_{k_i \in d_j \wedge k_i \in q} \frac{f_{i,j} \times (1 + k)}{f_{i,j} + k \left((1 - b) + \frac{b \times len(d_j)}{avg_doclen} \right)} \times \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right)$$


More words in common with the query =>
Good!

- Again, we calculate a similarity score for each document d_j .
- The similarity score increases whenever a term k_i is in document d_j and also in the query.

BM25 Formula

$$\text{sim}_{BM25}(d_j, q) \sim \sum_{k_i \in d_j \wedge k_i \in q} \frac{f_{i,j} \times (1 + k)}{f_{i,j} + k \left((1 - b) + \frac{b \times \text{len}(d_j)}{\text{avg_doclen}} \right)} \times \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right)$$

Common words are less important (similar to IDF)

- N is the total number of documents in the collection.
- n_i is the total number of documents in the collection that contain term k_i .

BM25 Formula

$$\text{sim}_{BM25}(d_j, q) \sim \sum_{k_i \in d_j \wedge k_i \in q} \frac{f_{i,j} \times (1 + k)}{f_{i,j} + k \left((1 - b) + \frac{b \times \text{len}(d_j)}{\text{avg_doclen}} \right)} \times \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right)$$

Repetition of query words in the document =>
Good

- $f_{i,j}$ is the frequency of k_i in d_j (i.e. the number of times the term appears in the document)
- k and b are constants that can be set to suit the document collection and the desired behaviour. For general collections. $k=1$ and $b=0.75$ have been found to work well.

BM25 Formula

$$\text{sim}_{\text{BM25}}(d_j, q) \sim \sum_{k_i \in d_j \wedge k_i \in q} \frac{f_{i,j} \times (1 + k)}{f_{i,j} + k \left((1 - b) + \frac{b \times \text{len}(d_j)}{\text{avg_doclen}} \right)} \times \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right)$$

Repetitions are important, but are less important than different query words.

- $\text{len}(d_j)$ is the length of d_j (i.e. the number of terms in it)
- avg_doclen is the average length of a document in the collection.

BM25 Formula

$$\text{sim}_{\text{BM25}}(d_j, q) \sim \sum_{k_i \in d_j \wedge k_i \in q} \frac{f_{i,j} \times (1 + k)}{f_{i,j} + k \left((1 - b) + \frac{b \times \text{len}(d_j)}{\text{avg_doclen}} \right)} \times \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right)$$

Repetition is more important if the document is long (relative to the average document length).

- $\text{len}(d_j)$ is the length of d_j (i.e. the number of terms in it)
- avg_doclen is the average length of a document in the collection.

BM25 Example (calculation for one document)

- **Query:** "President Lincoln"
- $N = 500,000$ documents
- "president" occurs in 40,000 documents ($n_i = 40,000$)
- "lincoln" occurs in 300 documents ($n_i = 300$)
- "president" occurs 15 times in this document ($f_{i,j} = 15$)
- "lincoln" occurs 25 times in this document ($f_{i,j} = 25$)
- The document is 90% of the length of the average ($\frac{\text{len}(d_j)}{\text{avg_doclen}} = 0.9$)
- $k = 1, b = 0.75$

BM25 Variations

- There are a number of variations on BM25, of which two are particularly notable:
 - BM25F: Allows different fields to be given different importance in the document (e.g. document title, headlines, main text, etc.).
 - BM25+: addresses an issue with BM25 whereby very short documents would be given scores that are too high.

BM25: Usefulness

- Unlike probabilistic model, BM25 does not require relevance information.
- Most IR researchers agree that it outperforms the vector model on general collections.