The Probabilistic Model:
Calculating Probabilities

COMP3009J: Information Retrieval

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

UCD School of Computer Science
Beijing Dublin International College

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:

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

- 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:

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

- 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:

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

- 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:

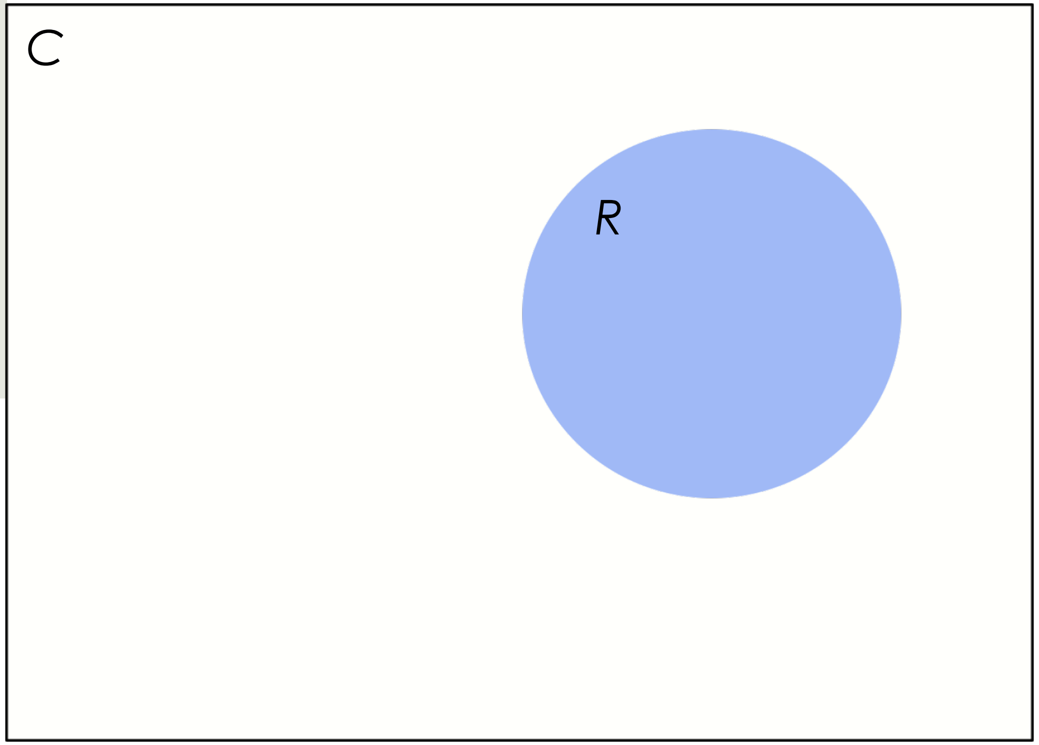
$$P(k_i|\bar{R}) = \frac{number\ of\ non{-}relevant\ documents\ containing\ k_i}{number\ of\ non{-}relevant\ documents}$$

- 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.

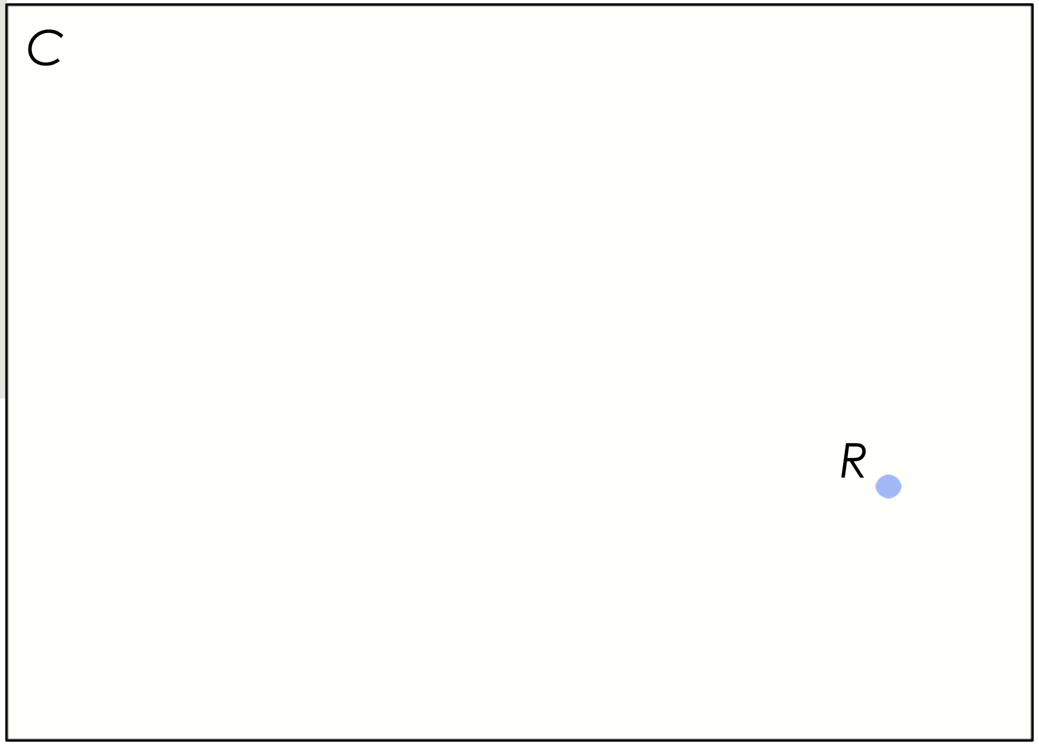


Observation: This more like it!

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:

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

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:

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

- 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).