Information Retrieval

BITS Pilani
Pilani Campus

Abhishek
February 2020

innovate achieve lead

BITS Pilani

Pilani Campus

CS F469, Information Retrieval

Lecture topics: Probabilistic information retrieval

innovate achieve lead

BITS Pilani

Pilani Campus

Most of these slides are based on:

https://web.stanford.edu/class/cs276/

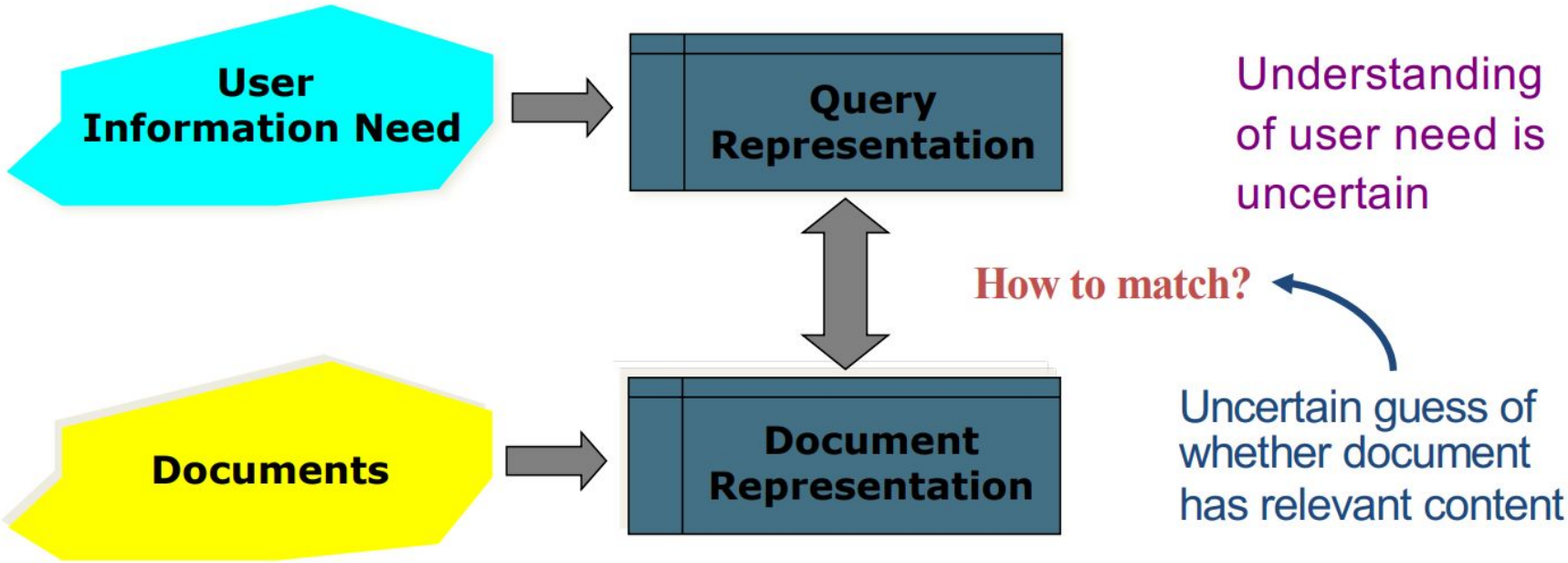
https://www.cis.uni-muenchen.de/~hs/teach/14s/ir/

This Lecture

- Probabilistic information retrieval

Why Probabilities in IR?

Why Probabilities in IR?



In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning.
Can we use probabilities to quantify our search uncertainties?

Probabilistic model vs. other models

Probabilistic model vs. other models

- Boolean model
  - Probabilistic models support ranking and thus are better than the simple Boolean model.

Probabilistic model vs. other models

- Boolean model
  - Probabilistic models support ranking and thus are better than the simple Boolean model.
- Vector space model
  - The vector space model is also a formally defined model that supports ranking.
  - Why would we want to look for an alternative to the vector space model?

Probabilistic vs. vector space model

Probabilistic vs. vector space model

- Vector space model: rank documents according to similarity to query.

Probabilistic vs. vector space model

- Vector space model: rank documents according to similarity to query.
- The notion of similarity does not translate directly into an assessment of “is the document a good document to give to the user or not?”

Probabilistic vs. vector space model

- Vector space model: rank documents according to similarity to query.
- The notion of similarity does not translate directly into an assessment of “is the document a good document to give to the user or not?”
- The most similar document can be highly relevant or completely non relevant.

Probabilistic vs. vector space model

- Vector space model: rank documents according to similarity to query.
- The notion of similarity does not translate directly into an assessment of “is the document a good document to give to the user or not?”
- The most similar document can be highly relevant or completely non relevant.
- Probability theory is arguably a cleaner formalization of what we really want an IR system to do: give relevant documents to the user.

The document ranking problem

The document ranking problem

- We have a collection of documents.
- User issues a query.
- A list of documents needs to be returned.
- Ranking method is the core of modern IR systems:
  - In what order do we present documents to the user?
  - We want the “best” document to be first, second best second, etc.

The document ranking problem

- We have a collection of documents.
- User issues a query.
- A list of documents needs to be returned.
- Ranking method is the core of modern IR systems:
  - In what order do we present documents to the user?
  - We want the “best” document to be first, second best second, etc.

- **Idea:** Rank by probability of relevance of the document w.r.t. information need.
  - **P(R=1 | document, query)**

The Probability Ranking Principle (PRP)

The Probability Ranking Principle (PRP)

- PRP in brief
  - If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable.

[1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

The Probability Ranking Principle (PRP)

- PRP in brief
  - If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable.
- PRP in full
  - If [the IR] system's response to each [query] is a ranking of the documents [...] in order of decreasing probability of relevance to the [query], where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.

[1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

Basic Probability Theory

Basic Probability Theory

For events **A** and **B**:

- Joint probability P(**A** ∩ **B**) (or P(**A, B**)) of both events occurring.
- Conditional probability P(**A** | **B**) of event **A** occurring given that event **B** has occurred.

Basic Probability Theory

For events **A** and **B**:

- Joint probability $P(\mathbf{A} \cap \mathbf{B})$ (or $P(\mathbf{A}, \mathbf{B})$) of both events occurring.
- Conditional probability $P(\mathbf{A} \mid \mathbf{B})$ of event **A** occurring given that event **B** has occurred.

- Chain rule gives fundamental relationship between joint and conditional probabilities:

$$P(\mathbf{A}, \mathbf{B}) = P(\mathbf{A} \cap \mathbf{B}) = P(\mathbf{A} \mid \mathbf{B})P(\mathbf{B}) = P(\mathbf{B} \mid \mathbf{A})P(\mathbf{A})$$

Basic Probability Theory

For events **A** and **B**:

- Joint probability $P(\mathbf{A} \cap \mathbf{B})$ (or $P(\mathbf{A}, \mathbf{B})$) of both events occurring.
- Conditional probability $P(\mathbf{A} \mid \mathbf{B})$ of event **A** occurring given that event **B** has occurred.

- Chain rule gives fundamental relationship between joint and conditional probabilities:

$$P(\mathbf{A}, \mathbf{B}) = P(\mathbf{A} \cap \mathbf{B}) = P(\mathbf{A} \mid \mathbf{B})P(\mathbf{B}) = P(\mathbf{B} \mid \mathbf{A})P(\mathbf{A})$$

- Partition rule: if **B** can be divided into an exhaustive set of disjoint subcases, then $P(\mathbf{B})$ is the sum of the probabilities of the subcases. A special case of this rule gives:

$$P(\mathbf{B}) = P(\mathbf{AB}) + P(\bar{\mathbf{A}}\mathbf{B})$$

Bayes Rule and Odds

Bayes Rule and Odds

Bayes' Rule for inverting conditional probabilities:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[\frac{P(B|A)}{\sum_{X\in\{A,\overline{A}\}} P(B|X)P(X)}\right] P(A)$$

Can be thought of as a way of updating probabilities:

- Start off with prior probability $P(A)$ (initial estimate of how likely event A is in the absence of any other information)
- Derive a posterior probability $P(A|B)$ after having seen the evidence B, based on the likelihood of B occurring in the two cases that A does or does not hold

Bayes Rule and Odds

Bayes' Rule for inverting conditional probabilities:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[\frac{P(B|A)}{\sum_{X\in\{A,\overline{A}\}} P(B|X)P(X)}\right] P(A)$$

Can be thought of as a way of updating probabilities:

- Start off with prior probability $P(A)$ (initial estimate of how likely event A is in the absence of any other information)
- Derive a posterior probability $P(A|B)$ after having seen the evidence B, based on the likelihood of B occurring in the two cases that A does or does not hold

Odds of an event provide a kind of multiplier for how probabilities change:

$$\text{Odds:} \qquad O(A) = \frac{P(A)}{P(\overline{A})} = \frac{P(A)}{1 - P(A)}$$

Binary Independence Model (BIM)

Binary Independence Model (BIM)

- Use simple assumptions that make estimating **P(R|d,q)** practical.

Binary Independence Model (BIM)

- Use simple assumptions that make estimating **P(R|d,q)** practical.
- 'Binary' (equivalent to Boolean): documents and queries represented as binary term incidence vectors
  - E.g., document d represented by vector $\vec{x} = (x_1, \ldots, x_M)$, where $x_t = 1$ if term t occurs in d and $x_t = 0$ otherwise
  - Different documents may have the same vector representation
- 'Independence': no association between terms (not true, but practically works - 'naive' assumption of Naive Bayes models)

Binary Independence Model (BIM)

- Use simple assumptions that make estimating **P(R|d,q)** practical.
- 'Binary' (equivalent to Boolean): documents and queries represented as binary term incidence vectors
  - E.g., document d represented by vector $\vec{x} = (x_1, \ldots, x_M)$, where $x_t = 1$ if term t occurs in d and $x_t = 0$ otherwise
  - Different documents may have the same vector representation
- 'Independence': no association between terms (not true, but practically works - 'naive' assumption of Naive Bayes models)

In BIM, the **P(R=1 | document, query)** or **P(R=1|d, q)** is modeled as

$$P(R = 1|\vec{x}, \vec{q})$$

Binary Independence Model (BIM) (Apply Bayes Rule)

$P(R|d,q)$ is modeled using term incidence vectors as $P(R|\vec{x},\vec{q})$

$$P(R=1|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=1,\vec{q})P(R=1|\vec{q})}{P(\vec{x}|\vec{q})}$$

$$P(R=0|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=0,\vec{q})P(R=0|\vec{q})}{P(\vec{x}|\vec{q})}$$

Binary Independence Model (BIM) (Apply Bayes Rule)

$P(R|d,q)$ is modeled using term incidence vectors as $P(R|\vec{x},\vec{q})$

$$P(R=1|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=1,\vec{q})P(R=1|\vec{q})}{P(\vec{x}|\vec{q})}$$

$$P(R=0|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=0,\vec{q})P(R=0|\vec{q})}{P(\vec{x}|\vec{q})}$$

- $P(\vec{x}|R=1,\vec{q})$ and $P(\vec{x}|R=0,\vec{q})$: probability that if a relevant or nonrelevant document is retrieved, then that document's representation is $\vec{x}$
- Use statistics about the document collection to estimate these probabilities

Binary Independence Model (BIM) (Apply Bayes Rule)

$P(R|d,q)$ is modeled using term incidence vectors as $P(R|\vec{x},\vec{q})$

$$P(R=1|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=1,\vec{q})P(R=1|\vec{q})}{P(\vec{x}|\vec{q})}$$

$$P(R=0|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=0,\vec{q})P(R=0|\vec{q})}{P(\vec{x}|\vec{q})}$$

- $P(R=1|\vec{q})$ and $P(R=0|\vec{q})$: prior probability of retrieving a relevant or nonrelevant document for a query $\vec{q}$
- Estimate $P(R=1|\vec{q})$ and $P(R=0|\vec{q})$ from percentage of relevant documents in the collection
- Since a document is either relevant or nonrelevant to a query, we must have that:

$$P(R=1|\vec{x},\vec{q}) + P(R=0|\vec{x},\vec{q}) = 1$$

Deriving a Ranking Function for Query Terms (Use Odds)

Deriving a Ranking Function for Query Terms (Use Odds)

- Given a query q, ranking documents by $P(R = 1|d, q)$ is modeled under BIM as ranking them by $P(R = 1|\vec{x}, \vec{q})$
- Easier: rank documents by their odds of relevance (gives same ranking)

$$O(R|\vec{x}, \vec{q}) = \frac{P(R = 1|\vec{x}, \vec{q})}{P(R = 0|\vec{x}, \vec{q})} = \frac{\frac{P(R=1|\vec{q})P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(R=0|\vec{q})P(\vec{x}|R=0,\vec{q})}{P(\vec{x}|\vec{q})}}$$

$$= \frac{P(R = 1|\vec{q})}{P(R = 0|\vec{q})} \cdot \frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})}$$

- $\frac{P(R=1|\vec{q})}{P(R=0|\vec{q})}$ is a constant for a given query - can be ignored

Deriving a Ranking Function for Query Terms (independence assumption)

Deriving a Ranking Function for Query Terms (independence assumption)

It is at this point that we make the Naive Bayes conditional independence assumption that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

Deriving a Ranking Function for Query Terms (independence assumption)

It is at this point that we make the Naive Bayes conditional independence assumption that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

$$\frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})} = \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

Deriving a Ranking Function for Query Terms (independence assumption)

It is at this point that we make the Naive Bayes conditional independence assumption that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

$$\frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})} = \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

So:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

Deriving a Ranking Function for Query Terms (split)

Deriving a Ranking Function for Query Terms (split)

Since each x_t is either 0 or 1, we can separate the terms:

Deriving a Ranking Function for Query Terms (split)

Since each x_t is either 0 or 1, we can separate the terms:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t=1|R=1,\vec{q})}{P(x_t=1|R=0,\vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1,\vec{q})}{P(x_t=0|R=0,\vec{q})}$$

Deriving a Ranking Function for Query Terms (4)

- Let $p_t = P(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in relevant document
- Let $u_t = P(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document

Deriving a Ranking Function for Query Terms (4)

- Let $p_t = P(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in relevant document
- Let $u_t = P(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document
- Can be displayed as contingency table:

	document	relevant $(R = 1)$	nonrelevant $(R = 0)$
Term present	$x_t = 1$	p_t	u_t
Term absent	$x_t = 0$	$1 - p_t$	$1 - u_t$

Deriving a Ranking Function for Query Terms (5)

Additional simplifying assumption: terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents

- If $q_t = 0$, then $p_t = u_t$

Deriving a Ranking Function for Query Terms (5)

Additional simplifying assumption: terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents

- If $q_t = 0$, then $p_t = u_t$

Now we need only to consider terms in the products that appear in the query:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1} \frac{1-p_t}{1-u_t}$$

- The left product is over query terms found in the document and the right product is over query terms not found in the document

Deriving a Ranking Function for Query Terms (5)

Including the query terms found in the document into the right product, but simultaneously dividing by them in the left product, gives:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}$$

Deriving a Ranking Function for Query Terms (5)

Including the query terms found in the document into the right product, but simultaneously dividing by them in the left product, gives:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}$$

- The left product is still over query terms found in the document, but the right product is now over all query terms, hence constant for a particular query and can be ignored.
- $\rightarrow$ The only quantity that needs to be estimated to rank documents w.r.t a query is the left product
- Hence the Retrieval Status Value (RSV) in this model:

$$RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

Deriving a Ranking Function for Query Terms (5)

Equivalent: rank documents using the log odds ratios for the terms in the query c_t:

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} - \log \frac{u_t}{1-u_t}$$

Deriving a Ranking Function for Query Terms (5)

Equivalent: rank documents using the log odds ratios for the terms in the query c_t:

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} - \log \frac{u_t}{1-u_t}$$

- The odds ratio is the ratio of two odds: (i) the odds of the term appearing if the document is relevant ($p_t/(1-p_t)$), and (ii) the odds of the term appearing if the document is nonrelevant ($u_t/(1-u_t)$)
- $c_t = 0$: term has equal odds of appearing in relevant and nonrelevant docs
- c_t positive: higher odds to appear in relevant documents
- c_t negative: higher odds to appear in nonrelevant documents

Term weight c_t in BIM

- $c_t = \log \frac{p_t}{(1-p_t)} - \log \frac{u_t}{1-u_t}$ functions as a term weight.
- Retrieval status value for document d: $RSV_d = \sum_{x_t=q_t=1} c_t$.
- So BIM and vector space model are identical on an operational level ...
- ...except that the term weights are different.
- In particular: we can use the same data structures (inverted index etc) for the two models.

History and summary of assumptions

- Among the oldest formal models in IR
  - Maron & Kuhns, 1960: Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities.

- Assumptions for getting reasonable approximations of the needed probabilities (in the BIM):
  - Boolean representation of documents/queries/relevance
  - Term independence
  - Out-of-query terms do not affect retrieval
  - Document relevance values are independent

Summary : BIM

P(R = 1 | document, query)

$$P(R = 1|\vec{x}, \vec{q})$$

$$O(R|\vec{x}, \vec{q}) \propto \prod_{t=1}^{M} \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})}$$

$$O(R|\vec{x}, \vec{q}) \propto \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1} \frac{1 - p_t}{1 - u_t}$$

Summary : BIM

$$O(R|\vec{x},\vec{q}) \propto \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1} \frac{1-p_t}{1-u_t}$$

$$O(R|\vec{x},\vec{q}) \propto \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

$$\mathbf{RSV_d} = \sum_{t:x_t=q_t=1} c_t$$

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} - \log \frac{u_t}{1-u_t}$$

Rank documents based on the RSV_d values.

How to compute RSV_d values and the experimental setup

Experimental Setup

- **Retrospective**
  - We have a corpus, queries and relevance judgement already marked by experts.

Experimental Setup

- **Retrospective**
  - We have a corpus, queries and relevance judgement already marked by experts.
- **Predictive**
  - We have two disjoints sets of documents. In one set, we have data about the document relevance with given queries.
  - We want to estimate the relevance of document in the other set for the same queries.

Retrospective Estimation

- **Benchmarking:**
  - We rank documents based on the RSV_d scores.
  - We compare the results with other scoring functions.
  - Compare results.

Reference:
http://www.staff.city.ac.uk/~sb317/papers/RSJ76.pdf

Retrospectively Estimation

- **Benchmarking:**
  - We rank documents based on the RSV_d scores.
  - We compare the results with other scoring functions.
  - Compare results.

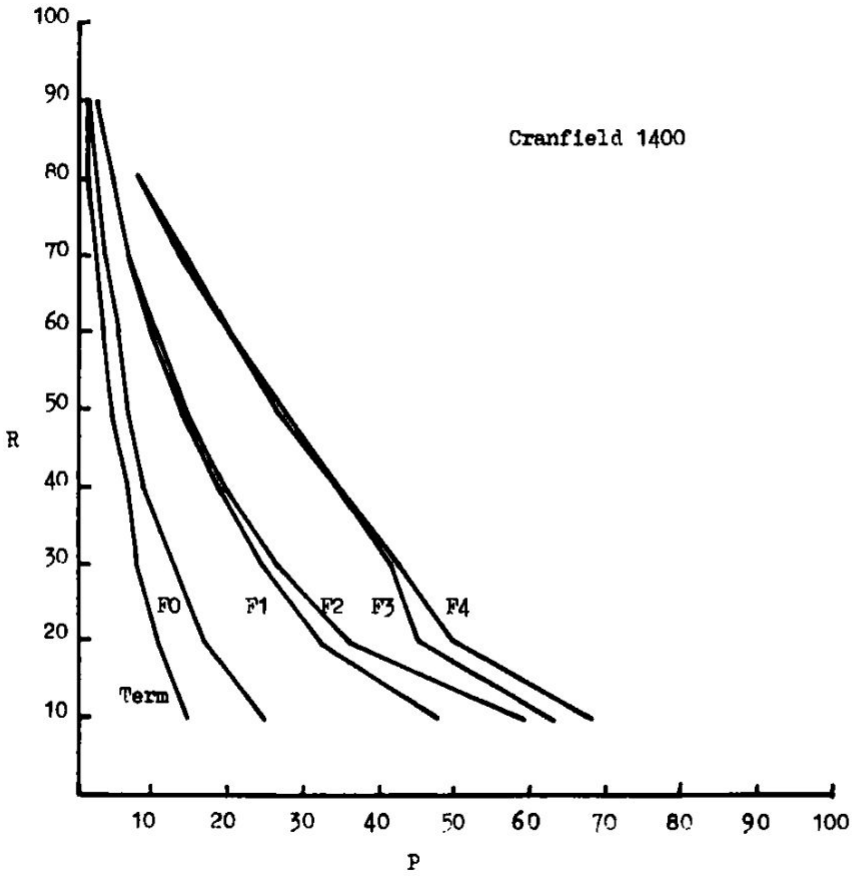


Fig. 2. Comparative Retrieval Performance Using Weights for Cranfield 1400 Collection

Reference:
http://www.staff.city.ac.uk/~sb317/papers/RSJ76.pdf

Predictive Estimation

- **Benchmarking:**
  - Compute p_t and u_t values from the corpus for which relevance is known.
  - Use the p_t and u_t values to compute RSV_d values for the other part of the corpus.
  - Compare results.

Reference:
http://www.staff.city.ac.uk/~sb317/papers/RSJ76.pdf

Predictive Estimation

- **Benchmarking:**
  - Compute p_t and u_t values from the corpus for which relevance is known.
  - Use the p_t and u_t values to compute RSV_d values for the other part of the corpus.
  - Compare results.

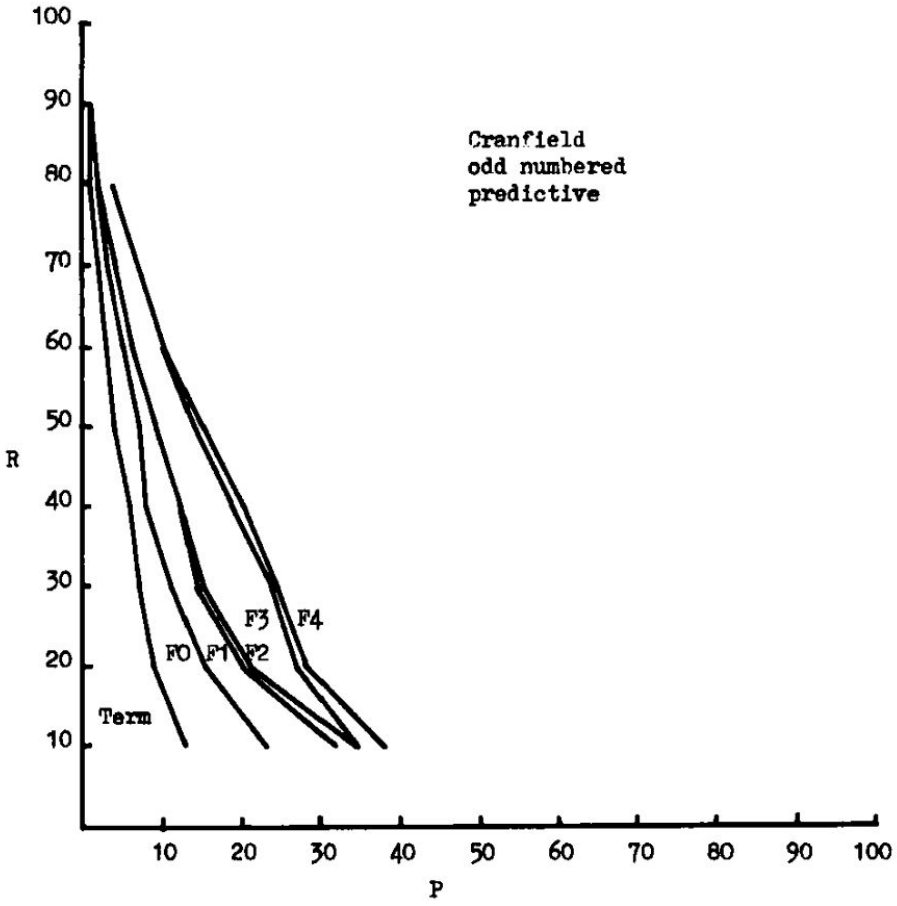


Fig. 3. Comparative Retrieval Performance Using Weights Predictively for Cranfield 700 Odd-Numbered Sub-collection

Reference:
http://www.staff.city.ac.uk/~sb317/papers/RSJ76.pdf

How to compute probability estimates

How to compute probability estimates

- Using a corpus (or its sample), where queries and relevance judgement already marked by experts.

How to compute probability estimates

- Using a corpus (or its sample), where queries and relevance judgement already marked by experts.
- We can define the followings for a given query:
  - The number of relevant documents = **S**.
  - The number of relevant documents where the term $\mathbf{x_t}$ is present = **s**
  - Number of documents in which term $\mathbf{x_t}$ is present = $\mathbf{df_t}$

How to compute probability estimates

- Using a corpus (or its sample), where queries and relevance judgement already marked by experts.
- We can define the followings for a given query:
  - The number of relevant documents = **S**.
  - The number of relevant documents where the term $\mathbf{x_t}$ is present = **s**
  - Number of documents in which term $\mathbf{x_t}$ is present = $\mathbf{df_t}$
- Total number of documents = **N**
- Using the above we need to estimate: $\mathbf{p_t}$ and $\mathbf{u_t}$

$$p_t = P(x_t = 1 | R = 1, \vec{q})$$

$$u_t = P(x_t = 1 | R = 0, \vec{q})$$

How to compute probability estimates (Contingency table)

How to compute probability estimates (Contingency table)

For each term t in a query, estimate c_t in the whole collection using a contingency table of counts of documents in the collection, where df_t is the number of documents that contain term t:

	documents	relevant	nonrelevant	Total
Term present	$x_t = 1$	s	$\mathrm{df}_t - s$	df_t
Term absent	$x_t = 0$	$S - s$	$(N - \mathrm{df}_t) - (S - s)$	$N - \mathrm{df}_t$
	Total	S	$N - S$	N

How to compute probability estimates (Contingency table)

For each term t in a query, estimate c_t in the whole collection using a contingency table of counts of documents in the collection, where df_t is the number of documents that contain term t:

	documents	relevant	nonrelevant	Total
Term present	$x_t = 1$	s	$\mathrm{df}_t - s$	df_t
Term absent	$x_t = 0$	$S - s$	$(N - \mathrm{df}_t) - (S - s)$	$N - \mathrm{df}_t$
	Total	S	$N - S$	N

$$p_t = s/S$$

$$u_t = (\mathrm{df}_t - s)/(N - S)$$

$$c_t = K(N, \mathrm{df}_t, S, s) = \log \frac{s/(S-s)}{(\mathrm{df}_t - s)/((N - \mathrm{df}_t) - (S - s))}$$

How to compute probability estimates (Contingency table)

For each term t in a query, estimate c_t in the whole collection using a contingency table of counts of documents in the collection, where df_t is the number of documents that contain term t:

	documents	relevant	nonrelevant	Total
Term present	$x_t = 1$	s	$\mathrm{df}_t - s$	df_t
Term absent	$x_t = 0$	$S - s$	$(N - \mathrm{df}_t) - (S - s)$	$N - \mathrm{df}_t$
	Total	S	$N - S$	N

Maximum Likelihood Estimate (MLE)

$$p_t = s/S$$

$$u_t = (\mathrm{df}_t - s)/(N - S)$$

$$c_t = K(N, \mathrm{df}_t, S, s) = \log \frac{s/(S-s)}{(\mathrm{df}_t - s)/((N - \mathrm{df}_t) - (S - s))}$$

Avoiding Zeros

Avoiding Zeros

- If any of the counts is a zero, then the term weight is not well-defined.

Avoiding Zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.

Avoiding Zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
- To avoid zeros: add 0.5 to each count (expected likelihood estimation = ELE)

Avoiding Zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
- To avoid zeros: add 0.5 to each count (expected likelihood estimation = ELE)
- For example, use S − s + 0.5 in formula for S − s

Exercise

Exercise

- Query:
  - Obama health plan

- Doc1: Obama rejects allegations about his own bad health
- Doc2: The plan is to visit Obama
- Doc3: Obama raises concerns with US health plan reforms

- Estimate the probability that the above documents are relevant to the query. Use a contingency table. These are the only three documents in the collection.

Simplifying assumption

Simplifying assumption

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for non relevant documents by statistics from the whole collection

Simplifying assumption

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for non relevant documents by statistics from the whole collection
- Hence, u_t (the probability of term occurrence in non relevant documents for a query) is df_t/N and

$$\log[(1 - u_t)/u_t] = \log[(N - df_t)/df_t] \approx \log N/df_t$$

- This should look familiar to you . . .

Simplifying assumption

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for non relevant documents by statistics from the whole collection
- Hence, u_t (the probability of term occurrence in non relevant documents for a query) is df_t/N and

$$\log[(1 - u_t)/u_t] = \log[(N - df_t)/df_t] \approx \log N/df_t$$

- This should look familiar to you . . .
- The above approximation cannot easily be extended to relevant documents.

How different are vector space and BIM?

How different are vector space and BIM?

- They are not that different.
- In either case you build an information retrieval scheme in the exact same way.

How different are vector space and BIM?

- They are not that different.
- In either case you build an information retrieval scheme in the exact same way.
- For probabilistic IR, at the end, you score queries not by cosine similarity and tf-idf in a vector space, but by a slightly different formula motivated by probability theory.
- **Next:** how to add term frequency and length normalization to the probabilistic model.

Okapi BM25: Overview

- **Okapi BM25** is a probabilistic model that incorporates term frequency (i.e., it's nonbinary) and length normalization.

Okapi BM25: Overview

- **Okapi BM25** is a probabilistic model that incorporates term frequency (i.e., it's nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts.

Okapi BM25: Overview

- **Okapi BM25** is a probabilistic model that incorporates term frequency (i.e., it's nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts.
- For modern full-text search collections, a model should pay attention to term frequency and document length

Okapi BM25: Overview

- **Okapi BM25** is a probabilistic model that incorporates term frequency (i.e., it's nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts.
- For modern full-text search collections, a model should pay attention to term frequency and document length
- BestMatch25 (a.k.a BM25 or Okapi) is sensitive to these quantities

Okapi BM25: Overview

- **Okapi BM25** is a probabilistic model that incorporates term frequency (i.e., it's nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts.
- For modern full-text search collections, a model should pay attention to term frequency and document length
- BestMatch25 (a.k.a BM25 or Okapi) is sensitive to these quantities
- BM25 is one of the most widely used and robust retrieval models

Use of BM25 in Wikipedia Search

- Recommended Reading:
  a. https://wikimedia-research.github.io/Discovery-Search-Test-BM25/
  b. https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/

- Tools with built in implementation:
  - Apache Lucene
  - Elastic search

Okapi BM25: Starting point

Okapi BM25: Starting point

- The simplest score for document **d** is just idf weighting of the query terms present in the document:

$$RSV_d = \sum_{t \in q} \log \frac{N}{\mathrm{df}_t}$$

Okapi BM25 basic weighting

Okapi BM25 basic weighting

- Improve idf term [log N/df] by factoring in term frequency and document length.

Okapi BM25 basic weighting

- Improve idf term [log N/df] by factoring in term frequency and document length.

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1 - b) + b \times (L_d / L_{\mathrm{ave}})) + \mathrm{tf}_{td}}$$

- $\mathbf{tf_{td}}$: term frequency in document d.
- $\mathbf{L_d}$ ($\mathbf{L_{ave}}$): length of document d (average document length in the whole collection).
- $\mathbf{k_1}$: tuning parameter controlling the document term frequency scaling.
- **b**: tuning parameter controlling the scaling by document length.

BM25's Take on TF

BM25's Take on TF

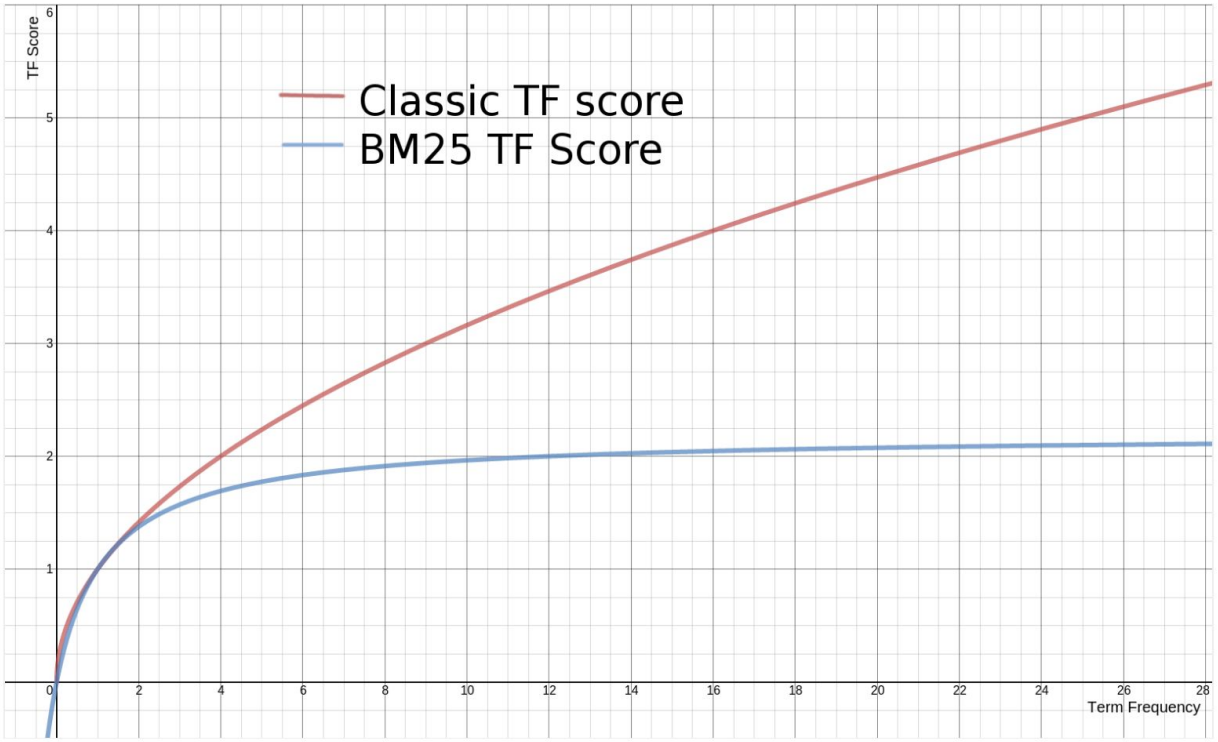
- Without any consideration for document length, term frequency follows the formula:

$$((k_1 + 1) * tf_{td}) / (k_1 + tf_{td})$$

BM25's Take on TF

- Without any consideration for document length, term frequency follows the formula:

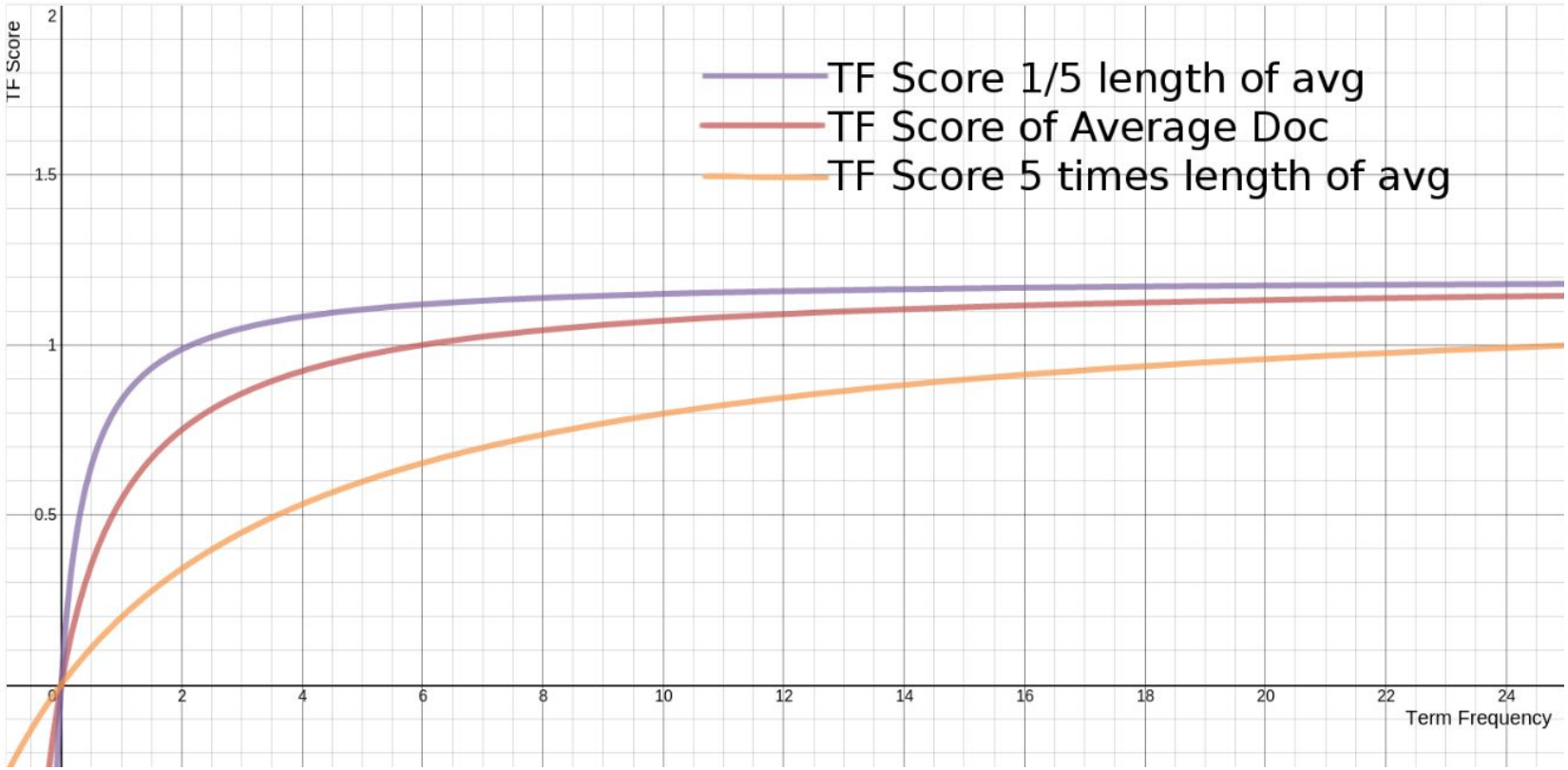
$$((k_1 + 1) * tf_{td}) / (k_1 + tf_{td})$$



Here here k_1=1.2

How Does BM25 Use Document Length?

How Does BM25 Use Document Length?



- Shorter docs hit the asymptote much faster.
- The more matches in these short docs, the more certain you can feel confident in the relevance.

Exercise

- Interpret BM25 weighting formula for $k_1 = 0$
- Interpret BM25 weighting formula for $k_1 = 1$ and $b = 0$
- Interpret BM25 weighting formula for $k_1 \rightarrow \infty$ and $b = 0$
- Interpret BM25 weighting formula for $k_1 \rightarrow \infty$ and $b = 1$

Okapi BM25 weighting for long queries

Okapi BM25 weighting for long queries

- For long queries, use similar weighting for query terms.

$$RSV_d = \sum_{t \in q} \left[\log \frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1 - b) + b \times (L_d / L_{\mathrm{ave}})) + \mathrm{tf}_{td}} \cdot \frac{(k_3 + 1)\mathrm{tf}_{tq}}{k_3 + \mathrm{tf}_{tq}}$$

- $\mathbf{tf_{tq}}$: term frequency in the query q.
- $\mathbf{k_3}$: tuning parameter controlling term frequency scaling of the query.
- No length normalization of queries (because retrieval is being done with respect to a single fixed query).
- The above tuning parameters should ideally be set to optimize performance on a development test collection. In the absence of such optimization, experiments have shown reasonable values are to set k_1 and k_3 to a value between 1.2 and 2 and b = 0.75.

Which ranking model should I use?

Which ranking model should I use?

- I want something basic and simple → use vector space with tf-idf weighting.

Which ranking model should I use?

- I want something basic and simple → use vector space with tf-idf weighting.
- I want to use a state-of-the-art ranking model with excellent performance → use language models or BM25 with tuned parameters

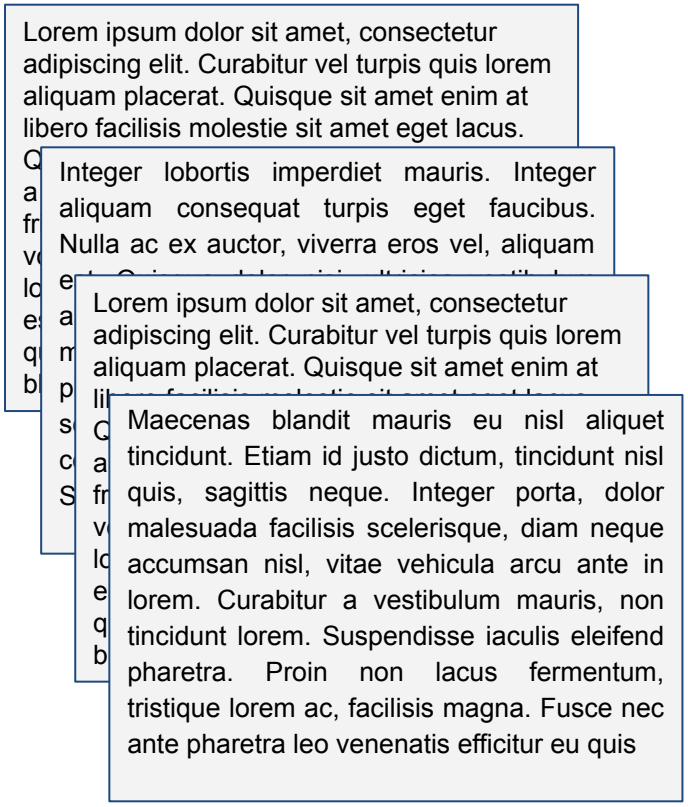
Which ranking model should I use?

- I want something basic and simple → use vector space with tf-idf weighting.
- I want to use a state-of-the-art ranking model with excellent performance → use language models or BM25 with tuned parameters
- In between: BM25 or language models with no or just one tuned parameter.

Text Classification

Text Classification Problem

Input



Output

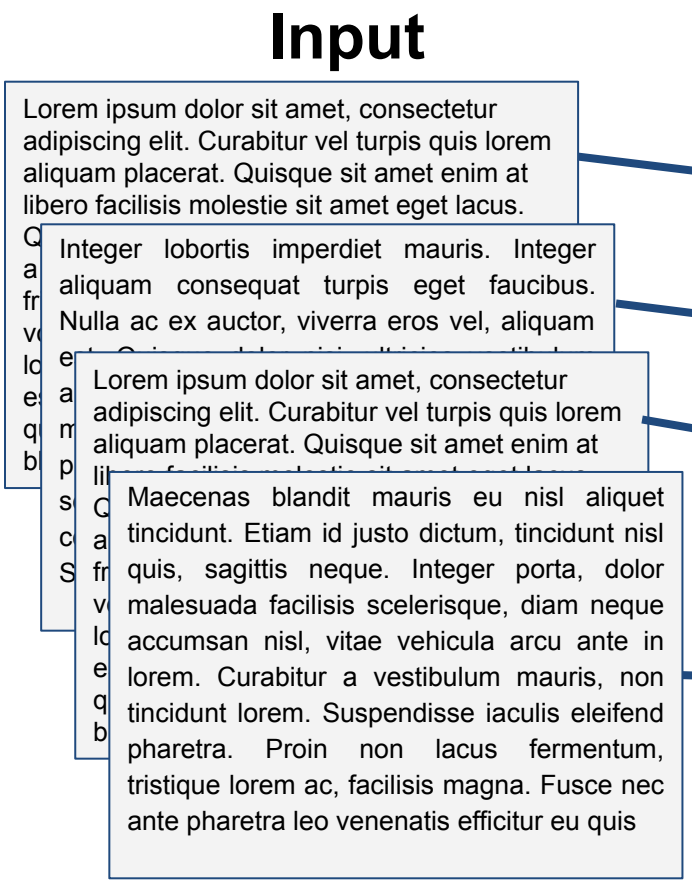
Category 1

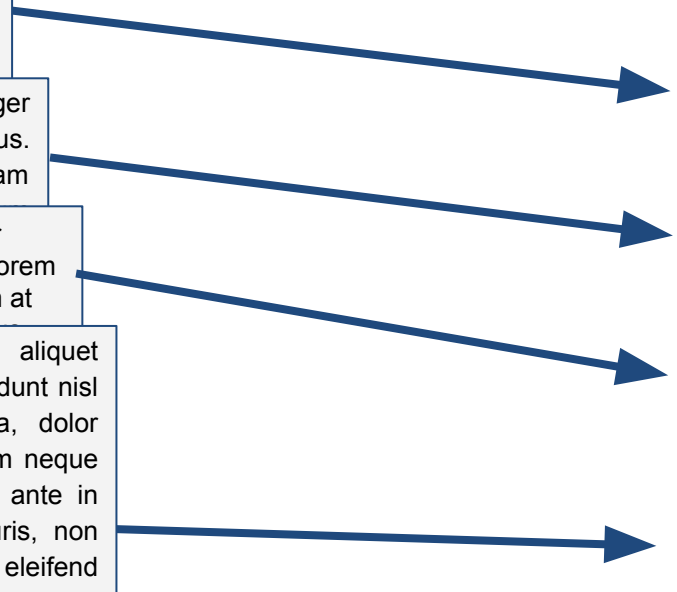
Category 2

Category 3

Category 1

Text Classification Problem

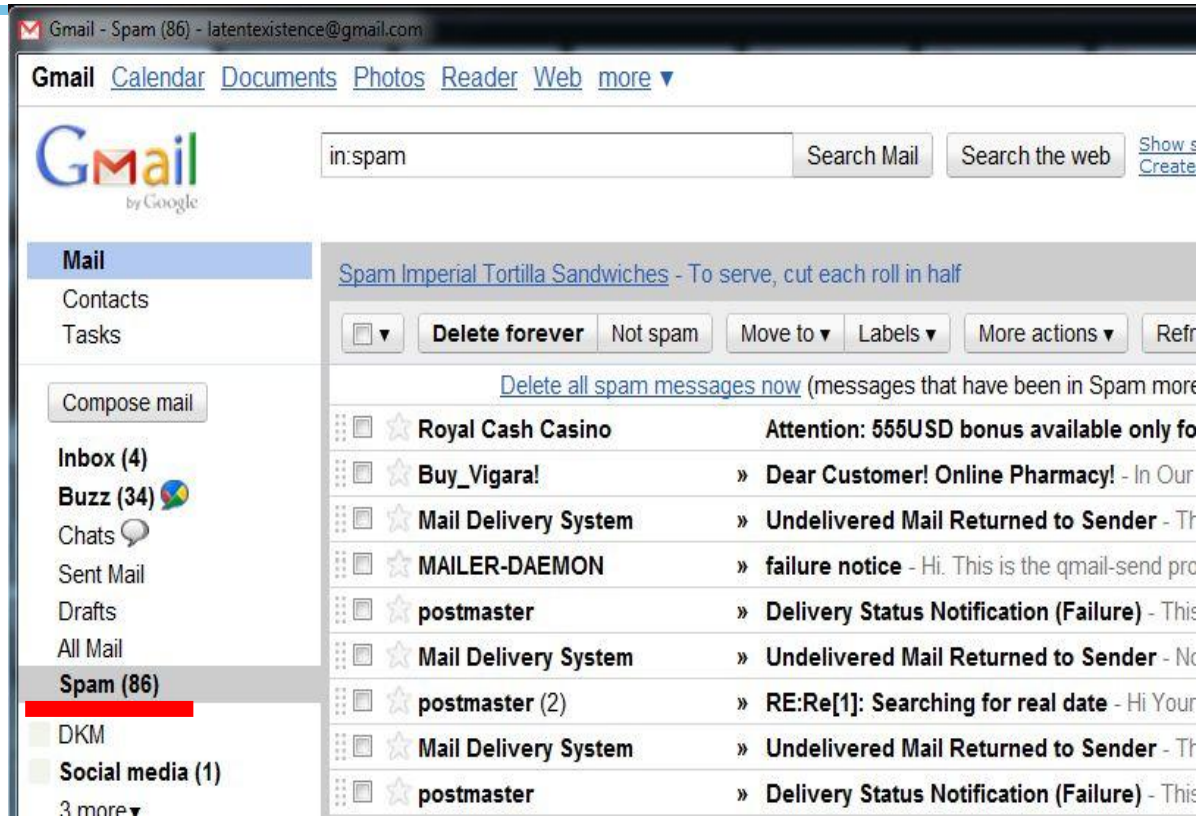




For every input assign an output. Outputs are called as labels or categories.

Classification Task in General

Image Classification



Spam Classification

How Search Engines Uses Text Classification

How Search Engines Uses Text Classification

- Language identification (classes: English vs. French etc.)

How Search Engines Uses Text Classification

- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. non spam).

How Search Engines Uses Text Classification

- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. non spam).
- Sentiment detection: is a movie or product review positive or negative (positive vs. negative).

How Search Engines Uses Text Classification

- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. non spam).
- Sentiment detection: is a movie or product review positive or negative (positive vs. negative).
- Topic-specific or vertical search – restrict search to a “vertical” like “related to health” (relevant to vertical vs. not) (Text classification).

Formal Definition of Text Classification

Formal Definition of Text Classification

Given:

- A document space **X**
  - Documents are represented in this space – typically some type of high-dimensional space.

Formal Definition of Text Classification

Given:

- A document space **X**
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes **C** = $\{c_1, c_2, \ldots, c_J\}$.
  - The classes are human-defined for the needs of an application (e.g., spam vs. non spam).

Formal Definition of Text Classification

Given:

- A document space **X**
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes **C** = $\{c_1, c_2, \ldots, c_J\}$.
  - The classes are human-defined for the needs of an application (e.g., spam vs. non spam).
- A training set **D** of labeled documents. Each labeled document $\langle d, c \rangle \in \mathbf{X} \times \mathbf{C}$

Formal Definition of Text Classification

Given:

- A document space **X**
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes **C** = $\{c_1, c_2, \ldots, c_J\}$.
  - The classes are human-defined for the needs of an application (e.g., spam vs. non spam).
- A training set **D** of labeled documents. Each labeled document $\langle d, c \rangle \in \mathbf{X} \times \mathbf{C}$

- Using a learning method or learning algorithm, we then wish to learn a classifier γ that maps documents to classes:

$$\gamma : \mathbf{X} \rightarrow \mathbf{C}$$

Classification methods: 1. Manual

Classification methods: 1. Manual

- Manual classification was used by Yahoo in the beginning of the web. Also: DMOZ (open directory project), PubMed.
- Very accurate if job is done by experts.
- Consistent when the problem size and team is small
- Scaling manual classification is difficult and expensive.
  - We need automatic methods for classification.

Classification methods: 2. Rule-based

Classification methods: 2. Rule-based

- E.g., Google Alerts is rule-based classification.
- Set of rules, that assigns a category to an input.
- Accuracy is very high if a rule has been carefully refined over time by a subject expert.
- Building and maintaining rule-based classification systems is cumbersome and expensive.

Classification methods: 3. Statistical/Probabilistic

Classification methods: 3. Statistical/Probabilistic

- This was our definition of the classification problem – text classification as a learning problem
  1. Supervised learning of a the classification function γ and
  2. application of γ to classifying new documents

- We will look at a methods for doing this: Naive Bayes.
- No free lunch: requires hand-classified training data.
- But this manual classification can be done by non-experts.

Naive Bayes Classifier

Majority of the slides on this topic is from:
https://www.cis.uni-muenchen.de/~hs/teach/14s/ir/pdf/13bayes.flat.pdf

Thank You!