Information Retrieval and Text Mining

Informtation Retrieval Models and Ranking Algorithms

Nuno Escudeiro (nfe@isep.ipp.pt)
Ricardo Almeida (ral@isep.ipp.pt)

Session outline

1. Informational Retrieval Models Basics
2. Common IR Models overview
3. Probabilistic Models (BIM, BM25)
4. Language Models (LMIR)
5. Smoothing techniques

Learning outcomes

At the end of this session we will be able to:

- Explain what is an IR model and its purpose
- Identify categories of IR models, their common and distinct characteristics and the tasks they are suited for
- Apply the BM25 model in IR applications
- Apply the LMIR model in IR applications
- Apply the main smoothing language models.

1. Information Retrieval Models

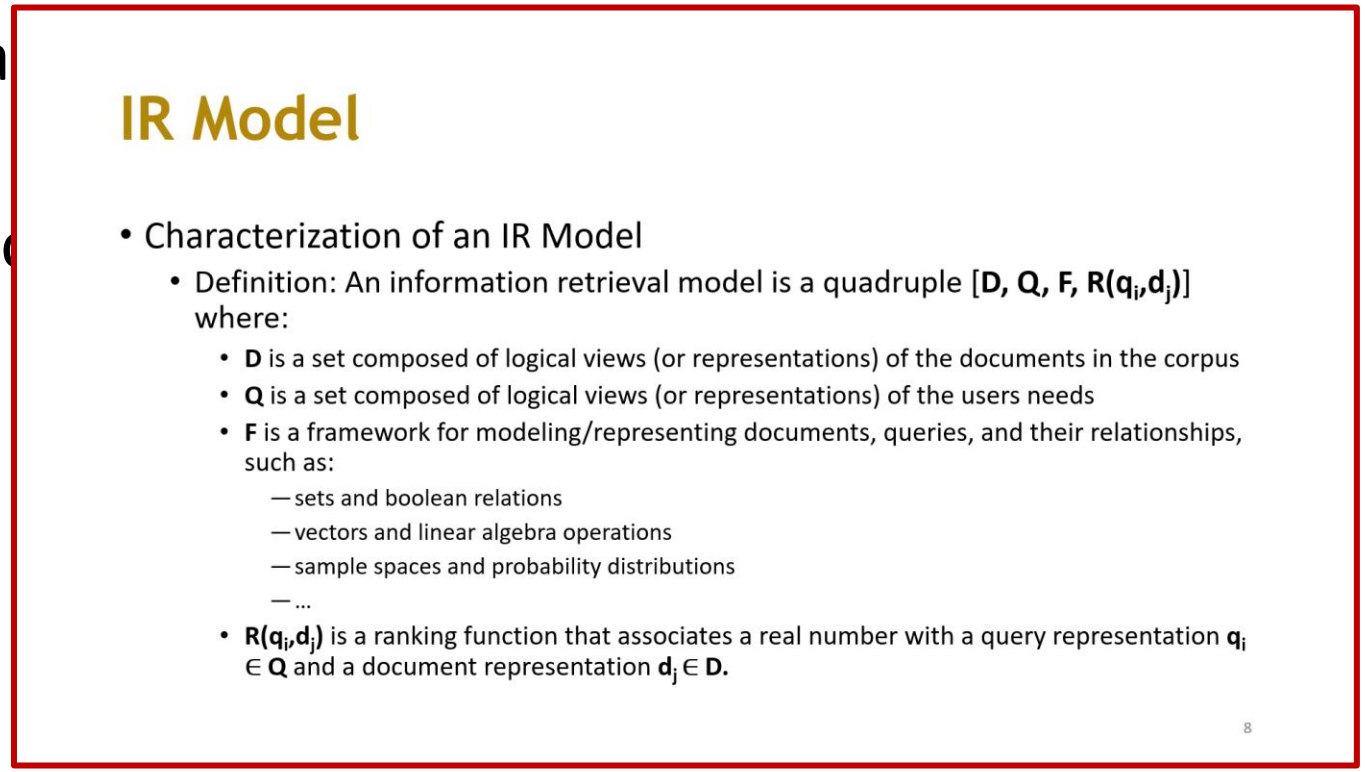
The basics.

Information Retrieval Models

- An Information Retrieval (IR) Model is a mathematical framework or computational method used to **represent**, **store**, **retrieve**, and **rank** relevant documents from a collection based on a user's query
- An IR Model is essential for search engines, document retrieval systems, and AI assistants
- It helps find the right information quickly and efficiently from vast amounts of unstructured data.

Information Retrieval Models

- An Information Retrieval (IR) Model is a mathematical framework or computational method used to **represent**, **store**, **retrieve**, and **rank** relevant documents from a collection based on a user's query
- An IR Model is essential for sea
  systems, and AI assistants
- It helps find the right informati
  amounts of unstructured data.



Information Retrieval Models

IR Models define:

- How documents and queries are represented (e.g., as sets of keywords, vectors, or probability distributions)
- How relevance to a given query is determined (e.g., exact match, term weighting, or semantic meaning)
- How documents are ranked (e.g., based on cosine similarity, BM25 scores, or deep learning embeddings).

Information Retrieval Models

The primary goal of an IR model is to help users find the most relevant information from a large collection of documents efficiently.

Purposes:

- Match queries with documents
- Rank documents by relevance
- Handle partial and inexact matches
- Improve search efficiency
- Support different types of searches

Information Retrieval Models

The primary goal of an IR model is to help users find the most relevant information from a large collection of documents efficiently.

Purposes:

- Match queries with documents
- Rank documents by releva
- Handle partial and inexact m
- Improve search efficiency
- Support different types of searches

Identify which documents are relevant to a user's search query

Unlike databases that require exact keyword matches, IR models rank results even if they are not identical to the query

Orders retrieved documents based on how well they satisfy the query

Enables fast retrieval from large-scale document collections (e.g., web search engines, digital libraries).

- Keyword-based search (Google search)
- Concept-based retrieval (semantic search using AI)
- Question-answering (chatbots like ChatGPT).

Classes of IR Models

- Algebraic models
- Statistical models

- Probabilistic models
- Language models
- Neural Networks, Deep Learning

- Exact match
- Ranked match

- Bag of Words
- Structural

- Probabilistic models
- Language models

Probabilistic Retrieval models

- Probabilistic models treat retrieval as a probabilistic inference task, estimating the **probability that a document is relevant given a query**
- They are based in:
  - Probability of relevance
  - Document and query modeling
  - Scoring functions
- Algorithms such as Binary Independence Model (BIM) and Okapi BM25 are examples of probabilistic retrieval models.

Language Modeling Approaches

- Language models view documents and queries as sequences of terms and model the **probability of observing a particular sequence**
- They are based in:
  - Document and query language modeling
  - Term smoothing techniques
  - Query likelihood models
- Some exemples are:
  - Language Model for Information Retrieval (LMIR)

2. Common IR Models

Boolean Model; Vector Space Model; Probabilistic Models, Language Models, Neural Retrieval Models

Common IR Models

- Boolean Model
- Vector Space Model
- Probabilistic Models
- Language Models
- Neural Retrieval Models

Common IR Models: Boolean

Boolean Retrieval Model

Uses Boolean logic (AND, OR, NOT) to match queries exactly with documents

How it works:

- Documents are represented as sets of index terms (keywords)
- A document is either retrieved or not (no ranking)
- Queries are written in Boolean expressions like:

  ("Machine AND Learning") → retrieves documents containing both words

  ("Deep OR Neural") → retrieves documents containing either word

  ("AI AND NOT Robotics") → retrieves documents with AI but not Robotics

Pros:
- ✅ Simple and easy to understand
- ✅ Works well in structured databases

Cons:
- ❌ No ranking of results (all matching documents are treated equally)
- ❌ Requires users to know exact keywords.

Common IR Models: Vector Space

Vector Space Model (VSM)

Represents documents and queries as vectors in a high-dimensional space

How it works:

- Uses Term Frequency-Inverse Document Frequency (TF-IDF) to weigh terms
- Measures similarity between query vector and document vectors using cosine similarity
- Ranks documents based on their similarity score

Pros: ✅ Provides ranking of results

✅ Handles partial matches better than Boolean retrieval

Cons: ❌ Assumes terms are independent (ignores word relationships)

❌ Computationally expensive for large datasets.

Common IR Models: Probabilistic

Probabilistic Retrieval Model

Uses probability theory to estimate the likelihood that a document is relevant to a query

How it works:

- Documents are ranked based on probability of relevance.
- Uses BM25 (Best Matching 25) formula, an improved ranking function.

Pros: ✅ More effective ranking than VSM.
✅ Handles document length normalization.

Cons: ❌ Requires tuning parameters for best performance.
❌ Assumes independence of terms (like VSM).

Common IR Models: Language Model

Language Model for Information Retrieval (LMIR)

Treats documents as probability distributions over words and ranks them based on how likely they are to generate the query.

How it works:

- Computes P(Query | Document) using statistical models.
- Common approaches:
  - Unigram Language Model (assumes word independence).
  - Smoothing Techniques (Laplace, Jelinek-Mercer, Dirichlet).

Pros:
✅ Works well in modern search engines.
✅ Allows query expansion and relevance feedback.

Cons:
❌ More complex to implement.
❌ Requires large datasets to estimate probabilities accurately.

Common IR Models: Neural Retrieval

Neural Retrieval Models (Deep Learning-Based)
Uses deep learning (neural networks) to model document-query relevance.

How it works:
- Uses word embeddings (e.g., Word2Vec, BERT) to capture word meanings.
- Learns complex relationships between words and documents.

Pros: ✅ Captures semantic meaning beyond keywords.
✅ Context-aware (e.g., "Apple" (fruit) vs. "Apple" (company)).
Cons: ❌ Computationally expensive.
❌ Requires large corpus of training data.

Common IR Models

Comparison of Retrieval Models				
Model	**Ranking**	**Partial matches**	**Computational cost**	**Real-World Use**
Boolean Model	No	No	Fast	Simple DBs, Legal Docs
Vector Space Model (VSM)	Yes	Yes	Medium High	Early search engines
BM25 (Probabilistic Model)	Yes	Yes	Medium	Modern search engines
Language Models (LMIR)	Yes	Yes	High	AI-based search
Neural Retrieval (BERT, ColBERT)	Yes	Yes	Very High	Google, AI assistants

Common IR Models: which one is best?

- If you need simple, exact searches → Boolean Model
- If you want ranked results with term weighting → Vector Space Model (VSM)
- If you need better ranking based on document relevance → BM25 (Probabilistic Model)
- If you want statistical probability-based retrieval → Language Models (LMIR)
- If you need deep learning and semantic search → Neural Models (BERT, ColBERT).

3. Probabilistic models

Binary Independence Model (BIM)

Okapi BM25

Binary Independence Model (BIM)

- The Binary Independence Assumption is that documents are binary vectors.
- Terms are independently distributed in the set of relevant documents and they are also independently distributed in the set of irrelevant documents.
- The representation is an ordered set of Boolean variables. More specifically, a document is represented by a vector $d = (x_1, ..., x_m)$ where $x_t=1$ if term t is present in the document d and $x_t=0$ if it's not.
- Many documents can have the same vector representation with this simplification. Queries are represented in a similar way.
- Terms in the document are considered independently from each other and no association between terms is modeled. This assumption is very limiting, but it has been shown that it gives good enough results for many situations.
- This independence is the "naive" assumption of a Naive Bayes classifier, where properties that imply each other are nonetheless treated as independent for the sake of simplicity. This assumption allows the representation to be treated as an instance of a Vector space model by considering each term as a value of 0 or 1 along a dimension orthogonal to the dimensions used for the other terms.

- C. T. Yu and G. Salton. 1976. Precision Weighting—An Effective Automatic Indexing Method. J. ACM 23, 1 (Jan. 1976), 76–88. https://doi.org/10.1145/321921.321930
- Robertson, S.E. and Jones, K.S. (1976), Relevance weighting of search terms. J. Am. Soc. Inf. Sci., 27: 129-146. https://doi.org/10.1002/asi.4630270302

Okapi BM25

- Was developed in the context of the Okapi System
- It is a ranking algorithm used in information retrieval and search engines for scoring and ranking documents based on their relevance
- Key features:
  - Term frequency and document length normalization
  - Inverse document frequency
  - Term saturation
- BM25 incorporates these features into a scoring formula to rank documents
- Goal: be sensitive to term frequency and document length while not adding too many parameters (Robertson and Zaragoza 2009; Spärck Jones et al. 2000).

Okapi BM25 - Generative model for documents

- Distribution of term frequencies (t_f) follows a binomial distribution – approximated by a Poisson distribution.

Okapi BM25 - Working principles

- Term Frequency and Inverse Document Frequency (TF-IDF):
  - builds upon the concept of TF-IDF (a statistical measure used to evaluate the importance of a term in a document relative to a collection of documents)
  - TF - measures the frequency of a term in a document, indicating how often the term appears
  - IDF - measures the rarity of a term across the entire document collection, indicating how much information the term provides
  - incorporates both TF and IDF components to calculate document scores

Okapi BM25 - Working principles

- Document Length Normalization:
  - Normalizes the term frequency component based on the length of the document
  - Longer documents tend to have higher term frequencies, so normalization helps in mitigating the bias towards longer documents
  - adjusts the impact of term frequency based on document length
- Term Saturation
  - to prevent the TF component from dominating the score for terms with very high frequencies
  - ensures that the score does not grow indefinitely with increasing term frequencies.

Okapi BM25 - Working principles

- Query Term Weighting:
  - Applies term weighting to query terms to adjust their contribution to the document score
  - Terms appearing in longer documents or being frequent across the entire collection are penalized to avoid over-emphasis.
  - Helps in capturing the specificity and relevance of query terms to the document.

Okapi BM25 - Working principles

Computing BM25:

- Calculates the relevance score S of a document D to a query Q as follows:
- $S(D,Q)=\sum_{i=1}^{n} IDF(qi)\ X\ \frac{f(q_i,D)\ X\ (K_1+1)}{f(q_i,D)+K_1\ X\ (1-b+b\ X\ \frac{|D|}{avg\_doc\_length})}$ where:
  - q_i is each term in the query Q
  - $f(q_i, D)$ is the frequency of term q_i in document D
  - $|D|$ is the length of document D
  - avg_doc_length is the average document length in the collection
  - K_1 and b are tuning parameters.

Okapi BM25 - Working principles

- Tuning Parameters:
  - K_1 and b are tuning parameters, which control the impact of term frequency and document length normalization, respectively
  - These parameters are typically empirically tuned based on the characteristics of the dataset and the retrieval task

By default b = 0.75 e K_1 = 1.2
K_1 varies in the range [0, 3], common choices in [0.5 e 2]
b varies in the range [0,1] , common choices in [0.3,0.9]

Okapi BM25 - Example

- d1: "The quick brown fox jumps over the lazy dog."
- d2: "A lazy dog is a happy dog."
- d3: "The brown fox is fast."
- d4: "The dog is brown."

- Query: "quick brown fox"
- k1=1,5
- b=0,75

Okapi BM25 - Example

	d1	d2	d3	d4	q	d1	d2	d3	d4		d1	d2	d3	d4
the	2	0	1	1	0	0,222222	0	0,2	0,25	0,124939	0,027764	0	0,024988	0,031235
quick	1	0	0	0	1	0,111111	0	0	0	0,60206	0,066896	0	0	0
brown	1	0	1	1	1	0,111111	0	0,2	0,25	0,124939	0,013882	0	0,024988	0,031235
fox	1	0	1	0	1	0,111111	0	0,2	0	0,30103	0,033448	0	0,060206	0
jumps	1	0	0	0	0	0,111111	0	0	0	0,60206	0,066896	0	0	0
over	1	0	0	0	0	0,111111	0	0	0	0,60206	0,066896	0	0	0
lazy	1	1	0	0	0	0,111111	0,142857	0	0	0,30103	0,033448	0,043004	0	0
dog	1	2	0	1	0	0,111111	0,285714	0	0,25	0,124939	0,013882	0,035697	0	0,031235
a	0	2	0	0	0	0	0,285714	0	0	0,60206	0	0,172017	0	0
is	0	1	1	1	0	0	0,142857	0,2	0,25	0,124939	0	0,017848	0,024988	0,031235
happy	0	1	0	0	0	0	0,142857	0	0	0,60206	0	0,086009	0	0
fast	0	0	1	0	0	0	0	0,2	0	0,60206	0	0	0,120412	0

Okapi BM25 - Example

- Média dos Documentos=(9+7+5+4)/4 = 6,25

			d1	d2	d3	d4
		IDF	$f(q_i,d1)$	$f(q_i,d2)$	$f(q_i,d3)$	$f(q_i,d4)$
q1	quick	0,6021	1,0000			
q2	brown	0,1249	1,0000			
q3	fox	0,3010	1,0000			
	S(D, qi)		0,502554			
			0,104289			
			0,251277			
	S(D, Q)		0,858121			

4. Language Models

LMIR

Smoothing

Language Model

- Are used in many natural language processing such as:
  - Part-of-speech tagging
  - Speech recognition
  - Machine translation and information retrieval

Language Model for Information Retrieval

- Based on the probability distribution of terms in documents and queries
- Key Features:
  - Document and query language modeling
  - Probability estimation using maximum likelihood or Bayesian approaches
  - Smoothing techniques to handle unseen terms
- LMIR computes the likelihood of generating a query given a document using language modeling principles.

Language Model for Information Retrieval

- Language models are probabilistic models that capture the probabilities of sequences of words in a language
  - Unigram model: how likely is the word “or” to appear (in a language)?
    - P(“or”) = ?
  - Bigram model: given that current word is “not”, what is the probability of next word being “or”?
    - P(“or” | “not”) = ?
  - Trigram model: given the current sequence “or not”, what is the probability of next word being “to”?
    - P(“to” | “or not”) = ?

Language Model for Information Retrieval

- We use the instantiations of the language to estimate the probabilities of words and sequences
  - Large corpora are required – the larger the corpora, the better approximation of the true word distributions in a given language
  - In IR, we build language models
    - From individual documents
    - From the whole document collections
  - In other applications we build language models over the largest corpora we can compile.

Language Model for Information Retrieval

- Language models for IR are also probabilistic models
- Language models for IR model the query generation process
- Given a document *d* and a query *q*, what is the probability of *q* being sampled from the language model of *d*
- In other words, we want to estimate P(Q = *q* | D = *d*)
  - Q:Compare this with the probability we estimated in classic probabilistic retrieval
    - P(R = 1 | Q = q, D = d)
- The probability of a document generating a query is directly used to rank documents
  - I.e., we rank the documents in decreasing order of P(Q = q | D = d)
- Central question: how do we estimate P(Q = q | D = d)?

“Language Bowl” Metaphor

- Consider the frequencies of d2 in the following table:

	d1	d2	d3	d4
to	4	2	0	0
do	2	0	3	3
is	2	0	0	0
be	2	2	2	2
or	0	1	0	0
not	0	1	0	0
I	0	2	2	0
am	0	2	1	0
what	0	1	0	0
think	0	0	1	0
therefore	0	0	1	0
da	0	0	0	3
let	0	0	0	2
it	0	0	0	2

“Language Bowl” Metaphor

- What is the probability of having “not” in d2
  - P(“not” | d2) = 1 / (11)
- What is the probability of having “to” and “not”
  - P(“to”, “not” | d2) = P(“to” | d2) * P(“not” | d2) = 2/11 * 1/11

- P(A, B) = P(A)*P(B) iif A and B are independent!

Estimating probabilities

- For the unigram language model we need to estimate
  - P(term) for every term in the text
- For the bigram language model we additionally need to estimate
  - P(term | previous term) for every pair of terms that appear one after another
- Q:How do we estimate these?
  - Unigram language model
    - $P(t_i) = n_i / n_T$
    - n_i is the number of occurrences of term t_i in the collection
    - n_t is the total number of word occurrences (i.e., tokens) in the collection
  - Bigram language model
    - $P(t_i \mid t_{i-1}) = n(t_{i-1}, t_i) / n(t_{i-1})$
    - $n(t_{i-1}, t_i)$ is the number of occurrences of bigram $t_{i-1}t_i$ in the collection
    - $n(t_{i-1})$ is the number of occurrences of term t_{i-1} in the collection

Estimating probabilities

- Consider the following Documents (D)
  - d1: "The quick brown fox jumps over the lazy dog."
  - d2: "A lazy dog is a happy dog."
  - d3: "The brown fox is fast."
  - d4: "The dog is brown.“
- Estimating word probabilities for the unigram model:

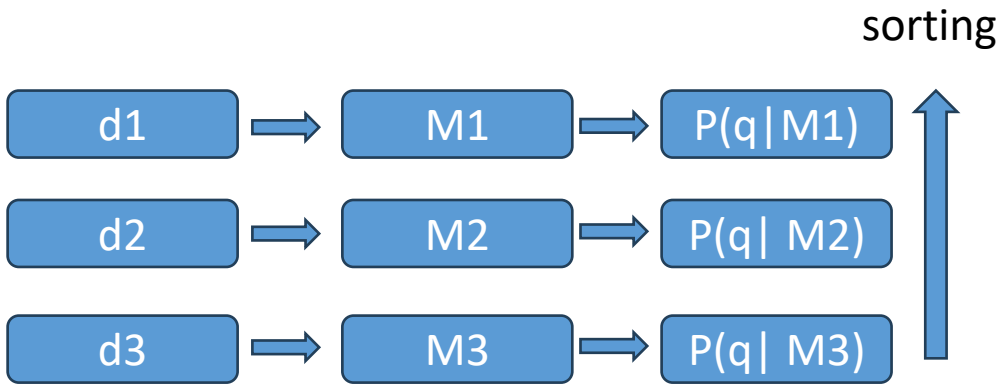
ti	the	quick	dog	fast	brown	…
P(ti)	4/25	1/25	4/25	1/25	3/25	…

- Estimating the conditional probabilities for the bigram model:

ti-1, ti	lazy, dog	the, fox	is, brown
P(ti|ti-1)	2/2 = 1	0	1/3

Query likelihood model for ranking

- Given a document collection D and a query q we need to estimate the probability P(q | d) for every document d in D
- In the query likelihood model, we estimate the probability P(q | d) as the probability that the language model built from d generates the query q
- Algorithm 1. Compute the language model Mi for every document di in D.
- Compute the probability P(q | Mi) for every language model Mi



Query likelihood model for ranking - example

- Consider the following Documents (D)
  - d2: "A lazy dog is a happy dog."
  - d3: "The brown fox is fast.“
  - d4: "The dog is brown.“
- We are given the query “lazy and dog and happy”
- Let’s rank the documents according to unigram LM for IR (ignore stopwords)
- **Step 1:** Compute language models of individual documents
  - M2: P(“lazy”)=0.25, P(“dog”)=0.5, P(“happy”)=0.25
  - M3: P(“brown”)=0.33, P(“fox”)=0.33, P(“fast”)=0.33
  - M4: P(“dog”)=0.5, P(“brown”)=0.5

Query likelihood model for ranking - example

- Consider the following Documents (D)
  - d2: "A lazy dog is a happy dog."
  - d3: "The brown fox is fast.“
  - d4: "The dog is brown.“
- We are given the query “lazy and dog and happy”
- Let’s rank the documents according to unigram LM for IR (ignore stopwords)
- **Step 2**: Compute the probabilities P(q | Mi)
  - P(q | M2) = P(“lazy”|M2) * P(“dog ”|M2) * P(“happy”|M2) = 0.25 * 0.5 * 0.25
  - P(q | M3) = P(“lazy”|M3) * P(“dog ”|M3) * P(“happy”|M3) = 0 * 0 * 0
  - P(q | M4) = P (“lazy”|M4) * P(“dog”|M4) * P(“happy”|M4) = 0 * 0.33 * 0

5. Smoothing techniques

Avoid 0 probability bias for query terms that are not present in a document

Smoothing language models

- Zero frequency problem: Models we've considered so far give probability of 0 to queries containing any term that does not occur in the document
- We can prevent this by using smoothing techniques
- Smoothing techniques
  - Change the probability distribution of terms in the language model
  - Assign some small probability to unseen word
- Three prominent smoothing schemes
  - Laplace smoothing
  - Jelinek-Mercer smoothing
  - Dirichlet smoothing

Laplace smoothing

1. Adds a fixed small count (often it's 1) to all word counts
2. (Re)normalizes to get a probability distribution

$$P(t_i \mid M_d) = \frac{n_{i\,d} + \alpha}{n_d + |V| \times \alpha}$$

- The probability of any unseen word equals

$$P(t_{uns} \mid M_d) = \frac{\alpha}{n_d + |V| \times \alpha}$$

Jelinek-Mercer smoothing

- Laplace smoothing assumes that all unseen words are equally likely
- Jelinek-Mercer smoothing (also known as interpolated smoothing)
  1. Additionally builds a language model MD from the whole document collection D
  2. Interpolates between probabilities of the query according to the
     1. Local LM– language model Md built from the particular document d
     2. Global LM – language model MD built from the whole collection

$$P(t_i|M_d)=\lambda \times P(ti|M_d) + (1- \lambda) \times P(ti|M_D)$$

- The probability of a word unseen in the document d still gets some probability from the global language model
  - Probability of an unseen word depends on its frequency in whole collection

Dirichlet smoothing

- Dirichlet smoothing can be seen as a generalization of the Laplace smoothing
  - Each word unseen in the document gets an artificial extra count
  - But the extra count is not fixed, depends on the global probability of the term
    - In this respect, Dirichlet smoothing is similar to Jelinek-Mercer smoothing

$$P(t_i|M_d) = \frac{n_{i,d} + \mu \times P(t_i|M_D)}{n_d + \mu}$$

- Less frequent words in the document get more probability from the global component
  - The value of the constant μ determines the scale of the global probability's contribution
  - Provides adaptive smoothing (small μ for short docs, large μ for long docs).

Ranking with non-textual Features

- Relying on other sources/indicators/features of relevance than content similarity
  - Query text features:
    - overlap with title, snippet, URL, domain, …
  - Browsing features:
    - time on page, deviation from average times, …
  - Click-through features :
    - click frequency, click probability, click deviation …
- Web search click logs are long --> TB/day

References

- https://users.dcc.uchile.cl/~rbaeza/mir2ed/
- https://ciir.cs.umass.edu/pubfiles/ir-318.pdf
- https://www.youtube.com/watch?v=k6FAZJGTZJo
- https://www.uni-mannheim.de/media/Einrichtungen/dws/Files_People/Profs/goran/6-Language-Modeling-IR-FSS20.pdf
- https://nlp.stanford.edu/IR-book/
- Agichtein, Eugene, et al. "Learning user interaction models for predicting web search result preferences." *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 2006.