Retrieval Models 2

CS4422/7263 Information Retrieval
Lecture 07

Jiho Noh

Department of Computer Science
Kennesaw State University

CS4422/7263 Summer 2025

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

1. **Okapi BM25**

2. **Binary Independence Model**
   - Probabilistic Motivation
   - $P(R|d,q)$

3. **Query-likelihood model**
   - Language Models
   - $P(q|R,d)$
   - Smoothing techniques

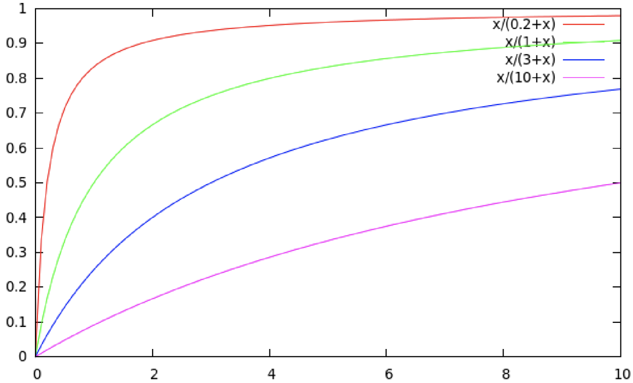
Okapi BM25

- Most successful term weighting scheme
  - Apache Lucene switched to Okapi BM25 as the default scoring function (2015)
- BM25 “Best Match 25” (and out of many trials, 25th worked the best!)
  - First used in the Okapi system at TREC-3 by Robertson
  - Started to be increasingly adopted by other teams during the TREC competitions
- Goal: be sensitive to term frequency and document length while not adding too many parameters (Robertson and Zaragoza 2009; Spark Jones et al. 200)

Term Frequency Saturation

- We want the contribution of term frequency in ranking documents to be saturated.
- What is **Saturation**
  - In Electronics, put (a device) into a state in which no further increase in current is achievable.
  - In CS, Saturation is a function that has been mathematically “smoothed” or “flattened”.

Term Frequency Saturation



$$\frac{tf}{k + tf}$$

- simple parametric curve for the saturation properties
  - For high values of k, increments in *tf* continue to contribute significantly to the score
  - Contributions tail off quickly for low values of k

“Early” versions of BM25

- **Version 1:** using the saturation function.

$$c_i^{BM25v1}(tf_i) = c_i^{BIM} \cdot \frac{tf_i}{k + tf_i}$$

- **Version 2:** BIM simplification to IDF

$$c_i^{BM25v1}(tf_i) = \log \frac{N}{df_i} \cdot \frac{(k+1)tf_i}{k + tf_i}$$

  - $(k+1)$ factor doesn’t change ranking, but makes term score 1 when $tf_i = 1$.
  - Similar to *tf-idf*, but term scores are bounded.

Document Length Normalization

- Issue: Longer documents are likely to have larger tf_i values.
- We can't just simply penalize a longer document. Why might documents be longer?
  - **Verbosity**: suggests observed tf_i is too high
  - **Larger scope**: suggests observed tf_i may be right
- A real document collection probably has both effects.
- So, should apply some kind of **partial normalization** to make it balanced between these two cases.

Document Length Normalization (Cont.)

- Document length: $dl = \sum_{i \in V} tf_i$
- Average document length over collection: $avgdl$
- Partial length normalization component

$$B = \left((1 - b) + b \cdot \frac{dl}{avgdl} \right), \quad 0 \leq b \leq 1$$

  - $b = 1$, full document length normalization
  - $b = 0$, no document length normalization

Okapi BM25 – Putting all together

- Normalize *tf* using document length: $tf_i' \rightarrow tf_i/B$

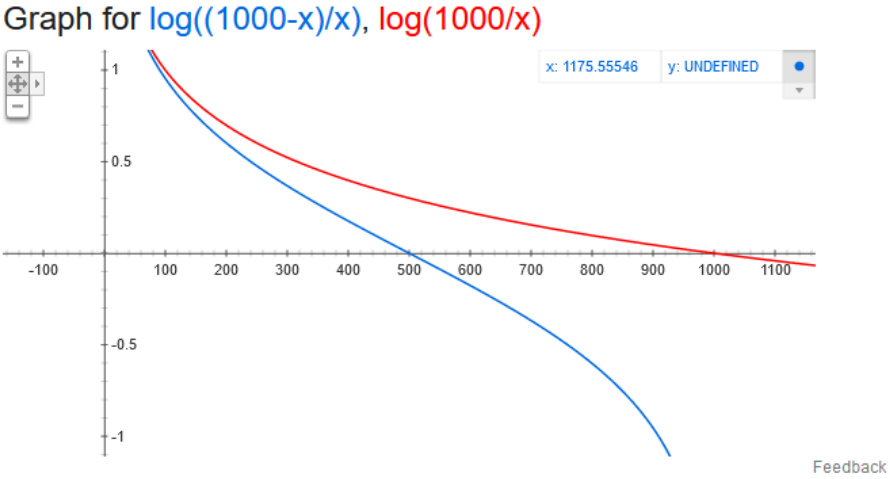
$$\begin{aligned}
c_i^{BM25v1}(tf_i) &= \log \frac{N}{df_i} \cdot \frac{(k+1)tf_i}{k+tf_i} \\
&= \log \frac{N}{df_i} \cdot \frac{(k+1)tf_i/B}{k+tf_i/B} \\
&= \log \frac{N}{df_i} \cdot \frac{(k+1)tf_i}{kB+tf_i} \\
&= \log \frac{N}{df_i} \cdot \frac{(k+1)tf_i}{k\left((1-b)+b\frac{dl}{avgdl}\right)+tf_i}
\end{aligned}$$

Okapi BM25 – Summary

$$score(D, Q) = \sum_{i \in Q} \log \frac{N}{df_i} \cdot \frac{(k+1)tf_i}{k\left((1-b) + b\frac{dl}{avgdl}\right) + tf_i}$$

- k controls term frequency scaling
- b controls document length normalization
- k, b are free parameters, usually $k \in [1.2, 2.0], b = 0.75$

Okapi BM25 – Better IDF (from BIM)



Graph for log((1000-x)/x), log(1000/x)

x: 1175.55546 y: UNDEFINED

Okapi BM25 – Summary

$$score(D,Q) = \sum_{i \in Q} \log\left(\frac{N - df_i + 0.5}{df_i + 0.5} + 1\right) \cdot \frac{(k+1)tf_i}{k\left((1-b) + b\frac{dl}{avgdl}\right) + tf_i}$$

- Sum the scores for each query term.
- There are several interpretations for IDF and slight variations on its formula. The original is derived from the BIM.
- Term frequency saturation.
- (k+1) doesn't change ranking, this can be removed.
- Document length normalization factor.

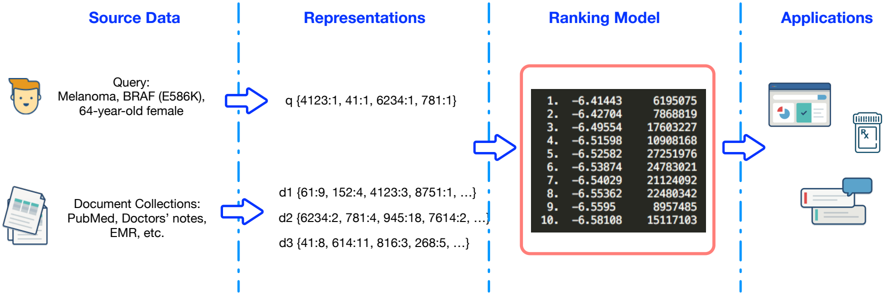
Okapi BM25 – Exercise

- Suppose your query is “president lincoln”
- Suppose you have documents with the term counts as below:

$$d = \{(15, 25), (15, 1), (15, 0), (1, 25), (0, 25) | (tf_{president}, tf_{lincoln})\}$$

- Suppose $N = 1,000,000$, df(president)=40,000, df(lincoln)=300,
- The document lengtyh is 90% of the average length: $dl/avgdl = 0.9$
- We pick $k = 1.2, b = 0.75$.
- (demo in python)

Uncertainties in IR



Source Data

Representations

Ranking Model

Applications

Query:
Melanoma, BRAF (E586K),
64-year-old female

Document Collections:
PubMed, Doctors' notes,
EMR, etc.

q {4123:1, 41:1, 6234:1, 781:1}

d1 {61:9, 152:4, 4123:3, 8751:1, ...}

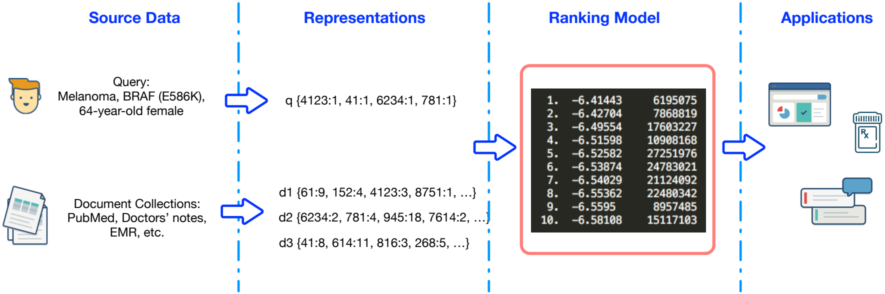
d2 {6234:2, 781:4, 945:18, 7614:2, ...}

d3 {41:8, 614:11, 816:3, 268:5, ...}

```
 1.  -6.41443     6195075
 2.  -6.42704     7868819
 3.  -6.49554    17603227
 4.  -6.51598    10908168
 5.  -6.52582    27251976
 6.  -6.53874    24783021
 7.  -6.54029    21124092
 8.  -6.55362    22480342
 9.  -6.5595      8957485
10.  -6.58108    15117103
```

- We want to present the documents in the order of relevance w.r.t the user's information need.
- The most "relevant" document to be the first, and so on.

Uncertainties in IR



Source Data

Representations

Ranking Model

Applications

Query:
Melanoma, BRAF (E586K),
64-year-old female

q {4123:1, 41:1, 6234:1, 781:1}

Document Collections:
PubMed, Doctors' notes,
EMR, etc.

d1 {61:9, 152:4, 4123:3, 8751:1, ...}

d2 {6234:2, 781:4, 945:18, 7614:2, ...}

d3 {41:8, 614:11, 816:3, 268:5, ...}

```
 1.  -6.41443     6195075
 2.  -6.42704     7868819
 3.  -6.49554    17603227
 4.  -6.51598    10908168
 5.  -6.52582    27251976
 6.  -6.53874    24783021
 7.  -6.54029    21124092
 8.  -6.55362    22480342
 9.  -6.5595      8957485
10.  -6.58108    15117103
```

- Conventional IR systems make **uncertain guesses** of how much a document is relevant
- Probabilities provide a principled foundation for uncertain reasoning.
- Key Idea: Rank by probability of relevance. i.e., $P(relevant|doc, query)$

The Probability Ranking Principle (PRP)

“If a reference retrieval system’s response to each request is a ranking of the documents in the collection in order of **decreasing probability of relevance** *to the user who submitted the request, 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.” — Robertson ’77*

1. **Okapi BM25**

2. **Binary Independence Model**
   - Probabilistic Motivation
   - $P(R|d,q)$

3. **Query-likelihood model**
   - Language Models
   - $P(q|R,d)$
   - Smoothing techniques

Probability basics

- **Uncertainty**: The lack of certainty, a state of limited knowledge where it is impossible to exactly describe the existing state
- **Probability**: the measurement of uncertainty

Joint Probability

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

when events A and B are independent

$$P(A \cap B) = P(A)P(B)$$

Bayes' Rule

- **Conditional probability** is the probability of some event A, given the occurrence of some other event B.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A), P(B)$ – **marginal probability** or **prior probability**, originally believed knowledge before new evidence is introduced
- $P(A|B)$ – **posterior probability**, degree of belief after incorporating new event B

Probability ranking strategy

- Let d represent a document in the collection
- Let R, NR represent a (relevant or not relevant) document w.r.t. the given query
- Given a document (d), we want to measure how much it is relevant (R), that is $P(R|d)$

$$P(R|d) = \frac{P(d|R)P(R)}{P(d)}$$
$$P(NR|d) = \frac{P(d|NR)P(NR)}{P(d)}$$

Since d is either relevant or irrelevant,

$$P(R|d) + P(NR|d) = 1$$

1. **Okapi BM25**
2. **Binary Independence Model**
   - Probabilistic Motivation
   - $P(R|d,q)$
3. **Query-likelihood model**
   - Language Models
   - $P(q|R,d)$
   - Smoothing techniques

Binary Independence Model (BIM)

- We need to take query q into consideration.
  - For each document d, we need to compute $P(R|q, d)$
- We are interested only in ranking documents. (This enables simplification in probability derivation.)
- This is a simple binary model; Documents and queries are represented by a vector of "binary" features indicating term occurrence.
- Assumption: Term occurrence event is an "independent" variable.
- We use **odds ratio** and **Bayes' rule** to derive a relevance scoring estimation function.

Binary Independence Model (BIM)

Odd Ratio

$$Odds = \frac{P(A)}{P(\overline{A})} = \frac{P(A)}{1 - P(A)}$$

Let $\vec{x}$ be binary term incidence vector representing d. We want to score a document d using the odd of the document being relevant given the query.

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

Binary Independence Model (BIM)

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

Let's look at the numerator first:
We want to express it in terms of $\vec{x}$, so that we can estimate the probability using term weights.

Using Bayes' rule while having q as a conditional variable, we get

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

Binary Independence Model (BIM)

The odds ratio becomes,

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

Simplify and re-arrange,

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

- First term is constant for a given query.
- Second term cannot be known beforehand. We need an estimation.

Binary Independence Model (BIM)

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

- The first term is the odd of a document being relevant given a query.

- Let's investigate the second term further.
- We will use **term independence assumption** (i.e., term occurrence events are independent) and the *chain rule*:

Binary Independence Model (BIM)

Joint probability chain rule

$$P(A, B) = P(A|B)P(B)$$

$$P(A, B, C) = P(A|B, C)P(B|C)P(C)$$

$$P(A_1, A_2, \ldots, A_n) = P(A_1|A_2, A_3, \ldots, A_n)P(A_2|A_3, \ldots, A_n) \cdots P(A_n)$$

When the events are independent to another:

$$P(A_1, A_2, \ldots, A_n) = P(A_1)P(A_2) \cdots P(A_n)$$

Binary Independence Model (BIM)

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

$\vec{x}$ is a sequence of words, and we can describe it as a joint probability of the occurrences of words. Using the chain rule:

$$O(R|q,\vec{x}) = O(R|q) \cdot \prod_{i=1}^{n} \frac{P(x_i|R,q)}{P(x_i|NR,q)}$$

Binary Independence Model (BIM)

$$O(R|q,\vec{x}) = O(R|q) \cdot \prod_{i=1}^{n} \frac{P(x_i|R,q)}{P(x_i|NR,q)}$$

x_i (term occurrence) is binary: either occur (1) or not (0)

$$\prod_{i=1}^{n} \frac{P(x_i|R,q)}{P(x_i|NR,q)} =$$

$$\prod_{x_i=1} \frac{P(x_i=1|R,q)}{P(x_i=1|NR,q)} \prod_{x_i=0} \frac{P(x_i=0|R,q)}{P(x_i=0|NR,q)}$$

Binary Independence Model (BIM)

For the sake of clarity, let

- $r_i = P(x_i = 1|R, q)$
- $s_i = P(x_i = 1|NR, q)$

$$\prod_{x_i=1} \frac{P(x_i = 1|R, q)}{P(x_i = 1|NR, q)} \prod_{x_i=0} \frac{P(x_i = 0|R, q)}{P(x_i = 0|NR, q)}$$

$$= \prod_{x_i=1} \frac{r_i}{s_i} \prod_{x_i=0} \frac{1 - r_i}{1 - s_i}$$

$$= \prod_{x_i=1,q_i=0} \frac{r_i}{s_i} \prod_{x_i=1,q_i=1} \frac{r_i}{s_i} \prod_{x_i=0,q_i=0} \frac{1 - r_i}{1 - s_i} \prod_{x_i=0,q_i=1} \frac{1 - r_i}{1 - s_i}$$

Binary Independence Model (BIM)

$$= \prod_{x_i=1,q_i=0} \frac{r_i}{s_i} \prod_{x_i=1,q_i=1} \frac{r_i}{s_i} \prod_{x_i=0,q_i=0} \frac{1-r_i}{1-s_i} \prod_{x_i=0,q_i=1} \frac{1-r_i}{1-s_i}$$

Assume, for all the terms not occur in the query (i.e., $q_i = 0$), $r_i = s_i$

$$= \prod_{x_i=1,q_i=1} \frac{r_i}{s_i} \prod_{x_i=0,q_i=1} \frac{1-r_i}{1-s_i}$$

- First product is for all matching query terms,
- Second product is for all non-matching query terms.

Binary Independence Model (BIM)

Putting it all together,

$$O(R|q,\vec{x}) = O(R|q) \cdot \prod_{x_i=1,q_i=1} \frac{r_i}{s_i} \prod_{x_i=0,q_i=1} \frac{1-r_i}{1-s_i}$$

- We want to convert the last term free from the document dependent terms, x_i's, so that it can be a constant value for each query.
- That is, we want to combine the last term over $(x_i = 0)$ with the previous term over $(x_i = 1)$.

Binary Independence Model (BIM)

Putting it all together,

$$= O(R|q) \cdot \prod_{x_i=1,q_i=1} \frac{r_i}{s_i} \prod_{x_i=1,q_i=1} \left(\frac{1-s_i}{1-r_i} \frac{1-r_i}{1-s_i} \right) \prod_{x_i=0,q_i=1} \frac{1-r_i}{1-s_i}$$

The second term is product of 1's, and we can combine the last term with the second term.

$$= O(R|q) \cdot \prod_{x_i=1,q_i=1} \frac{r_i(1-s_i)}{s_i(1-r_i)} \cdot \prod_{q_i=1} \frac{1-r_i}{1-s_i}$$

For ranking purpose, we can ignore all the constant terms and take the document dependent terms to estimate the document ranking scores.

Binary Independence Model (BIM)

Retrieval Status Value (RSV)

$$RSV = \log \prod_{x_i=1, q_i=1} \frac{r_i(1-s_i)}{s_i(1-r_i)} = \sum_{x_i=q_i=1} \log \frac{r_i(1-s_i)}{s_i(1-r_i)}$$

Interpretation:

RSV becomes the sum of log odds ratios of the matching terms.

BIM – Estimating RSV coefficients

- How do we compute $\sum \log \frac{r_i(1-s_i)}{s_i(1-r_i)}$?
  - $r_i = P(x_i = 1|R, q) \approx ?$
  - $s_i = P(x_i = 1|NR, q) \approx ?$
- For each term i, look at the table of document counts:

term presence	relevant (R)	non-relevant (NR)	total
$x_i = 1$	m	(n-m)	n
$x_i = 0$	(M-m)	(N-M-n+m)	(N-n)
total	M	(N-M)	N

BIM – Estimating RSV coefficients

- r_i (the probability of term occurrence in a relevant document) is difficult to estimate; Let's put it aside.
- From $\sum \log \frac{r_i(1-s_i)}{s_i(1-r_i)}$, let's investigate the s_i term.
- M is small enough to be ignored relatively to N (so does m)

$$\log \frac{1-s_i}{s_i} \approx \log \frac{N-M-n+m}{n-m} \approx \log \frac{N-n}{n} \approx \log \frac{N}{n}$$

Estimated RSV is the inverse document frequency (IDF) !!

Estimation challenge – r_i?

- r_i cannot be approximated as easily.
- r_i can be estimated in various ways:
    1. Constant; if we assume that $r_i = 0.5$, then we can just get *idf* weighting of terms

$$RSV = \sum_{x_i = q_i = 1} \log \frac{N}{n_i}$$

    2. From relevant document (relevance feedback); we will discuss this in later lecture.
    3. Proportional to probability of occurrence in collection: $\frac{1}{3} + \frac{2}{3} \cdot \frac{df_i}{N}$

Binary Independence Model (BIM) – Findings

- **IDF** measures the importance of the term in a document collection.
- **BIM** shows that RSV can be simplified in terms of IDF.
- We still need to consider the *term frequency factor* in computing the relevance of a document.
  - More occurrence of the term in a document means higher relevance.
  - But the frequency needs to be saturated.

1. **Okapi BM25**

2. **Binary Independence Model**
   - Probabilistic Motivation
   - $P(R|d,q)$

3. **Query-likelihood model**
   - Language Models
   - $P(q|R,d)$
   - Smoothing techniques

N-gram Language Models

- Language Model (LM) is the probability distribution over strings of text.
- Given a sequence of words $s = w_1, w_2, w_3, \ldots, w_n$, the LM estimates $P(s)$:
- Using the joint probability chain rule:

$$\begin{aligned} P(s) &= P(w_1, w_2, w_3, \ldots, w_n) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \cdots P(w_n|w_1, w_2, \ldots, w_{n-1}) \end{aligned}$$

Markov Assumption

- Only the last k words are relevant to the next word.
  - $\Rightarrow$ **k-th order Markov model**
- For example, in a bigram model ($k = 1$), the next word depends only on the previous word.
- $P(w_1, w_2, w_3, \ldots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2)\cdots P(w_n|w_{n-1})$
- Using, **Maximum Likelihood Estimation (MLE)**, we can estimate the probability of a word given the previous k words.

$$P(w_2|w_1) = \frac{C(w_1, w_2)}{C(w_1)}$$

where $C(w_1, w_2)$ is the count of the word pair (w_1, w_2) and $C(w_1)$ is the count of the word w_1.

Language Models

- The simplest form of this language model ignores the term dependency ($k = 0$), and this model is called **unigram language model**.
- Other language models
  - Grammar-based language models
  - Neural language models
- Some models are vital for *speech recognition, spelling correction, machine translation, and language generation.*
- However, unigram language model is sufficient for IR.

1. **Okapi BM25**

2. **Binary Independence Model**
   - Probabilistic Motivation
   - $P(R|d, q)$

3. **Query-likelihood model**
   - Language Models
   - $P(q|R, d)$
   - Smoothing techniques

Query-Likelihood Model — Intuitions

- A user forms a query thinking of words that would likely appear in a relevant document.
- A topic in a document can be represented as a language model, M_d.
- In the traditional probabilistic retrieval model, we compute $P(R|q, d)$.
- The basic language modeling approach ranks documents based on the probability of the model generating the query: $P(q|M_d)$.

Query-likelihood model

$$P(q|M_d) = K_q \prod_{t \in V} P(t|M_d)^{tf_{t,q}}$$

- where $K_q = L_q!/(tf_{t1,q}!\, tf_{t2,q}! \cdots tf_{N,q}!)$ with
  - L_q is the length of query q given the term frequencies tf in the query vocabulary V.
- K_q is a constant given a particular query, hence we can ignore it in ranking documents.
- Each document d is treated as a separate "language" which model is denoted by M_d.
- We need to estimate $P(t|M_d)$.

Estimating the query generation probability

- Estimating using Maximum Likelihood Estimation (MLE) and the unigram assumption.

$$\hat{P}(q|M_d) = \prod_{t \in q} \hat{P}_{mle}(t|M_d) = \prod_{t \in q} \frac{tf_{t,d}}{|d|}$$

- MLE makes the observed value of $tf_{t,d}$ most likely
- *Any problem?*

  What if $tf_{t,d} = 0$, a query term missing from document?

1. **Okapi BM25**

2. **Binary Independence Model**
   - Probabilistic Motivation
   - $P(R|d,q)$

3. **Query-likelihood model**
   - Language Models
   - $P(q|R,d)$
   - Smoothing techniques

Smoothing Techniques

- Document texts are a sample from the language model.
  - Missing words should not have zero probability of occurring.
- **Smoothing** is a technique for estimating probabilities for missing (or unseen) words.
  - Lower (or discount) the probability estimates for words that are **seen** in the document text.
  - Assign that “left-over” probability to the estimates for the words that are **not seen** in the text.

Laplace Smoothing

- Also called "add-one smoothing"
- Just add 1 to all the counts

MLE

$$P_{mle}(t|M_d) = \frac{tf_{t,d}}{|d|}$$

Laplace

$$P_{laplace}(t|M_d) = \frac{tf_{t,d} + 1}{|d| + |V|}$$

Collection LM Smoothing

- We want to mix two different probability distributions
  - For seen words (foreground probability): $P(t|M_d)$
  - For unseen words (background probability): $P(t|M_c)$, where M_c is the collection language model
- We take the linear combination of two

$$(1 - \lambda)P(t|M_d) + \lambda P(t|M_c)$$

Jelinek-Mercer Smoothing

- Document ranking score:

$$P(q|M_d, C) = \prod_{t \in q} (1 - \lambda) P(t|M_d) + \lambda P(t|M_c)$$

  - λ is a model parameter, $0 \leq \lambda \leq 1$
  - cf_t is the **collection frequency**, defined to be the total number of occurrences of a term in the collection. (c.f., *df* is document frequency)

- For practical reason, use logs:

$$\log P(q|M_d, C) = \sum_{t \in q} \log \left((1 - \lambda) \frac{tf_{t,d}}{|d|} + \lambda \frac{cf_t}{\sum_t cf_t} \right)$$

Dirichlet Smoothing

- Dirichlet smoothing is the same as Jelinek-Mercer smoothing, picking λ based on document length and a parameter μ as an estimate of the average document length.

$$\lambda = 1 - \frac{\mu}{|d| + \mu}$$

which estimates document score

$$\log P(q|M_d, C) = \sum_{t \in q} \log \frac{tf_{t,d} + \mu \frac{cf_t}{\sum_t cf_t}}{|d| + \mu}$$

Query likelihood (QL) model – Exercise

- Compute the document score using QL model with Dirichlet smoothing.
  Query: "president lincoln",
- text statistics:
  - collection frequencies: $cf_{t1} = 160,000, cf_{t2} = 2,400$
  - Number of word occurrences in the collection: $\sum_{t \in V} cf_t = 10^9$
  - Number of word occurrences in the document: $|d| = 1,800$
  - Average document length: $\mu = 2,000$

Query likelihood (QL) model – Exercise

Query: “president lincoln”,

tf_{t1}	tf_{t2}	MLE	Smoothed score
15	25		
15	1		
15	0		
1	25		
0	25		

Query likelihood (QL) model – Exercise

Query: “president lincoln”,

tf_{t1}	tf_{t2}	MLE	Smoothed score
15	25	-9.06	-10.53
15	1	-12.28	-13.75
15	0	na	-19.10
1	25	-11.77	-12.99
0	25	na	-14.41

Discussion

- Summary
- Questions?