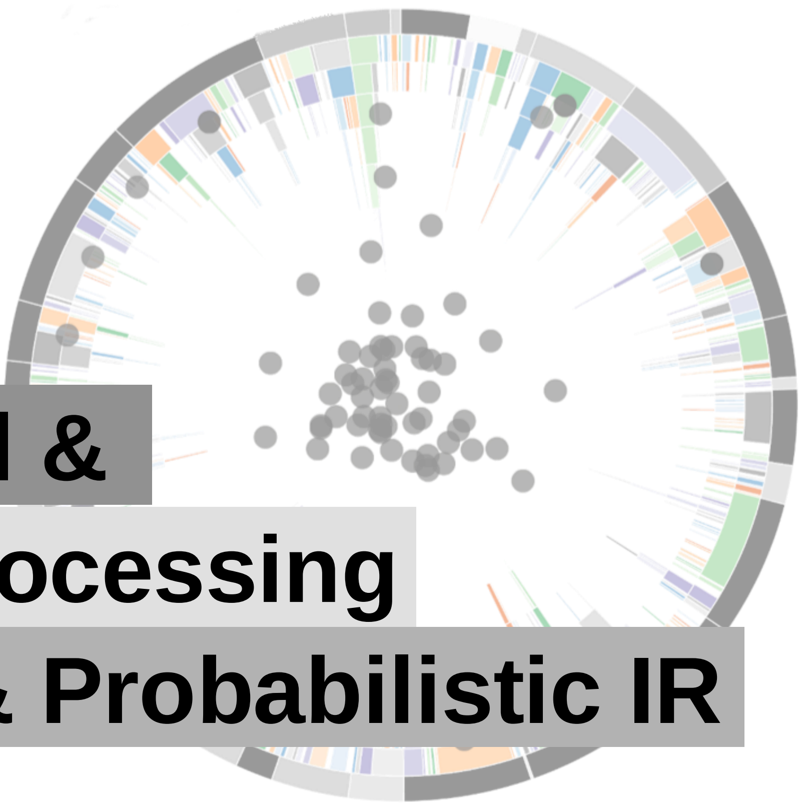
Cluster of Excellence
The Politics of Inequality

Steinbeis Transfer Center
Linguistic Data Analysis

UNIVERSITÄT PASSAU



Information Retrieval & Natural Language Processing
Week 4: Relevance & Probabilistic IR

Annette Hautli-Janisz, Prof. Dr.

21 November 2024

IR & NLP: Course schedule winter 2024/25

	When?	What?
Week 1	17 October 2024	Introduction
Week 2	24 October 2024	Indexing, Boolean IR
Week 3	31 October 2024	--
Week 4	7 November 2024	--
Week 5	14 November 2024	Scoring, term weighting, the vector space model
Week 6	21 November 2024	Relevance and probabilistic IR
Week 7	28 November 2024	Tolerant retrieval and index compression
Week 8	5 December 2024	Evaluation in IR
Week 9	12 December 2024	Distributed word representations for IR

IR & NLP: Course schedule winter 2024/25

	When?	What?
Week 10	19 December 2024	Natural Language Processing
Week 11	9 January 2025	NLP with Python
Week 12	16 January 2025	Personalization in IR systems
Week 13	23 January 2025	AI & ethics
Week 14	30 January 2025	Recap
Week 15	6 February 2025	No class (conference trip)

(also on stud.IP)

Removed: Classification and clustering in IR, Question-answering

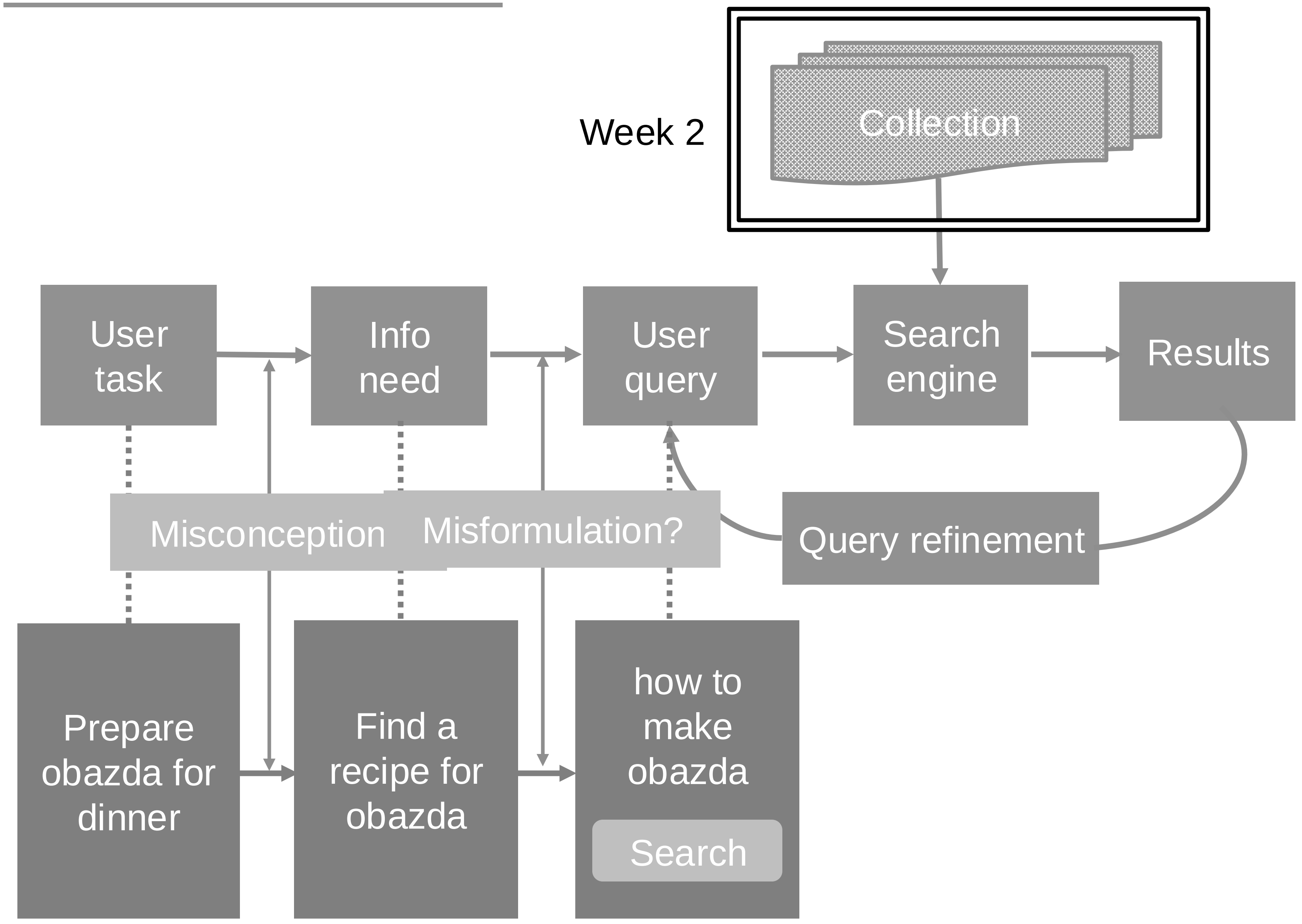
Information Retrieval (IR): Today

(IIR): Manning, Raghavan and Schütze, Introduction to IR, 2008

Chapter 11: Probabilistic information retrieval

Chapter 12: Language models for information retrieval

The classic search model



tf-idf

tf-idf (term frequency - inverse document frequency)

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

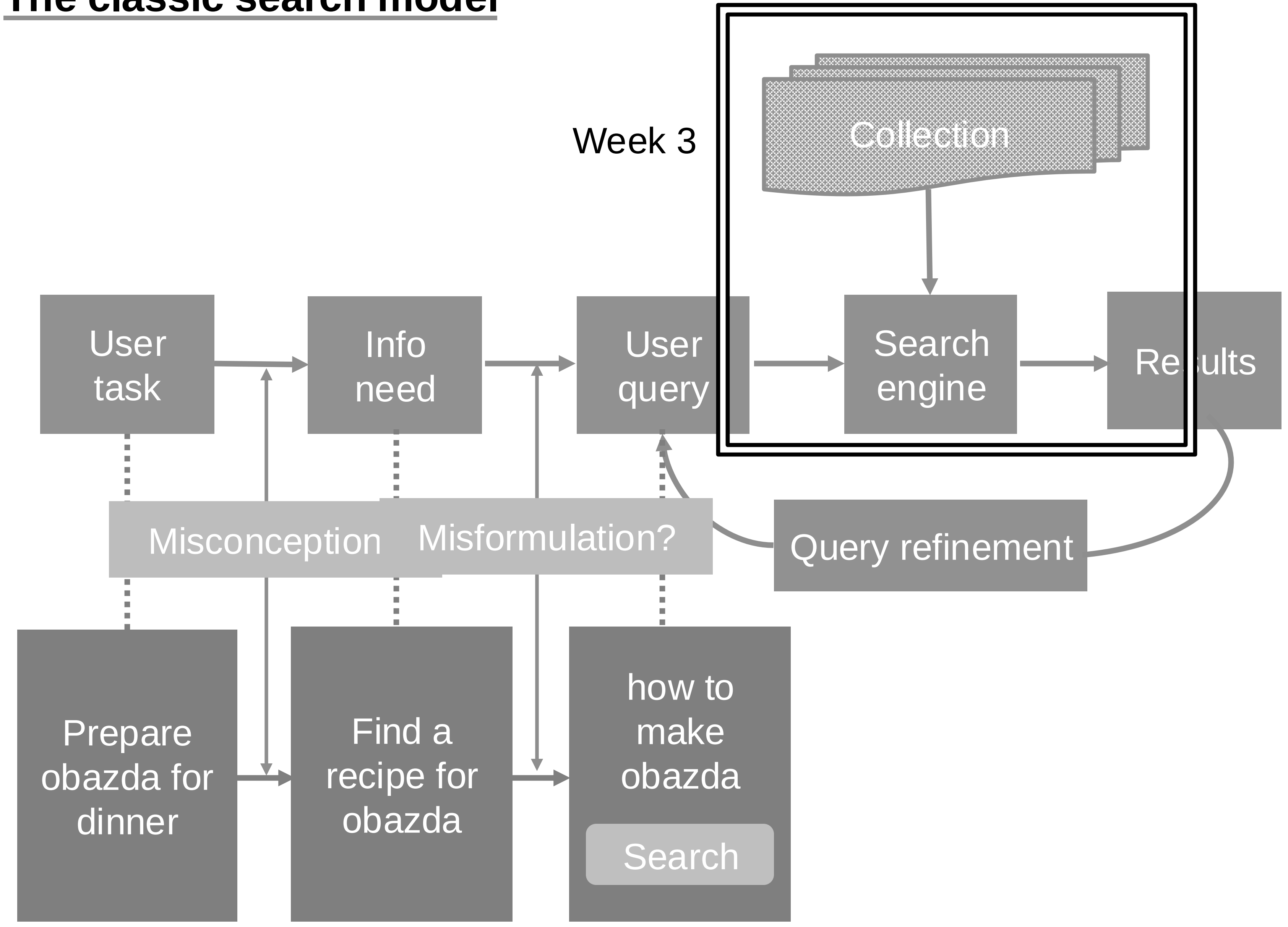
Best-know weighting scheme in information retrieval

- the “-” in tf-idf is a hyphen, not a minus sign!
- alternative names: tf.idf, tf x idf

- increases with the number of occurrences within a document

- increases with the rarity of the term in the collection

The classic search model

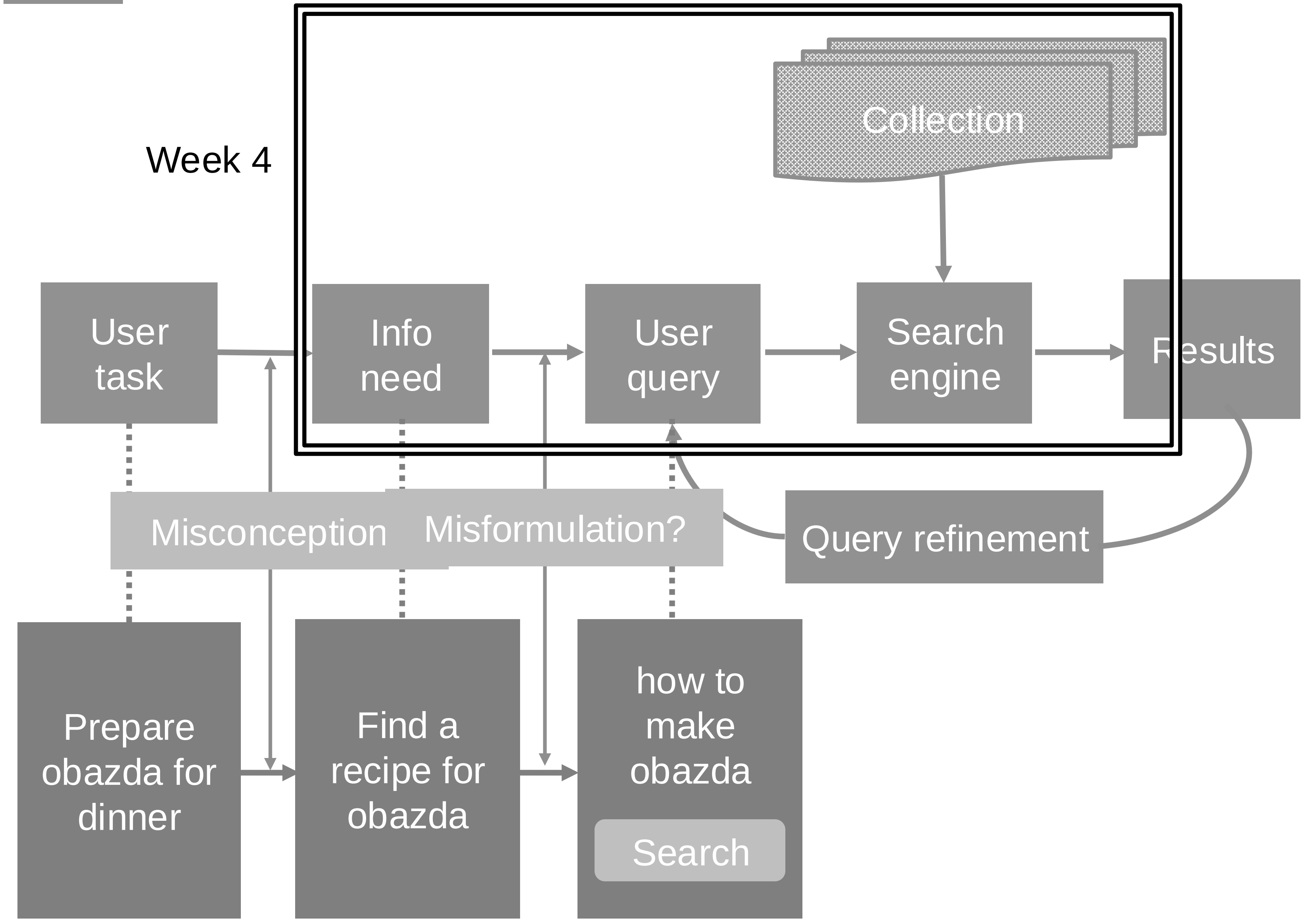


Vector-spaced ranking

Summary:

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top *K* (e.g., *K* = 10) to the user

Today


Collection
Week 4
User task
Info need
User query
Search engine
Results
Misconception
Misformulation?
Query refinement
Prepare obazda for dinner
Find a recipe for obazda
how to make obazda
Search

Past lectures

Problems with Boolean search: Feast or famine.

- Boolean queries often result in either too few (=0) or too many (1000s) results

Vector space models:

- Rank documents according to term occurrence in a document or the collection.

It takes a lot of skill to come up with a query that produces a manageable number of hits

Suggested solution:
Rank documents by goodness – a sort of clever “soft AND”

Probabilistic IR

Probabilistic ranking models aim to model uncertainty about

- information need of the query
- relevance of the document’s content to the query

Probability theory: principled foundation for reasoning under uncertainty.

Relevance as a probability, two assumptions:

1. relevance of a term to a document: the probability of the user satisfaction if that term would be used as a query
2. relevance as a dichotomous variable: the user is either satisfied or not satisfied with the retrieved document

Probabilistic IR

Today:

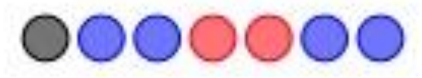
1. Brief recap of probability theory
2. Probability ranking principle
3. The binary independence model
4. The Okapi BM25 weighting scheme

Probability theory

Starting point: the probability of *x* can be seen as the relative frequency of *x* in the long run.

$$\lim_{n \to \infty} \text{rel. frequency of } x = P(x)$$

Quick exercise:



What are the frequencies, relative frequencies and probabilities?

Briefly: Probability theory

Joint probability of two **events A and B occurring independently**:
$P(A, B) = P(A \cap B) = P(A) \times P(B)$

Conditional probability: $P(A|B)$ is **the probability of event A occurring, given that event B occurs**.

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

Probability theory

Some more examples.

1. What’s the joint probability of rolling a ‘5’ twice in a fair 6-sided dice?

2. What’s the joint probability of getting head followed by tail in a coin toss?

3. In a group of 100 sports car buyers, 40 bought alarm systems, 30 purchased bucket seats, and 20 purchased an alarm system and bucket seats. If a car buyer chosen at random bought an alarm system, what is the probability they also bought bucket seats?

Probability theory

Some examples.

4. What is the probability that a randomly selected person is a BSc student, given that they own an iPad?

	Have iPads	Do not have iPads	Total
BSc students	0.41	0.08	0.49
MSc students	0.45	0.06	0.51
Total	0.86	0.14	1

Probability theory

The fundamental relationship between joint and conditional probabilities is given by the *chain rule (*remember, A and B are independent):

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

From that we derive Bayes' Theorem: $P(A|B) = \dfrac{P(B|A) \times P(A)}{P(B)}$

Probability theory

Bayes' Theorem: $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$

P(A) is the *prior* probability: it represents what is originally believed before new evidence is introduced

P(A|B) is the *posterior* probability: it takes this new information into account.

→ This equation can be thought of as a way of updating probabilities: start off with an initial estimate (the prior) and derive the posterior after having seen the evidence B.

Probability of relevance

Recall: relevance as a dichotomous variable, i.e., the user is either satisfied (R=1) or not satisfied (R=0) with the retrieved document.

Model this dichotomy:

- compute probability of relevance *P(R=1)*
- compute probability of non-relevance *P(R=0)*

We compute probabilities by counting relative frequencies → we need to count terms in relevant and non-relevant documents.

We need to know in advance which documents are relevant and which not. How?

Probability of relevance

We can estimate $P(x_t = 1|R = 1)$, the probability of a term t appearing in a document, depending on whether it is relevant or not.

N = total number of docs, df_t is the number of docs that contain term *t*, VR is the set of known relevant documents, VR_t is the subset of this set containing *t.*

$P(x_t = 1|R = 1) = |VR_t|/|VR|$
$P(x_t = 0|R = 0) = (df_t - |VR_t|) / (N - |VR|)$

Those estimates are a quite reasonable start.

Principles of probabilistic retrieval

The documents that are most likely to satisfy the information need should be presented first → 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 (Robertson 1977).

Principles of probabilistic retrieval

The documents that are most likely to satisfy the information need should be presented first → Probability Ranking Principle (PRP).

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**.

Probability Ranking Principle (PRP)

Let D represent a document in the collection.

Need to find P(R=1|D) (the probability that a document D is relevant)

$$P(R=1|D) = \frac{P(D|R=1) \cdot P(R=1)}{P(D)} \quad \text{and } P(R=0|D) = \frac{P(D|R=0) \cdot P(R=0)}{P(D)}$$

P(R=1), P(R=0), respectively, are the prior probabilities of retrieving a (non-)relevant document at random.

P(D|R=1), P(D|R=0) are the probabilities that if a (non-)relevant document is retrieved, it is document D.

Probability Ranking Principle (PRP)

Measuring success:

In the simplest case of the PRP: no retrieval costs and other utility concerns.
→ loss of a point for either returning a nonrelevant document or failing to return a relevant one (1/0 loss)

→ The goal is to return the best possible results as the top *k* documents, for any value of *k* the user chooses to examine.

→ The PRP then says to simply rank all documents in decreasing order o f $P(R=1|d,q)$

Probability Ranking Principle (PRP)

If a set instead of an ordering is to be returned → Bayes optimal decision rule: the decision that minimizes the risk of loss – simply return documents that are more likely relevant than nonrelevant:

d is relevant iff $P(R=1|d,q) > P(R = 0, d,q)$.

The PRP is optimal, in the sense that it minimizes the expected loss under 1/0 loss.

Requirement: all probabilities are known correctly (never the case in practice).

The binary independence model

The binary independence model (BIM) is the model that has traditionally been used with the PRP.

Three assumptions to make the function *P(R=1|D)* practical:

1. Binary in the sense of Boolean: Documents and queries are represented as binary term incidence vectors.

2. Terms are modeled as occurring in documents independently (the '*naïve*' in the *Naïve* Bayes model)

3. The relevance of each document is independent of the relevance of other documents.

The binary independence model

To make a probabilistic retrieval strategy precise, we need to estimate how terms in documents contribute to relevance.

Include information

- term frequency
- document frequency
- document length
- etc...

The binary independence model

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

and

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

$P(R=1|\vec{q})$, $P(R=0|\vec{q})$, respectively, are the prior probabilities of retrieving a (non-)relevant document at random for a query $\vec{q}$.

$P(\vec{x}|R=1,\vec{q})$, $P(\vec{x}|R=0,\vec{q})$ are the probabilities that if a (non-)relevant document is retrieved, then that document's representation is $\vec{x}$.

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

Deriving the ranking function

Given the query $\vec{q}$, we wish to order returned documents by descending P(R=1| $\vec{x},\vec{q}$). Under BIM, this is modeled as *P(R=1|$\vec{x},\vec{q}$).*

See the derivations in Chapter 11.3 in IIR for the *retrieval status value* (RSV).

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

u_t is the probability of a term appearing in a non-relevant document
p_t is the probability of a term appearing in a relevant document

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

Deriving the ranking function

The c_t quantities function as term weights in the model, and the document score for a query is

$$RSV_d = \sum_{x_t=q_t=1} c_t$$

Probabilistic retrieval model

1. We run the system to retrieve documents w.r.t. a query (*first pass*)

2. We present the results to a user who judges which are relevant and which are not (*relevance feedback*)

3. We combine human judgement + term frequency statistics to compute *P(R=1)* (probability of relevance) and *P(R=0)* (probability of non-relevance) for the documents.

4. We run the system again to retrieve documents, using the above probabilities (*second pass*)

5. The system displays a revised set of retrieval results.

Probabilistic assumptions

In probabilistic IR, assumptions replace human judgement about relevance.

These assumptions are **empirical** → IR is empirical.

- Assumption 1: relevant documents are a very small percentage of the collection → approximate statistics for non-relevant documents by statistics from the whole collection.

- Assumption 2: the probability of a query term appearing in a relevant document is the same for all terms in a given query.

Okapi BM25

Probabilistic model sensitive to term frequency and document length (without too many additional parameters) (Spärck Jones et al. 2000).

We won’t go through the full theory behind it.

The simplest score for a document’s *retrieval status value (RVS):*

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

Okapi BM25

Version 2: factoring in term frequency and document length.

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

L_d = document length
L_{avg} = average document length in the collection
k_1 = positive tuning parameter that calibrates the document term frequency scaling (if 0, binary model, large values correspond to raw frequencies)
b = 1 corresponds to fully scaling the term weight by document length, b = 0 corresponds to no length normalization

Okapi BM25

Version 3: weighting query terms for long queries (unnecessary for short queries).

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

k_3 = another positive tuning parameter, calibrates term frequency scaling of the query

How does k_3 impact the RSV?

Okapi BM25

Version 4: include relevance judgements, if unavailable, use regular idf.

Replace idf

$$\begin{aligned}\underline{RSV_d} \quad \underline{=} \quad & \sum_{t \in q} \log \left[\left[\frac{(|VR_t| + \frac{1}{2}) / (|VNR_t| + \frac{1}{2})}{(\mathrm{df}_t - |VR_t| + \frac{1}{2}) / (N - \mathrm{df}_t - |VR| + |VR_t| + \frac{1}{2})} \right] \right. \\ & \left. \times \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b(L_d / L_{ave})) + \mathrm{tf}_{td}} \times \frac{(k_3 + 1)\mathrm{tf}_{tq}}{k_3 + \mathrm{tf}_{tq}} \right]\end{aligned}$$

Appraisal of probabilistic models

Probabilistic IR has neat ideas, but the methods perform weakly.

- Approximating the needed probabilities is possible, but it requires some major assumptions.
- Perhaps the severity of the modeling assumptions makes achieving good performance difficult.
- General problem: either partial relevance information or inferior models.
- Best Match 25 (BM25) a.k.a. Okapi: very good performance (Robertson et al. 1994)

Ranking with language models (LMs)

LMs can be seen as an extension of mainstream probabilistic models for IR:

- Mainstream probabilistic models: given a query Q, estimate the probability of the relevance of document D with respect to that query.

  $P(R|D,Q)$

- Language models: given a document D, build a language model from each document and then estimate the probability of a query having been generated from that document

  $P(Q|D)$

What are language models?

What does it mean that a document model generates a query?

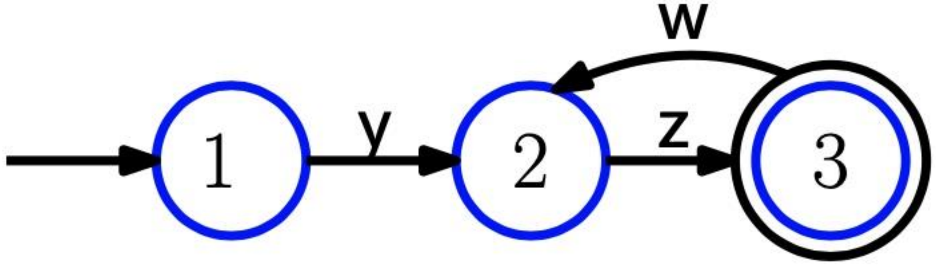
A generative model of language can recognize or generate strings in that language.

Simple language models: finite state automata.

What are language models?

FSA are a quintuple ($\sum$, S, s_0, δ, F)

- $\sum$: input alphabet
- S: set of states
- s_0: initial state
- δ: transition function: $\delta(1,y) = 2$, $\delta(2,z) = 3$, $\delta(3,w) = 2$
- F: Final states

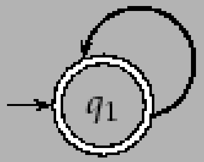


What are language models?

Instead of letters on the transitions, use words for the language models.

Add a probability distribution over generating different terms.

Voilà: a language model.



$P(\text{STOP}|q_1) = 0.2$

the	0.2
a	0.1
frog	0.01
toad	0.01
said	0.03
likes	0.02
that	0.04
…	…

► **Figure 12.2** A one-state finite automaton that acts as a unigram language model. We show a partial specification of the state emission probabilities.

What are language models?

After generating each word, we decide whether to stop or loop around and produce another word.

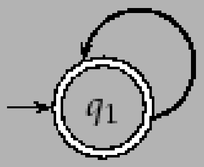
→ The model also requires a probability of stopping in the finishing state.

→ Such a model places a probability distribution over any sequence of words.

→ By construction, it also provides a model for generating text according to its distribution.

What are language models?

What’s the probability of the sequence ‘frog said that toad likes frog’ using the below finite state automaton. (Also take into account the decision whether to stop or continue at each step).



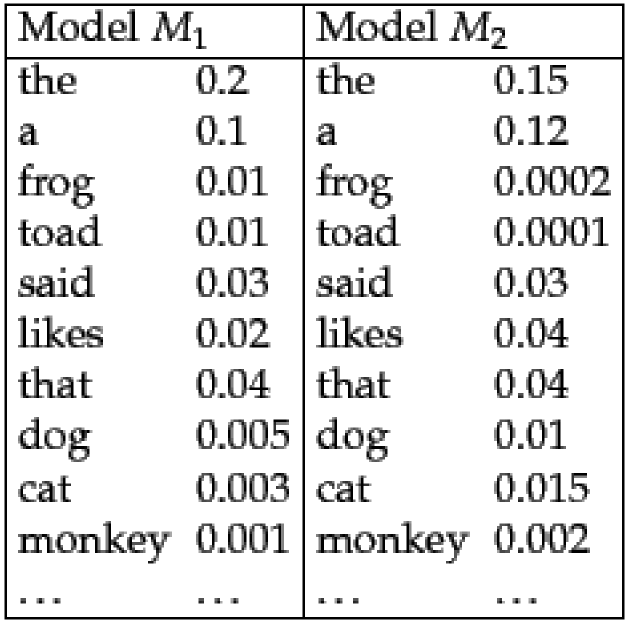
$P(\text{STOP}|q_1) = 0.2$

the	0.2
a	0.1
frog	0.01
toad	0.01
said	0.03
likes	0.02
that	0.04
…	…

► **Figure 12.2** A one-state finite automaton that acts as a unigram language model. We show a partial specification of the state emission probabilities.

What are language models?

Suppose now we have two language models M_1 and M_2. The LM that gives the higher probability to the sequence ‘frog said that toad likes frog’ is more likely to have generated the term sequence. (This time we omit stop probabilities.)



Model M_1		Model M_2	
the	0.2	the	0.15
a	0.1	a	0.12
frog	0.01	frog	0.0002
toad	0.01	toad	0.0001
said	0.03	said	0.03
likes	0.02	likes	0.04
that	0.04	that	0.04
dog	0.005	dog	0.01
cat	0.003	cat	0.015
monkey	0.001	monkey	0.002
...	...	...	...

Ranking with language models (LMs)

Let t denote a term in a query Q (t ∈ Q)

Let P(t|D) denote the probability of term t in document D.

Then, the probability of query Q having been generated from document D is:

$$P(Q|D) = \prod_{t \in Q} P(t|D)$$

How to estimate the individual term probabilities?

$$P(Q|D) = \prod_{t \in Q} P(t|D)$$

P(t|D) can be estimated from the term frequencies:

$$P(t|D) = \frac{tf_{t,D}}{L_D}$$

$tf_{t,D}$ is the term frequency in the document
L_D is the document length (number of terms in the document)

Maximum Likelihood Estimation (MLE) = $\frac{\text{how often a term occurs in a document}}{\text{total number of terms in a document}}$

How to estimate the individual term probabilities?

In practice, a query term may be missing from a document.

→ its probability will be zero

→ the final probability for that document will be zero

Smoothing

- we assign some probability to missing terms
- Various smoothing techniques:
    - Jelinek-Mercer
    - Dirichlet
    - Good-Turning
    - Laplace (a.k.a *add-one)*
    - *…*

Jelinek-Mercer (JM) smoothing

Using JM smoothing, the probability of a term t in document D is:

$$P(t|D) = \lambda \frac{tf_{t,D}}{L_D} + (1 - \lambda) \frac{tf_{t,C}}{L_C}$$

- L_D: number of terms in D (document length)
- $tf_{t,C}$: number of times term t occurs in the collection
- L_C: number of terms in the collection (collection length)
- λ: Smoothing parameter that can be set manually (experimentally or automatically), $0 \le \lambda \le 1$ (recommended value from Zhai and Lafferty 2002: $\lambda \approx 0.6 - 0.7$

LM with JM smoothing

Using JM smoothing, the probability of a query being generated from a document is:

$$P(Q|D) = \prod_{t \in Q} P(t|D) = \prod_{t\ inQ} \lambda \frac{tf_{t,D}}{L_D} + (1-\lambda)\frac{tf_{t,C}}{L_C}$$

$\frac{tf_{t,D}}{L_D}$ can be seen as the document model (M_D)

$\frac{tf_{t,C}}{L_C}$ can be seen as a collection model (M_c)

LM with JM smoothing

Example

We have a collection that contains two documents:

D1 = `Xyzzy reports a profit but revenue is down`
D2 = `Quorus narrows quarter loss but revenue decreases further`

Our query Q is: `revenue down`

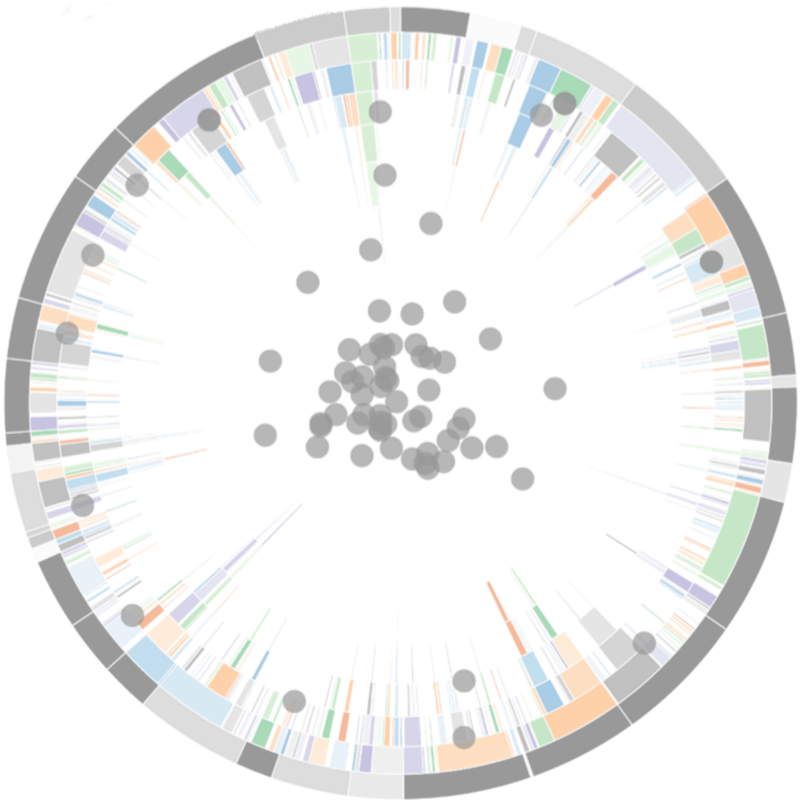
Task: Rank the documents with respect to the query using a LM with JM smoothing (λ = 0.5).

Cluster of Excellence
The Politics of Inequality

Steinbeis Transfer Center
Linguistic Data Analysis

UNIVERSITÄT
PASSAU

Thank you.
Questions?
Comments?



Annette Hautli-Janisz, Prof. Dr.
cornlp-teaching@uni-passau.de