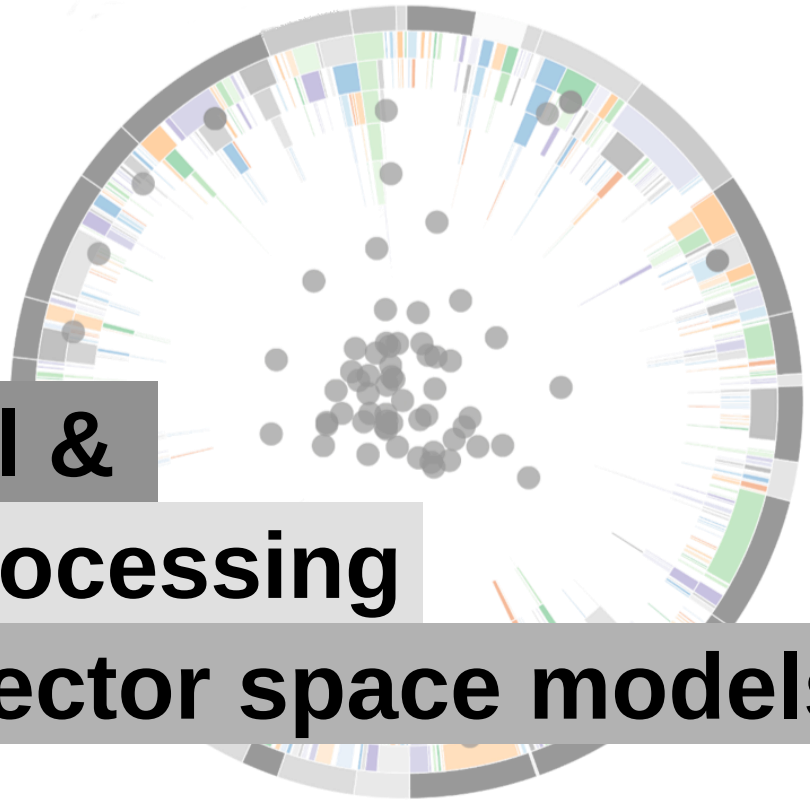
Cluster of Excellence
The Politics of Inequality

Steinbeis Transfer Center
Linguistic Data Analysis

UNIVERSITÄT PASSAU



Information Retrieval & Natural Language Processing
Week 3: Scoring & vector space models

Annette Hautli-Janisz, Prof. Dr.

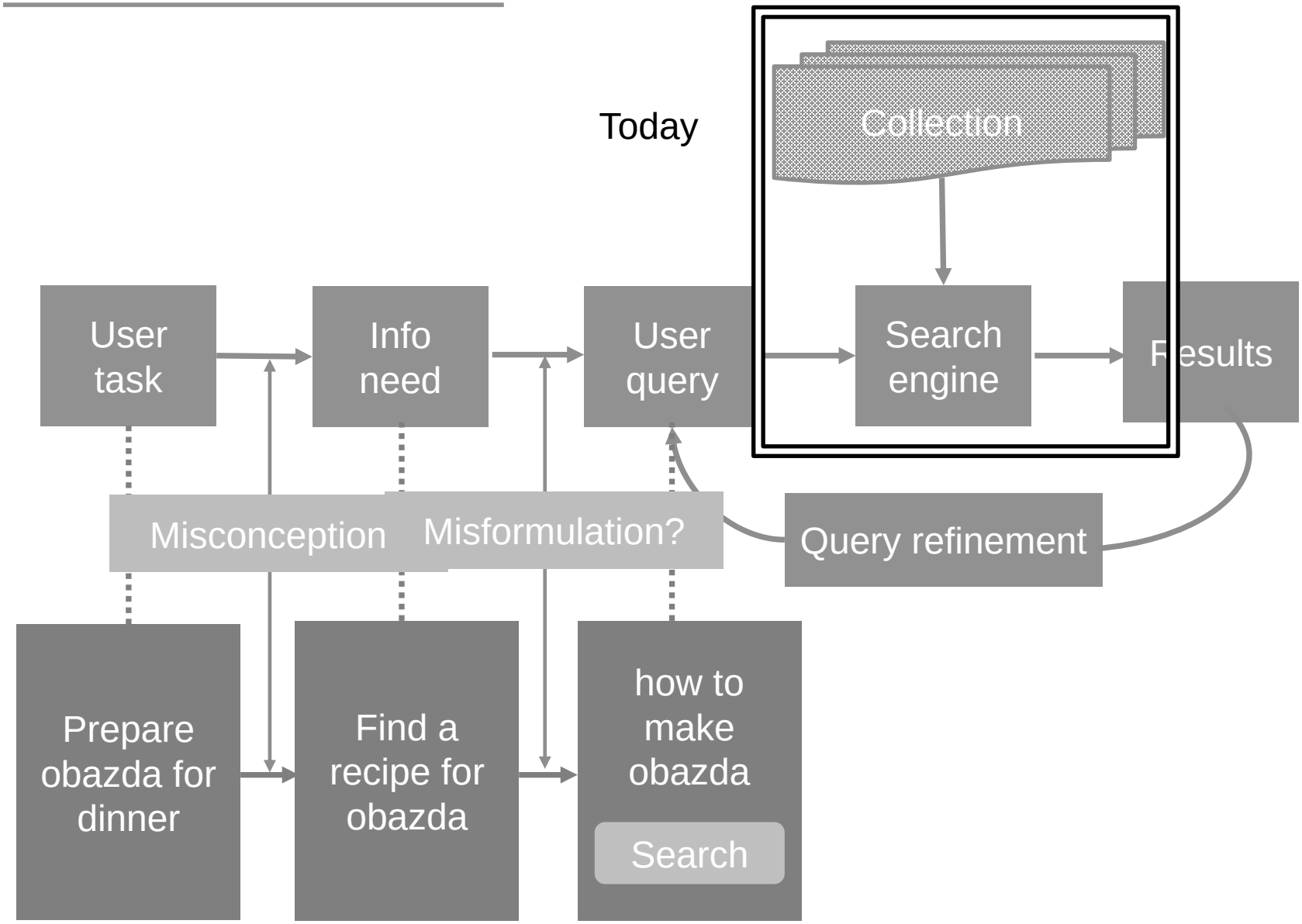
14 November 2024

Information Retrieval (IR): Today

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

Chapter 6: Scoring, term weighting and the vector space model

The classic search model



From Boolean to ranked retrieval

So far: Boolean queries → Documents either match or don't.

Good for

- expert users (precise understanding of the need and collection)
- applications (easy consumption of 1000s of results)

Not good for

- most users (unwilling/incapable to write Boolean queries)
- web search

Search engine: rank-order the documents matching a query.

Ranked retrieval

The system returns an ordering of the top documents in the collection.

Typically, free text queries: more than one word in natural language (no operators etc.)

→ If search algorithm works, large sets are not a problem anymore (or at least, should not be):

- rank result according to usefulness to the user
- only top *k* results are shown

Ranked retrieval

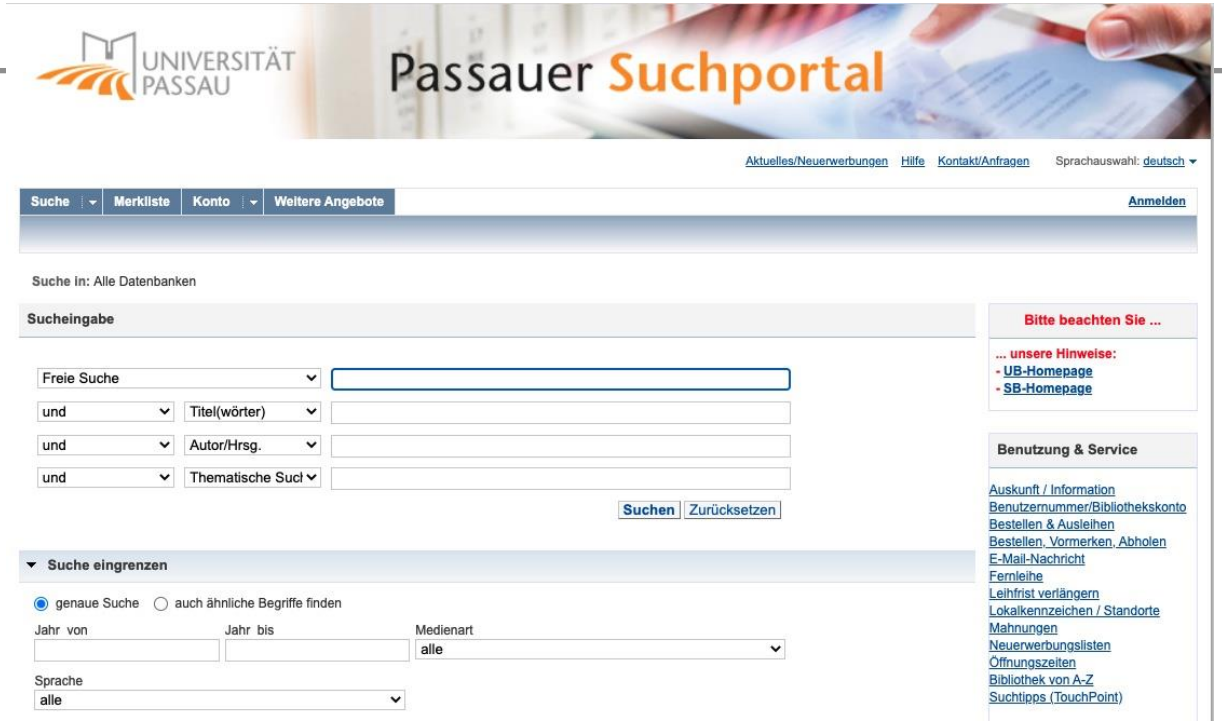
So far: Document = sequence of terms

But: Most documents have additional structure.

One option: Parametric and zone indexes – they make use of this additional structure.

The parametric index

A parametric index is an index that allows retrieval of documents based on the values of parameters. It serves as an extension of the collection's word index.



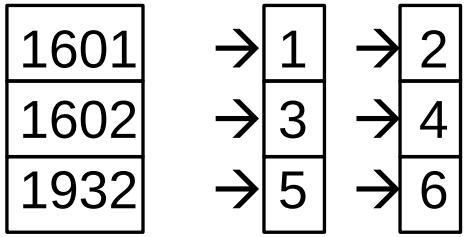
The parametric index

Make use of metadata associated with each document.

Fields, e.g. date of creation, with a (finite) set of possible *values, e.g*. the set of all dates of authorship.

Each field has one parametric index.

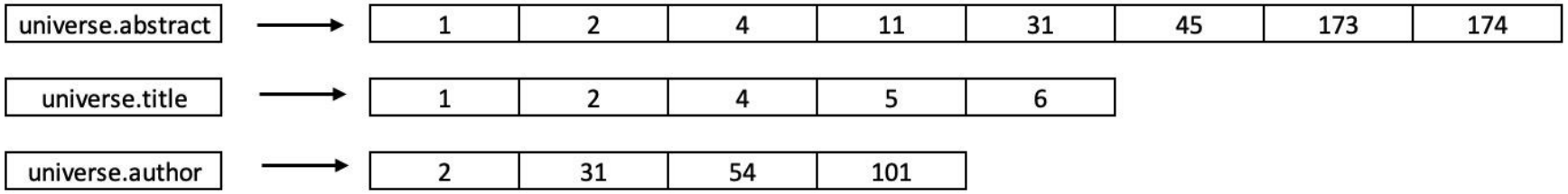
Date of publication index:



Allows us to to select only the documents matching a data specified in the query.

Zone indexes

Zones are similar to fields, except the contents of a zone can be arbitrary free text. They are encoded as extensions of dictionary entries.



Ranked retrieval without zones

The system returns an ordering over the top documents in the collection for a query.

Queries without any operators, formulated in natural language.

Scoring is the basis of ranked retrieval:

- assign a score to a query/document pair, based on the weight of the query in the document.
- aka: the higher the score, the more relevant the document is to the query.

A first take at ranking…

Using the Jaccard coefficient

Measuring the similarity of two sets A (the query) and B (the document) with a number between 0 and 1.

$J(A, B) = |A \cap B| / |A \cup B|$
$J(A, A) = 1$
$J(A, B) = 0$ if $A \cap B = 0$

A and B do not have to have the same size.

A first take at ranking…

Using the Jaccard coefficient

What is the query-document Jaccard score for each of the two documents below?

Query = ‘ides of march’
doc1 = ‘caesar died in march’
doc2 = ‘the long march’

What are the issues?

Recall: Term-document incidence matrix

	The Hitch-hiker's Guide to the Galaxy	The Restau-rant at the End of the Universe	Life, the Universe and Every-thing	So Long, and Thanks for all the Fish	Mostly Harmless	And Another Thing...
Arthur	1	1	0	0	0	1
Ford	1	1	0	1	0	0
Zaphod	1	1	0	1	1	1
Trillian	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
Marvin	1	0	1	1	1	1
Random	1	0	1	1	1	0

Possibility 1: Each document is represented as a binary vector [0, 1] → binary matrix.

Recall: Term-document incidence matrix

	The Hitch-hiker's Guide to the Galaxy	The Restau-rant at the End of the Universe	Life, the Universe and Every-thing	So Long, and Thanks for all the Fish	Mostly Harmless	And Another Thing...
Arthur	157	73	0	0	0	5
Ford	4	157	0	1	0	0
Zaphod	232	227	0	2	1	1
Trillian	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
Marvin	2	0	3	5	5	1
Random	2	0	1	1	1	0

Possibility 2: Number of term occurrences in a document (count vectors are columns in the table) → count matrix.

Term frequency

Term frequency $tf_{t,d}$ of term *t* in document *d* is the number of times that t occurs in *d*.

But: absolute frequencies are not what we want.

- Not all terms are equally important.
- A document with 10 occurrences is more relevant than a document with one occurrence.
- But not 10 times more relevant.

Relevance does not increase proportionally with term frequency.

→ Reduce the effect of term frequency in determining relevant docs.

Term frequency

Choose log-frequency weighting of term *t* in document *d.*

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

(tf = 0 → w = 0; tf = 1 → w = 1; tf = 2 → w = 1,3; tf = 1000 → w = 4)

Term frequency

Score for a document-query pair: sum over terms t in both query and document.

$$\text{score}(q,d) = \sum_{t \in q \cap d} (1 + \log \text{tf}_{t,d})$$

Example:
q = ‘ides of march’, d1 = ‘caesar died in march’, d2 = ‘the long march’

What is score(q,d1) and score(q,d2)?

Term frequency

Occurrence of each term (`term frequency') is central.

E.g., *John is quicker than Mary* has the same vector as *Mary is quicker than John.*

But the assumption is: two documents with similar bag of words representations are similar in content.

Collection frequency versus document frequency

We need to scale down the weight of *tf* in determining relevant documents.

In other words, we need a weight for each term in the document which reflects how much this term helps in distinguishing a document from all other documents in the collection.

Two possibilities:

- Collection frequency (cf): the total number of occurrences of term *t* in the collection.
- Document frequency (df): the number of documents in the collection that contain a term *t.*

Collection frequency versus document frequency

Example:

word	cf	df
try	10422	8760
insurance	10440	3997

Should we prefer collection frequency (cf) or document frequency (df)? Why?

Inverse document frequency (idf)

Document frequency df_t is an inverse measure of the informativeness of *t*

→ the higher the document frequency (*df)*, the lower the informativeness of term *t.*

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

N is the number of documents in the collection.

Log $\left(\frac{N}{df_t}\right)$ reduces the effect of the *idf* weight on the overall term weight.

Inverse document frequency (idf)

Example: N = 1 Million, $idf_t = \log_{10}(\frac{N}{df_t})$

term	df_t	idf_t
Calpurnia	1	6
animal	100	4
Sunday	1.000	3
fly	10.000	2
under	100.000	1
the	1.000.000	0

tf-idf

TF-IDF (term frequency - inverse document frequency)

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

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

tf-idf

In other words…

- term frequency: how many times a term occurs in a document. (Assumption: a document containing many times a given word, is likely to be about that word)
- inverse document frequency: how many documents in the collection contain a term. (Assumption: if a term occurs in many documents, it is not very discriminative.)

A survey conducted by Breitinger et al. in 2015 shows that 83% of text-based recommender systems in digital libraries use tf–idf.

Score for a document given a query

score(q,d) = $\sum_{t \in q}$ tf−idf$_{t,d}$

The sum of tf-idf weights for all terms in the query q that are matching in document d.

Different weighting schemes for tf and idf available. We use here:

$$w_{t,d} = (1 + \log_{10} tf_{t,d}) \times \log_{10} \left(\frac{N}{df_t}\right)$$

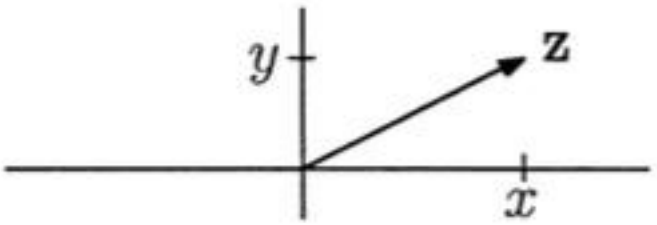
Term-document weight matrix

	The Hitch-hiker's Guide to the Galaxy	The Restau-rant at the End of the Universe	Life, the Universe and Every-thing	So Long, and Thanks for all the Fish	Mostly Harmless	And Another Thing...
Arthur	5.25	3.18	0	0	0	0.35
Ford	1.21	6.1	0	1	0	0
Zaphod	8.59	2.54	0	1.51	0.25	0.35
Trillian	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
Marvin	1.51	0	1.9	0.12	5.25	0.88
Random	1.37	0	0.11	4.15	0.25	0

Possibility 3: Each document is represented by a real-valued vector of tf-idf weights → weight matrix

Terms and documents as vectors

This is a vector ($\vec{V}$(z)) defined over the coordinate axes x and y.



In IR, vectors are used to represent terms and documents.

Each document is a vector defined over co-ordinate axes; the co-ordinate axes are the terms contained in the document.

High-dimensional: tens of millions of dimensions when you apply this to a web search engine (but: very sparse vectors - most entries are zero).

Documents as vectors

$\vec{V}(d_1) = (t_1, t_2, t_3, t_4)$
$\vec{V}(d_2) = (t_1, t_2, t_3, t_4)$

- the vector value is derived from the term weights, e.g., the tf-idf weights
- the same term may receive different weights in different documents

→ Comparing documents and the query by comparing their vectors.

Standard mathematics: vector cosine similarity

Documents as vectors

tf-idf weights are used to characterize the documents.

$$sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| \cdot |\vec{V}(d_2)|}$$

$\vec{V}(d_1) \cdot \vec{V}(d_2)$: **dot product** of $\vec{V}(d_1)$ and $\vec{V}(d_2)$

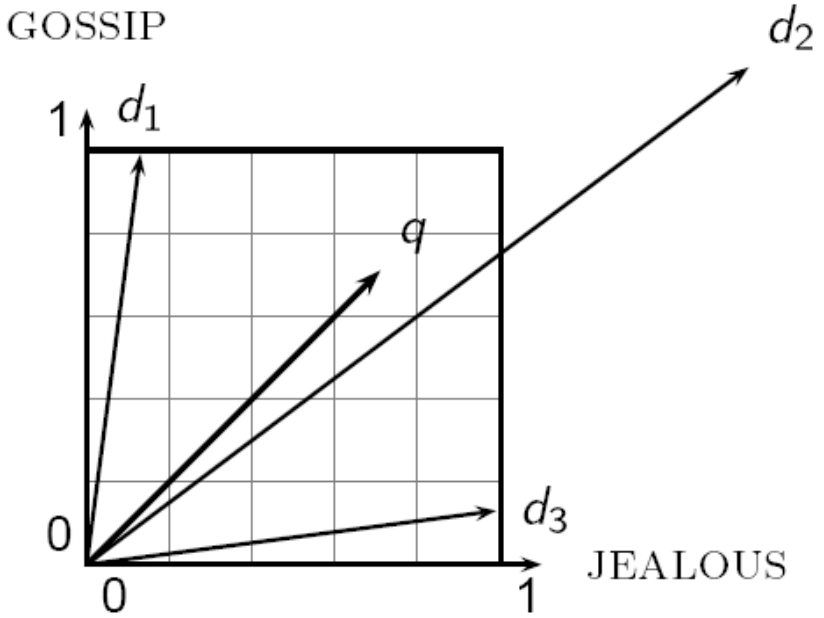
$|\vec{V}(d_1)| \cdot |\vec{V}(d_2)|$: product of their **Euclidean lengths**

The denominator normalizes vectors of different lengths → allows us to compare documents of different lengths.

Vector space proximity

Why normalizing for length?

Euclidean distance is large for vectors of different length.



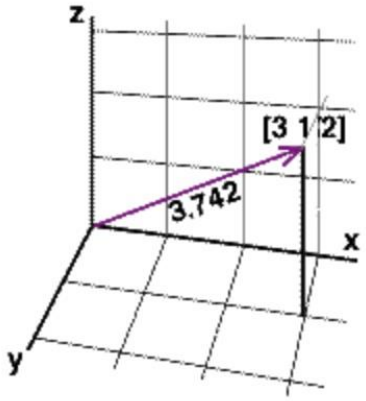
Length normalization

A vector is normalized by dividing each of its components by its length.

$$\left\|\vec{x}\right\|_2 = \sqrt{\sum_i x_i^2}$$

($\vec{V}$(x)) = [3 1 2]

|x| = $\sqrt{9 + 1 + 4}$ = 3,724

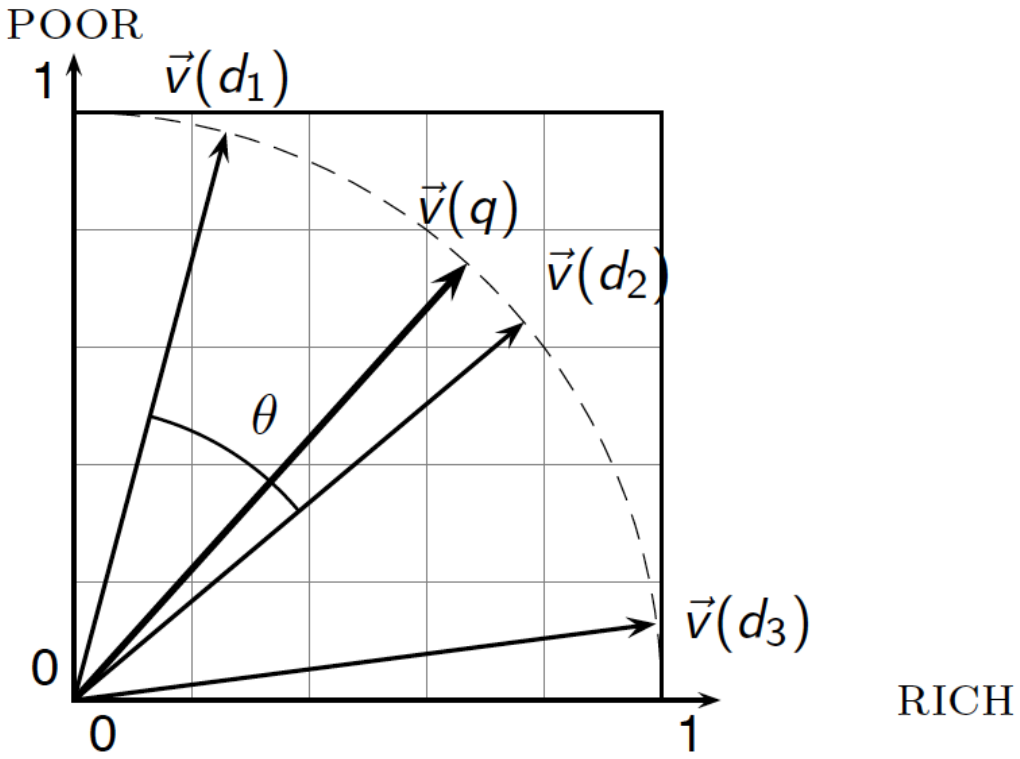


→ Long and short documents now have comparable weights

Length normalization

For length-normalized vectors, the cosine similarity is simply the dot product.

Find more like this feature in search engines.



Queries as vectors

Instead of comparing $\vec{V}(d1)$ to $\vec{V}(d2)$, we can compare $\vec{V}(d)$ to $\vec{V}(q)$, where $\vec{V}(d)$ is every document in our index, and $\vec{V}(q)$ is a query.

$$sim(q, d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)| \cdot |\vec{V}(d)|}$$

Given a query *q*, all the documents in the collection can be ranked according to their similarity score *sim*(*q*,*d*). The higher the similarity, the more relevant the document to the query.

Example

Compute the cosine similarities between the three documents.

term	d1	d2	d3
Arthur	115	58	20
Ford	10	7	11
Zaphod	2	0	6
Trillian	0	0	38

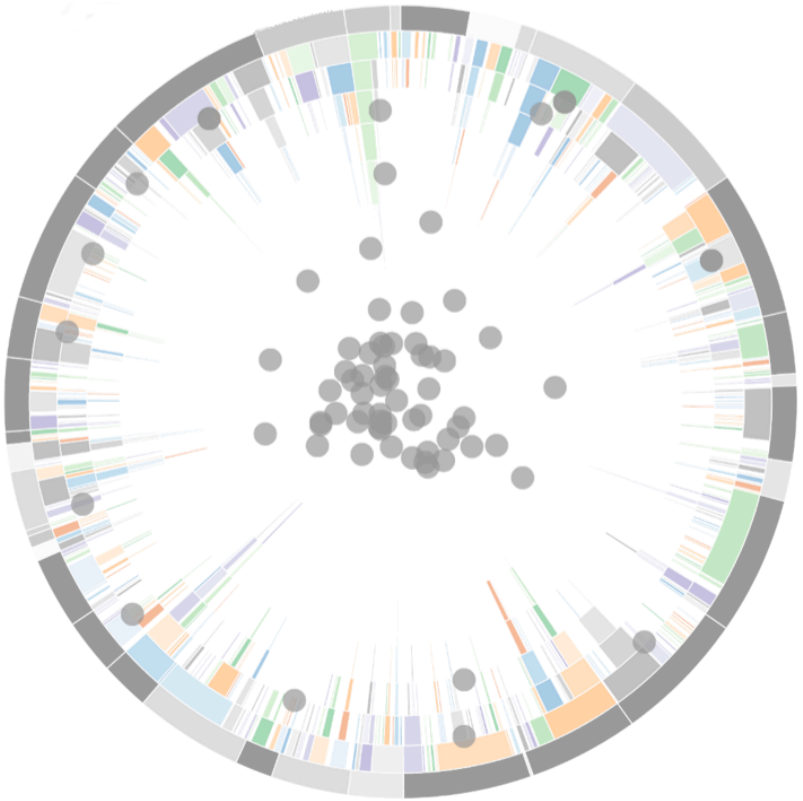
For simplification: Only use $tf_{t,d}$ with log-frequency weighting.

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