George Mason University | Computational and Data Sciences

CSI 695: Scientific Databases

Fall Term 2017

Lecture 5: Introduction to Searching in Scientific databases

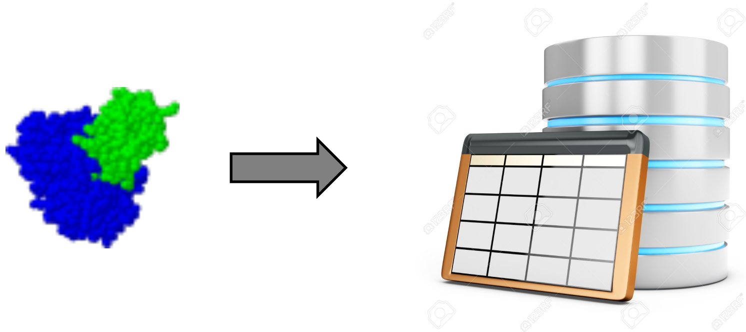
Lectures: Prof. Dr. Matthias Renz

Exercises: TBA

Recap: Searching in Scientific Databases: Introduction

- Managing scientific data usually requires methods that go beyond the capabilities of standard database management systems.
- Scientific data often involves complex structured data not well supported by the table schema used in standard databases.
- Exact match queries as provided in standard dabase management systems often do not suffice for searching in scientific data, we need something different!!!

Example:
How would you store/manage molecules in a standard relational database ?

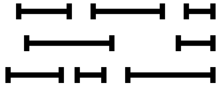
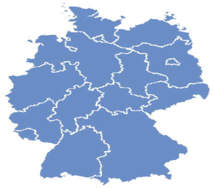
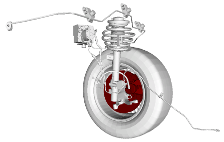


Think about it ...

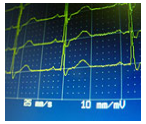
Recap: Searching in Scientific Databases: Introduction

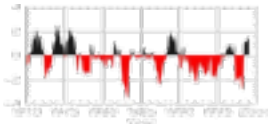
- Scientific data often consists of complex stuctured data including spatial, temporal, spatio-temporal, and multi-media data?

  - Spatial data: 1D, 2D, 3D

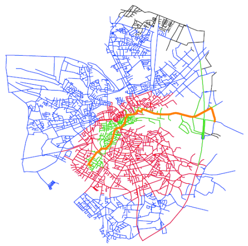




  - Temporal data: time series

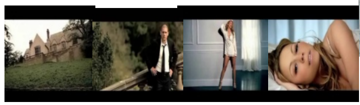
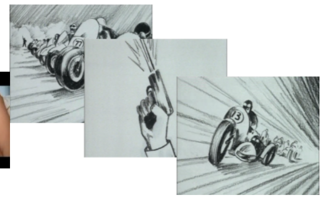




  - Spatio-temporal data:
    - objects moving in a given space

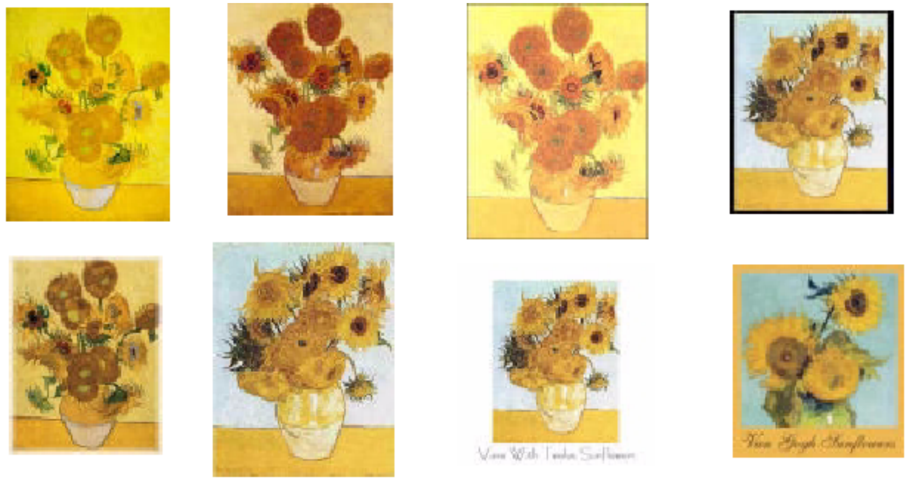


  - Multi-media data:
    - audio sequences, video sequences

Recap: Searching in Scientific Databases: Introduction

- A motivating example
  - Given an archive with 2,000,000 images (2D objects)
  - Is a given image included in that archive?



  - Challenges
    - “included” does not necessarily mean “identical binary representation!
    - Images may vary in
    - Size (scaling, resolution)
    - Perspective (reflection, …)
    - Coloring, shading
    - Clipping, cutting
    - Add-ons (border, annotation, …)

Recap: Searching in Scientific Databases: Introduction

- Instead of searching for

    exact matches (supported by standard databases (relational-, object-relational DBMS)

    we need to search for

    “similar” objects! → **new concepts for managing data required (non-standard databases)**

Recap: Searching in Scientific Databases: Introduction

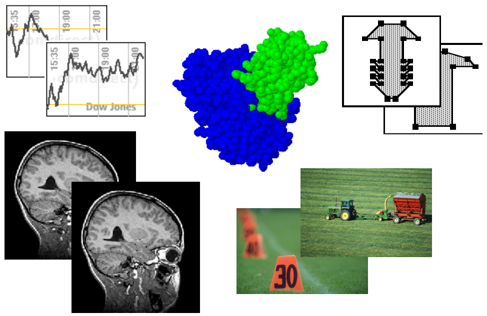
- Searching:
  - General problems when searching for similar objects
    - Informal level
      - Similarity depends on the application
        - Searching for images showing “sunset” => color is important
        - Searching for images showing “animals” => shape is important
      - Similarity depends on the user’s notion
    - Formal level
      - How are the objects represented?
      - How can the similarity between objects be modeled?
    - Pragmatic level
      - Efficient algorithm for computing the similarity
      - Efficient algorithm for searching in a large disc-resident database

Recap: Searching in Scientific Databases: Introduction

- Searching:
  - Here, we focus on the sub-problem
    - Efficiently searching for similar objects in a large database and to some extend on
    - Efficiently computing the similarity between objects
  - We assume a very common model of similarity: ***Feature-based similarity***
- Feature-based similarity
  - How can we model the similarity between complex objects like images, 3D objects, video sequences, etc.?
  - Considerations
    - Efficiency: Model should allow efficient query processing => use of index structures should be possible
    - Generality: Avoid the necessity to develop algorithms and index structures for each application separately but develop a general way to model similarity
      - We have indexes for
      - Spatial data and multi-dimensional vectors
      - General metric data (objects with a metric distance function)

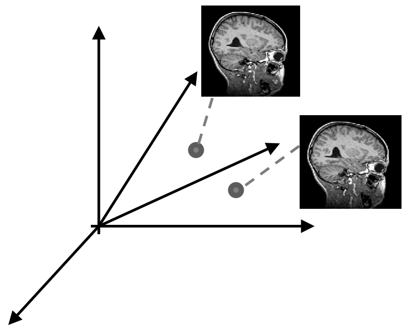
Recap: Searching in Scientific Databases: Introduction

- Feature-based similarity (cont.)



Feature Transformation

Histogramms
Shape descriptors
Fourier coefficients
etc.



Application Domains (Object Space) | Feature (Vector) Space

- Objects from real world are transformed into multi-dimensional feature vector points
  1. Identify a set (sequence) of (numerical) features from objects that best describe the objects
  2. Build a multidimensional point vector from the set (sequence) of features
  3. Manage the point vectors where each point vector has a link to the detailed object descriptions
  4. (Object) point vectors are efficiently organized (managed) using appropriate index structures

Outline

- Searching in Scientific Databases: Introduction
- Feature spaces and proximity measures
- Feature transformation for text data
- Algorithmic Paradigms for Similarity Query Processing

Feature spaces and proximity measures

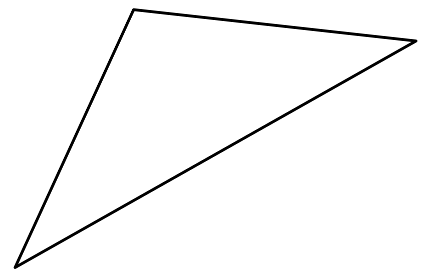
Feature space

- Intuitively: a domain with a distance function
- Formally: feature space **F** = (*Dom*, *dist*):
  - *Dom* is a set of attributes / features
  - *dist*: a numerical measure of the degree to which the two compared objects differ
    - $dist : Dom \times Dom \rightarrow R^{+}_{0}$
- For all *x*, *y* in *Dom, x≠y*, *dist* is required to satisfy the following properties:
  - $dist(x,y) > 0$ (non-negativity)
  - $dist(x,x) = 0$ (reflexivity)

Feature spaces and proximity measures

Metric space

- Formally: Metric space $M = \{Dom, dist\}$:
  - M is a feature space
    - i.e, *dist*(*x*,*y*) > 0 (non-negativity) and,
    - *dist*(*x*,*x*) = 0 (reflexivity)
  - $dist(x, y) = 0 \Rightarrow x = y$ (strictness)
  - $\forall x, y \in Dom: dist(x, y) = dist(y, x)$ *(symmetry)*
  - $\forall x,y,z \in Dom : dist(x,z) \leq dist(x,y) + dist(y,z)$
    (triangle inequality)



- Measures that satisfy all the above properties are called metrics.

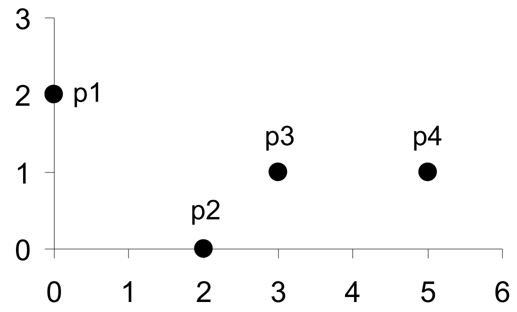
Feature spaces and proximity measures

- Famous example: Euclidean vector space *E*=(*Dom*, *dist*)
  - (*Dom*, *dist*) is a metric space
  - $Dom = \mathbb{R}^d$
  - $dist(x, y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$
- Notation:
  - Euclidean vector space =: “Feature space”
  - Vectors (Objects in the Euclidean feature space) =: “Feature vectors”
  - The *d* dimensions of the vector space =: “Features”
- Standardization is necessary, if scales differ!
  - e.g., age (e.g., range [0-100] vs salary (e.g., range: 10000-100000))

We will come back to this in a few slides

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Point coordinates



	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance matrix

Feature spaces and proximity measures

- Manhattan distance or City-block distance (L_1 norm)
  - $dist_1 = |p_1\text{-}q_1|+|p_2\text{-}q_2|+\ldots+|p_d\text{-}q_d|$
  - The sum of the absolute differences of the *p,q* coordinates

- Euclidean distance (L_2 norm)
  - $dist_2 = ((p_1\text{-}q_1)^2+(p_2\text{-}q_2)^2+\ldots+(p_d\text{-}q_d)^2)^{1/2}$
  - The length of the line segment connecting p and q

- Supremum distance (L_{max} norm or L_∞ norm)
  - $dist_\infty = \max\{|p_1\text{-}q_1|, |p_2\text{-}q_2|,\ldots,|p_d\text{-}q_d|\}$
  - The max difference between any attributes of the objects.

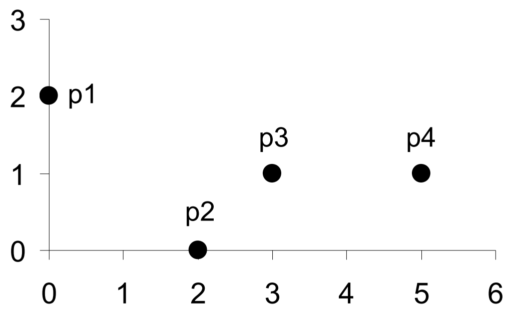
- Minkowski Distance (Generalization of L_p-distance)
  - $dist_p = (|p_1\text{-}q_1|^p + |p_2\text{-}q_2|^p + \ldots + |p_d\text{-}q_d|^p)^{1/p}$

Feature spaces and proximity measures

- Example

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Point coordinates



L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L1 distance matrix

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

L2 distance matrix

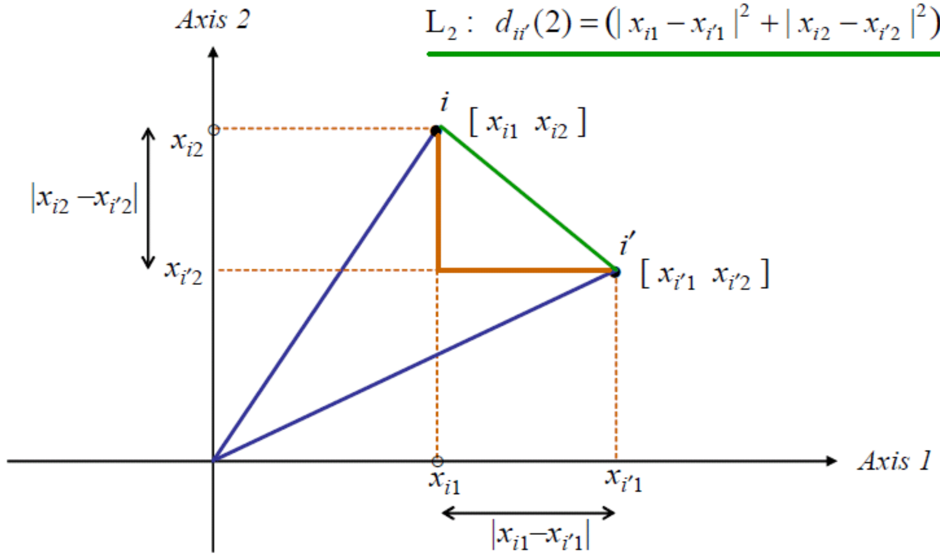
L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

L_∞ distance matrix

Feature spaces and proximity measures

$$L_1: \; d_{ii'}(1) = |x_{i1} - x_{i'1}| + |x_{i2} - x_{i'2}|$$

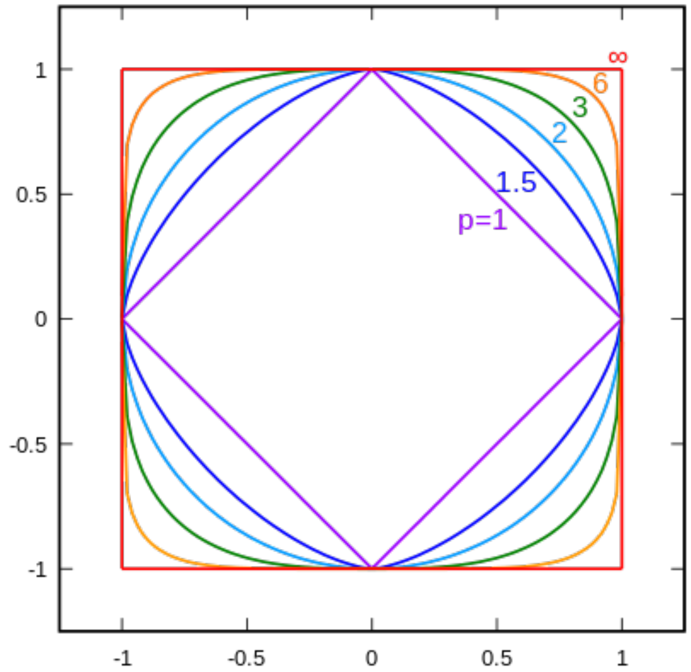
$$L_2: \; d_{ii'}(2) = (|x_{i1} - x_{i'1}|^2 + |x_{i2} - x_{i'2}|^2)^{1/2}$$



Source:http://www.econ.upf.edu/~michael/stanford/maeb5.pdf

Feature spaces and proximity measures

- Let *x,y* in [-1,1]
- For L1 norm
  - $|(x,y)|_1=1$ => x+y=1
  - If x=1, y=0
  - If x=0.8, y=0.2
  - …
- For L2 norm
  - $(x^2+y^2)^{1/2}=1$
  - It is circle
- …



Unit Circle for different Lp-distances

Source:https://de.wikipedia.org/wiki/P-Norm

Normalization

- Attributes with large ranges outweigh ones with small ranges
  - e.g. income [10K-100K]; age [10-100]
- To balance the “contribution” of an attribute *A* in the resulting distance, the attributes are scaled to fall within a small, specified range.
- min-max normalization: to [new_min_A , new_max_A]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  - e.g. normalize age=30 in [0-1], when min=10,max=100. new_age=((30-10)/(100-10))*(1-0)+0=2/9
- z-score normalization also called zero-mean normalization
  - After zero-mean normalizing each feature will have a mean value of 0

$$v' = \frac{v - mean_A}{stand\_dev_A}$$

e.g. normalize 70,000 iff μ=50,000, σ=15,000.
new_value = (70,000-50,000)/15,000=1.33

Proximity between binary attributes 1/2

- A binary attribute has only two states: 0 (absence), 1 (presence)
- A contingency table for binary data

Instance j

Instance i	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

q = the number of attributes where i was 1 and j was 1
t = the number of attributes where i was 0 and j was 0

s = the number of attributes where i was 0 and j was 1
r = the number of attributes where i was 1 and j was 0

- Simple matching coefficient

  (for symmetric binary variables)

$$d(i, j) = \frac{r+s}{q+r+s+t}$$

- for asymmetric binary variables:

$$d(i, j) = \frac{r+s}{q+r+s}$$

- Jaccard coefficient

$$sim_{Jaccard}(i, j) = \frac{q}{q+r+s}$$

  (for *asymmetric* binary variables)

Proximity between binary attributes 2/2

- Example:

Name	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	1	0	1	0	0	0
Mary	1	0	1	0	1	0
Jim	1	1	0	0	0	0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

(from previous slide)

q = the number of attributes where i was 1 and j was 1
t = the number of attributes where i was 0 and j was 0

s = the number of attributes where i was 0 and j was 1
r = the number of attributes where i was 1 and j was 0

$$d(i, j) = \frac{r+s}{q+r+s}$$

Proximity between categorical attributes

- A nominal attribute has >2 states (generalization of a binary attribute)
  - e.g. color={red, blue, green}
- Method 1: Simple matching
  - m: # of matches, p: total # of variables

$$d(i,j) = \frac{p-m}{p}$$

Name	Hair color	Occupation
Jack	Brown	Student
Mary	Blond	Student
Jim	Brown	Architect

- Method 2: Map it to binary variables
  - create a new binary attribute for each of the *M* nominal states of the attribute

Name	Brown hair	Blond hair	IsStudent	IsArchitect
Jack	1	0	1	0
Mary	0	1	1	0
Jim	1	0	0	1

Selecting the right proximity measure

- The proximity function should fit the type of data
  - For dense continuous data, metric distance functions like Euclidean are often used.
  - For sparse data, typically measures that ignore 0-0 matches are employed
    - We care about characteristics that objects share, not about those that both lack
- Domain expertise is important, maybe there is already a state-of-the-art proximity function in a specific domain and we don't need to answer that question again.
- In general, choosing the right proximity measure can be a very time consuming task
- Other important aspects: How to combine proximities for heterogenous attributes (binary and numeric and nominal etc.)
  - e.g., using attribute weights

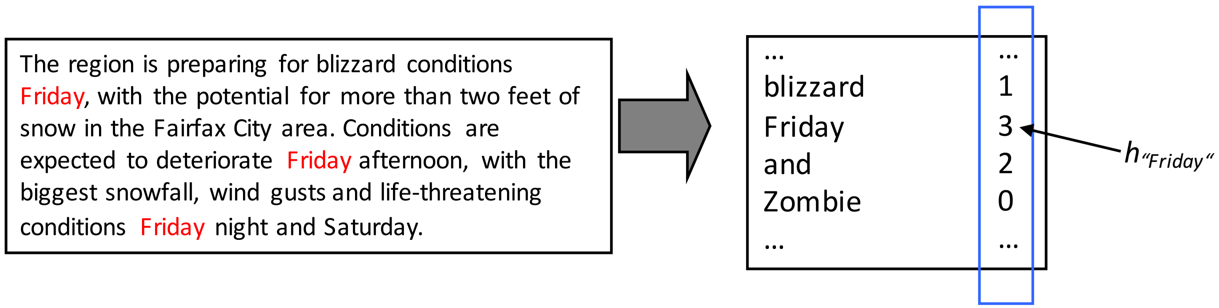
Outline

- Searching in Scientific Databases: Introduction
- Feature spaces and proximity measures
- Feature transformation for text data
- Algorithmic Paradigms for Similarity Query Processing

Feature transformations for text data 1/6

- Text represented as a set of terms (“Bag-Of-Words“ model)
  - Terms:
    - Single words (“cluster“, “analysis“..)
      or
    - bigrams, trigrams, …n-grams (“cluster analysis“..)

  - Transformation of a document d in a vector $r(d) = (h_1, \ldots, h_d)$, $h_i \geq 0$: the frequency of term t_i in d



Feature transformations for text data 2/6

- Challenges/Problems in Text Mining:
  1. Common words (“e.g.”, “the”, “and”, “for”, “me”)
  2. Words with the same root (“fish”, “fisher”, “fishing”,…)
  3. Very high-dimensional space (dimensionality d > 10.000)
  4. Not all terms are equally important
  5. Most term frequencies $h_i = 0$ (“sparse feature space“)
- More challenges due to language:
  - Different words have same meaning (synonyms)
    - “freedom” – “liberty”
  - Words have more than one meanings
    - e.g. “java”, “mouse”

Feature transformations for text data 3/6

- Problem 1: Common words (“e.g.”, “the”, “and”, “for”, “me”)
  - Solution: ignore these terms (Stopwords)

    There are stopwords list for all languages in WWW.

- Problem 2: Words with the same root (“fish”, “fisher”, “fishing”,…)
  - Solution: Stemming

    Map the words to their root

    - "fishing", "fished", "fish", and "fisher" to the root word, "fish".

    For English, the Porter stemmer is widely used.
    (Porters Stemming Algorithms: http://tartarus.org/~martin/PorterStemmer/index.html)

    Stemming solutions exist for other languages also.

  The root of the words is the output of stemming.

Feature transformations for text data 4/6

- Problem 3: Too many features/ terms
  - Solution: Select the most important features (“Feature Selection“)
  - Example: average document frequency for a term
    - Very frequent items appear in almost all documents
    - Very rare terms appear in only a few documents

Ranking procedure:

$$DF(t_i) = \frac{\#Docs\ containing\ t_i}{\#All\ documents}$$

1. Compute document frequency for all terms t_i :
2. Sort terms w.r.t. $DF(t_i)$ and get $rank(t_i)$
3. Sort terms by score(t_i)= $DF(t_i)$ · rank(t_i)
   e.g. score(t_{23}) = 0.82 · 1 = 0.82
   score(t_{17}) = 0.65 · 2 = 1.3
4. Select the k terms with the largest score(t_i)

Rank	Term	DF
1.	t_{23}	0.82
2.	t_{17}	0.65
3.	t_{14}	0.52
4.	…	…

Feature transformations for text data 5/6

- Problem 4: Not all terms are equally important
  - Idea: Very frequent terms are less informative than less frequent words. Define such a term weighting schema.
  - Solution: TF-IDF (Term Frequency · Inverse Document Frequency)

    Consider both the importance of the term in the document and in the whole collection of documents.

$$TF(t,d) = \frac{n(t,d)}{\sum_{w \in d} n(w,d)}$$

The relative frequency of term t in d [n(t,d) = # t in d]

$$IDF(t) = \log\left(\frac{|DB|}{\left|\{d \mid d \in DB \wedge t \in d\}\right|}\right)$$

Inverse frequency of term t in all DB

$$TF \times IDF = TF(t,d)IDF(t)$$

Feature vector with TF IDF : $r(d) = (TF(t_1, d) \cdot IDF(t_1), \ldots, TF(t_n, d) \cdot IDF(t_n))$

Feature transformations for text data 6/6

- Problem 5: for most of the terms $h_i = 0$
  - Euclidean distance is not a good idea: it is influenced by vectors lengths
  - Idea: use more appropriate distance measures

Jaccard Coefficient: Ignore terms absent in both documents

$$d_{Jaccard}(d_1, d_2) = 1 - \frac{|d_1 \cap d_2|}{|d_1 \cup d_2|} = \frac{|\{t | t \in d_1 \wedge t \in d_2\}|}{|\{t | t \in d_1 \vee t \in d_2\}|}$$

Cosine Coefficient: Consider term values (e.g. TFIDF values)

$$d_{\cos inus}(d_1, d_2) = 1 - \frac{\langle d_1, d_2 \rangle}{\|d_1\| \cdot \|d_2\|} = 1 - \frac{\sum_{i=0}^{n} \left(d_{1,i} \cdot d_{2,i} \right)}{\sqrt{\sum_{i=0}^{n} d_{1,i}{}^2} \cdot \sqrt{\sum_{i=0}^{n} d_{2,i}{}^2}}$$