IN2110 SPRING 2022

SPRÅKTEKNOLOGISKE METODER

Eriv Velldal & Jan Tore Lønning

Plan

Lectures 2-5

- How to represent (language) data in a mathematical model.
- Vector space models.
- Representing
  - Documents (today)
  - Words (week 5)
- Vector-based machine learning
  - Classification (week 3)
  - Clustering (week 5)

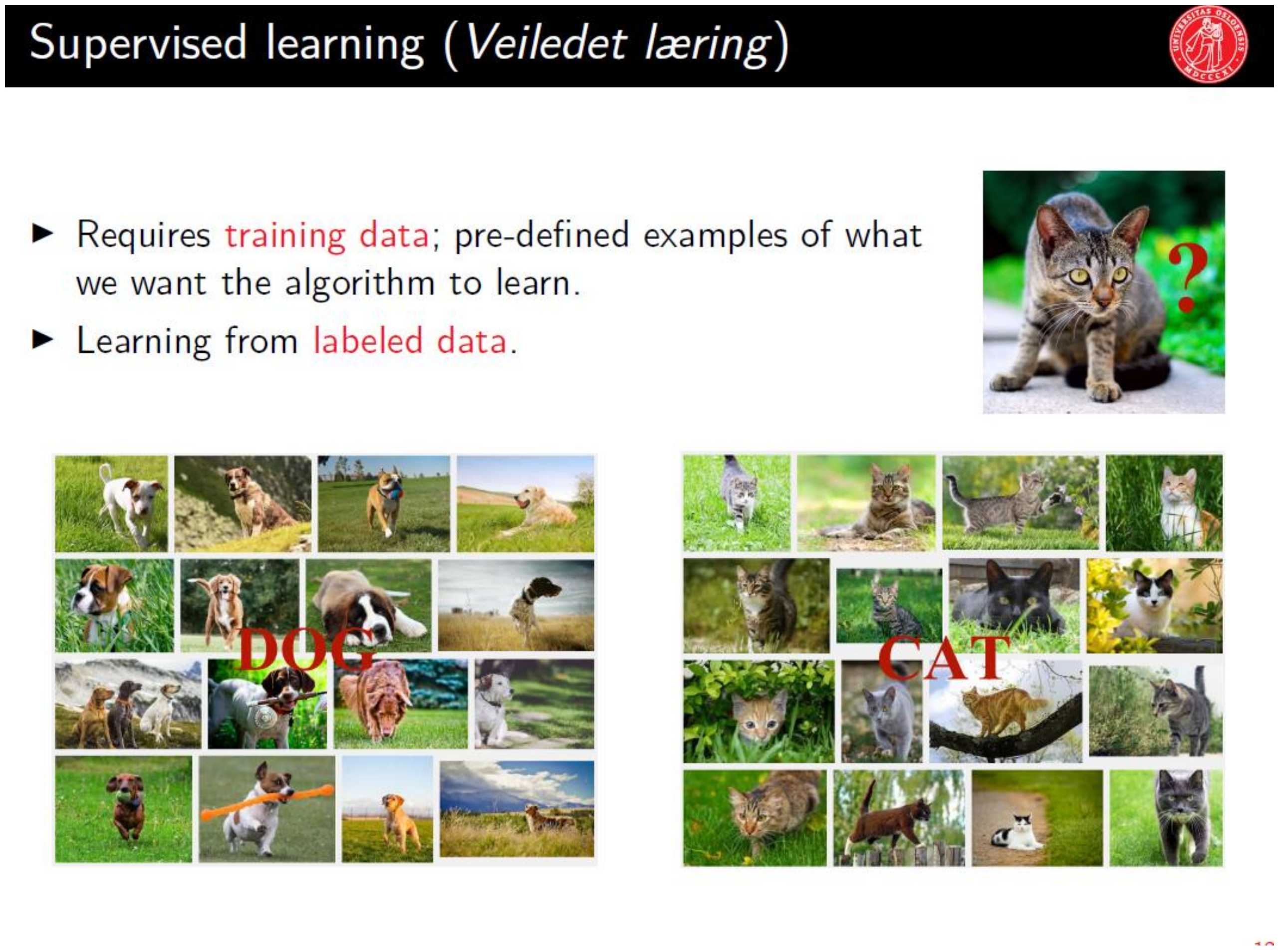
Today

- Examples
- Features (no: "trekk")
- Geometrical views:
  - Vector space models
- Application to documents and Information retrieval

Disclaimer

- I am only a substitute teacher for Erik Velldal
- The slides will be a mixture
  - Erik's slides from last year
  - My slides from IN3050 and IN4080
  - Some new slides (like this one)

Similarity

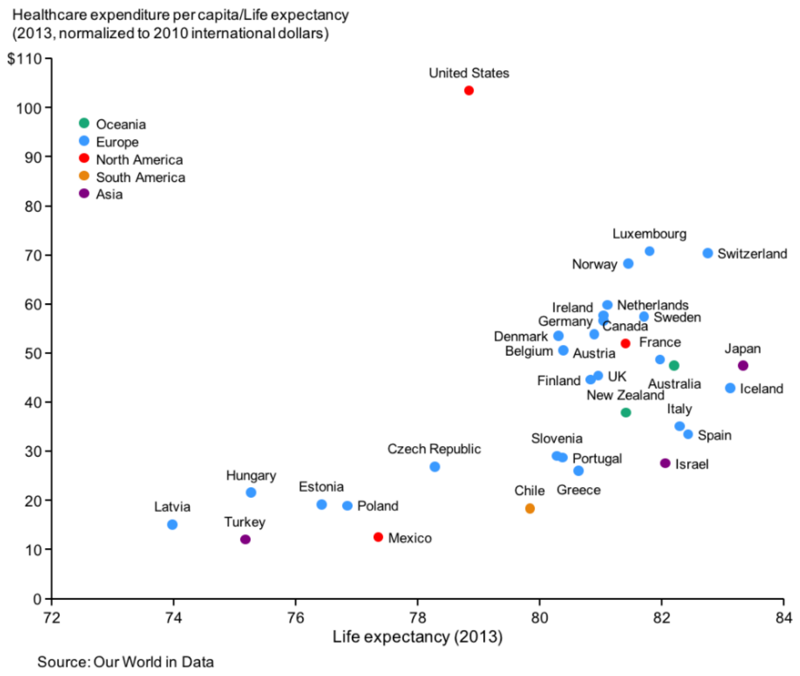


- We classify the image from how similar it is to the training images in the two classes
- But how do we measure similarity?

Features

- To classify cats and dogs, we (as humans) will typically consider attributes or features like:
  - color
  - size
  - fur
  - ears
  - …
- For a machine to classify images of cats and dogs it has to consider features of the image (pixels)

Numerical features and geometry



https://www.aploris.com/blog/charts/category/scatter-chart/

(Arbitrary) example:

- Countries
  - many possible features
  - choose by purpose
- With two numerical features
  - Plot the numbers
  - See which country is similar to which other countries
    - use this to predict other properties (ML)
  - Compare countries with respect to one feature given the other
    - Vectors

Example data set: email spam

	spam	chars	lines breaks	'dollar' occurs. numbers	'winner' occurs?	format	number
1	no	21,705	551	0	no	html	small
2	no	7,011	183	0	no	html	big
3	yes	631	28	0	no	text	none
4	no	2,454	61	0	no	text	small
5	no	41,623	1088	9	no	html	small
…							
50	no	15,829	242	0	no	html	small

From OpenIntro Statistics
Creative Commons license

There are more variables (attributes) in the data set

- Data are typically represented in a table
- Each column one attribute (feature)
- Each row an observation (n-tuple, vector)
- (cf. Data base)

Example data set: email spam

	spam	chars	lines breaks	'dollar' occurs. numbers	'winner' occurs?	format	number
1	no	21,705	551	0	no	html	small
2	no	7,011	183	0	no	html	big
3	yes	631	28	0	no	text	none
4	no	2,454	61	0	no	text	small
5	no	41,623	1088	9	no	html	small
…							
50	no	15,829	242	0	no	html	small

50 observations, rows

7 variables, columns

4 categorical variables

3 numerical variables

The larger picture

- This is how data sets are presented in texts on statistics or machine learning.
- But in real life, you want to apply ML to new tasks, then there is a lot of work before you have a data set like that:
  1. Data Collection and Preparation
  2. Feature Selection and extraction
- And for supervised learning, in particular
  3. Label the data, e.g., whether an x-ray shows cancer

Transforming the classification task

- The task of predicting from an e-mail

yes/no

- is transformed to the task of predicting from some features

(Chars: 21,705, Lines: 551, ‘dollar’: 0,
‘winner’: no, format: html, number: small)

yes/no

- is transformed to the task of predicting from a numerical vector to a number

$$\boldsymbol{x} = (x_1, x_2, \dots, x_n)$$

$$y \in \{0, 1\}$$

Types of features (and statistical variables)

Categorical

- Person: Name
- Word: Part of Speech (POS)
  - {Verb, Noun, Adj, …}
- Noun: Gender
  - {Mask, Fem, Neut}
- Sequence of words: Grammatical English sentence or not?

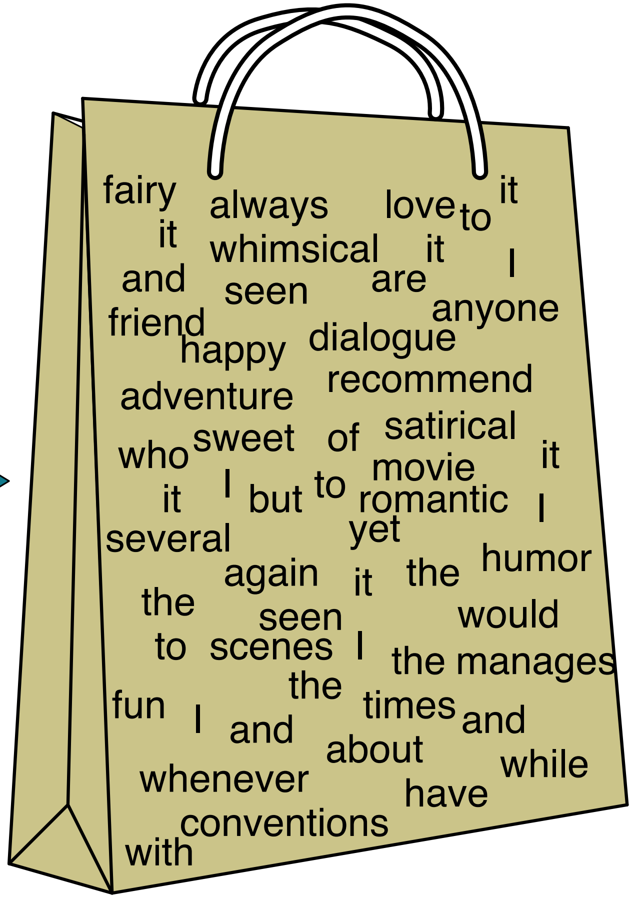
Numerical

- Person: Years of age, Weight, Height
- Word: length
- Text: number of occurrences of *great*, (42)
- Relative frequency of a word in a text: (0.0186%)

A binary categorical feature can be considered numerical: 0 or 1
Other categorical features can be represented by several binary features

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



From Jurafsky & Martin

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Term-document matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

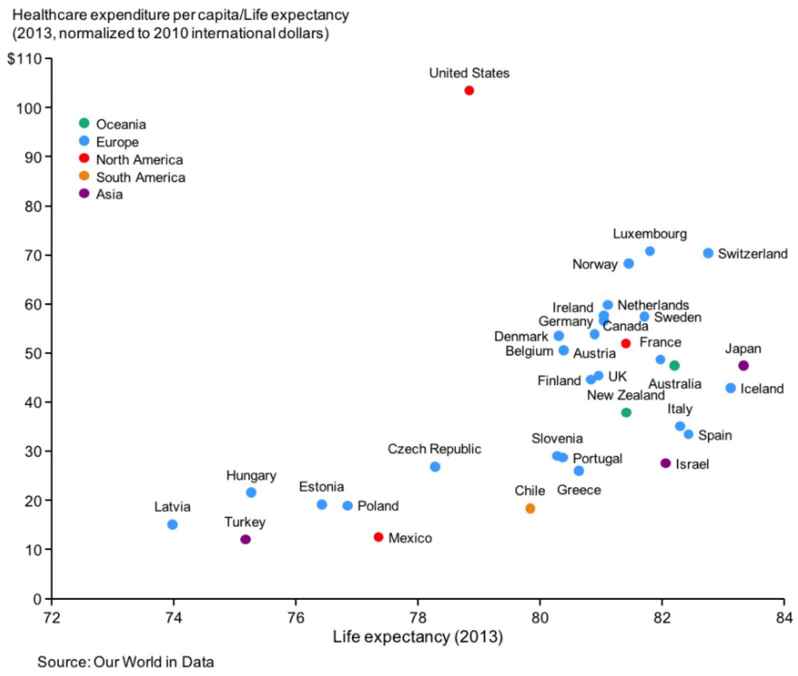
- Example of a co-occurrence matrix
- More specifically,
  a $m \times n$ term-document matrix
  - m terms, n documents
- Count the number of occurrences of the terms in each document
- Each column represent a document
- Each row represents a term (word, feature)
- With 4 key words each document is represented as a 4-d vector
- (We could use any set of key words)

Shakespeare (from J & M)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Vectors are similar for the two comedies
- Different than the historical dramas
- Comedies have more fools and wit and fewer battles.

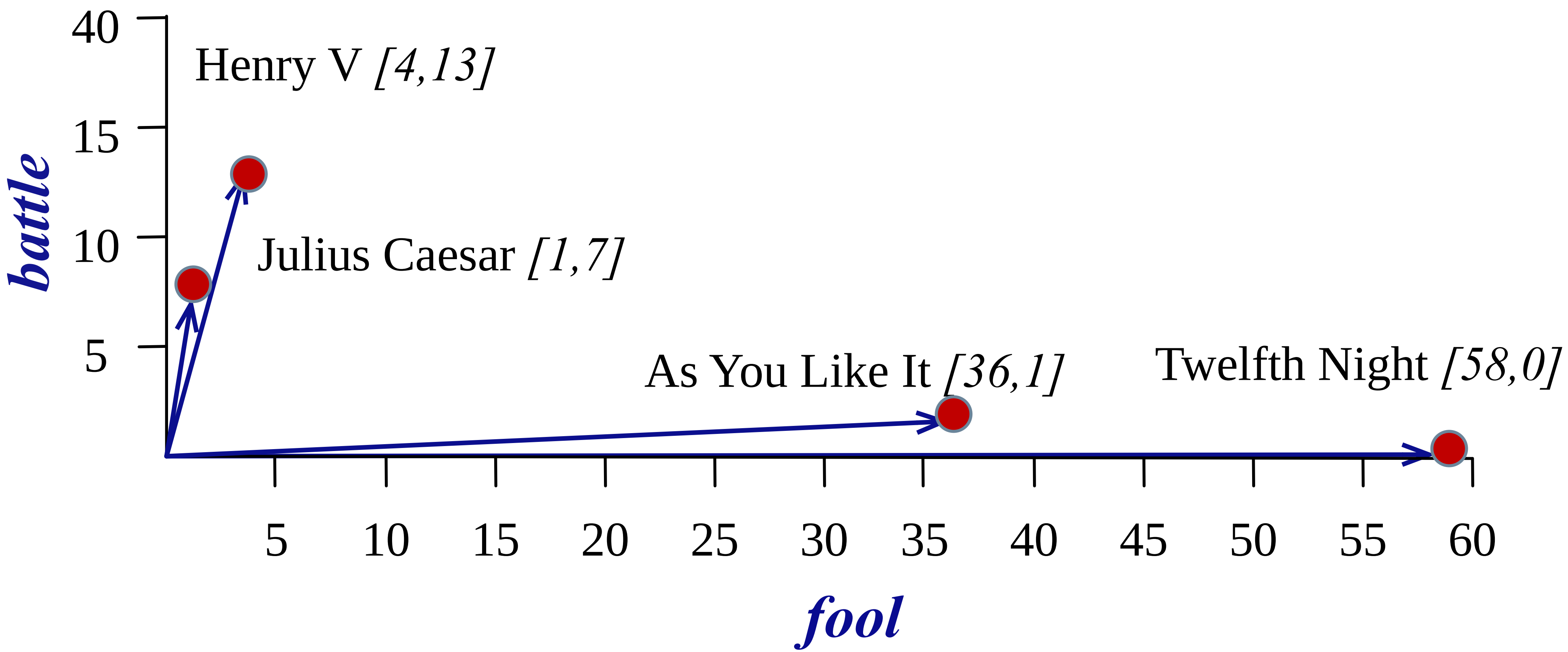
Numerical features understood geometrically



https://www.aploris.com/blog/charts/category/scatter-chart/

- Features correspond to dimensions
- Objects correspond to points
- Similarity by distance
- Two features
  - a plane
- Three features
  - a 3d space
- More features:
  - an abstract n-dimensional space, $\mathbb{R}^n$

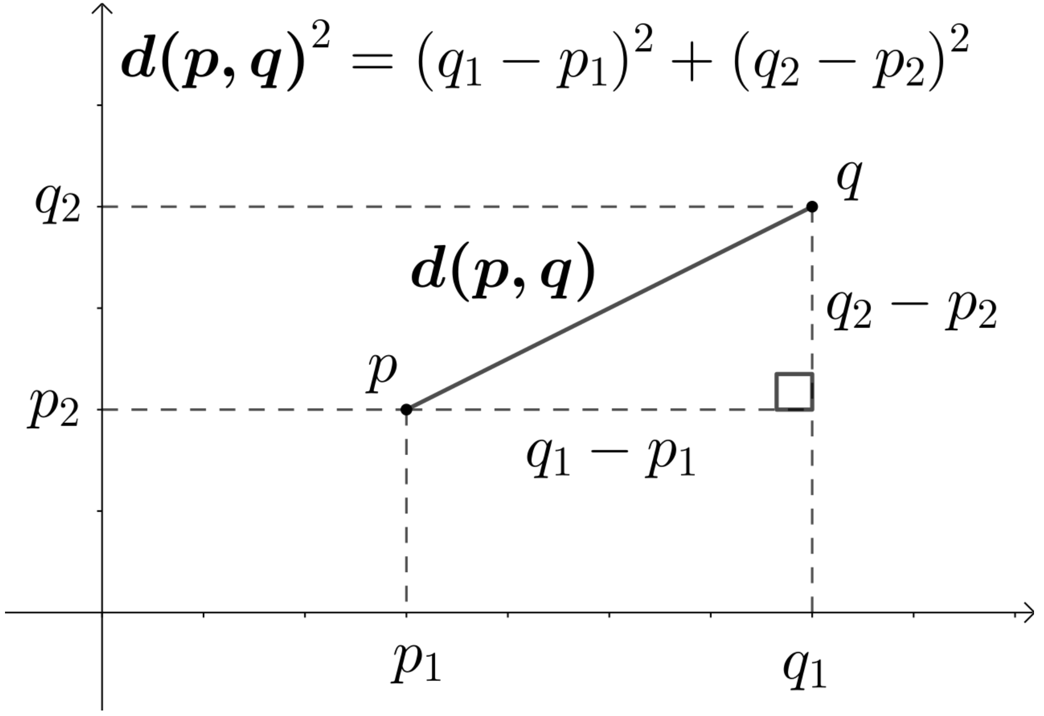
Plotting the documents with two key words



Distance between points

- Pythagorean theorem

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



- General form

$$d((x_1, x_2, \ldots, x_n), (y_1, y_2, \ldots y_n)) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

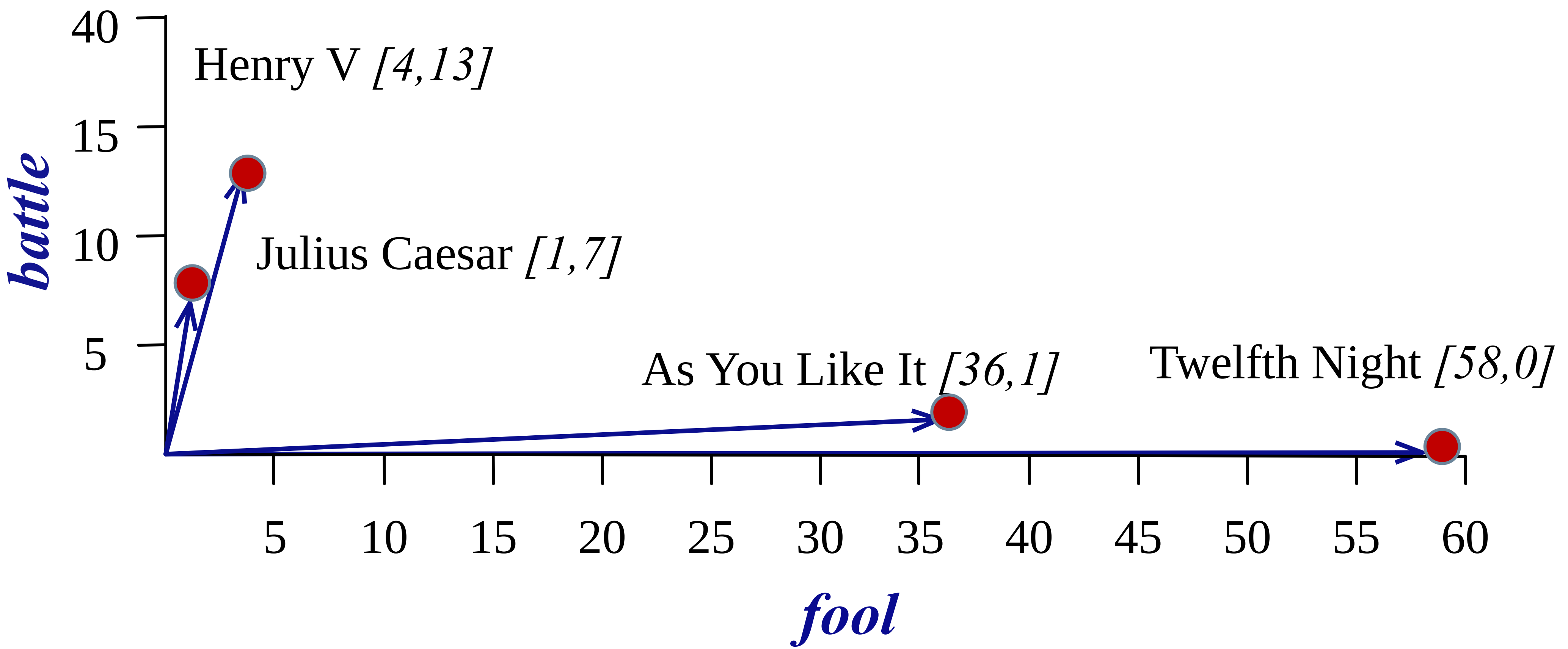
Vector spaces

Shakespeare (from J & M)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Vectors are similar for the two comedies
- Different than the historical dramas
- Comedies have more fools and wit and fewer battles.

Plotting the documents with two key words



Information retrieval (IR)

- We see clear patterns
- But the (red) points, representing documents, are not necessarily close together for similar documents.
- One reason is, of course, that documents have different lengths
- The directions towards the points are more interesting than the location of the points.
- This is where vectors (blue arrows) come at rescue.

Vectors

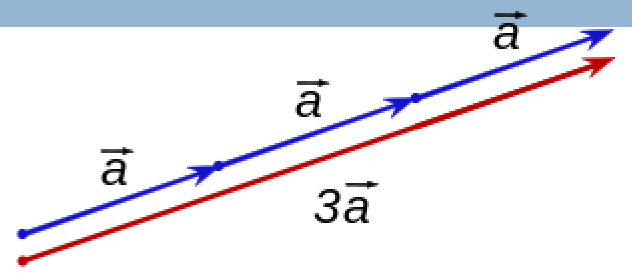
- An n-dimensional vector is an array of n scalars (real numbers)
  - $(x_1, x_2, \dots x_n)$

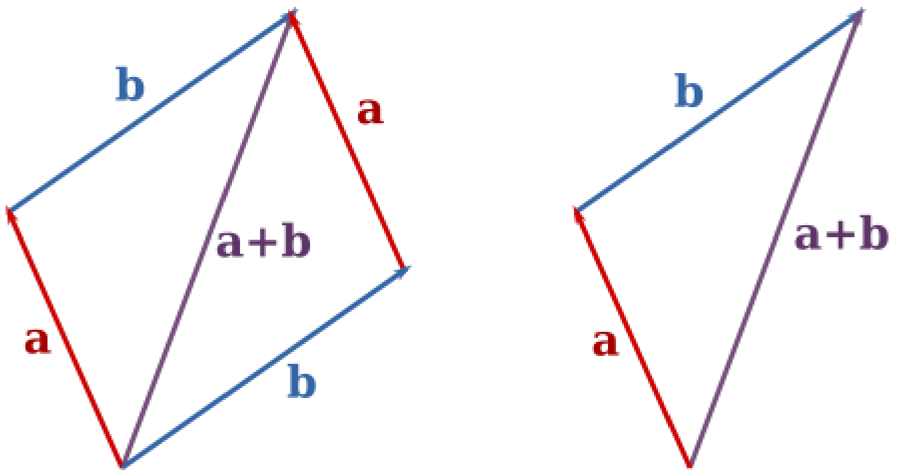
- Two operations on vectors
  - Scalar multiplication
  - $a(x_1, x_2, \dots x_n) = (ax_1, ax_2, \dots ax_n)$

  - Addition
  - $(x_1, x_2, \dots x_n) + (y_1, y_2, \dots y_n) = (x_1 + y_1, x_2 + y_2, \dots x_n + y_n)$

Euclidean vectors

- Also called geometric or spatial vectors
- 2D or 3D
- Characterized by
  - length
  - direction
- Used in physics for e.g.
  - forces, speed, acceleration, etc.

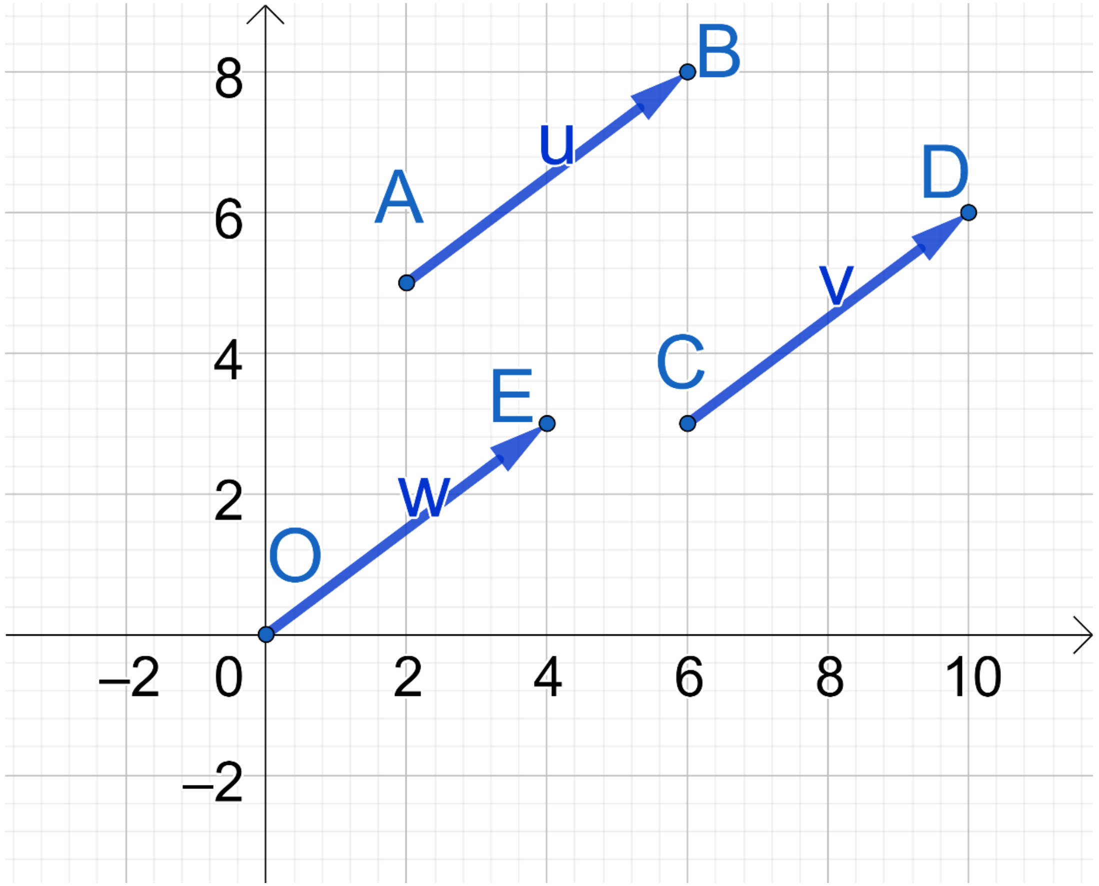




Figures from Wikipedia

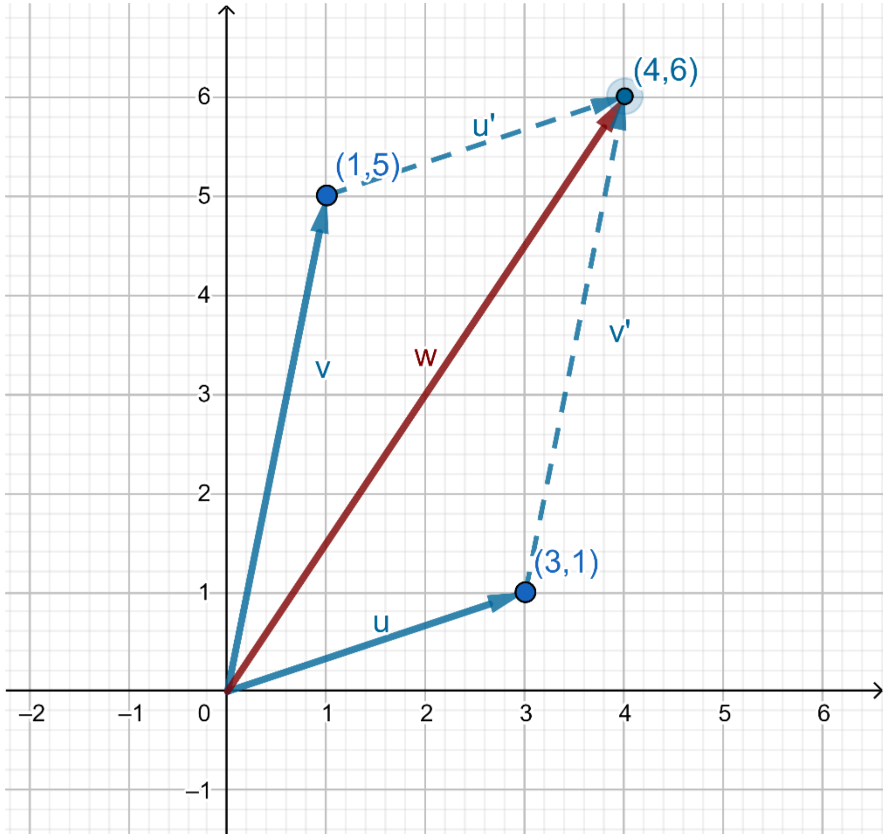
The connection

- Vectors with the same length and direction are considered equivalent
- A vector can be described by
  - start- and end-point
    - $\boldsymbol{u} = (A, B) = \big((2,5), (6,8)\big)$
    - $\boldsymbol{w} = \big((0,0), (4,3)\big)$
  - end-point
    - $\boldsymbol{w} = E = (4,3)$
    - the numeric form we use for addition and scalar multiplication

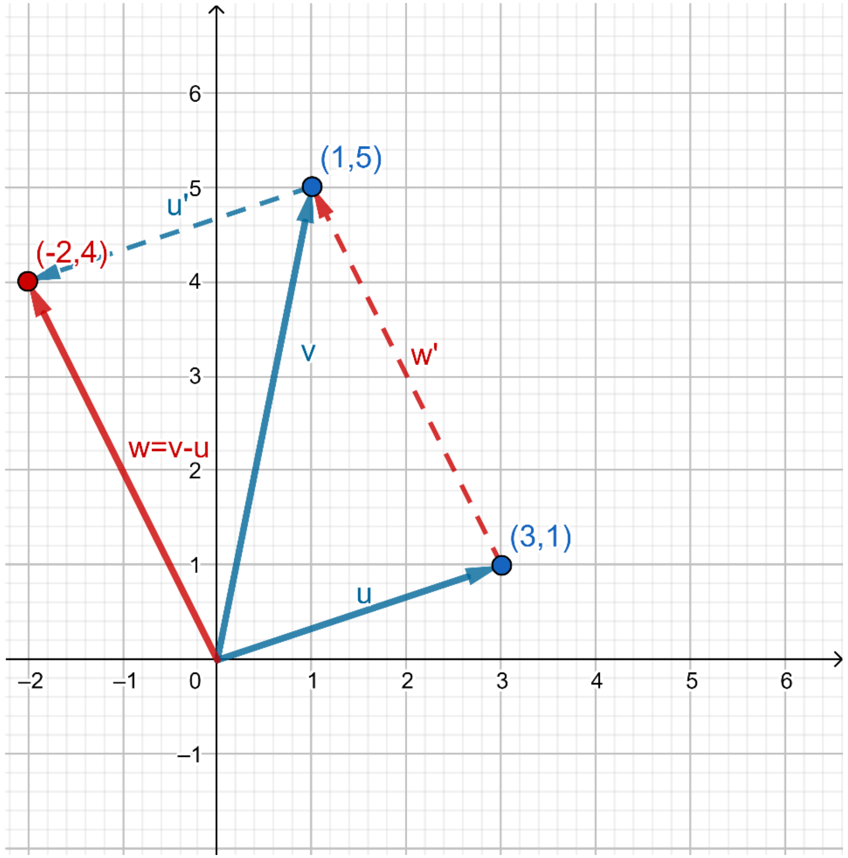


Vector operations

Sum



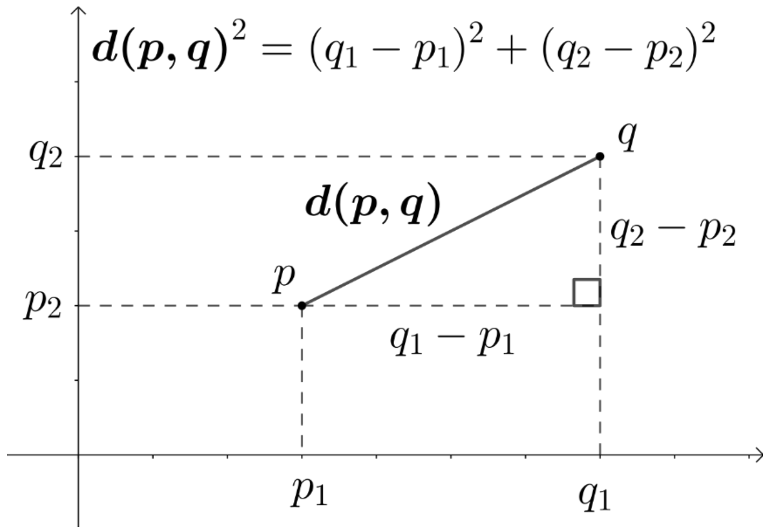
Difference



Norm of a vector

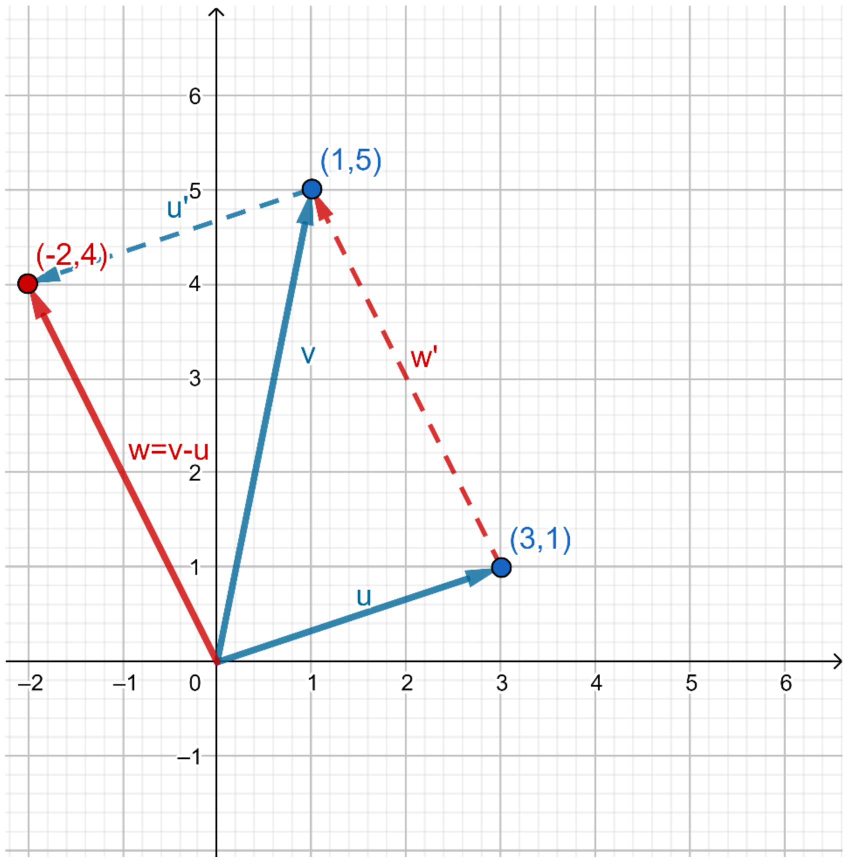
The norm (length) of a vector

- $||(x_1, x_2, \dots x_n)|| = \sqrt{{x_1}^2 + {x_2}^2 + \cdots + {x_n}^2}$
- This is called L2-norm
- Equals the distance between the end points.



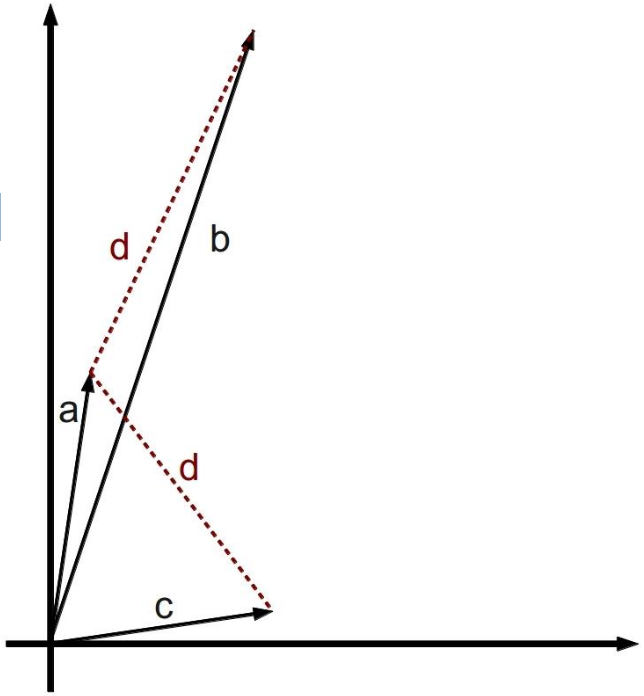
Euclidean distance between vectors

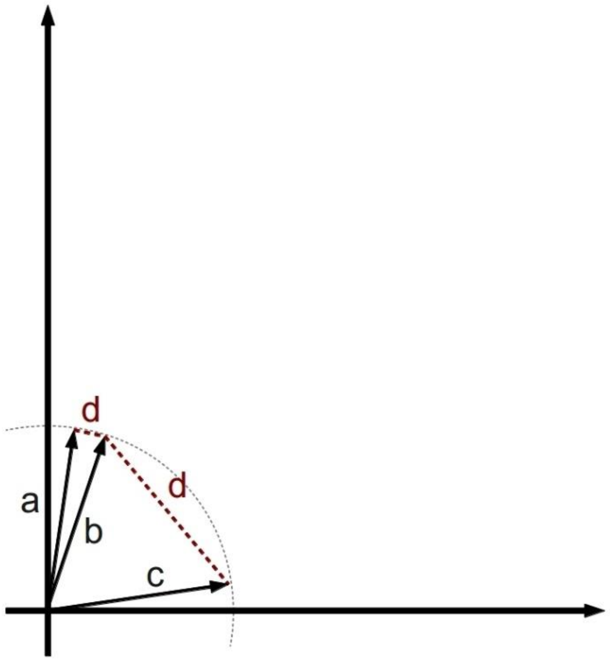
- The distance between the two end-points
- The length of the vector which is the distance between the two
- $dist((x_1, x_2, \dots x_n), (y_1, y_2, \dots y_n) = \|(x_1, x_2, \dots x_n) - (y_1, y_2, \dots y_n)\|$



$$d((x_1, x_2, \dots, x_n), (y_1, y_2, \dots y_n)) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

Normalize



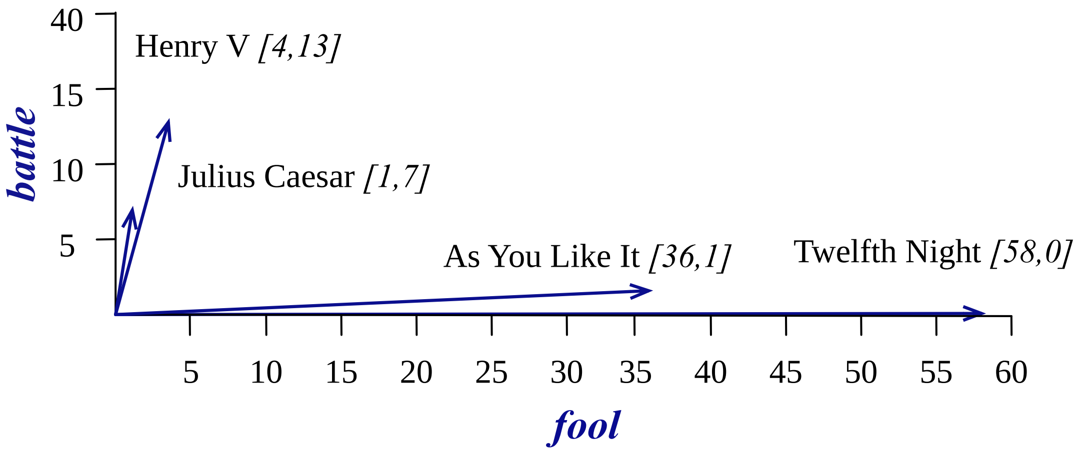


- We have seen that distance does not reflect document similarity
- One option is to normalize the vector to a vector of length one pointing in the same direction
- Replace $\boldsymbol{u}$ with $\mathbf{v} = \dfrac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$
- (Also other possibilities for documents, e.g., the relative frequency of the terms)

Cosine similarity

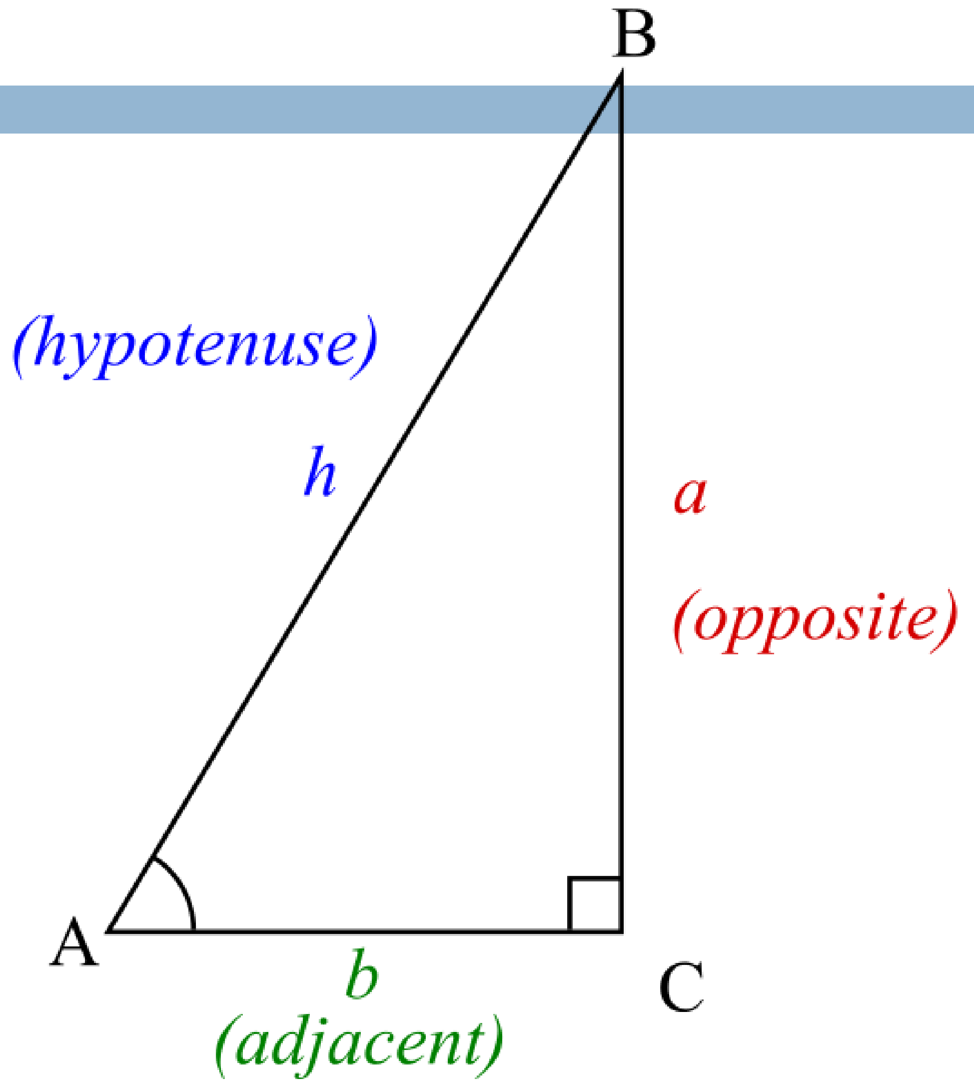
- Several possible ways to define similarity, e.g.,
  - Euclidean
  - Manhattan
- Most common: cosine
  - Do the arrows point in the same direction?

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\cdot\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$



Cosine

- $\cos(A) = \frac{b}{h}$
- $\sin(A) = \frac{a}{h}$

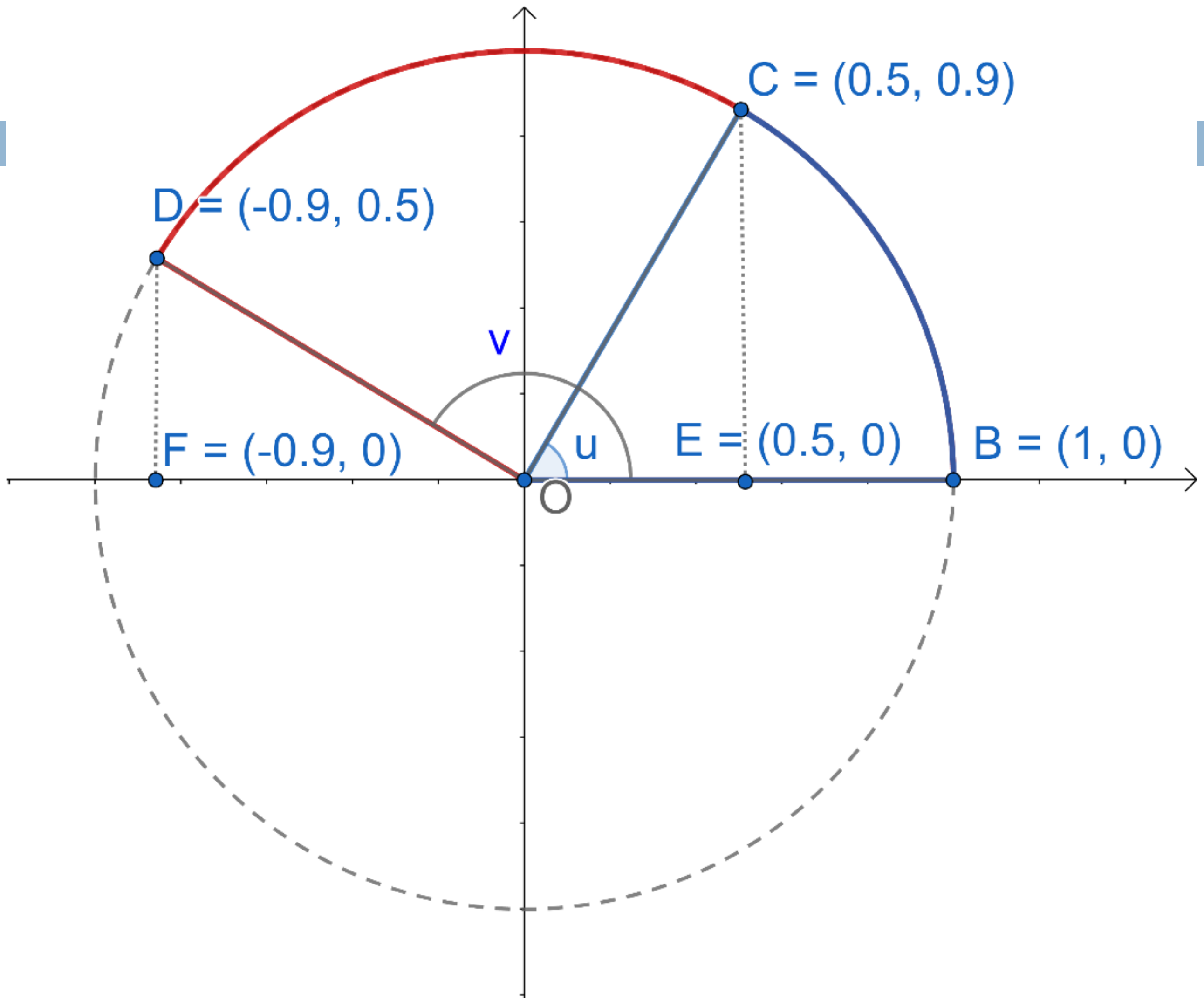


Cosine

Also defined for obtuse (non-acute) angles:

- $\cos(u) = C_1 = 0.5$
- $\cos(v) = D_1 =$

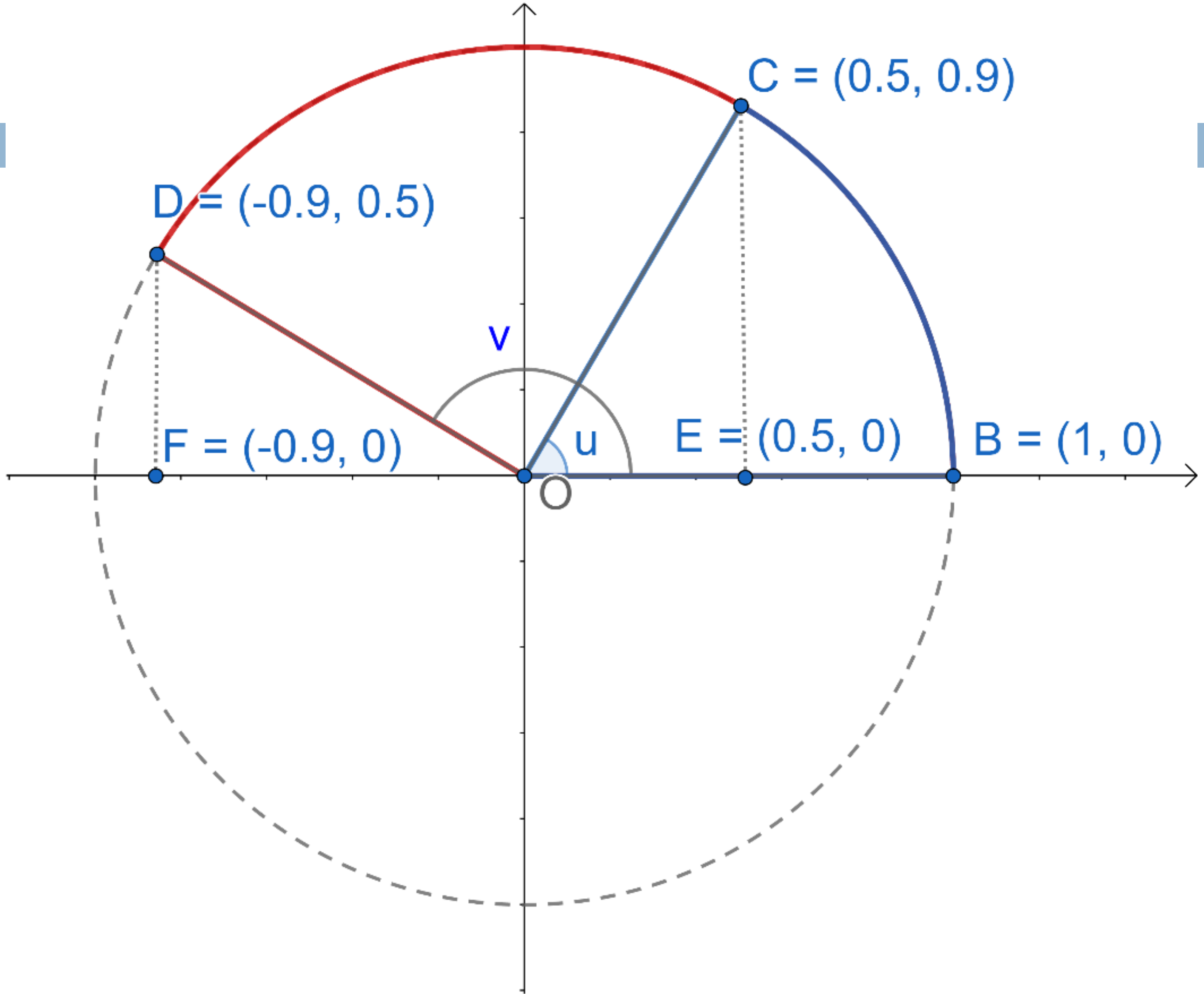
$$\sqrt{1 - 0.5^2} \approx -0.9$$



Cosine

Observations:

- $\cos(0) = 1$
- $\cos(u) = 0$ iff $u = \frac{\pi}{2} = 90^{\circ}$
- $0 < \cos(u) < 1$ iff $0 < u < \frac{\pi}{2}$
- $\cos(u) < 0$ iff $\frac{\pi}{2} < u \leq \pi$



Dot product

- $(x_1, x_2, \dots x_n) \cdot (y_1, y_2, \dots y_n) = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n = \sum_{i=1}^{n} x_i y_i$
- This is a scalar (real number) – not a vector
- $\boldsymbol{x} \cdot \boldsymbol{y} = ||\boldsymbol{x}||\,||\boldsymbol{y}||\cos(u)$ where u is the angle between the two vectors
- $\cos(u) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}||\,||\boldsymbol{y}||}$
- In 2D and 3D we can prove this
- In higher dimensions, we can use it to define cosine
  - and show that cosine get the expected properties

Let us try: $\cos(v_1, v_2)$

Full vectors (4 key words)

	AYLI	TwNi	JuCa	HenV
AYLI	1.000	0.950	0.945	0.949
TwNi	0.950	1.000	0.809	0.822
JuCa	0.945	0.809	1.000	0.999
HenV	0.949	0.822	0.999	1.000

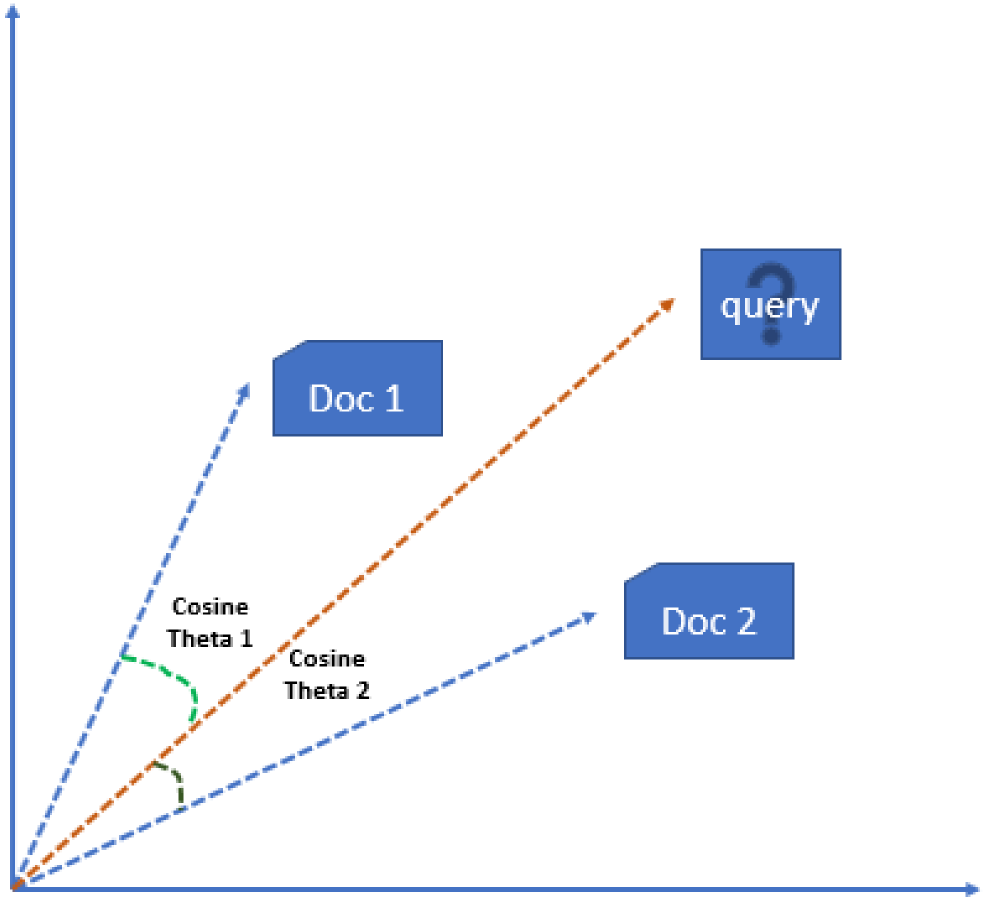
battles & fools

	AYLI	TwNi	JuCa	HenV
AYLI	1.000	1.000	0.169	0.321
TwNi	1.000	1.000	0.141	0.294
JuCa	0.169	0.141	1.000	0.988
HenV	0.321	0.294	0.988	1.000

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

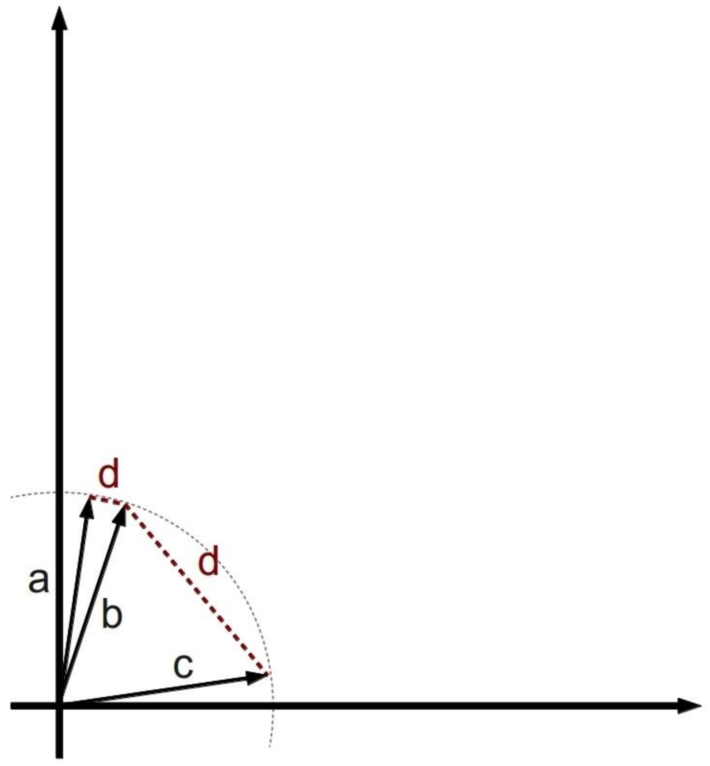
Information retrieval

- Consider the query as a short document
- Represent it as a vector in the same space as the documents
- Measure the similarity between the query and the documents
- Rank the relevance of the documents according to similarity with the quey

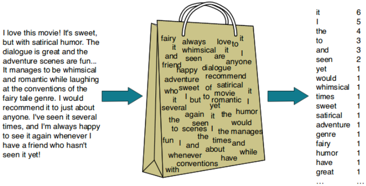


Observations

- Cosine similarity measures the similarity between vectors
  - Larger is better
- Euclidean distance measure the distance between vectors
  - Smaller is better
- With length normalized vectors, they will yield the same ranking of candidates



Frequencies



- Problem: Raw frequency counts not always good indicators of relevance.
- The most frequent words will typically not be very discriminative.
- A weighting function is therefore usually applied to the raw counts.

TF-IDF

- The most commonly used weighting function is tf-idf:

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.
  - The document frequency $\mathrm{df}(t_i)$ denotes the total number of documents in the collection that the term occurs in.

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.
  - The document frequency $\mathrm{df}(t_i)$ denotes the total number of documents in the collection that the term occurs in.
  - The inverse document frequency is defined as $\mathrm{idf(t_i)} = \log\left(\frac{N}{df(t_i)}\right)$, where N is the total number of documents in the collection.

TF-IDF

- The most commonly used weighting function is tf-idf:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.
  - The document frequency $\mathrm{df}(t_i)$ denotes the total number of documents in the collection that the term occurs in.
  - The inverse document frequency is defined as $\mathrm{idf(t_i)} = \log\left(\frac{N}{df(t_i)}\right)$, where N is the total number of documents in the collection.
  - The weight given to term t_i in document d_j is then computed as

$$\text{tf-idf}(t_i, d_j) = \mathrm{tf}(t_i, d_j) \times \mathrm{idf}(t_i)$$

TF-IDF

- The most commonly used weighting function is $\mathrm{tf\text{-}idf}$:
  - The term frequency $\mathrm{tf}(t_i, d_j)$ denotes the number of times the term t_i occurs in document d_j.
  - The document frequency $\mathrm{df}(t_i)$ denotes the total number of documents in the collection that the term occurs in.
  - The inverse document frequency is defined as $\mathrm{idf(t_i)} = \log\left(\frac{N}{df(t_i)}\right)$, where N is the total number of documents in the collection.
  - The weight given to term t_i in document d_j is then computed as

$$\mathrm{tf\text{-}idf}(t_i, d_j) = \mathrm{tf}(t_i, d_j) \times \mathrm{idf}(t_i)$$

  - A high $\mathrm{tf\text{-}idf}$ is obtained if a term has a *high* frequency in the given *document* and a *low* frequency in the document *collection* as whole.
  - The weights hence tend to filter out common terms.

Footnote: Variants of TF-IDF

- There are variants to both the TF and the IDF
- For TF:
  - Instead of raw frequency one could use relative frequencies or length normalize.
    - The result is the same as with raw frequency when we use cos-similarity
    - An option for other (classification) tasks
  - J&M uses Sublinear TF: (1 + log(tf)), 0 when tf=0
    - which can give a different ranking also with cosine similarity
- For IDF:
  - $idf_t = \log \frac{N}{df_t}$
  - Smooth: some avoid dividing by zero $idf_t = \log \frac{N}{df_t+1} + 1$, or other variants
- You don't have to learn these, but beware why you might get a result different from the book(s).

The effect of tf-idf

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.8 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. For example the 0.049 value for *wit* in *As You Like It* is the product of $\text{tf} = \log_{10}(20+1) = 1.322$ and $\text{idf} = .037$. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

Text pre-processing. Or, what is a word?

Raw: "The programmer's programs had been programmed."

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, ...?
  - Full-form words vs. lemmas vs. stems ...
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Text pre-processing. Or, what is a word?

Raw:	"The programmer's programs had been programmed."
Tokenized:	the programmer 's programs had been programmed .

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, . . . ?
  - Full-form words vs. lemmas vs. stems . . .
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Text pre-processing. Or, what is a word?

Raw:	"The programmer's programs had been programmed."
Tokenized:	the programmer 's programs had been programmed .
Lemmatized:	the programmer 's program have be program .

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, . . . ?
  - Full-form words vs. lemmas vs. stems . . .
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Text pre-processing. Or, what is a word?

Raw:	"The programmer's programs had been programmed."
Tokenized:	the programmer 's programs had been programmed .
Lemmatized:	the programmer 's program have be program .
W/ stop-list:	programmer program program

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, ...?
  - Full-form words vs. lemmas vs. stems ...
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Text pre-processing. Or, what is a word?

Raw:	"The programmer's programs had been programmed."
Tokenized:	the programmer 's programs had been programmed .
Lemmatized:	the programmer 's program have be program .
W/ stop-list:	programmer program program
Stemmed:	program program program

- Tokenization: Splitting a text into sentences and words or other units.
- Different levels of abstraction and morphological normalization:
  - What to do with case, numbers, punctuation, compounds, . . . ?
  - Full-form words vs. lemmas vs. stems . . .
- Stop-list: filter out closed-class words or function words.
  - The idea is that only *content words* provide relevant context.

Practical comments: Sparsity

- BoW feature vectors will be extremely high-dimensional.
- The number of *non-zero* elements will be very low.
- Few active features per word.
- We say that the vectors are sparse.
- This has implications for how to implement our data structures and vector operations:
- Don't want to waste space representing and iterating over zero-valued features.

Next: Two categorization tasks in machine learning

Classification

- Supervised learning, requiring labeled training data.
- Train a classifier to automatically assign *new* instances to *predefined* classes, given some set of training examples.
- (Topic for next week.)

Next: Two categorization tasks in machine learning

Classification

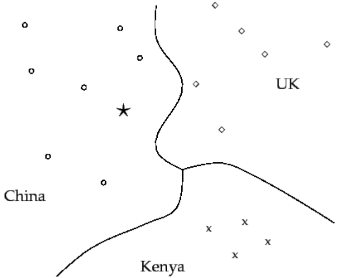
- Supervised learning, requiring labeled training data.
- Train a classifier to automatically assign *new* instances to *predefined* classes, given some set of training examples.
- (Topic for next week.)

Clustering

- Unsupervised learning from unlabeled data.
- Automatically group similar objects together.
- No predefined classes or structure, we only specify the similarity measure.
- (The topic for the week after.)

Classes and classification

- In our vector space model, objects are represented as points, so classes will correspond to collections of points; regions.
- Vector space classification is based on the contiguity hypothesis:



- Objects in the same class form a contiguous region, and regions of different classes do not overlap.
- Classification amounts to computing the boundaries in the space that separate the classes; *the decision boundaries*.

Vector space classification

- Classifiers based on vector space representations are well-suited for introducing the notion of classification:
- Little math required, easy to understand on the basis of geometrical intuitions.
- We will consider two very simple but powerful methods:
- K-Nearest Neighbor (KNN) classification
- Rocchio classification (a.k.a. Nearest centroid)
- Example task: text classification

Mandatory assignment 1 a

- Topic classification of news articles (reviews in NoReC)
  - using *k*NN (k nearest neighbors)
  - with BoW features and TF-IDF weighting
- In: 25/2, Out: 2/2
- Group work encouraged
- https://www.uio.no/studier/emner/matnat/ifi/IN2110/v22/obliger/
- Groups:
  - Wed. 14.15-16, Chill, digital this week
  - Thurs. 12.15-14, digital

Next week

- Classification algorithms:
  - *k*NN (k nearest neighbors)
  - Rocchio classification
- Reading: The chapter Vector Space Classification (sections 14–14.4) in Manning, Raghavan & Schütze (2008); https://nlp.stanford.edu/IR-book/
- PS: Want to learn more about IR? Take IN3120 – Search Technology