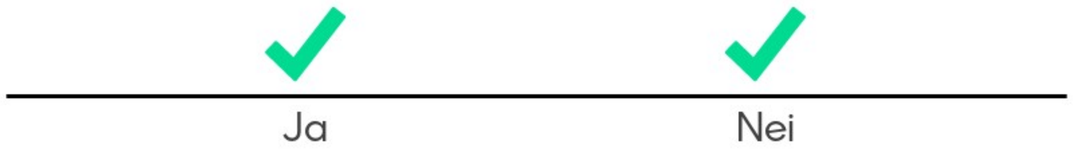
IN[34]120 Søketeknologi - chatbot workshop

2023-10-27 10:15 @ Chill

- Vectors
- (1) tf-idf
- (2) cosine similarity
- Workshop: bruke (1) og (2) for å lage chatbot
- Evt: Oblighjelp

Vil du at det skal forekomme oblighjelp i dag?



Pensumrelevant

- → TF-IDF
- → Cosine similarity
- → Vektorer

IKKE relevant:

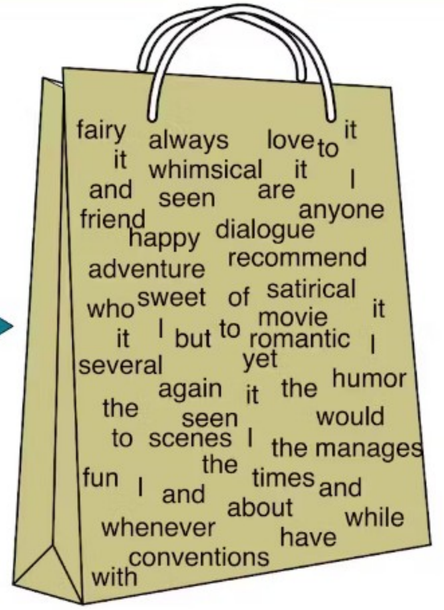
- → Chatbots
- → For chatting: IN4080
- → Fint å lage chatbots for å bruke teorien i praksis

(Slides lånt fra IN4080 og IN2110)

- IN4080: Natural language processing
- https://www.uio.no/studier/emner/matnat/ifi/IN4080/h23/slides/06-lm-vectors.pdf
- IN2110: Språkteknologiske metoder
- https://www.uio.no/studier/emner/matnat/ifi/IN2110/v23/foiler/02_vektorrom_2023.pdf

The Bag of Words Representation

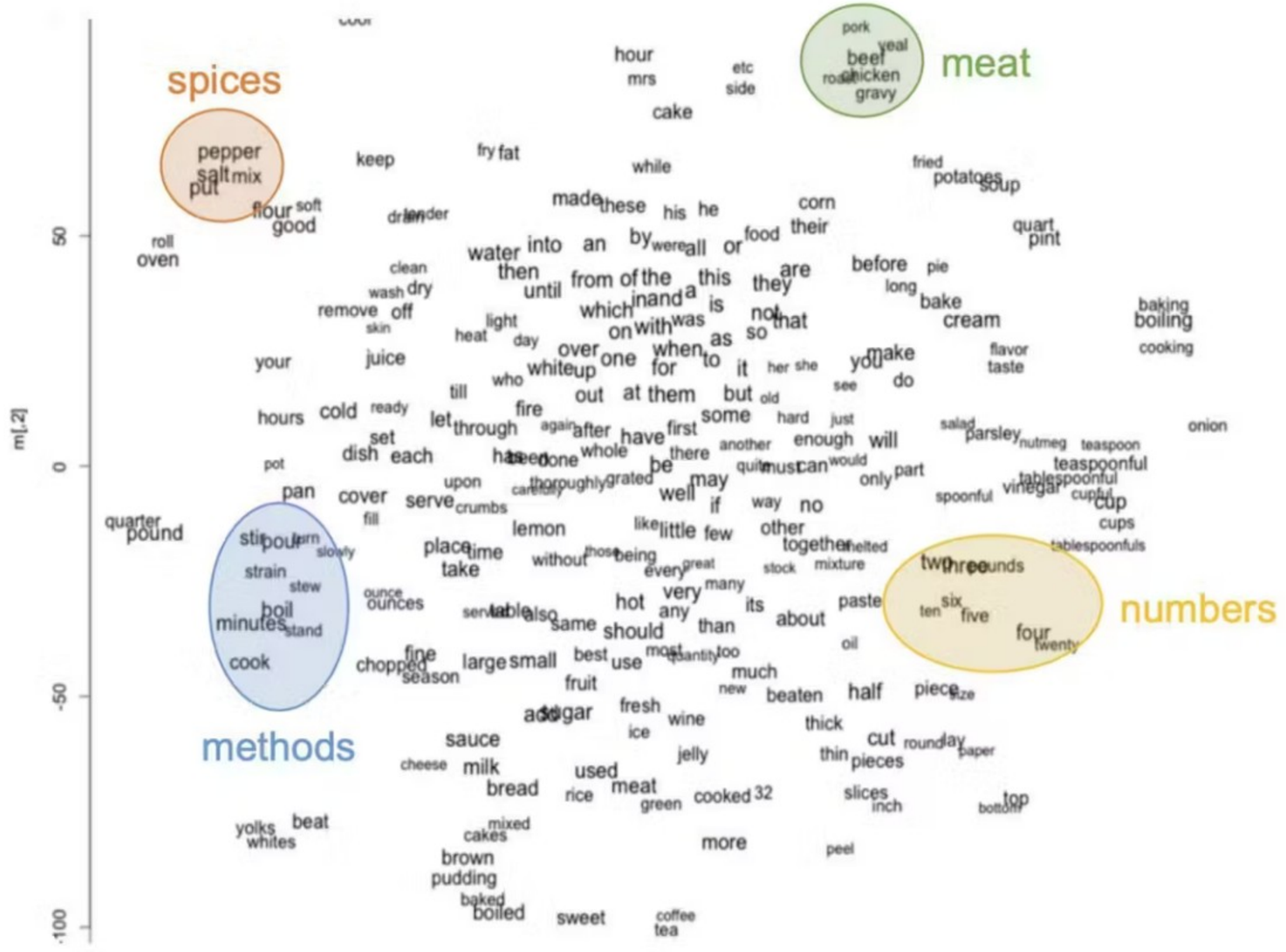
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy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



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
…	…

From Jurafsky & Martin

Vector semantics



https://www.adityathakker.com

- One row per term/word
- One column per document
- Values represent counts of terms in documents

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Term-document matrices

What can we do with such a matrix?

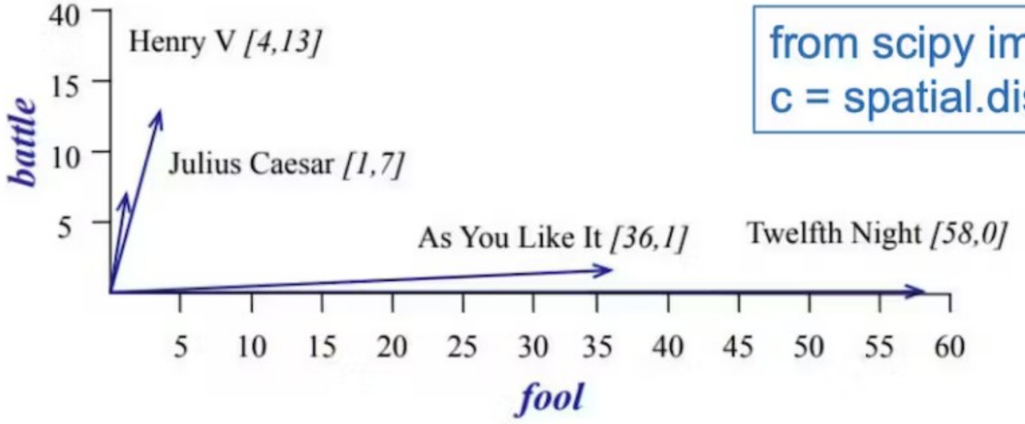
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Compute similarity between words
  - *fool* and *wit* are similar
- Compute similarity between documents
  - *As You Like It* and *Twelfth Night* are similar (comedies)
  - *Julius Caesar* and *Henry V* are similar (historical dramas)

Cosine similarity

Cosine similarity represents the **angle** between two vectors:

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

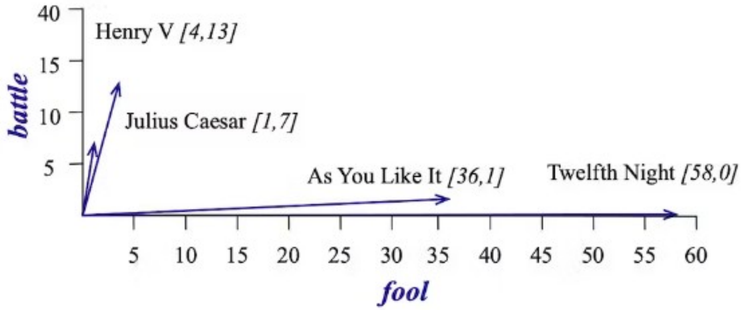



```
from scipy import spatial
c = spatial.distance.cosine(v, w)
```

Cosine similarity

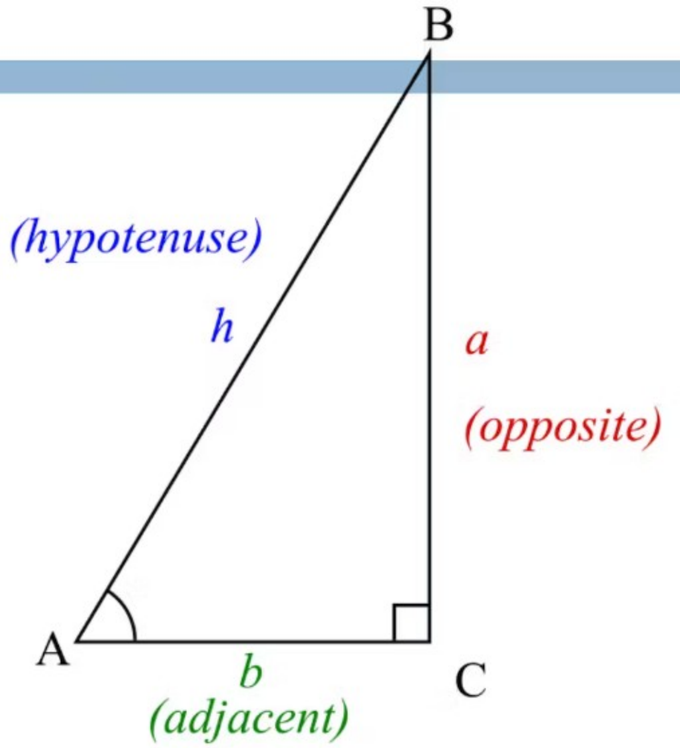
- Several possible ways to define similarity, e.g.,
  - Euclidean
  - Manhattan
- Most common: cosine
  - Do the arrows point in the same direction?

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



Cosine

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

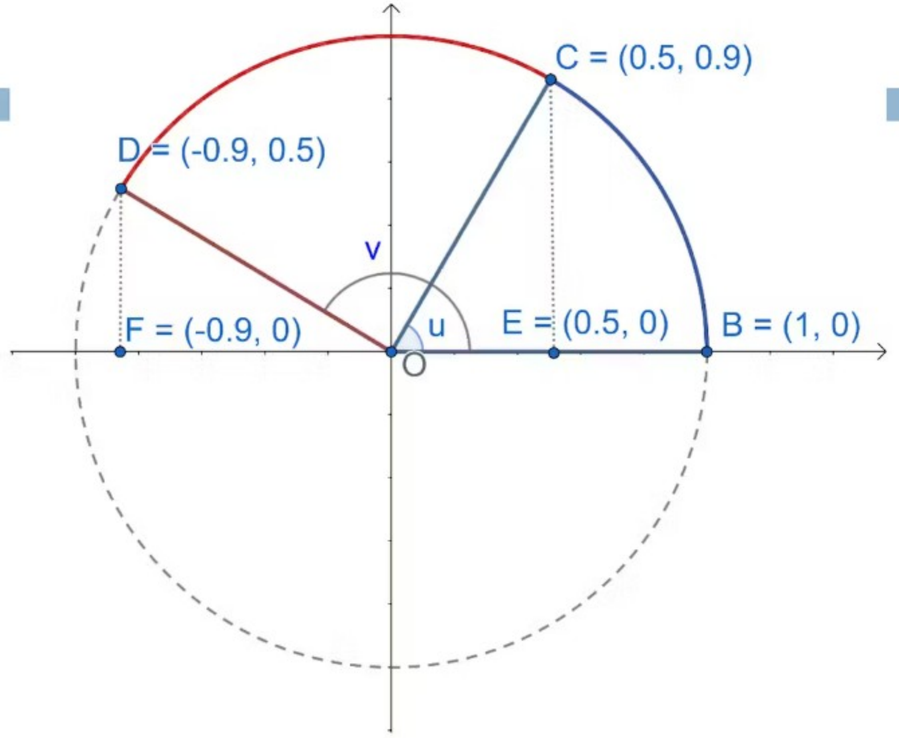


Cosine

Also defined for obtuse (non-acute) angles:

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

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



Term-document matrices

What can we do with such a 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

- Compute similarity between words
  - *fool* and *wit* are similar
  - $cosine(\text{fool}, \text{wit}) = cosine([36,58,1,4], [20,15,2,3]) = 0.93$
  - $cosine(\text{fool}, \text{battle}) = cosine([36,58,1,4], [1,0,7,13]) = 0.09$
- Compute similarity between documents
  - As You Like It and Twelfth Night are similar (comedies)
  - Julius Caesar and Henry V are similar (historical dramas)
  - $cosine(\text{AYLI}, \text{TN}) = cosine([1,114,36,20], [0,80,58,15]) = 0.95$
  - $cosine(\text{JC}, \text{HV}) = cosine([7,62,1,2], [13,89,4,3]) = 0.69$
  - $cosine(\text{TN}, \text{JC}) = cosine([0,80,58,15], [7,62,1,2]) = 0.81$

Term-document matrices

What can we do with such a matrix?

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

- Compute similarity between words
- Compute similarity between documents
- **Information retrieval:**
  - Encode the **query** as an additional document
  - Find documents that are most similar to the query

TF-IDF count weighting

- TF – term frequency:
  - $tf_{t,d}$ is the frequency of term t in document d
- DF – document frequency:
  - df_t is the number of documents containing term t
- IDF – inverse document frequency:
  - $idf_t = \frac{1}{df_t}$
  - Normalize by the collection size: $\frac{N}{df_t}$
  - By convention, take the log: $\log \frac{N}{df_t}$
- TF-IDF: $tf_{t,d} \cdot \log \frac{N}{df_t}$

Implementation: replace Scikit-Learn *CountVectorizer* by *TfIdfVectorizer*

Effekten av TF-IDF

TF-IDF count weighting

Raw counts:

	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

TF-IDF weights:

	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

TF-IDF did its job!

Dot product

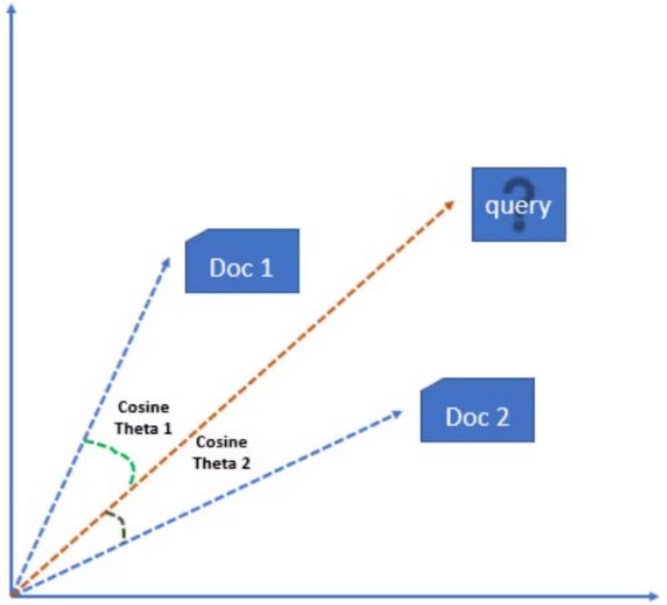
33

- $(x_1, x_2, \ldots x_n) \cdot (y_1, y_2, \ldots y_n) = x_1y_1 + x_2y_2 + \cdots + x_ny_n = \sum_{i=1}^{n} x_iy_i$
- 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

The missing link: dot product

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



Chatbots! Query.

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 $\mathrm{idf}(\mathrm{t_i}) = \log\left(\frac{N}{df(t_i)}\right)$, 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

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

  - 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.

Lag chatbots!

Litt prekode finnes i repoet: https://github.com/aohrn/in3120-2023/blob/main/gruppetimer/gruppe1/