UNIVERSITÀ
DEGLI STUDI
DI PADOVA

# Network Science

A.Y. 23/24

ICT for Internet & multimedia, Data science, Physics of data

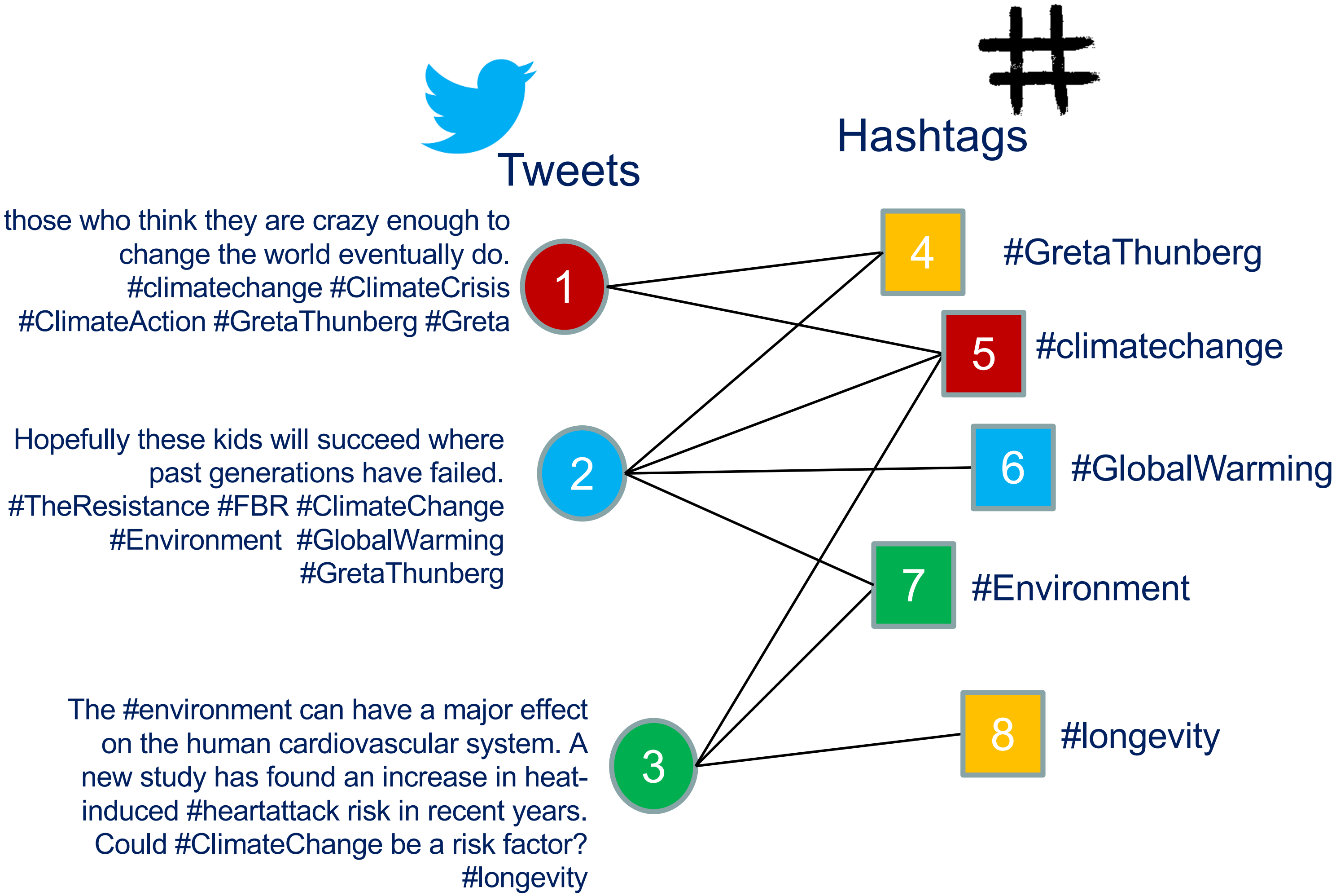
# Semantic networks

recap

Università degli Studi di Padova

# Conceptual picture

## of a semantic network on Twitter



# Probability matrices

## linking words to documents

number of occurrences of words in documents

$$N_{wd} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \begin{array}{l} \text{\#globalwarming} \\ \text{\#climatechange} \\ \text{\#climateaction} \\ \text{\#gretathunberg} \\ \text{\#environment} \end{array}$$

probability of words given a documents

$$P_{w|d} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{5} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{5} & \frac{1}{4} \\ \frac{1}{3} & 0 & \frac{1}{5} & 0 \\ 0 & 0 & \frac{1}{5} & \frac{1}{4} \\ \frac{1}{3} & 0 & \frac{1}{5} & \frac{1}{4} \end{bmatrix}$$

we capture the statistical properties by normalizing by columns

we identify a document probability

$$\boldsymbol{p}_d = \begin{cases} \dfrac{1}{D} & \text{equally likely} \\ \dfrac{n_d}{\sum_d n_d} & \text{custom} \end{cases}$$

# Probability matrices

## projecting to words or documents

bipartite network

joint probability of words and documents

$\boldsymbol{P}_{wd} = \boldsymbol{P}_{w|d}\,\mathrm{diag}(\boldsymbol{p}_d)$

$$\begin{matrix} 0 & 1/8 & 1/20 & 1/16 \\ 1/12 & 1/8 & 1/20 & 1/16 \\ 1/12 & 0 & 1/20 & 0 \\ 0 & 0 & 1/20 & 1/16 \\ 1/12 & 0 & 1/20 & 1/16 \end{matrix}$$

marginal probabilities

$\boldsymbol{p}_w = \boldsymbol{P}_{wd}\,\boldsymbol{1}$ $\quad$ $\boldsymbol{p}_d = \boldsymbol{P}_{wd}^{\mathrm{T}}\,\boldsymbol{1}$

$$p_{w_1,w_2} = \sum_d p_{w_1|d,\cancel{w_2}}\, p_{w_2,d}$$

$$\boxed{\boldsymbol{P}_{ww} = \boldsymbol{P}_{wd}\,\mathrm{diag}(\boldsymbol{p}_d)^{-1}\,\boldsymbol{P}_{wd}^{\mathrm{T}}}$$

$\boldsymbol{p}_w = \boldsymbol{P}_{ww}\,\boldsymbol{1}$

projection on words

projection on documents

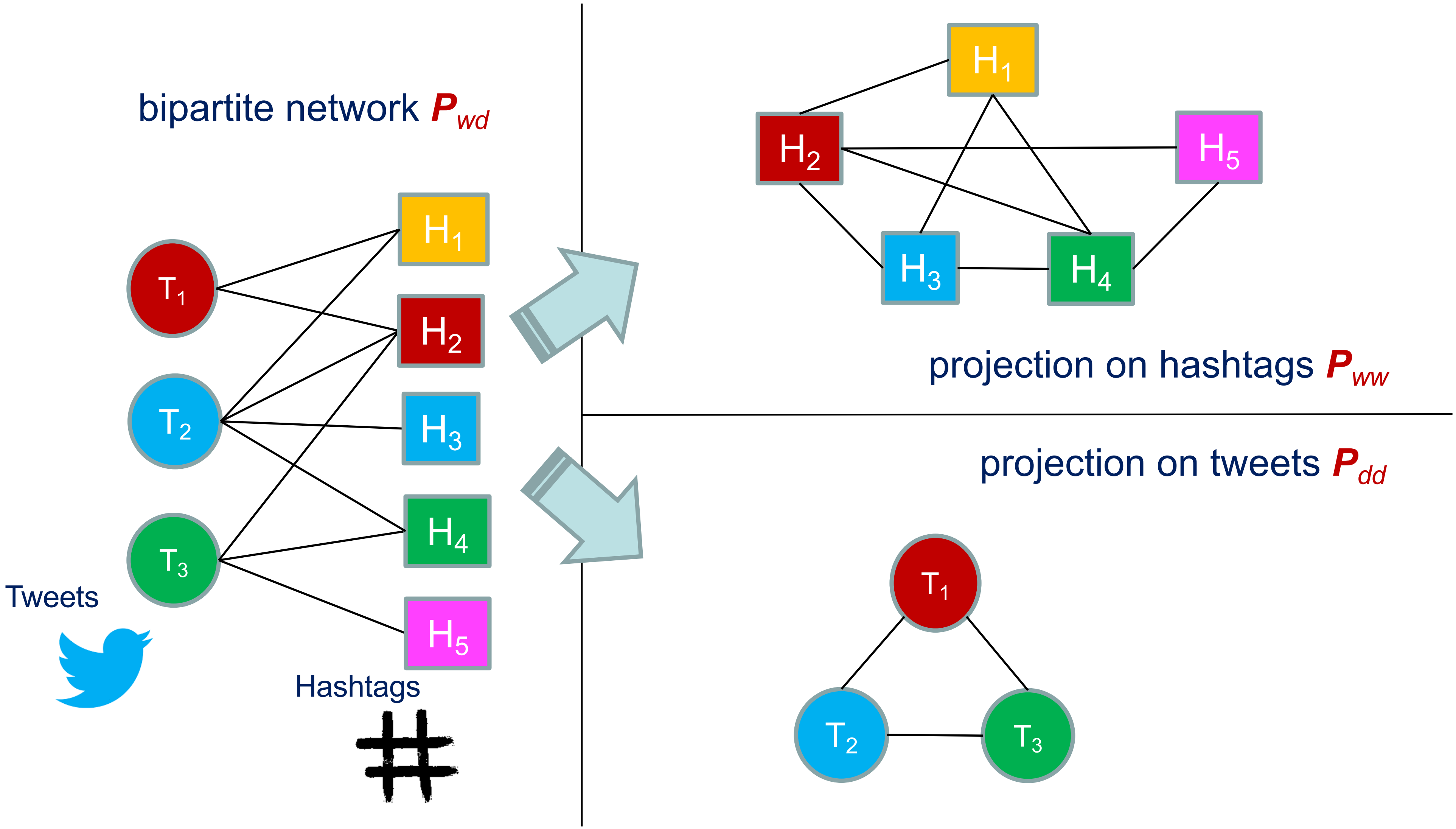
$$p_{d_1,d_2} = \sum_w p_{d_1|w,\cancel{d_2}}\, p_{d_2,w}$$

$$\boxed{\boldsymbol{P}_{dd} = \boldsymbol{P}_{wd}^{\mathrm{T}}\,\mathrm{diag}(\boldsymbol{p}_w)^{-1}\,\boldsymbol{P}_{wd}}$$

$\boldsymbol{p}_d = \boldsymbol{P}_{dd}\,\boldsymbol{1}$

# Bipartite and projected networks

a comparison



# The role of TF-IDF

## term frequency – inverse document frequency

term frequency = frequency (probability) of the word in the document

inverse document frequency = (log) fraction of documents that contain the word

$$\text{TF-IDF}_{w|d} = p_{w|d} \cdot -\log\left(\frac{\sum_d (nwd > 0)}{D}\right)$$

- An heuristic
- Punishes words that appear in many documents
- Enhances words that are document specific

$$\boldsymbol{T}_{wd} = \textbf{TF-IDF}_{w|d}\ \mathrm{diag}(\textbf{tf-idf}_d)^{-1} \cdot \mathrm{diag}(\boldsymbol{p}_d)$$

→ $\boldsymbol{T}_{ww}$, $\boldsymbol{T}_{dd}$

to be used in place of $\boldsymbol{P}_{wd}$

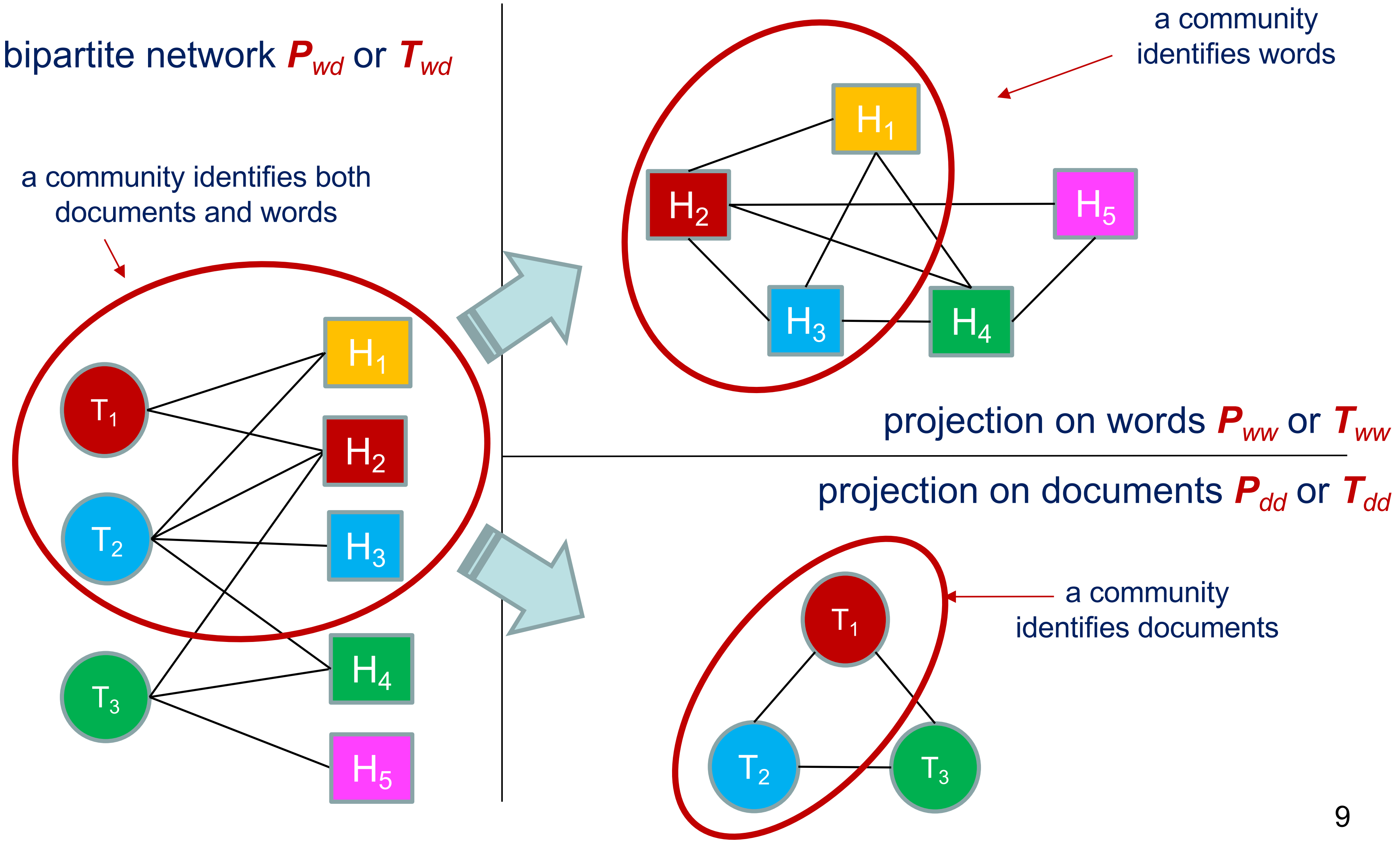
normalization vector $\textbf{tf-idf}_d = \textbf{TF-IDF}_{w|d}^{T}\ \boldsymbol{1}$ (to guarantee that columns sum up to 1)

# Topic detection

i.e., community detection in semantic networks

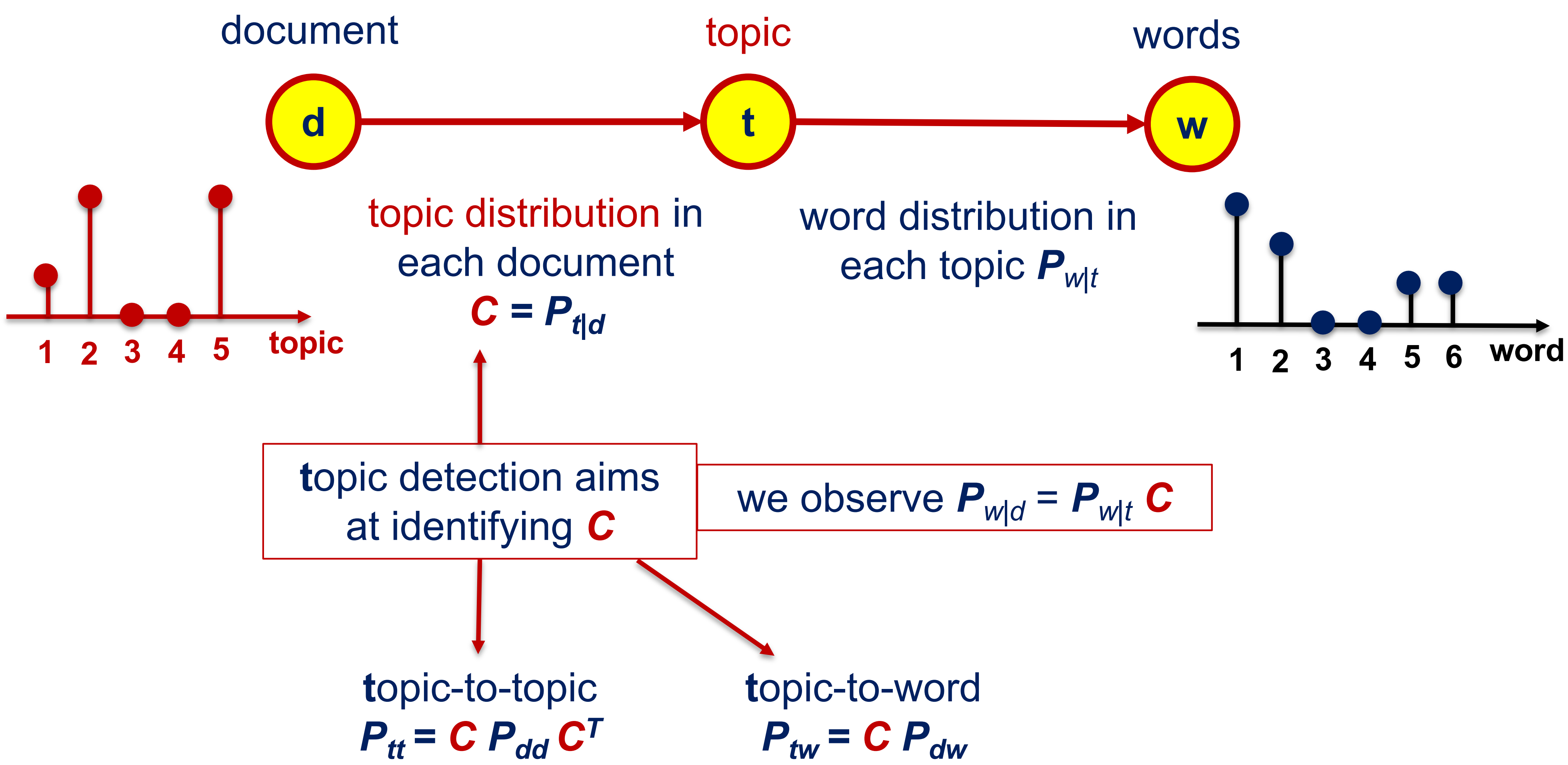
# Topic detection

## in bipartite and projection networks



# The reference model

## Under the presence of topics



# Modularity and normalized cut

## a wrap-up in topic detection

***C*** topic assignment to be assessed for quality

$$\boldsymbol{P}_{tt} = \boldsymbol{C}\, \boldsymbol{P}_{dd}\, \boldsymbol{C}^T$$

can be interpreted as a probability matrix linking topics, its entries are the sum of the links of **A** from topic i to topic j

| | | |
|---|---|---|
| $P_{11}$ | $P_{12}$ | $P_{13}$ |
| $P_{21}$ | $P_{22}$ | $P_{23}$ |
| $P_{31}$ | $P_{32}$ | $P_{33}$ |

$$\boldsymbol{p}_t = \boldsymbol{P}_{tt}\, \boldsymbol{1}$$

can be interpreted as the probability vector of topics

modularity

$$Q = \sum_t (P_{tt} - p_t^2) < 1$$

to be maximized

normalized cut

normalized version

$$\text{Ncut} = 1 - \frac{\sum_t P_{tt}/p_t}{\sum_t 1} > 0$$

to be minimized

PageRank vector (ranking of documents)

$$\boldsymbol{r} = (1-c)\, \boldsymbol{P}_{d|d}\, \boldsymbol{r} + c\, \boldsymbol{1}/N$$

Here $\boldsymbol{c}_i$ is the $i$th row of $\boldsymbol{C}$

$$\boldsymbol{P}_{d|d} = \boldsymbol{P}_{dd}\, \mathrm{diag}^{-1}(\boldsymbol{p}_d)$$

$$q_i = \left(1 - (1-c)\frac{\boldsymbol{c}_i \boldsymbol{1}}{N}\right) \boldsymbol{z}_i \boldsymbol{1} - c\, \boldsymbol{c}_i \boldsymbol{P}_{d|d} \boldsymbol{z}_i^T$$

$$\boldsymbol{z}_i = \boldsymbol{c}_i\, \mathrm{diag}(\boldsymbol{r})$$

$$\boxed{\mathrm{InfoMap} = f(\boldsymbol{q}) + \sum_i f([q_i, \boldsymbol{z}_i])}$$

entropy function

$$f(\boldsymbol{x}) = -\sum_i x_i \log\left(\frac{x_i}{\sum_j x_j}\right)$$

normalized version

$$\boxed{\frac{\mathrm{InfoMap}}{f(\boldsymbol{r})} - 1}$$

to be minimized

# Normalized mutual information

## a wrap-up in topic detection

statistical dependencies about words and topics

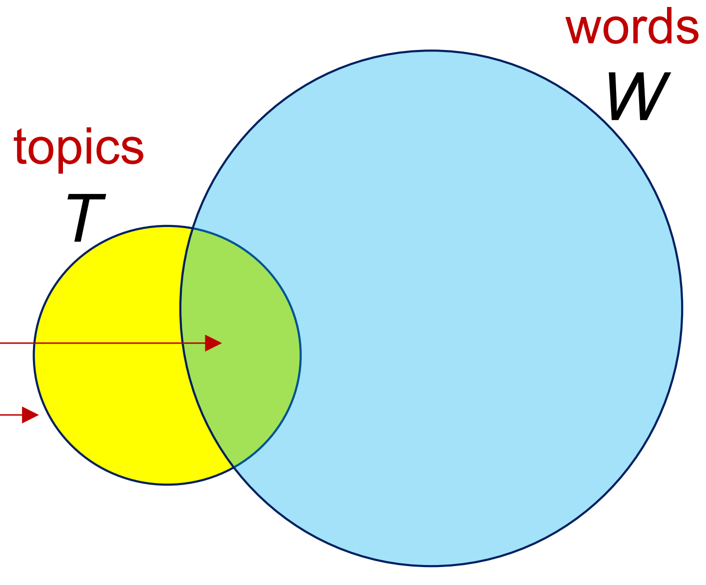
$$\boldsymbol{P}_{\mathrm{wt}} = \boldsymbol{P}_{wd}\, \boldsymbol{C}^{\mathrm{T}}$$

probability of a topic

$$\boldsymbol{p}_{\mathrm{t}} = \boldsymbol{P}_{wt}^{\mathrm{T}}\, \boldsymbol{1}$$

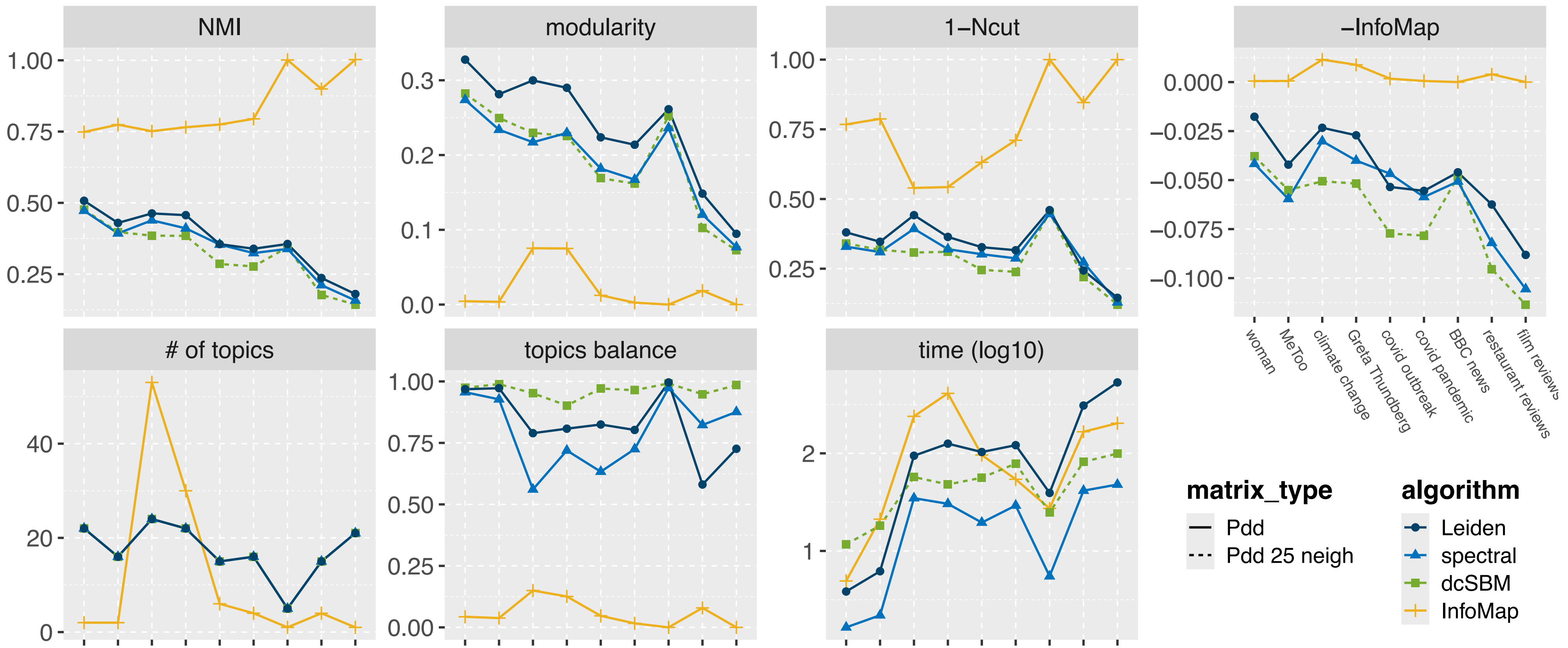
fraction of knowledge related to the topic that is explained by words (equal to 1 if topics use different words)

$$\mathrm{NMI} = \frac{\mathrm{I(W;T)}}{\mathrm{H(T)}}$$



# Louvain, InfoMap and Spectral clust.

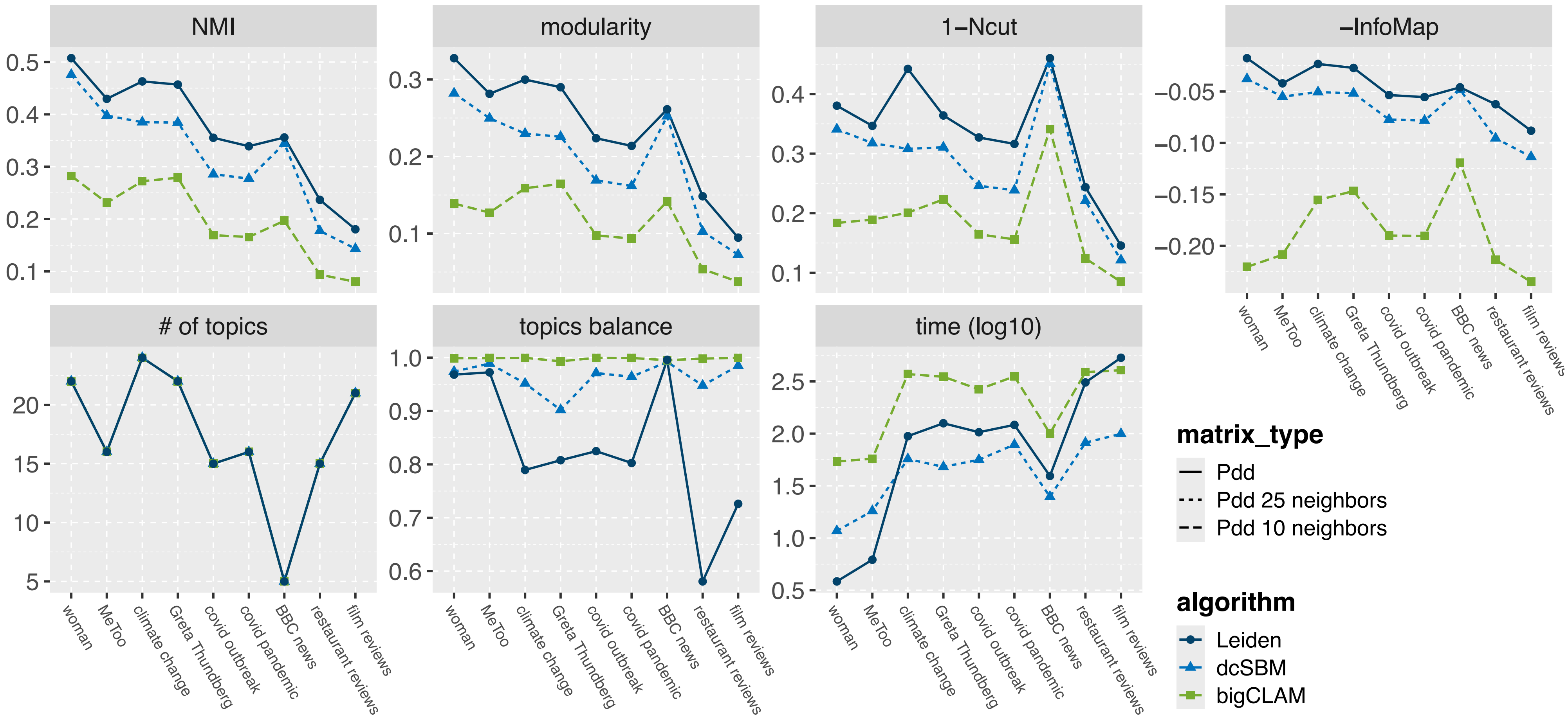
## On a semantic network



# BigClam and SBMs at work

## On a semantic network

### SBM approaches



# Other approaches

## about topic detection

- Non-negative matrix factorization (NMF)
- Latent Dirichlet allocation (LDA)
- Variational auto-encoders (VAE)
- Embeddings and BERTopic

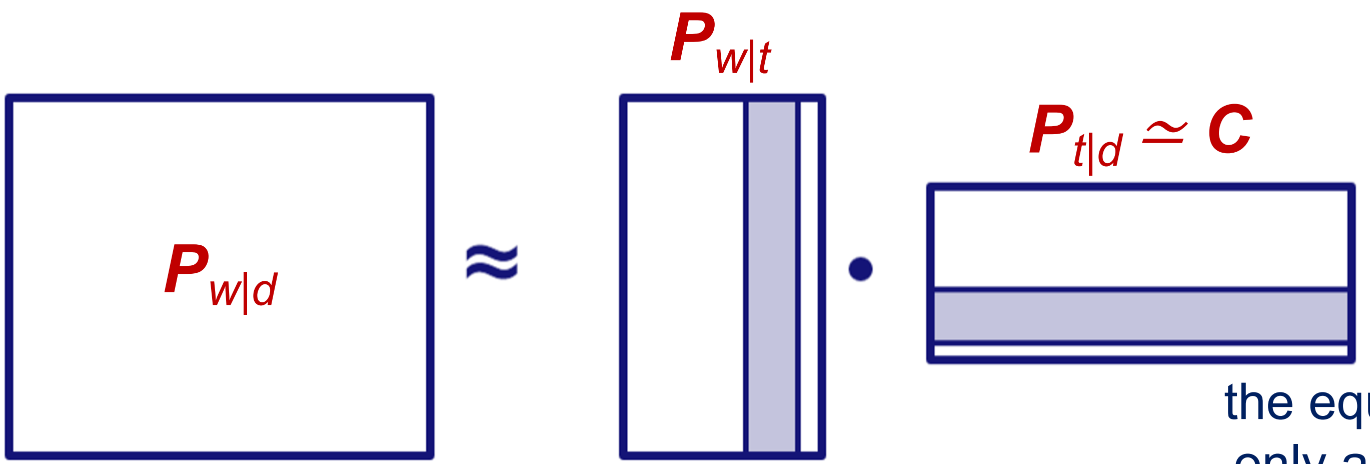
# Non-negative Matrix Factorization

and its application to topic detection

Università degli Studi di Padova

# NMF = nonnegative matrix factorization

rationale




the equivalence is only approximate

underlying model $d \to t \to w$

each document is associated to a topic distribution (one of the colums of $C$)

each topic is associated to a word distribution (one of the columns of $P_{w|t}$)

# NMF optimization

## Frobenius norm and generalized Kullbak-Leibler divergence

***A*** = ***P***$_{w|d}$ is column stochastic

$$\mathrm{argmin}_{\boldsymbol{W}\geq\boldsymbol{0},\boldsymbol{H}\geq\boldsymbol{0}} \sum_{ij} |A_{ij} - [\boldsymbol{WH}]_{ij}|^2$$

minimizing the Frobenius norm does not ensure a column stochastic product ***W H***

---

$$\mathrm{argmin}_{\boldsymbol{W}\geq\boldsymbol{0},\boldsymbol{H}\geq\boldsymbol{0}} \sum_{ij} A_{ij} \log\left(\frac{A_{ij}}{[\boldsymbol{WH}]_{ij}}\right) - A_{ij} + [\boldsymbol{WH}]_{ij}$$

$$f(y) = x \log\left(\frac{x}{y}\right) - x + y$$

$$f'(y) = -\frac{x}{y} + 1 = 0 \rightarrow \mathrm{y} = x$$

minimizing the generalized Kullback-Leibler divergence ensures a column stochastic product ***W H***

Ho & Van Dooren. "Non-negative matrix factorization with fixed row and column sums." (2008)

```
from sklearn.decomposition import NMF
Pwgd = Pwd/Pwd.sum(axis=0).flatten()
```

wisely initialize
for best
performance

run on different number of topics, then choose
the best fit, e.g., according to modularity

choose generalized
Kullback-Leibler
divergence, and the
related solver

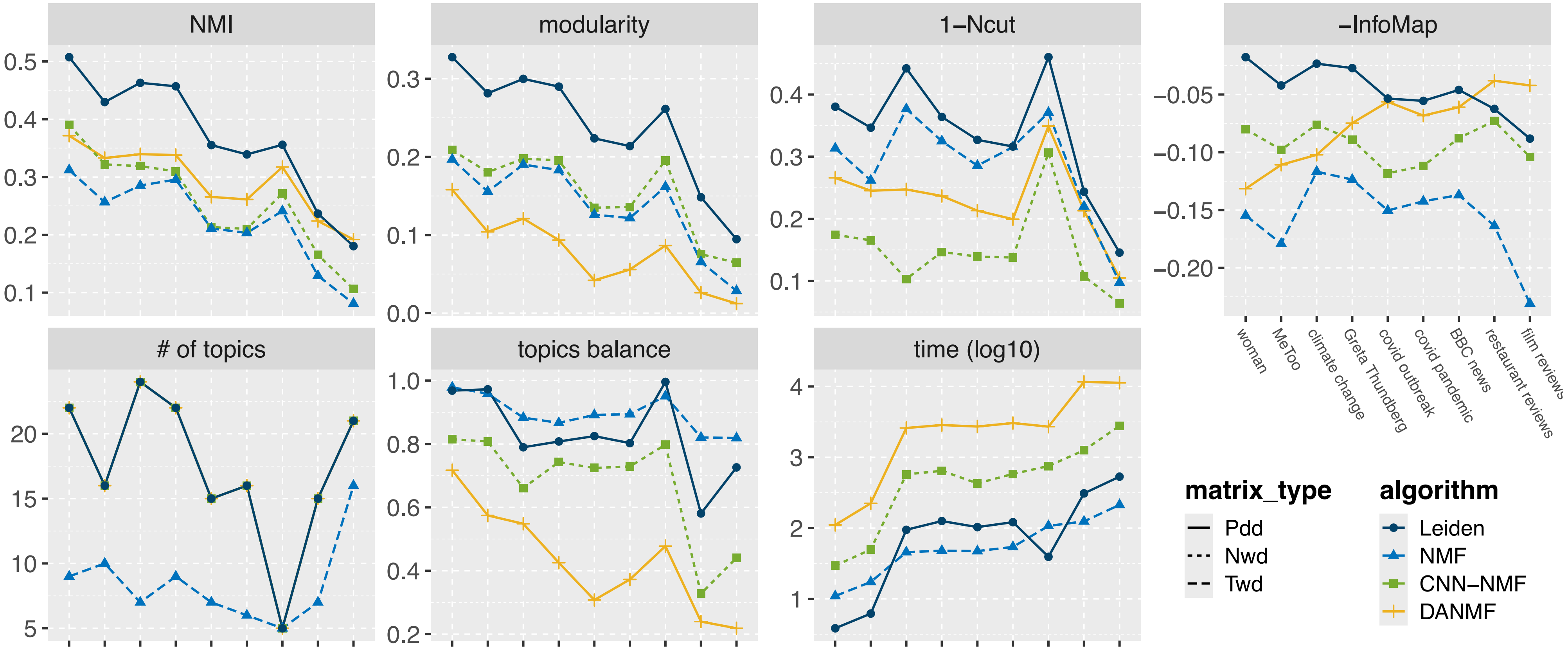
```
# fit nmf model X = W*H
model = NMF(n_components=i, init='nndsvd',
            solver='mu', beta_loss='kullback-leibler')
W = model.fit_transform(Pwgd)
H = sps.csr_matrix(model.components_)
# column normalized versions
H = sps.diags(W.sum(axis=0).flatten())*H # Ptgd
W = W/W.sum(axis=0).flatten() # Pwgt
# community assignment C
C = sps.csr_matrix(np.transpose(H/H.sum(axis=0).flatten()))
```

need to make W
column stochastic,
to have H column
stochastic too

force column stochasticity in H (not needed though)

# NMF at work

## On a semantic network



# Takeaways
## on NMF

- Naturally provides a soft topic assignment
- NMF – not strikingly good

  probably due to the fact that we want to express
  a sparse matrix through an eigenvector-like product
  with few eigenvectors (the fit is far from ideal)

- Comparison – with Louvain

  much weaker

- Complexity – generally slow

  need to test it for different numbers of topics ☹
  fast for fixed topic number

# Latent Dirichlet allocation

LDA = a stochastic model for topic detection

# LDA model

Blei,, Ng, Jordan. "Latent dirichlet allocation." (2003)

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf?ref=https://githubhelp.com

document

topics vector length $N_d$

words vector length $N_d$

$d \rightarrow \mathbf{t}_d \rightarrow \mathbf{w}_d$

topics distribution

$p(t_{d,n} = j \mid \boldsymbol{\theta}_d) = \theta_{d,j}$

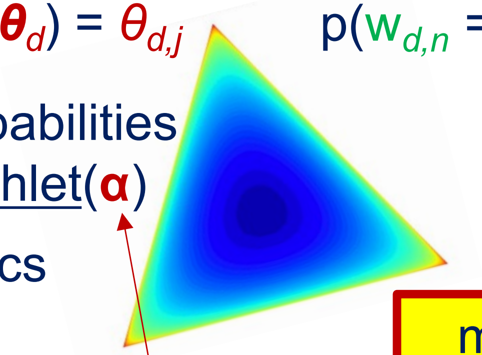
words distribution

$p(w_{d,n} = i \mid \boldsymbol{\beta}, t_{d,n} = j) = \boldsymbol{\beta}_{i,j}$

topics probabilities

$\boldsymbol{\theta}_d \sim$ Dirichlet($\boldsymbol{\alpha}$)

K topics



$\boldsymbol{\theta}_d$ is the topic assignment $\boldsymbol{C} = \boldsymbol{P}_{t|d}$

the number of topics K is fixed

vector $\boldsymbol{\alpha}$, to be estimated, is the topic probability $\boldsymbol{p}_t = \boldsymbol{\alpha}/|\boldsymbol{\alpha}|_1$

matrix $\boldsymbol{\beta}$ , to be estimated, is the probability of words given a topic, $\boldsymbol{P}_{w|t} = \boldsymbol{\beta}$

# LDA optimization

can be solved using variational inference = suboptimum approach

topics assignment probability (Dirichlet)

$$p(\boldsymbol{\theta}_d |\, \boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\sum_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} [\theta_{d,k}]^{\alpha_k - 1}$$

words probability

$$p(\boldsymbol{w}_d |\, \boldsymbol{\beta}, \boldsymbol{\theta}_d) = \prod_{n=1}^{N_d} [\boldsymbol{\beta}\, \boldsymbol{\theta}_d]_{w_{d,n}}$$

this dependence between $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ is the trickiest part

overall probability

$$p(\text{corpus} \,|\, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{d} \int p(\boldsymbol{w}_d |\, \boldsymbol{\beta}, \boldsymbol{\theta}_d)\, p(\boldsymbol{\theta}_d |\, \boldsymbol{\alpha}) d\boldsymbol{\theta}_d$$

target optimization

$$\boxed{\operatorname{argmax}_{\alpha,\beta}\, p(\text{corpus} \,|\, \boldsymbol{\alpha}, \boldsymbol{\beta})} \Rightarrow \boxed{\begin{array}{c} \boldsymbol{C} = \boldsymbol{P}_{t|d} = \boldsymbol{\theta} \\ \boldsymbol{P}_{wt} = \boldsymbol{\beta}\, \mathrm{diag}(\boldsymbol{\alpha}/|\boldsymbol{\alpha}|_1) \end{array}}$$

this is what we get

# LDA in Python

## sklearn package

```
from sklearn.decomposition import LatentDirichletAllocation
```

```
# fit lda model
lda = LatentDirichletAllocation(n_components=i,
                                learning_method="batch")
lda.fit(Mwd.T)
# community assignment C = Ptgd'
C = sps.csr_matrix(lda.transform(Mwd.T))
```

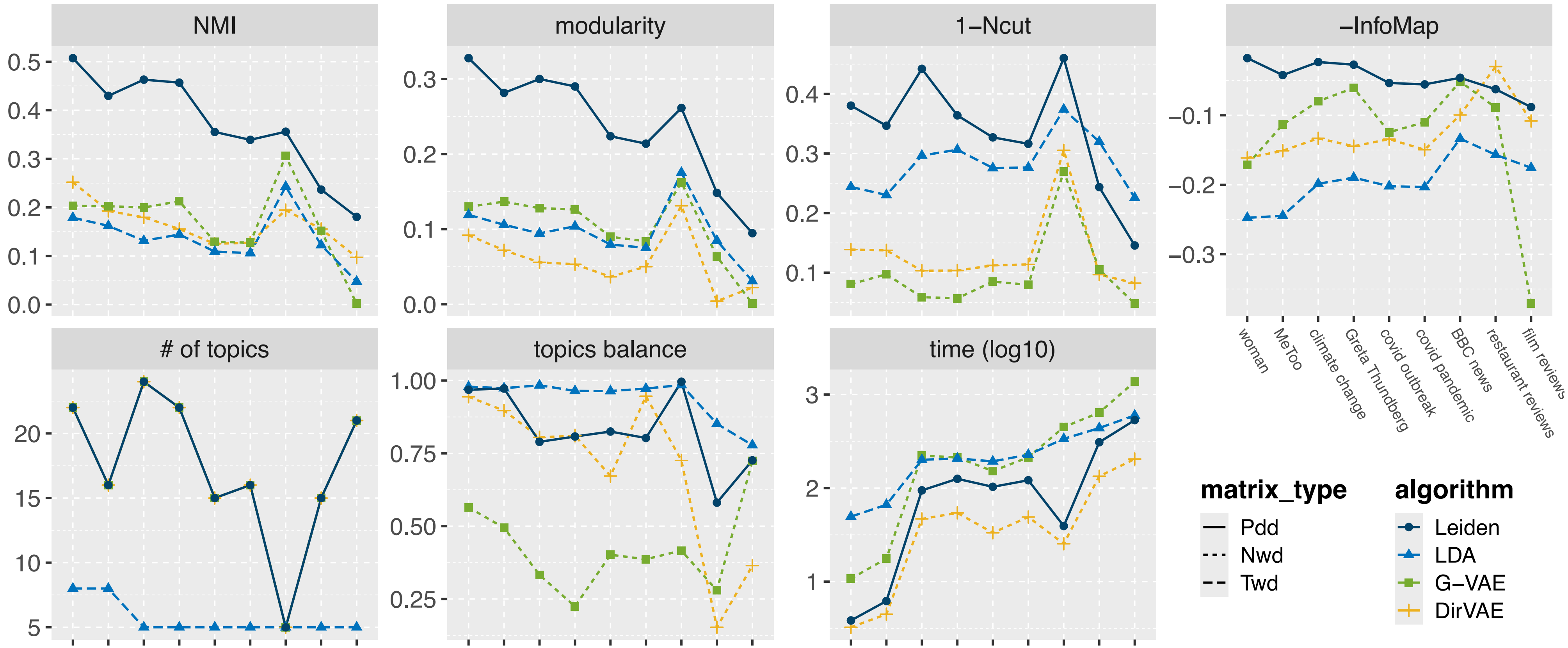
initialise and fit model

LDA

extract topic assignment

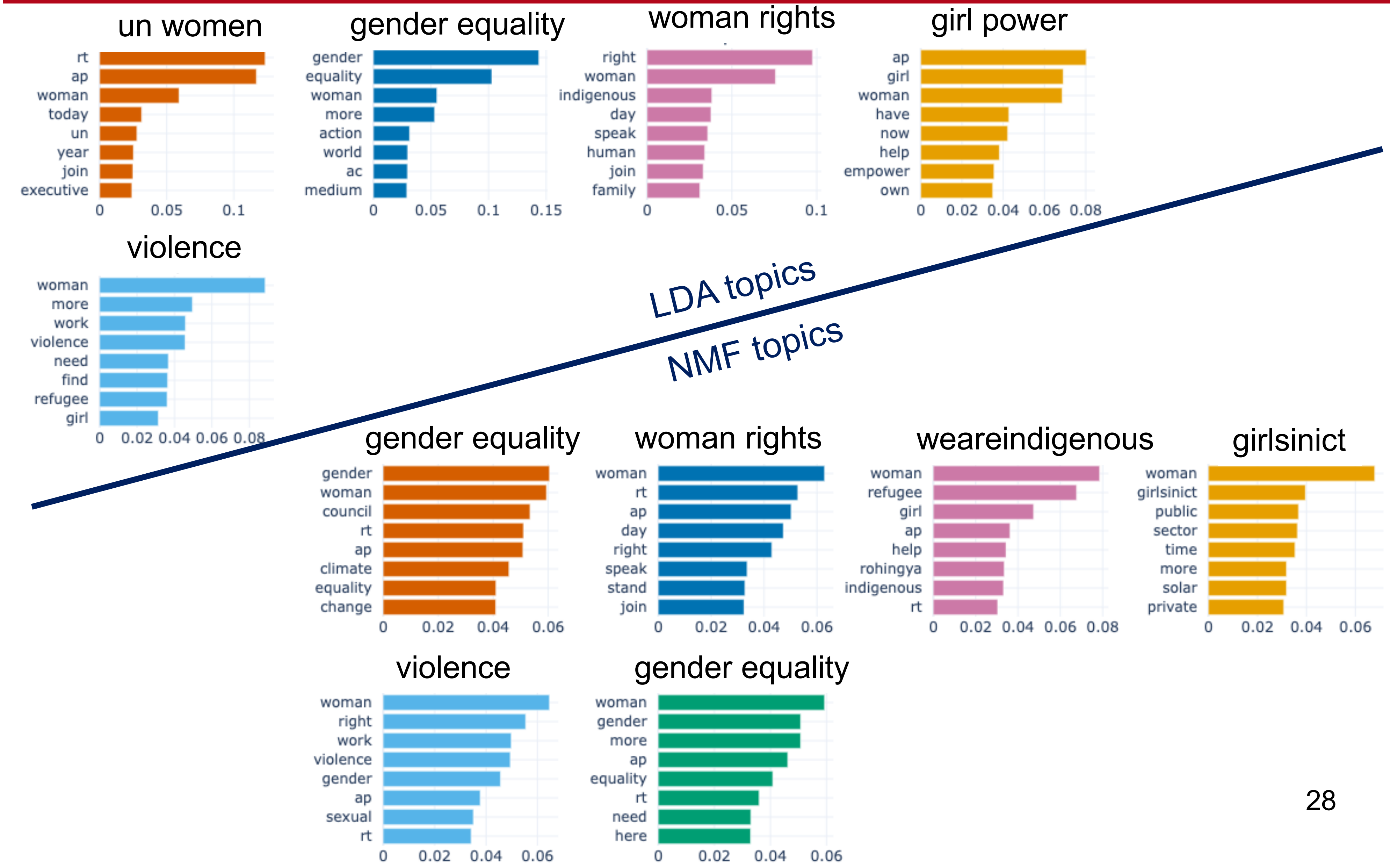
# LDA at work

## On a semantic network



# A comparison

## NMF versus LDA topics



# Takeaways
## on LDA

- Naturally provides a soft topic assignment

- LDA – not strikingly good
  - same eigenvector-like product as NMF
  - worse than NMF … known issue ☹
  - probably due to the Dirichlet assumption (questionable)
  - and the variational inference (suboptimum approach)

- Comparison – with Louvain
  - much weaker

- Complexity – generally slow
  - need to test it for different numbers of topics ☹
  - fast for fixed topic number

# Variational Auto Encoders

an application to topic analysis

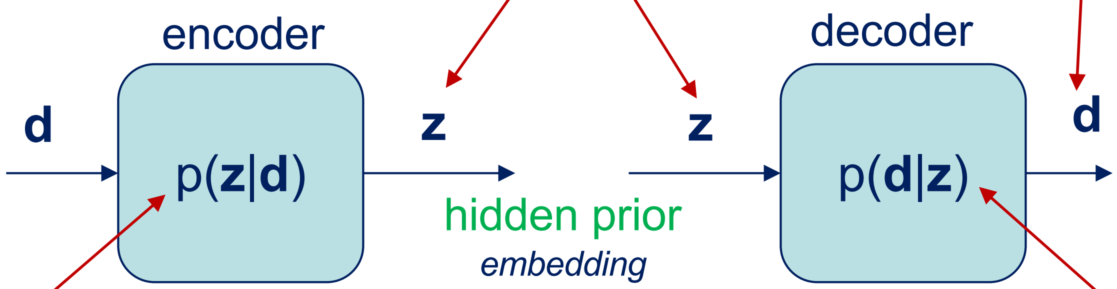
Università degli Studi di Padova

# Variational Auto-Encoders

Kingma, Welling, "Auto-encoding variational Bayes," (2013)

the hidden prior in our case
is the topic (distribution)

the output in our case
is the document
(collection of words)




inference model
*posterior probabilities*
*to be estimated*

generative model
*given*

the (stochastic) model
explains how a
document is generated
from a topic (distribution)

but we are interested in the inverse link that, given
a document tells what topic it is associated with

$$p(\boldsymbol{z}|\boldsymbol{d}) = \frac{p(\boldsymbol{d}|\boldsymbol{z})\, p(\boldsymbol{z})}{p(\boldsymbol{d})} \cong q(\boldsymbol{z}|\boldsymbol{d})$$

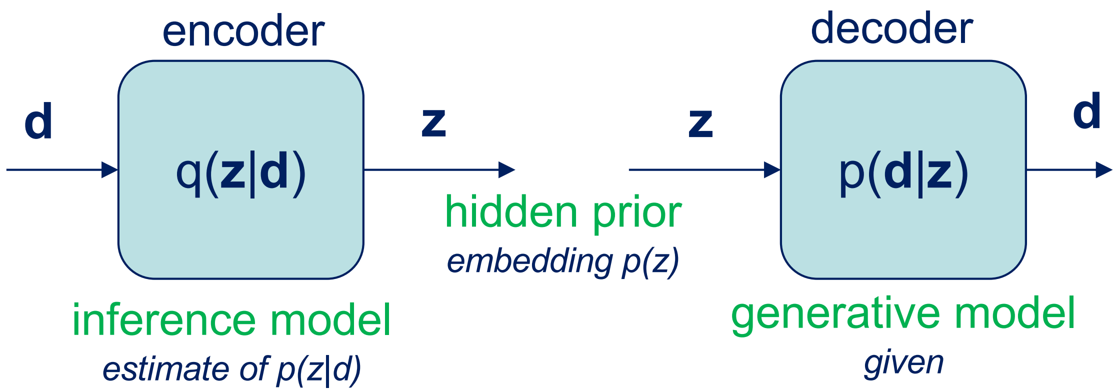
impossible to know
in the closed form

needs an a-priori
model for the
embedding

is approximated
by a simple
alternative model

# VAE optimization rationale

## ELBO = evidence lower bound




ELBO

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{d}) \leq \log p_\theta(\boldsymbol{d})$$

$$\begin{aligned}
\mathcal{L}_{\theta,\phi}(\boldsymbol{d}) &= \log p_\theta(\boldsymbol{d}) - D_{\mathrm{KL}}\left(q_\phi(\boldsymbol{z}|\boldsymbol{d}) \middle\| p_\theta(\boldsymbol{z}|\boldsymbol{d})\right) \\
&= \int dz\, q_\phi(\boldsymbol{z}|\boldsymbol{d}) \log\left(\frac{p_\theta(\boldsymbol{z},\boldsymbol{d})}{q_\phi(\boldsymbol{z}|\boldsymbol{d})}\right) \\
&= \underbrace{\int dz\, q_\phi(\boldsymbol{z}|\boldsymbol{d}) \log\left(p_\theta(\boldsymbol{d}|\boldsymbol{z})\right)}_{\mathcal{L}_1} + \underbrace{\int dz\, q_\phi(\boldsymbol{z}|\boldsymbol{d}) \log\left(\frac{p_\theta(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{d})}\right)}_{\mathcal{L}_2}
\end{aligned}$$

to be maximized wrt parameters $\theta$ and $\phi$ provides fitting on p(**z**), p(**d**|**z**), and q(**z**|**d**)

a-priori model (given)

inference model (approximate)

generative model (given)

# L2 ELBO function

usually has a compact expression

$$\underbrace{\int dz\, q_\phi(\boldsymbol{z}|\boldsymbol{d}) \log\left(\frac{p_\theta(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{d})}\right)}_{\mathcal{L}_2}$$

both should have a simple parametrization on $\theta$ and $\phi$

e.g., the Gaussian case

$$p_\theta(\boldsymbol{z}) = \frac{1}{\sqrt{\det\left(2\pi\, \mathrm{diag}\left(\boldsymbol{\sigma}_\theta^2\right)\right)}} \exp\left(-½(\boldsymbol{z}-\boldsymbol{\mu}_\theta)^T \mathrm{diag}^{-1}\left(\boldsymbol{\sigma}_\theta^2\right)(\boldsymbol{z}-\boldsymbol{\mu}_\theta)\right)$$

$$q_\phi(\boldsymbol{z}|\boldsymbol{d}) = \frac{1}{\sqrt{\det\left(2\pi\, \mathrm{diag}\left(\boldsymbol{\sigma}_\phi^2(\boldsymbol{d})\right)\right)}} \exp\left(-½\left(\boldsymbol{z}-\boldsymbol{\mu}_\phi(\boldsymbol{d})\right)^T \mathrm{diag}^{-1}\left(\boldsymbol{\sigma}_\phi^2(\boldsymbol{d})\right)\left(\boldsymbol{z}-\boldsymbol{\mu}_\phi(\boldsymbol{d})\right)\right)$$

$$\mathcal{L}_2(\theta,\phi) = ½ \sum_i 1 + \log\left(\frac{\sigma_{\phi,i}^2(\boldsymbol{d})}{\sigma_{\theta,i}^2}\right) - \frac{\sigma_{\phi,i}^2(\boldsymbol{d})}{\sigma_{\theta,i}^2} - \frac{\left(\mu_{\phi,i}(\boldsymbol{d})-\mu_{\theta,i}\right)^2}{\sigma_{\theta,i}^2}$$

# L1 ELBO function

## approximated through Monte Carlo estimation

$$\underbrace{\int dz\, q_\phi(z|d) \log\left(p_\theta(d|z)\right)}_{\mathcal{L}_1}$$

mostly too complex to be written in the closed form

solution: Monte Carlo approximation

$$\mathcal{L}_{\mathbf{1}}(\theta, \phi) = \frac{1}{L} \sum_{\ell=1}^{L} \log\left(p_\theta(\boldsymbol{d}|\boldsymbol{z}_\ell)\right)$$

samples generated according to the correct distribution

$$\boldsymbol{z}_\ell \sim q_\phi(\boldsymbol{z}|\boldsymbol{d})$$

e.g., the Gaussian case

$$\boldsymbol{z}_\ell = \boldsymbol{\mu}_\phi(\boldsymbol{d}) + \boldsymbol{\sigma}_\phi(\boldsymbol{d})\, \boldsymbol{n}_\ell$$

need to generate these once, then use them throughout the process

$$\boldsymbol{n}_\ell \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

# Take 1 - NVDM

Miao, Yu, Blunsom, "Neural variational inference for text processing," (2016)
http://proceedings.mlr.press/v48/miao16.pdf

here **W** builds a topic-to-word map that identifies the word distribution inside each document given the topic distribution

*word probabilities*

somehow related to a topic, but we do not know how

**z** → softmax(**Wz+b**) → $\mathbf{p}_w$ → **d**

***hidden prior***

p(**z**) ***Gaussian*** *distributed*

$\boldsymbol{z} = \boldsymbol{\mu}_1 + \boldsymbol{\sigma}_1\, \boldsymbol{n}$

*with* ***n*** *normalized Gaussian*

***document***

$p_\theta(\mathbf{d}|\mathbf{z})$ ***multinomial***

*distributed according to* $\boldsymbol{p}_w(\boldsymbol{z})$

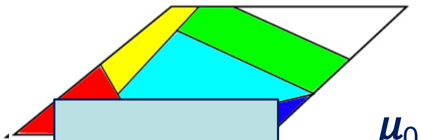
probability function for the multinomial

$\log(p_\theta(\mathbf{d}|\mathbf{z}))$

$= \mathbf{d}^T \log(\mathbf{p}_w(\mathbf{z})) + \text{const}$

***d*** *one-hot-representation*
*= number of occurrences*
*of words in the document*

*different document regions are associated with different output values*



**d** → **FFN** → $\boldsymbol{\mu}_0$, $\log \boldsymbol{\sigma}_0^2$ → **z**

***document***

*one-hot-representation of appearing words*

*e.g., 2x 2-layer, ReLu activation,* ***linear*** *output*

***hidden prior***

$q_\phi(\mathbf{z}|\mathbf{d})$ ***Gaussian*** *distributed*

$\boldsymbol{z} = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_0\, \boldsymbol{n}$

*with* ***n*** *normalized Gaussian*

# NVDM ELBO

optimization target, to be maximized

*one-hot-representation of a document = number of occurrences of words in the document*

normalized Gaussian samples

**decoder model** **encoder model**

$$\mathcal{L}(\theta,\phi) = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{m} \boldsymbol{d}_m^T \log\Big(\underbrace{\mathrm{softmax}\big(\boldsymbol{b} + \boldsymbol{W}}_{\text{decoder map}}(\boldsymbol{\mu}_0(\boldsymbol{d}_m) + \boldsymbol{\sigma}_0(\boldsymbol{d}_m)\,\boldsymbol{n}_{m,\ell})\big)\Big)$$

$$+ \tfrac{1}{2}\sum_{m}\sum_{i} 1 + \log\left(\frac{\sigma_{0,i}^2(\boldsymbol{d}_m)}{\sigma_{1,i}^2}\right) - \frac{\sigma_{0,i}^2(\boldsymbol{d}_m)}{\sigma_{1,i}^2} - \frac{\left(\mu_{0,i}(\boldsymbol{d}_m) - \mu_{1,i}\right)^2}{\sigma_{1,i}^2}$$

**a-priori model**

Not very clear where the topic is, though!

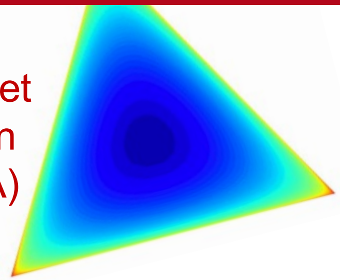
# Take 2 - DirVAE

Joo, Li, Park, Moon, "Dirichlet variational autoencoder," (2020)
https://www.sciencedirect.com/science/article/pii/S003132032030317

a multigamma distribution is introduced

**z**/sum(**z**)

true Dirichlet distribution (as in LDA)



**z** → softmax(log()) → **t** → softmax(**Wt**+**b**) $\xrightarrow{\mathbf{p}_w}$ **d**

***hidden prior***
p(**z**) ***multigamma*** *distributed with parameters* $\boldsymbol{\alpha}_1$ and $\beta$=1

***topic***
***Dirichlet*** *distributed*

***document***
$p_\theta$(**d**|**z**) ***multinomial*** *distributed*

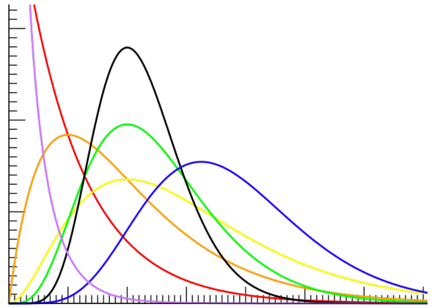
shape vector

**d** → **FFN** $\xrightarrow{\boldsymbol{\alpha}_0>0}$ **z**

***document***

*e.g., 2x 2-layer, ReLu activation,* ***softplus*** *ln(1+$e^x$) output*

***hidden prior***
$q_\phi$(**z**|**d**) ***multigamma*** *distributed with parameters* $\boldsymbol{\alpha}_0$ and $\beta$=1



# DirVAE ELBO

optimization target, to be maximized

$$\boldsymbol{f}(\boldsymbol{u},\boldsymbol{\alpha}) = \left(\boldsymbol{u}\,\boldsymbol{\alpha}\,\Gamma(\boldsymbol{\alpha}_1)\right)^{1/\boldsymbol{\alpha}}$$

approx. uniform to multigamma map

normalized uniform samples

*one-hot-representation of a document = number of occurrences of words in the document*

**decoder model**

**encoder model**

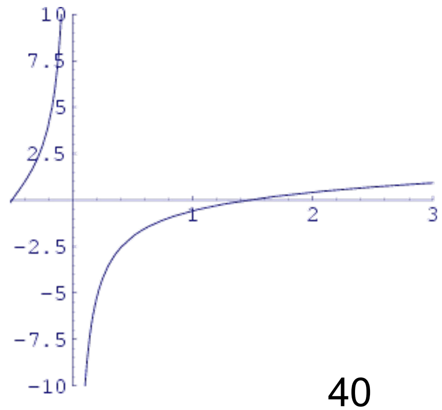
$$\mathcal{L}\left(\theta,\phi\right) = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{m}\boldsymbol{d}_m^T \log\Bigg(\underbrace{\text{softmax}\Big(\boldsymbol{b} + \boldsymbol{W}\,\text{softmaxlog}}_{\text{decoder map}}\Big(\boldsymbol{f}\big(\boldsymbol{u}_{m,\ell},\boldsymbol{\alpha}_0(\boldsymbol{d}_m)\big)\Big)\Big)\Bigg)$$

$$+\sum_{m}\sum_{i}\log\left(\frac{\Gamma\left(\alpha_{0,i}(\boldsymbol{d}_m)\right)}{\Gamma\left(\alpha_{1,i}\right)}\right) - \left(\alpha_{0,i}(\boldsymbol{d}_m) - \alpha_{1,i}\right)\psi\left(\alpha_{0,i}(\boldsymbol{d}_m)\right)$$

**a-priori model**

digamma function

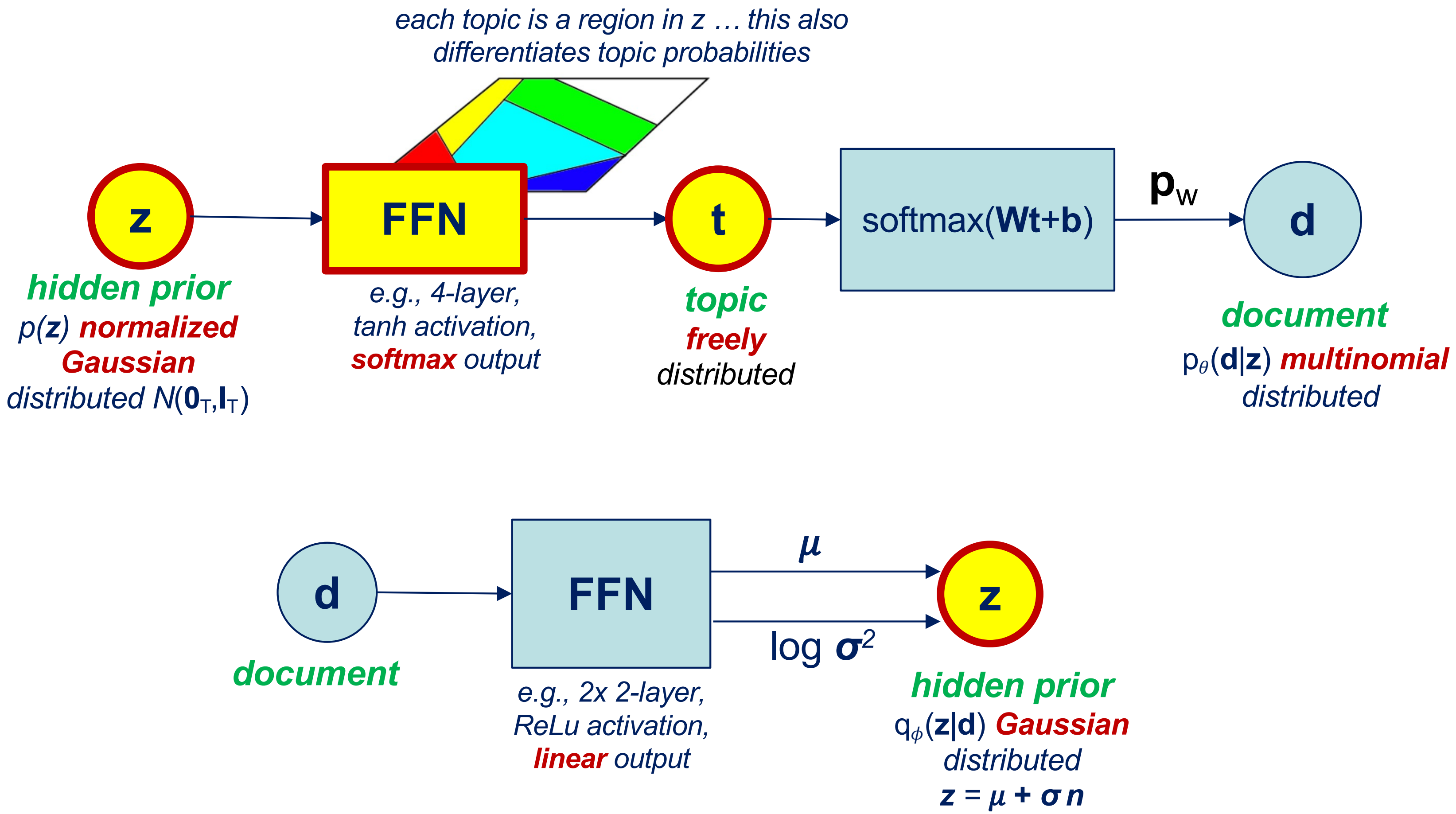
$$\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$$




Now we know where the <u>topic</u> is!

# Take 3 – NFTM

Gui, Lin, et al. "Multi task mutual learning for joint sentiment classification and topic detection," (2020) https://ieeexplore.ieee.org/document/9112648



# NFTM ELBO

optimization target, to be maximized

*one-hot-representation of a document = number of occurrences of words in the document*

normalized Gaussian samples

**decoder model**

**encoder model**

$$\mathcal{L}(\theta,\phi) = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{m} \boldsymbol{d}_m^T \log\Big(\underbrace{\text{softmax}\big(\boldsymbol{b} + \boldsymbol{W}\,\text{FFN}_1}_{\text{decoder map}}(\boldsymbol{\mu}_0(\boldsymbol{d}_m) + \boldsymbol{\sigma}_0(\boldsymbol{d}_m)\,\boldsymbol{n}_{m,\ell})\big)\Big)$$

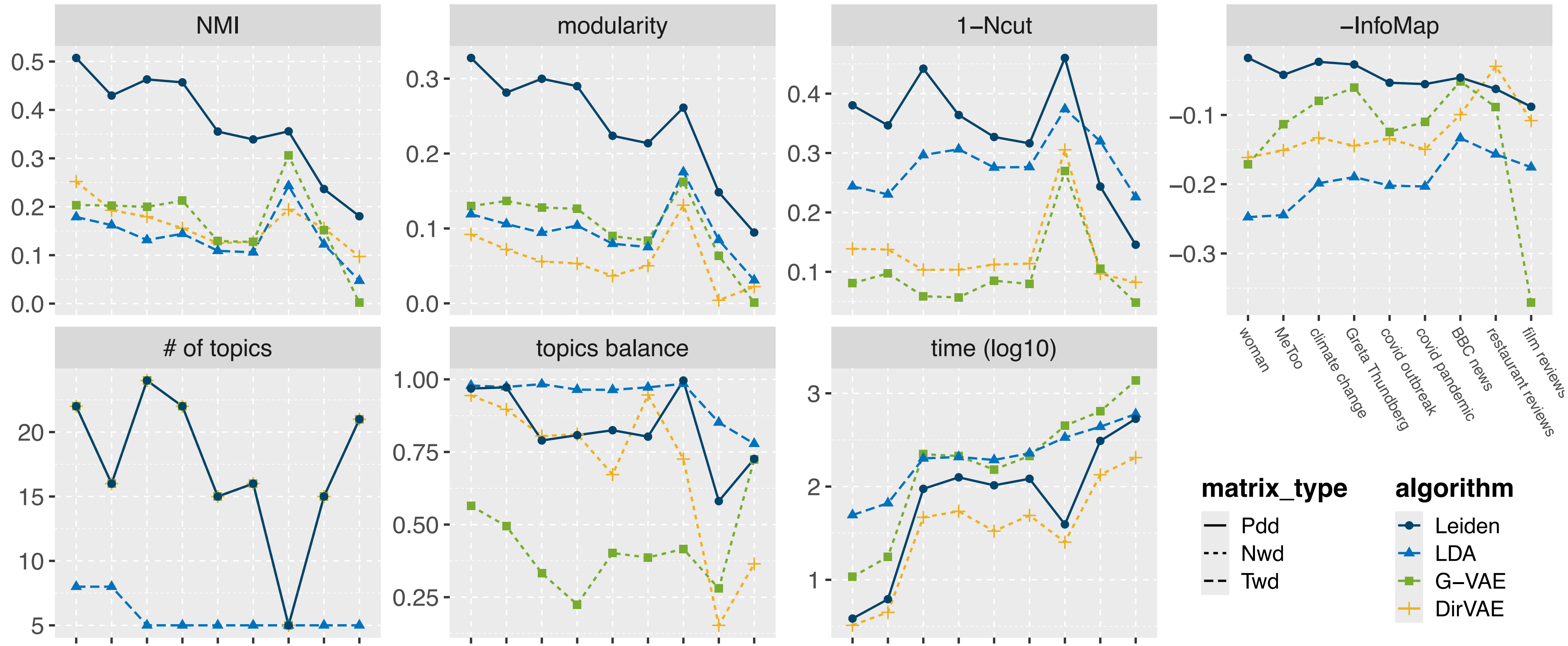
$$+\tfrac{1}{2}\sum_{m}\sum_{i} 1 + \log\big(\sigma_{0,i}^2(\boldsymbol{d}_m)\big) - \sigma_{0,i}^2(\boldsymbol{d}_m) - \big(\mu_{0,i}(\boldsymbol{d}_m)\big)^2$$

Our estimate of the topic distribution for the *m*th document!

$$\boxed{\boldsymbol{c}_m = \frac{1}{L}\sum_{\ell=1}^{L} \text{FFN}_1(\boldsymbol{\mu}_0(\boldsymbol{d}_m) + \boldsymbol{\sigma}_0(\boldsymbol{d}_m)\,\boldsymbol{n}_{m,\ell})}$$

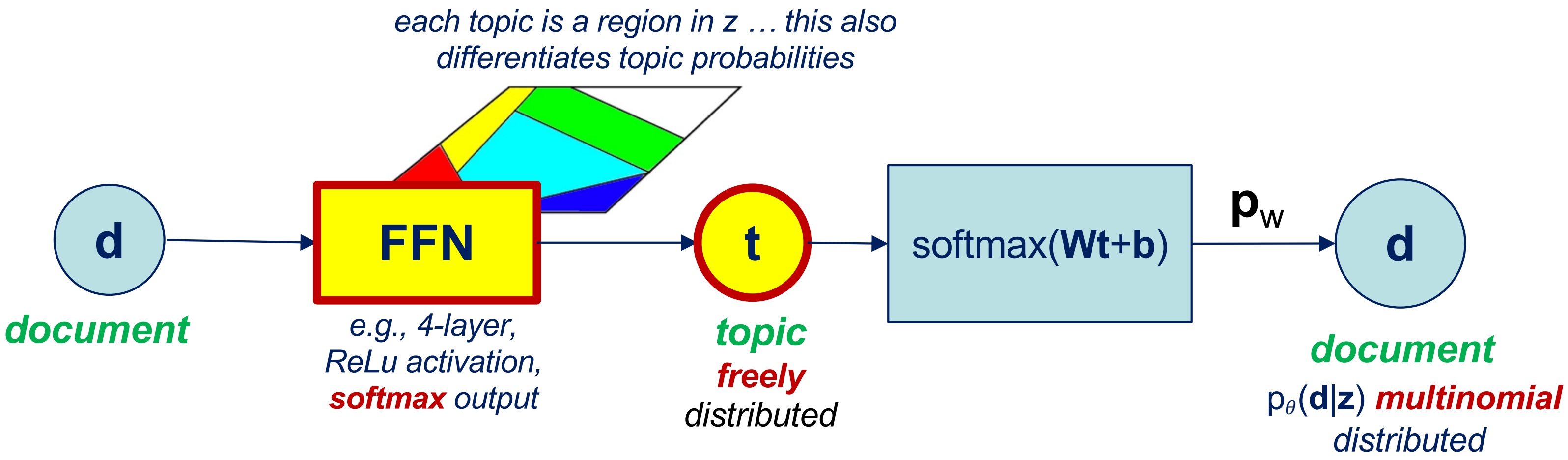
# VAE at work

## On a semantic network



UNIVERSITÀ DEGLI STUDI DI PADOVA

# NFTM relaxation
## mimics NMF




*one-hot-representation of a document = number of occurrences of words in the document*

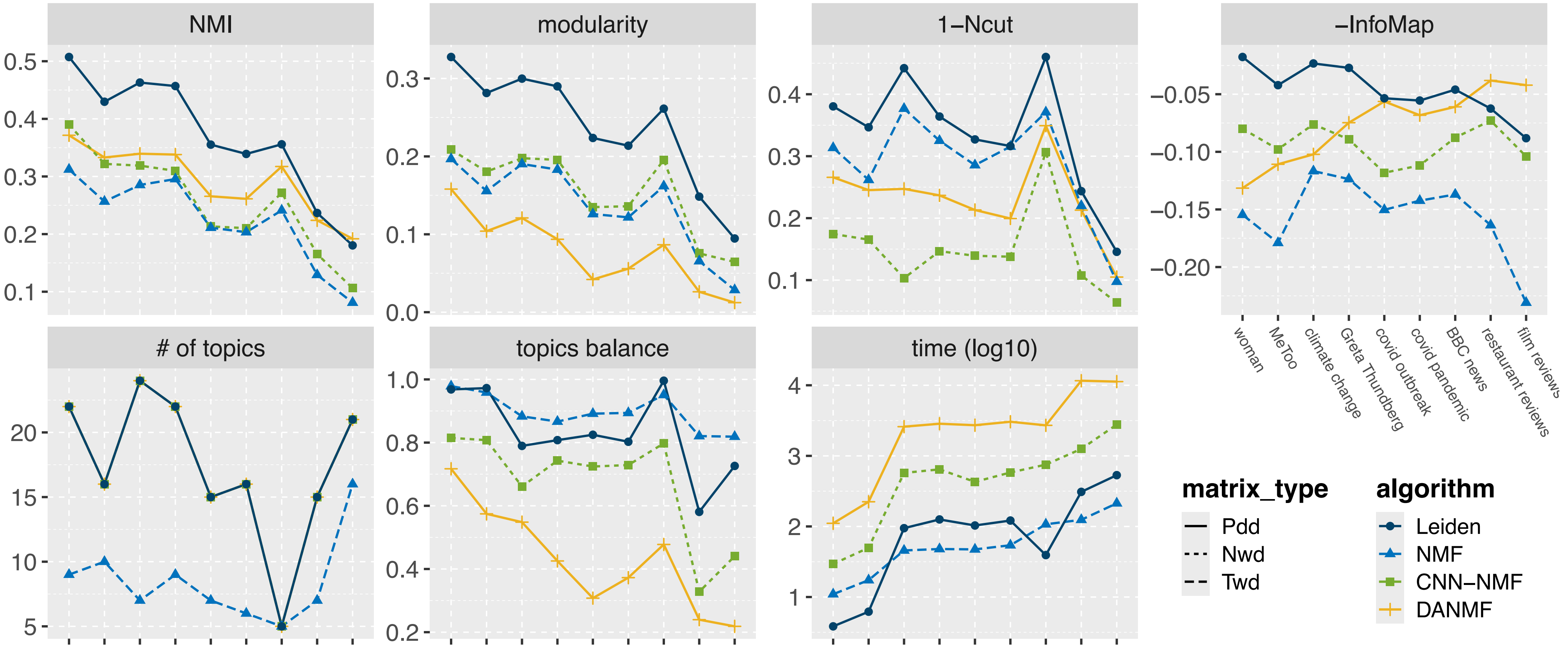
**decoder model** **encoder model**

$$\mathcal{L}\,(\theta,\phi) = \sum_m \boldsymbol{d}_m^T \log\!\big(\mathrm{softmax}(\boldsymbol{b} + \boldsymbol{W}\,\mathrm{FFN}(\boldsymbol{d}_m))\big)$$

UNIVERSITÀ DEGLI STUDI DI PADOVA

# CNN-NMF at work

## On a semantic network



# Takeaways
## on VAE applied to topic detection

- Naturally provides a soft topic assignment

- VAE – interesting approach
  - more flexible model than NMF or LDA
  - gives improvements

- Comparison – with Louvain
  - still far away
  - would be nice to see other Deep Learning approaches

… your task! ☺

# Transformer Architecture

with application to BERT, RoBERTa, OpenAI GPT

# Attention (machine learning)

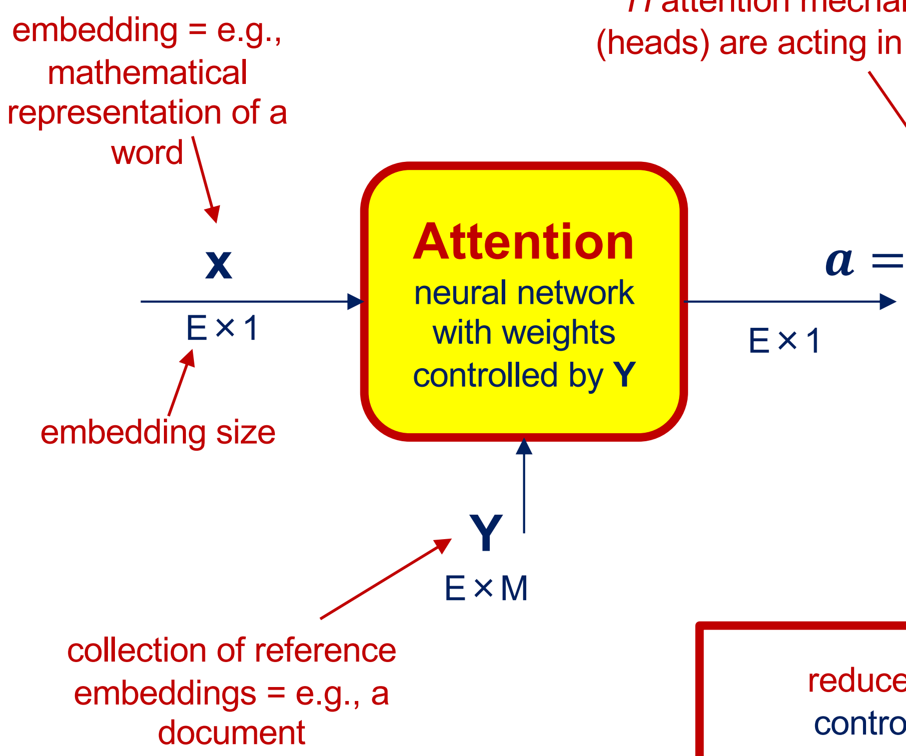
Article Talk

From Wikipedia, the free encyclopedia

In artificial neural networks, **attention** is a technique that is meant to mimic cognitive attention. This effect enhances some parts of the input data while diminishing other parts — the motivation being that the network should devote more focus to the important parts of the data, even though they may be small portion of an image or sentence. Learning which part of the data is more important than another depends on the context, and this is trained by gradient descent.

# The Attention Module

Vaswani, Ashish, et al. "Attention is all you need" (2017)




*H* attention mechanisms (heads) are acting in parallel

$$\boldsymbol{a} = \sum_{j=1}^{H} \overbrace{\boldsymbol{W}_{2j}\,\boldsymbol{Y}}^{\text{expansion}} \,\mathrm{softmax}(\overbrace{\boldsymbol{Y}^T \boldsymbol{W}_{1j}\,\boldsymbol{x}}^{\text{inner products}})$$

similar in spirit to projecting **x** onto the space spanned by **Y,** but with **softmax** normalization

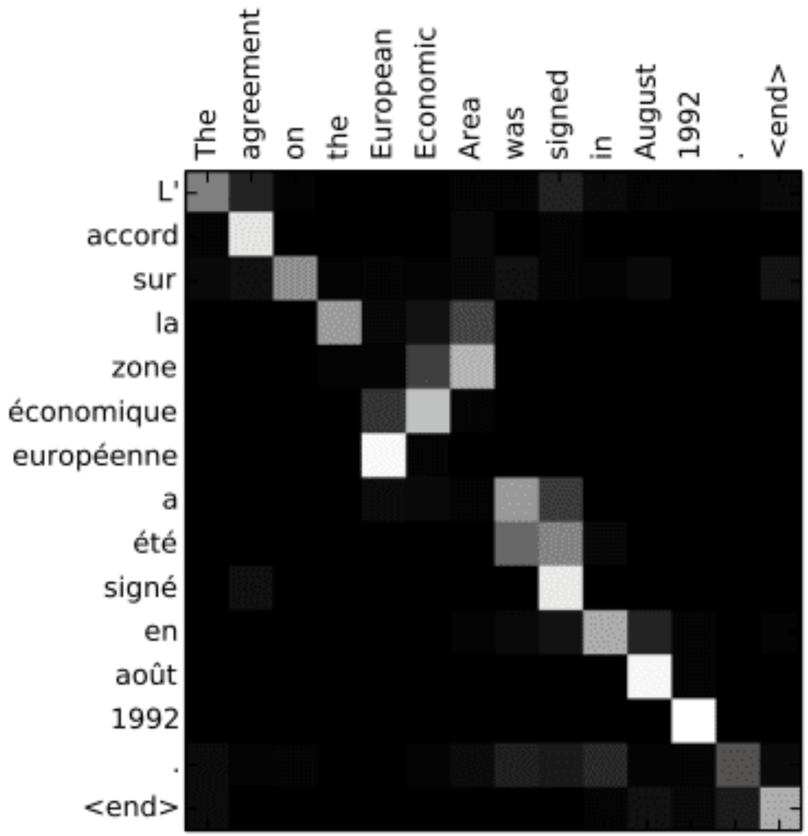
reduced-rank mappings, K<E, control the overall complexity

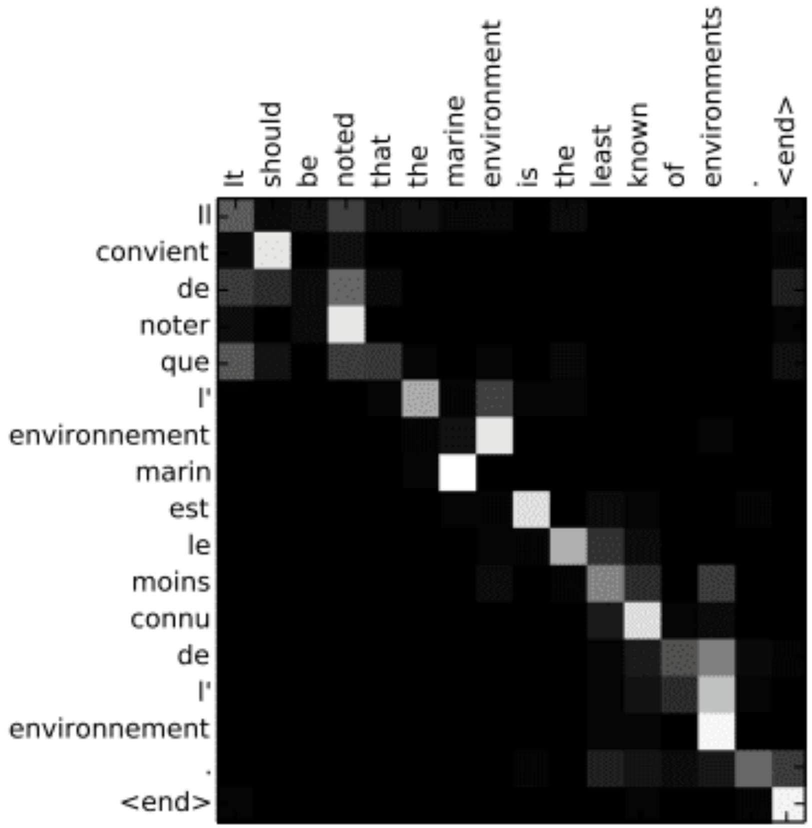
$$\underset{E \times E}{\boldsymbol{W}_{ij}} = \underset{E \times K}{\boldsymbol{V}_{ij}}\ \underset{K \times E}{\boldsymbol{U}_{ij}}$$

overall complexity 4EKH

# Visualizing Attention

## in a translation experiment (X English, Y French)

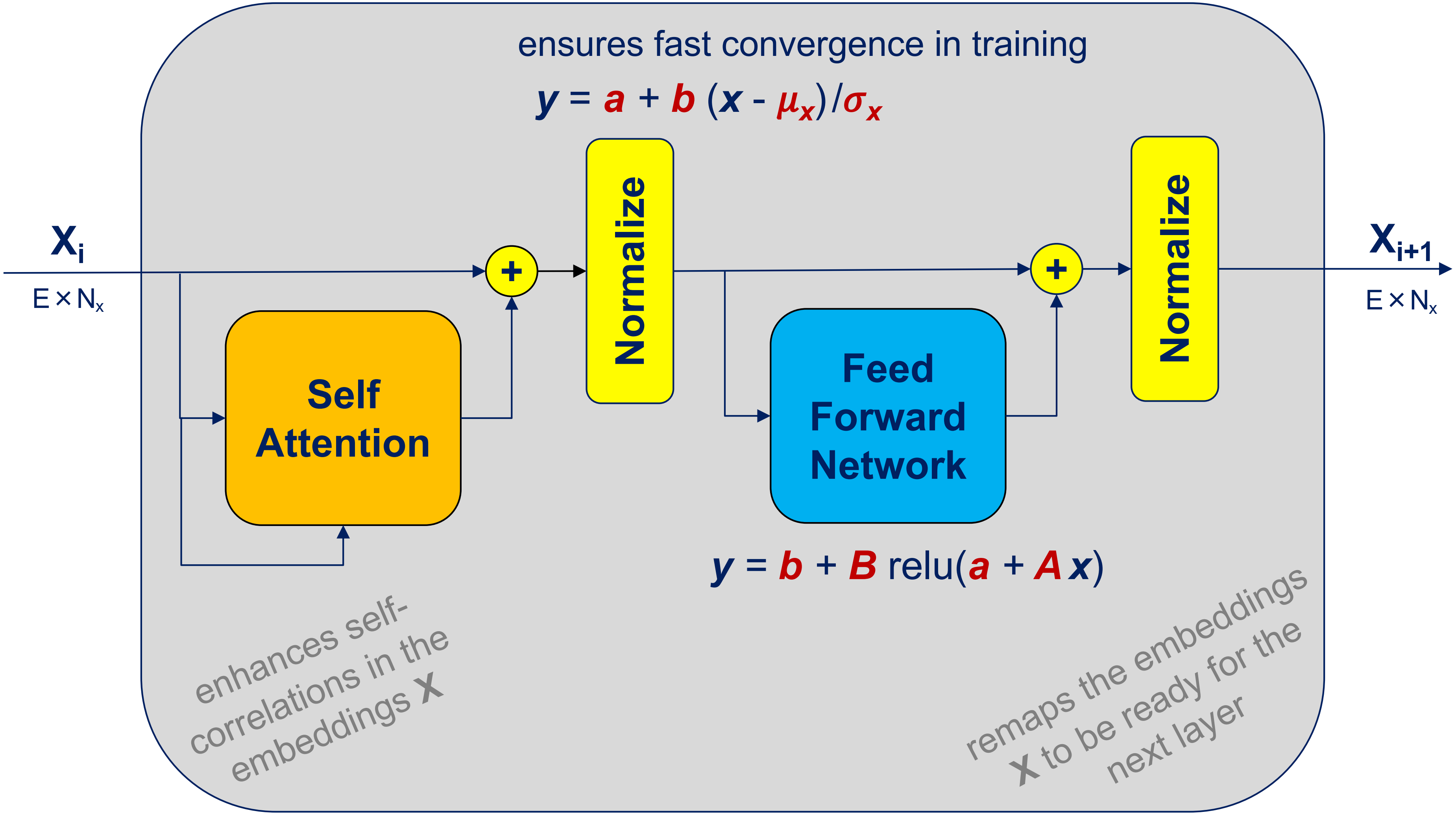



softmax($\boldsymbol{Y^T W_{1j} X}$)

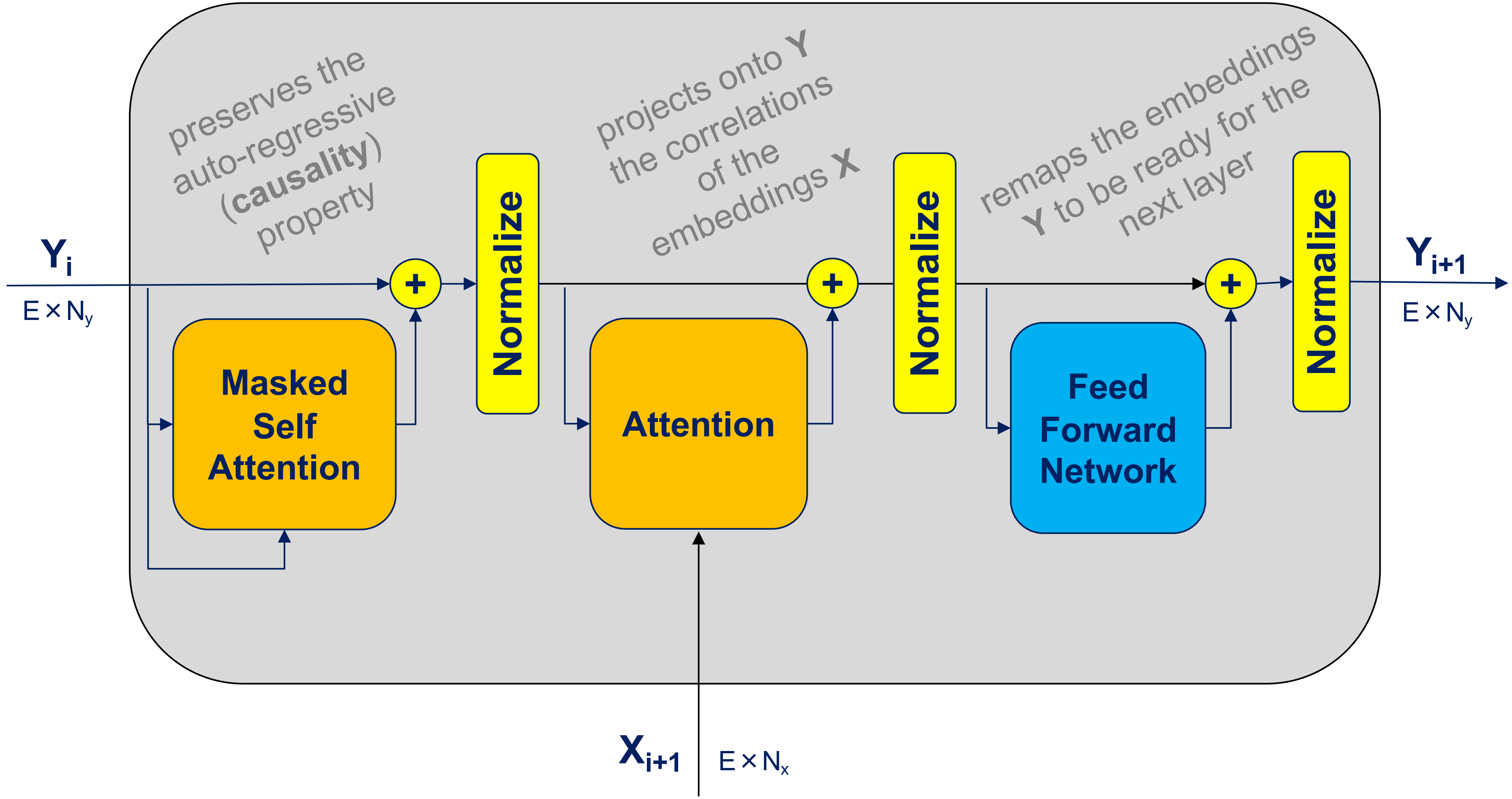
# Encoder

## a serie of multi-head self-attention modules



# Decoder

a serie of attention modules preserving causality

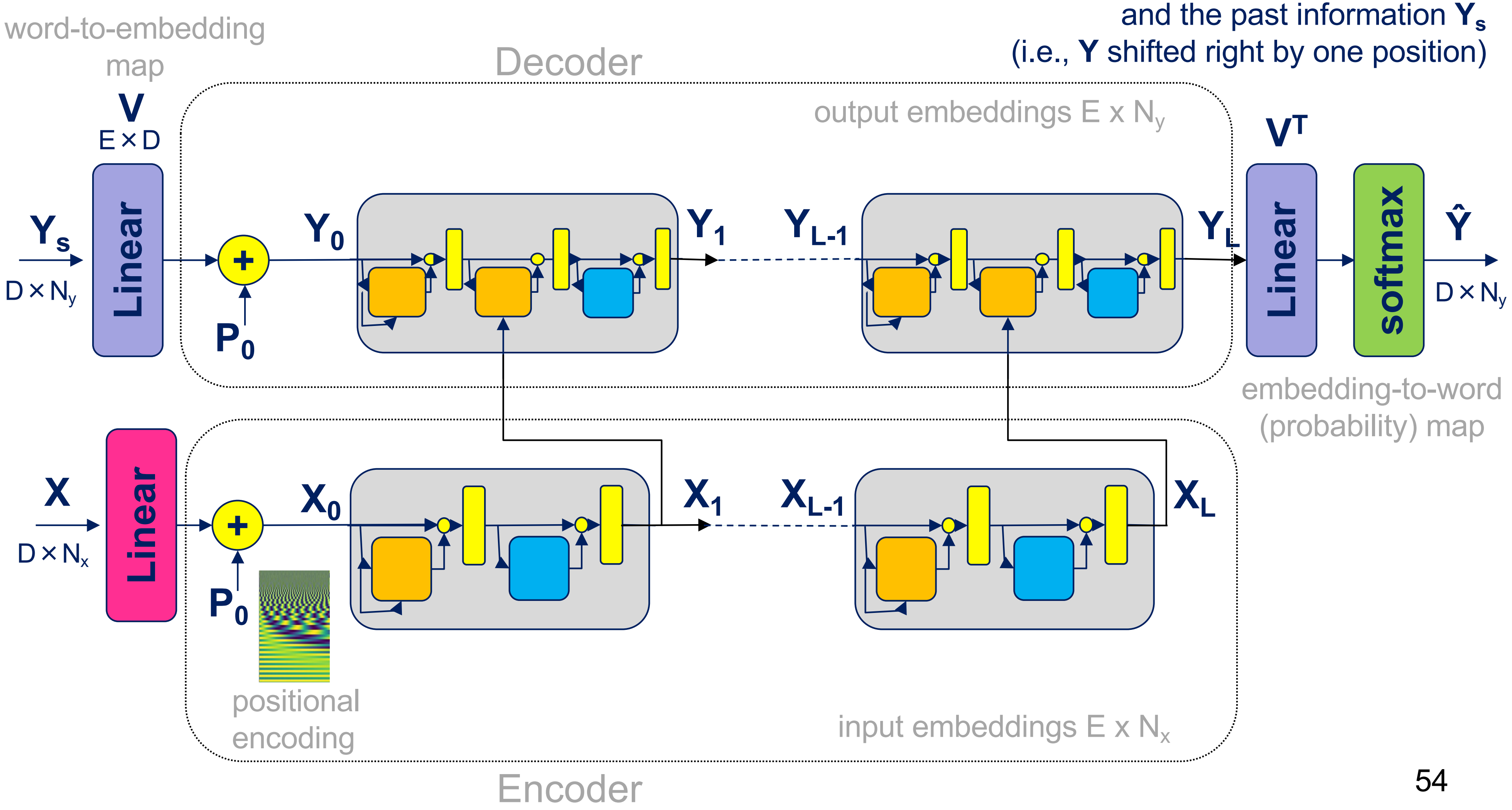


# Transformer Architecture

Vaswani, Ashish, et al. "Attention is all you need" (2017)

Google's patent https://patents.google.com/patent/US10452978B2/en

estimate **Ŷ**,
given the alternative representation **X**,
and the past information $\mathbf{Y}_s$
(i.e., **Y** shifted right by one position)



# The Annotated Transformer

http://nlp.seas.harvard.edu/2018/04/03/attention.html

tensor2tensor library https://github.com/tensorflow/tensor2tensor

harvardnlp

Members PI Code Publications

## The Annotated Transformer

Apr 3, 2018

---

**There is now a new version of this blog post updated for modern PyTorch.**

---

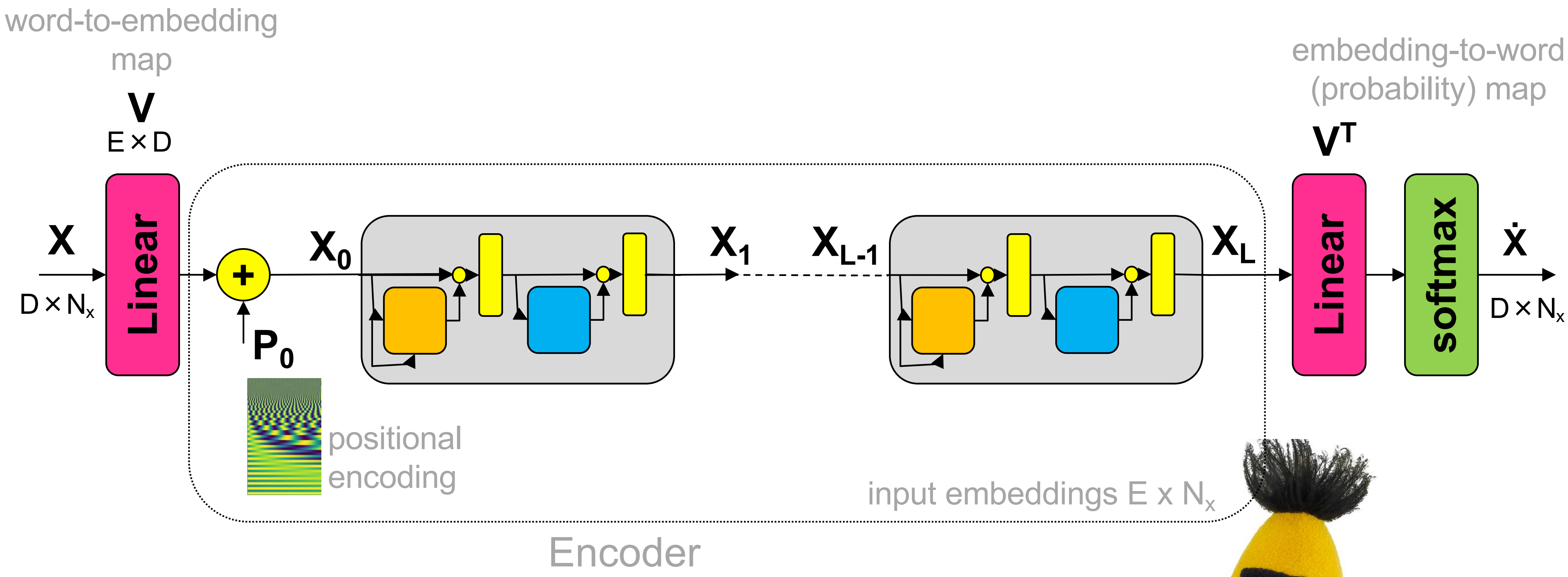
```
from IPython.display import Image
Image(filename='images/aiayn.png')
```

**Attention Is All You Need**

# BERT

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding" (2018)

https://github.com/google-research/bert




Created by researchers at Google AI Language

UNIVERSITÀ DEGLI STUDI DI PADOVA

# BERT parameters

| | Embeddings size E | Self-attention heads H | Head dimension K = E/H | FFN inner size I = 4E | Parameters per layer $12E^2+9E$ | Layers L | Dictionary size D | Total parameters |
|---|---|---|---|---|---|---|---|---|
| BERT base | 768 | 12 | 64 | 3072 | 7.1M | 12 | 30.5K | 110M |
| BERT large | 1024 | 16 | 64 | 4096 | 12.6M | 24 | 30.5K | 340M |

max tokens $N_x = 512$

Created by researchers at Google AI Language

# BERT pre-training procedure

BooksCorpus (800M words) + English Wikipedia (2,500M words)

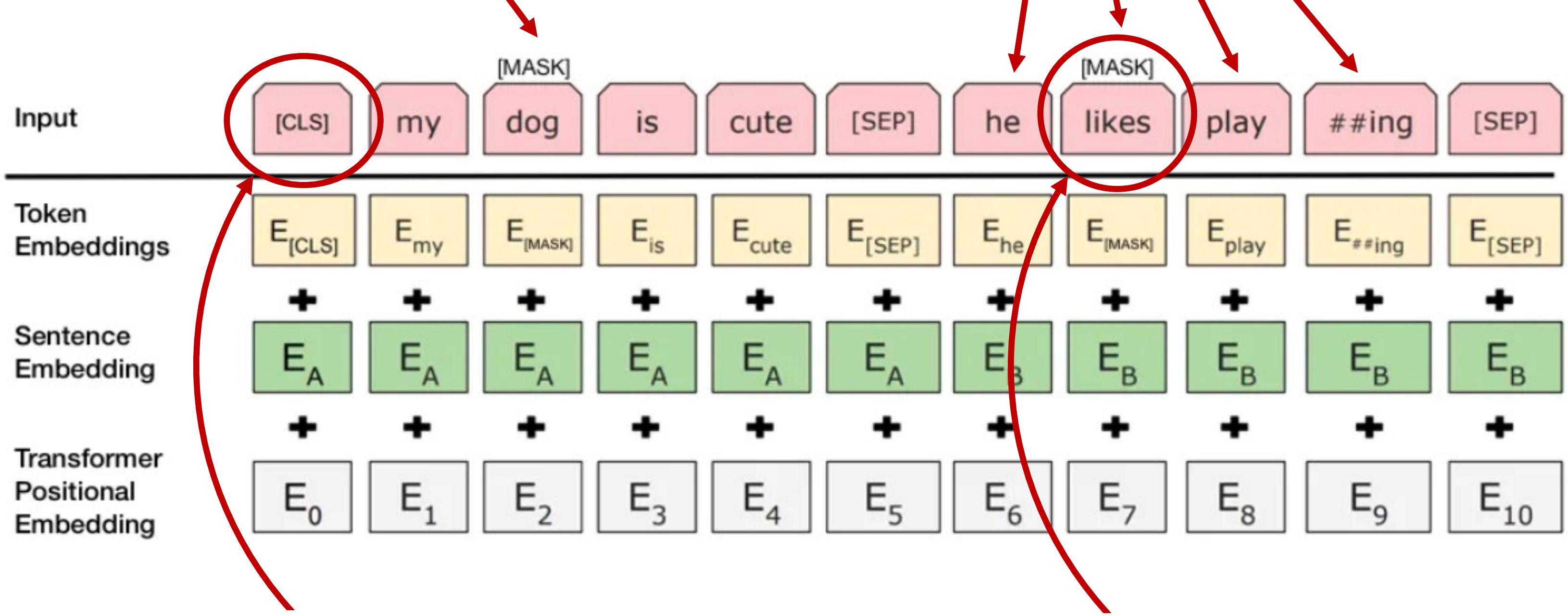
**Masked Language Model**

15% masked tokens replaced with:

- [MASK] token (80% of the times)
- Original token (10%)
- Random token (10%)

**Next Sequence Prediction**

- Next sequence (50% of the times)
- Random sequence (50%)




Output [CLS] fed into an additional output layer for softmax classification (of correct/wrong next sequence)

Output masked tokens fed into the output layer $V^T$ and evaluated for probability of correct estimate

# RoBERTa

Liu, Yinhan, et al. "Roberta: A robustly optimized BERT pretraining approach" (2019)

**Larger training corpora** (10x larger)

*training on BookCorpus + Wikipedia and also CC-News, OpenWebText, Stories*

**Dynamic masking**

*training data was duplicated 10 times so that each sequence is masked in 10 different ways over the 40 epochs of training*

**Full-sentences without NSP loss**

*full sentences sampled contiguously from one or more documents, such that the total length is at most 512 tokens*

**Large mini-batches**

**A larger byte-level BPE (byte pair encoding)** of 50K subword units

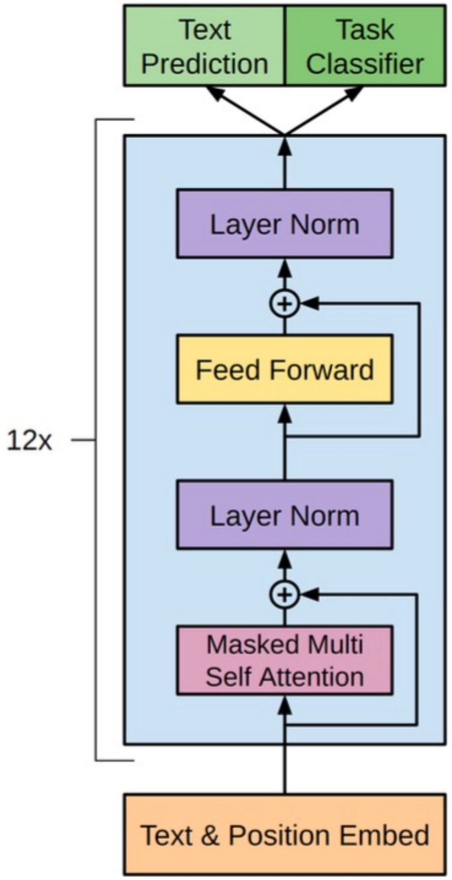
*a hybrid between character- and word-level representations that allows handling the large vocabularies common in natural language corpora*

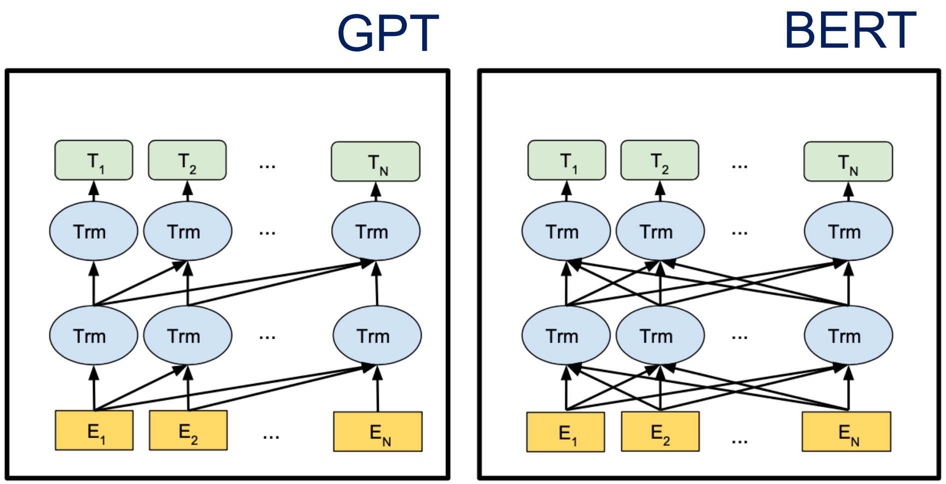
# Generative Pre-Training (GPT)

Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018)

(unsupervised) pre-training on Language Modelling (no mask)

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$




same parameters of BERT-base, but with Masked Attention trained on BookCorpus only

Created by researchers at OpenAI

# GPT-2

Radford, Alec, et al. "Language models are unsupervised multitask learners" (2019)

## McCann et al. (2018)

language provides a flexible way to specify tasks, inputs, and outputs all as a sequence of symbols… it is therfore possible to **train a single model** with **sufficient capacity** to infer and perform many **different tasks**

model gets complex!

data gets larger!

| Parameters | Layers | $d_{model}$ | |
|---|---|---|---|
| 117M | 12 | 768 | GPT, BERT-base |
| 345M | 24 | 1024 | BERT-large |
| 762M | 36 | 1280 | |
| 1542M | 48 | 1600 | GPT-2 |

## WebText

scraping all outbound links (45M links) from Reddit, a social media platform, which received at least 3 karma – exclude WikiPedia

Created by researchers at OpenAI

# GPT-3

Brown, Tom, et al. "Language models are few-shot learners" (2020)

increasingly larger data and model!

| Model Name | $n_{\text{params}}$ | $n_{\text{layers}}$ | $d_{\text{model}}$ | $n_{\text{heads}}$ | $d_{\text{head}}$ | Batch Size | Learning Rate |
|---|---|---|---|---|---|---|---|
| GPT-3 Small | 125M | 12 | 768 | 12 | 64 | 0.5M | $6.0 \times 10^{-4}$ |
| GPT-3 Medium | 350M | 24 | 1024 | 16 | 64 | 0.5M | $3.0 \times 10^{-4}$ |
| GPT-3 Large | 760M | 24 | 1536 | 16 | 96 | 0.5M | $2.5 \times 10^{-4}$ |
| GPT-3 XL | 1.3B | 24 | 2048 | 24 | 128 | 1M | $2.0 \times 10^{-4}$ |
| GPT-3 2.7B | 2.7B | 32 | 2560 | 32 | 80 | 1M | $1.6 \times 10^{-4}$ |
| GPT-3 6.7B | 6.7B | 32 | 4096 | 32 | 128 | 2M | $1.2 \times 10^{-4}$ |
| GPT-3 13B | 13.0B | 40 | 5140 | 40 | 128 | 2M | $1.0 \times 10^{-4}$ |
| GPT-3 175B or "GPT-3" | 175.0B | 96 | 12288 | 96 | 128 | 3.2M | $0.6 \times 10^{-4}$ |

**Layer normalization at the input** (plus one at the output)

**Sparse attention patterns**
*alternating dense and locally banded sparse attention patterns in the layers*

**Byte-level BPE (byte pair encoding)** of 50K subword units
*also prevent BPE from merging across character categories (to avoid dog, dog!, dog?)*

**Modified initialization**

Created by researchers at OpenAI

# Supervised fine-tuning

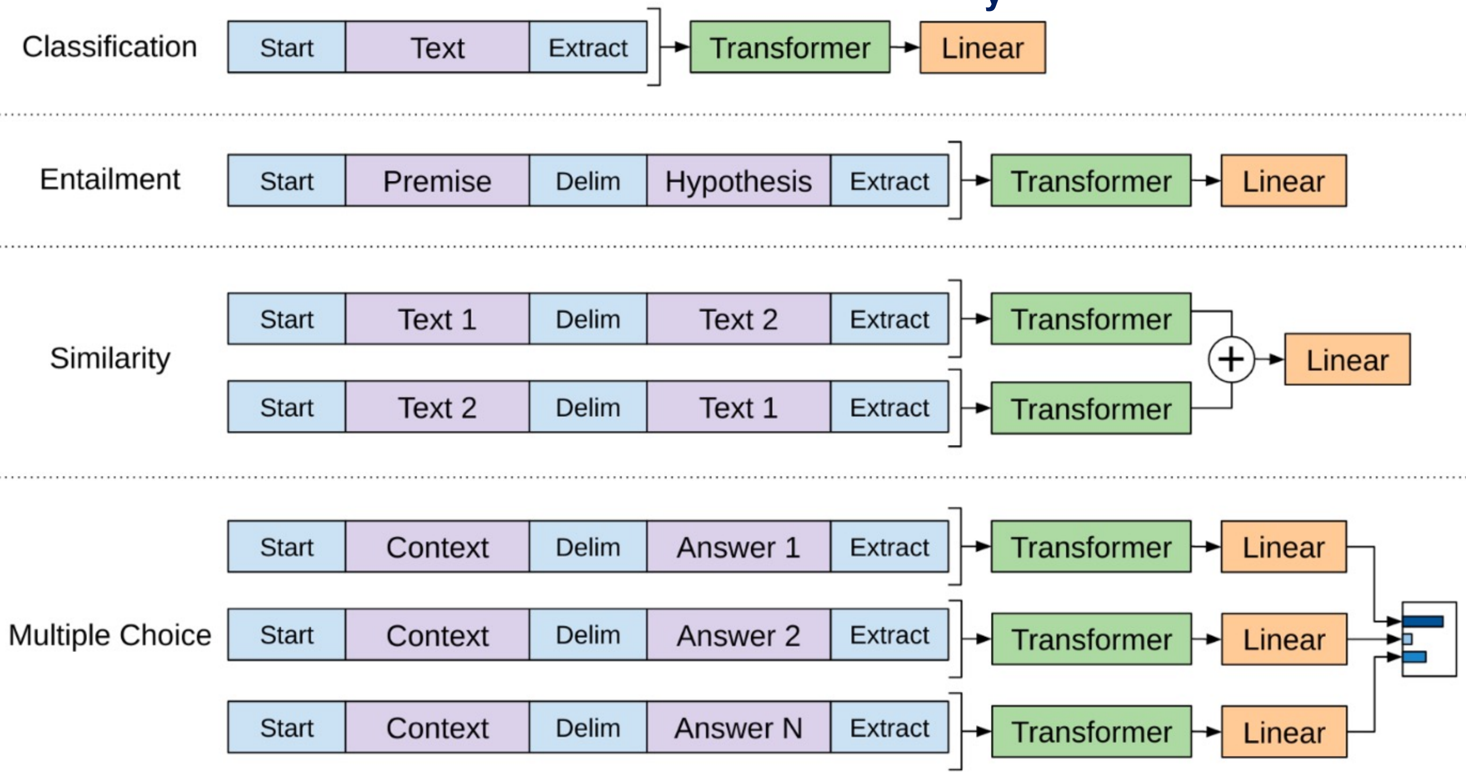
## training on specific tasks

$$L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

log softmax($\mathbf{V}_y^T \mathbf{X}_L$)

Language Modelling loss

$\mathbf{V}_y^T$



# NLP tasks

## some fine-tuning possibilities in NLP

| Task | Description | Possible approach |
|---|---|---|
| Masked language prediction | predict masked words in a text | This is what BERT model is pre-trained for |
| Text classification or Sentiment analysis | assign a label to a given sequence of text | Apply linear transform+softmax on K classes, and train the model for the specific classification task |
| Text translation | translate a text | Need to pre-train a full Transfomer Architecture for this task |
| Summarization | generate a summary of a document | GPT example: context given by a document; then generate 100 tokens by top-2 random sampling (Fan et al., 2018), i.e., take at each step the most likely next word at random among the top-2 candidates; finally select first 3 sentences as abstract |
| Question answering | answer a question | GPT example: the context of the language model is seeded with example question answer pairs which helps the model infer the short answer style of the dataset |
| Document question answering | answer a question on a given text | GPT example: context seeded by a text; then as for question answering |
| Conversational | ChatBot | InstructGPT/ChatGPT: Fine-tuned models using reinforcement learning from human feedback |

# Hugging Face

https://huggingface.co/docs/transformers/v4.29.1/en/index

State-of-the-art Machine Learning
for PyTorch, TensorFlow, and JAX

PyTorch TensorFlow

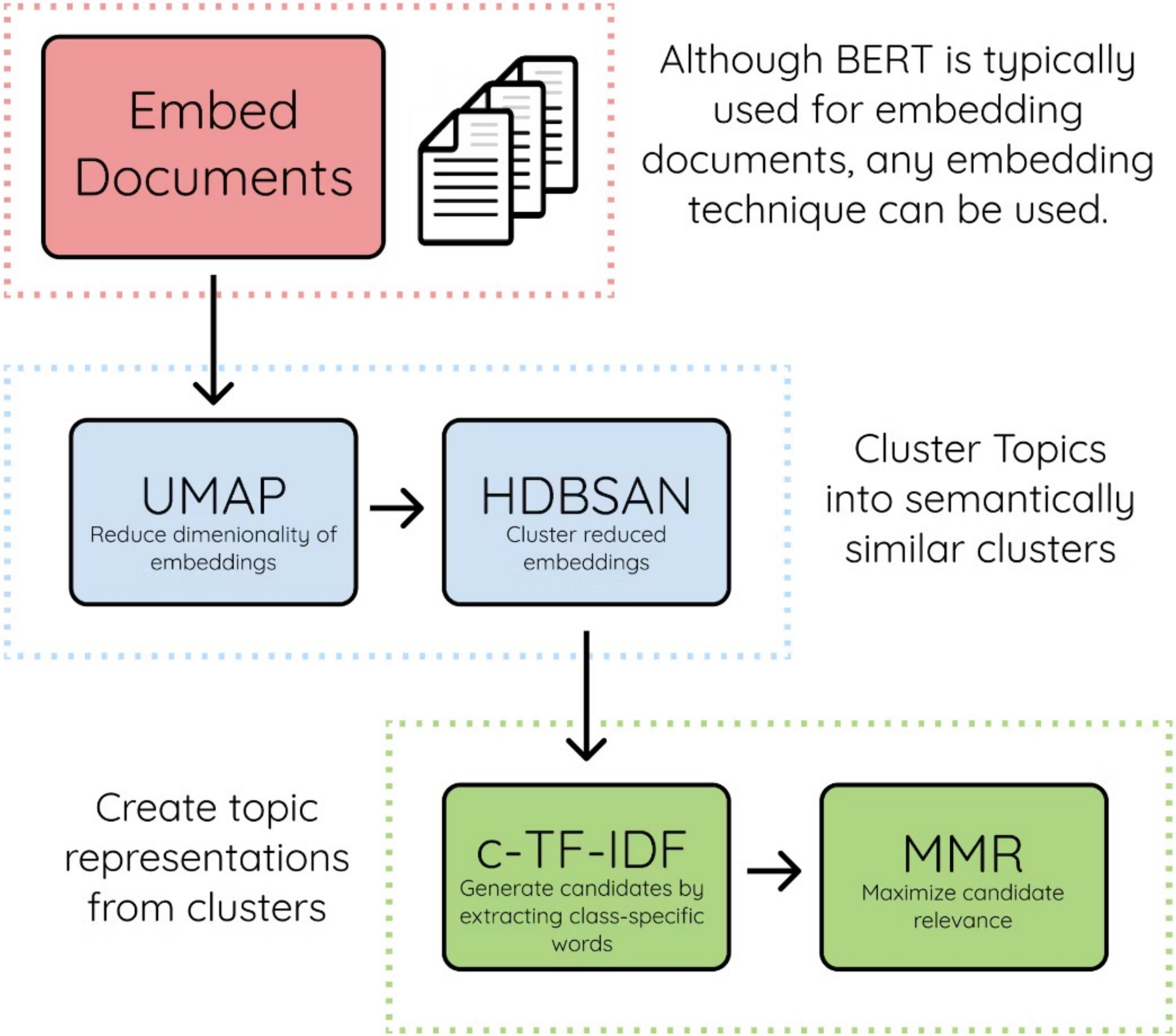
ALBERT, BART, **BERT**, BigBird, BigBird-Pegasus, BioGpt, BLOOM, CamemBERT, CANINE, ConvBERT, CTRL, Data2VecText, DeBERTa, DeBERTa-v2, DistilBERT, ELECTRA, ERNIE, ErnieM, ESM, FlauBERT, FNet, Funnel Transformer, GPT-Sw3, **OpenAI GPT-2**, GPTBigCode, GPT Neo, GPT NeoX, GPT-J, I-BERT, LayoutLM, LayoutLMv2, LayoutLMv3, LED, LiLT, LLaMA, Longformer, LUKE, MarkupLM, mBART, MEGA, Megatron-BERT, MobileBERT, MPNet, MVP, Nezha, Nyströmformer, OpenLlama, **OpenAI GPT**, OPT, Perceiver, PLBart, QDQBert, Reformer, RemBERT, **RoBERTa**, RoBERTa-PreLayerNorm, RoCBert, RoFormer, SqueezeBERT, TAPAS, Transformer-XL, XLM, XLM-RoBERTa, XLM-RoBERTa-XL, XLNet, X-MOD, YOSO

# BERT Topic

exploiting embeddings for topic detection

# BERTopic

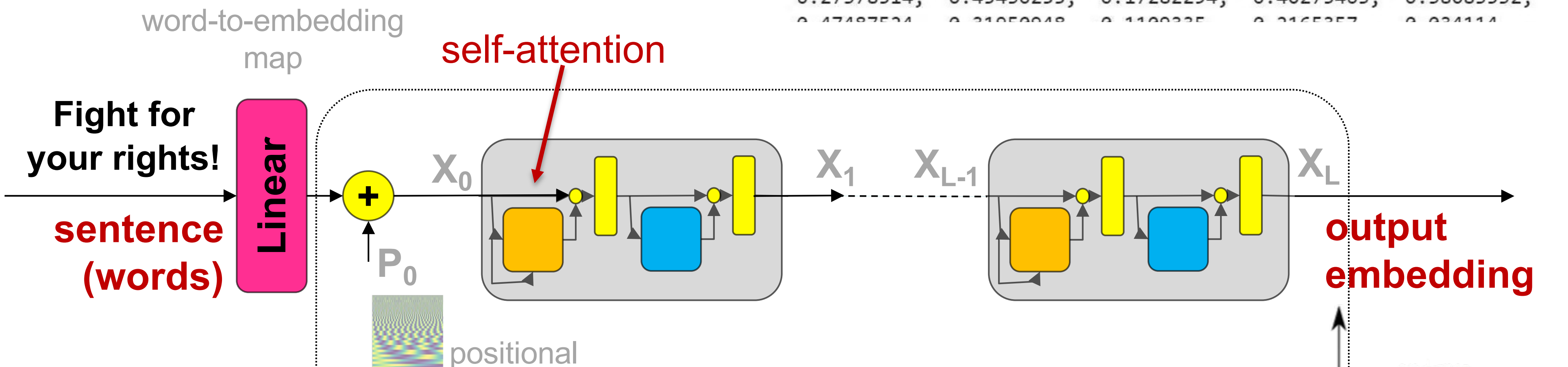
Grootendorst, «BERTopic: Neural topic modeling with a class-based TF-IDF procedure» (2022) https://arxiv.org/abs/2203.05794



# 1. Embed documents

## Using BERT

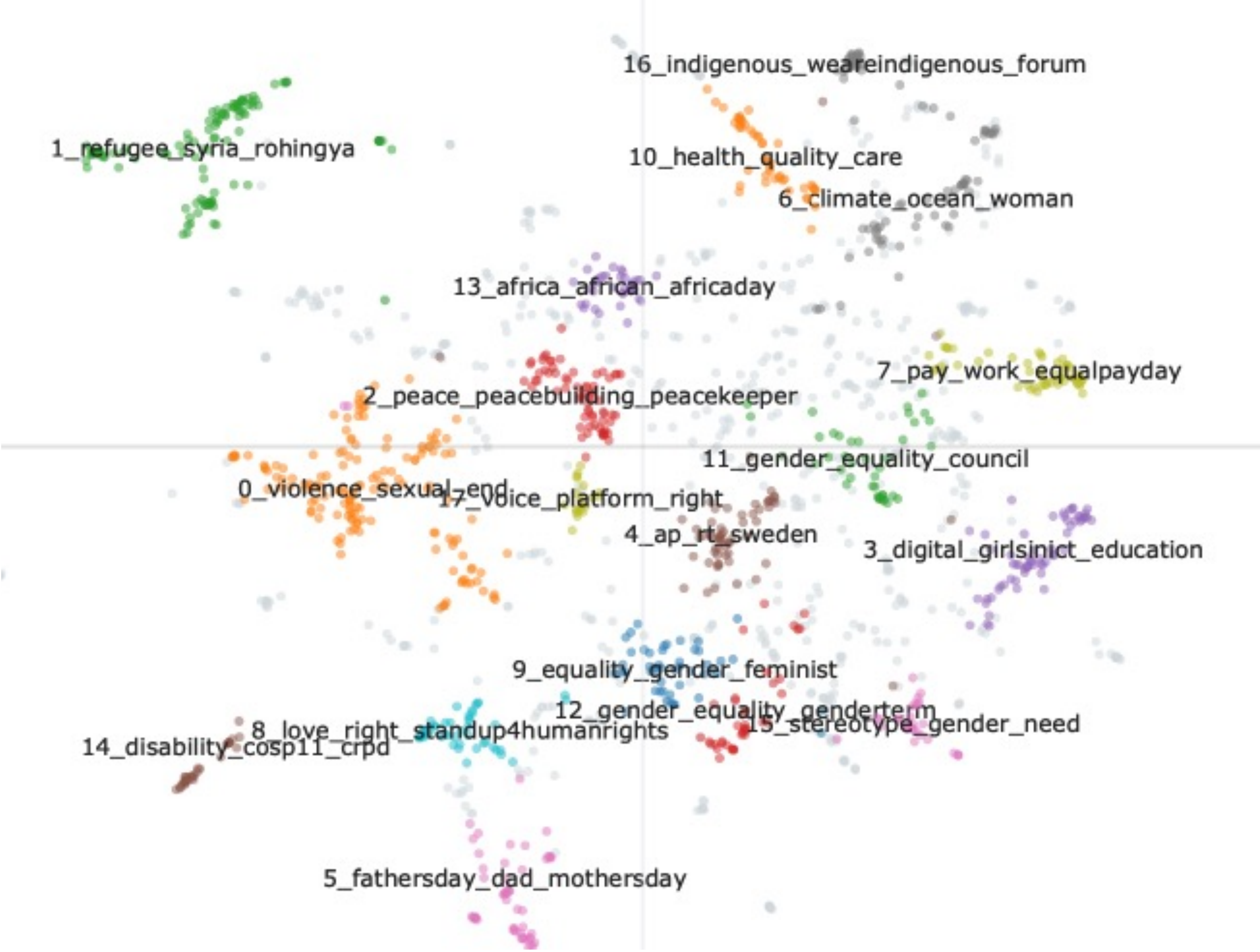
```
array([-0.5968882 , -0.33086956, -0.32643065, -0.3670732 ,  0.628059  ,
       -0.3692328 , -0.37902787, -0.12308089, -0.38124698, -0.03940517,
        0.2260839 ,  0.10852845, -0.2873811 , -0.42781743,  0.06604357,
       -0.07114276, -0.29775023, -0.99628943, -0.54497653, -0.11718027,
       -0.15935768,  0.09587188, -0.2503798 ,  0.06768776,  0.3311586 ,
        0.43098116,  0.06936899,  0.24311952,  0.14515282,  0.19245838,
        0.10462623, -0.45676082,  0.5662387 ,  0.69908774,  0.48064467,
        0.27378514, -0.45430255,  0.17282294, -0.40275463, -0.38083532,
```




# 2. Reinterpret embeddings

Using UMAP

bert_model.visualize_documents(docs)



# BERTopic in Python

bertopic package https://maartengr.github.io/BERTopic/

```
!pip install bertopic
from bertopic import BERTopic
from sentence_transformers import SentenceTransformer
```

initialise model

```
sentence_model = SentenceTransformer("all-MiniLM-L6-v2")
bert_model = BERTopic(embedding_model=sentence_model,
                      min_topic_size=20,nr_topics='auto')
```

fit model

```
docs = list(df2["text_sup_clean"])
topics, probabilities = bert_model.fit_transform(docs)
```

```
topics = bert_model.reduce_outliers(docs, topics)
```

reduce outliers

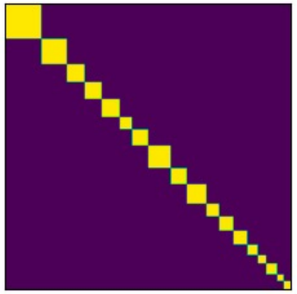
```
# extract community assignments
C = sps.csr_matrix((len(topics),max(topics)+2))
for i in range(C.shape[1]):
  C[np.array(topics)==(i-1),i] = 1

# remove zero assignments
C = C[:,np.unique(scipy.sparse.find(C)[1])]
```

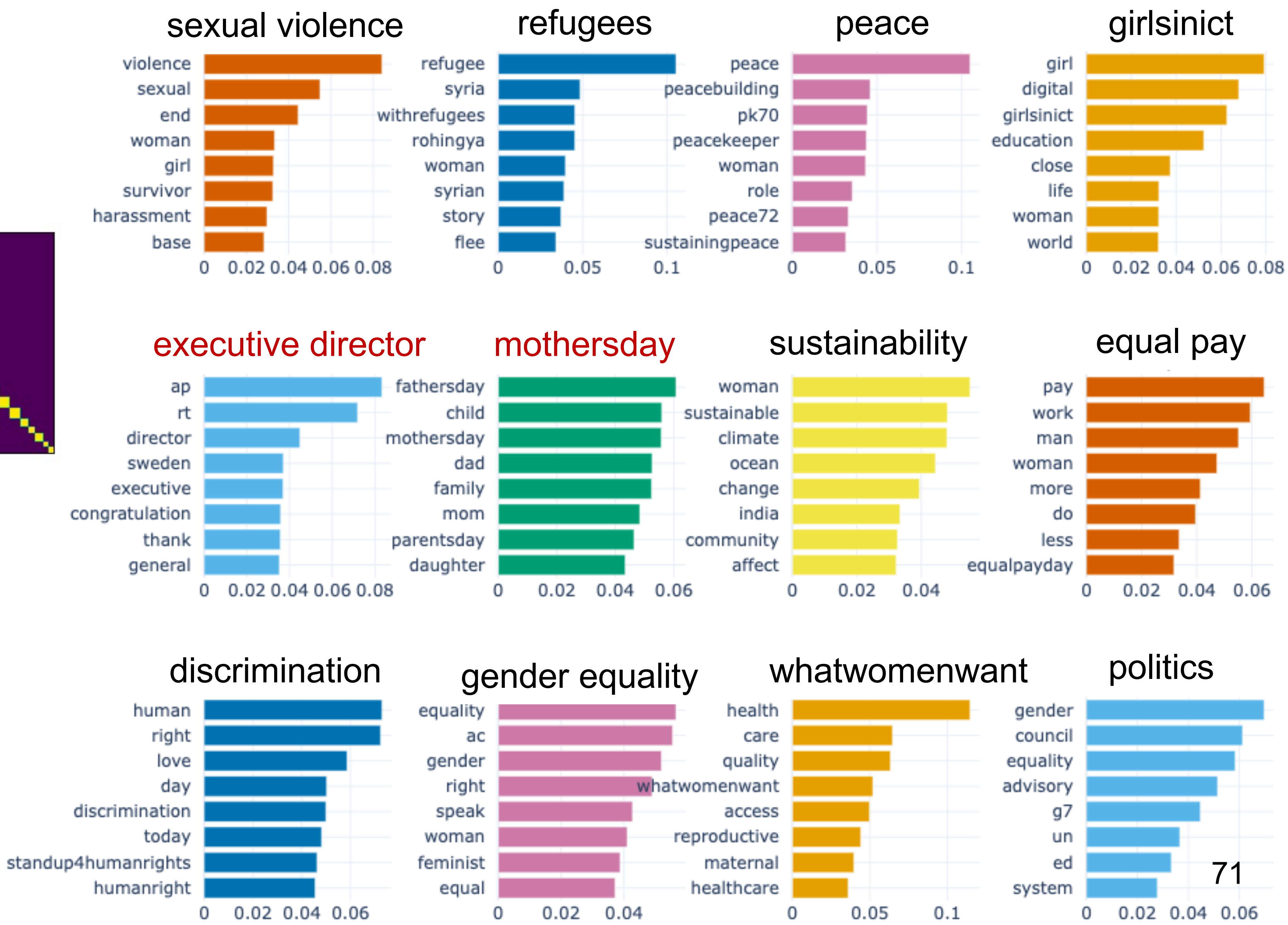
extract C from topic assignment

UNIVERSITÀ DEGLI STUDI DI PADOVA

# bert_model.visualize_barchart()

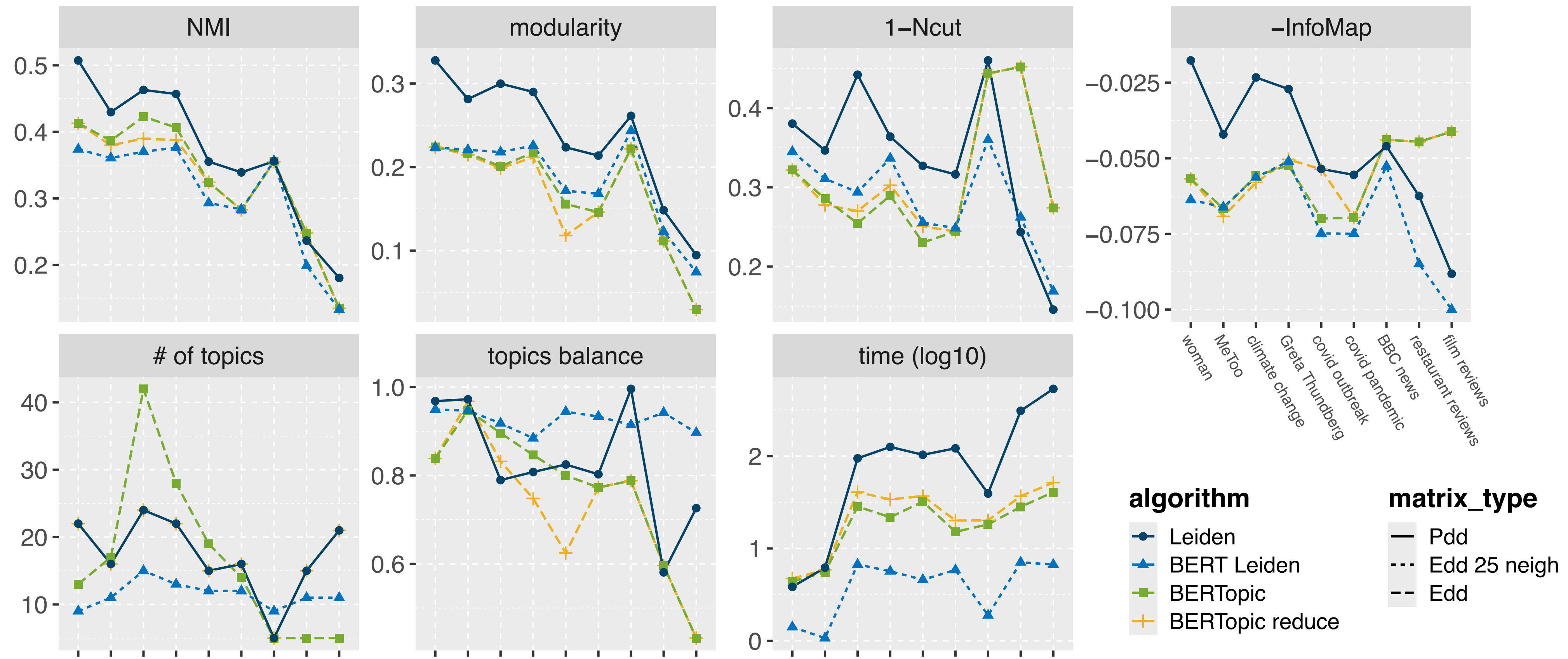
## #metoo2018





# BERTopic at work

## On a semantic network



# Takeaways

## on BERTopic applied to topic detection

- Naturally provides a hard topic assignment
- Useful tool
- More readable output with deep cleaned text
  but same performance
- Comparison – with Louvain
  weaker in general, especially in modularity
  equivalent NMI = relevant topics
  lower modularity = the documents that identify the topics are less distinguishable
  higher complexity involved
  less balanced topics, but generally meaningful
  topics correlated with Louvain

# Wrap-up

on topic detection

# Takeaways
## for your projects

- What available tools should be used
  - Louvain & BERTopic
  - compare their performance through NMI, modularity, etc.

- What available tools should NOT be used
  - InfoMap, NMF & LDA
  - they show poor performance

- What would be nice to see implemented
  - soft Louvain made fast
  - advanced SMBs
  - deep learning models