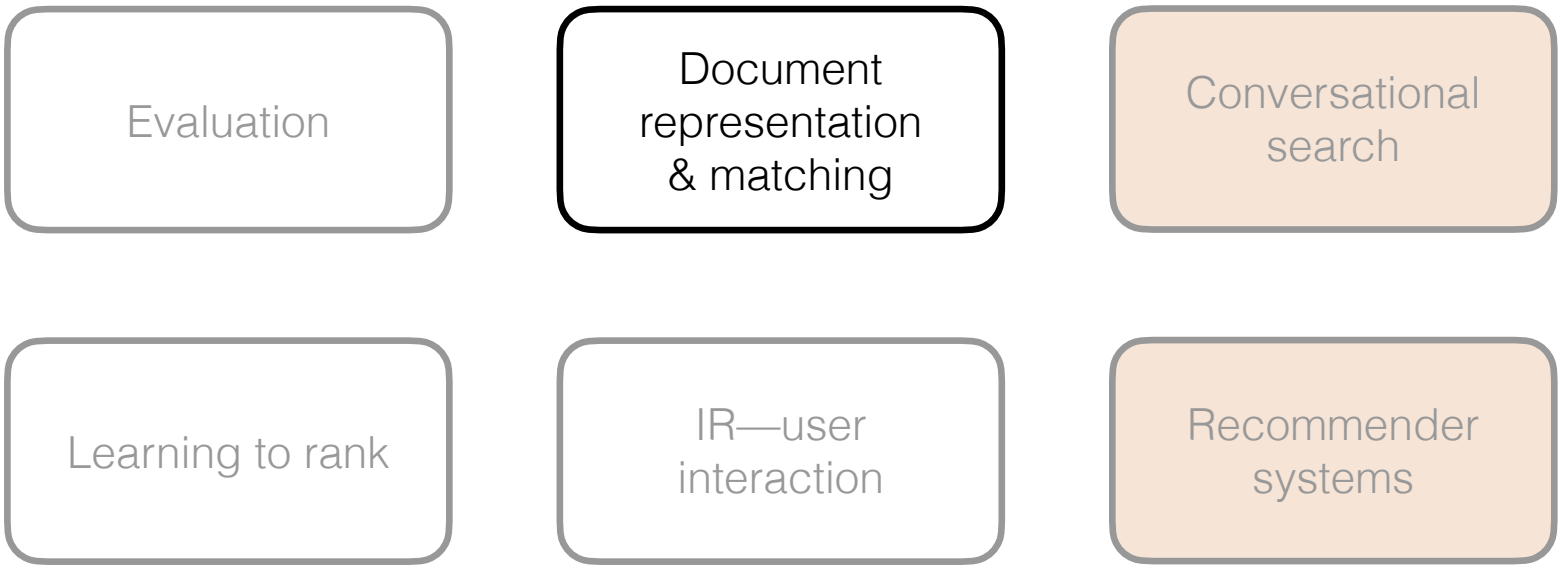
Information Retrieval 1

Semantic-based Retrieval

Ilya Markov
i.markov@uva.nl

University of Amsterdam

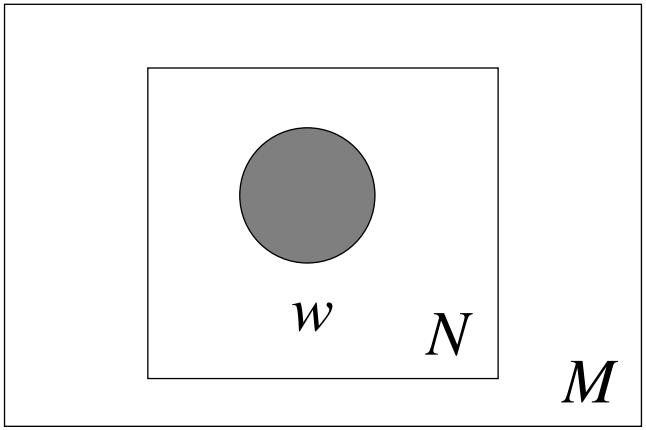
Document representation and matching



Outline

Outline

Unigram language model



$$w_{ij} \sim Mult(d_i)$$
$$i \in \{1, \ldots, M\}$$
$$j \in \{1, \ldots, N_i\}$$

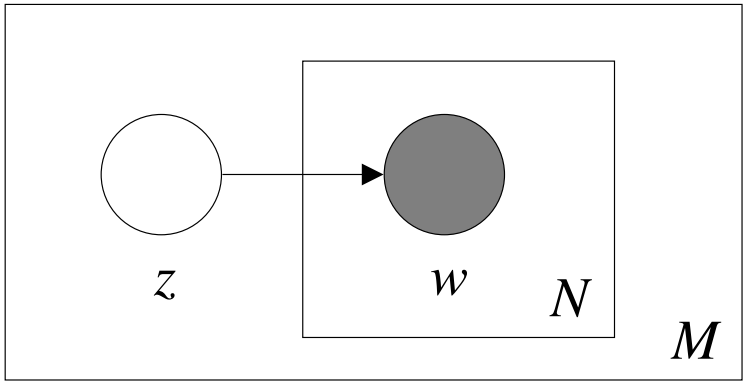
Blei et al., "Latent Dirichlet Allocation"

Mixture of unigrams

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

Blei et al., "Latent Dirichlet Allocation"

Mixture of unigrams



$$z_i \sim Mult(\theta)$$
$$w_{ij} \sim Mult(\phi_{z_i})$$

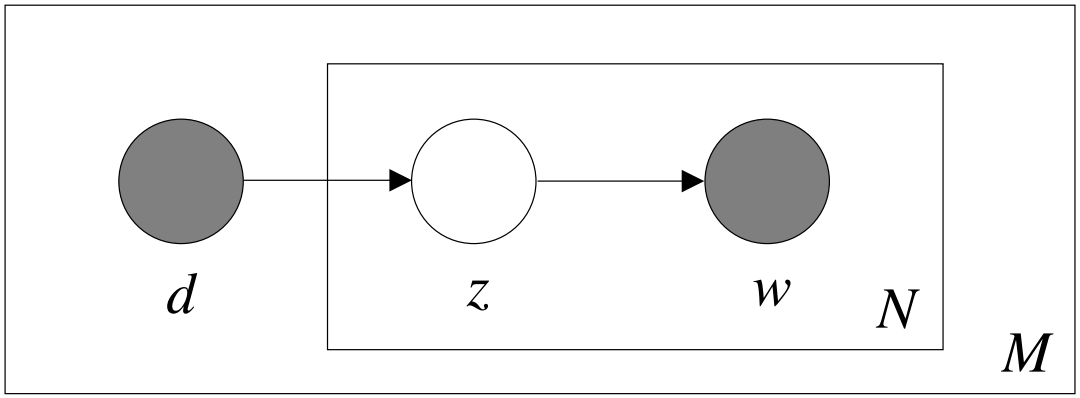
Blei et al., "Latent Dirichlet Allocation"

Probabilistic latent semantic analysis (pLSA)

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Blei et al., "Latent Dirichlet Allocation"

pLSA

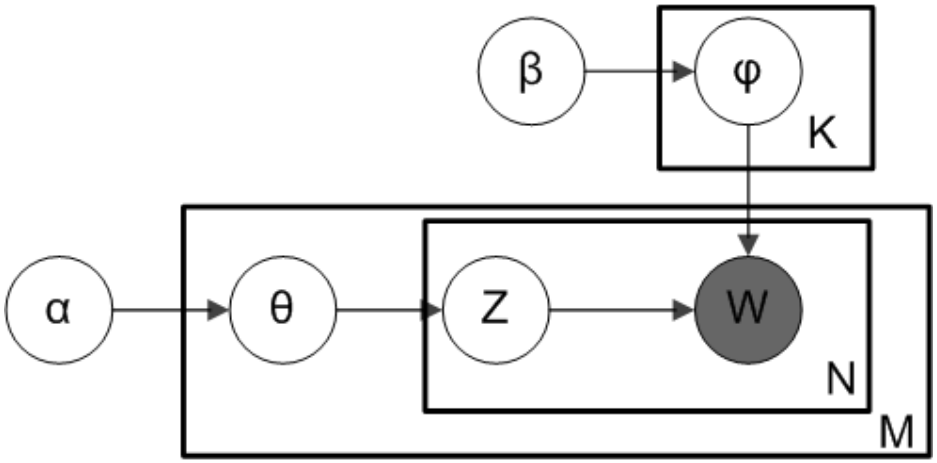


$$z_{ij} \sim Mult(\theta_i)$$
$$w_{ij} \sim Mult(\phi_{z_{ij}})$$

$$P(w \mid d) = \sum_z P(w \mid \phi_z) P(z \mid \theta_d)$$

Blei et al., "Latent Dirichlet Allocation"

Latent Dirichlet allocation (LDA)



1. Choose $\theta_i \sim Dir(\alpha)$, where $i \in \{1, \ldots, M\}$
2. Choose $\phi_k \sim Dir(\beta)$, where $k \in \{1, \ldots, K\}$
3. For each position j, where $j \in \{1, \ldots, N_i\}$
   a. Choose a topic $z_{ij} \sim Mult(\theta_i)$
   b. Choose a word $w_{ij} \sim Mult(\phi_{z_{ij}})$

https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

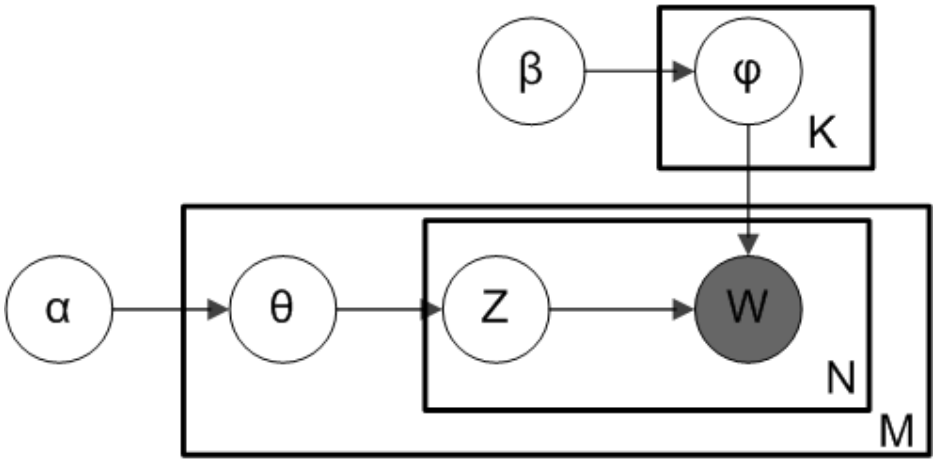
Documents as distributions

- Documents and queries are distributions over words

$$P(w \mid d) = \sum_z P(w \mid \phi_z) P(z \mid \theta_d)$$

- $P(w \mid \phi_z)$ and $P(z \mid \theta_d)$ are estimated through topic modeling (discussed next)
- Match documents and queries using QLM or KL-divergence

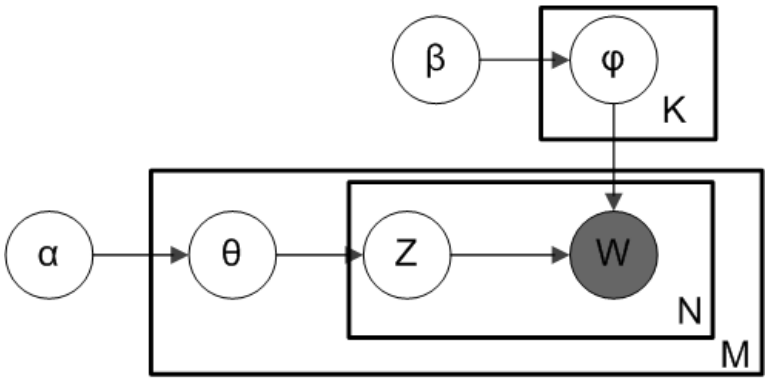
Estimating LDA



$$z_{ij} \sim Mult(\theta_i)$$
$$w_{ij} \sim Mult(\phi_{z_{ij}})$$

We need to find θ_d for every document and ϕ_z for every topic

Estimating LDA: notation



W	Words in documents (observed)
Z	Topics for words (not observed)
$\boldsymbol{\theta}$	Distributions of topics in documents (parameters)
$\boldsymbol{\phi}$	Distributions of words in topics (parameters)
$L(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathrm{W}, \mathrm{Z}) =$ $p(\mathrm{X}, \mathrm{Z} \mid \boldsymbol{\theta}, \boldsymbol{\phi})$	Log-likelihood of observed data W and unobserved random variables Z, given parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$

Estimating LDA: expectation-maximization

- E-step: define the expected value of the log-likelihood function, with respect to the current estimates of the parameters $\boldsymbol{\theta}^{(t)}, \boldsymbol{\phi}^{(t)}$:

$$Q(\boldsymbol{\theta}, \boldsymbol{\phi} \mid \boldsymbol{\theta}^{(t)}, \boldsymbol{\phi}^{(t)}) = \mathrm{E}_{\mathrm{Z}|\mathrm{W}, \boldsymbol{\theta}^{(t)}, \boldsymbol{\phi}^{(t)}} [\log L(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathrm{W}, \mathrm{Z})]$$

- M-step: find the parameters that maximize this quantity

$$\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\phi}^{(t+1)} = \arg\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} Q(\boldsymbol{\theta}, \boldsymbol{\phi} \mid \boldsymbol{\theta}^{(t)}, \boldsymbol{\phi}^{(t)})$$

- Repeat until convergence

https://en.wikipedia.org/wiki/Expectation-maximization_algorithm

Outline

Vector space model

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Anthony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Manning et al., "Introduction to Information Retrieval"

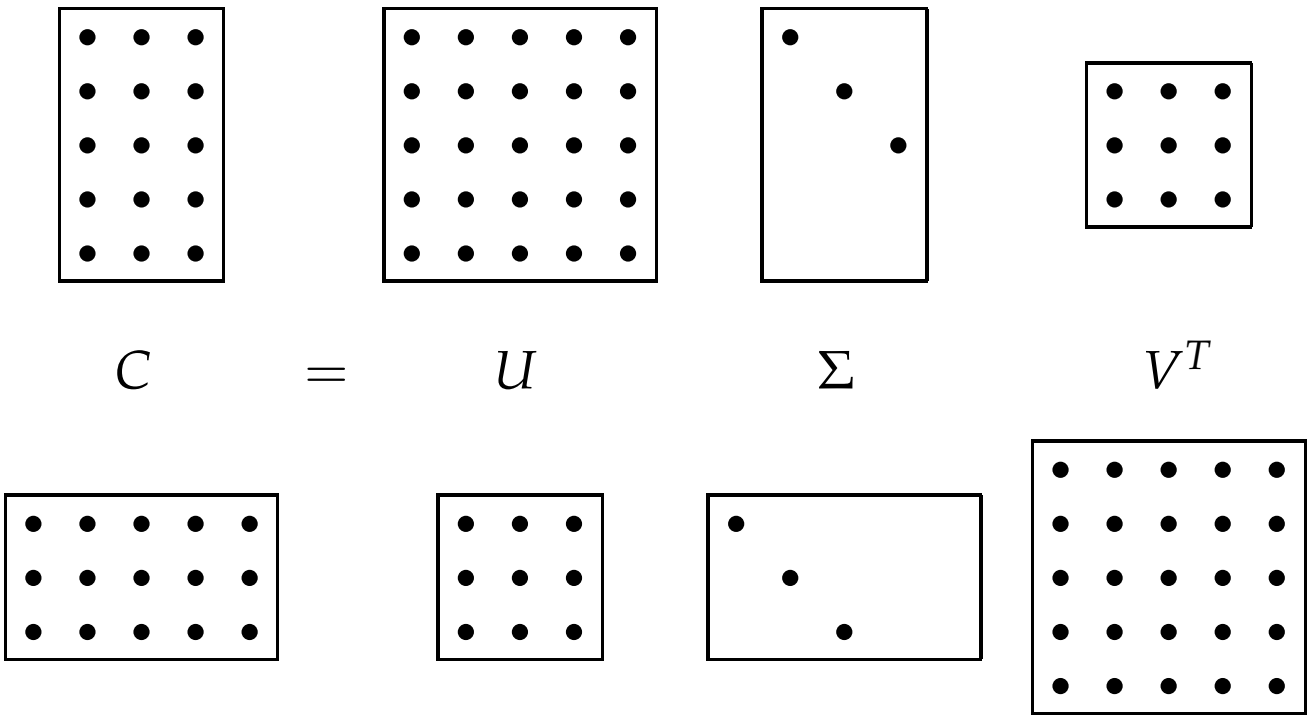
Singular value decomposition

- C is a $m \times n$ matrix (term-document)
- C can be decomposed as

$$C = U\Sigma V^T$$

- U is a $m \times m$ unitary matrix
- Σ is a diagonal $m \times n$ matrix with singular values
- V^T is a $n \times n$ unitary matrix

Singular value decomposition



Manning et al., "Introduction to Information Retrieval"

SVD example: original matrix

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

Manning et al., "Introduction to Information Retrieval"

SVD example: decomposition

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
voyage	−0.70	0.35	0.15	−0.58	0.16
trip	−0.26	0.65	−0.41	0.58	−0.09

$\times$

2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	1.28	0.00	0.00
0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.39

$\times$

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Manning et al., "Introduction to Information Retrieval"

Low-rank approximation

$$C = U\Sigma V^T = \sum_{i=1}^{\min(m,n)} \sigma_i \vec{u}_i \vec{v}_i^T$$

$$\approx \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^T = U_k \Sigma_k V_k^T$$

LSI/LSA example: low-rank approximation

$$\begin{matrix} 2.16 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.59 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{matrix} \times$$

	d_1	d_2	d_3	d_4	d_5	d_6
1	-1.62	-0.60	-0.44	-0.97	-0.70	-0.26
2	-0.46	-0.84	-0.30	1.00	0.35	0.65

Manning et al., "Introduction to Information Retrieval"

Latent semantic indexing/analysis

$$
\begin{array}{ccccccc}
 & C & & U_k & & \Sigma_k & V_k^T \\
 & (\mathbf{d}_j) & & & & & (\hat{\mathbf{d}}_j) \\
 & \downarrow & & & & & \downarrow \\
(\mathbf{t}_i^T) \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} & = (\hat{\mathbf{t}}_i^T) \rightarrow & \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ \mathbf{u}_k \\ \\ \end{bmatrix} \end{bmatrix} & \cdot \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_k \end{bmatrix} & \cdot \begin{bmatrix} \begin{bmatrix} & \mathbf{v}_1 & \end{bmatrix} \\ \vdots \\ \begin{bmatrix} & \mathbf{v}_k & \end{bmatrix} \end{bmatrix}
\end{array}
$$

$$
d_j = U_k \Sigma_k \hat{d}_j \Longrightarrow \hat{d}_j = \Sigma_k^{-1} U_k^T d_j
$$

https://en.wikipedia.org/wiki/Latent_semantic_analysis

Documents as vectors

- Given a collection of documents, perform SVD and low-rank approximation to obtain Σ_k and U_k
- Given a document and a query, represent them as a vectors in the obtained "semantic" vector space

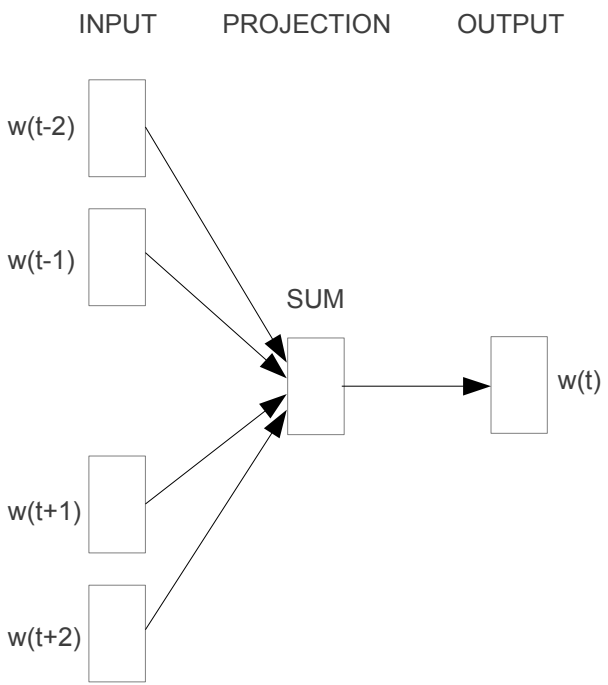
$$\hat{d} = \Sigma_k^{-1} U_k^T d$$
$$\hat{q} = \Sigma_k^{-1} U_k^T q$$

- Match the obtained "semantic" vector representations $\hat{d}$ and $\hat{q}$ using cosine similarity

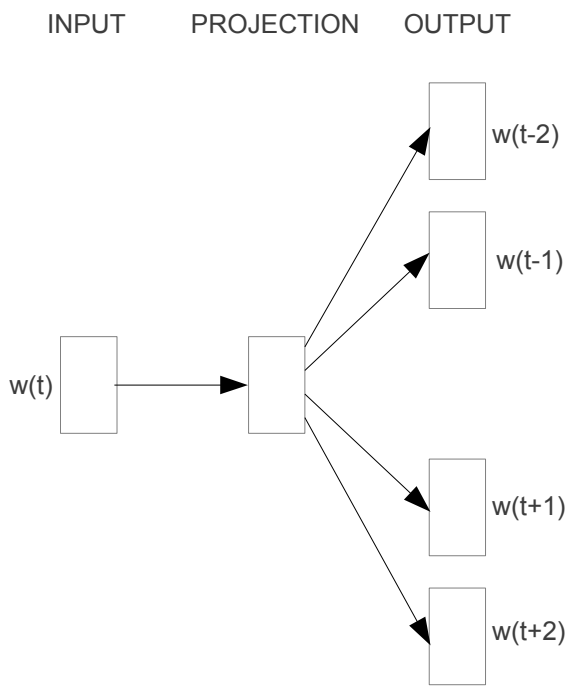
Outline

Outline

Word2vec



CBOW



Skip-gram

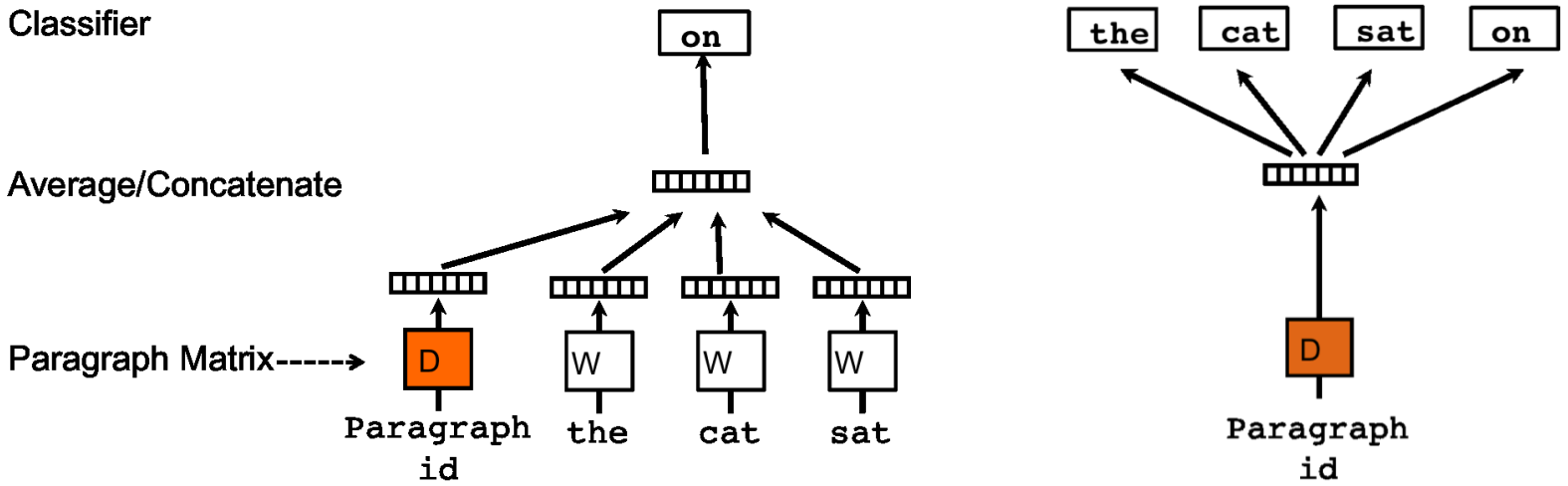
Mikolov et al., "Efficient Estimation of Word Representations in Vector Space"

Documents as vectors

- Compute word embeddings
- Given a document and a query, compute their vector representations as average word embeddings (AWEs)
- Match using cosine similarity

Outline

Paragraph vector



1. At every step of stochastic gradient descent, sample a fixed-length context from a random paragraph
2. Compute the error gradient from the network
3. Use the gradient to update parameters

Le and Mikolov, "Distributed Representations of Sentences and Documents"

Document as vectors

- Comput
- Given a
  1. Fix
  2. Ad
     D (
  3. Up
  4. Get
     matrix D
- Match using cosine similarity

We have learned the document representations (in the document matrix D where the columns represent the different documents)
We add a randomly initialized column to this matrix, representing the query
We perform SGD with fixed word matrix again learn D and especially column q (I assume we add this column to D to learn the representation of q in a similar way as the document representations are learned?)
We separate D and q again and match all documents (columns of D) with q using cosine similarity

Semantic retrieval summary

- Documents as distributions
  - Topic modeling (pLSA, LDA)
- Documents as vectors
  - Latent semantic indexing/analysis
  - Words embeddings (word2vec and AWE)
  - Document embeddings

Materials

- Manning et al., Chapter 18
- Blei et al.
  Latent Dirichlet Allocation
  Journal of Machine Learning Research, 2003

Materials

- Mikolov et al.
  Distributed Representations of Words and Phrases and their Compositionality
  Advances in neural information processing systems, 2013
- Le and Mikolov
  Distributed Representations of Sentences and Documents
  Proceedings of JMLR, 2014