Tensor methods in NLP

Igor Vorona
Anh-Huy Phan

Skoltech

March 2022

Outline

- Intro
- Topic modeling
- Representation learning
- Rotation document model

Intro

Bag-of-words

It’s a way to represent text as a multiset of words, i.e. we ignore any order of words inside textual data

```
(1) John likes to watch movies. Mary likes movies too.
```

```
(2) Mary also likes to watch football games.
```

```
"John","likes","to","watch","movies","Mary","likes","movies","too"

"Mary","also","likes","to","watch","football","games"
```

Compositionality property

- **Principle of Compositionality**

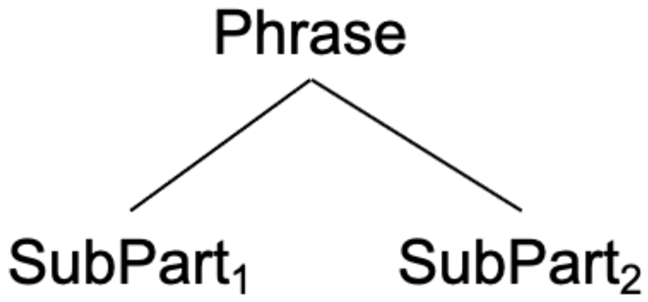
–Meaning(Phrase) =

composition of

Meaning($SubPart_1$),

Meaning($SubPart_2$)

and so on…



–means meaning of sentence is derived systematically from the meaning of the words contained in that sentence

Data in NLP

- Data in NLP have usually a hierarchical structure:

    - Language organized as sequence of
        i. Documents
        ii. Paragraphs
        iii. Sentences
        iv. Words
        v. Subword units

- **Applicability of tensor models:**
  Data should have **a clear tensor structure with a natural meaning of each dimension**. Tensor models try to reconstruct functions for each element of data. For example, CPD encode dot product similarity between vectors corresponding to each element of our data.

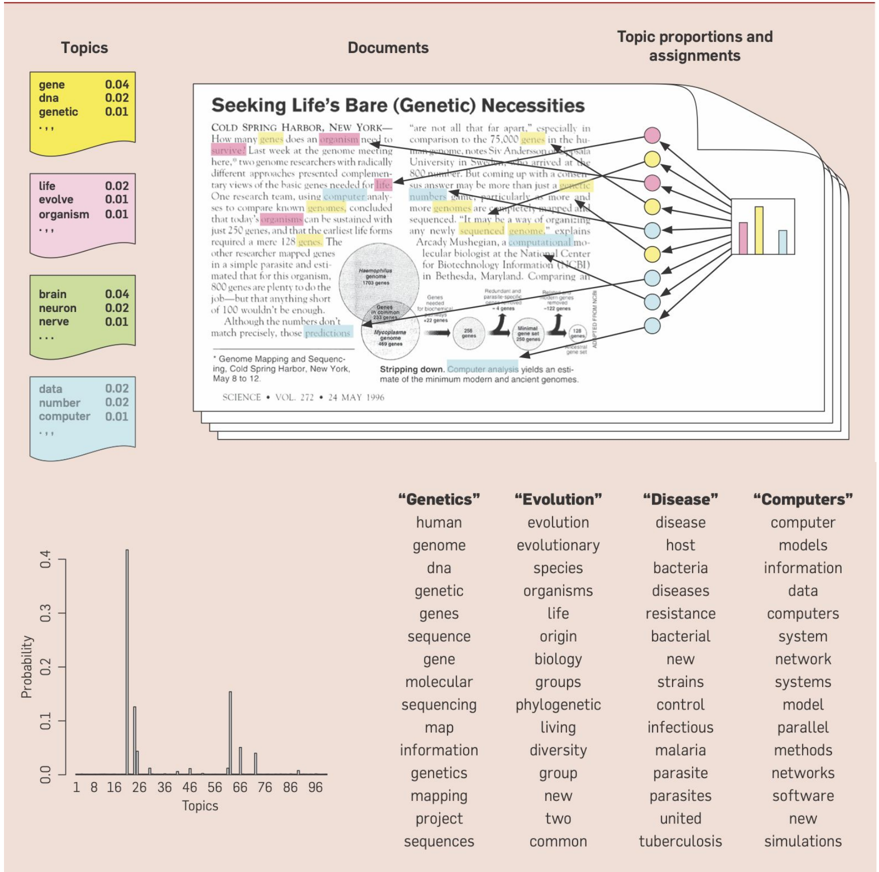
  Due to this structure, any textual data can be represented as a similarity tensor of different language elements based on their co-occurrence information.

Topic modeling

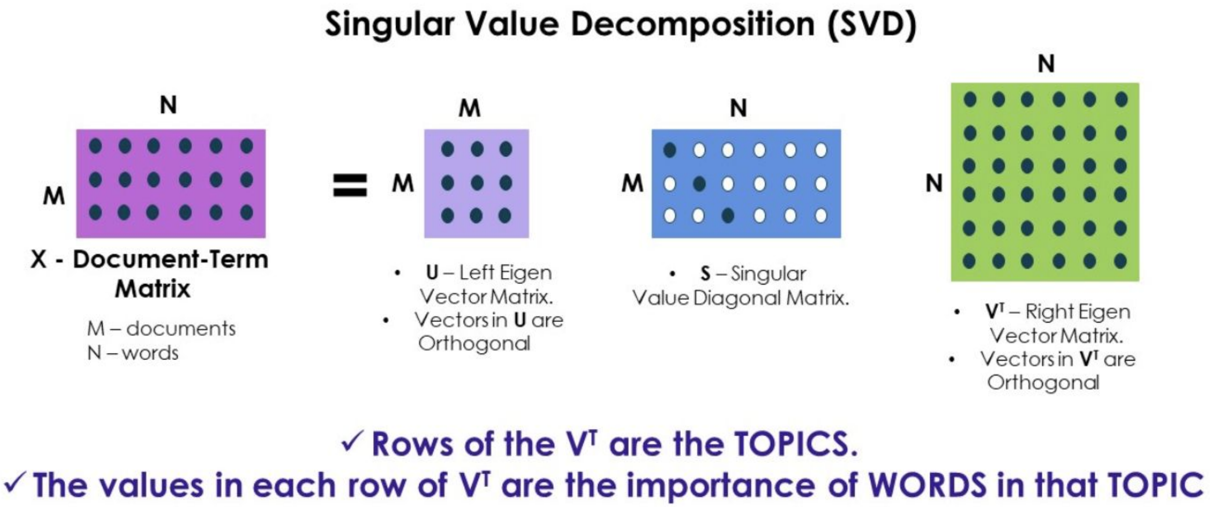
Topic models

Topic models are algorithms for discovering the main themes that pervade a large and otherwise unstructured collection of documents. Topic models can organize the collection according to the discovered themes.

Usually, this problem formulated as matrix factorization problem of document - word matrix.



Latent Semantic Analysis



Main drawback: this model is uninterpretable and based on wrong statistical assumptions.

Probabilistic Latent Semantic Analysis

Asymmetric: $P(d,w) = P(d)P(w|d), \; P(w|d) = \sum_{z \in \mathcal{Z}} P(w|z)P(z|d).$

Symmetric: $P(d,w) = \sum_{z \in \mathcal{Z}} P(z)P(d|z)P(w|z)$

This model can be understood as variant of NMF with KL divergence, when sum of elements in rows of factor matrices are restricted to be 1. MU algorithm for this model:

E-step equation

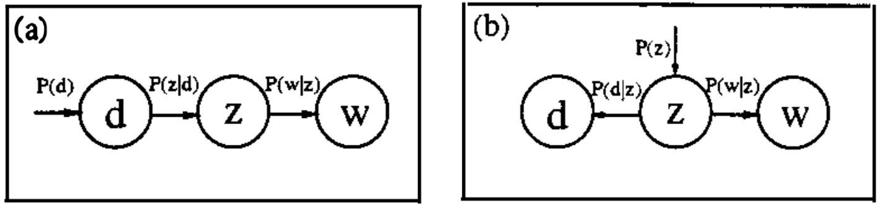
$$P(z|d,w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z' \in \mathcal{Z}} P(z')P(d|z')P(w|z')}, \quad (3)$$

as well as the following M-step formulae

$$P(w|z) \propto \sum_{d \in \mathcal{D}} n(d,w)P(z|d,w), \quad (4)$$

$$P(d|z) \propto \sum_{w \in \mathcal{W}} n(d,w)P(z|d,w), \quad (5)$$

$$P(z) \propto \sum_{d \in \mathcal{D}} \sum_{w \in \mathcal{W}} n(d,w)P(z|d,w). \quad (6)$$

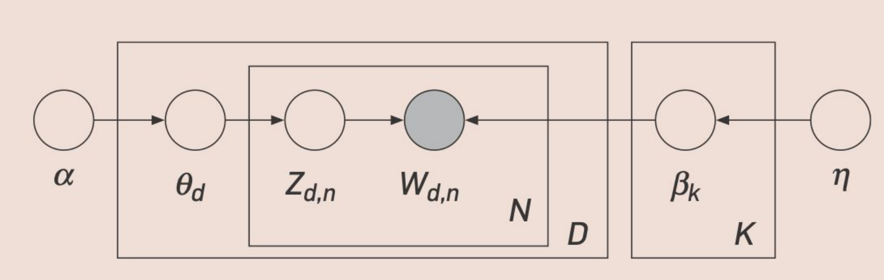


“segment 1”	“segment 2”	“matrix 1”	“matrix 2”	“line 1”	“line 2”	“power 1”	power 2”
imag	speaker	robust	manufactur	constraint	alpha	POWER	load
SEGMENT	speech	MATRIX	cell	LINE	redshift	spectrum	memori
texture	recogni	eigenvalu	part	match	LINE	omega	vlsi
color	signal	uncertainti	MATRIX	locat	galaxi	mpc	POWER
tissue	train	plane	cellular	imag	quasar	hsup	systolic
brain	hmm	linear	famili	geometr	absorp	larg	input
slice	source	condition	design	impos	high	redshift	complex
cluster	speakerind.	perturb	machinepart	segment	ssup	galaxi	arrai
mri	SEGMENT	root	format	fundament	densiti	standard	present
volume	sound	suffici	group	recogn	veloc	model	implement

Figure 3: Eight selected factors from a 128 factor decomposition. The displayed word stems are the 10 most probable words in the class-conditional distribution $P(w|z)$, from top to bottom in descending order.

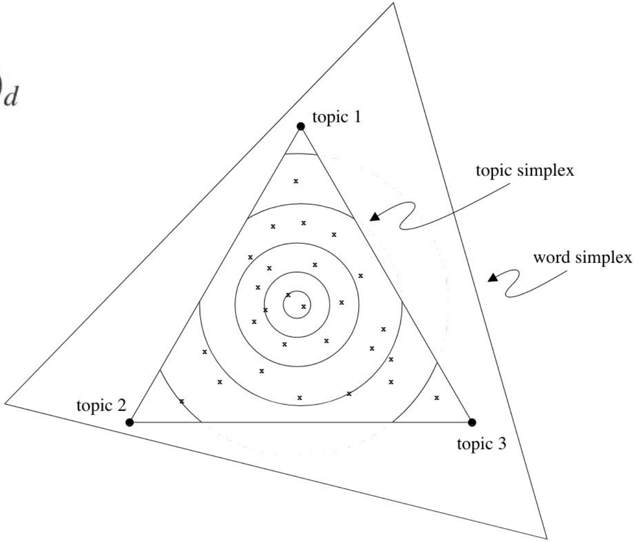
Main drawback: overfitting

Latent Dirichlet Allocation (LDA)

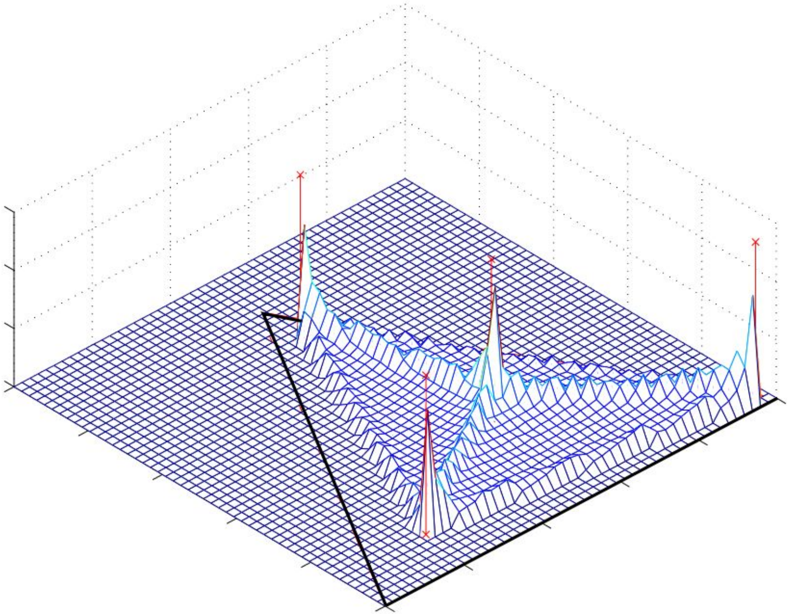


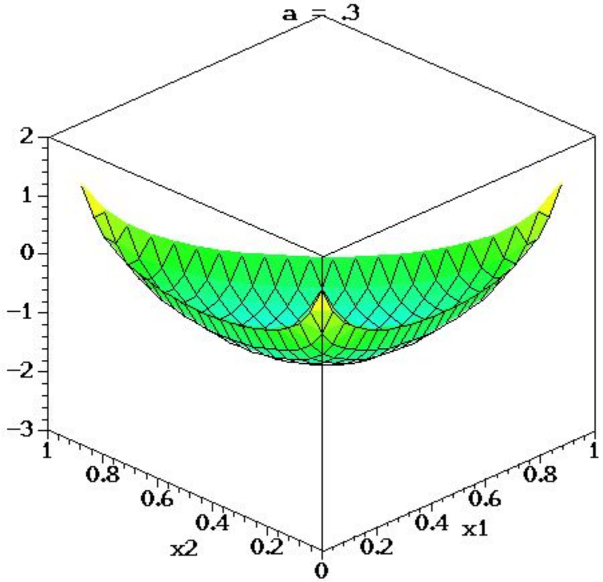
$$p(D\,|\,\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d\,|\,\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}\,|\,\theta_d) p(w_{dn}\,|\,z_{dn},\beta) \right) d\theta_d$$

$$(\gamma^*, \phi^*) = \arg\min_{(\gamma,\phi)} \mathrm{D}(q(\theta, \mathbf{z}\,|\,\gamma, \phi) \,\|\, p(\theta, \mathbf{z}\,|\,\mathbf{w}, \alpha, \beta))$$



Dirichlet prior





A Spectral Algorithm for LDA

Main idea: Method of moments for LDA.

Specifically, we set

$$x_t = e_i \quad \text{if and only if} \quad \text{the } t\text{-th word in the document is } i, \quad t \in [\ell],$$

where $e_1, e_2, \ldots e_d$ is the standard coordinate basis for $\mathbb{R}^d$.

One advantage of this encoding of words is that the (cross) moments of these random vectors correspond to joint probabilities over words. For instance, observe that

$$\begin{aligned}\mathbb{E}[x_1 \otimes x_2] &= \sum_{1 \le i,j \le d} \Pr[x_1 = e_i, x_2 = e_j]\; e_i \otimes e_j \\ &= \sum_{1 \le i,j \le d} \Pr[\text{1st word} = i, \text{2nd word} = j]\; e_i \otimes e_j,\end{aligned}$$

$$\mathbb{E}[x_t | h = j] \;=\; \sum_{i=1}^{d} \Pr[t\text{-th word} = i | h = j]\; e_i \;=\; \sum_{i=1}^{d} [\mu_j]_i\; e_i \;=\; \mu_j, \quad j \in [k]$$

A Spectral Algorithm for LDA

Theorem 3.5 (Anandkumar et al., 2012a) *Define*

$$
\begin{aligned}
M_1 &:= \mathbb{E}[x_1] \\
M_2 &:= \mathbb{E}[x_1 \otimes x_2] - \frac{\alpha_0}{\alpha_0+1} M_1 \otimes M_1 \\
M_3 &:= \mathbb{E}[x_1 \otimes x_2 \otimes x_3] \\
&\quad - \frac{\alpha_0}{\alpha_0+2}\Big(\mathbb{E}[x_1 \otimes x_2 \otimes M_1] + \mathbb{E}[x_1 \otimes M_1 \otimes x_2] + \mathbb{E}[M_1 \otimes x_1 \otimes x_2]\Big) \\
&\quad + \frac{2\alpha_0^2}{(\alpha_0+2)(\alpha_0+1)} M_1 \otimes M_1 \otimes M_1.
\end{aligned}
$$

Then

$$
\begin{aligned}
M_2 &= \sum_{i=1}^{k} \frac{\alpha_i}{(\alpha_0+1)\alpha_0}\, \mu_i \otimes \mu_i \\
M_3 &= \sum_{i=1}^{k} \frac{2\alpha_i}{(\alpha_0+2)(\alpha_0+1)\alpha_0}\, \mu_i \otimes \mu_i \otimes \mu_i.
\end{aligned}
$$

A Spectral Algorithm for LDA

First, let $W \in \mathbb{R}^{d \times k}$ be a linear transformation such that

$$M_2(W, W) = W^\top M_2 W = I$$

Now define $\widetilde{M}_3 := M_3(W, W, W) \in \mathbb{R}^{k \times k \times k}$, so that

$$\widetilde{M}_3 = \sum_{i=1}^{k} w_i \, (W^\top \mu_i)^{\otimes 3} = \sum_{i=1}^{k} \frac{1}{\sqrt{w_i}} \, \tilde{\mu}_i^{\otimes 3}$$

Algorithm 1 Robust tensor power method

input symmetric tensor $\tilde{T} \in \mathbb{R}^{k \times k \times k}$, number of iterations L, N.
output the estimated eigenvector/eigenvalue pair; the deflated tensor.

for $\tau = 1$ to L **do**
  Draw $\theta_0^{(\tau)}$ uniformly at random from the unit sphere in $\mathbb{R}^k$.
  for $t = 1$ to N **do**
    Compute power iteration update

$$\theta_t^{(\tau)} := \frac{\tilde{T}(I, \theta_{t-1}^{(\tau)}, \theta_{t-1}^{(\tau)})}{\|\tilde{T}(I, \theta_{t-1}^{(\tau)}, \theta_{t-1}^{(\tau)})\|} \tag{7}$$

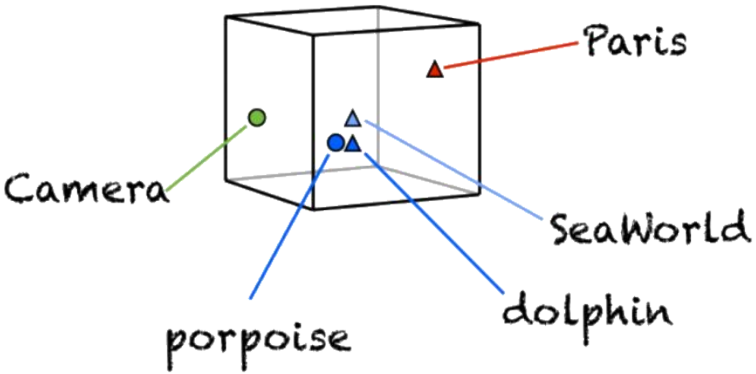
  end for
end for
Let $\tau^* := \arg\max_{\tau \in [L]} \{\tilde{T}(\theta_N^{(\tau)}, \theta_N^{(\tau)}, \theta_N^{(\tau)})\}$.
Do N power iteration updates (7) starting from $\theta_N^{(\tau^*)}$ to obtain $\hat{\theta}$, and set $\hat{\lambda} := \tilde{T}(\hat{\theta}, \hat{\theta}, \hat{\theta})$.
return the estimated eigenvector/eigenvalue pair $(\hat{\theta}, \hat{\lambda})$; the deflated tensor $\tilde{T} - \hat{\lambda}\, \hat{\theta}^{\otimes 3}$.

Representation learning

Representation learning

Definition of task from physical viewpoint:

- Our main task is to represent objects (words, documents, ...) as vectors in real space, in such way that similar objects, should have geometrically close vectors.



In practise this task allow us to pre-train vectors for different downstream tasks, which can significantly reduce sample complexity of downstream models. This is mostly **unsupervised task**.

Hellinger PCA

This model derived for word embedding task:

”You shall know a word by the company it keeps” (Firth, 1957)

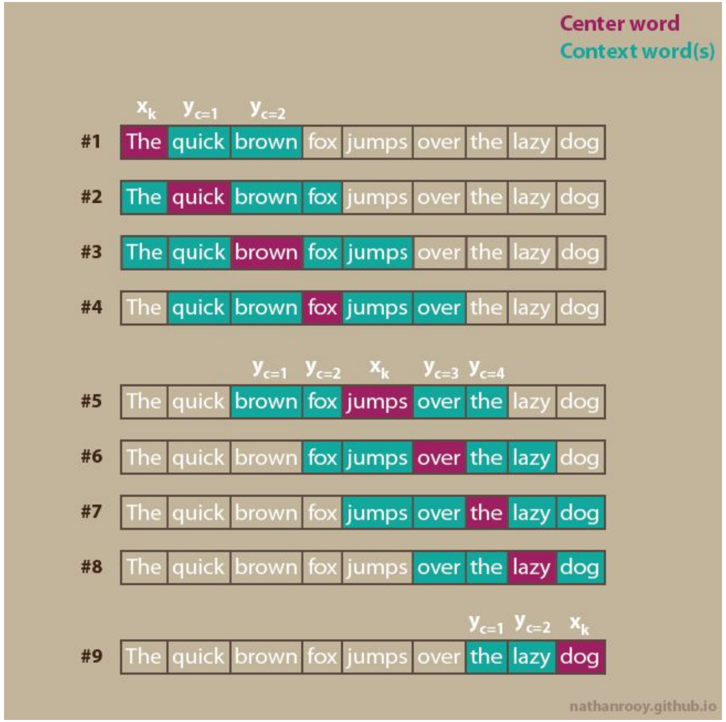
Let’s define sliding window and compute statistics

$$p(c|w) = \frac{p(c,w)}{p(w)} = \frac{n(c,w)}{\sum_c n(c,w)}$$

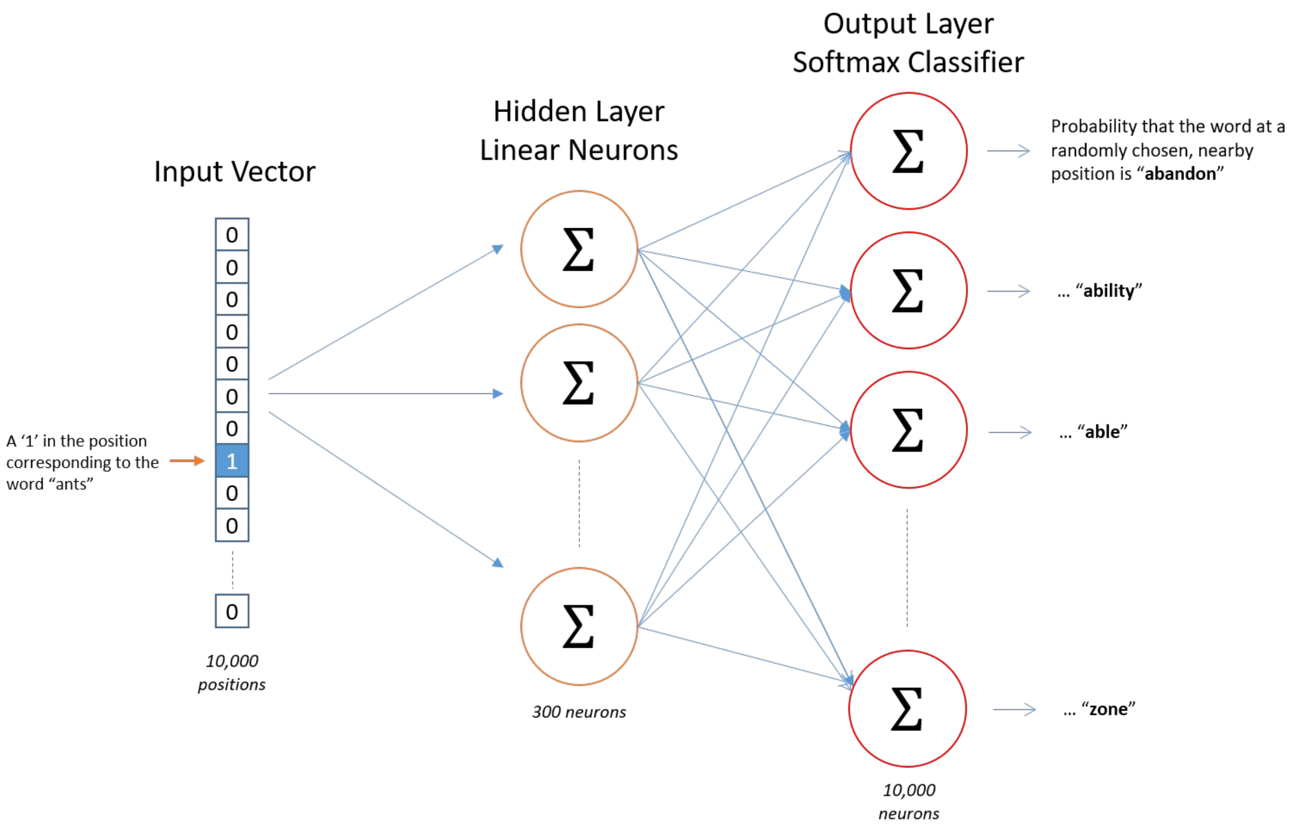
$$C = \begin{pmatrix} p(c_1|w_1) & \cdots & p(c_{|\mathcal{D}|}|w_1) \\ p(c_1|w_2) & \cdots & p(c_{|\mathcal{D}|}|w_2) \\ \vdots & \ddots & \vdots \\ p(c_1|w_{|\mathcal{W}|}) & \cdots & p(c_{|\mathcal{D}|}|w_{|\mathcal{W}|}) \end{pmatrix} = \begin{pmatrix} P_{w_1} \\ P_{w_2} \\ \vdots \\ P_{w_{|\mathcal{W}|}} \end{pmatrix}$$

SLRA: $||VU^T\sqrt{P_w} - \sqrt{P_w}||^2$

We find U and V via SGD



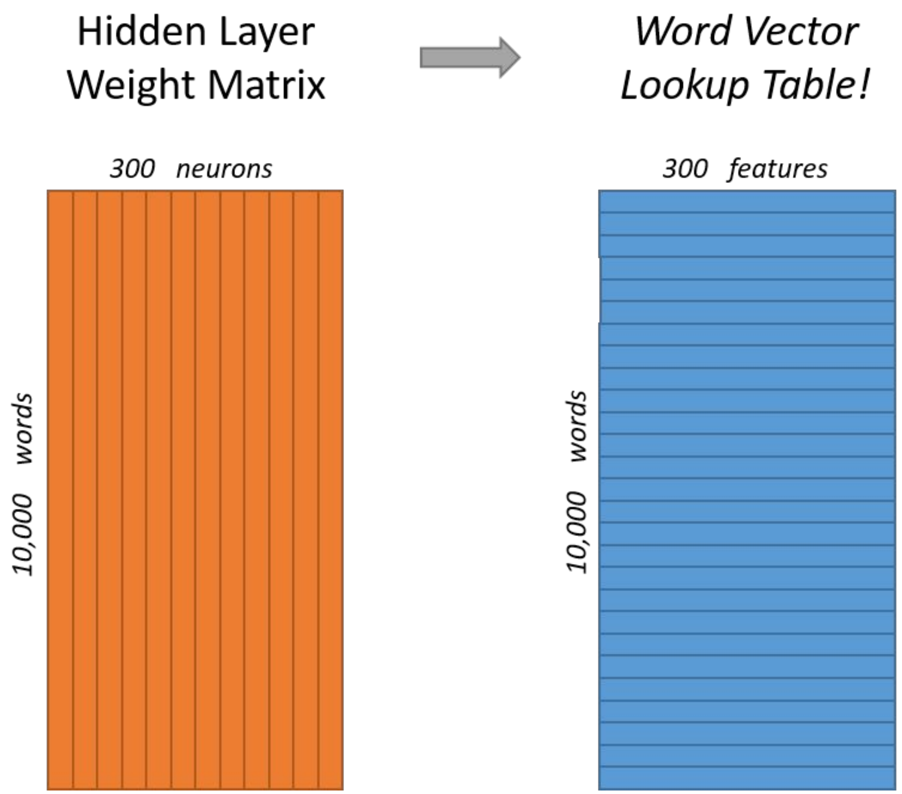
Word2vec as Shallow Neural Network



Word2vec as Matrix Factorization

We have two factor matrices:

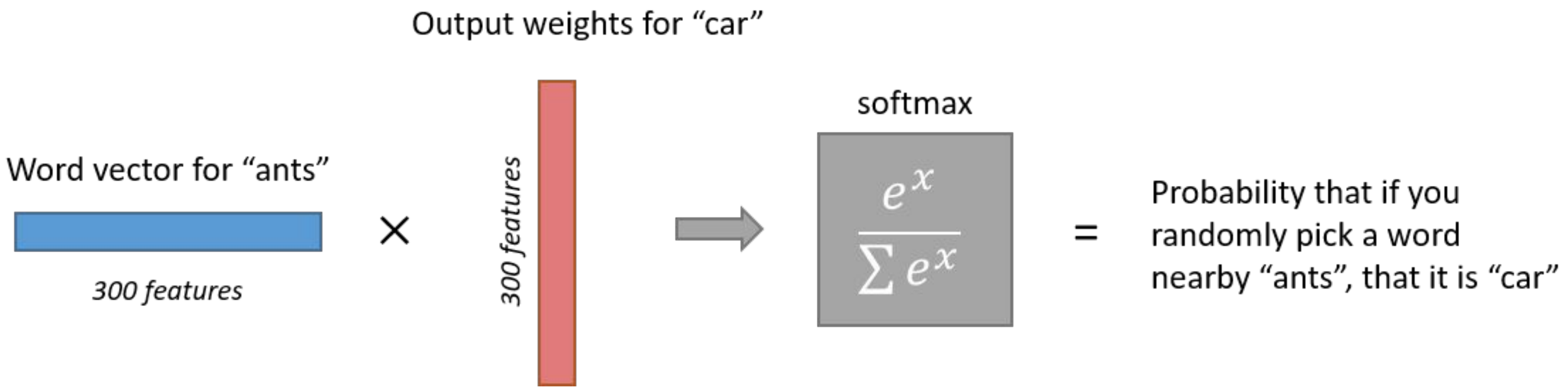
First “hidden” called context matrix and second called word matrix



Word2vec as Matrix Factorization

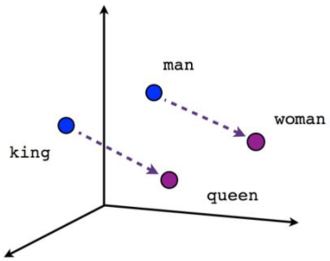
If we rewrite any matrix factorization (NMF, PCA) in entrywise fashion, we optimize dot product similarity between two vectors.

The same holds for word2vec.

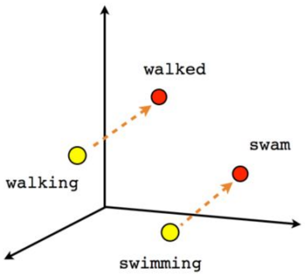


Impressive result of word2vec

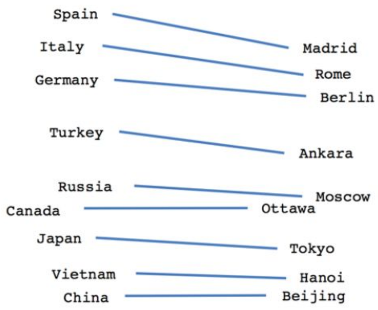
Full geometric structure of vector space now became meaningful.



Male-Female



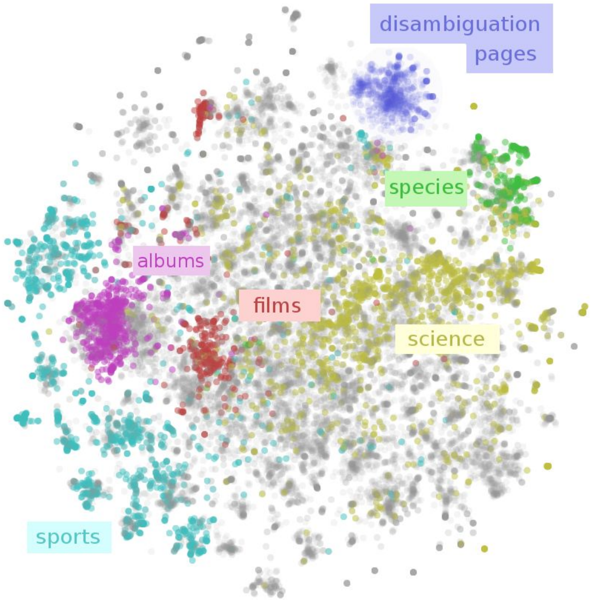
Verb tense



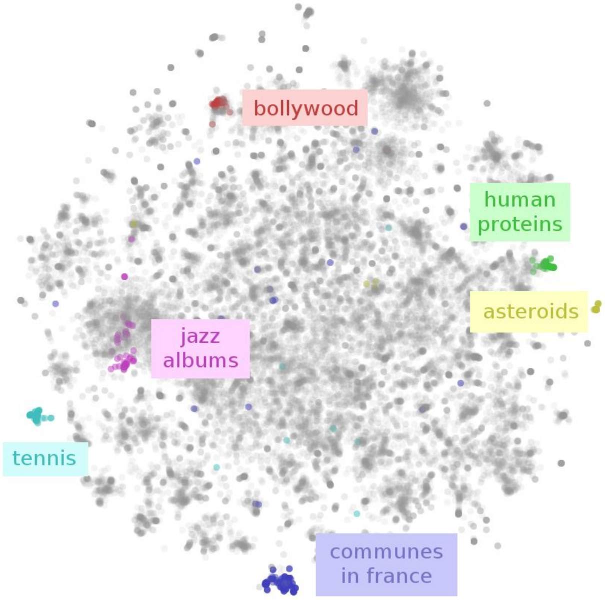
Country-Capital

Semantic similarity

Large Clusters



Small Clusters

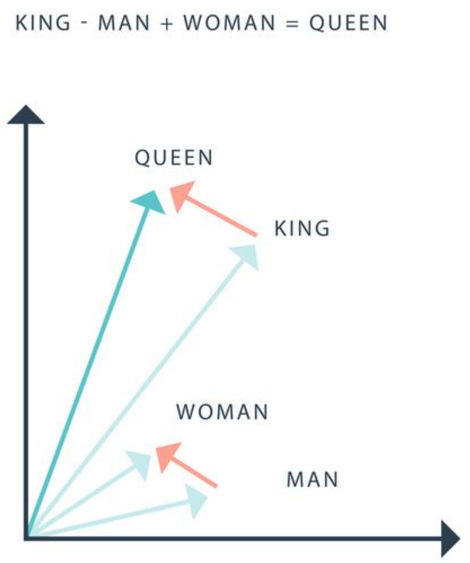


Semantic analogy

We can extract for our trained matrices 3 vectors for king, man, woman and calculate new vector:

vec(X) = vec(“king”) - vec(“man”) + vec(“woman”)

Closest in cosine distance to this vector is vector which correspond to “queen”.



GloVe

It’s attempt to incorporate Word2vec (predictive) ideas in count-based framework (like Hellinger PCA).

Main idea: Using log-transform to recast logistic regression of word2vec to ordinary L2 regression.

We learn homomorphism between additive group of real numbers and multiplicative group of positive numbers:

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}$$

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i)$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

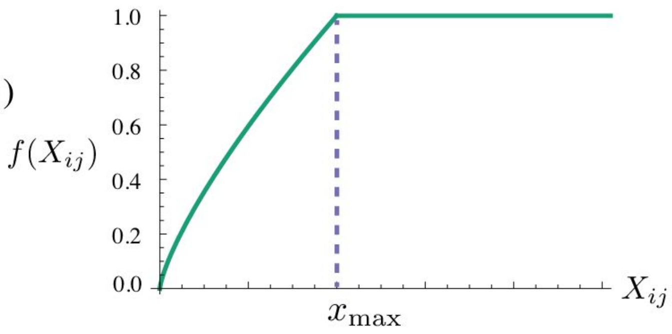


Figure 1: Weighting function f with $\alpha = 3/4$.

Similarity measure from W2V

Idea is to solve Word2vec local objective function for one specific pair of word and context:

$$\ell(w,c) = \#(w,c)\log\sigma(\vec{w}\cdot\vec{c}) + k\cdot\#(w)\cdot\frac{\#(c)}{|D|}\log\sigma(-\vec{w}\cdot\vec{c})$$

To optimize the objective, we define $x = \vec{w}\cdot\vec{c}$ and find its partial derivative with respect to x:

$$\frac{\partial\ell}{\partial x} = \#(w,c)\cdot\sigma(-x) - k\cdot\#(w)\cdot\frac{\#(c)}{|D|}\cdot\sigma(x)$$

We compare the derivative to zero, and after some simplification, arrive at:

$$e^{2x} - \left(\frac{\#(w,c)}{k\cdot\#(w)\cdot\frac{\#(c)}{|D|}} - 1\right)e^{x} - \frac{\#(w,c)}{k\cdot\#(w)\cdot\frac{\#(c)}{|D|}} = 0$$

$$y = \frac{\#(w,c)}{k\cdot\#(w)\cdot\frac{\#(c)}{|D|}} = \frac{\#(w,c)\cdot|D|}{\#w\cdot\#(c)}\cdot\frac{1}{k}$$

Substituting y with e^x and x with $\vec{w}\cdot\vec{c}$ reveals:

$$\vec{w}\cdot\vec{c} = \log\left(\frac{\#(w,c)\cdot|D|}{\#(w)\cdot\#(c)}\cdot\frac{1}{k}\right) = \log\left(\frac{\#(w,c)\cdot|D|}{\#(w)\cdot\#(c)}\right) - \log k$$

PPMI decomposition via Symmetric SVD

When representing words, there is some intuition behind ignoring negative values: humans can easily think of positive associations (e.g. “Canada” and “snow”) but find it much harder to invent negative ones (“Canada” and “desert”). This suggests that the perceived similarity of two words is more influenced by the positive context they share than by the negative context they share.

$$PPMI(w,c) = \max\left(PMI\left(w,c\right),0\right)$$

$$SPPMI_k(w,c) = \max\left(PMI\left(w,c\right) - \log k, 0\right)$$

$$W^{\text{SVD}_{1/2}} = U_d \cdot \sqrt{\Sigma_d} \qquad C^{\text{SVD}_{1/2}} = V_d \cdot \sqrt{\Sigma_d}$$

WS353 (WordSim) [13]			MEN (WordSim) [4]			Mixed Analogies [20]			Synt. Analogies [22]		
Representation		Corr.	Representation		Corr.	Representation		Acc.	Representation		Acc.
SVD	(k=5)	0.691	SVD	(k=1)	0.735	SPPMI	(k=1)	0.655	SGNS	(k=15)	0.627
SPPMI	(k=15)	0.687	SVD	(k=5)	0.734	SPPMI	(k=5)	0.644	SGNS	(k=5)	0.619
SPPMI	(k=5)	0.670	SPPMI	(k=5)	0.721	SGNS	(k=15)	0.619	SGNS	(k=1)	0.59
SGNS	(k=15)	0.666	SPPMI	(k=15)	0.719	SGNS	(k=5)	0.616	SPPMI	(k=5)	0.466
SVD	(k=15)	0.661	SGNS	(k=15)	0.716	SPPMI	(k=15)	0.571	SVD	(k=1)	0.448
SVD	(k=1)	0.652	SGNS	(k=5)	0.708	SVD	(k=1)	0.567	SPPMI	(k=1)	0.445
SGNS	(k=5)	0.644	SVD	(k=15)	0.694	SGNS	(k=1)	0.540	SPPMI	(k=15)	0.353
SGNS	(k=1)	0.633	SGNS	(k=1)	0.690	SVD	(k=5)	0.472	SVD	(k=5)	0.337
SPPMI	(k=1)	0.605	SPPMI	(k=1)	0.688	SVD	(k=15)	0.341	SVD	(k=15)	0.208

Table 2: A comparison of word representations on various linguistic tasks. The different representations were created by three algorithms (SPPMI, SVD, SGNS) with $d = 1000$ and different values of k.

PPMI decomposition via Symmetric CP

PMI can be generalized to N variables:

$$PMI(x_1^N) = \log \frac{p(x_1, \ldots, x_N)}{p(x_1) \cdots p(x_N)}$$

$$x_{ijk} \approx \sum_{r=1}^{R} u_{ir} v_{jr} w_{kr} = \langle \mathbf{u}_{:,i} * \mathbf{v}_{:,j}, \mathbf{w}_{:,k} \rangle$$

$$\mathcal{L}^{(3)}(\mathcal{M}^t, \mathbf{U}) = \sum_{m_{ijk}^t \in \mathcal{M}^t} \left(m_{ijk}^t - \sum_{r=1}^{R} u_{ir} u_{jr} u_{kr}\right)^2$$

$$\mathcal{L}_{\text{joint}}((\mathcal{M}^t)_{n=2}^N, \mathbf{U}) = \sum_{n=2}^{N} \mathcal{L}^{(n)}(\mathcal{M}_n^t, \mathbf{U})$$

Table 4: Word Similarity Scores (Spearman's ρ)

(Method)	MEN	MTurk	RW	SimLex999
Random	0.04147	-0.0382	-0.0117	0.0053
SGNS	0.5909	0.5342	**0.3704**	0.2264
CBOW	0.5537	0.4225	0.3444	**0.2727**
NNSE	0.5055	0.5068	0.1993	0.1263
GloVe	0.4914	0.4733	0.1750	0.1403
CP-S	0.4723	0.4738	0.0875	0.0399
JCP-S	**0.6158**	**0.5343**	0.3546	0.2272

Table 5: Nearest neighbors (in cosine similarity) to elementwise products of word vectors

Composition	Nearest neighbors (CP-S)	Nearest neighbors (JCP-S)	Nearest neighbors (CBOW)
star $*$ *actor*	*oscar, award-winning, supporting*	*roles, drama, musical*	*DNA, younger, tip*
star $+$ *actor*	*stars, movie, actress*	*actress, trek, picture*	*actress, comedian, starred*
star $*$ *planet*	*planets, constellation, trek*	*galaxy, earth, minor*	*fingers, layer, arm*
star $+$ *planet*	*sun, earth, galaxy*	*galaxy, dwarf, constellation*	*galaxy, planets, earth*
tank $*$ *fuel*	*liquid, injection, tanks*	*vehicles, motors, vehicle*	*armored, tanks, armoured*
tank $+$ *fuel*	*tanks, engines, injection*	*vehicles, tanks, powered*	*tanks, engine, diesel*
tank $*$ *weapon*	*gun, ammunition, tanks*	*brigade, cavalry, battalion*	*persian, age, rapid*
tank $+$ *weapon*	*tanks, armor, rifle*	*tanks, battery, batteries*	*tanks, cannon, armored*

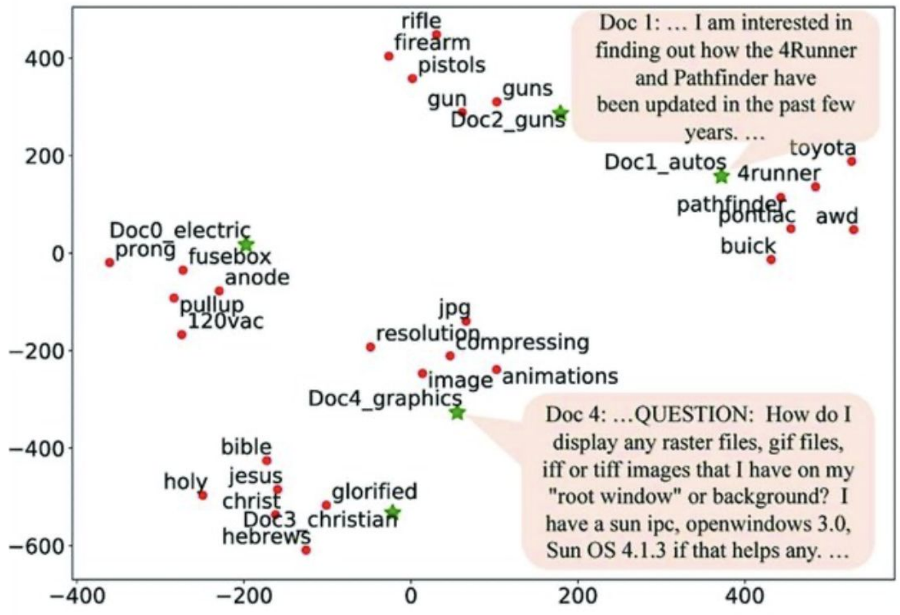
RDM

Document embedding

- In the document embedding problem we try to represent documents as vectors of features in such way that vectors can capture all information from documents suitable for solving any document classification tasks.

- For solving this task we usually learn representation of both documents and words from occurrence information of words inside documents.

- **Our main hypothesis** is that a novel <u>tensor models</u> combined with <u>geometrical constraints</u> can achieve significant boost in classification accuracy in comparison with other models in this task.



Main intuition

We use the fact that ***each document can be represented in different ways via connection with different sets of n-grams with fixed lengths.***

Example:

“ An annual meeting is held each summer in locations where significant computational linguistics research is carried out.”

➔ 1-gram representation: an, annual, meeting, is, held, each, summer, in, locations, where, significant, computational, linguistics, research, is, carried, out

➔ 2-gram representation: an annual, annual meeting, meeting is, is held, held each, ...

RDM

Word: $\mathbf{U}_w \in \mathbb{R}^{R\times R} \longrightarrow \mathbf{U}_{\mathbf{w}} = \prod_{k=1}^{n} \mathbf{U}_{w_k}$ (N-gram)

Document: $\mathbf{V}_d \in \mathbb{R}^{R\times R}$

Similarity measure: $\langle \mathbf{U}_{\mathbf{w}}, \mathbf{V}_d \rangle_F = \mathrm{tr}(\mathbf{U}_{\mathbf{w}}^T \mathbf{V}_d) = \mathrm{tr}(\mathbf{U}_{\mathbf{w}} \mathbf{V}_d^T) = \mathrm{vec}(\mathbf{U}_{\mathbf{w}})^T \mathrm{vec}(\mathbf{V}_d)$

Ordinary TC: $x_{i_1 i_2 \ldots i_N} = \mathrm{tr}(\mathbf{A}_{i_1}^{(1)} \mathbf{A}_{i_2}^{(2)} \cdots \mathbf{A}_{i_N}^{(N)})$

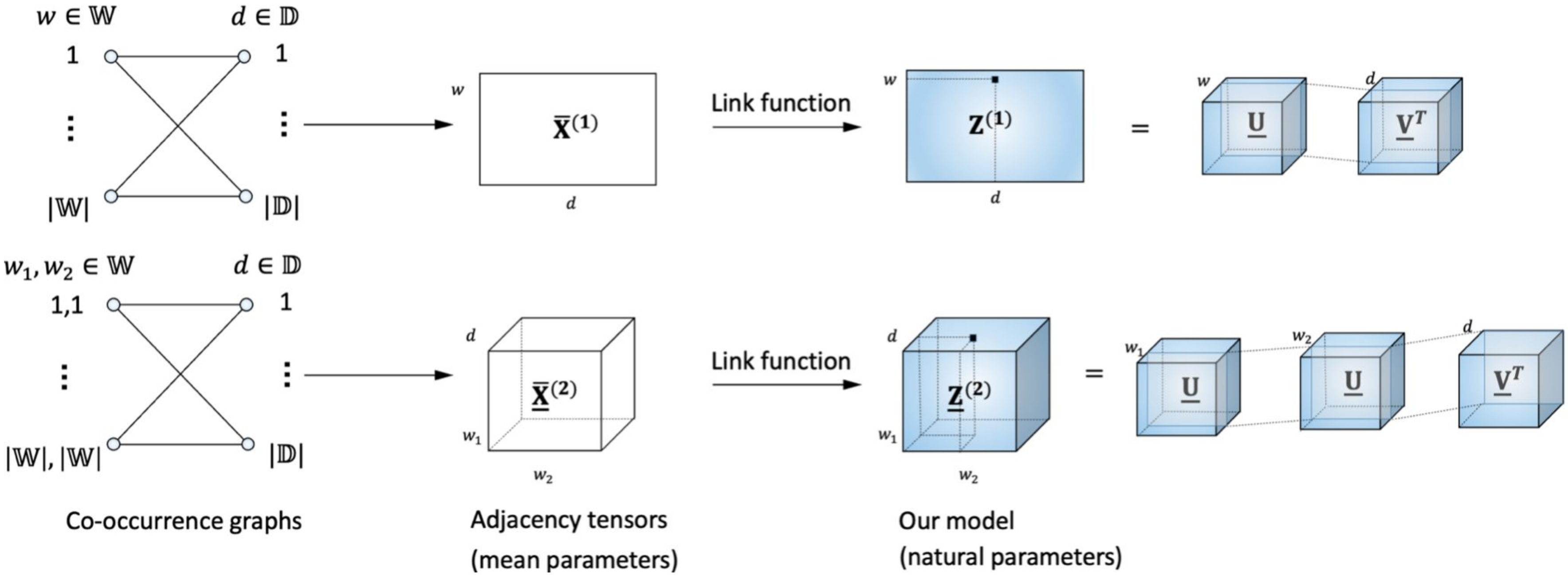
RDM:

$$\mathrm{KL}[\bar{\underline{\mathbf{X}}}^{(n)} || \hat{\underline{\mathbf{X}}}^{(n)}; \Theta] = \sum_{\mathbf{w}\in\mathbb{W}^n} \sum_{d\in\mathbb{D}} \bar{x}_{\mathbf{w}d}^{(n)} \log \left(\frac{\bar{x}_{\mathbf{w}d}^{(n)}}{\hat{x}_{\mathbf{w}d}^{(n)}} \right)$$

$$= \sum_{w_1=1}^{|W|} \cdots \sum_{w_n=1}^{|W|} \sum_{d=1}^{|D|} \bar{x}_{w_1\ldots w_n d}^{(n)} \log \left(\frac{\bar{x}_{w_1\ldots w_n d}^{(n)}}{\hat{x}_{w_1\ldots w_n d}^{(n)}} \right),$$

and $\hat{x}_{\mathbf{w}d}^{(n)} = p(\mathbf{w}, d)$.

RDM


$w \in \mathbb{W}$
$d \in \mathbb{D}$
1
1
$|\mathbb{W}|$
$|\mathbb{D}|$
$w_1, w_2 \in \mathbb{W}$
$d \in \mathbb{D}$
1,1
1
$|\mathbb{W}|, |\mathbb{W}|$
$|\mathbb{D}|$
$\bar{\mathbf{X}}^{(1)}$
$\underline{\bar{\mathbf{X}}}^{(2)}$
Link function
Link function
$\mathbf{Z}^{(1)}$
$\underline{\mathbf{Z}}^{(2)}$
$=$
$\underline{\mathbf{U}}$
$\underline{\mathbf{V}}^T$
$=$
$\underline{\mathbf{U}}$
$\underline{\mathbf{U}}$
$\underline{\mathbf{V}}^T$
Co-occurrence graphs
Adjacency tensors
(mean parameters)
Our model
(natural parameters)

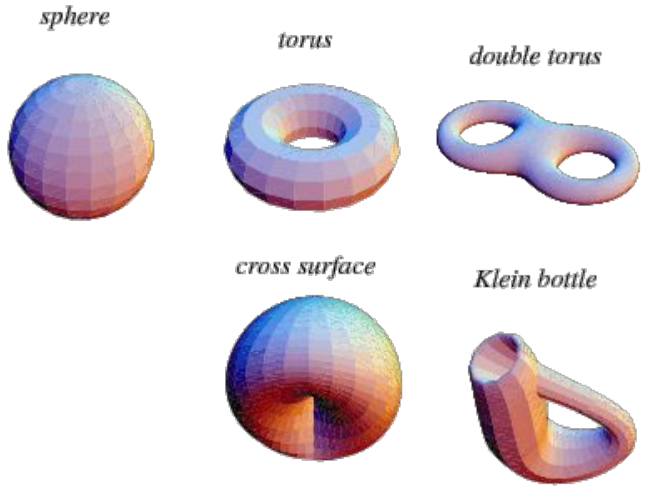
Manifold constraints

- In ML we have a common belief that data internally lay on the low-dimensional manifold. Thus, if we impose some geometric constraints on the model parameters (factor matrices, cores, vectors), our model can perform more robustly in practice.

- However, manifold constraints can define a challenging task, making optimization complicated. It is an art to work with the tradeoffs of these constraints.

- In my task, I impose rotation manifold constraints on the document and word representations. These constraints transform the similarity function in my model to be a cosine angle. Intuitively, if the angle between two vectors corresponding to some word and document is small then these word and document co-occur more frequently in our data.

$$\mathrm{SO}(R) = \{\mathbf{A} | \mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}, \det(\mathbf{A}) = +1\}$$



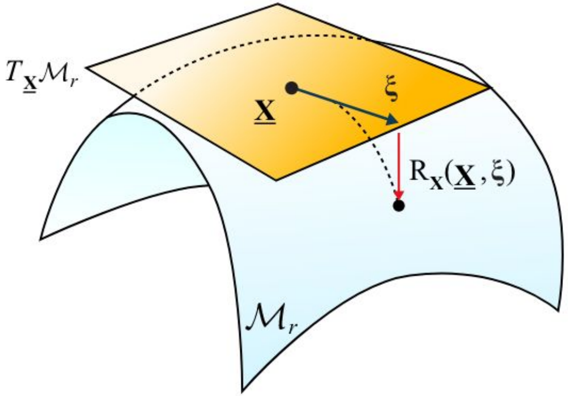
Riemannian optimization

For rotation manifold projection on the tangent plane defined as :

$$\mathrm{proj}_{\mathbf{A}}(\mathbf{G}) = \frac{1}{2}(\mathbf{G} - \mathbf{A}\mathbf{G}^T\mathbf{A}),$$

With QR -based retraction:

$$[\mathbf{Q}, \mathbf{R}] = \mathrm{QR}\left(\mathbf{A}_t - \alpha \nabla_R \mathcal{L}(\mathbf{A}_t)\right)$$
$$\mathbf{A}_{t+1} = \mathbf{Q}.$$



Noise Contrastive Estimation

$$
\begin{aligned}
\min_{\Theta} \quad & \sum_{n=1}^{N} \mathcal{L}^{(n)}(\Theta) + \frac{\lambda}{2} \sum_{n=1}^{N} (\kappa^{(n)})^2 \\
\text{s.t.} \quad & \mathbf{U}_w \in \mathrm{SO}(R), \quad w = 1, \ldots, |\mathbb{W}| \\
& \mathbf{V}_d \in \mathrm{SO}(R), \quad d = 1, \ldots, |\mathbb{D}| \\
& \kappa^{(n)} \geq 0, \quad n = 1, \ldots, N,
\end{aligned}
$$

where each risk function is equal to

$$
\mathcal{L}^{(n)}(\Theta) = \mathbb{E}_{\{(\mathbf{w}_i, d_i)\}_{i=1}^{I} \sim \prod_{i=1}^{I} \bar{x}^{(n)}_{\mathbf{w}_i d_i}}
- \log \frac{\exp\left(\kappa^{(n)} \mathrm{tr}(\mathbf{U}_{\mathbf{w}_i} \mathbf{V}_{d_i}^T)\right)}{\sum_{j=1}^{I} \exp\left(\kappa^{(n)} \mathrm{tr}(\mathbf{U}_{\mathbf{w}_j} \mathbf{V}_{d_i}^T)\right)}.
$$

$$
\Theta = \{\mathbf{U}_w\}_{w=1}^{|\mathbb{W}|} \cup \{\mathbf{V}_d\}_{d=1}^{|\mathbb{D}|} \cup \{\kappa^{(n)}\}_{n=1}^{N},
$$

We add concentration parameters $\kappa^{(n)}$ to our loss function to overcome the problem of fixed scale. This makes our model more flexible to represent sharp distributions. Due to the fact that each n-gram distribution has its own scale, it can be reasonable to have a different for differe $\kappa^{(n)}$ gram distributions.

Experimental setup

In the experimental part we work with the following pipeline:

1. Model learn representation of documents (document embedding).
2. We teach a linear classifier on top of learned vectors to solve one particular document classification problem.
3. We estimate the quality of classification.

Datasets: 20 Newsgroups (short documents) and ArXiv (long documents).

We use NLTK stopword list for preprocessing and nested 10-fold CV with Wilcoxon Signed rank test for statistical significance estimation.

Method	Time	Space
PV-DBOW	$O(kd)$	$O(d)$
PV-DM (Concat)	$O(kd)$	$O(kd)$
ELMo	$O(kld^2)$	$O(ld^2)$
BERT	$O(k^2hld)$	$O(khld)$
RDM (Ours)	$O(kd^{1.5})$	$O(d)$

Dataset	#cls	$\|\mathbb{W}\|$	$\|\mathbb{D}\|$	#w
20 Newsgroups	20	75752	18846	180
ArXiv	6	251108	16371	3829

Experimental results

- RDM is the proposed model RDM-R is RDM but without rotation constraints.
- (1), (3), (5) show the maximum length of n-gram.
- RDM uses 400D vectors. (n=20)
- RDM has less degrees-of-freedom than RDM-R ($\frac{n(n-1)}{2}$ vs n^2).

Models	20 Newsgroups				ArXiv			
	Acc	**Prec**	**Rec**	**F1**	**Acc**	**Prec**	**Rec**	**F1**
PV-DBOW	88.7	88.4	88.2	88.2	89.1	89.2	89.2	89.2
PV-DM	77.2	76.8	76.5	76.5	42.2	42.3	42.0	41.9
Skipgram+Average	90.3	90.2	90.0	90.0	92.4	92.3	92.3	92.3
Skipgram+TF-IDF	90.4	90.2	90.1	90.1	92.6	92.5	92.4	92.4
Skipgram+SIF	90.4	90.2	90.1	90.1	92.4	92.3	92.3	92.3
Sent2vec	87.9	87.6	87.5	87.5	91.9	91.7	91.7	91.7
Doc2vecC	90.0	89.8	89.7	89.7	93.2	93.1	93.1	93.1
JoSe	87.8	87.6	87.4	87.4	91.3	91.3	91.2	91.2
ELMo	79.2	78.8	78.8	78.7	91.6	91.5	91.4	91.4
BERT	74.3	73.6	73.6	73.5	92.3	92.2	92.1	92.1
Sentence BERT	79.5	79.1	79.0	79.0	89.0	88.9	88.8	88.8
RDM-R (1)	86.8	86.4	86.3	86.3	94.0*	93.9*	93.8*	93.8*
RDM-R (3)	87.9	87.6	87.4	87.4	94.5*	**94.4***	**94.4***	**94.4***
RDM-R (5)	88.3	88.0	87.9	87.9	**94.6***	**94.4***	**94.4***	**94.4***
RDM (1)	89.3	89.2	89.0	89.0	94.0*	93.9*	93.9*	93.9*
RDM (3)	90.7	90.5	90.4	90.4	94.0*	93.9*	93.9*	93.9*
RDM (5)	**91.1***	**90.9***	**90.8***	**90.8***	94.0*	93.9*	93.9*	93.9*

Conclusion

- RDM show that representation of the word compositionality via Tensor Chain model can increase quality of embedding.
- Rotation group constraint prevents the model from overfitting on the collections with short documents.
- RDM is computationally efficient alternative to deep learning models (i.e. ELMo (RNN) and BERT (Transformer)) for the document embedding task.

Thank you for your attention! :)