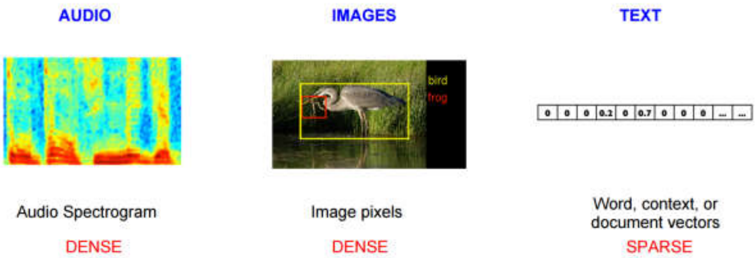
智能系统原理与开发
第08章 潜在语义分析

李斌
复旦大学 计算机科学技术学院

Why Word Embedding

- Natural language processing systems traditionally treat words as discrete atomic symbols, provide no useful information to the system regarding the relationships that may exist between the individual symbols



One-Hot Representation

- Represent a word as a one-hot vector
  - Example: He studies machine learning

Dictionary									
	He	studies	machine	learning	is	interesting	supports	big	data
v_{He}	1	0	0	0	0	0	0	0	0
v_{is}	0	0	0	0	1	0	0	0	0
v_{big}	0	0	0	0	0	0	0	1	0
v_{data}	0	0	0	0	0	0	0	0	1

- How large is this dictionary (universe set)?
  - Penn Treebank dataset: ~50K
  - Google 1T dataset: 13M

Issues of One-Hot Vector

- High-dimensional
- Sparse
- Fixed dimensionality (cannot represent new words)
- Orthogonal semantic similarity between pair of words

Dictionary							
	king	queen	professor	interesting	supports	big	data
v_{king}	1	0	0	0	0	0	0
v_{queen}	0	1	0	0	0	0	0
$v_{professor}$	0	0	1	0	0	0	0

$$\langle v_{king}, v_{queen} \rangle = \langle v_{king}, v_{professor} \rangle = 0$$

Distributional Representation

- "You shall know a word by the company it keeps" (John R. Firth, 1957)
- A word is characterized by its context

Dictionary						
	royal	palace	duke	speech	university	research
v_{king}	1	1	1	1	0	0
v_{queen}	1	1	1	1	0	0
$v_{professor}$	0	0	0	1	1	1

$$\langle v_{king}, v_{queen} \rangle > \langle v_{king}, v_{professor} \rangle = \langle v_{queen}, v_{professor} \rangle$$

- Still not good enough ⋯

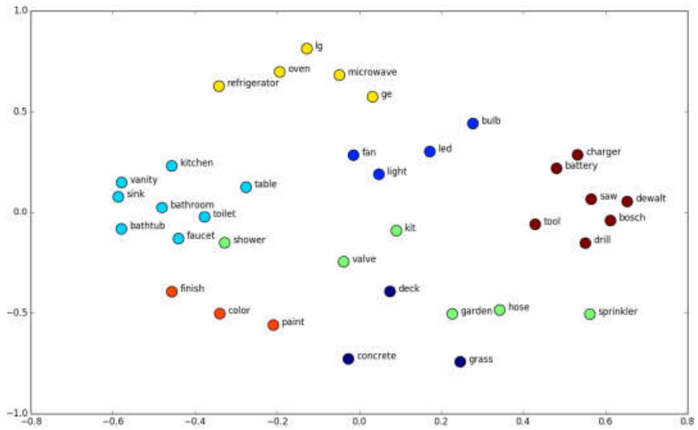
Vector Representation

- The vector space is spanned by semantic "concepts"
- Each word is represented by a distribution of weights over these concepts
  - The representation of a word is spread across all of the concepts in the vector
  - Each concept in the vector contributes to the definition of many words

Concepts				
	Royalty	Masculinity	Femininity	Celebrity
v_{king}	0.9	0.9	0.1	0.9
v_{queen}	0.8	0.2	0.9	0.8
v_{actor}	0.1	0.8	0.2	0.7

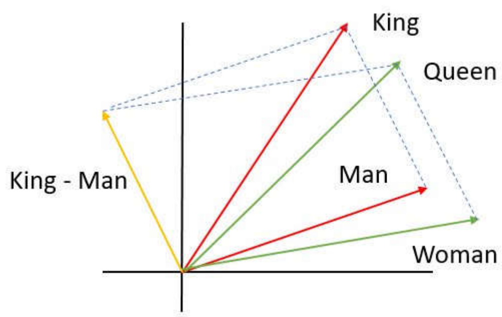
Vector Representation

- An illustration of 2-D vector representation



How to Learn Word Vectors?

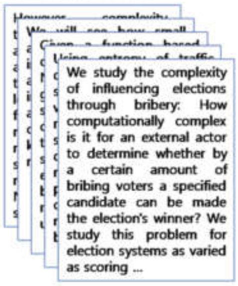
- How to find semantic concepts – bases
- How to assign weights – vectors
- How to define similarity/distance – metric



A Simple Vector Representation

- A word is represented by the documents (bases of the vector space) in which it appears
- A document is represented by the words it contain (i.e., bag-of-words representation for the document)

Documents



Vector-space representation

	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

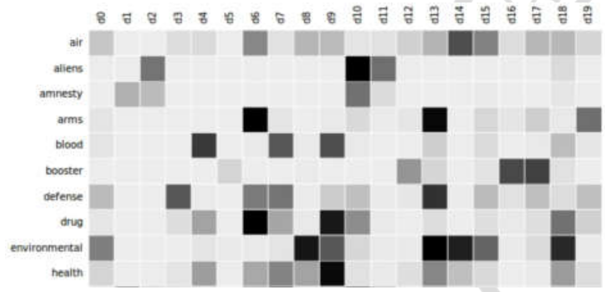
Term-document matrix

Issues of Doc-Word Co-occurrence

- Number of concepts (bases) is too large
- Concepts (bases) are not orthogonal
- High-dimensional
- Sparse
- etc.

Latent Semantic Analysis

- Represent a corpus as a document-word co-occurrence matrix (frequency, tf-idf, etc.)



- Factorize the document-word co-occurrence matrix to find latent components (semantic concepts)

Latent Semantic Analysis

- Latent semantic analysis (LSA) is a technique of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.
- LSA assumes that words that are close in meaning will occur in similar documents (the distributional hypothesis).
- LSA applies singular value decomposition (SVD) to find latent concepts $A = USV^{\top}$
- Words are then compared by taking the cosine of the angle between the two vectors.

Latent Semantic Analysis

- LSA applies singular value decomposition (SVD) to find latent concepts $A = USV^\top$
  - A: $m \times n$ word-document co-occurrence matrix
  - U: $m \times k$ orthogonal matrices for representing words
  - V: $n \times k$ orthogonal matrices for representing documents
  - S: $k \times k$ diagonal singular value matrix
  - Select $k' \ll n, k' \ll m$ for a low-rank approximation of A

A = **U** x **S** x **Vt**

A:

	d1	d2	d3	d4
a	6	7	1	0
b	8	6	0	1
c	6	9	8	5
d	0	1	8	8
e	2	0	9	7
f	2	0	7	7

U:

	f1	f2	f3	f4
a	0.24	-0.51	0.08	0.06
b	0.25	-0.54	-0.64	-0.23
c	0.58	-0.28	0.57	0.13
d	0.42	0.37	0.16	-0.68
e	0.44	0.34	-0.24	0.66
f	0.39	0.29	-0.40	-0.09

S:

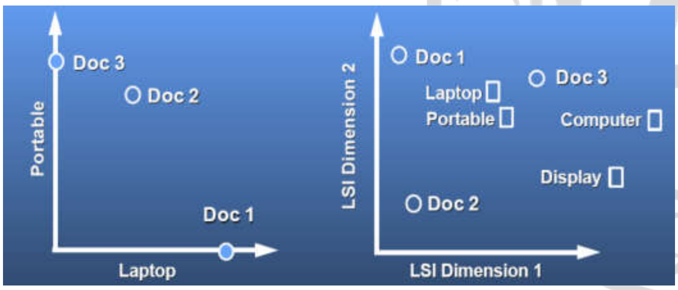
	f1	f2	f3	f4
f1	23.1	0	0	0
f2	0	14.3	0	0
f3	0	0	3.5	0
f4	0	0	0	1.5

Vt:

	d1	d2	d3	d4
f1	0.37	0.38	0.65	0.53
f2	-0.55	-0.63	0.37	0.38
f3	-0.69	0.59	0.27	-0.21
f4	0.26	-0.29	0.59	-0.69

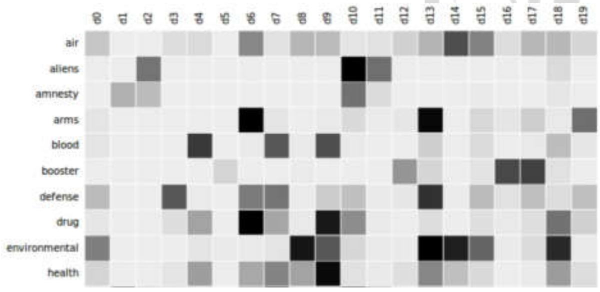
Latent Semantic Analysis

- After applying SVD to the word-document co-occurrence matrix and obtain the factorization $A = USV^\top$
  - U: similar words have large inner products
  - V: similar documents have large inner products
  - Related word and document have large inner products



Probabilistic LSA

- Probabilistic LSA (PLSA) is a statistical technique for the analysis of co-occurrence matrix.
- Compared to standard LSA stemming from a low-rank decomposition (SVD), PLSA is based on a mixture decomposition derived from a latent class model



PLSA Model

- Observations in the form of co-occurrences (w, d) of words and documents
- PLSA models the probability of (w, d) as a mixture of conditionally independent multinomial distributions

$$p(w,d) = \sum_z p(z)p(d|z)p(w|z) = p(d)\sum_z p(z|d)p(w|z)$$

- An advantage of PLSA is that the latent variable z can be interpreted as a topic
  - z: topic (latent class)
  - $p(z|d)$: each document has a distribution over K latent topics
  - $p(w|z)$: each topic has a distribution over the vocabulary

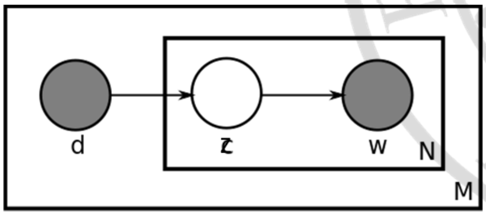
[1] Hofmann (1999). "Probabilistic Latent Semantic Indexing".

PLSA Model

- PLSA is a generative model of the documents in the collection it is estimated on

$$p(w,d) = p(d)\sum_z p(z|d)p(w|z)$$

  - For each document d, a topic z is generated conditionally to d according to $p(z|d)$
  - A word is then generated from topic z according to $P(w|z)$



EM Algorithm for Latent Variable Models

- Given a joint distribution $p(X,Z|\Theta)$ over observed variables X and latent variables Z, governed by parameters Θ, the goal is to maximize the likelihood function $p(X|\Theta)$ w.r.t. Θ.
- The general EM algorithm:
  - Initialize the parameters Θ^{old};
  - E-Step: Evaluate $p(Z|X,\Theta^{\text{old}})$;
  - M-Step: Evaluate Θ^{new} given by

$$\Theta^{\text{new}} = \underset{\Theta}{\operatorname{argmax}} \sum_Z p(Z|X,\Theta^{\text{old}})p(X,Z|\Theta)$$

  - Check the convergence of the parameter values; if not convergence condition not satisfied set $\Theta^{\text{old}} = \Theta^{\text{new}}$ and go to E-step.

Learning for PLSA

- The parameters $p(z|d)$ and $p(w|z)$ of PLSA can be learned by using the EM algorithm
- EM algorithm for PLSA:
  - E-Step: Evaluate $p(z_k|\, d_i, w_j; \Theta^{\text{old}})$;

$$p(z_k|d_i, w_j; \Theta^{\text{old}}) = \frac{p(w_j|z_k)p(z_k|d_i)}{\sum_{l=1}^{K} p(w_j|z_l)p(z_l|d_i)}$$

  - M-Step: Evaluate Θ^{new} given by

$$p(w_j|z_k) = \frac{\sum_{i=1}^{M} \#(d_i, w_j)p(z_k|d_i, w_j)}{\sum_{n=1}^{N}\sum_{m=1}^{M} \#(d_m, w_n)p(z_k|d_m, w_n)}$$

$$p(z_k|d_i) = \frac{\sum_{j=1}^{N} \#(d_i, w_j)p(z_k|d_i, w_j)}{\#(d_i)}$$

word2vec

- Latent semantic analysis (LSA)
  - Low-rank factorization of the co-occurrence matrix
  - Latent space can be interpreted as latent concepts
  - Words are distributional representations in the latent space
- word2vec
  - Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space
  - Predict surrounding words (skip-gram)
  - Also can be used represent similarity

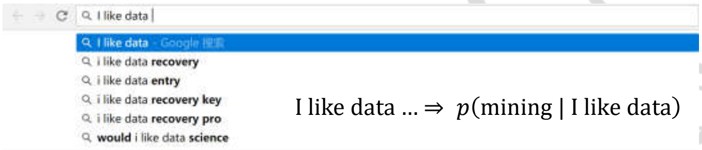
[1] Mikolov (2013). "Efficient Estimation of Word Representations in Vector Space".

Language Model

- A statistical language model is a probability distribution over sequences of words $w_1, \dots, w_N$
- Given such a sequence, it assigns a probability $p(w_1, \dots, w_N)$ to the whole sequence
  - Rank possible sentences (e.g., spelling correction)

    $p(\text{“I like data analytics”}) > p(\text{“I like Dota analytics”})$
    $p(\text{“I like data analytics”}) > p(\text{“Data analytics like I”})$

  - Generate possible sentences (e.g., autocomplete query)



I like data … ⇒ $p(\text{mining} \mid \text{I like data})$

n-gram Language Model

- The probability of a word only depends on the previous $n - 1$ words, known as an n-gram model

$$p(w_1, \dots, w_N) = \prod_{i=1}^{N} p(w_i | w_1, \dots, w_{i-1}) \approx \prod_{n=1}^{N} p(w_i | w_{i-(n-1)}, \dots, w_{i-1})$$

  - Bigram ($n = 2$) language model

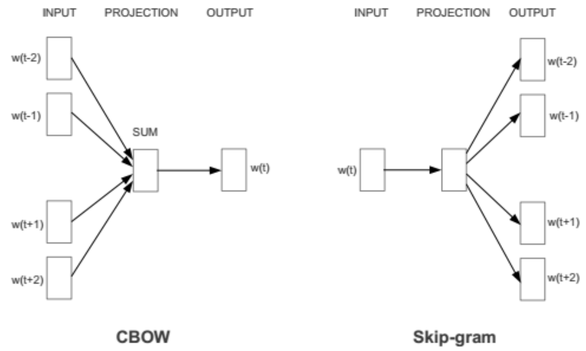
$$p(\text{“I like data analytics”})$$
$$\approx p(\text{I} \mid \langle s \rangle) p(\text{like} \mid \text{I}) p(\text{data} \mid \text{like}) p(\text{analytics} \mid \text{data}) p(\langle /s \rangle \mid \text{analytics})$$

- The conditional probability can be calculated from n-gram model frequency counts

$$p(w_i | w_{i-(n-1)}, \dots, w_{i-1}) = \frac{\#(w_{i-(n-1)}, \dots, w_{i-1}, w_i)}{\#(w_{i-(n-1)}, \dots, w_{i-1})}$$

CBOW and Skip-Grams

- word2vec can use either continuous bag-of-words (CBOW) or continuous skip-gram to produce a distributed representation of words



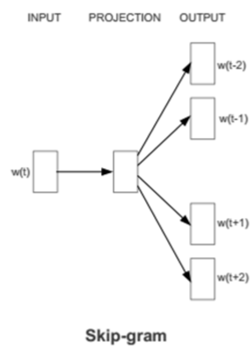
Objective of word2vec (Skip-gram)

- Maximize the log likelihood of the context words $w_{t-m}, w_{t-m+1}, \dots, w_{t-1}, w_{t+1}, w_{t+2}, \dots, w_{t+m}$, given w_t
  - m is usually 5~10

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m \le j \le m, j \ne 0} \log p(w_{t+j}|w_t)$$

- Use softmax to model $p(w_{t+j}|w_t)$

$$p(w_{t+j}|w_t) = \frac{\exp(v_{w_{t+j}} \cdot v_{w_t})}{\sum_{w' \in Context} \exp(v_{w'} \cdot v_{w_t})}$$



Optimization of word2vec

- How to minimize the objective of word2vec to obtain v_{w_t} for $w_1, \dots, w_T$? – Gradient descent
  - Let the current center word be c and one of its context word be s, then the conditional probability becomes

$$p(s|c) = \frac{\exp(v_s \cdot v_c)}{\sum_{w'} \exp(v_{w'} \cdot v_c)}$$

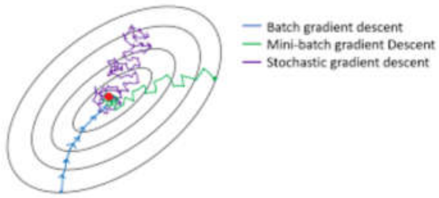
  - The gradient of the log likelihood w.r.t. v_c is

$$\frac{\partial \log p(s|c)}{\partial v_c} = v_s - \sum_{w'} \frac{\exp(v_{w'} \cdot v_c)}{\sum_{w''} \exp(v_{w''} \cdot v_c)} v_{w'} = v_s - E_{w' \sim p(w'|c)} v_{w'}$$

  - Alternate minimize $J(\theta)$ w.r.t. v_{w_t} for $w_1, \dots, w_T$

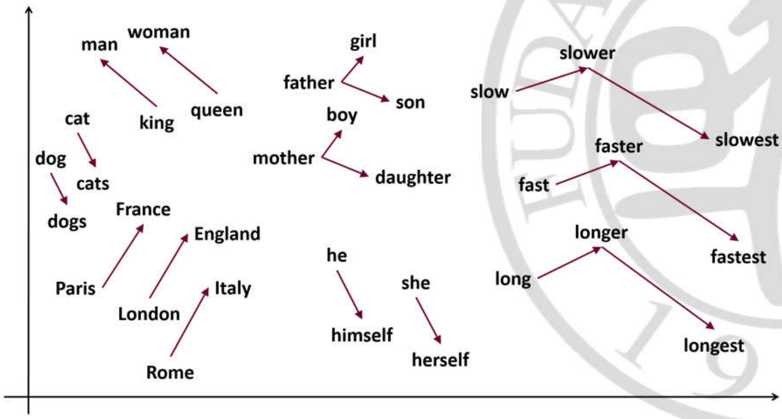
Optimization of word2vec

- Gradient descent
  - Let $J(\theta) = \frac{1}{n}\sum_{i=1}^{n} J_i(\theta)$
  - update rule: $\theta \leftarrow \theta - \frac{\eta}{n}\sum_{i=1}^{n} \nabla J_i(\theta)$
- Stochastic gradient descent
  - Replace $\frac{1}{n}\sum_{i=1}^{n} \nabla J_i(\theta)$ by the gradient at a single example $\nabla J_i(\theta)$
  - At each iteration randomly select an example i and update: $\theta \leftarrow \theta - \eta \nabla J_i(\theta)$



Word Embedding with word2vec

■



Summary

- Vector space models (VSMs) represent words in a continuous vector space where semantically similar words are located in close proximity to one another
- All methods depend on the distributional hypothesis, which states that words that appear in the same contexts share semantic meaning
- There are two main categories: count-based methods (e.g. Latent Semantic Analysis), and predictive methods (e.g. word2vec/Skip-gram).

Thanks

Email: libin@fudan.edu.cn