Learning With Text Data

Naresh Manwani
IIIT Hyderabad

Latent Semantic Indexing

Motivation for LSI

A. To find and fit a useful model of the relationships between terms and documents.
B. To find out what terms "really" are implied by a query .
C. LSI allow the user to search for concepts rather than specific words.
D. LSI can retrieve documents related to a user's query even when the query and the documents do not share any common terms.

Word Representation

One hot representation :

- Most NLP tasks regard words as atomic symbol, e.g., book, key, stair etc.
- In vector space, this is a sparse vector with one 1 and a lot of zeros
  [0 0 0 0 0 0 1 0 0 0 0 0 . . . . . 0 0 0 0 0]

Problems :

- Dimensionality of the vector will be same as the size of the vocabulary
- Vectors will be very high dimensional
- E.g., 13M for Google 1T, 500k for big data

Term frequency

- Terms tell us about documents
  - If "rabbit" appears a lot, it may be about rabbits
- Documents tell us about terms
  - "the" is in every document -- not discriminating
- Documents are most likely described well by rare terms that occur in them frequently
  - Higher "term frequency" is stronger evidence
  - Low "collection frequency" makes it stronger still

LSI Procedure

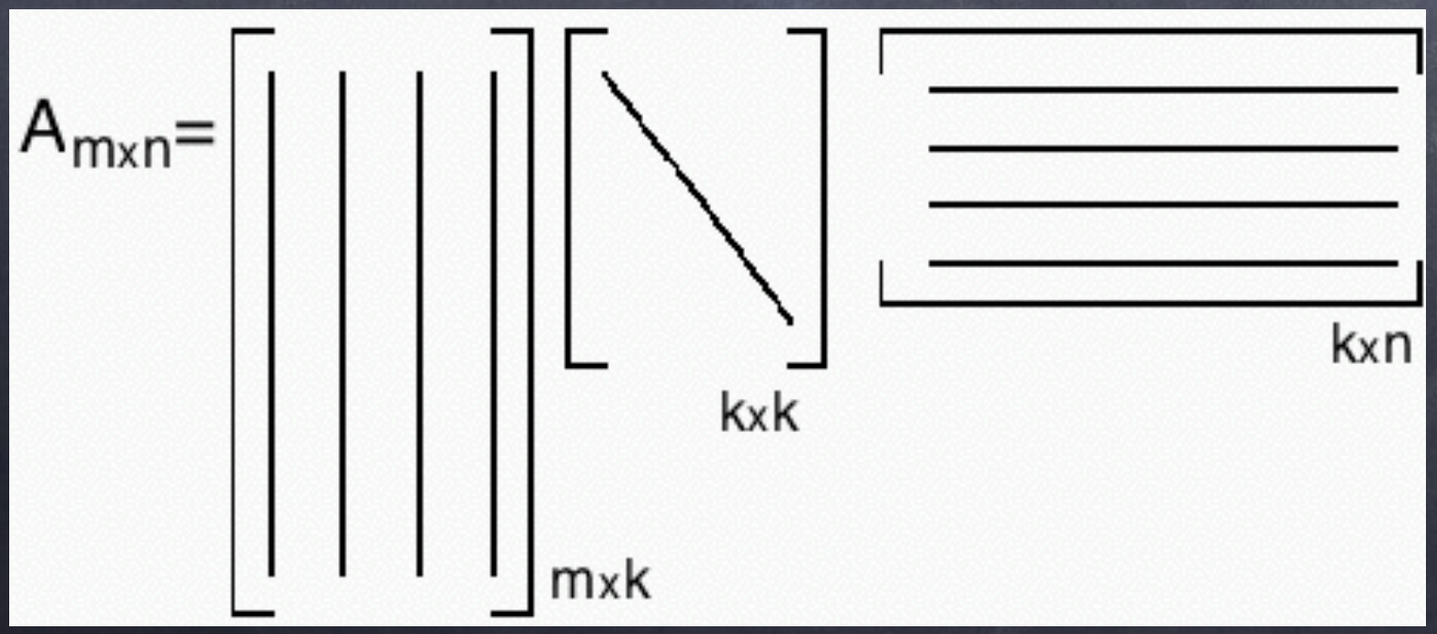
- Obtain term-document matrix.
- Compute the SVD.
- Truncate-SVD into reduced-k LSI space.
  - k-dimensional semantic structure
  - similarity on reduced-space
    - term-term
    - term-document
    - document-document

Latent Semantic Analysis (LSA)

- Singular Value Decomposition on TF-IDF matrix

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

- Columns of U are eigenvectors of AA'
- Columns of V are eigenvectors of A'A
- Σ is diagonal matrix which contains the eigenvalues of AA' or A'A



tf-idf Matrix

A

M	D_1	D_2	D_3	D_4	D_5	D_6	⋯	D_n
T_1	0.00060	0.00012	0.00003	0.00003	0.00333	0.00048	⋯	a_{1n}
T_2	0	0	0	0	0	0	⋯	a_{2n}
T_3	0	2.98862	0	0	0	1.49431	⋯	a_{3n}
T_4	0	0	0	13.32555	0	0	⋯	a_{4n}
T_5	0	0	0	0	0	0	⋯	a_{5n}
T_6	1.03442	1.03442	0	0	0	3.10326	⋯	a_{6n}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
T_m	a_{m1}	a_{m2}	a_{m3}	a_{m4}	a_{m5}	a_{m6}	⋯	a_{mn}

B

$$U = U_k = \begin{array}{c|cccc} & C_1 & C_2 & C_3 & \cdots & C_m \\ T_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ T_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2m} \\ T_3 & a_{31} & a_{32} & a_{33} & \cdots & a_{3m} \\ T_4 & a_{41} & a_{42} & a_{43} & \cdots & a_{4m} \\ T_5 & a_{51} & a_{52} & a_{53} & \cdots & a_{5m} \\ T_6 & a_{61} & a_{62} & a_{63} & \cdots & a_{6m} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ T_m & a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mm} \end{array}$$

$$\Sigma = \Sigma_k = \begin{array}{c|ccccc} & D_1 & D_2 & D_3 & \cdots & D_n \\ T_1 & a_{11} & 0 & 0 & \cdots & 0 \\ T_2 & 0 & a_{22} & 0 & \cdots & 0 \\ T_3 & 0 & 0 & a_{33} & \cdots & 0 \\ T_4 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ T_m & 0 & 0 & 0 & \cdots & a_{mm} \end{array}$$

$$V^T = V_k^T = \begin{array}{c|ccccc} & D_1 & D_2 & D_3 & \cdots & D_n \\ C_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ C_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ C_3 & a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ C_4 & a_{41} & a_{42} & a_{43} & \cdots & a_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_n & a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{array}$$

Query Processing

- Map the query to reduced k-space

$$q' = q^T U_k S_k^{-1}$$

- Retrieve documents or terms within a proximity.
  - -cosine
  - -best m

Document Processing

Folding-in

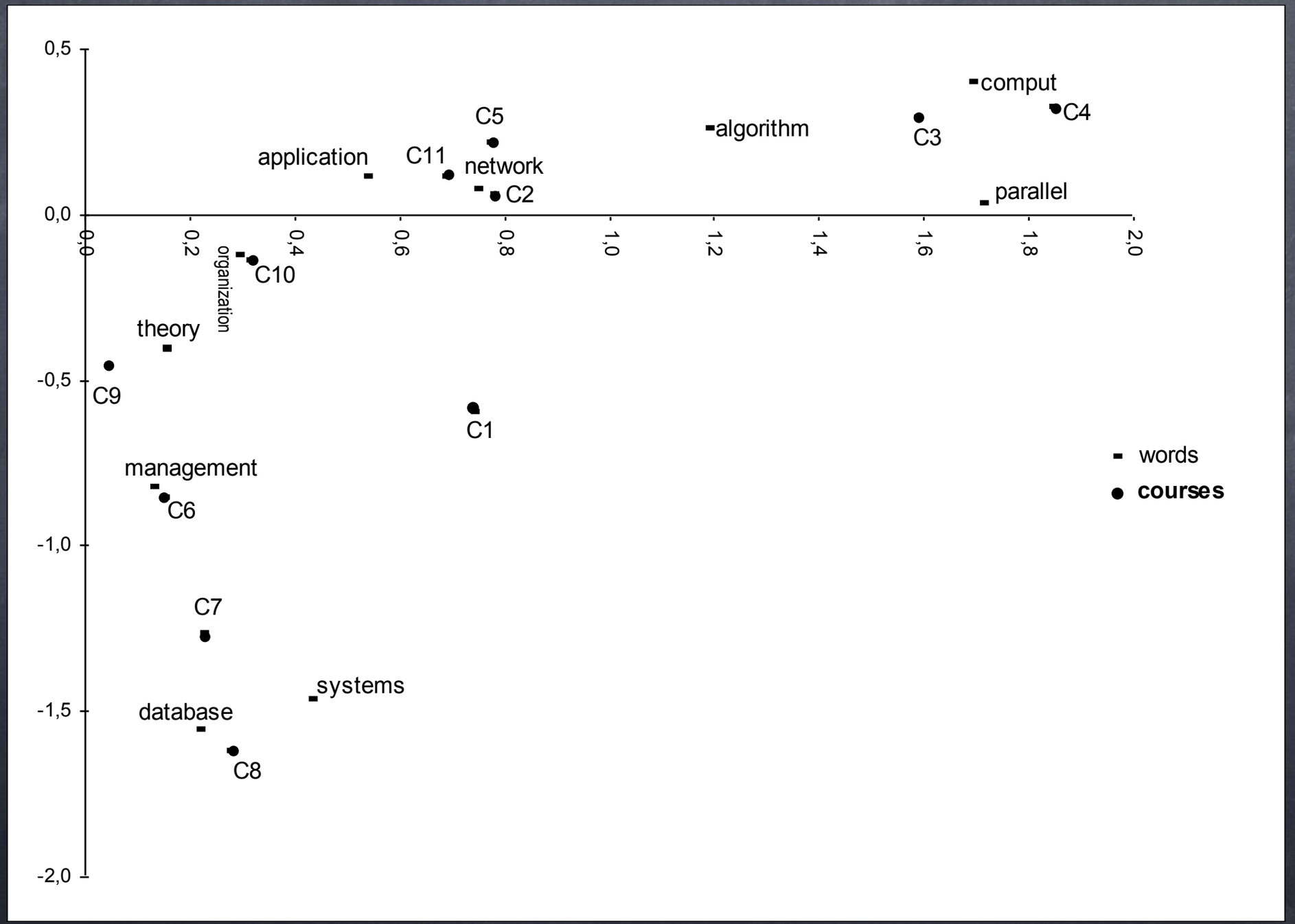
A. $d' = d^T U_k S_k^{-1}$

similar to query projection

Example : Collection

Label	Course Title
C1	Parallel Programming Languages Systems
C2	Parallel Processing for Noncommercial Applications
C3	Algorithm Design for Parallel Computers
C4	Networks and Algorithms for Parallel Computation
C5	Application of Computer Graphics
C6	Database Theory
C7	Distributed Database Systems
C8	Topics in Database Management Systems
C9	Data Organization and Management
C10	Network Theory
C11	Computer Organisation

Mapping


0,5
0,0
-0,5
-1,0
-1,5
-2,0
0,0
0,2
0,4
0,6
0,8
1,0
1,2
1,4
1,6
1,8
2,0
comput
C4
C5
algorithm
C3
application
C11
network
C2
parallel
organization
C10
theory
C9
C1
management
C6
C7
systems
database
C8
words
courses

Example Query and New Terms

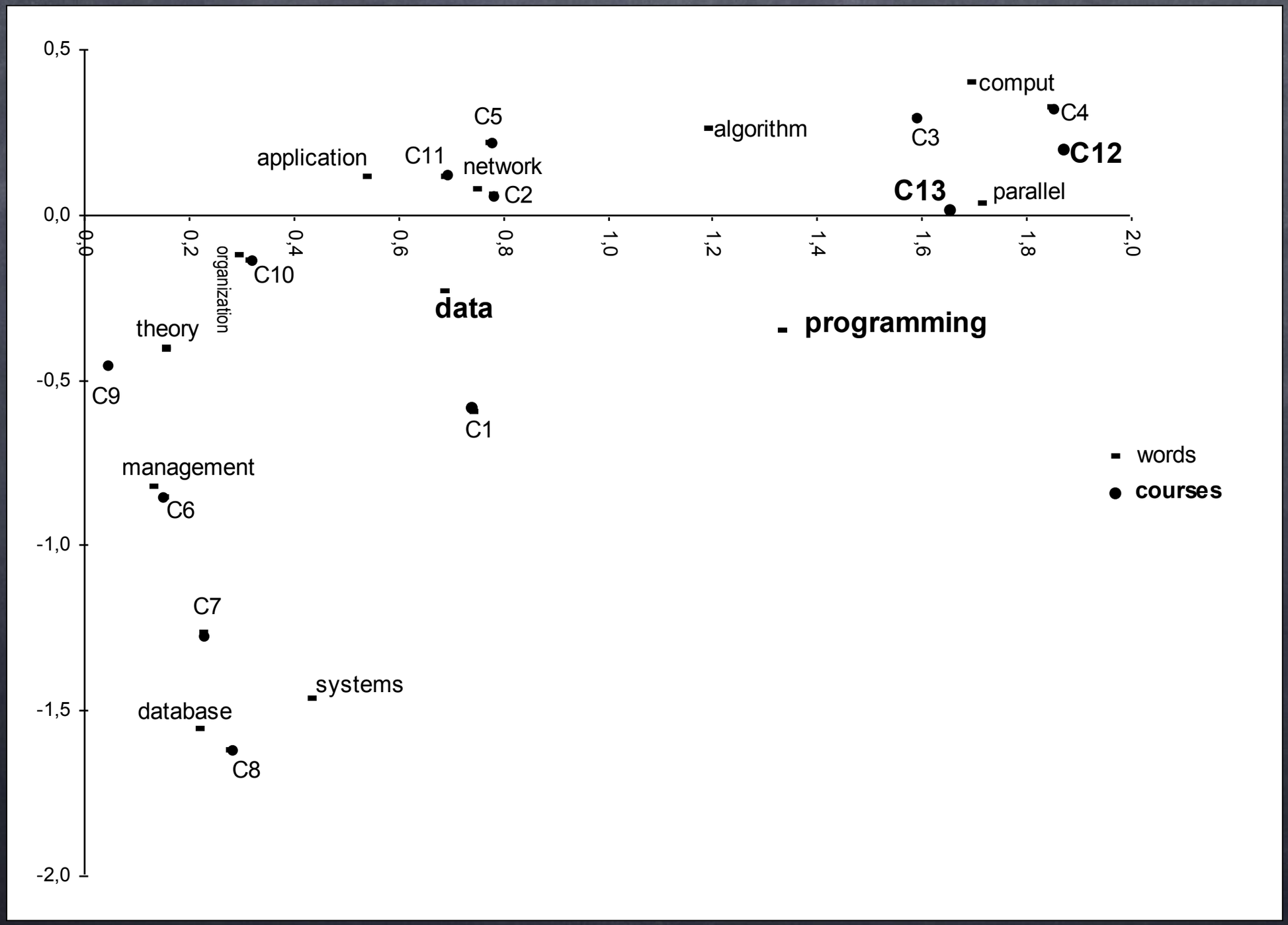
A. Query: computer database organizations

q^T = [0 1 0 0 0 0 1 0 0 1].

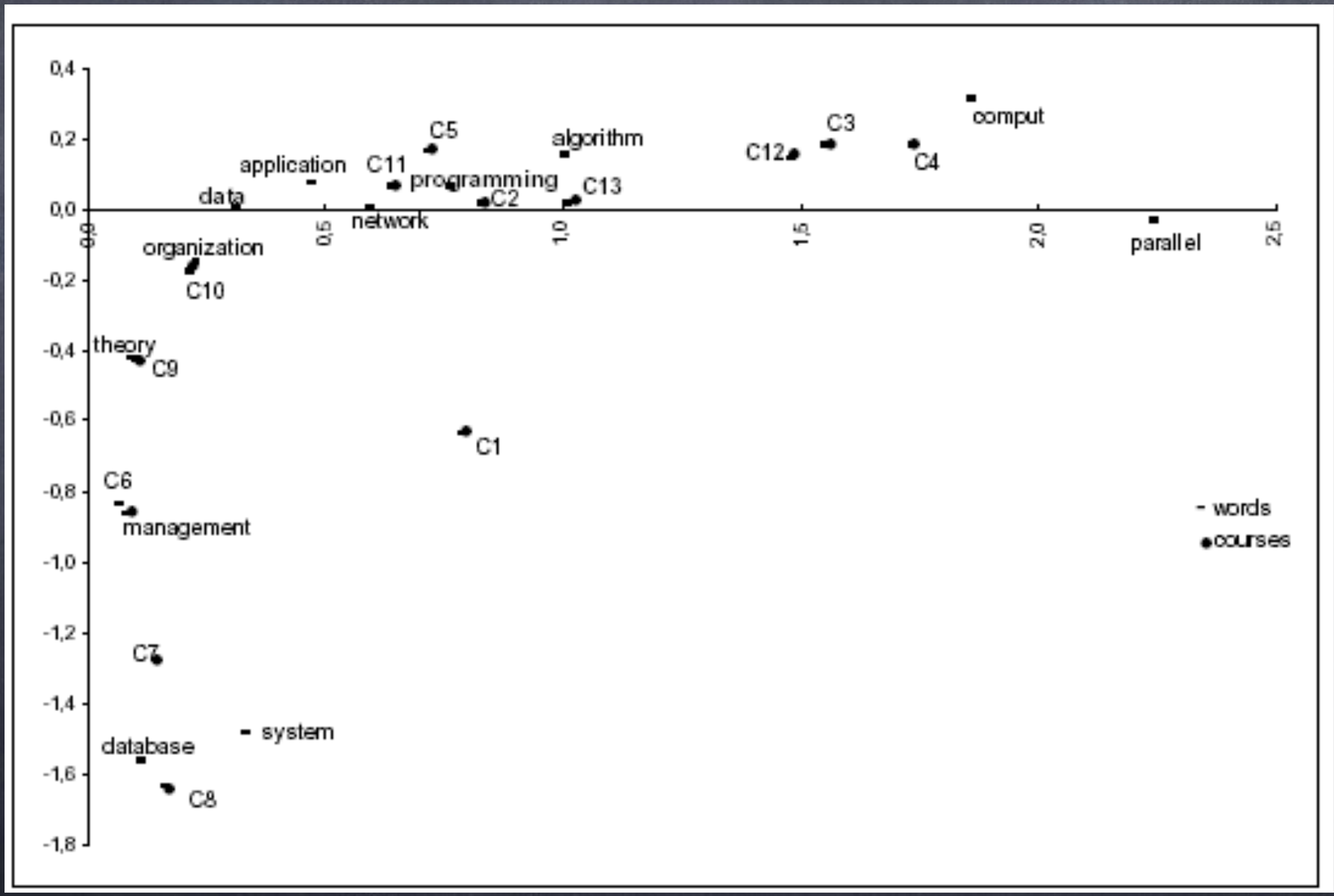
B. Update:

Label	Course Title
C12	Parallel Programming for Scientific Computations
C13	Data Structures for Parallel Programming

Query


0,5
0,0
-0,5
-1,0
-1,5
-2,0
0,0
0,2
0,4
0,6
0,8
1,0
1,2
1,4
1,6
1,8
2,0
comput
C4
algorithm
C3
C5
application
C11
network
C12
C13
parallel
C2
organization
C10
data
programming
theory
C9
C1
management
C6
C7
systems
database
C8
words
courses

Recomputed Space



LSI Similarity Versus Lexical Similarity

	LSI		Lexical	
Label	**Sim.**	**Retrvl.**	**Sim.**	**Retrvl.**
C1	0.3238	R	0	
C2	0.2213		0	
C3	0.2119		0.4082	R
C4	0.2150		0.3780	R
C5	0.1751		0.4472	R
C6	0.2412	R	0.4472	R
C7	0.2567	R	0.4472	R
C8	0.2586	R	0.4082	R
C9	0.2356		0.4472	R
C10	0.2301		0	
C11	0.1973		0.8944	R

Word2Vec

Issues with SVD

- Does not capture the linear relationships among words well
- Computational cost in SVD scales quadratically for $m \times n$ matrix: $O(nm^2)$ (when $m < n$)
- This is not possible for large number of words or document.

Idea : Learning Low Dimensional Vectors Directly

- Learning representations by back-propagating errors. (Rumelhart, Hinton, Williams 1986)
- A Neural Probabilistic Language Model (Bengio et al., 2003)
- Natural Language Processing (Almost) from Scratch (Collobert et al., 2011)
- Efficient Estimation of Word Representations in Vector Space (Mikolov et al., 2013)
- GloVe: Global Vectors for Word Representation (Pennington et al., 2014)

Distribution similarity based representations

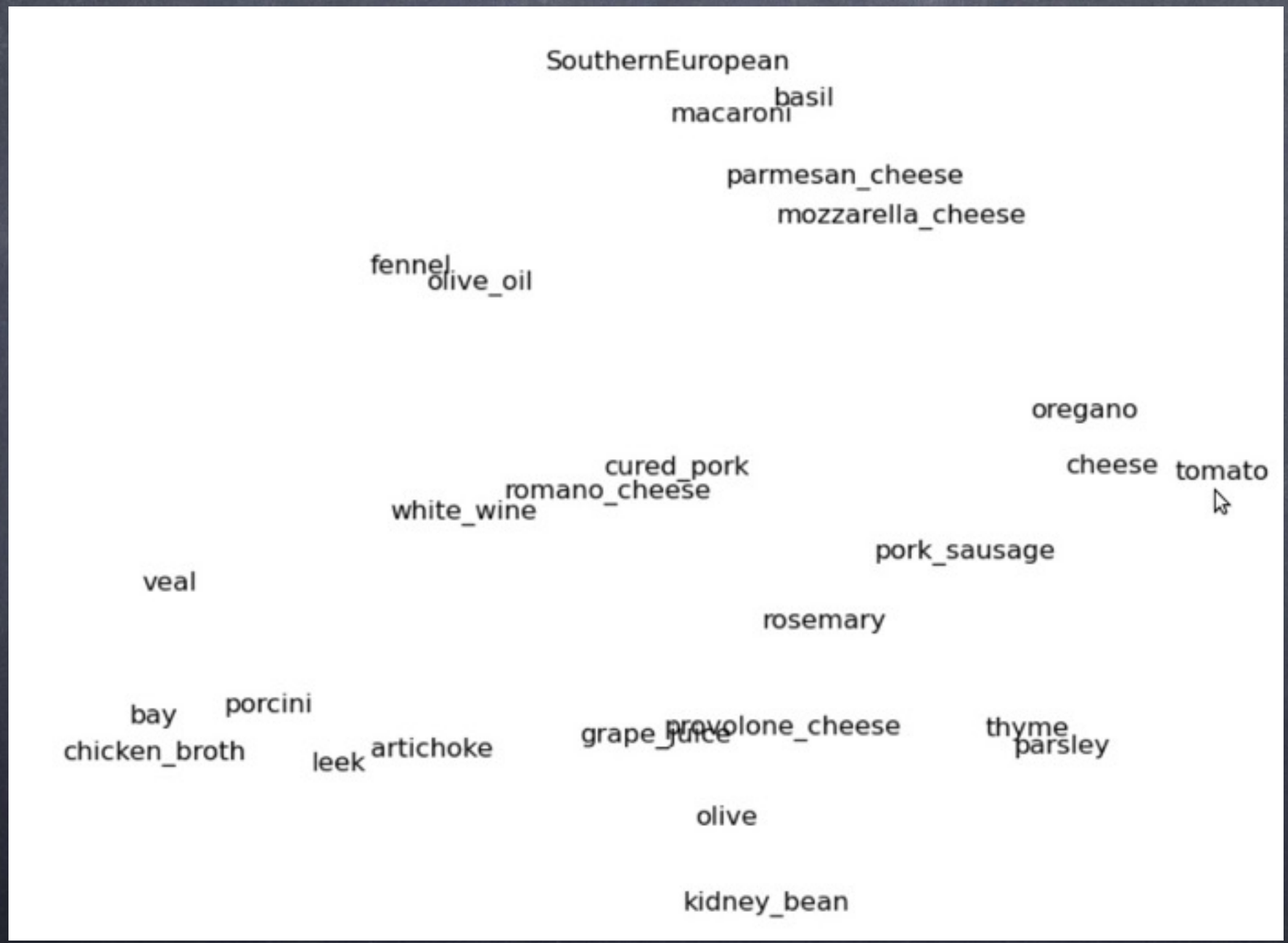
> "You shall know a word by the company it keeps". -Firth, J.R. (1957)

- Representing a word by means of its neighbours adds a lot of value
- Leads to one of the most successful ideas of modern statistical NLP

government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge

These words will represent banking

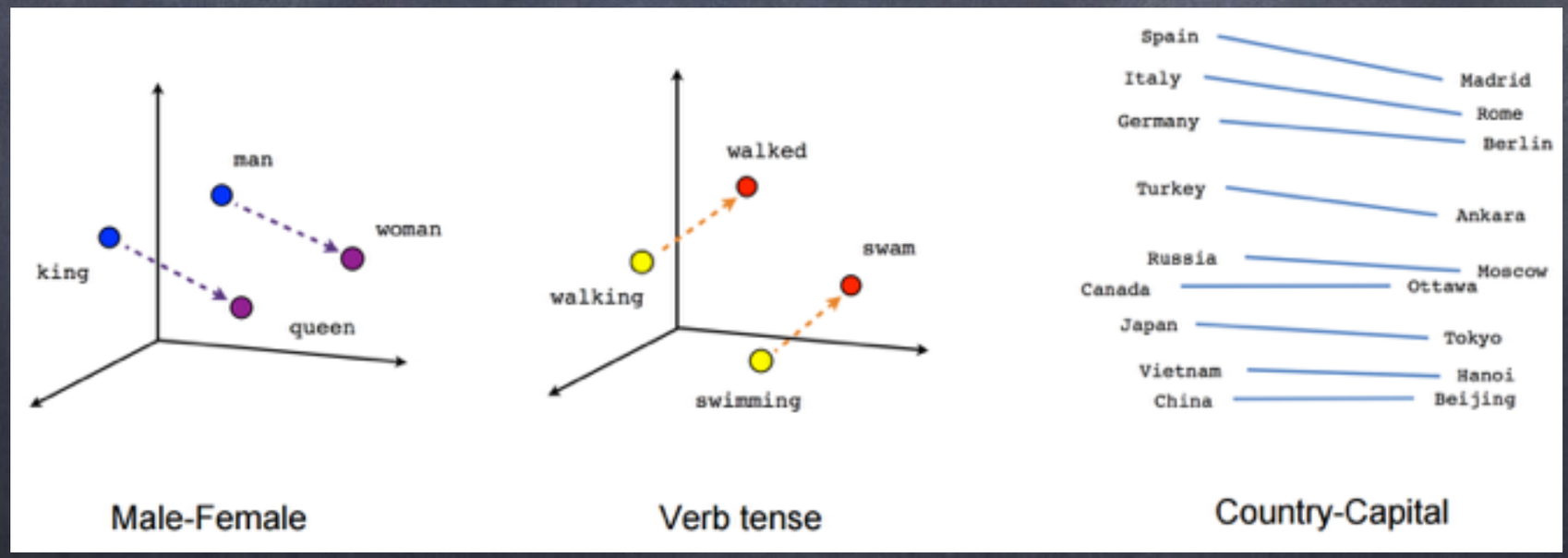
Neural Word Embeddings Visualization



Vectors for Word Representation

- word2vec learns relationships between words.
- The output are vectors with interesting relationships.
- Example:

vec(king)-vec(man) + vec(woman) ≈ vec(queen)



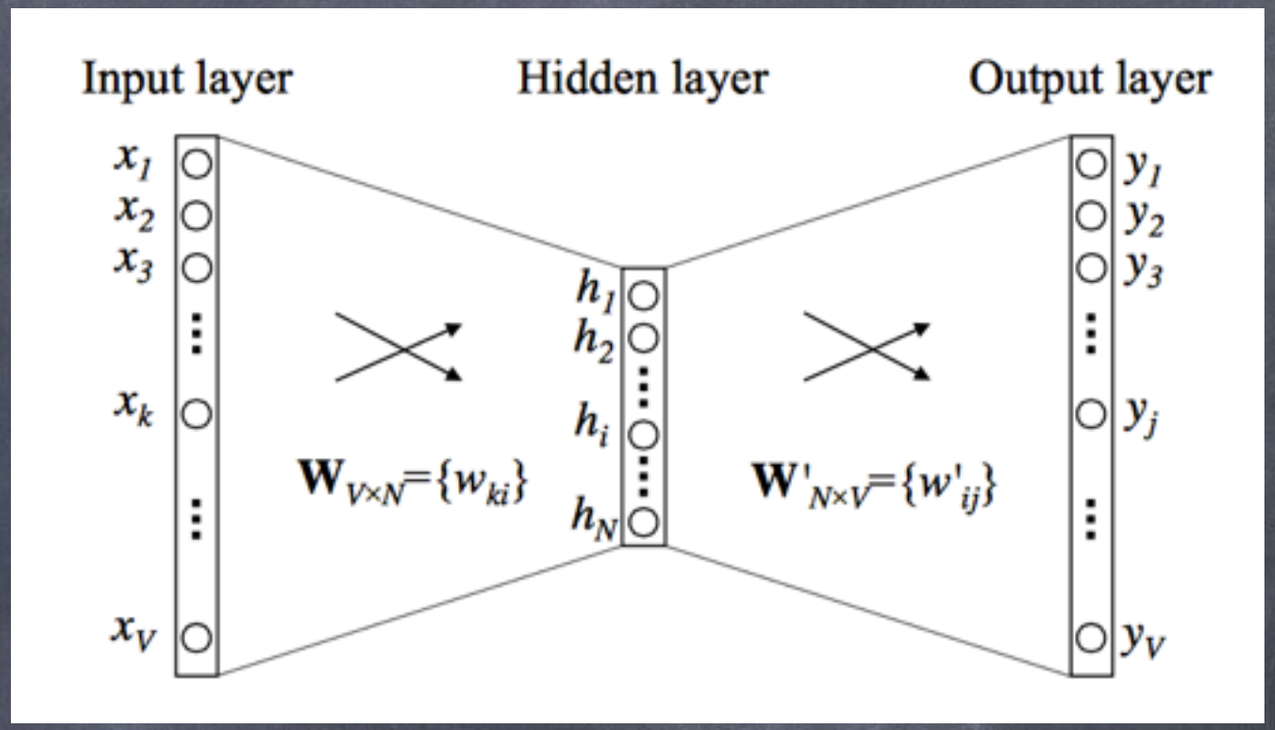
Word2Vec

A. Associates words to points in the space
B. Captures the word meaning and associations between similar words spatially
C. Trained as a very simple neural network
    A. single hidden layer with no non-linearities
    B. Supervised approach

Reference : Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, Distributed Representations of Words and Phrases and their Compositionality, NIPS 2013.

Simplest Continuous Bag of Words Model (CBOW)

- Only one word in the context and one word in the target.
- This is like a bigram model.



$$p(w_O|w_I) = \frac{\exp(v'^T_{w_O} v_{w_I})}{\sum_{w\in V} \exp(v'^T_w v_{w_I})}$$

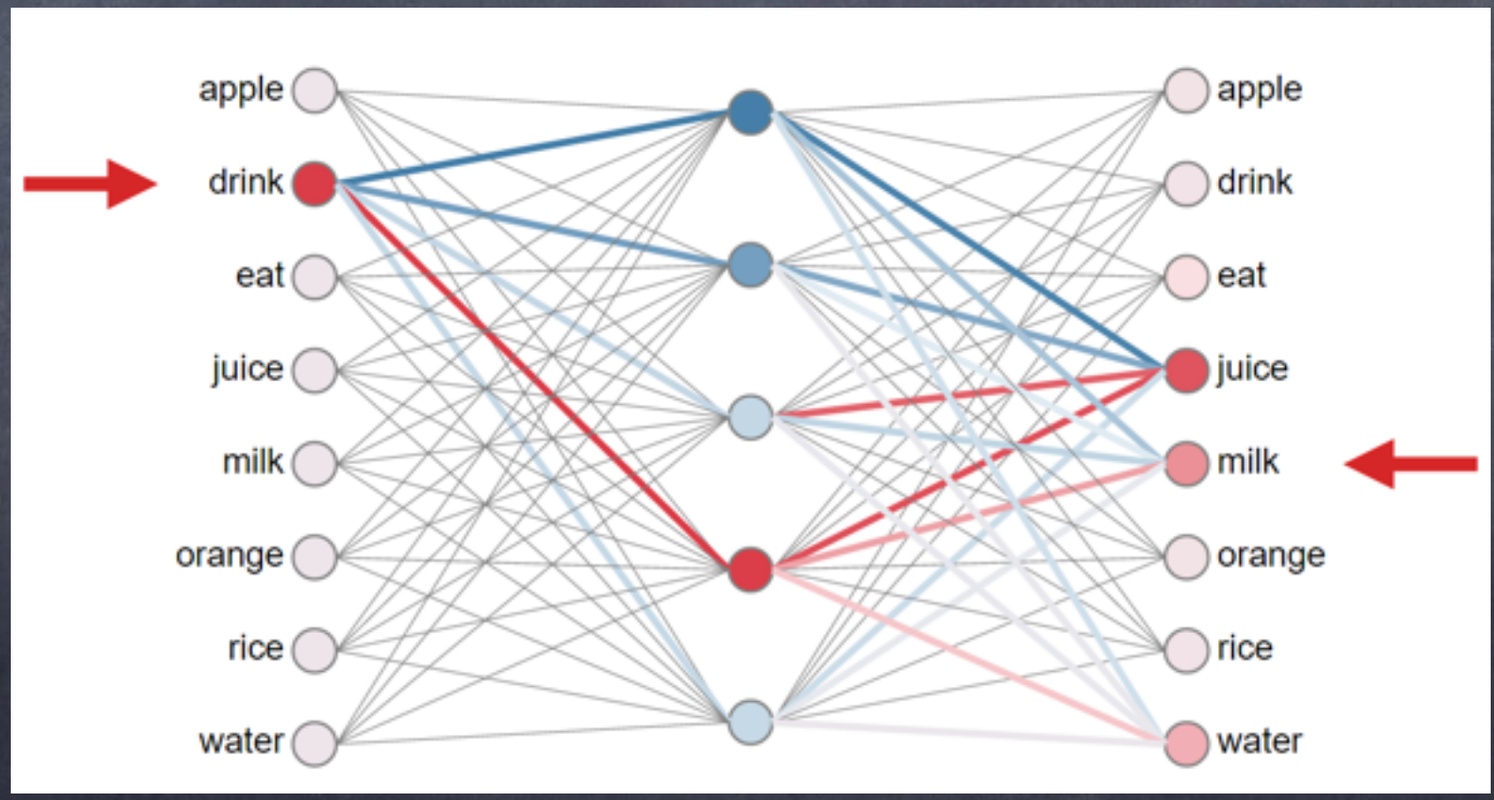
Reference : Xin Rong, word2vec Parameter Learning Explained, https://arxiv.org/abs/1411.2738

Example : Single Word Context

Training Data

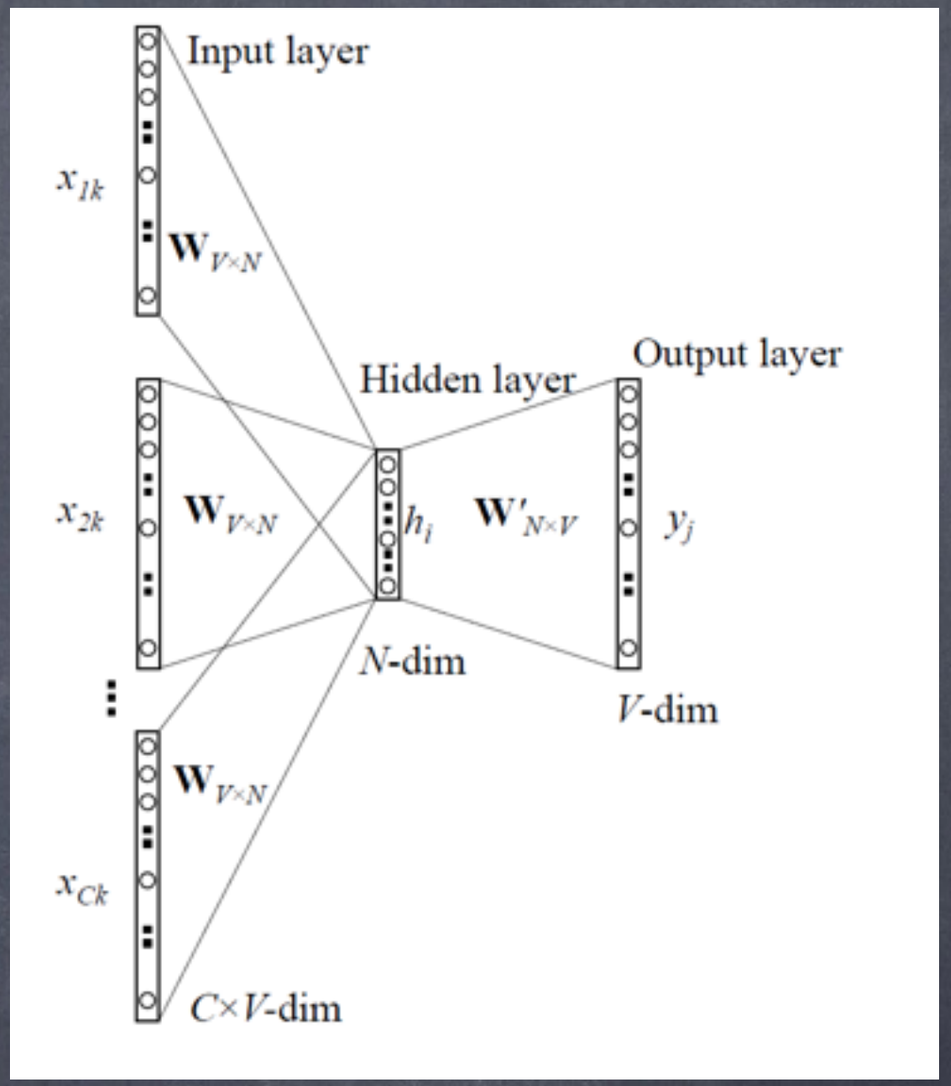
1. eat | apple
2. eat | orange
3. eat | rice
4. drink | juice
5. drink | milk
6. drink | water
7. orange | juice
8. apple | juice
9. rice | milk
10. milk | drink
11. water | drink
12. juice | drink

Trained Model



Multiword Context CBOW

- Predicting a center word form the surrounding context
- Example :
  - context = {"The", "cat", 'over", "the', "puddle"}
  - target = "jumped"
- Multi class classification on the vocabulary V
- CBOW Model takes the average of the vectors of the context words

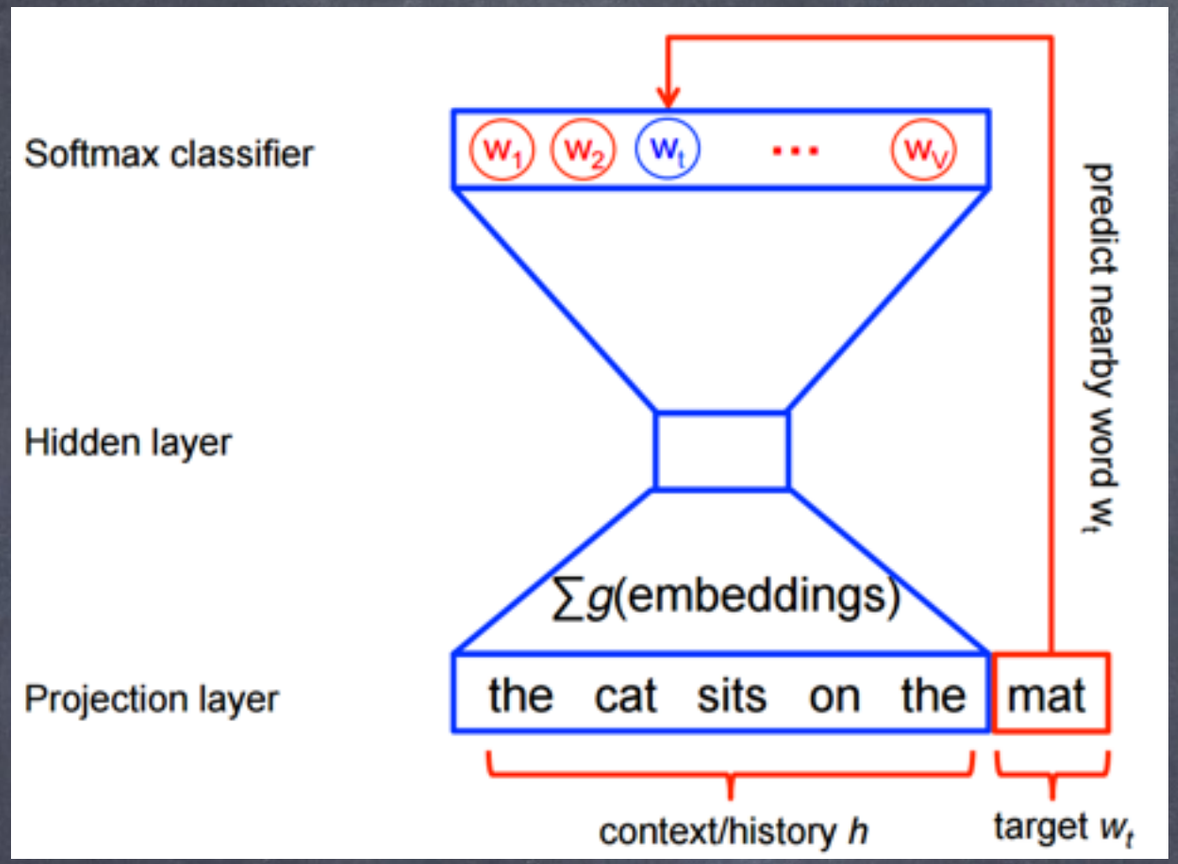


$$h = \frac{1}{C} W^T (x_1 + x_2 + \ldots + x_C)$$
$$= \frac{1}{C} (v_{w_1} + v_{w_2} + \ldots + v_{w_C})^T$$

- the loss function is

$$E = -log\ p(w_O | w_{I,1}, \ldots, w_{I,C})$$
$$= -log\ \frac{\exp(v'^T_{w_O} h)}{\sum_{w \in V} \exp(v'^T_w h)}$$

Training CBOW



Objective function : Softmax

$$E = -log\ p(w_O|w_{I,1}, \ldots, w_{I,C})$$
$$= -log\ \frac{\exp(v'^T_{w_O} h)}{\sum_{w \in V} \exp(v'^T_w h)}$$

- Need to find matrix W (corresponding to input-hidden layer) and W' (corresponding to hidden-output layer)
- Use stochastic gradient descent
- For each (current word, context) pair update the parameters once
- Which means in every iteration update close to 100K (|W|) parameters
- Too much computation !!!

Alternative Models with Fewer Parameter Updates

Two alternatives to replace softmax

- "Hierarchical Softmax" (Frederic Morin, Yoshua Bengio, Hierarchical probabilistic neural language model, AISTATS, 2005)
- "Negative sampling", an adaptation of "Noise contrastive estimation" (Michael Gutmann, Aapo Hyvarinen, Noise-contrastive estimation: A new estimation principle for unnormalized statistical models, AISTATS, 2010.)

Properties :

1. Negative sampling scales better in vocabulary size
2. Quality of word vectors is comparable
3. Significantly fewer parameter updates in the second layer

Word Vectors

Hidden Layer
Weight Matrix

Word Vector
Lookup Table!