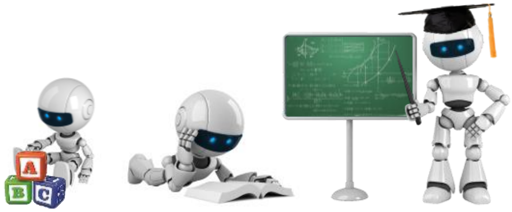
CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department

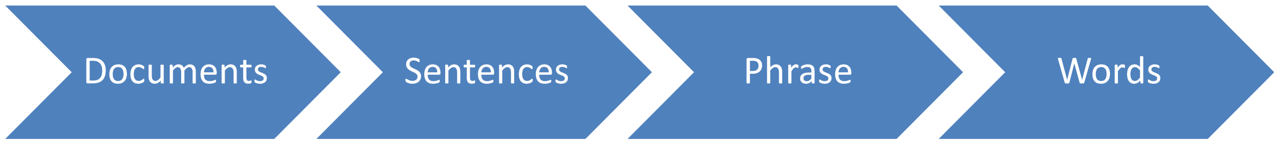


Logistical Notes

- Project Group Formation Deadline: 09/12 (next week!)
  - Do NOT form more than 50 groups
  - Meet weekly, helpful for both HW and project
- Paper Selection Deadline: 09/19 (in two weeks)
  - Check and then enter your paper: https://docs.google.com/spreadsheets/d/1_vafG77ijmETCnuVZvKpT35k--5op5wn71GZXgAY7O0
- Project Proposal Deadline: 10/03
- HW2:
  - No libraries are allowed, except for common libraries such as Pandas or NumPy

Natural Language Representation

- Language processing hierarchy levels:



- Sparsity in the NLP training datasets: natural language has a very huge space
  - Example: Average Wikipedia page size is 580 words and English has ~1M word roots, yet the actual number of possibilities is far more.
- We need **interpretable** representations or **embeddings** to represent natural language data for model training

- One-hot representation: too large (15M words) and meaningless

  Hotel: [0,0,0,0,1,0,0,0,0,0,…,0,0,0]

  Motel:[0,0,0,0,0,0,0,0,1,0,…,0,0,0]

Similarity of Vectors

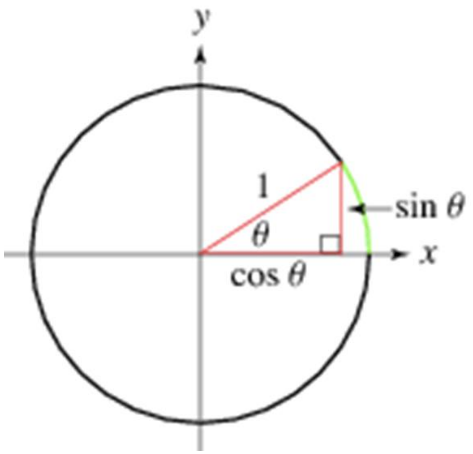
- Euclidean distance, i.e., geometric closeness :

- Curse of dimensionality

- Dot product:

$a \bullet b = ||a||\,||b||\cos(\theta_{ab})$

$= a_1b_1+a_2b_2+ \ldots +a_nb_n$



- Cosine similarity (scale invariant)

$\cos \theta_{ab} = a \bullet b \;/\; ||a||\,||b||$ -> $1- \cos \theta_{ab}$ is a metric

- Invariant with respect to the vector starting point

- EX: Hotel: [0,0,0,0,1,0,0,0,0,0,0,…,0,0,0], Motel:[0,0,0,0,0,0,0,0,0,0,1,0,…,0,0,0]

Hotel'*Motel = 0

The Distributional Hypothesis

Zellig Harris, 1954

- Words that occur in the **same contexts** tend to have similar meanings
- Example: nice, good

Budanitsky and Hirst, 2006

- Word relatedness association: related words **co-occur** in different contexts
- Example: cup, coffee

- If semantic similarity and association of words can be encoded into their representations, we may be able to address the challenge of sparsity

- In the absence of a particular word during training, we can rely on its synonyms that exist in the training dataset: Motel vs Hotel
- We can draw conclusions:

Lecturers teach in the university-> Professors ___ in the university.

Vector Embedding of Words

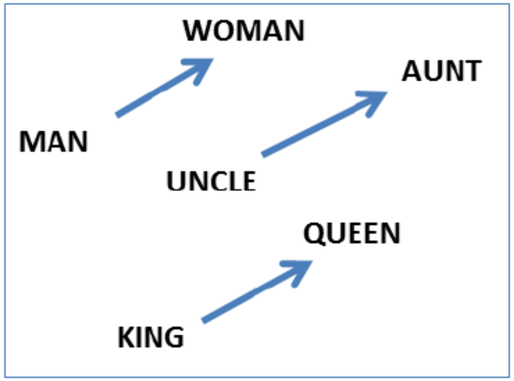
- Represent words using dense vectors:
  - Latent Semantic Analysis/Indexing (SC Deerwester et al, 1988)
  - Word2Vec (Mikolov et al, 2013)
  - GloVe (Pennington et al, 2014)

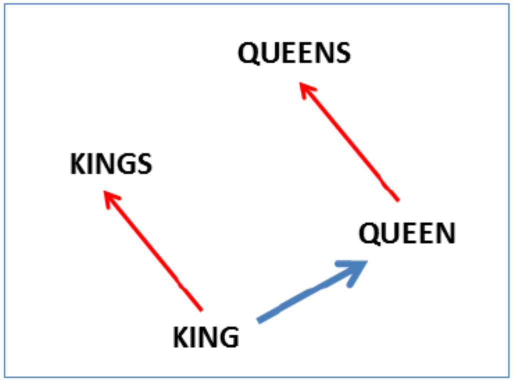
LSA	Word2Vec	GloVe
• Term weighting-based model • Consider occurrences of terms at **document level**	• Prediction-based model • Consider occurrences of terms at **context level**	• Count-based model • Consider occurrences of terms at **context level**

Word Embedding

- Each word is represented by a vector:
- The same size is used for all words
- Relatively low dimensional (~300)
- Vectors for similar words are similar (measured in dot product)
- Vector operations can be used for semantic and syntactic deductions, e.g.,

Queen – Woman + Man = King





- The key idea is to derive the embeddings from the distributions of word context as they appear in a large corpus.

Singular Value Decomposition

- Every matrix $A \in \mathbb{R}^{m \times n}$ can be factorized as $A = U\Sigma V^T$ where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix
  - The diagonal entries of A are called the singular values of the matrix A

- Singular value $\sigma_i = \sqrt{\lambda_i}$

$$U = AV\Sigma^{-1}$$
$$U\Sigma = AV$$
$$U\Sigma V^T = A$$

Matrix Factorization

- We can form a matrix of M using the idea of Bag of Words: the word representations are highly sparse

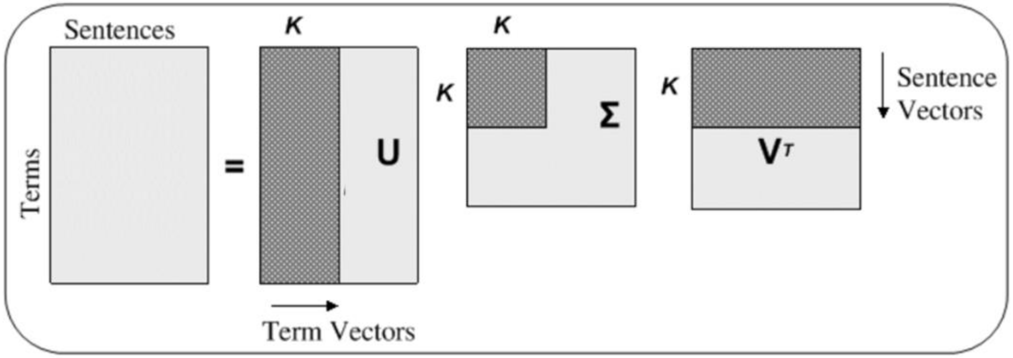
Words

Contexts

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

- Singular value decomposition (U, V are orthonormal)

$$M_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$$

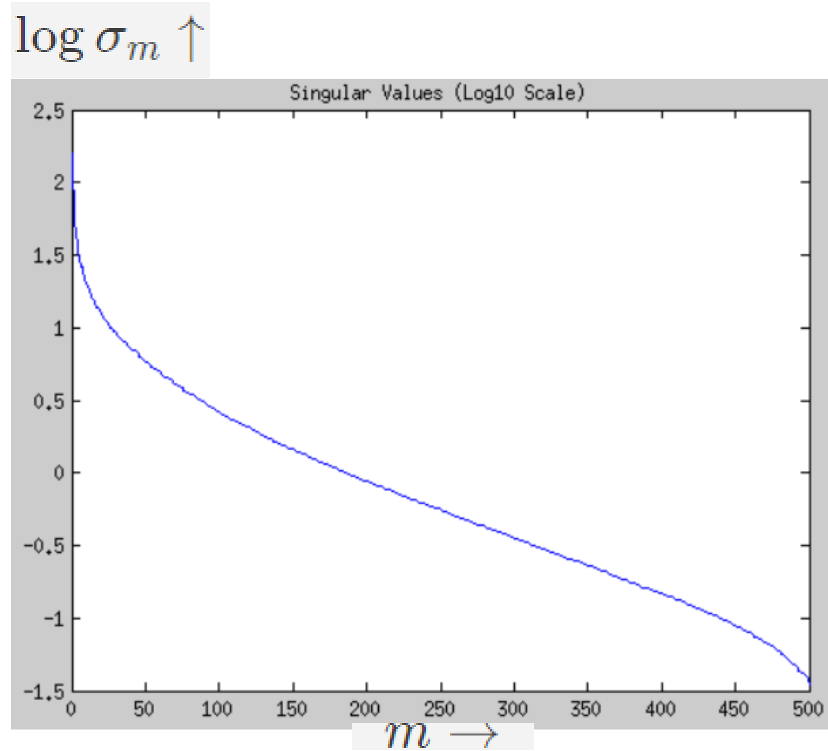


Decrease in σ_m

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 & \dots & 0 \\ & & \dots & & & & \\ 0 & 0 & 0 & \dots & \sigma_m & \dots & 0 \end{bmatrix}$$

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots \geq \sigma_m$$

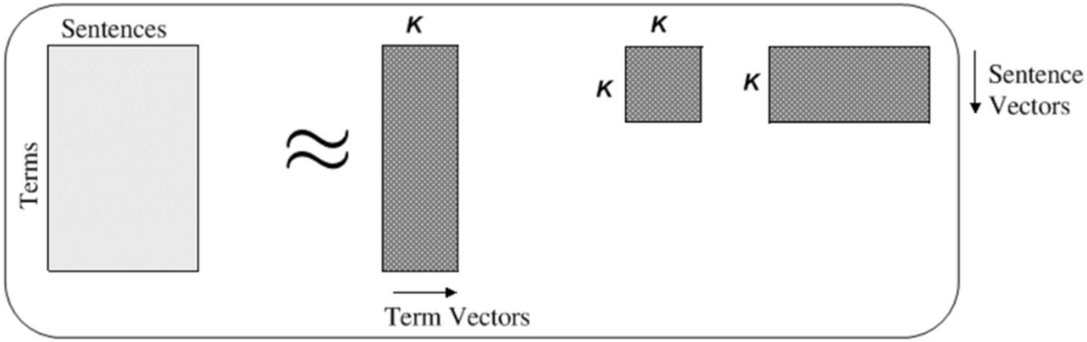


- What can we do with this information?
  - We can set any sigma after **k** to be equal to 0
  - Therefore we effectively have a square diagonal matrix of shape **K** x **K** for the new sigma matrix.
  - Because of the 0s in sigma matrix, we can now ignore chunks of U and V matrix as they will result in 0s when we matrix multiply them.

Matrix Factorization

- Many singular values are going to be zero or negligible

$$M_{m\times n} \approx U'_{m\times k}\Sigma'_{k\times k}V'^{T}_{k\times n}$$



$$m_l \approx u_l \Sigma' V'^{T}$$
$$u_l \approx m_l V' \Sigma'^{-1}$$

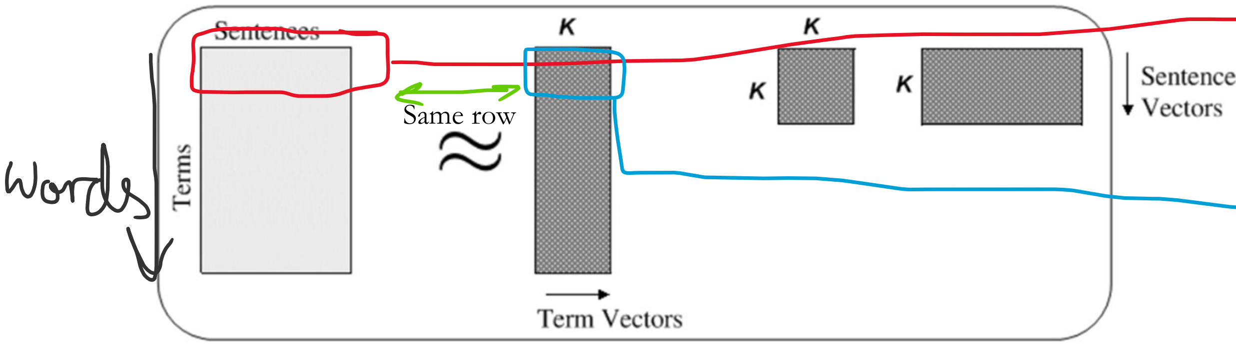
- We can use rows of U as word embeddings

- An old idea for dimensionality reduction (it is possible to use other matrix factorization methods, e.g., non-negative matrix factorization)
- Determining context is heuristic
- computational expensive with O(mn^2) cost for an n*m matrix
- Hard to incorporate new words

Matrix Factorization

- Many singular values are going to be zero or negligible

$$M_{m\times n} \approx U'_{m\times k}\Sigma'_{k\times k}V'^{T}_{k\times n}$$



A row of the M matrix

$$m_l \approx u_l \Sigma' V'^{T}$$

$$u_l \approx m_l V' \Sigma'^{-1}$$

Word embedding for word on the l^{th} row

- We can use rows of U as word embeddings

- An old idea for dimensionality reduction (it is possible to use other matrix factorization methods, e.g., non-negative matrix factorization)
- Determining context is heuristic
- computational expensive with O(mn^2) cost for an n*m matrix (SVD)
- Hard to incorporate new words

Word2Vec

Core idea: find embeddings using a prediction task involving **neighboring words** in a huge real-world corpus.

Concept	Example
Input data: sets of **successive word-patterns** from meaningful sentences in the corpus	"One of the most important"
We build a **synthetic** prediction task using these patterns	Given an input predict the dash word Input: [One, of, ____, most, important] Target: the
We train a model to solve this prediction task	Loss(Model(["One", "of", ____, "most", "important"]), "the")
Embeddings will be the **byproduct** of this task	

Word2Vec

Concept

Input data: sets of **successive word-patterns** from meaningful sentences in the corpus

We build a **synthetic** prediction task using these patterns

We train a model to solve this prediction task

Embeddings will be the **byproduct** of this task

Specifics

consider a window with the center word w_t and "context words" $w_{t'}$ with a window fixed size, e.g., ($t'=t-5, \ldots t-1, t+1, \ldots , t+5$)

predict all w_t given $w_{t'}$ such that $p(w_t|w_{t'})$ is maximized

A Two Layer Neural Network

We learn embeddings such that the prediction loss is minimized, i.e., if two words occur in close proximity, their representations become similar.