Introduction to Natural Language Processing (Word Embedding)

Resana Machine Learning Workshop

prepared by Hosein Hasani

Summer 1399

Outline

- Introduction
- Word Representation
- Co-occurrence Matrix
- Word Embedding based on Matrix Factorization
- Skip-gram
- Word Embedding Analogies

Natural Language Processing

- Applications:
  - Sentiment Classification
  - Machine Translation
  - Visual Question Answering
  - Machine Translation
  - Part of Speech Tagging
  - Named Entity Recognition
  - Image Captioning
  - Text Generation
  - …

Word Representation

- Convert words to numbers!
- A traditional solution: one-hot embedding

  apple = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Word Representation

- Convert words to numbers!
- A traditional solution: one-hot embedding

  apple = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

- Document vectors are **orthogonal**

$$\text{motel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$
$$\text{hotel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = 0$$

- Same distance between each pair of words.
- Vector dimension = number of words in vocabulary (e.g., 1000,000)

One-hot Representation

Main problems:

- Lack of notion of similarity:

$$\text{motel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$
$$\text{hotel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = 0$$

- Dimensionality problem:
  20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)

Word Embedding

- We wish to preserve semantic and syntactic relations in a dense low-dimensional **word embedding.**

apple = [0.834 -0.238 0.379 0.229 -0.384]

pear   = [0.793 -0.296 0.303 0.230 -0.453]

An innovative idea:

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

Word Embedding

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

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

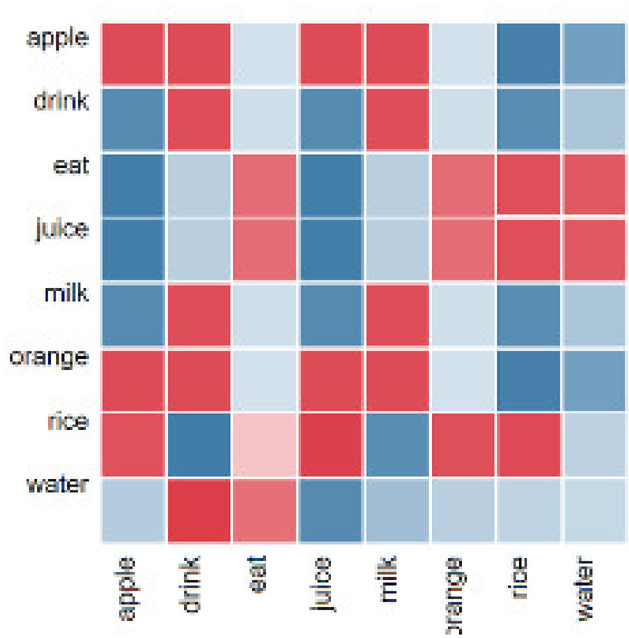
These **context words** will represent ***banking***

The cat got squashed in the garden on Friday.
The dog got flattened in the yard on Monday.

Representation Based on Context

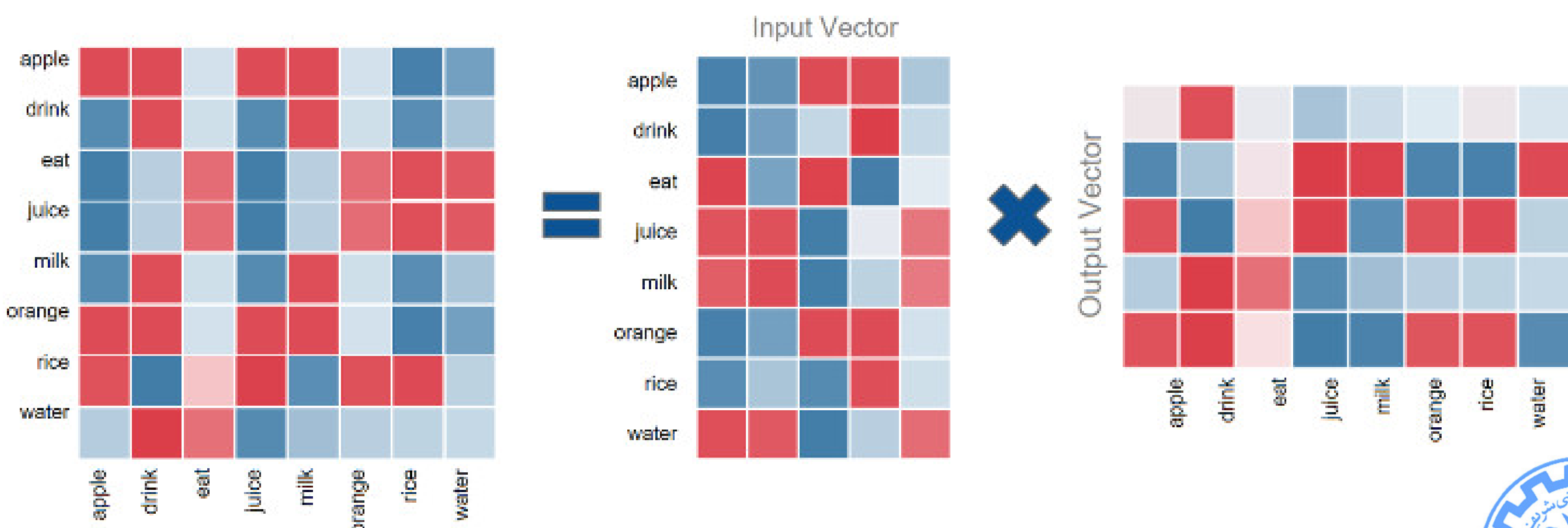
- Word embedding using co-occurrence matrix:

1. Full word-document co-occurrence matrix
   - Captures semantic information
2. Window based co-occurrence matrix
   - Captures both semantic and syntactic information



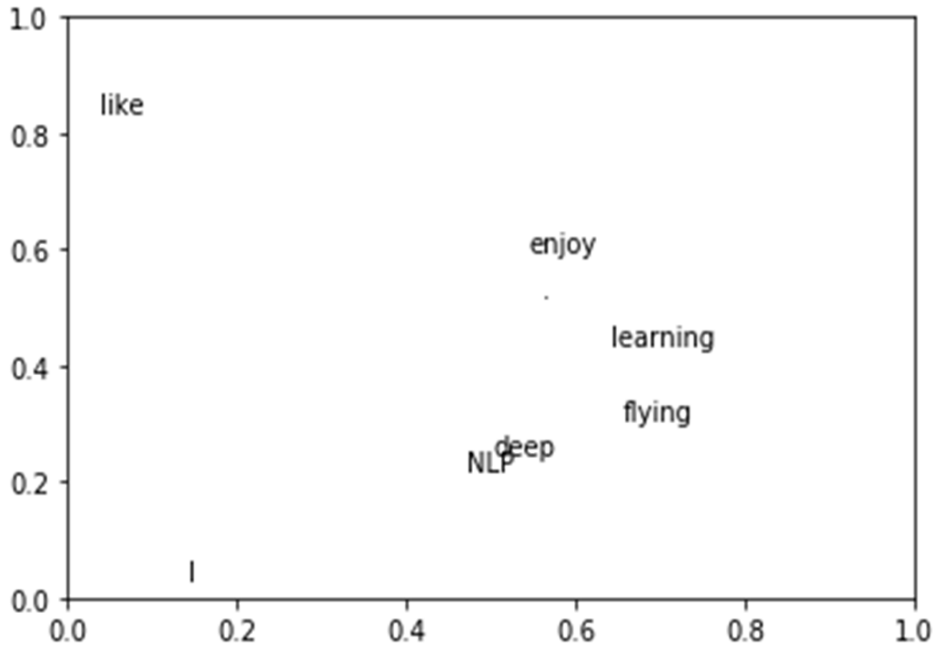
Representation Based on Context

- Word embedding with dimensionality reduction of co-occurrence matrix:
- Extract most informative part of word vectors using matrix factorization (like SVD or QR decomposition).



Window Based Co-occurrence Matrix

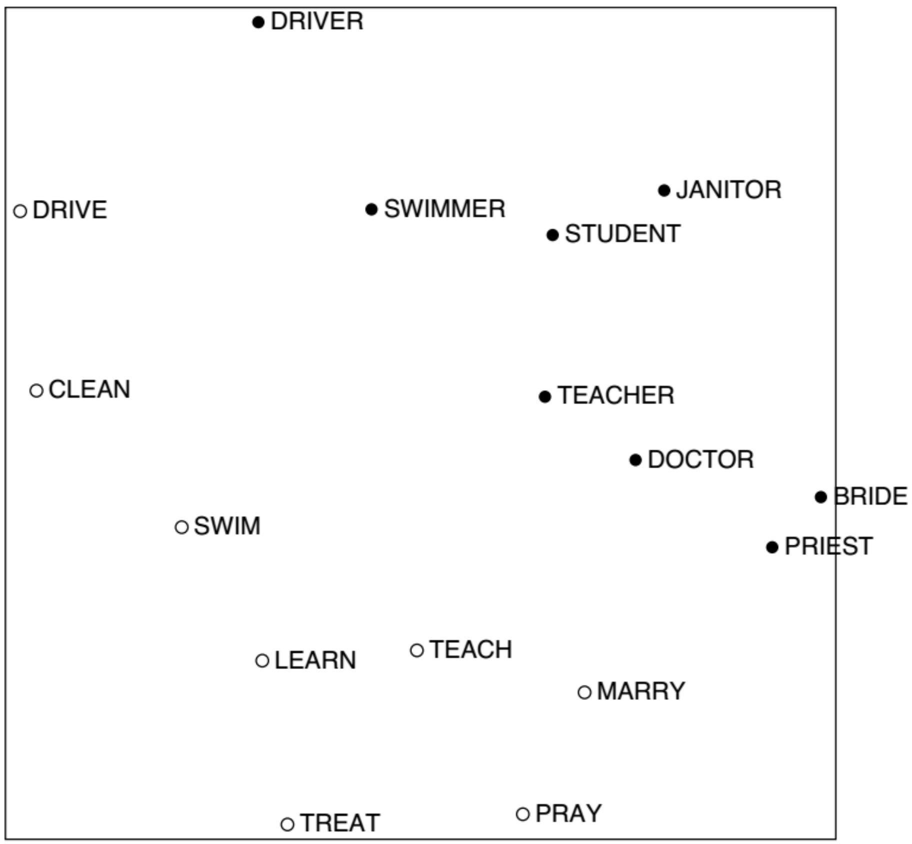
Corpus: I like NLP. I like deep learning. I enjoy flying



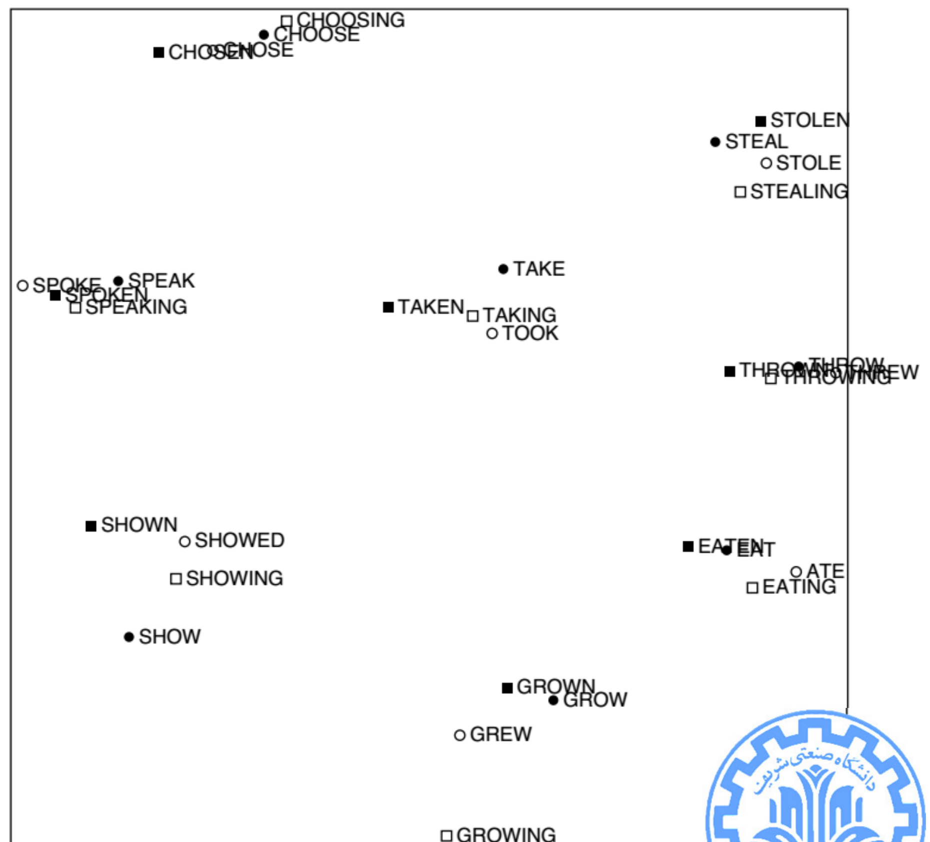

```python
words = ['I', 'like', 'enjoy', 'deep',
     'learning', 'NLP', 'flying', '.']

X = np.array([[0, 2, 1, 0, 0, 0, 0, 0],
              [2, 0, 0, 1, 0, 1, 0, 0],
              [1, 0, 0, 0, 0, 0, 1, 0],
              [0, 1, 0, 0, 1, 0, 0, 0],
              [0, 0, 0, 1, 0, 0, 0, 1],
              [0, 1, 0, 0, 0, 0, 0, 1],
              [0, 0, 1, 0, 0, 0, 0, 1],
              [0, 0, 0, 0, 1, 1, 1, 0]])

U, S, V = np.linalg.svd(X)

for i in range(len(words)):
  plt.text(U[i,0], U[i,1], words[i])
```

Word Vectors Visualization

Syntactic similarity

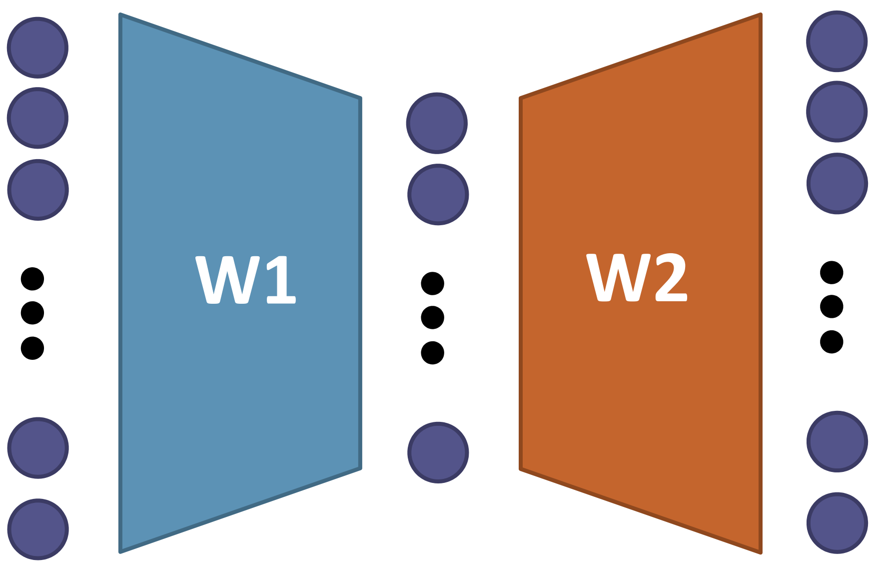


Semantic similarity



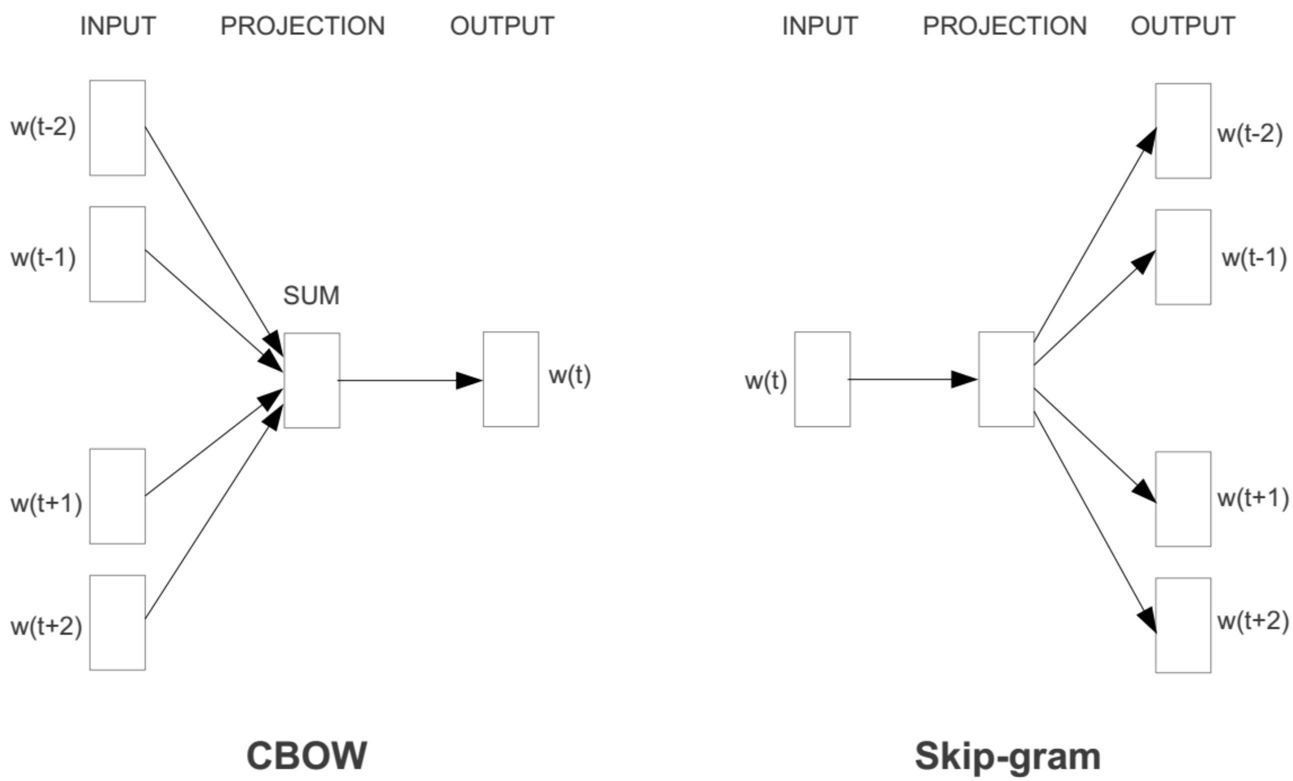
Word2vec

- Dimensionality reduction with a simple two layer MLP network!



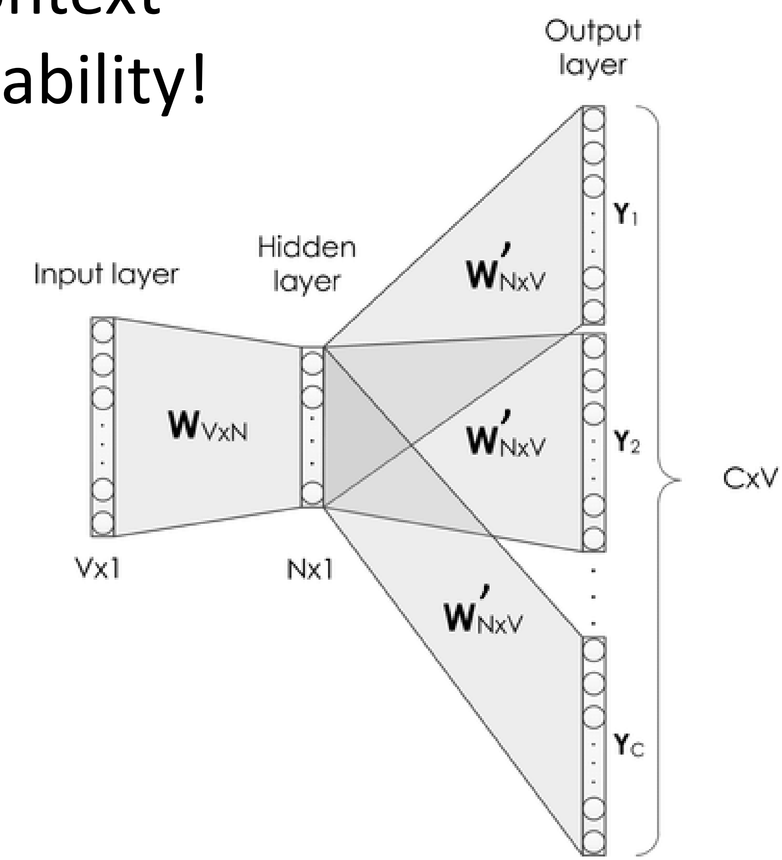
Word2vec

- CBOW: Predict center word from sum of surrounding context words
  Skip-gram: Predict context words given a center word vector.



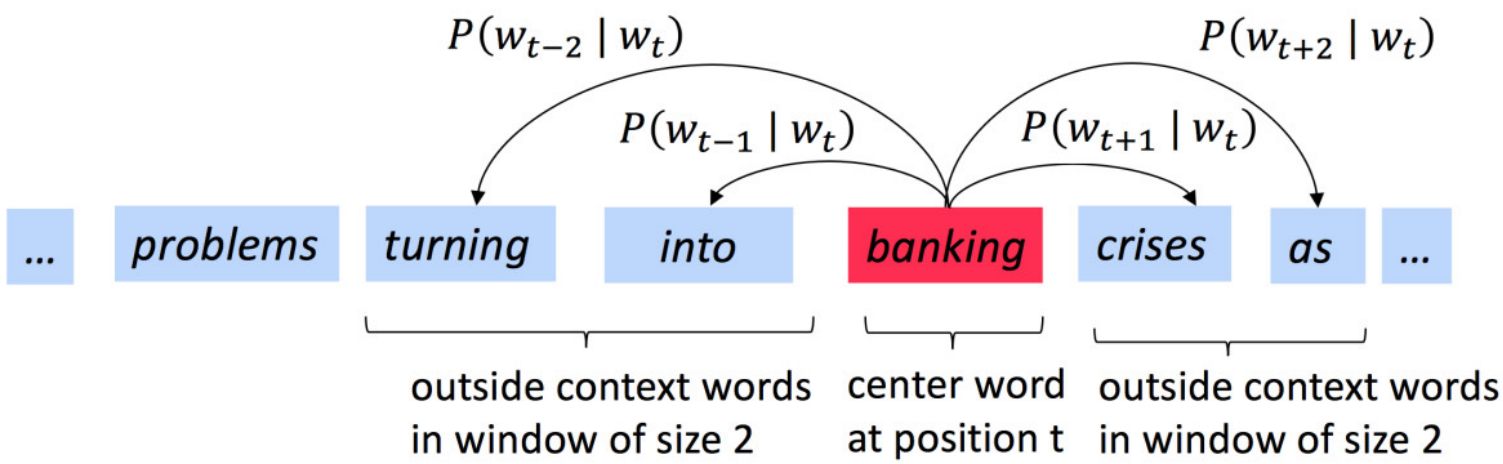
Skip-gram

- Given a center word,
  we can predict the context
  words with high probability!



Skip-gram

- Given a center word, we can predict the context words with high probability!



Skip-gram

- Our final goal is to predict output distribution (softmax) properly, so during the training of skip-gram, we want to maximize the following function:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

model parameters

training samples

words in the context

Skip-gram: Loss Function

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

- The loss function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Skip-gram: Loss Function

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

- The loss function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log \boxed{P\left(w_{t+j} \mid w_t; \theta\right)}$$

How to calculate $P(w_{t+j}|w_t;\theta)$?

Skip-gram

- for a center word *c* and a context word *o*, we have:

Hidden layer:

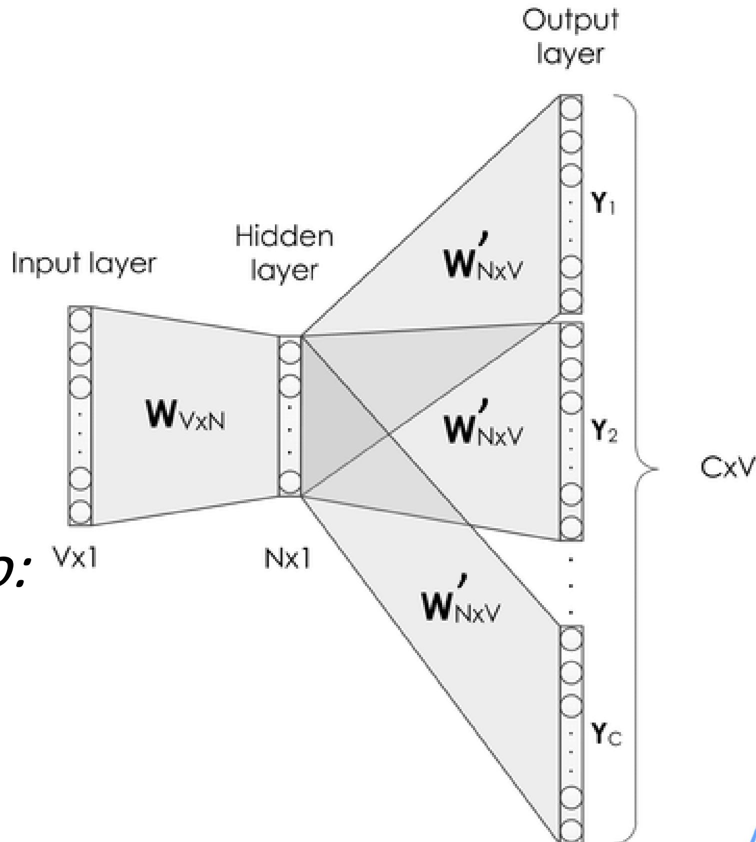
$h_c = W^T x_c = W_{c,} = v_c$

Output layer (befor softmax):

$y_c = {W'}^T h_c$

meanwhile for a context word *o:*

$W'_{.,o} = u_o$



Skip-gram

- Recall softmax function

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_{k=1}^{n} \exp(x_k)}$$

This function maps arbitrary values x_i to a probability distribution p_i

- So, if we apply softmax to the output of skip-gram, we have:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

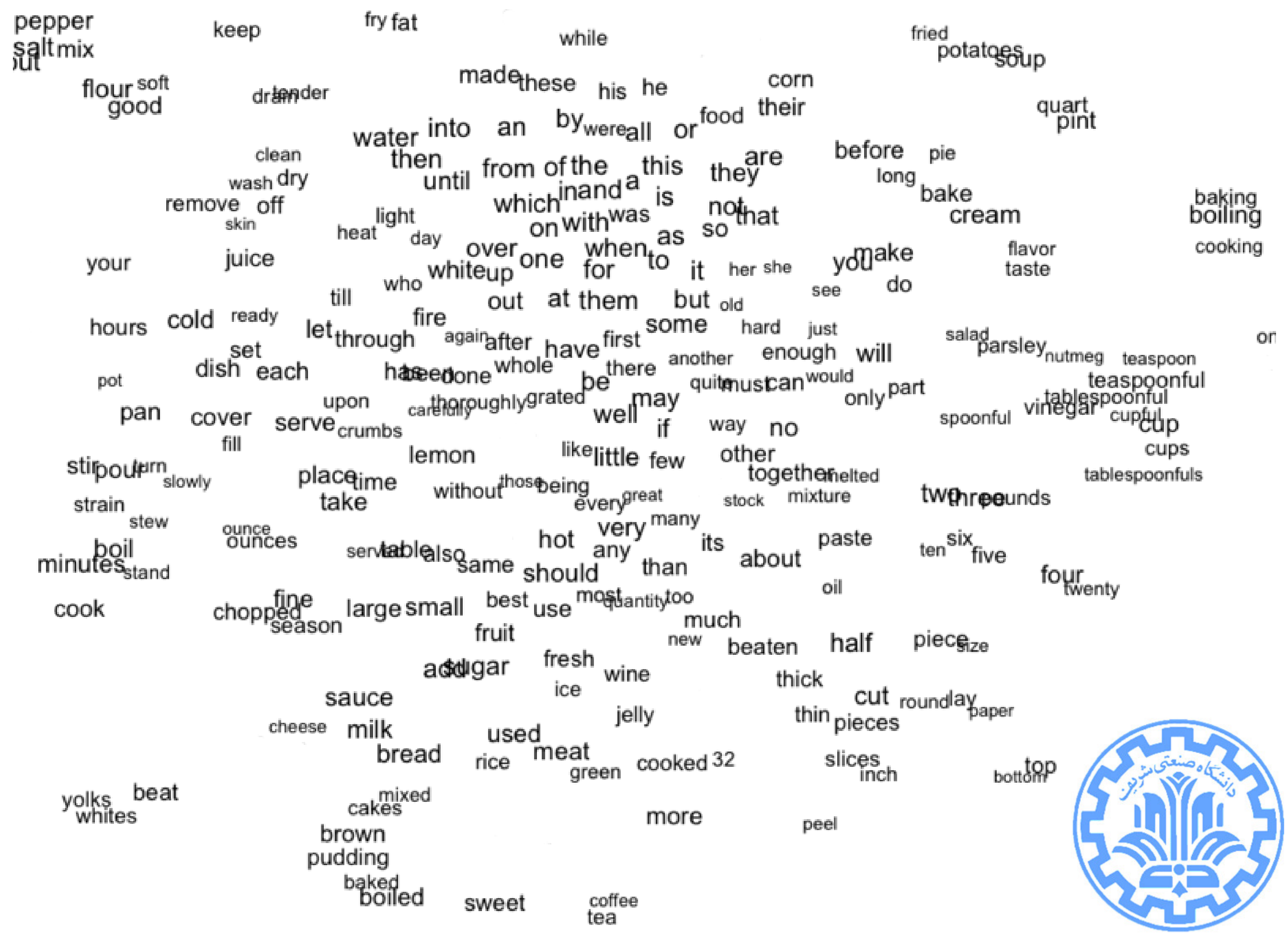
Skip-gram

- Finally, we have new word embedding v_i for a given one-hot word x_i.

Skip-gram

- Finally, we have new word embedding v_i for a given one-hot word x_i.

$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$



Word Vector Analogies

a:b :: c:?

man:woman :: king:?

$$\longrightarrow$$

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

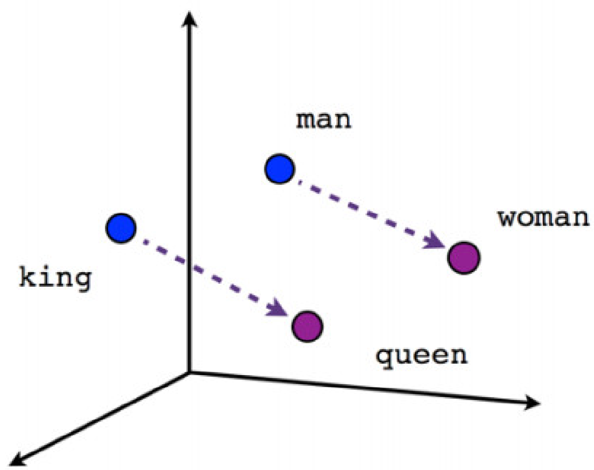
Word Vector Analogies

a:b :: c:?

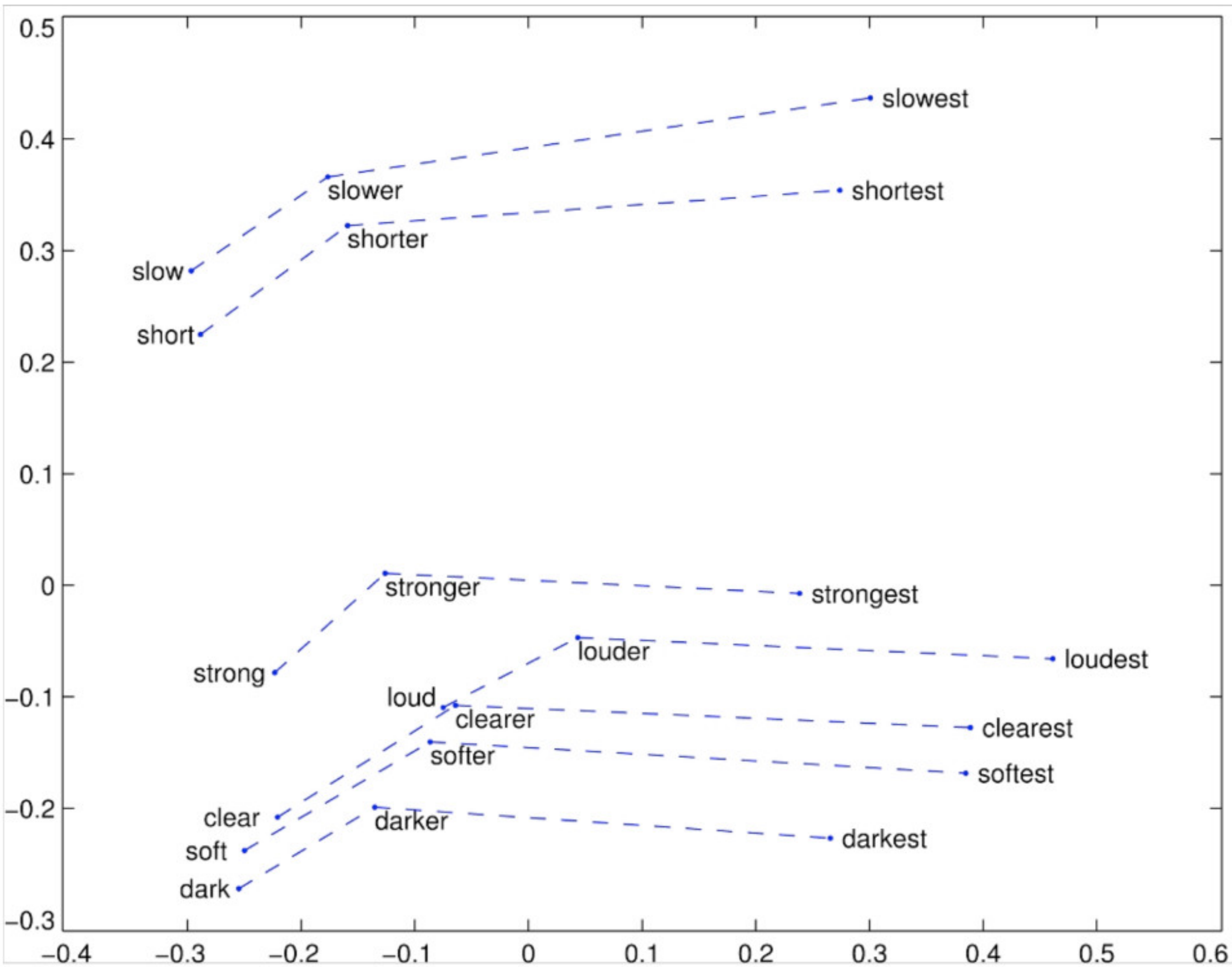
$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

man:woman :: king:?

vector[Queen] ≈ vector[King] - vector[Man] + vector[Woman]



Word Vector Analogies



Main Resources

- Stanford NLP coarse (CS 224N)