Association- and Embedding-based Recommendations

Sungkyunkwan University (SKKU)

Prof. Jongwuk Lee

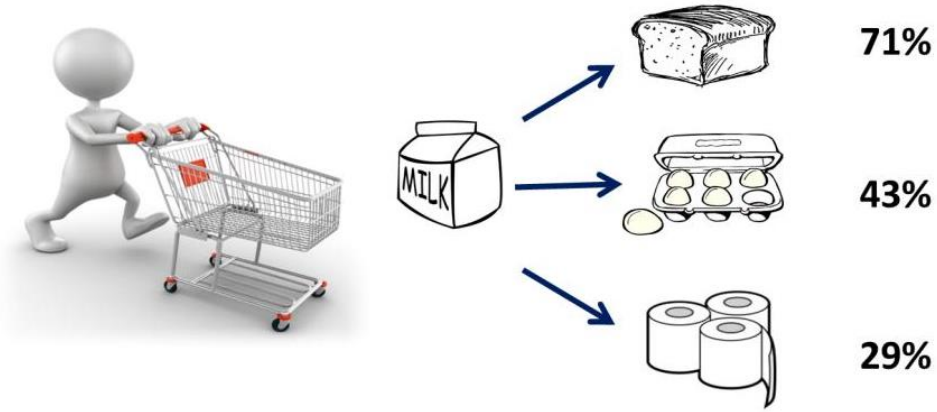
Contents

- Association Rule Mining
- Association-based Recommendation
- Concept of Word Embeddings
- Embedding-based Recommendation
- Co-occurrence-based Recommendation
- StarSpace: Embed All The Things!

Association Rule Mining

Association and Correlation Analysis

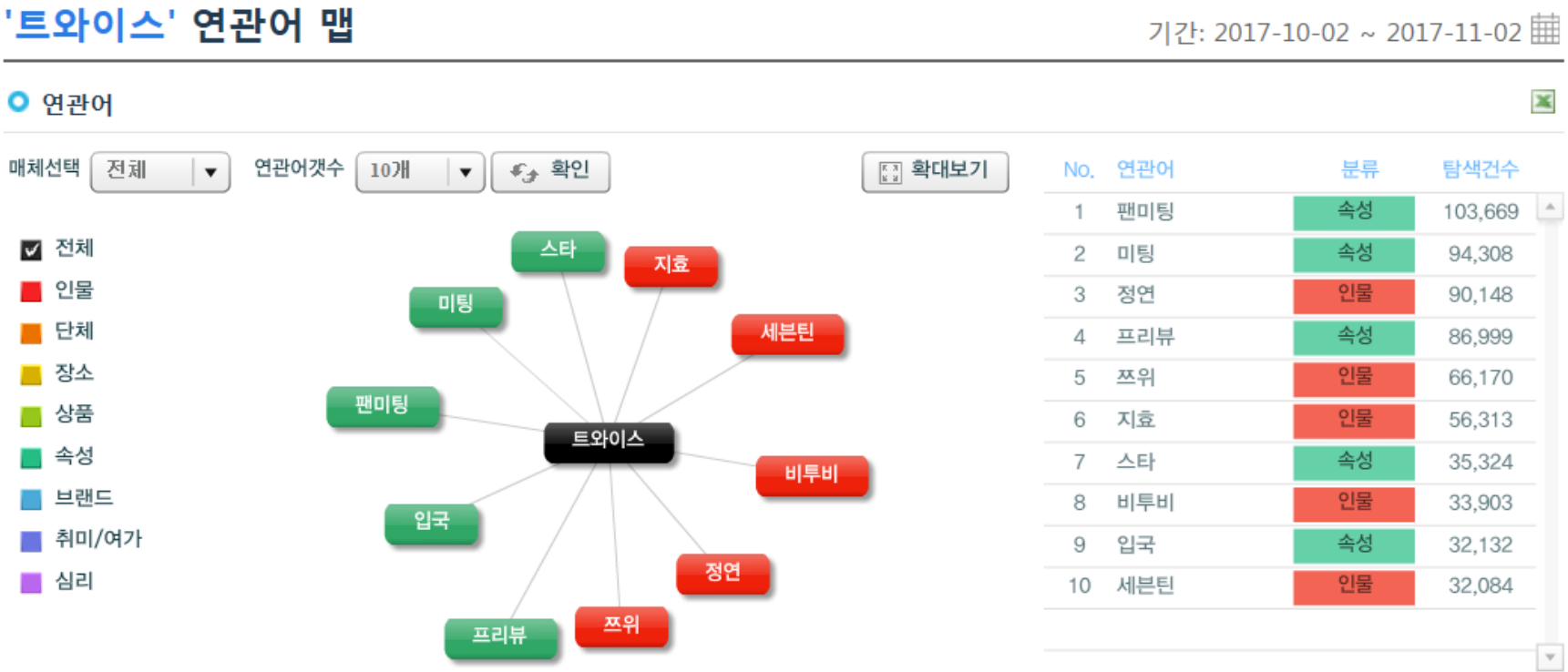
- **Frequent patterns (or frequent item sets)**
  - What items are frequently purchased together in Walmart?
- **A typical association rule:**
  - Diaper → Beer [0.5%, 75%] (support, confidence)



- **Correlation vs. causality**
  - **Note: correlation does not imply causality.**

Example: Word Association Map

➢ Demo: http://www.socialmetrics.co.kr/



What is Frequent Pattern Analysis?

- **Motivation: finding inherent regularities in data**
  - What products were often purchased together?
    - Beer and diapers?
  - What are the subsequent purchases after buying a PC?

- **Applications**
  - Basket data analysis, cross-marketing
  - Catalog design, sale campaign analysis
  - Web log (click stream) analysis
  - DNA sequence analysis

The Market-Basket Model

- **A large set of items**
  - E.g., things sold in a supermarket
- **A large set of baskets**
  - Each basket is a small subset of items.
  - E.g., the things one customer buys on one day

Input:

Tid	Items
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs

Output:

Rules Discovered:
{Milk} → {Coke}
{Diaper, Milk} → {Beer}

- **We want to discover association rules, i.e., if-then rule!**
  - People who bought $\{A, B, C\}$ tend to buy $\{D, E\}$ in Amazon!

Example: Frequent Pattens Analysis

- **Items = products**
- **Baskets = sets of products someone bought**
  - Real market baskets: Chain stores keep TBs of data about what customers buy together
  - Tells how typical customers navigate stores, lets them position tempting items
- **Amazon's people who bought X also bought Y**
- **Suggests tie-in tricks.**
  - E.g., run sale on **diapers** and raise the price of **beer**.

Terminology

- **Itemset: a set of items *k-itemset*** $X = \{X_1, \dots, X_k\}$

- **Support**
  - (**Absolute**) **support** or **support count** of X
    - **Frequency** of occurring an itemset X
  - (**Relative**) **support** is the fraction of transactions that contains X
    - The **probability** that a transaction contains X

- **Minimum support: an itemset X is frequent if X's support is no less than a minimum support threshold.**

Terminology

- **Finding all rules $A \rightarrow B$ with minimum support and confidence**

- **Support: the probability of transactions that contain $A \rightarrow B$**

$$support(A \rightarrow B) = P(A, B) = \boldsymbol{support\_count(A, B)}$$

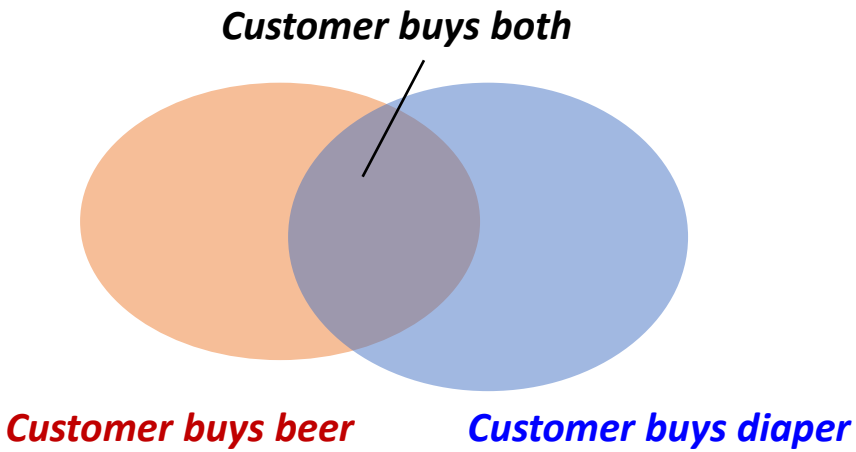
- **Confidence: the conditional probability that the transactions having A also contain B**

$$confidence(A \rightarrow B) = P(B \mid A) = \frac{\boldsymbol{support\_count(A, B)}}{\boldsymbol{support\_count(A)}}$$

Basics of Frequent Patterns

- **Examples**
  - Beer → Diaper (60%, 100%)
  - Diaper → Beer (60%, 75%)

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

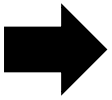


What are Frequent Itemsets?

- An itemset X is frequent if X's support is no less than a minimum support threshold.

- Suppose that minimum support threshold is 2.
- Finding all frequent itemsets with size 1
  - $\{B\} = 3, \{C\} = 2, \{D\} = 4, \{N\} = 3, \{E\} = 3, \{M\} = 2$

Tid	Items bought
10	B, N, D
20	B, C, D
30	B, D, E
40	N, E, M
50	N, C, D, E, M

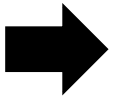


Tid	Items bought
10	$\{B\}, \{N\}, \{D\}$
20	$\{B\}, \{C\}, \{D\}$
30	$\{B\}, \{D\}, \{E\}$
40	$\{N\}, \{E\}, \{M\}$
50	$\{N\}, \{C\}, \{D\}, \{E\}, \{M\}$

Example: Frequent Itemsets

- Suppose that minimum support threshold is 2.
- Finding all frequent itemsets with size 2
  - $\{B, C\} = 1, \{B, D\} = 3, \{B, N\} = 1, \{B, E\} = 1, \{C, D\} = 2,$
  - $\{C, N\} = 1, \{C, E\} = 1, \{C, M\} = 1, \{D, E\} = 2, \{N, D\} = 2,$
  - $\{D, M\} = 1, \{N, E\} = 2, \{N, M\} = 2, \{E, M\} = 2$

Tid	Items bought
10	B, N, D
20	B, C, D
30	B, D, E
40	N, E, M
50	N, C, D, E, M

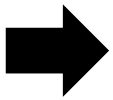


Tid	Items bought
10	$\{B, N\}, \{B, D\}, \{N, D\}$
20	$\{B, C\}, \{B, D\}, \{C, D\}$
30	$\{B, D\}, \{B, E\}, \{D, E\}$
40	$\{N, E\}, \{N, M\}, \{E, M\}$
50	$\{N, C\}, \{N, D\}, \{N, E\}, \{N, M\}, \{C, D\}, \{C, E\}, \{C, M\}, \{D, E\}, \{D, M\}, \{E, M\}$

Example: Frequent Itemsets

- **Suppose that minimum support threshold is 2.**
- **Finding all frequent itemsets with size 3**
  - The answer is $\{N, E, M\} = 2$.

Tid	Items bought
10	B, N, D
20	B, C, D
30	B, D, E
40	N, E, M
50	N, C, D, E, M



Tid	Items bought
10	$\{B, N, D\}$
20	$\{B, C, D\}$
30	$\{B, D, E\}$
40	$\{N, E, M\}$
50	$\{N, C, D\}, \{N, C, E\}, \{N, C, M\},$ $\{N, D, E\}, \{N, D, M\}, \{N, E, M\}$ $\{C, D, E\}, \{C, D, M\}, \{C, E, M\}$ $\{D, E, M\}$

Downward Closure Property

- **If an itemset X is frequent, all subsets of X should be frequent.**

- **Example**
  - If $\{A, B, C\}$ is frequent, is $\{A, B\}$ frequent?
  - If $\{A, B, C\}$ is frequent, is $\{B, D\}$ frequent?
  - If $\{A, B\}$ and $\{A, C\}$ are frequent, is $\{A, B, C\}$ frequent or not?
  - If $\{A, B\}$, $\{A, C\}$, and $\{B, C\}$ are frequent, is $\{A, B, C\}$ frequent or not?

Apriori: Candidate Generation

- **Apriori pruning principle: If any itemset is infrequent, its superset should not be generated!**

- **Overall procedure**
  - Initially, scan DB once to get **frequent 1-itemset**.
  - **Generate length $(k+1)$ candidate itemsets** from length k frequent itemsets.
  - **Test the candidates** against DB.
  - Terminate when no frequent or candidate set can be generated.

Apriori: Overall Procedure

- **Suppose that** $sup_{min} = 2$.

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan

Itemset	Sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{D\}$	1
$\{E\}$	3

Itemset	Sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

Updated itemsets

1-Itemset candidates
$\{A\}$
$\{B\}$
$\{C\}$
$\{D\}$
$\{E\}$

Apriori: Overall Procedure

- **Suppose that** $sup_{min} = 2$.

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

2nd scan

Itemset	Sup
$\{A, B\}$	1
$\{A, C\}$	2
$\{A, E\}$	1
$\{B, C\}$	2
$\{B, E\}$	3
$\{C, E\}$	2

Itemset	Sup
$\{A, C\}$	2
$\{B, C\}$	2
$\{B, E\}$	3
$\{C, E\}$	2

Updated itemsets

2-Itemset candidates
$\{A, B\}$
$\{A, C\}$
$\{A, E\}$
$\{B, C\}$
$\{B, E\}$
$\{C, E\}$

Apriori: Overall Procedure

- **Suppose that** $sup_{min} = 2$.

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

3rd scan

Itemset	Sup
$\{B, C, E\}$	2

3-itemset candidates
$\{B, C, E\}$

Summary of the Apriori Algorithm

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

Itemset	Sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{D\}$	1
$\{E\}$	3

Itemset	Sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

Itemset	Sup
$\{A, B\}$	1
$\{A, C\}$	2
$\{A, E\}$	1
$\{B, C\}$	2
$\{B, E\}$	3
$\{C, E\}$	2

Itemset	Sup
$\{A, C\}$	2
$\{B, C\}$	2
$\{B, E\}$	3
$\{C, E\}$	2

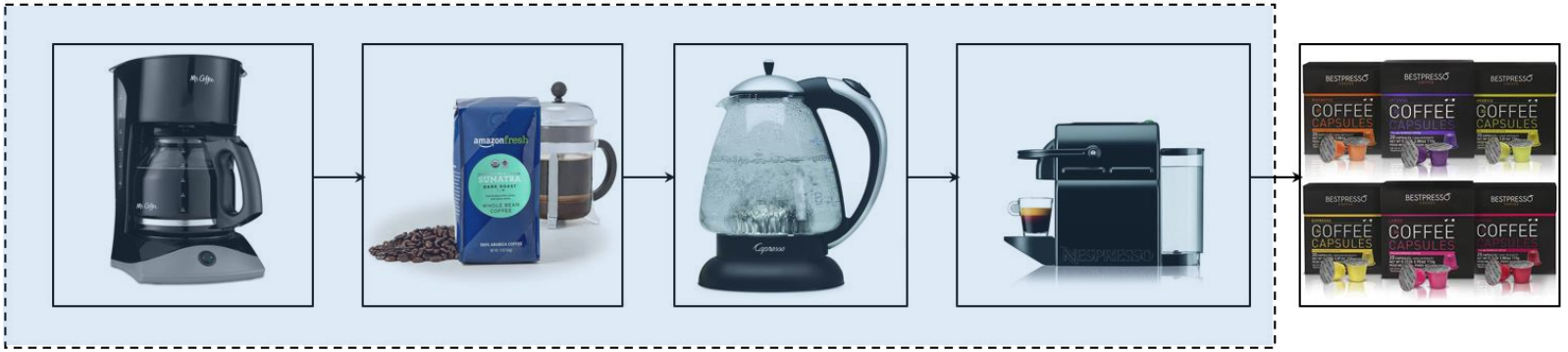
Itemset	Sup
$\{B, C, E\}$	2

Association-based Recommendation

Session-based Recommendation

- Do not require the existence of user-profiles or their entire historical preferences.

- Provide recommendations solely based on a user's interactions in an ongoing session.



Previously visited items

Items to be recommended

Malte Ludewig, Dietmar Jannach, *Evaluation of Session-based Recommendation Algorithms*, 2018 (https://arxiv.org/pdf/1803.09587.pdf)

Markov Chains (MC)

- **Consider the transition probability between two subsequent items in a session.**

- **Count how often users viewed item q immediately after viewing item p.**

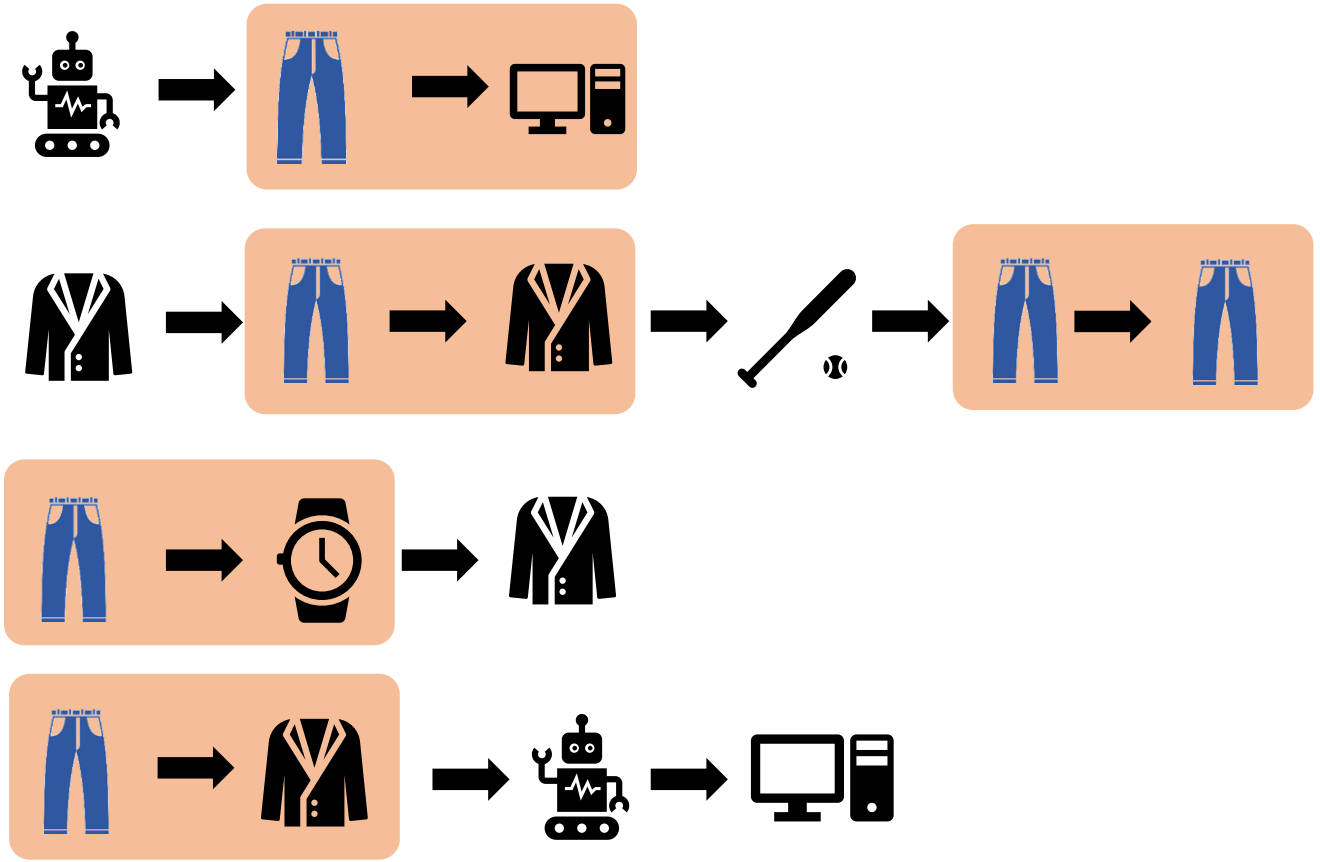
$$score_{MC}(i,s) = \frac{1}{\sum_{p \in S_p} \sum_{x=1}^{|p|-1} \mathbf{1}_{EQ}(s_{|s|}, p_x)} \sum_{p \in S_p} \sum_{x=1}^{|p|-1} \mathbf{1}_{EQ}(s_{|s|}, p_x) \cdot \mathbf{1}_{EQ}(i, p_{x+1})$$

Normalization **Counting scheme**

- S_p: a set of all past sessions
- Let a session $\boldsymbol{s} = (\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_m})$ be an ordered items.
- $\mathbf{1}_{EQ}(a, b)$: 1 if two items a and b refer to the same item, 0 otherwise.

Markov Chains (MC)

- Which item is recommended after viewing jean?



Simple Association Rules (AR)

- A simplified version of the association rule mining technique with a maximum rule size of two.

- Capturing the frequency of two co-occurring events

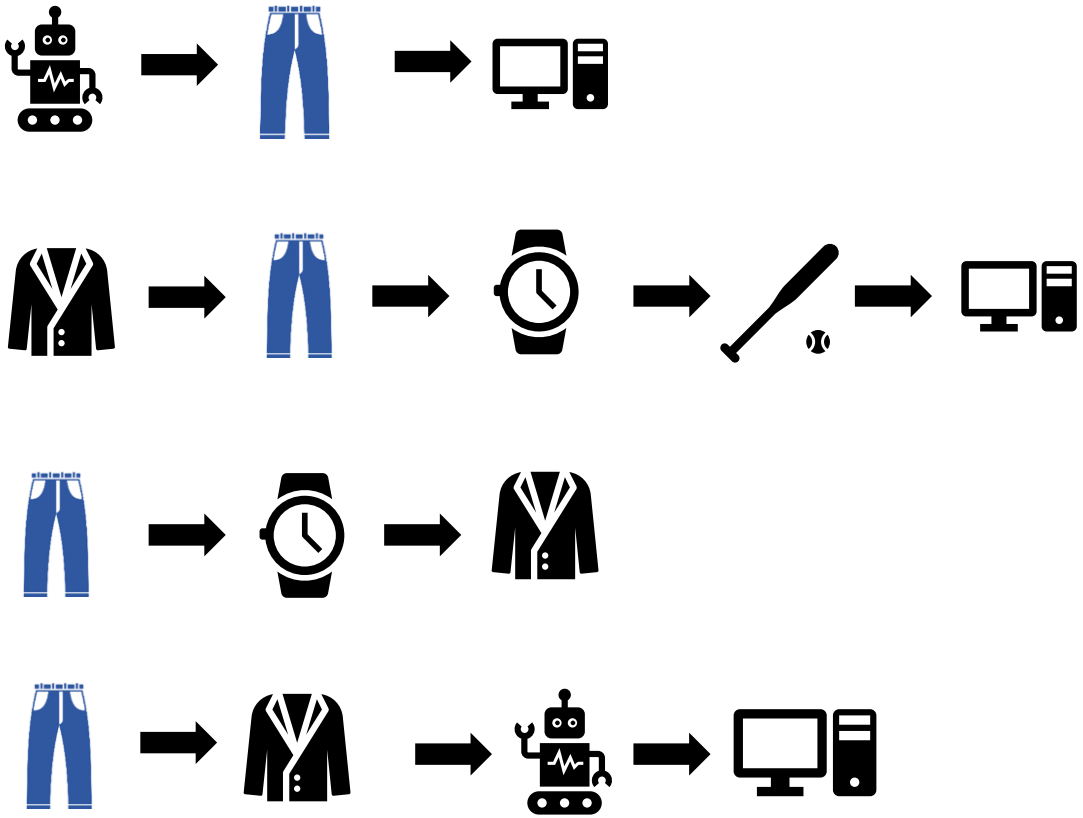
$$score_{AR}(i,s) = \frac{1}{\sum_{p \in S_p} \sum_{x=1}^{|p|} \mathbf{1}_{EQ}(s_{|s|}, p_x) \cdot (|p|-1)} \sum_{p \in S_p} \sum_{x=1}^{|p|} \sum_{y=1}^{|p|} \mathbf{1}_{EQ}(s_{|q|}, p_x) \cdot \mathbf{1}_{EQ}(i, p_y)$$

Normalization

Counting scheme

Simple Association Rules (AR)

- The sequential order does not matter.
- Which item is recommended after viewing jean?



Sequential Rules (SR)

- A variation of MC or AR
- Creating a rule when an item q after an item p in a session even when other events happened between p and q

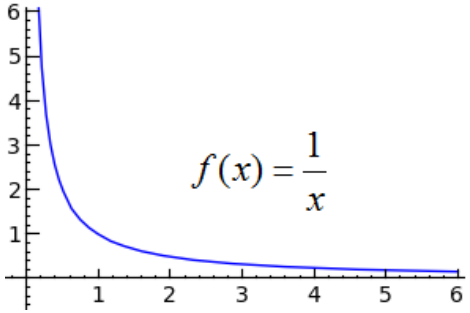
$$score_{SR}(i,s) = \frac{1}{\sum_{p \in S_p} \sum_{x=2}^{|p|} \mathbf{1}_{EQ}(s_{|s|}, p_x) \cdot x} \sum_{p \in S_p} \sum_{x=1}^{|p|} \sum_{y=1}^{x-1} \mathbf{1}_{EQ}(s_{|s|}, p_y) \cdot \mathbf{1}_{EQ}(i, p_x) \cdot w_{SR}(x-y)$$

Normalization **Counting scheme** **Weighting scheme**

$w_{SR}(x) = 1/x$, where x corresponds to the number of steps between the two items

Sequential Rules (SR)

- Given a jean, the co-occurring weights for items are different depending on the position.



1/2

1/3

1/4

Concept of Word Embeddings

Recap: How to Represent Words?

- **In traditional NLP, words are regarded as discrete symbols.**
- **Words are represented by one-hot vectors.**
  - Each word vector has one 1 and the rest 0s.

$$\text{dog} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots\ 0]$$

$$\text{cat} = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots\ 0]$$

$$\text{car} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ \ldots\ 0]$$

$$\text{truck} = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots\ 0]$$

- **Vector dimension = # of words in vocabulary (e.g., $|V| = 10^5$)**

Problems with Discrete Symbols

- If user searches for "Seattle car", we would like to match documents containing "Seattle truck".

$$\text{car} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ \ldots\ 0]$$

$$\text{truck} = [0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots\ 0]$$

- However, two vectors are **orthogonal**.
  - No notion of similarity for one-hot vectors!

- Solution: Learn to **encode similarity** in the vector space.

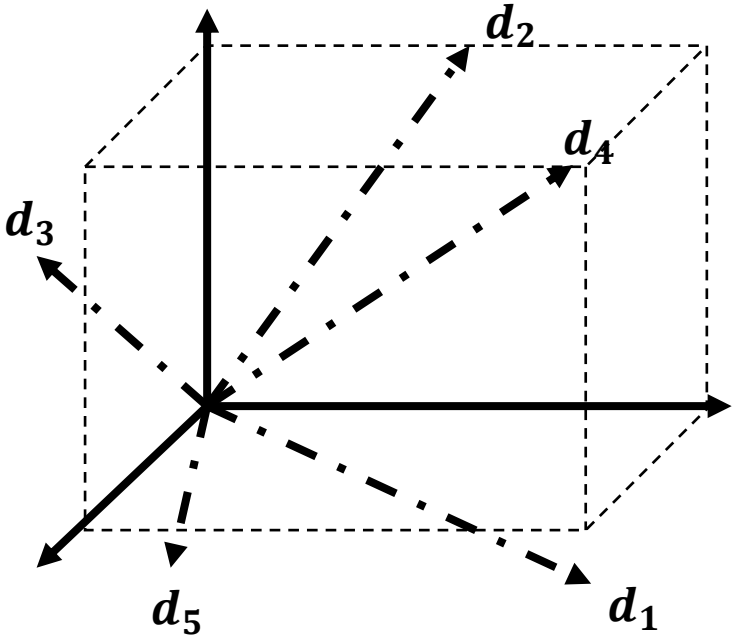
How to Define a Word as a Vector?

- **It is called an embedding because words are embedded into a vector space.**
- **The standard way to represent meanings of words in NLP**
- **A fine-grained model of word meaning for similarities**
- **It is commonly used for pre-trained NLP models.**
  - Sentimental analysis
  - Named entity recognition (NER)
  - Question answering (QA)

Two Representative Embeddings

TF-IDF representation

- A common baseline model
- Words are represented by a simple function of the counts of words.
- **Sparse vector representation**



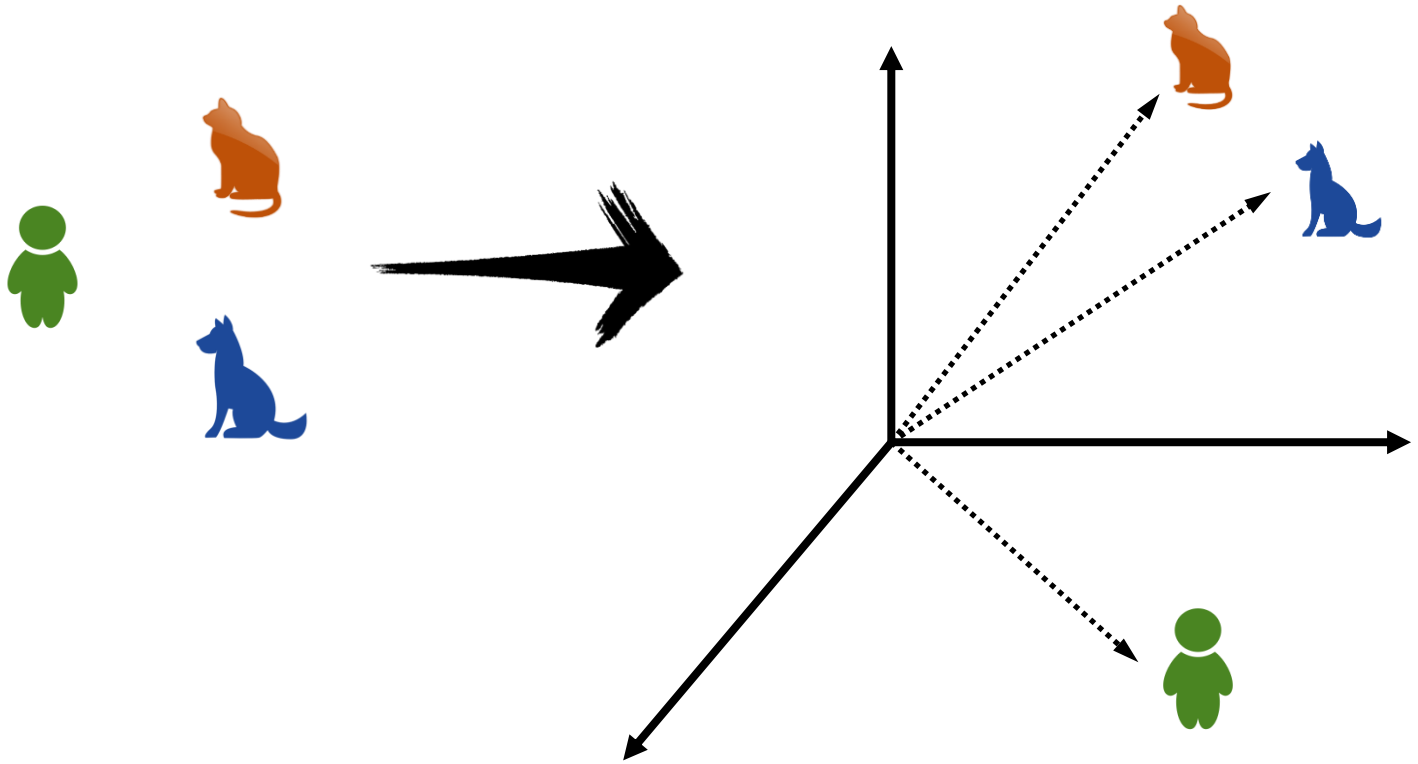
Word2Vec

- Word representation is created by training a classifier to distinguish nearby and far-away words.
- **Dense vector representation**

Word Vectors

- Build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts.

- It is called word embeddings or word representation.



Distributional Hypothesis

- **"You shall know a word by the company it keeps." by J.R. Firth (1957)**
  - Words with **similar contexts** share **similar meanings**.
  - One of the most successful ideas of modern NLP

- **What is Tejuino?**
  - A cup of [ ] is on the table.
  - A woman likes [ ].
  - [ ] makes you drunk.
  - I usually drink [ ] every morning.

Distributional Hypothesis

- **When a word w appears in a text, its context is the set of words that appear nearby (within a fixed-size window).**

… debt problems turning into **banking** crises as happened …
…saying that Europe needs unified **banking** regulation to replace …
… and India have just given their **banking** system a shot …

(window size = 3).

- **Context words represent a target word.**

Word Vectors

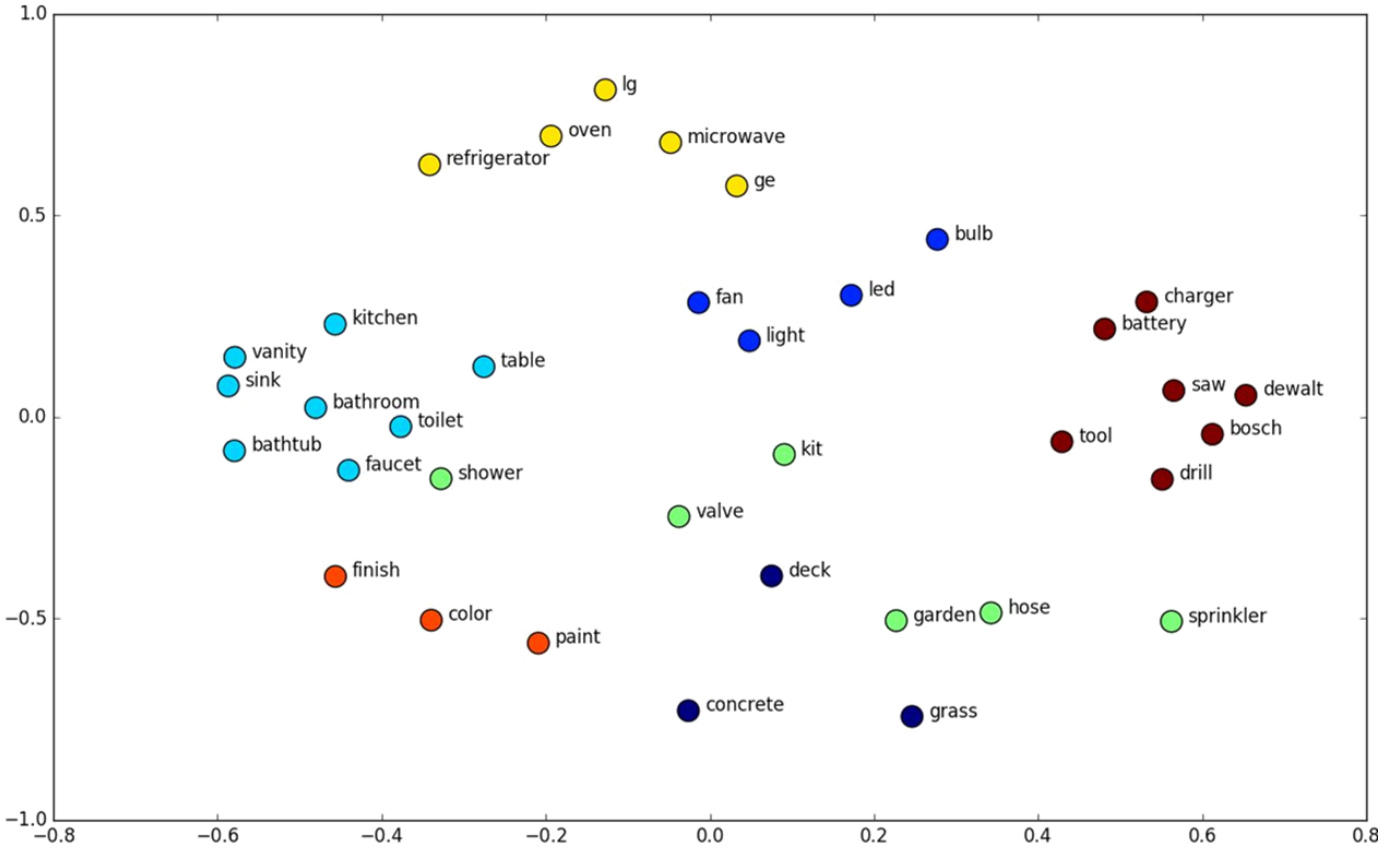
- Build a **dense vector for each word**, chosen so that it is similar to vectors of words that appear in similar contexts.

$$\textbf{banking} = \begin{bmatrix} +2.312 \\ -0.214 \\ +6.792 \\ +1.292 \\ -3.211 \\ +4.249 \end{bmatrix}$$

- They are a distributed or dense representation.
- Note: word vectors are also called **word embeddings**.

Visualization of Word Embeddings

- Each word = a vector
- Similar words are nearby in space.



Two Ideas for Word2Vec

- **A neural networks with a bottleneck, words and context as input and output respectively**

- **Continuous Bag-of-word (CBoW) model**

A cup of _____ is on the table.

- **Skip-gram model**

A __ __ **Tejuino** __ __ the table.

Word2Vec: CBoW

- Maximizing the probability of a target word given a set of context words

"A cup of [????] is on the table."

$\frac{C}{2}$ $\frac{C}{2}$

- C: Window size for context

Word2Vec: Skip-Gram

- Maximizing the probability of a context word given a target word

"A cup [??] Tejuino [??] on the table."

$\frac{C}{2}$ $\frac{C}{2}$

? ? ? ? ? word ? ? ? ? ?

- C: Window size for context

Word2Vec: Overview

- **Mikolov et al., "Distributed Representations of Words and Phrases and their Compositionality", 2013**

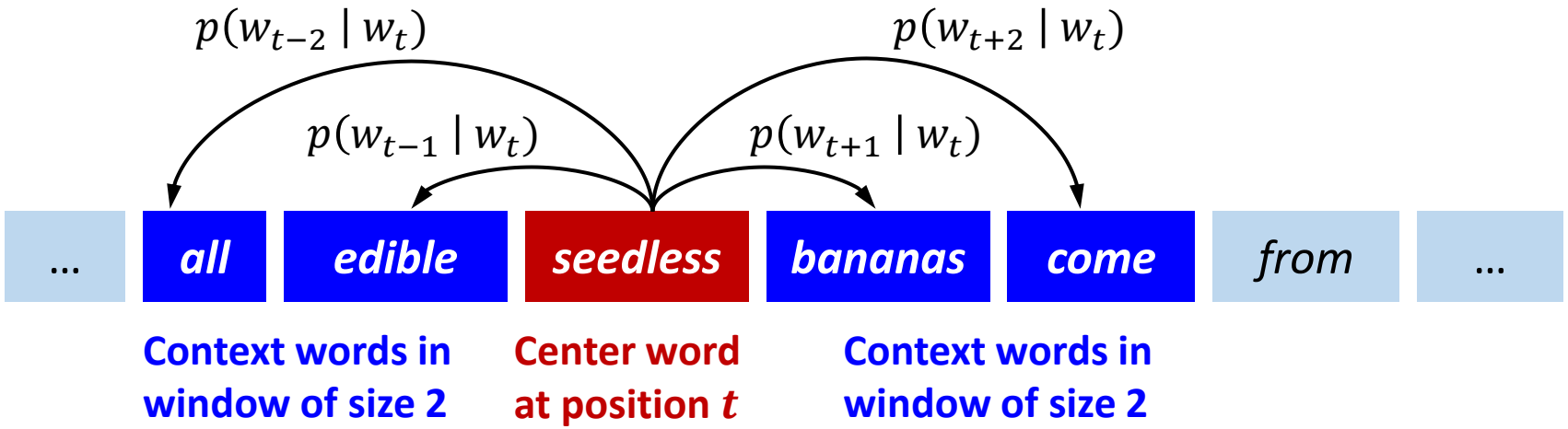
- **Suppose that we have a large corpus of text.**

- **It is a framework for learning word vectors.**
  - Every word in a fixed vocabulary is represented by a vector.
  - Go through each position in the text, which has a target word t and context words c.
  - Use the similarity of the word vectors for t and c to **calculate the probability of c given t** (or vice versa).
  - **Keep adjusting the word vectors** to maximize this probability.

https://arxiv.org/pdf/1310.4546.pdf

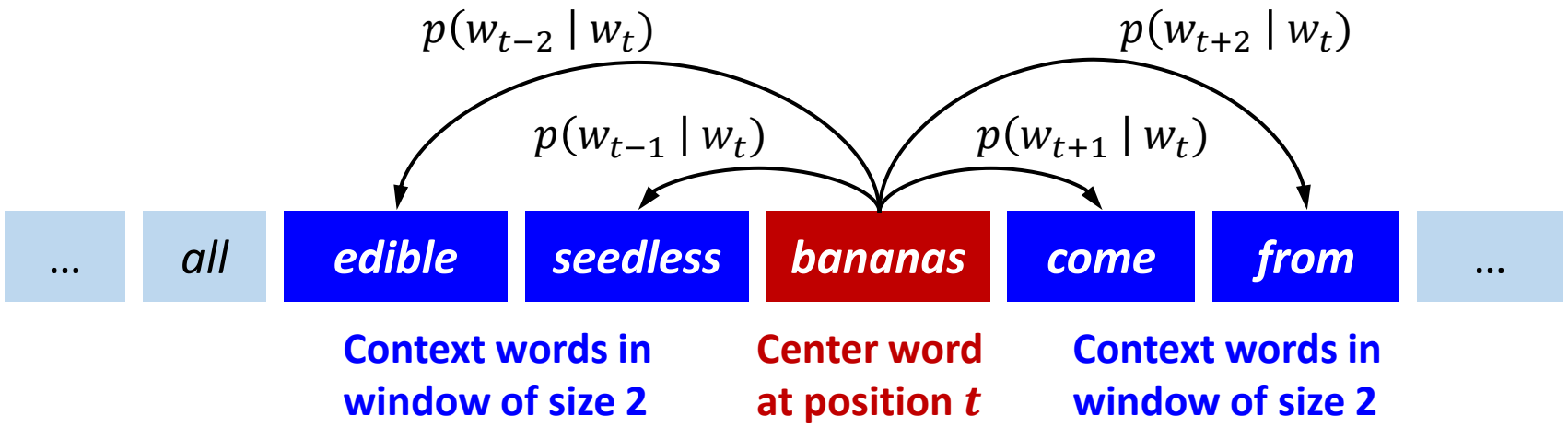
Example of Word2Vec

- Computing $p(w_{t\pm j} \mid w_t)$ in window of size 2



Example of Word2Vec

- Computing $p(w_{t\pm j} \mid w_t)$ in window of size 2



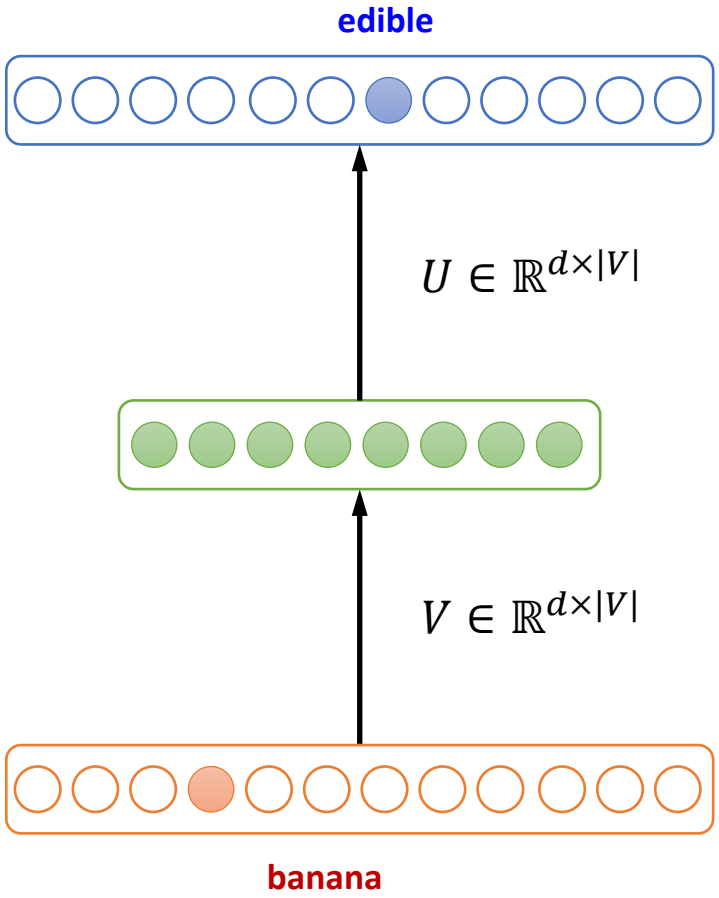
Overview: Word2Vec

- How to compute $p(w_{i\pm j} \mid w_t)$?

- Input layer
  - **One-hot vector** for target word w_t

- Hidden layer
  - **Projection**: Embedding vector for target word w_t

- Output layer
  - **Softmax**: Predicting context word $w_{i\pm j}$



Word2Vec: Projection Layer

- **Projecting a one-hot vector into a hidden layer**
  - V is an embedding matrix for context words.
  - x_i corresponds to i-th column in V.

$$\begin{bmatrix} 0.1 & 1.5 & 1.9 & \dots & 1.2 \\ 0.2 & 2.3 & 0.5 & \dots & 1.1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.3 & 0.1 & 2.1 & \dots & 0.2 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 2.3 \\ \vdots \\ 0.1 \end{bmatrix}$$

$V \in \mathbb{R}^{d \times |V|}$ $\quad x_i \in \mathbb{R}^{|V| \times 1}$ $\quad v_i \in \mathbb{R}^{d \times 1}$

Word2Vec: Output Layer

- **Multiplying v_i to another embedding matrix U^T**
  - U is an embedding matrix for target words.
  - Normalize the output as the probability distribution.

$$\begin{bmatrix} 1.2 & 0.2 & \cdots & 0.5 \\ 1.1 & 0.3 & \cdots & 1.2 \\ 0.8 & 0.5 & \cdots & 1.8 \\ \vdots & \vdots & \ddots & \dots \\ 0.7 & 1.2 & \cdots & 2.3 \end{bmatrix} \cdot \begin{bmatrix} 1.5 \\ 2.3 \\ \vdots \\ 0.1 \end{bmatrix} = \begin{bmatrix} 3.7 \\ 8.5 \\ \vdots \\ 5.1 \end{bmatrix} \rightarrow \begin{bmatrix} 0.2 \\ 0.6 \\ \vdots \\ 0.1 \end{bmatrix} \approx \begin{bmatrix} 0.0 \\ 1.0 \\ \vdots \\ 0.0 \end{bmatrix}$$

Predicted **Actual**

$U^T \in \mathbb{R}^{|V| \times d}$ $v_i \in \mathbb{R}^{d \times 1}$ $u_{i+j}^T v_i \in \mathbb{R}^{|V| \times 1}$

Word2Vec: Objective Function

- For each position $i = 1, \ldots, T$, predict **context words** within a window of fixed size m, given center word w_j.

Likelihood =

$$\mathcal{L}(\theta) = \prod_{i=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} p(w_{i+j} \mid w_i ; \theta)$$

θ is all variables to be optimized.

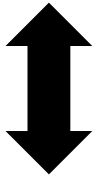
- The objective function is the (average) negative log likelihood.

$$J(\theta) = -\frac{1}{T} \log \mathcal{L}(\theta) = -\frac{1}{T} \sum_{i=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log p(w_{i+j} \mid w_i ; \theta)$$

Word2Vec: Objective Function

Maximizing the likelihood function

$$\mathcal{L}(\theta) = \prod_{i=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} p(w_{i+j} \mid w_i ; \theta)$$



Minimizing the objective function

$$J(\theta) = -\frac{1}{T} \log \mathcal{L}(\theta) = -\frac{1}{T} \sum_{i=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log p(w_{i+j} \mid w_i ; \theta)$$

Word2Vec: Objective Function

- Our goal is to minimize the objective function.

$$J(\theta) = -\frac{1}{T}\log \mathcal{L}(\theta) = -\frac{1}{T}\sum_{i=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{i+j} \mid w_i\,;\,\theta)$$

- How to calculate $p(w_{i+j} \mid w_i; \theta)$?

Word2Vec: Objective Function

- **To calculate $p(w_{i+j} \mid w_i; \theta)$, use two vectors per word w.**
  - v_i when word w_i is a context word
  - u_{i+j} when word w_{i+j} is a target word

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{i=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log p(w_{i+j} \mid w_i ; \theta)$$

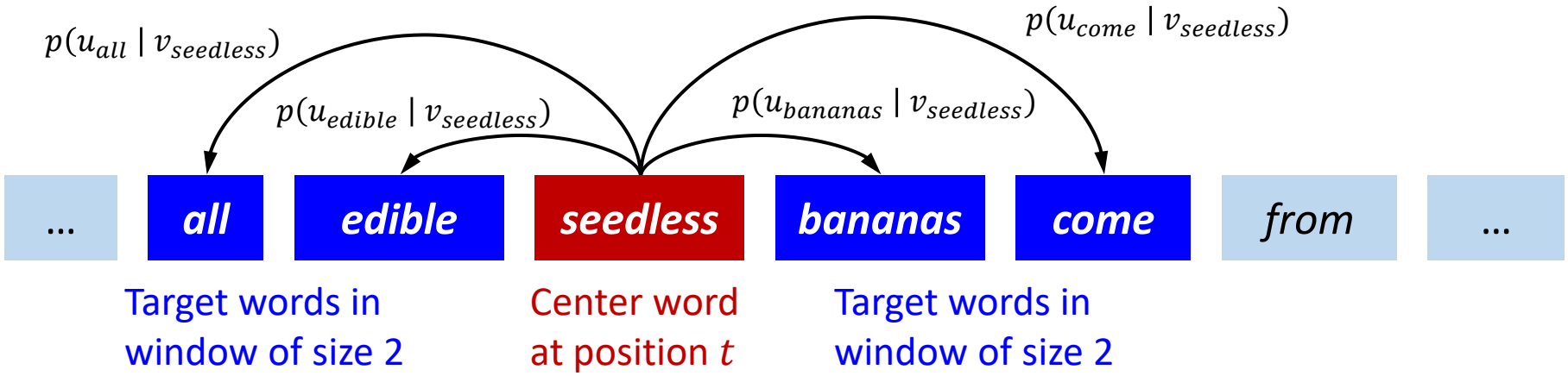
- **For a target word t and a context word c**

$$p(w_{i+j} \mid w_i) = \frac{\exp(u_{i+j}^T v_i)}{\sum_{w\in V}\exp(u_w^T v_i)}$$

Example: Word2Vec with Vectors

- Computing $p(\boldsymbol{all} \mid \boldsymbol{seedless})$ in window of size 2
- Use two vectors u_{all} and $v_{seedless}$.

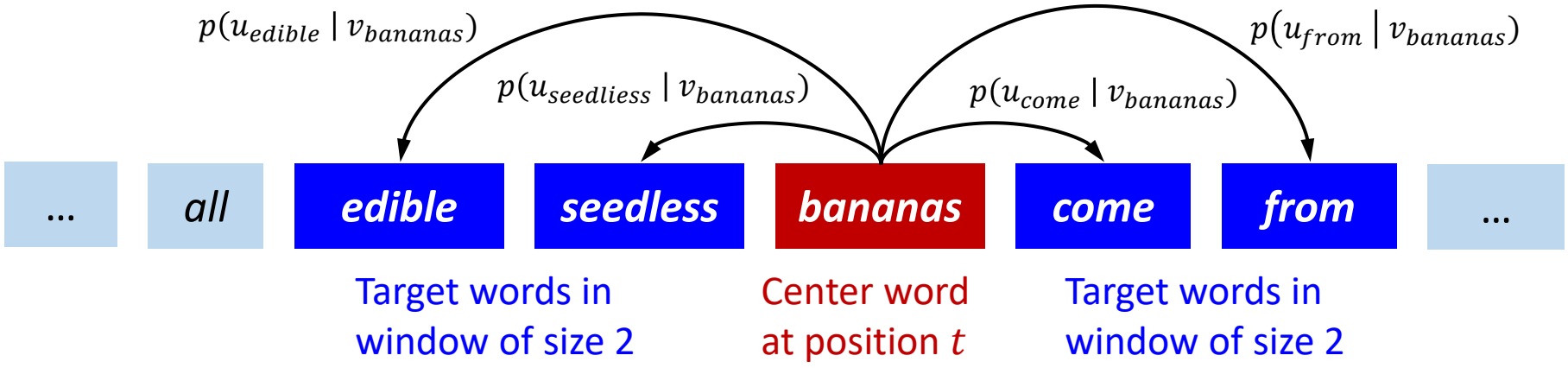
$$p(all \mid seedless) = \frac{\exp(u_{all}^T v_{seedless})}{\sum_{w \in V} \exp(u_w^T v_{seedless})}$$



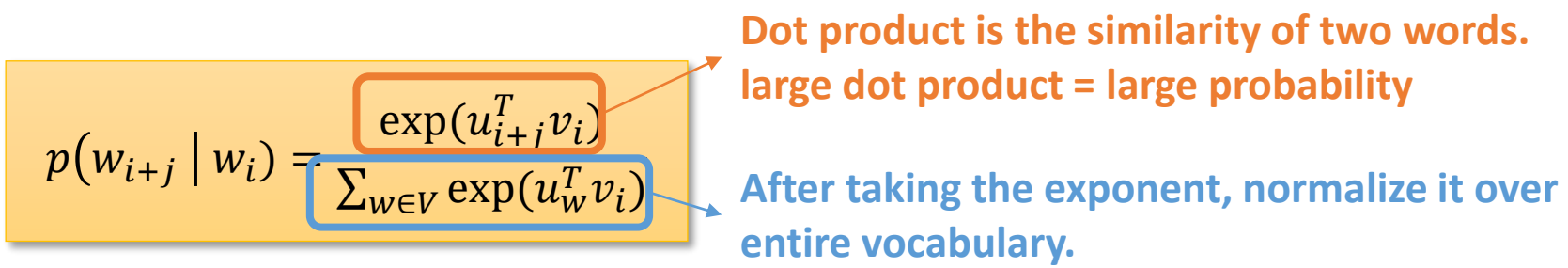
Example: Word2Vec with Vectors

- Computing $p(\boldsymbol{edible} \mid \boldsymbol{bananas})$ in window of size 2
- Use two vectors $u_{editable}$ and $v_{bananas}$.

$$p(edible \mid banana) = \frac{\exp(u_{edible}^T v_{bananas})}{\sum_{w \in V} \exp(u_w^T v_{bananas})}$$



Word2Vec: Prediction Function



- This is an example of the **softmax function** $\mathbb{R}^n \rightarrow \mathbb{R}^n$

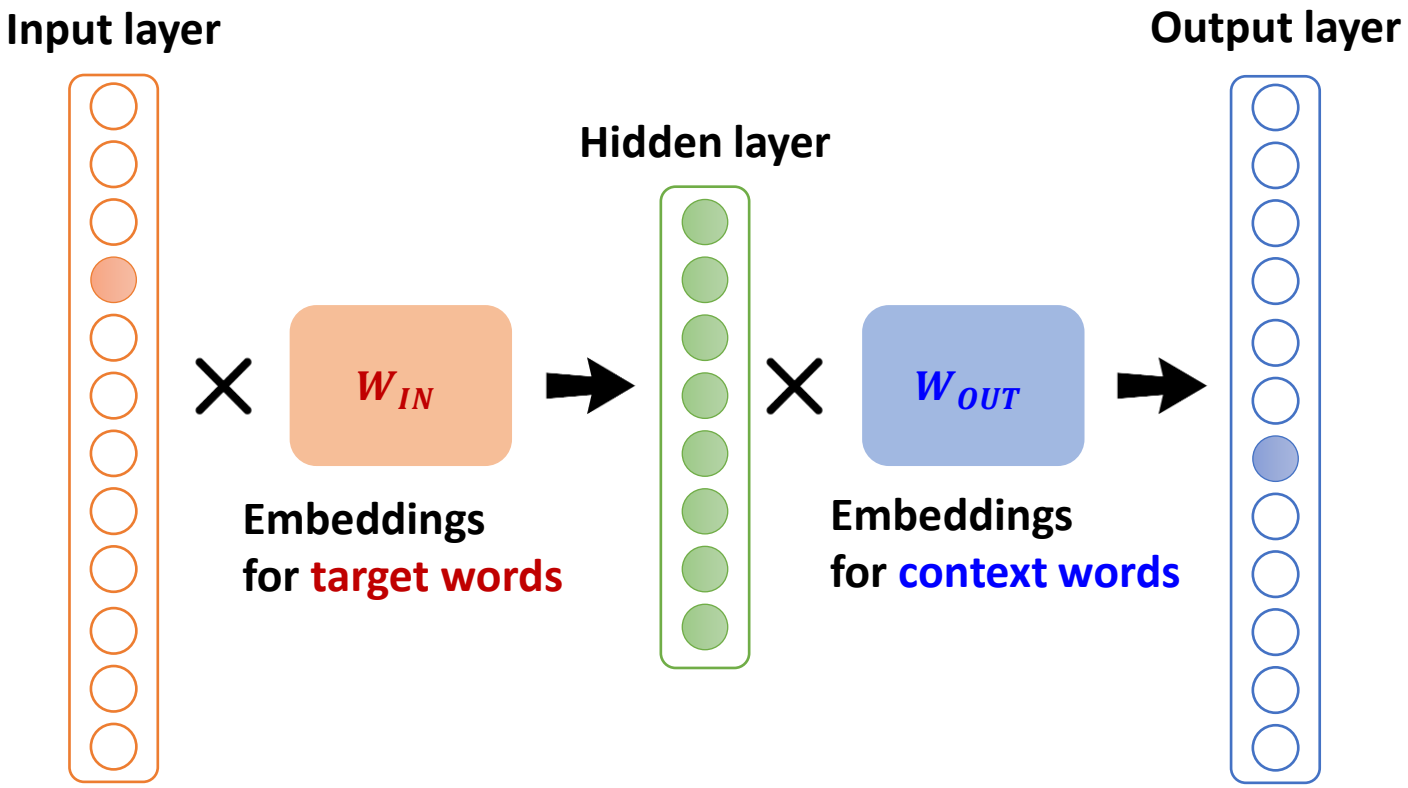
$$softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- This maps a value x_i to a probability distribution p_i
  - **"soft"** because it still assigns some probability to smaller x_i
  - **"max"** because it amplifies the probability of the largest x_i

Word2Vec: Prediction Function

- Each word is represented as an embedding vector

$$p(w_{i+j} \mid w_i) = \frac{\exp(u_{i+j}^T v_i)}{\sum_{w \in V} \exp(u_w^T v_i)}$$



Word2Vec: Parameters to Be Learned

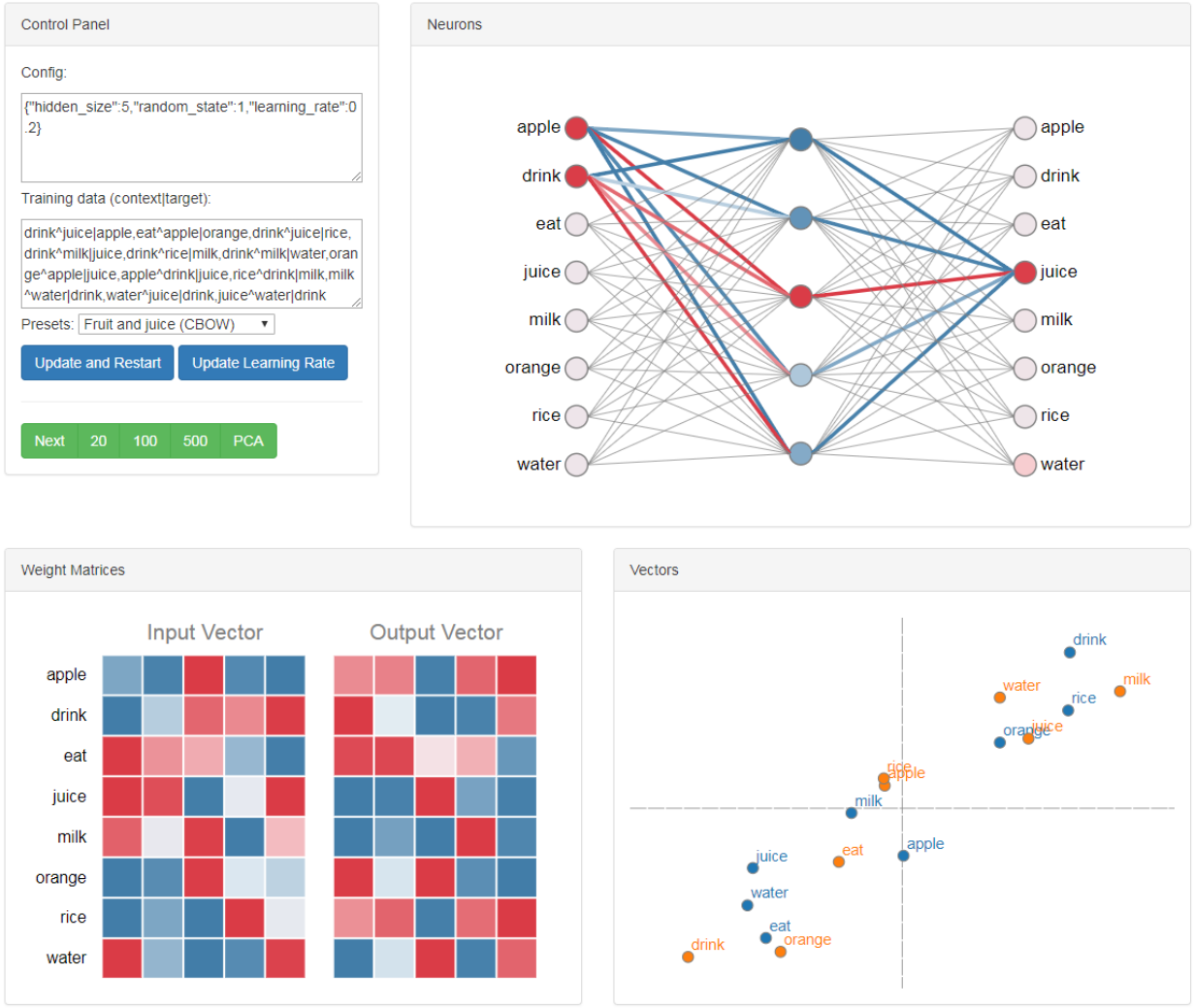
- **Each word has two vectors as a context and a target word.**
  - v_i: word w_i is a context word.
  - u_i: word w_i is a target word.
- **Consider d-dimensional vectors for V words.**

$$\begin{bmatrix} u_a \\ u_{all} \\ \vdots \\ u_{banana} \\ \vdots \\ u_{zebra} \\ u_{zoo} \end{bmatrix} \in \mathbb{R}^{d \times |V|} \qquad \begin{bmatrix} v_a \\ v_{all} \\ \vdots \\ v_{banana} \\ \vdots \\ v_{zebra} \\ v_{zoo} \end{bmatrix} \in \mathbb{R}^{d \times |V|}$$

- **We have to learn parameter $\theta \in \mathbb{R}^{2(d \times |V|)}$.**

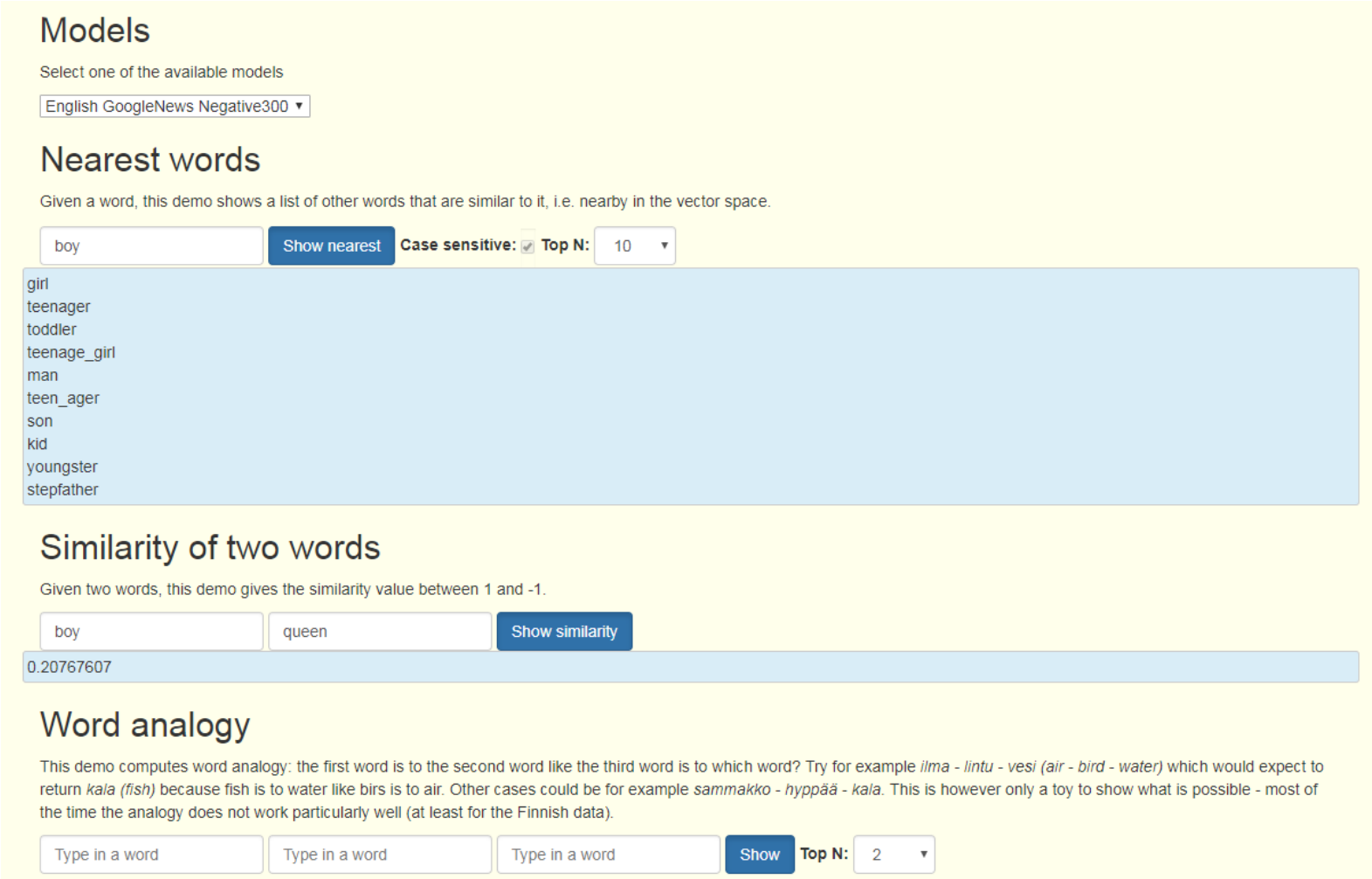
Online Demo: Training Word2Vec

- https://ronxin.github.io/wevi/



Online Demo: Word2Vec

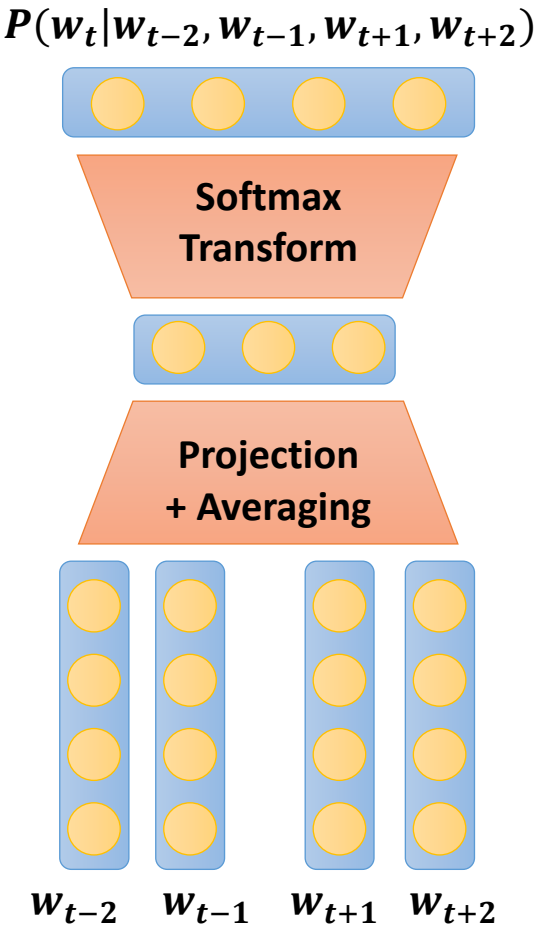
- http://bionlp-www.utu.fi/wv_demo/



Embedding-based Recommendation

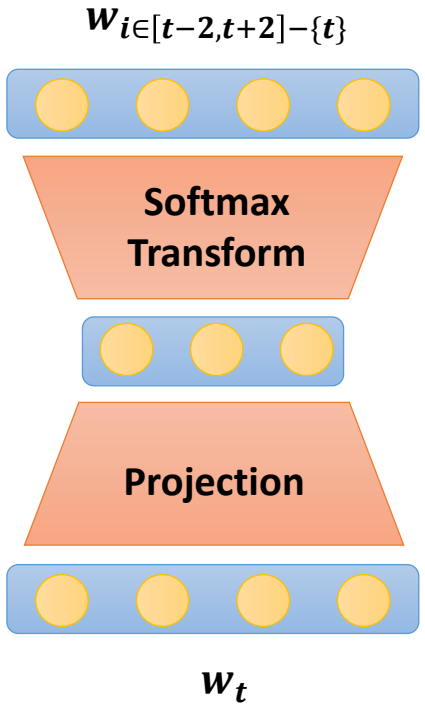
Word2Vec: CBoW

- **Maximizing the probability of a target word given a set of context words**
  - Input: one-hot encoded context words
  - Input to hidden weights
    - Embedding matrix of words
  - Hidden layer
    - Sum of the embedding of the context words
  - Hidden to output weights
  - Softmax transformation
    - Highlight the highest value.
  - Output: likelihood of words given the context words



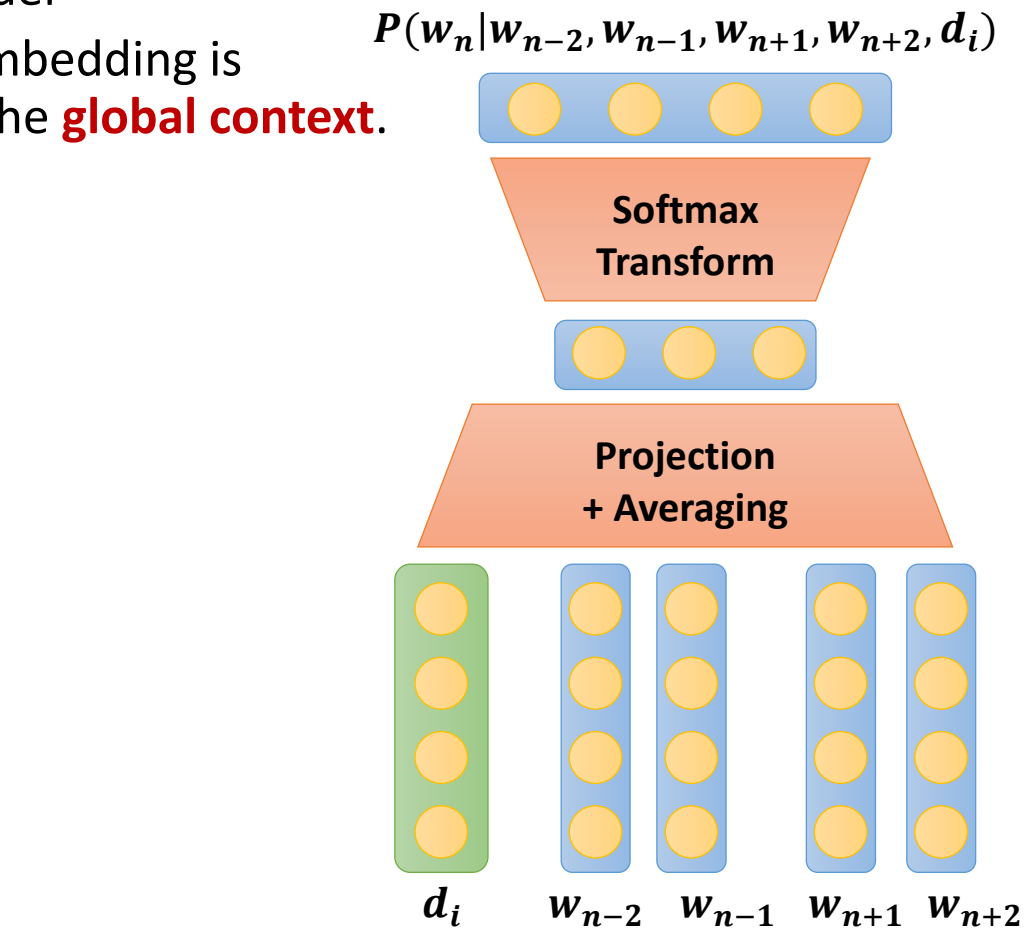
Word2Vec: Skip-Gram

- **Maximizing the probability of a context word given a target word**
  - Input: one-hot encoded target word
  - Input to hidden weights
    - Embedding matrix of words
  - Hidden layer
    - Embedding of the target word
  - Hidden to output weights
  - Softmax transformation
  - Output: likelihood of a context word given the target word



Doc2Vec

- **Learning latent representations of paragraphs/documents**
  - Based on the CBoW model
  - Paragraph/document embedding is added to the model as the **global context**.



Item Embedding

- **Learning a vector representing an item using co-occurrence between items**
  - Similar items may be embedded as similar vectors.

- **Use in recommendations**
  - **Initialization for item representations** in more advanced algorithms
  - **Item-to-item recommendation**

Prod2Vec

- **How to perform product embedding?**

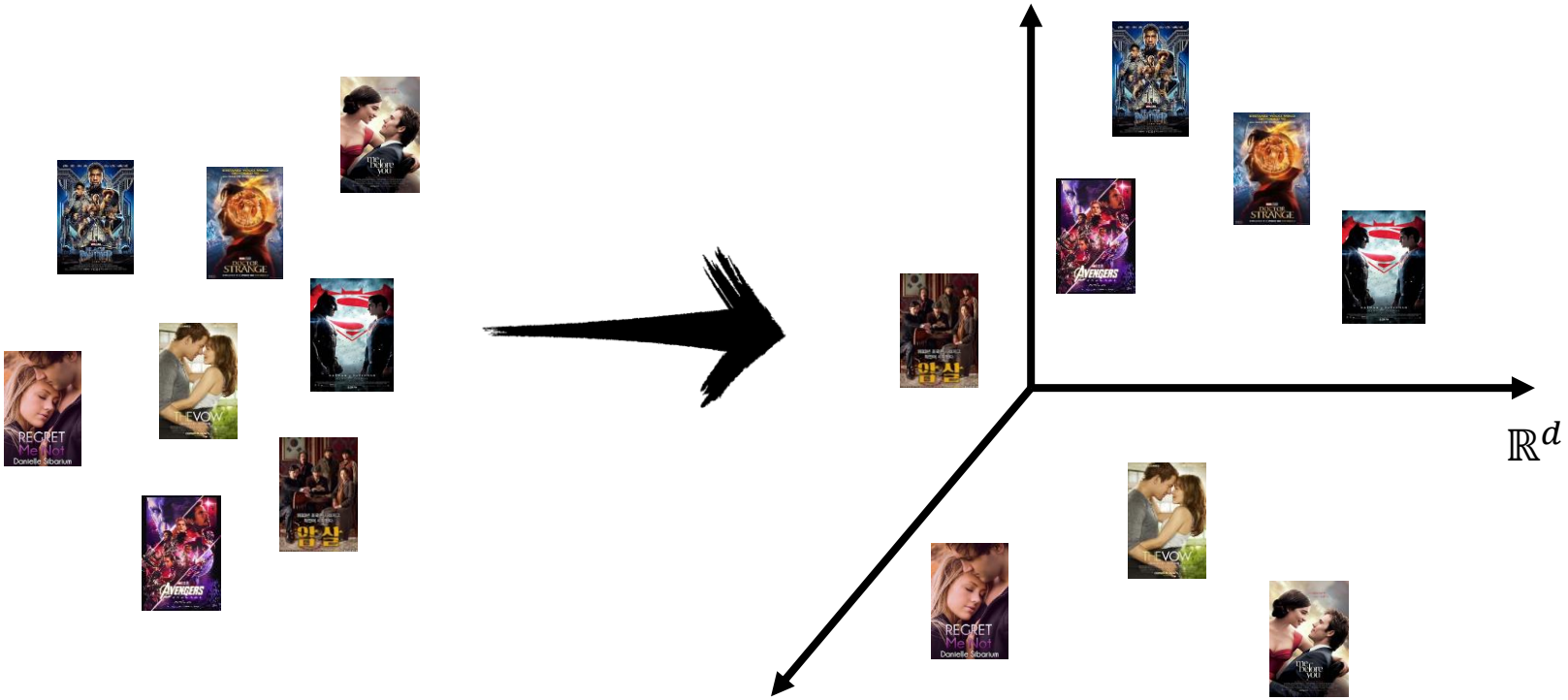
A set of words → a set of products purchased by a user

- **Adopt the skip-gram model on products.**
  - Input: i-th product purchased by the user
  - Output: the other product purchased by the user

- **Learning user representation: Use the doc2vec model.**
  - User embedding is added as the **global context**.
    - Input: $user\ id$ + products except for the i-th product purchased by the user
    - Output: the i-th product purchased by the user

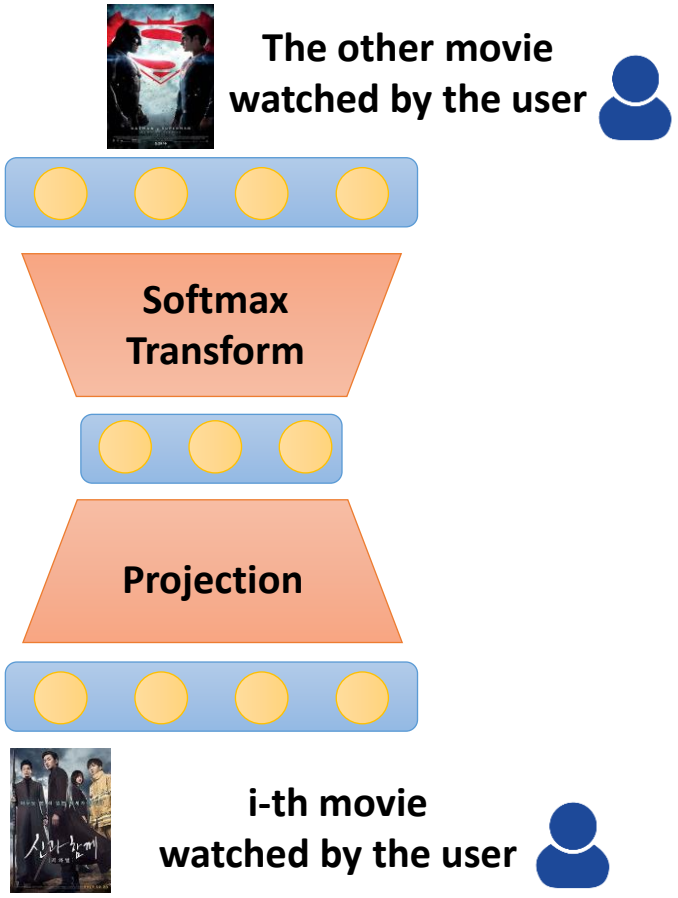
Product Embedding Vectors

- Build a **dense vector for each product** so that it is similar to vectors of products that appear in similar contexts.

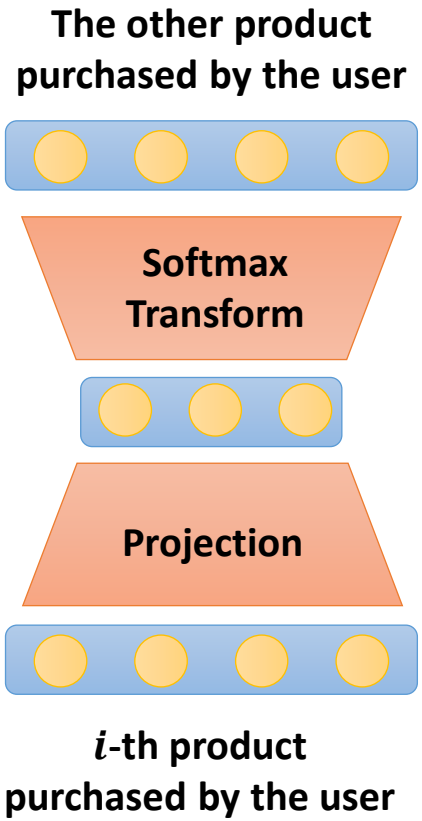


Example of Prod2Vec

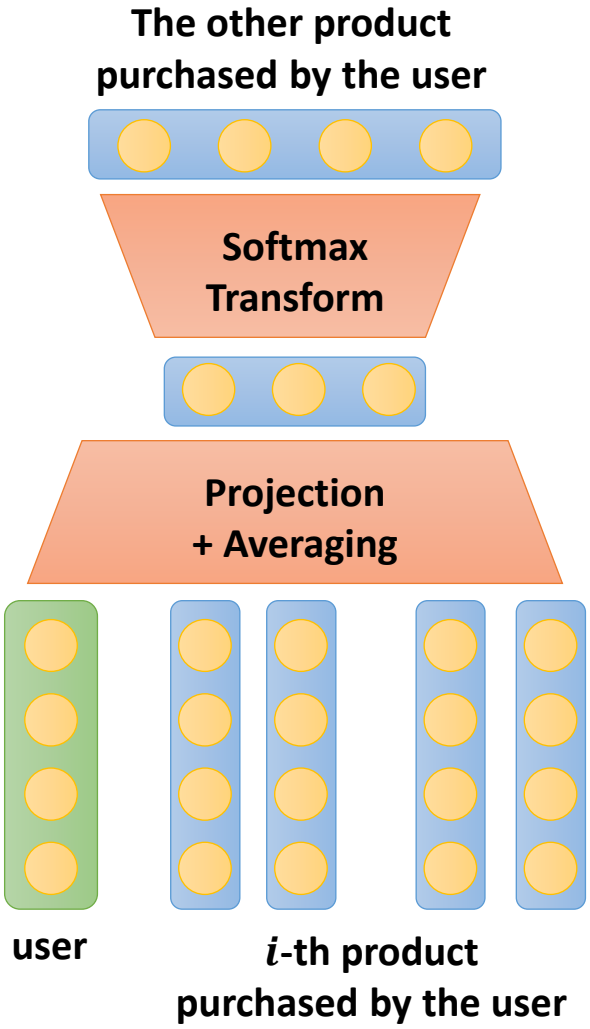
- **Imagine that the existing user-item matrix.**
  - Word → movie, a set of words → user
  - The window size is ignored.



Possible Extension of Prod2Vec



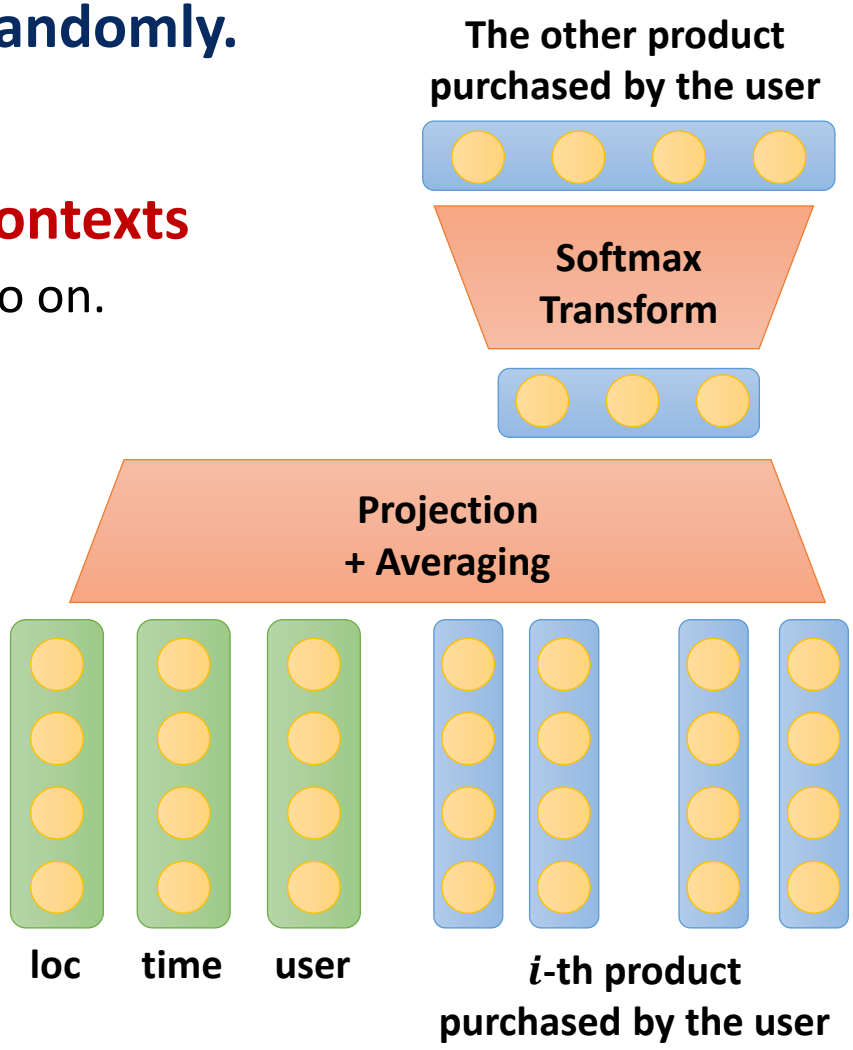
Prod2Vec skip-gram model



User embedding + Prod2Vec

Extensions of ProdVec2

- **Initialize user embeddings randomly.**
- **Possible to consider other contexts**
  - Location, time, country, and so on.



Co-occurrence-based Recommendation

Theoretical Analysis for Word2Vec

- **Pointwise mutual information (PMI) is a measure of association between two events.**

- **Some of the neural methods, e.g., Word2Vec, are shown to implicitly approximate the factorization of a (shifted) point-wise mutual information (PMI) matrix.**

- **Shifted PPMI (SPPMI) approximates Word2Vec.**
  - t: target word, c: context word, k: window size

$$SPPMI_k(t,c) = \max(PMI(t,c) - \log k\,, 0)$$

Omer Levy, Yoav Goldberg, *Neural word embedding as implicit matrix factorization*, NIPS 2014
Omer Levy, Yoav Goldberg, Ido Dagan, *Improving Distributional Similarity with Lessons Learned from Word Embeddings*, Trans. Assoc. Comput. Linguistics, 2015

Co-occurrence Matrix

- Define a **basis vocabulary** C of context words.
- Define a **word window** size w.
- **Count the basis vocabulary words** occurring w words to the left or right of each instance of a target word.
- Form a **vector representation** of the target word based on these counts.

Co-occurrence Matrix

- **Example corpus (window size = 1)**
  - I like deep systems.
  - I like game.
  - I enjoy studying.

	I	like	enjoy	game	studying	deep	systems
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	1	0	0
game	0	1	0	0	0	0	0
studying	0	0	1	0	0	0	0
deep	0	1	0	0	0	0	1
systems	0	0	0	0	0	1	0

Co-occurrence Matrix

- **Use a similarity measure, e.g., inner product or cosine.**
  - $like = [2, 0, 0, 1, 0, 1, 0]$
  - $enjoy = [1, 0, 0, 0, 1, 0, 0]$
  - $game = [0, 1, 0, 0, 0, 0, 0]$

- **Reminder:** $\boldsymbol{cosine(u, v) = \frac{u \cdot v}{||u|| \times ||v||}}$
  - $cosine(like, enjoy) \approx 0.71$
  - $cosine(like, game) \approx 0.00$

- **Not all features are equal.**
  - We must **distinguish counts** to represent more **informative** words.
  - Normalization methods: TF-IDF, PMI, etc.

Pointwise Mutual Information

- Do two events x and y co-occur more than if they were independent?

$$\text{PMI}(x, y) = \log_2 \frac{p(x, y)}{p(x)p(y)}$$

- PMI between two words: Do two words x and y co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{p(word_1, word_2)}{p(word_1)p(word_2)} = \log_2 \frac{count(word_1, word_2) \cdot |D|}{count(word_1)count(word_2)}$$

$|D|$: # of entire documents

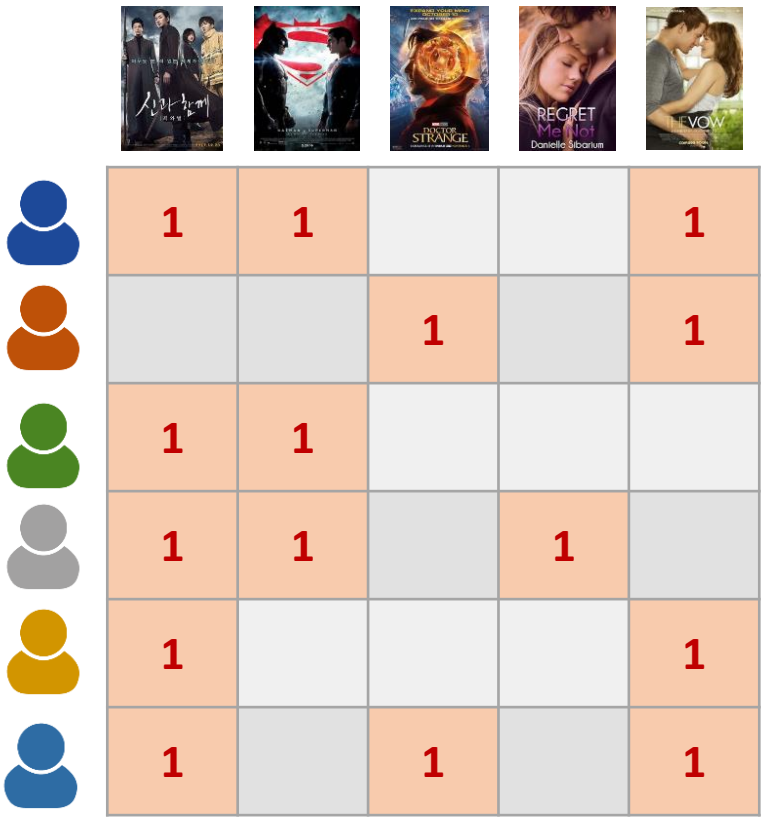
Positive Pointwise Mutual Information

- **PMI ranges from $-\infty$ to $+\infty$.**
  - The negative values are problematic.
  - Things are co-occurring **less than** we expect by chance.
- **Unreliable without enormous corpora**
  - Imagine w_1 and w_2 whose probability is each 10^{-6}
  - Hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}.
  - It is not clear people are good at **unrelatedness**.
- **Replace negative PMI values by 0: Positive PMI (PPMI) between $word_1$ and $word_2$**

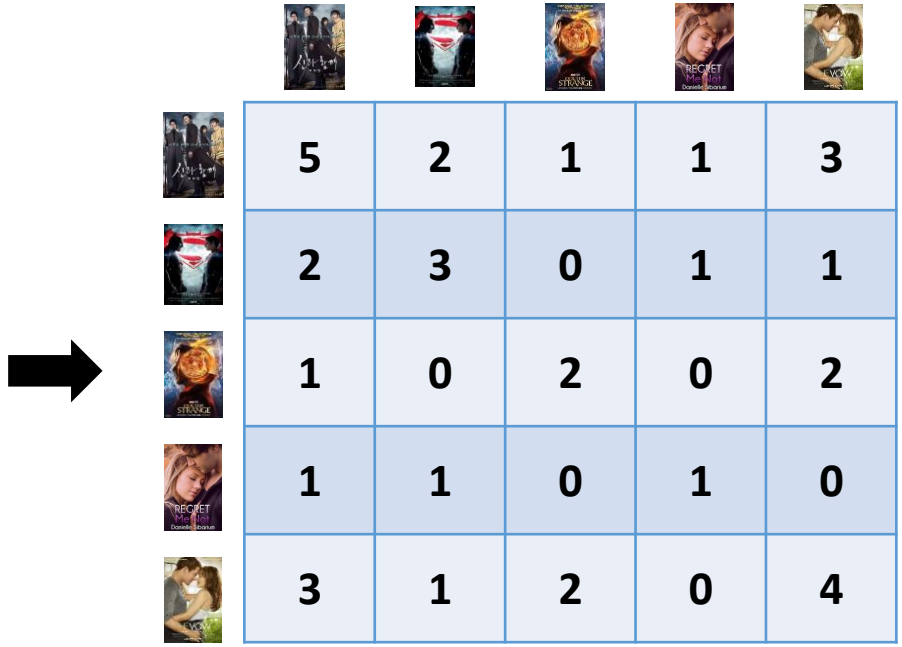
$$\text{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{p(word_1, word_2)}{p(word_1)p(word_2)}, 0\right)$$

Example: Co-occurrence for Items

- Building the co-occurrence matrix for items



1	1			1
		1		1
1	1			
1	1		1	
1				1
1		1		1



5	2	1	1	3
2	3	0	1	1
1	0	2	0	2
1	1	0	1	0
3	1	2	0	4

Example: PMI for Items

- **Normalize the values by the number of users.**
  - **Diagonal** elements: the probability for **each item $p(i)$**
  - **Non-diagonal** elements: the probability for **pairwise items $p(i, j)$**

					
	4	2	1	1	3
	2	3	0	1	1
	1	0	2	0	2
	1	1	0	1	0
	3	1	2	0	4

					
	4/6	2/6	1/6	1/6	3/6
	2/6	3/6	0/6	1/6	1/6
	1/6	0/6	2/6	0/6	2/6
	1/6	1/6	0/6	1/6	0/6
	3/6	1/6	2/6	0/6	4/6

Example: PMI for Items

➢ How to compute PMI?

PMI( , ) $= \log_2 \frac{2/6}{4/6 * 3/6}$

PMI( , ) $= \log_2 \frac{1/6}{4/6 * 2/6}$

PMI( , ) $= \log_2 \frac{1/6}{4/6 * 1/6}$

PMI( , ) $= \log_2 \frac{3/6}{4/6 * 4/6}$

	4/6	2/6	1/6	1/6	3/6
	2/6	3/6	0/6	1/6	1/6
	1/6	0/6	2/6	0/6	2/6
	1/6	1/6	0/6	1/6	0/6
	3/6	1/6	2/6	0/6	4/6

About Time (2013)

드라마/판타지/성장

어떠한 순간을 다시 살게 된다면, 과연 완벽한 사랑을 이룰 수 있을까?

모태솔로 팀은 성인이 된 날, 아비저로부터 놀랄만한 가문의 비밀을 듣게 된다. 바로 사긴을 되돌릴 수 있는 능력이 있다는 것! 여자친구를 만든다는 꿈을 이루기 위해 런던으로 간 팀은 메리에게 첫눈에 반하게 되는데..

PMI

What If (2013)
로맨스/코미디

Man Up (2015)
드라마/로맨스

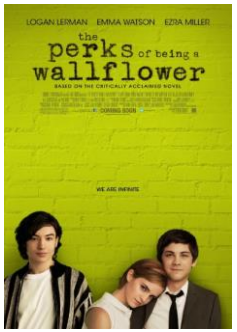
Love, Rosie (2014)
로맨스/코미디

Two Night Stand (2014)
로맨스/코미디

GloVe

Silver Linings Playbook (2012)
드라마/코미디

Secret Life of Walter Mitty, The (2013)
판타지/드라마



Perks of Being a Wallflower, The (2012)
드라마/성장

The Theory of Everything (2014)
드라마/로맨스

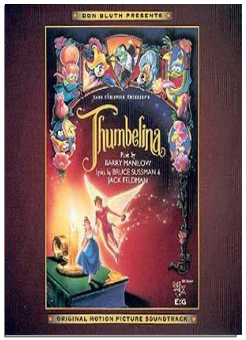
Mulan (1998)

애니메이션

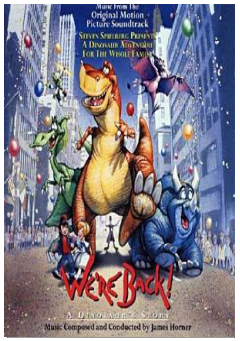
동양의 분위기를 살린 디즈니의 역작!

파씨 가문의 외동딸 뮬란은 선머슴 같은 성격 때문에 중매를 볼 때마다 퇴짜를 맞는다. 때마침 훈족의 침입으로 징집명령이 떨어지고 늙은 아버지를 대신하여 남장을 하고 나서는데..

PMI



Thumbelina (1994)
애니메이션

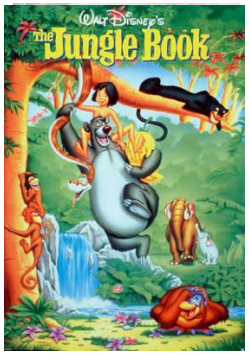


A Dinosaur's Story (1993)
애니메이션

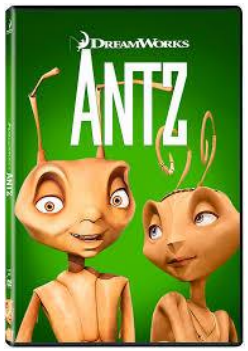
Quest for Camelot (1998)
애니메이션

Return to Never Land (2002)
애니메이션

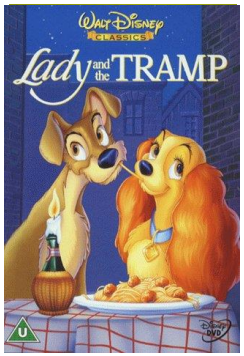
GloVe



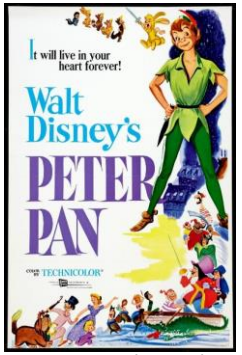
Jungle Book, The (1967)
애니메이션



Antz (1998)
애니메이션



Lady and the Tramp (1955)
애니메이션



Peter Pan (1953)
애니메이션

Machine, The (2013)

판타지/SF(로봇)

인간과 로봇 그 경계가 사라진다!

신 냉전시대에 인간의 뇌 데이터를 바탕으로 탄생한 살인로봇 머신 에이바는 점차 인간의 감정을 느껴가고, 그녀를 주축으로 머신들은 인간과의 최후의 전쟁을 선포하는데...

PMI

Signal, The (2014)
스릴러/SF (컴퓨터)

Autómata (2014)
스릴러/액션(로봇)

the east(2013)
스릴러/액션(스파이)

Cargo (2009)
스릴러/미스터리(우주)

GloVe

Signal, The (2014)
스릴러/SF (컴퓨터)

Zero Theorem, The (2013)
판타지/드라마(컴퓨터)

Autómata (2014)
스릴러/액션(로봇)

Cargo (2009)
스릴러/미스터리(우주)

StarSpace: Embed All The Things!

Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, Jason Weston, *StarSpace: Embed All The Things!*, AAAI 2017

What is StarSpace?

- **A general-purpose neural model for efficient learning of entity embeddings for solving a wide variety of problems**
  - **Classification**: embed document entities and label entities
  - **Information retrieval**: Ranking of sets of entities, ranking web documents for a query
  - **Recommendation**: embed user entities and item entities
  - **Embedding graphs**, e.g., multi-relational graphs such as Freebase
  - Learning word, sentence or document embeddings

Motivation: Embedding Different Types

- **How to embed documents and tags?**

$D1$: This food is delicious! *#positive*

$D2$: Lovely veggie pizza. *#pizza #positive #vegetarian*

$D3$: Cheap Korean restaurant *#Korean*

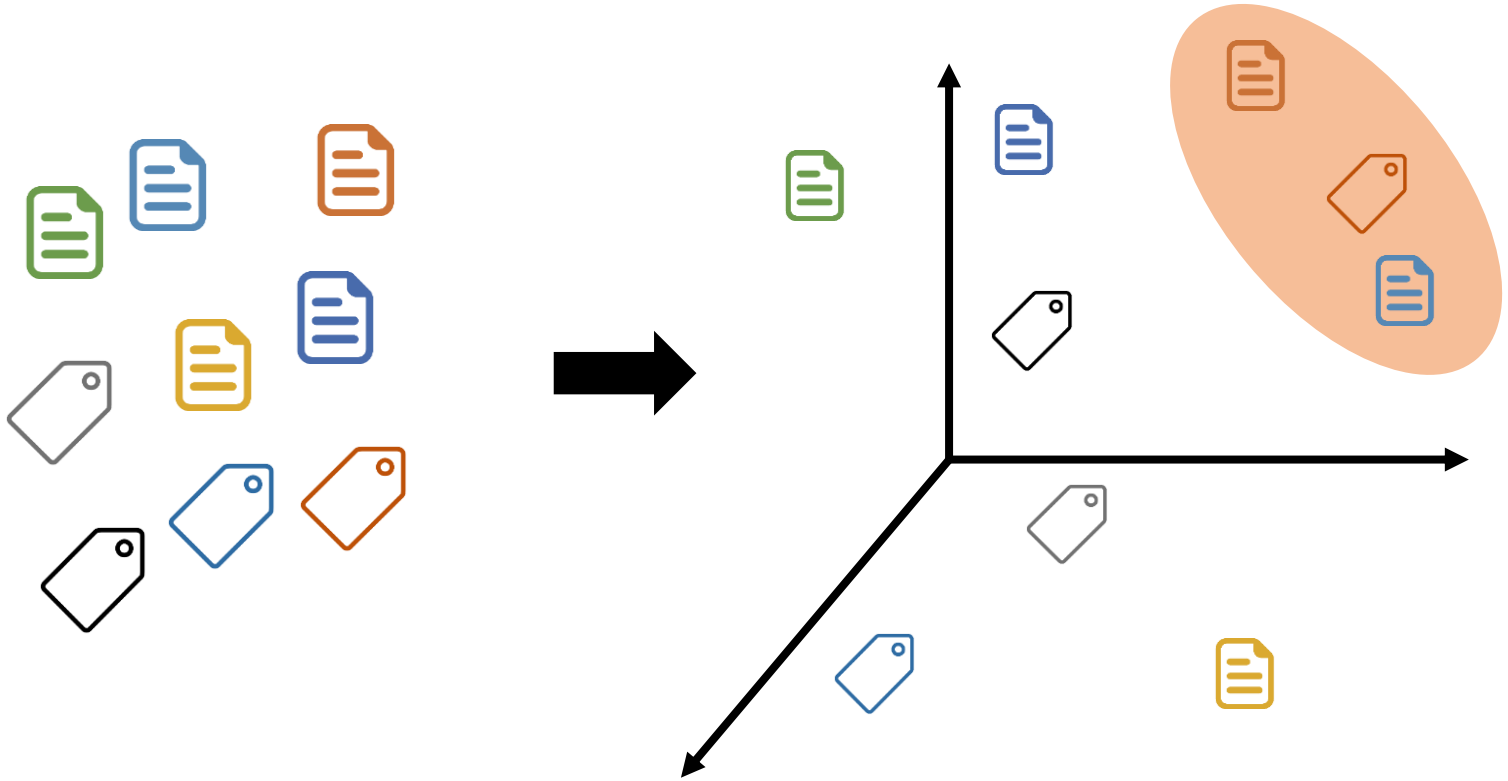
$D4$: I like a vegetarian place. *#vegetarian #positive*

$D5$: Bad pepperoni pizza *#pizza #negative*

	#positive	#negative	#pizza	#vegetarian	#Korean
D1	1				
D2	1		1	1	
D3					1
D4	1			1	
D5		1	1		

Motivation: Embedding Different Types

- How to embed documents and tags?



How to Embed Everything?

- Each entity is represented by a set of discrete features.

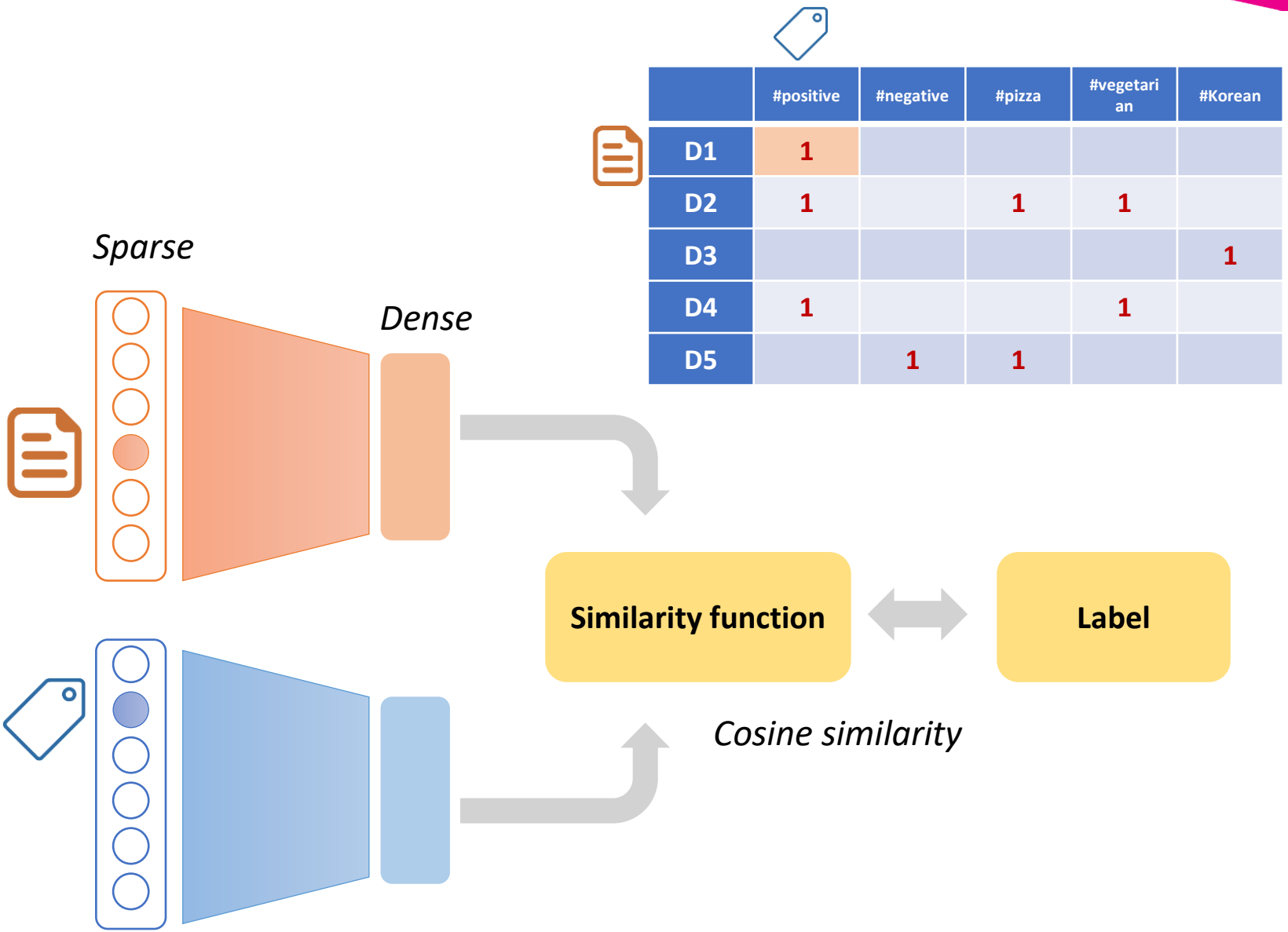
$$e = \sum_{i \in e} F_i$$

- Examples

Vec() = Vec() + Vec() + Vec()

Vec() = Vec() + Vec() + Vec() + Vec()

Learning StarSpace



Learning StarSpace

➢ Learning to compare entities by minimizing the loss function

$$\sum_{(a,b)\in E^+, b^-\in E^-} L^{batch}(sim(a,b), sim(a,b_1^-), \dots, sim(a,b_k^-))$$

- The generator of positive entity pairs (a, b) combing from the set E^+
- The generator of negative entities b_i^- coming from the set E^-
  - k-negative sampling is used to select k negative pairs for each batch size.
- Loss function: the **margin ranking loss** and **negative log loss of softmax**.

➢ Example

- For text classification, a are **documents** and b are **labels**.
- We sample b^- from the set of labels.

Tag Embeddings

- D_1: I like a vegetarian place. *#vegetarian #positive*
- E^+: {(D_1, *#vegetarian*), (D_1, *#positive*)}
- E^-: {*#negative*, *#pizza*}
- Generate negative pairs {(D_1, *#negative*), (D_1, *#pizza*)}.

Tag Embeddings

- E^+: {('Restaurant has great food, #yum), ('Restaurant has great food, #restaurant)}
- E^-: {'#animals', '#cityvbafc', …}

a: sentence	b^+: positive labels	b^-: sampled negative labels
Restaurant has great food	#yum #restaurant	#animals # cityvbafc …

Text Classification

- E^+: {(Baseball is…, Sports)}
- E^-: {Business, Science, Economy, …}

a: document	b^+: positive labels	b^-: sampled negative labels
Baseball is a bat-and-ball game played between two opposing teams who take turns batting and fielding.	Sports	Business, Science, Economy, …

CF-based Recommendation

- E^+: {(Bob, computer), (Bob, jean)}
- E^-: {Guitar, Watch, Jacket, Baseball, …}

a: user

b^+: positive labels

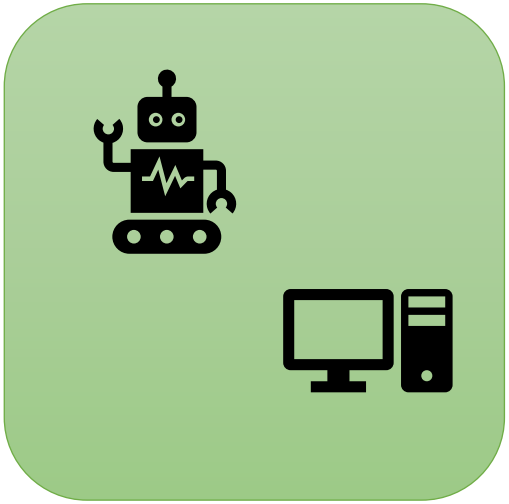
b^-: sampled negative labels

- Impossible to recommend items for new users

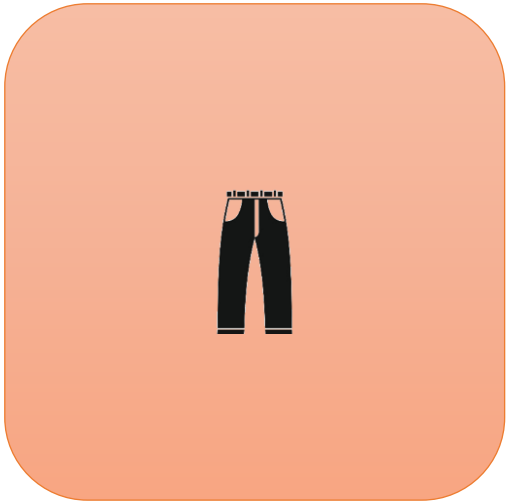
CF with Out-of-Sample Users

- User = sum of items embeddings
- Example: Bob = {Robot, computer, jean}

a: items except for a positive item

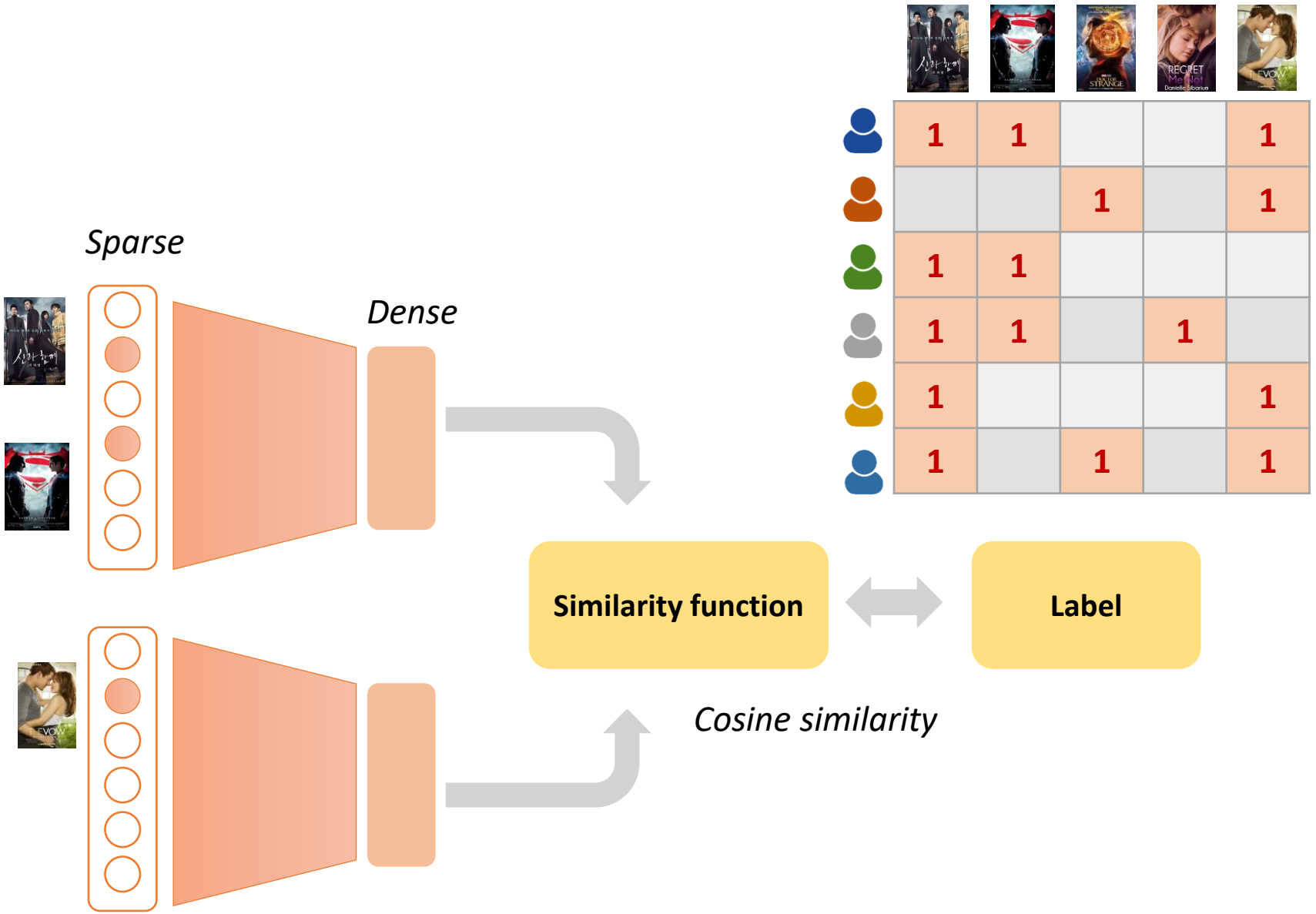


b^+: positive labels



b^-: sampled negative labels

CF with Out-of-Sample Users



Experiments: Text Classification

- **Three text datasets**
  - AG news: 4 class text classification task given title and descriptions
  - DBpedia: 14 class given the title and abstract of articles
  - Yelp review: 1-5 star ratings for review

Model	AG news	DBpedia	Yelp15
BoW*	88.8	96.6	-
ngrams*	92.0	98.6	-
ngrams TFIDF*	92.4	98.7	-
char-CNN*	87.2	98.3	-
char-CRNN⋆	91.4	98.6	-
VDCNN◇	91.3	98.7	-
SVM+TF†	-	-	62.4
CNN†	-	-	61.5
Conv-GRNN†	-	-	66.0
LSTM-GRNN†	-	-	67.6
fastText (ngrams=1)‡	91.5	98.1	**62.2
StarSpace (ngrams=1)	91.6	98.3	62.4
fastText (ngrams=2)‡	92.5	98.6	-
StarSpace (ngrams=2)	92.7	98.6	-
fastText (ngrams=5)‡	-	-	66.6
StarSpace (ngrams=5)	-	-	65.3

Experiments: Article Recommendation

- For 641,385 people, we collected public articles that the user clicked to read, giving a total of 3,119,909 articles.

- Recommending new documents to a user given their past history of liked documents

Metric	Hits@1	Hits@10	Hits@20	Mean Rank	Training Time
Unsupervised methods					
TFIDF	0.97%	3.3%	4.3%	3921.9	-
word2vec	0.5%	1.2%	1.7%	4161.3	-
fastText (public Wikipedia model)	0.5%	1.7%	2.5%	4154.4	-
fastText (our dataset)	0.79%	2.5%	3.7%	3910.9	4h30m
Tagspace†	1.1%	2.7%	4.1%	3455.6	-
Supervised methods					
SVM Ranker: BoW features	0.99%	3.3%	4.6%	2440.1	-
SVM Ranker: fastText features (our dataset)	0.92%	3.3%	4.2%	3833.8	-
StarSpace	3.1%	12.6%	17.6%	1704.2	12h18m

Q&A

References

Association-based recommendation

- Malte Ludewig and Dietmar Jannach, *Evaluation of Session-based Recommendation Algorithms*, 2018

Embedding-based recommendation

- Mihajlo Grbovic et al., *E-commerce in Your Inbox: Product Recommendations at Scale*, KDD 2015
- Flavian Vasile et. al., *Meta-Prod2Vec: Product Embeddings using Side-Information for Recommendation*, RecSys 2016
- Ledell Wu et al., *StarSpace: Embed All The Things!*, AAAI 2017