

# Word Embeddings

# Distributed and Contextual Representation

# One-hot Vector

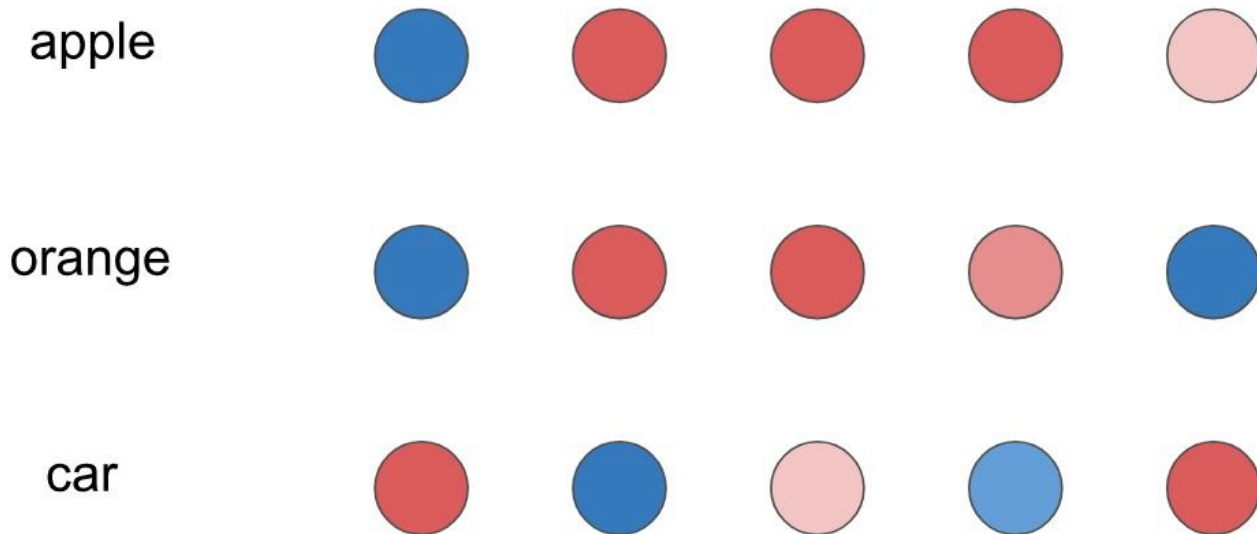
apple [0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]

orange [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 ... 0 0 0 0 0 0]

car [0 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]



# Distributed Representation



**inhibited**



**excited**

# Contextual Representation

- Words is represented by their context.



I eat an **apple** every day.



I eat an **orange** every day.

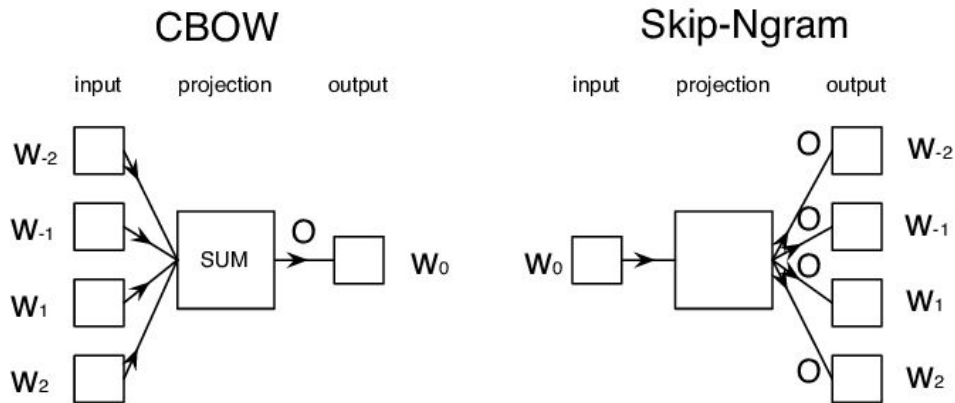


I like driving my **car** to work.

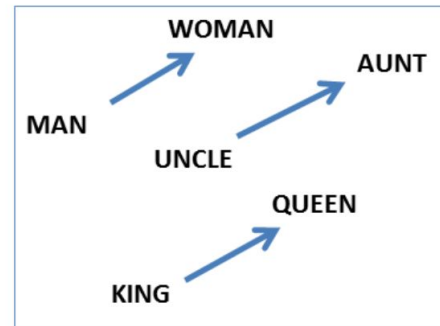
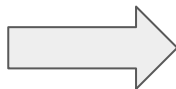
**What is Word2Vec?**

# Word2Vec

- A method of computing vector representation of words developed by Google.
- Open-source version of Word2Vec hosted by Google (in C)
- Train a simple neural network with a single hidden layer to perform word prediction tasks
- Two structures proposed **cbow** vs **skip-gram**:



# Word2Vec as BlackBox



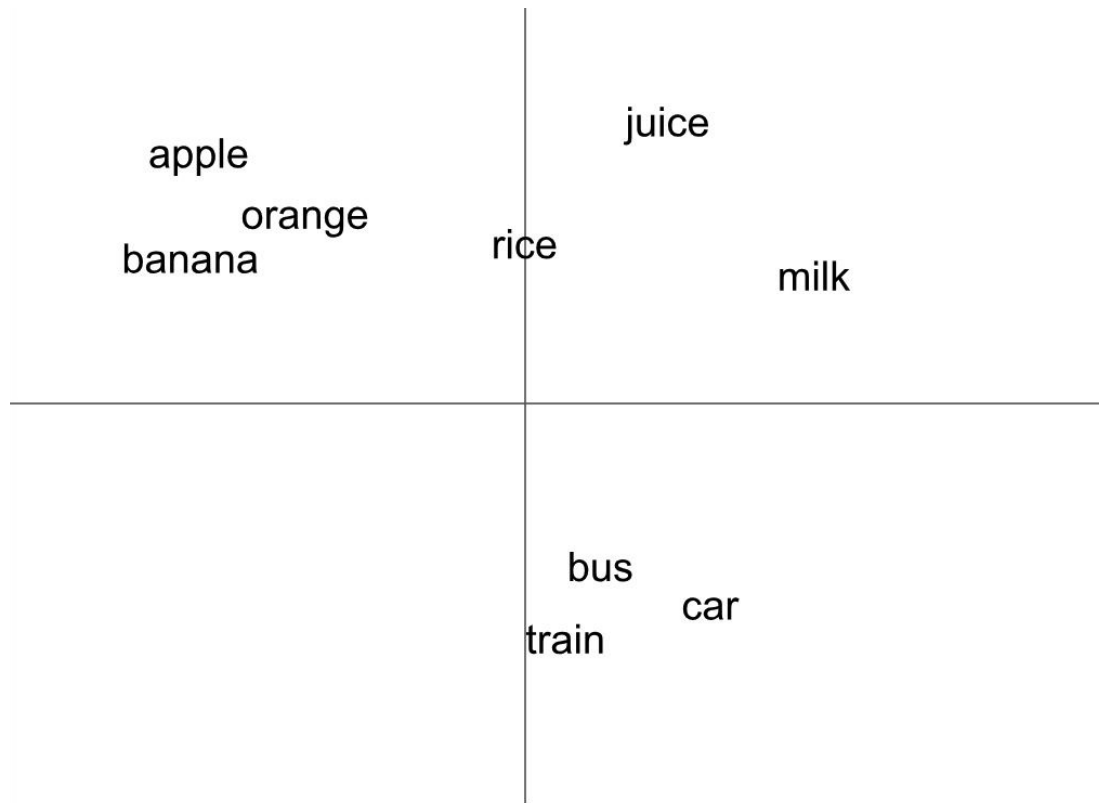
**Corpus**

**Word2Vec Tool**

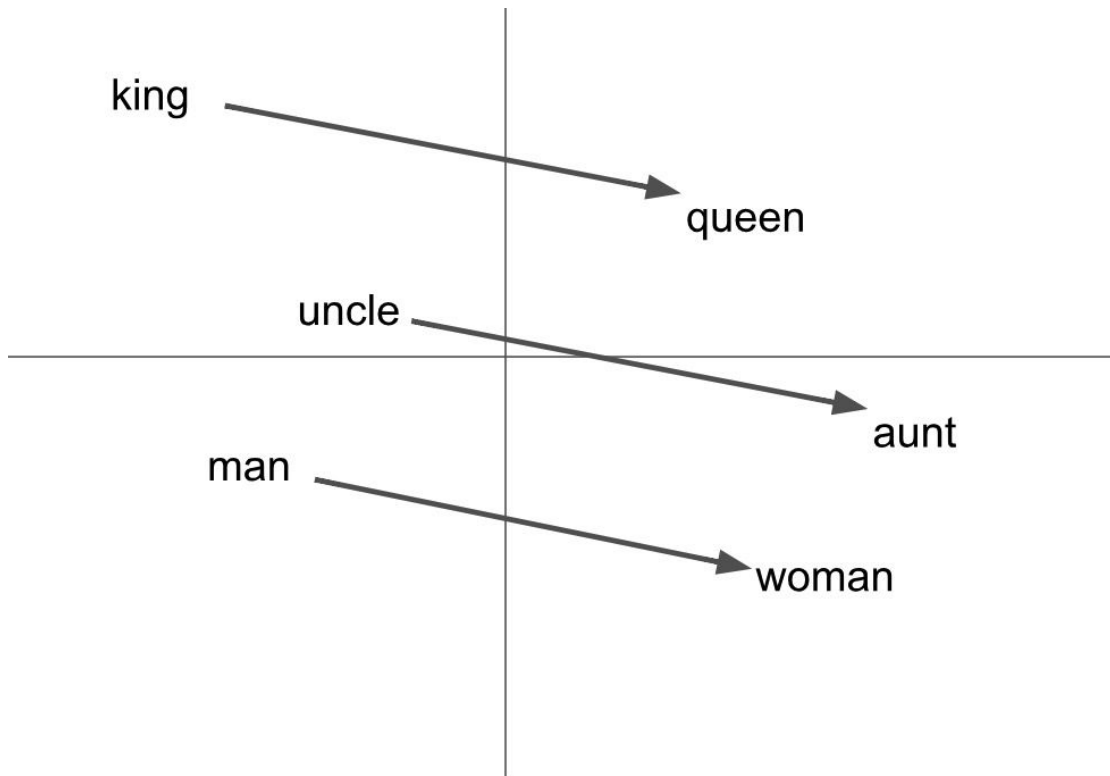
**Word Embeddings**



# Word Vectors



# Word Analogy

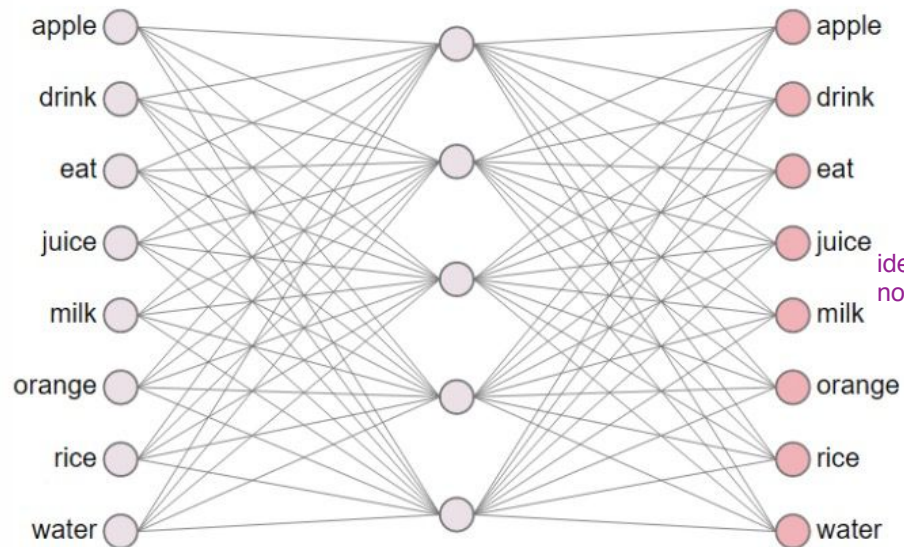


Mikolov & Chen et al. 2013  
Mikolov & Sutskever et al. 2013

# A Good Visualization for Word2Vec

<https://ronxin.github.io/wevi/>

# Model Architecture



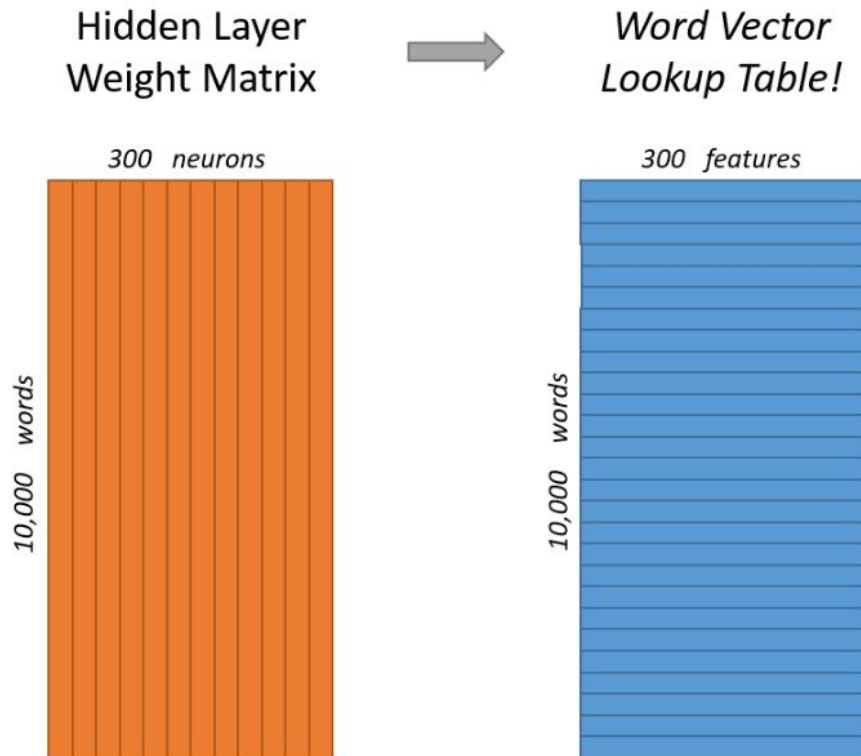
## Structure Highlights:

- input layer
  - one-hot vector
- hidden layer
  - linear (identity)
- output layer
  - softmax

# Hidden Layer

the matrix  $W$

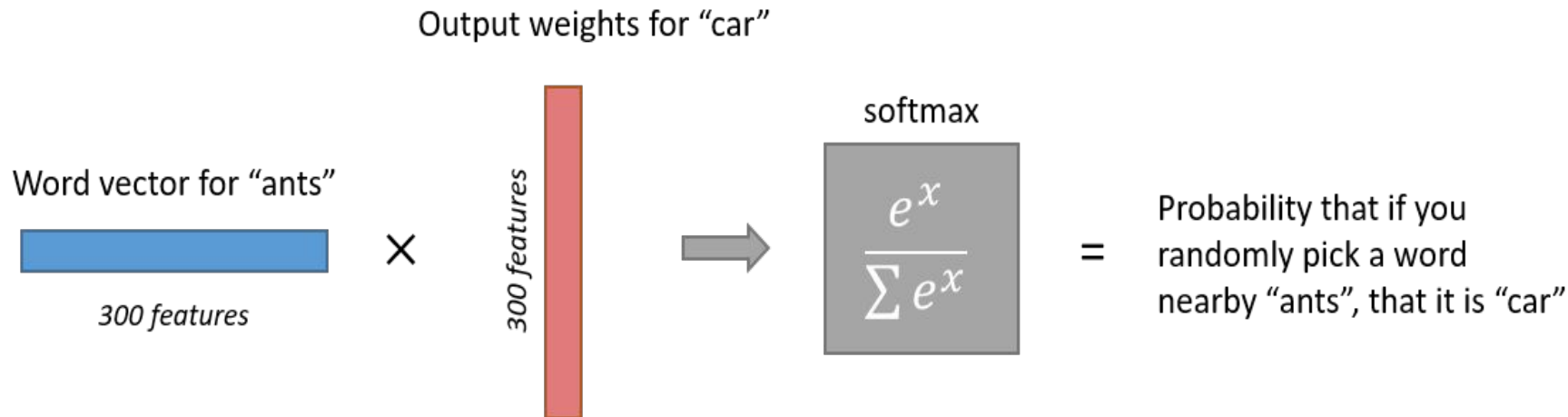
- There is not activation function here
- 300 neurons are the word vec. dimensions
- This layer is operating as a 'lookup' table
- Input word matrix



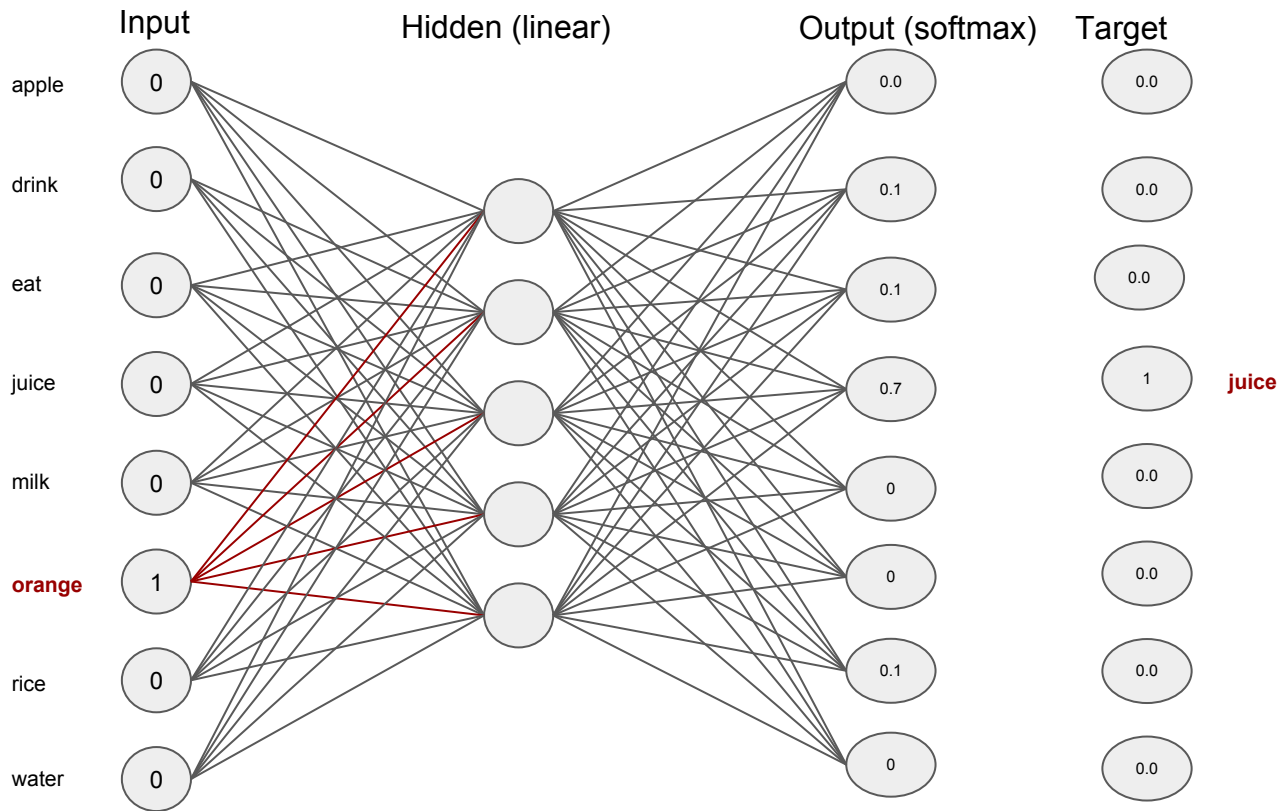
# Output Layer

the matrix  $W'$

- Softmax classifier
- Cross-entropy error can be used here
- Output word matrix

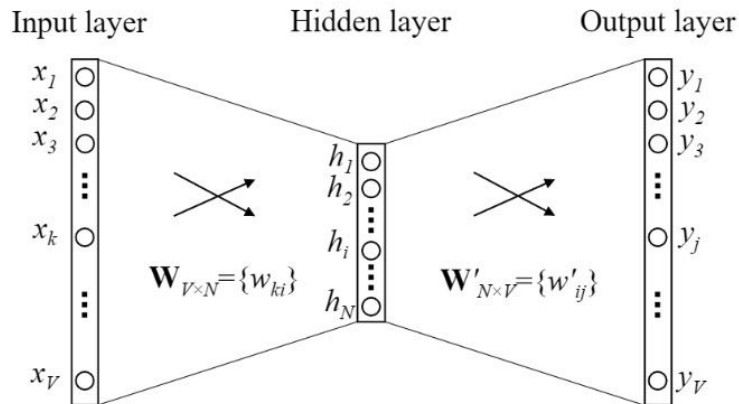
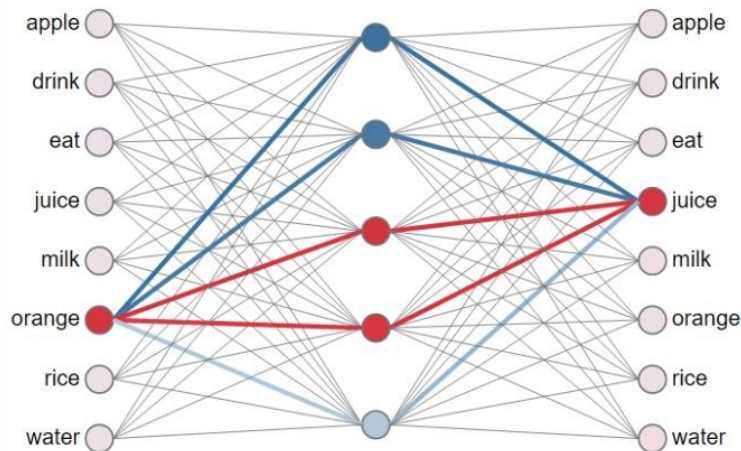


# Word2Vec Network



Then, we can compute the **loss** and call gradient descent to update model parameters.

# word2vec network



$$\mathbf{h} = \mathbf{x}^T \mathbf{W} := \mathbf{v}_{w_I}$$

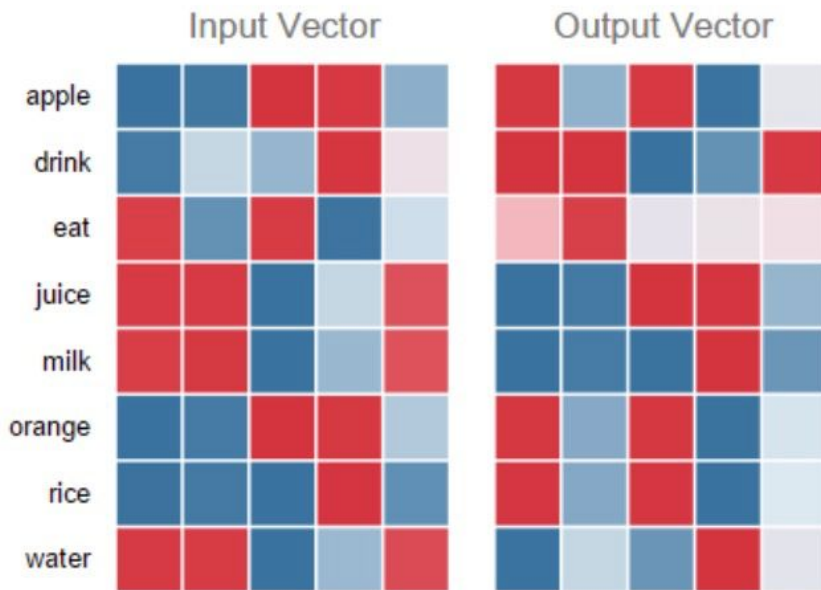
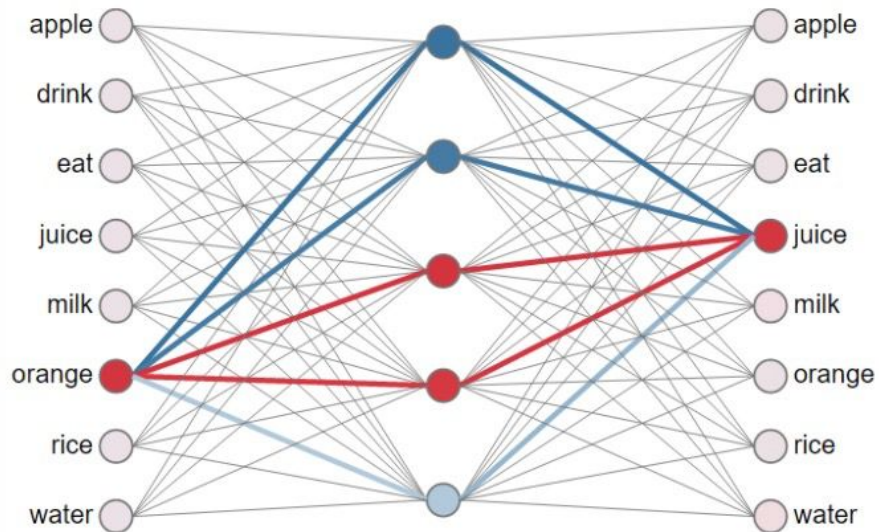
$$u_j = \mathbf{v}'_{w_j}{}^T \cdot \mathbf{h}$$

$$p(w_j | w_I) = \frac{\exp \left( \mathbf{v}'_{w_O}{}^T \mathbf{v}_{w_I} \right)}{\sum_{j'=1}^V \exp \left( \mathbf{v}'_{w'_j}{}^T \mathbf{v}_{w_I} \right)}$$

From Xin Rong 2016

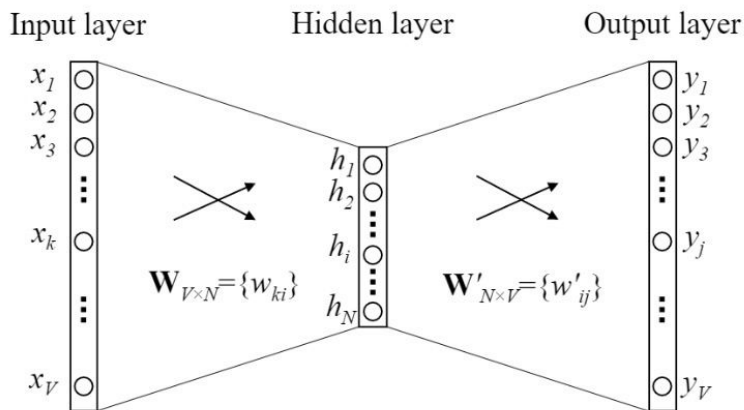
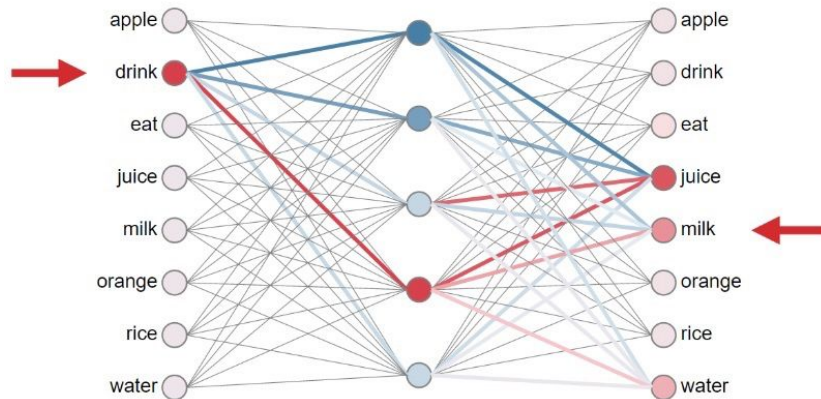


# Weight Matrix and Word Vectors



input for orange is similar to output for juice, so their dot pdt is large

# Training: updating word vectors



$$E = -\log \frac{\exp(\mathbf{v}'_{w_O}{}^T \mathbf{v}_{w_I})}{\sum_{j'=1}^V \exp(\mathbf{v}'_{w'_j}{}^T \mathbf{v}_{w_I})}$$

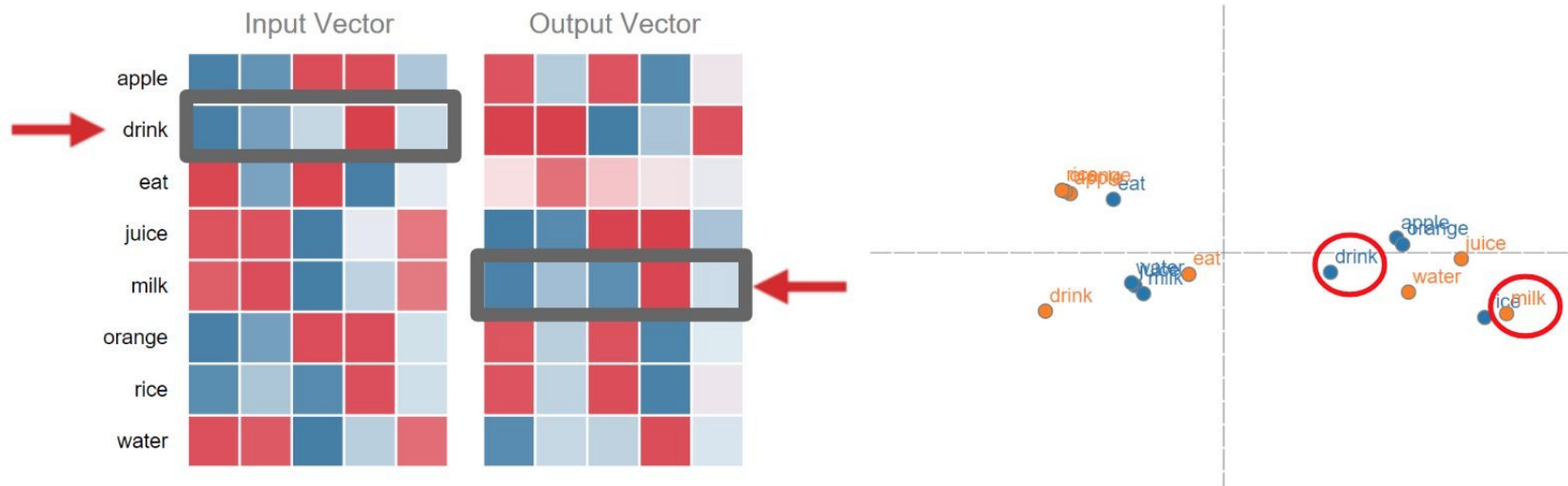
reduce the loss

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j$$

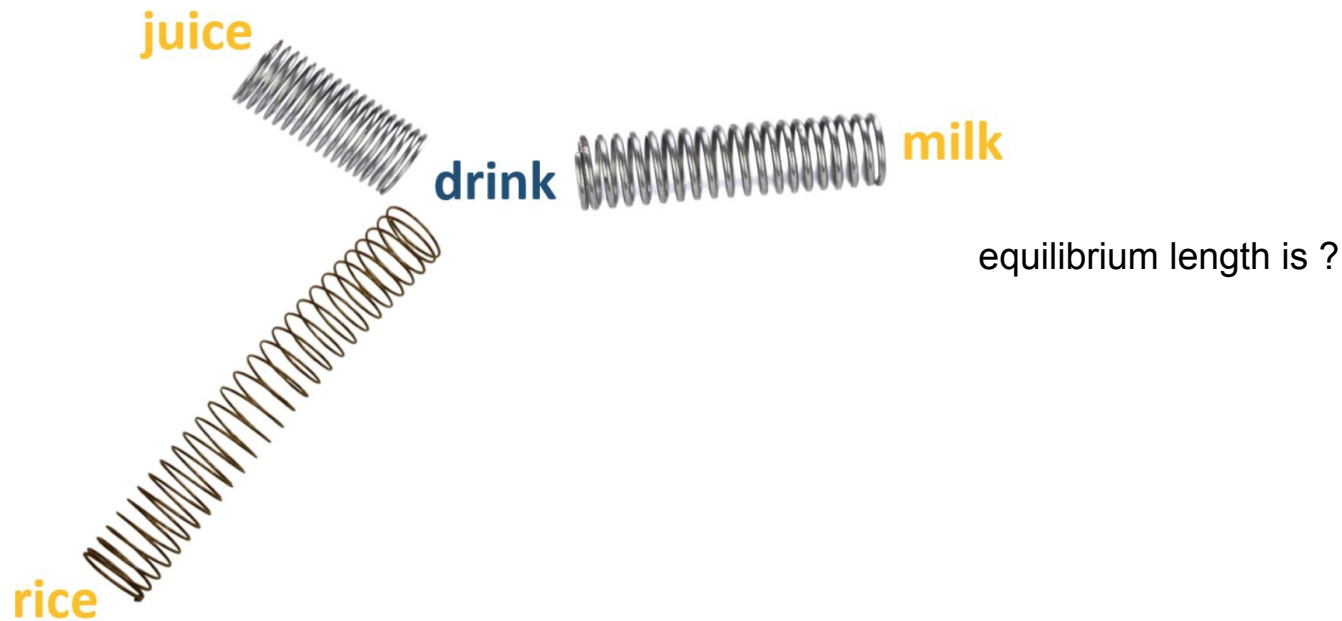
$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}}$$

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^V \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i}$$

# Updating Word Vectors



# A force-directed graph



# Idea behind Word2Vec

- Feature vector assigned to a word will be adjusted if it can not be used for accurate prediction of that word's context.
- Each word's context in the corpus is the teacher sending error signals back to modify the feature vector.
- It means that words with **similar context** will be assigned **similar vectors**!

## Distributional Semantics

# Input vs Output Word Vectors

- Inputs: semantics encoder from one-hot/word index to semantics
- Outputs: semantics decoder from semantics to probability distributions over words.
- In most cases, input word vectors are used. Some have observed that combinations of these two vectors may perform better.

Can also concatenate both  
input and output vectors

Question

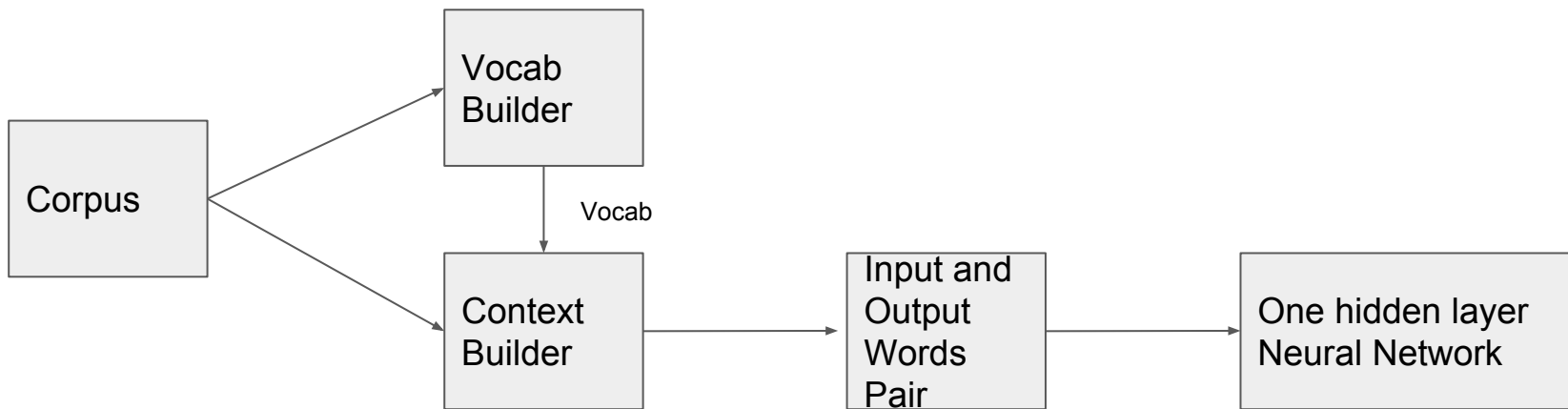
	Vector size	Overall	Semantic	Syntactic
DVRS	300	0.41	0.59	0.26
DVRS	1024	0.43	0.62	0.28
SG	300	<b>0.64</b>	<b>0.69</b>	<b>0.60</b>
SG	1024	0.57	0.60	0.55
Add 300-DVRS, 300-SG	300	0.64	0.72	0.58
Concatenate 300-DVRS, 300-SG	600	<b>0.67</b>	<b>0.74</b>	<b>0.60</b>
Add 1024-DVRS, 1024-SG	1024	0.60	0.66	0.55
Concatenate 1024-DVRS, 1024-SG	2048	0.61	0.68	0.55
Concatenate DVRS-1024, SG-300	1324	0.66	0.73	<b>0.60</b>
Oracle DVRS-1024, SG-300	1024/300	0.70	0.79	0.62

Garten, 2014

Table 2: Performance on word analogy problems with vectors trained against the first  $10^9$  bytes of Wikipedia.

# Input and Output Words

- How to select them from corpus
- **Skip-gram** and **CBoW** differs here. use centre word to predict the context



# Skip-Gram

- Task Definition: given a specific word, predict its nearby word (probability output)
- Model input: source word, Model output: nearby word
- Training data is a list of word pairs
- The output can be interpreted as prob. scores, which are regarded as that how likely it is find each vocabulary word can be nearby your input word.



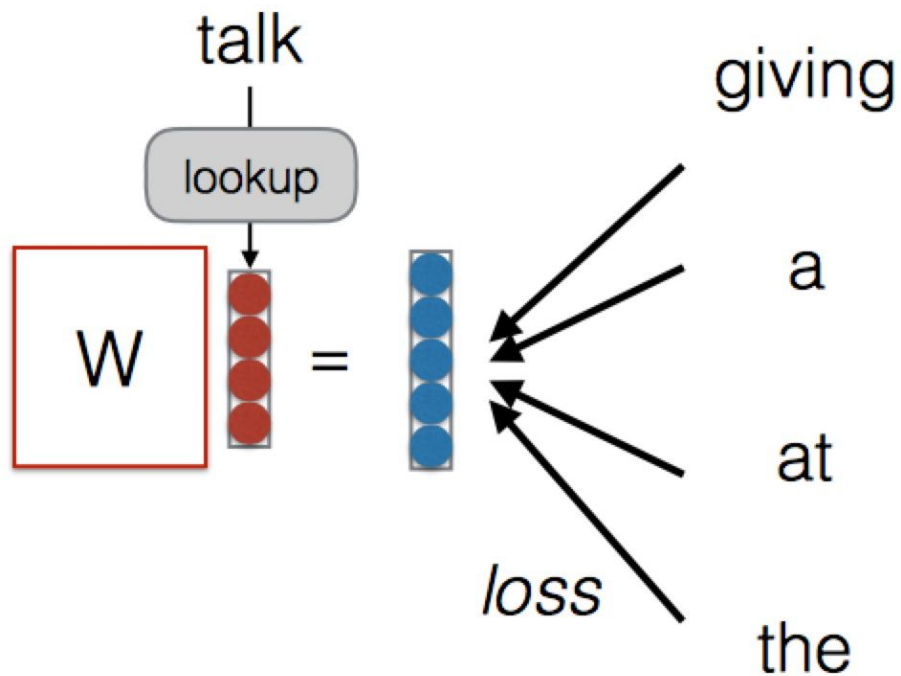
# Training Data Preparation

- Given a toy example: *The quick brown fox jumps over the lazy dog*
- Windows size: 2
- Left word is source word, and right word is target word

Source Text	Training Samples			
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)
The	quick	brown		
The <table><tr><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)
quick	brown	fox		
The quick <table><tr><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
brown	fox	jumps		
The quick brown <table><tr><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
fox	jumps	over		

# Skip-Gram

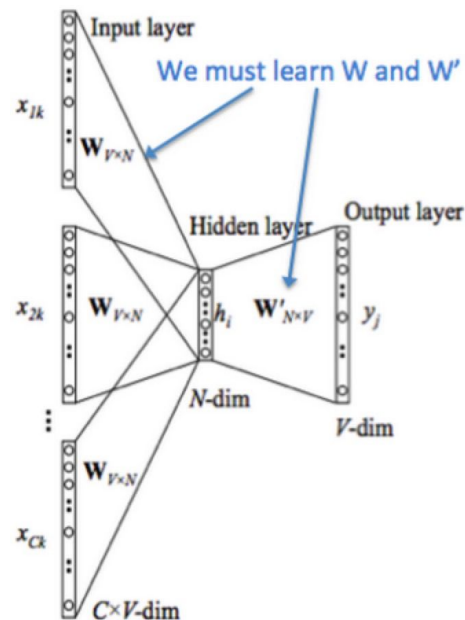
- Predict nearby word in the context given the word



# CBoW

use context to predict the centre word

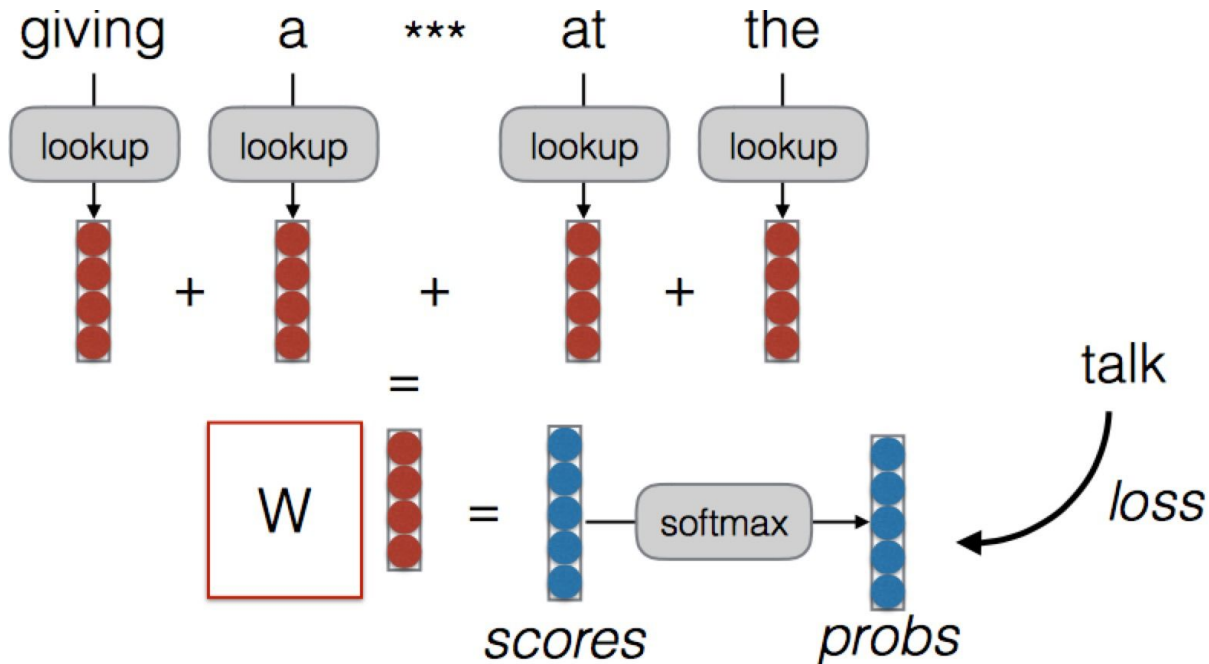
- Task Definition: given context, predict its target word
- Model input: context (several words), Model output: center word  
input is several words, can average the words
- Training data is a list of word pairs
- Core Trick: **average** these context vectors for prob score computing



# CBoW

skipped gram will give more training corpus giving the same window  
so will take a longer time than CBoW

- Predict word based on sum of embeddings behind nearby words



# Skip-Gram Vs CBoW

- **CBoW: learning to predict the word by the context**
- **Skip-gram: learning to predict the context by the center word**

CBoW: “giving a talk at the”  
Skipgram: (talk, giving),  
(talk, a), (talk, at), (talk, the)  
[i.e. generate more corpus]

a, at, the - these r noise, as  
the noise doesn't define  
“talk”, so CBoW works  
better for frequent words

Good choice for large amt of data

- CBoW: several times faster to train the skip-gram, slightly better accuracy for the frequent words
- Skip-gram: works well with small amount of the training data, represents well even rare words or phrases.      Cos u can get more corpus out for small amt of data

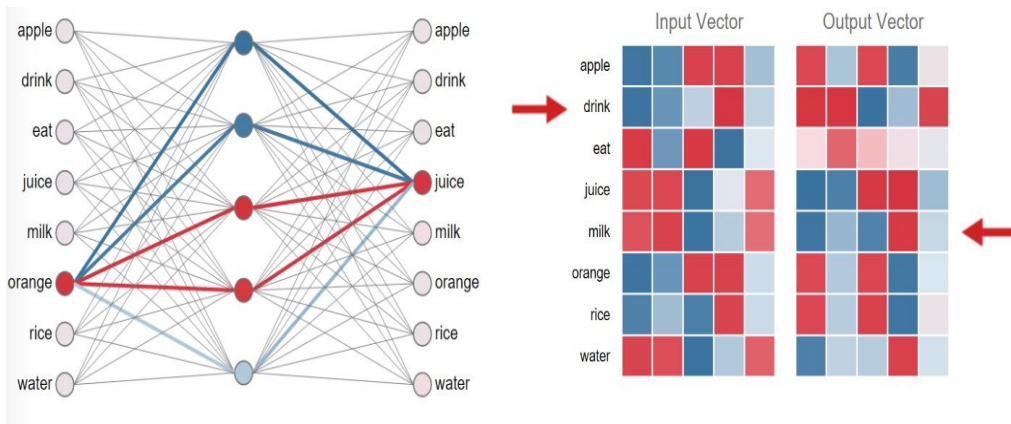
# Context Selection

- In count-based or predict-based methods, context has a large effect.
- Small context window: more syntax-based embeddings
- Large context window: more semantics-based, topical embeddings
- Engineering practice: window size is randomly sampled between 1 and maximum window size

# Huge Number of Parameters

- Vocab size is huge
- The Sum of operation in softmax layer is very expensive, i.e.,  $O(v)$ .
- Two solutions: **Hierarchical softmax** and **negative sampling**

negative sampling more efficient, just update the correct sample and eg. 5 of the incorrect sample



# How to Evaluate Word Embeddings?

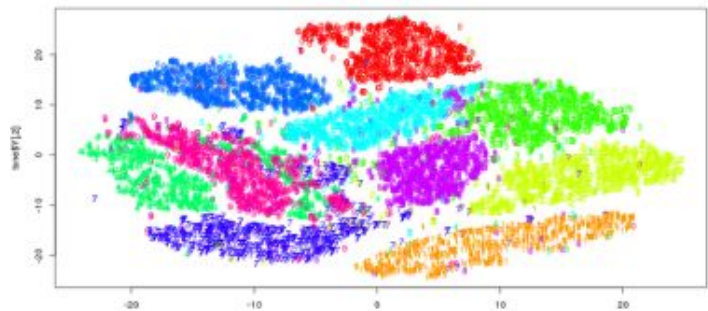


# Types of Evaluation

- Intrinsic vs Extrinsic:
  - Intrinsic: How good is it based on embeddings themselves?
  - Extrinsic: How useful is it downstream?
- Qualitative vs Quantitative:
  - Qualitative: Examine the characteristics of examples
  - Quantitative: Calculate statistics

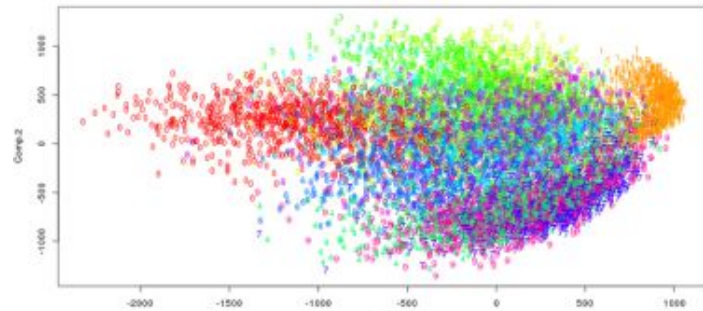
# Visualization of Word Embeddings

- Dimensionality Reduction: project word vectors into 2/3D for visualization
- T-SNE: group things that are close in high-dimensional space based on Gaussian



T-SNE

non-linear



PCA

linear

# Intrinsic Tasks

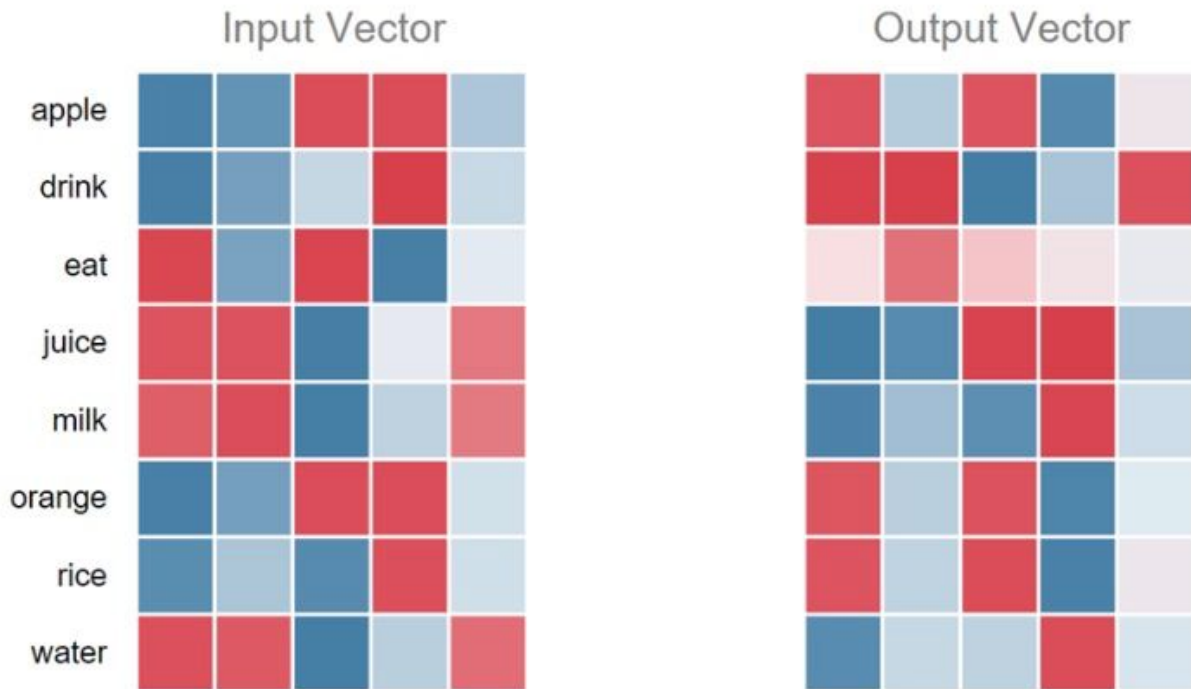
- Relatedness: the correlation btw. Embedding cosine similarity and human eval of similarity?
- Analogy: Find x for “ a is to b, as x is to y”
- Purity: create clusters based on the embeddings, and measure purity of clusters

# Extrinsic Tasks

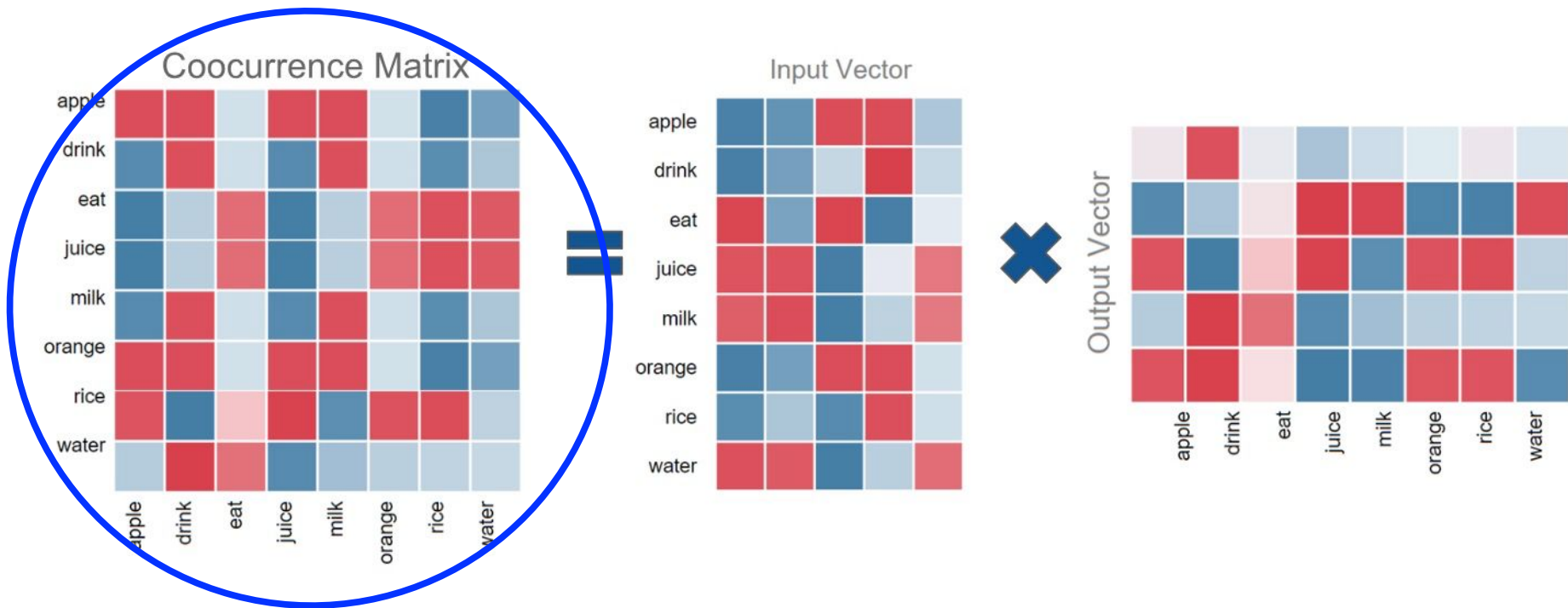
- Use word embeddings in external tasks to check the performances of the model
- Check which one has achieved the best performance

# Advanced Topics for Word Embeddings

# Interpreting word embedding model



# Interpreting word embedding model



# Usage of Pre-trained Word Embeddings

- Useful when training data is limited (pre-training from a large corpus)
- Very useful: tagging, parsing, text classification
- Less useful: machine translation
- Basically not useful: language modeling



# Limitations

- Word Ambiguity
- Lack of Explanation
- Sequence
- Can not handle OOV problem

# Sub-Word Embeddings 1

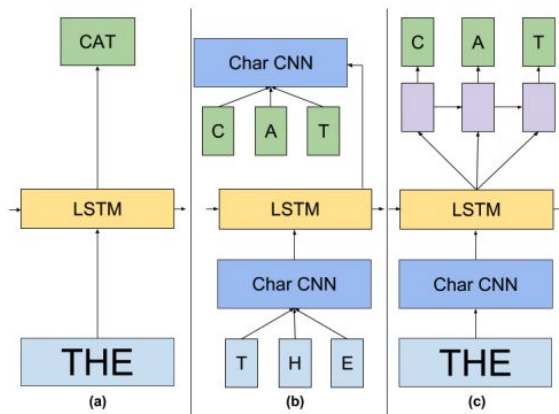
- Bag of character n-grams used to represent word

Where -> (wh, whe, here, ere, re)

- In 'fasttext' toolkit, this trick has been used
- From bag of character n-grams and adding them as final vectors

# Sub-Word Embeddings 2

- Start from char. Level
- For English, a list of character are defined 70 characters which including 26 English letters, 10 digits, 33 special characters and new line character
- Build upon neural network to learn the composition from char. Embeddings to word embeddings

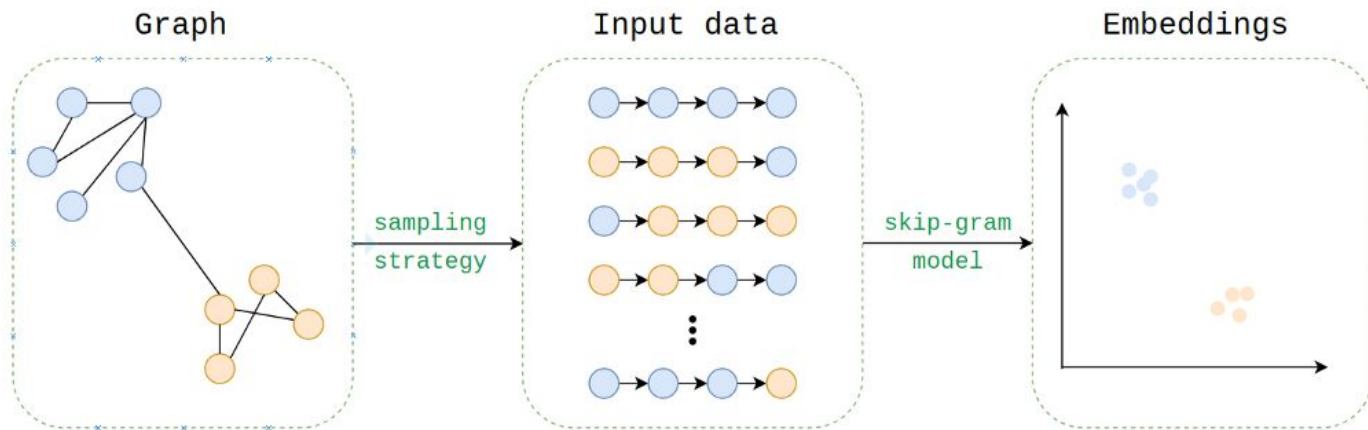


# Sub-Word Embeddings

- Can handle out-of-vocabulary words
- Work better for infrequent words
- Improve the training speed (less model parameters )

# Embedding for Everything

- Embeddings can be extended beyond NLP domain
- Embeddings can be learned for any nodes in a graph



- Node can be items, web page and so on in user clicked stream data
- Embeddings can be learned for any group of discrete and co-occurring states.

# Summary

- Neural network structure of word2vec is a feedforward network with one hidden layer with a linear activation function
- Large amount of text data provide solid co-occurrence word statics information
- Shallow and unsupervised Learning