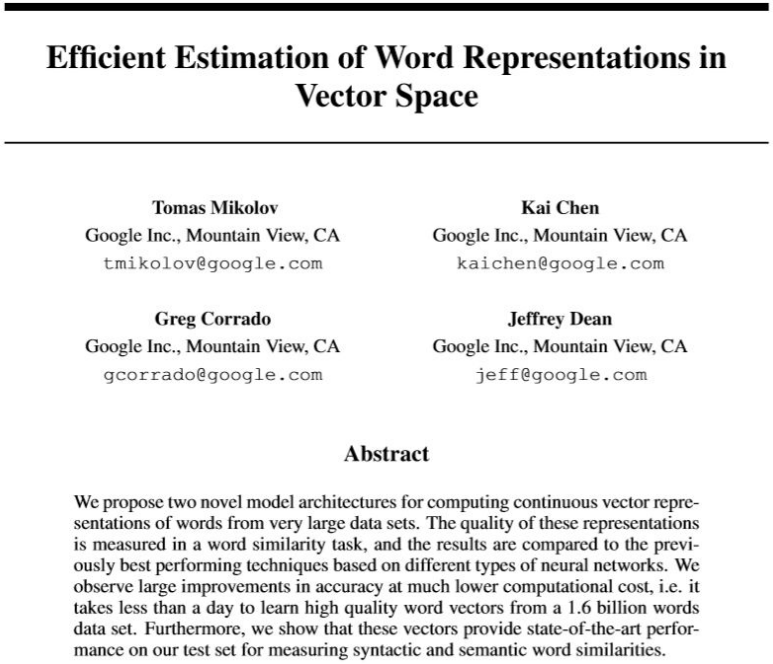
Word Embedding Explained and Visualized

Xin Rong
School of Information
University of Michigan

About word2vec...

Two original papers published in association with word2vec by Mikolov et al. (2013)



Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

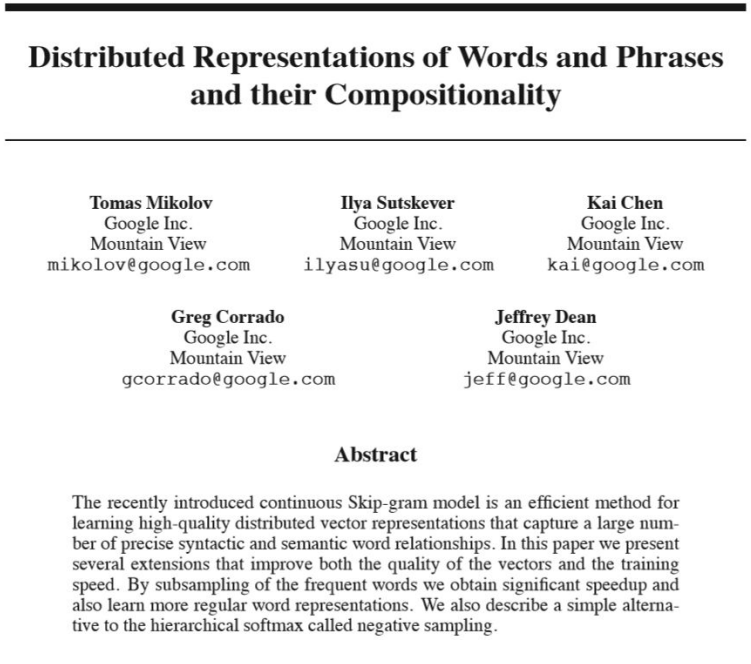
Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.



Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

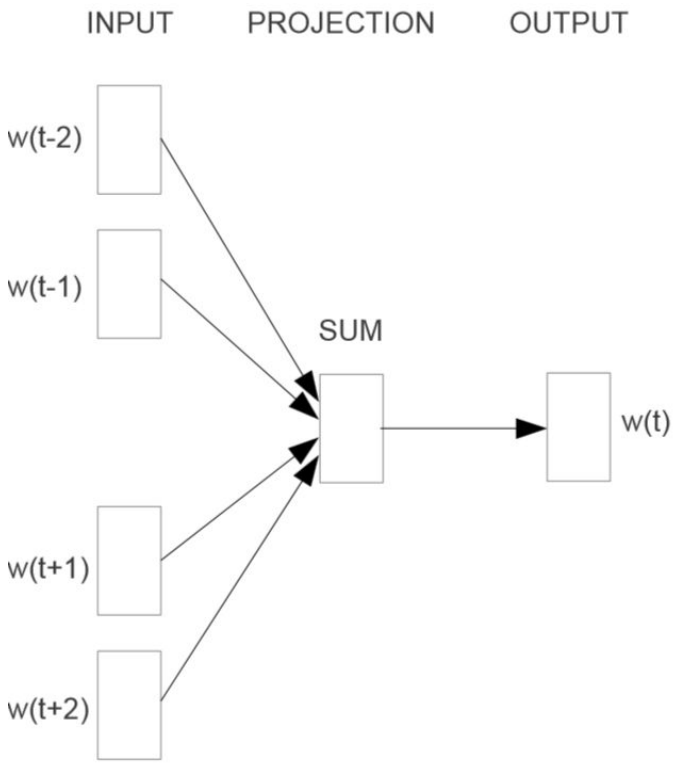
Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

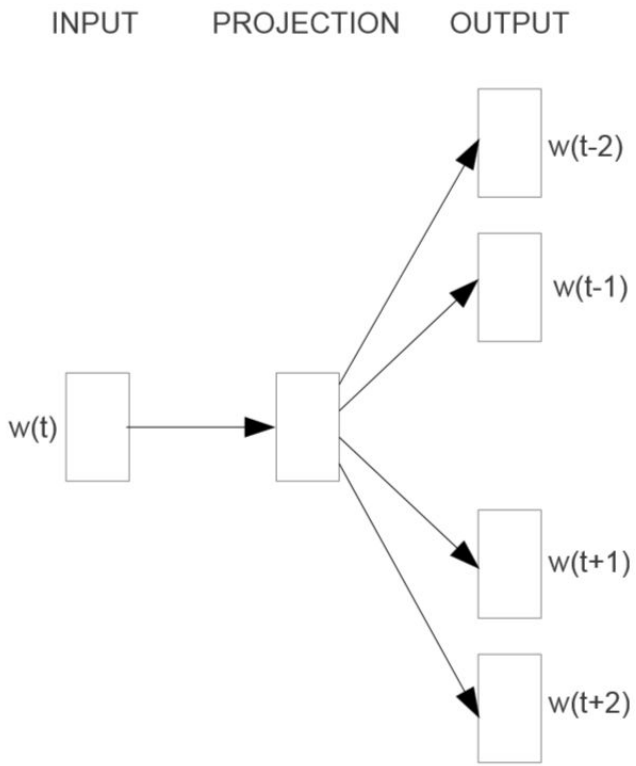
Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.



CBOW



Skip-gram

word2vec Parameter Learning Explained

Xin Rong
ronxin@umich.edu

Abstract

The word2vec model and application by Mikolov et al. have attracted a great amount of attention in recent two years. The vector representations of words learned by word2vec models have been proven to be able to carry semantic meanings and are useful in various NLP tasks. As an increasing number of researchers would like to experiment with word2vec, I notice that there lacks a material that comprehensively explains the parameter learning process of word2vec in details, thus preventing many people with less neural network experience from understanding how exactly word2vec works.

This note provides detailed derivations and explanations of the parameter update equations for the word2vec models, including the original continuous bag-of-word (CBOW) and skip-gram models, as well as advanced tricks, hierarchical soft-max and negative sampling. In the appendix a review is given on the basics of neuron network models and backpropagation.

1 Continuous Bag-of-Word Model

1.1 One-word context

We start from the simplest version of the continuous bag-of-word model (CBOW) introduced in Mikolov et al. (2013a). We assume that there is only one word considered per context, which means the model will predict one target word given one context word, which is like a bigram model.

Figure 1 shows the network model under the simplified context definition[1]. In our setting, the vocabulary size is V, and the hidden layer size is N. The nodes on adjacent layers are fully connected. The input vector is a one-hot encoded vector, which means for a given input context word, only one node of $\{x_1, \cdots, x_V\}$ will be 1, and all other nodes are 0.

The weights between the input layer and the output layer can be represented by a $V \times N$ matrix $\mathbf{W}$. Each row of $\mathbf{W}$ is the N-dimension vector representation $\mathbf{v}_w$ of the

[1] In Figures 1, 2, 3, and the rest of this note, $\mathbf{W}'$ is not the transpose of $\mathbf{W}$, but a different matrix instead.

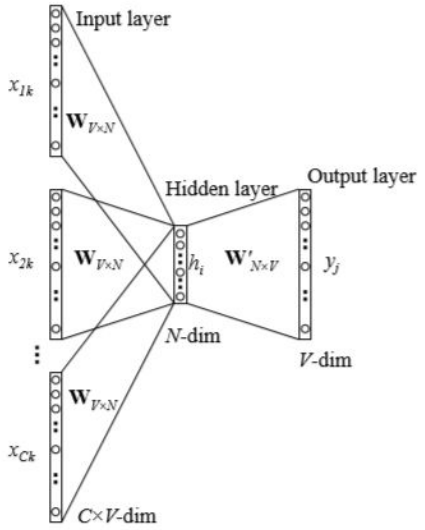


Figure 2: Continuous bag-of-word model

The update equation for the hidden→output weights stay the same as that for the one-word-context model (11). We copy it here:

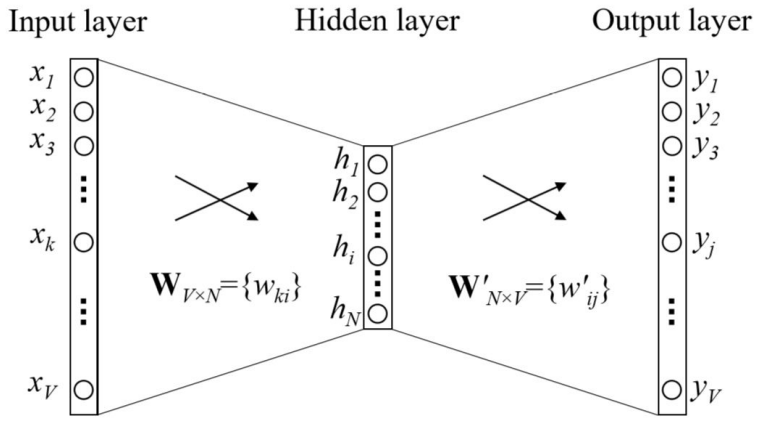
$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V. \tag{22}$$

Note that we need to apply this to every element of of the hidden→output weight matrix for each training instance.

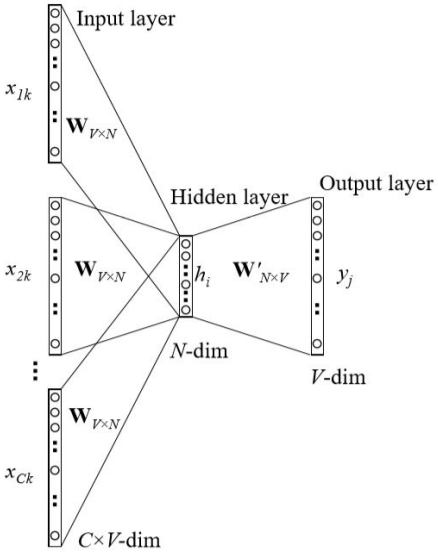
The update equation for input→hidden weights is similar to (16), except that now we need to apply the following equation for every word $w_{I,c}$ in the context:

$$\mathbf{v}^{(\text{new})}_{w_{I,c}} = \mathbf{v}^{(\text{old})}_{w_{I,c}} - \frac{1}{C} \cdot \eta \cdot \text{EH} \qquad \text{for } c = 1, 2, \cdots, C. \tag{23}$$

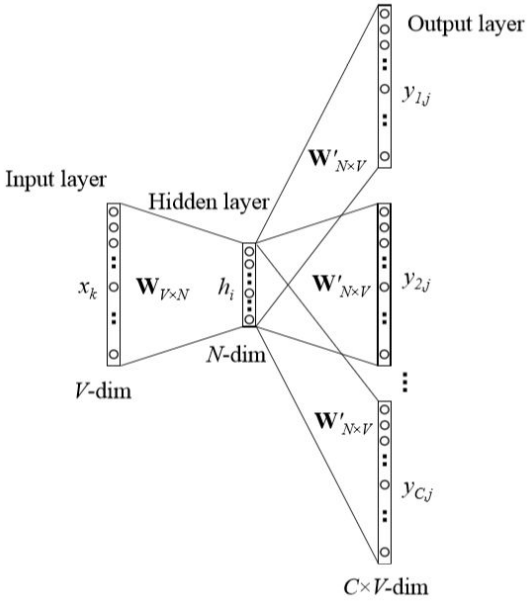
where $\mathbf{v}_{w_{I,c}}$ is the input vector of the c-th word in the input context; η is a positive learning rate; and $\text{EH} = \frac{\partial E}{\partial h_i}$ is given by (12). The intuitive understanding of this update equation is the same as that for (16).



word2vec model architecture



CBOW



Skip-gram

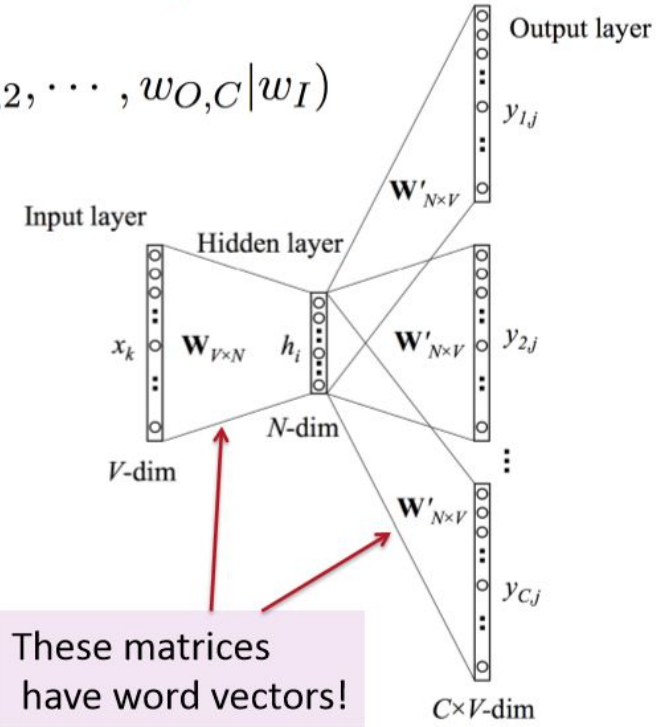
Word2vec (SkipGram)

$$E = -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C}|w_I)$$

- Loss function is negative probability of predictions of context words
- The hidden layer simply selects a row of **W** based on the input word

$$\mathbf{h} = \mathbf{x}^T\mathbf{W} = \mathbf{W}_{(k,\cdot)}$$

- This is then mapped by another matrix to an output probability



Christopher Manning
Deep Learning Summer School (2015)

Atomic Word Representation

Word is represented by identity

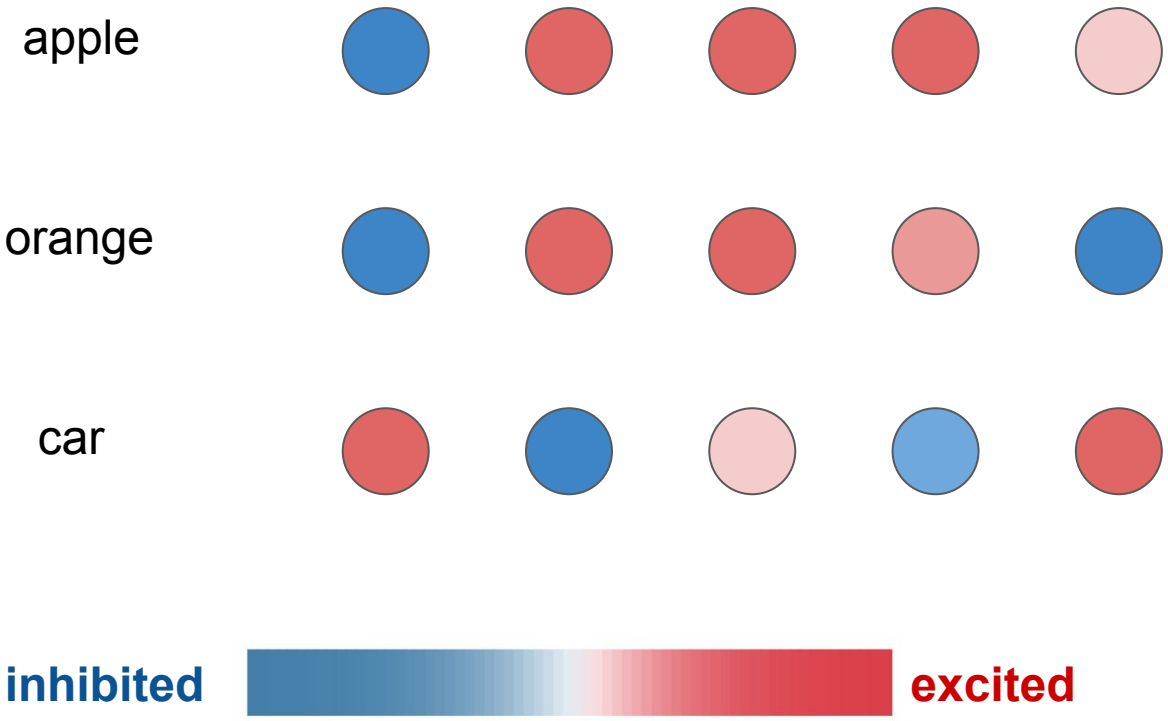
apple [0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]

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

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

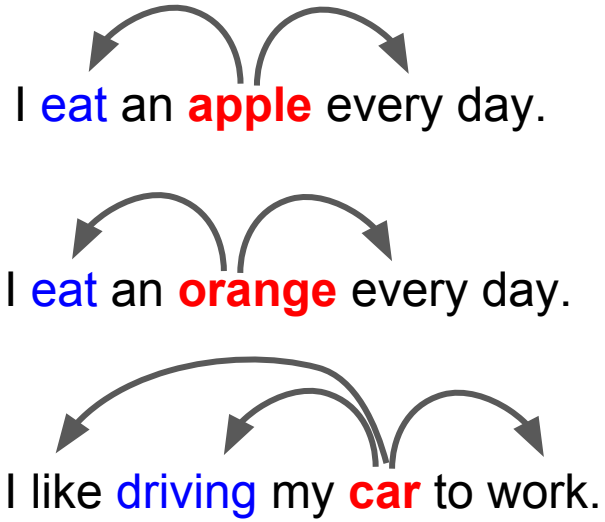
Distributed Representation

Word is represented as continuous level of activations

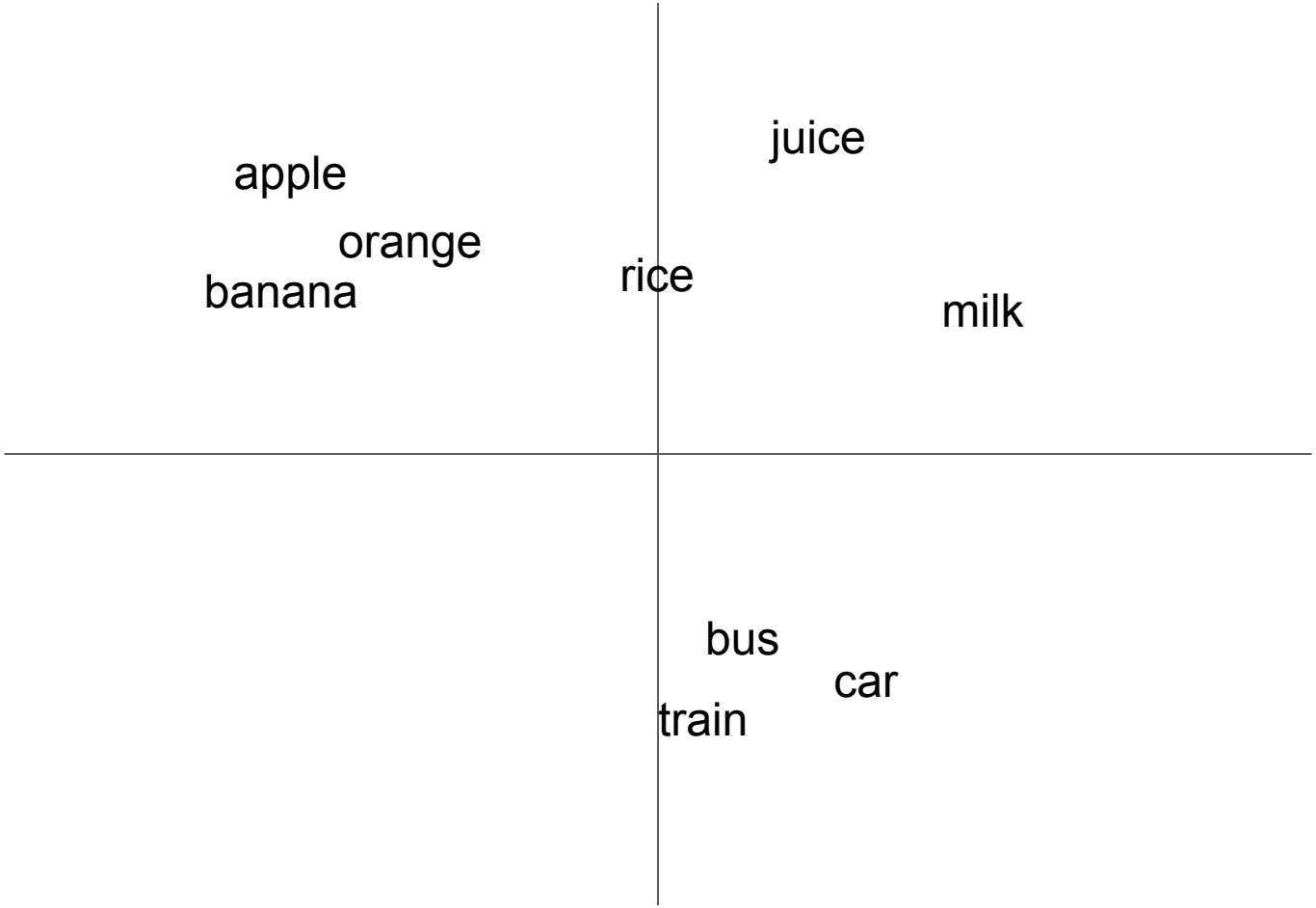


Contextual Representation

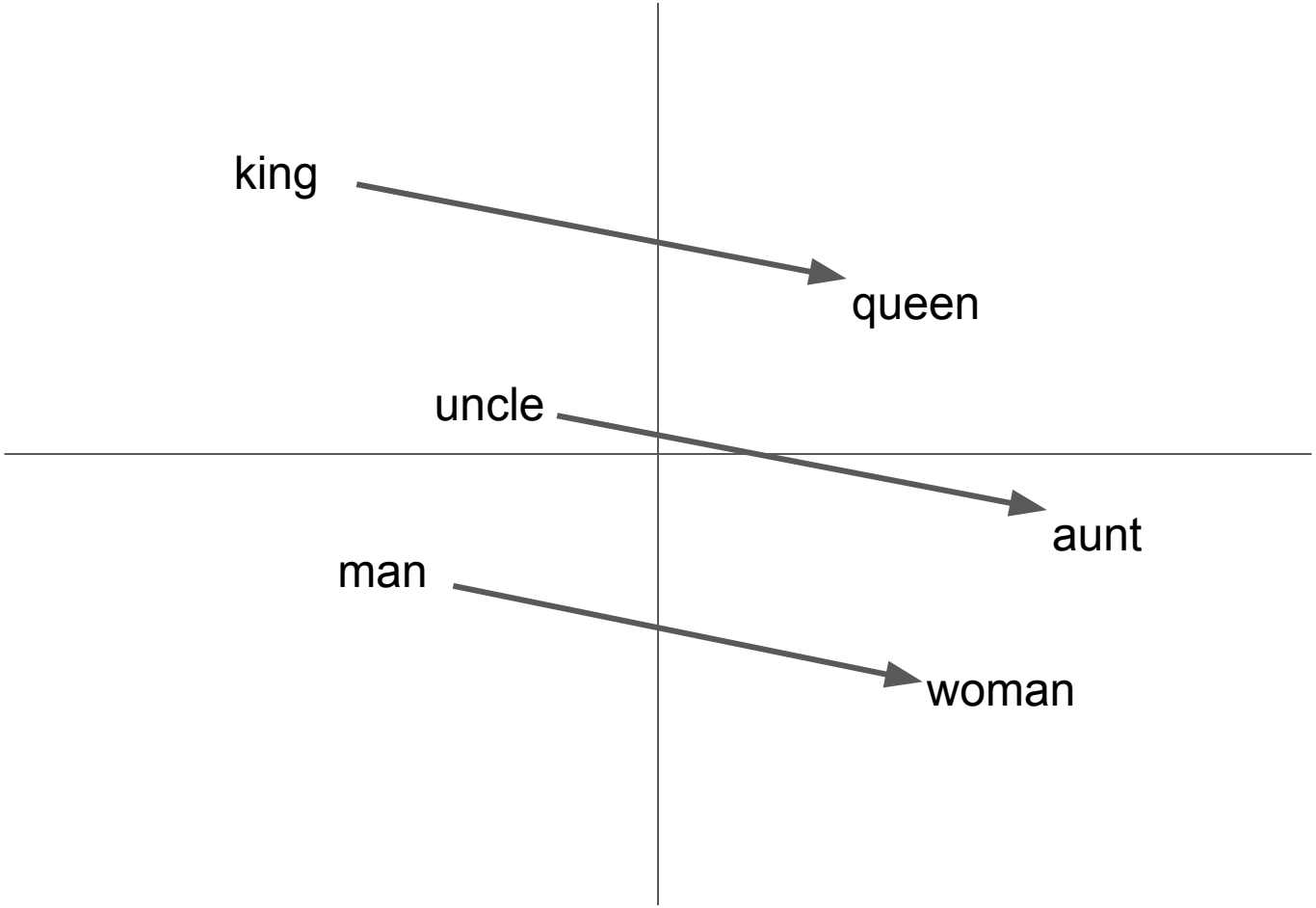
Word is represented by context in use



Word Vectors



Word Analogy



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

Applications benefited by word embeddings

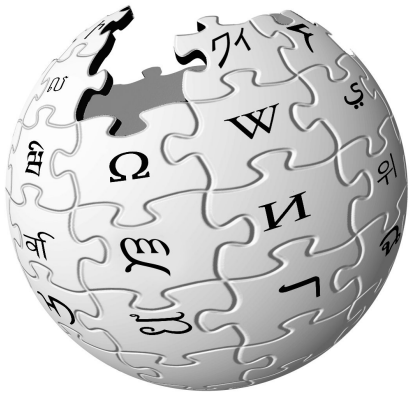
- Dependency parsing
- Named entity recognition
- Document classification
- Sentiment Analysis
- Paraphrase Detection
- Word Clustering
- Machine Translation

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

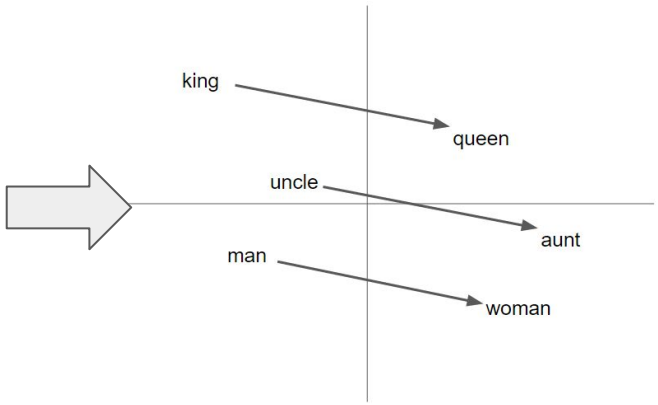
Table 7: Comparison and combination of models on the Microsoft Sentence Completion Challenge

"Efficient estimation of word representations in vector space" Mikolov & Chen et al. 2013

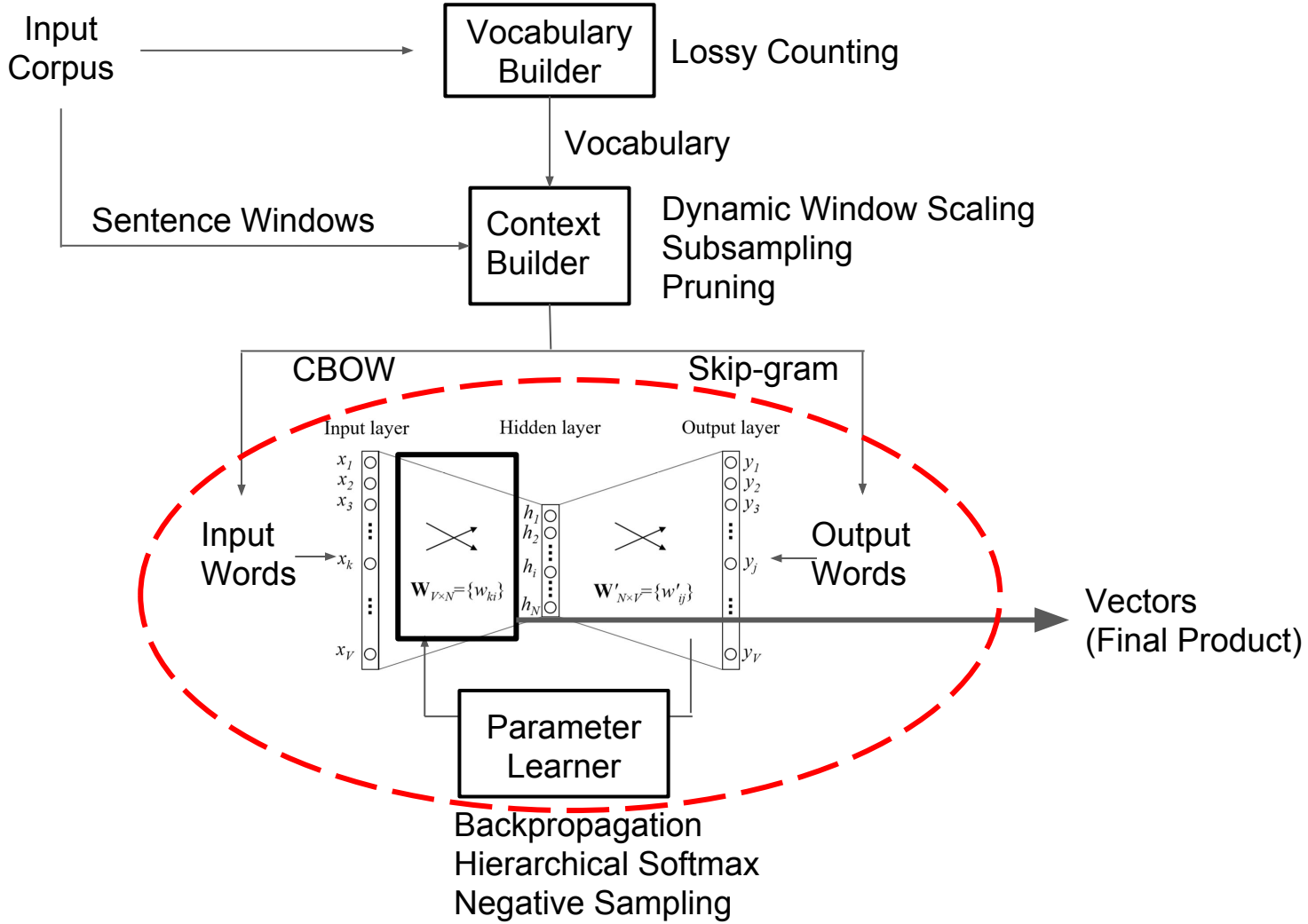
word2vec as a (powerful) black box



Wikipedia

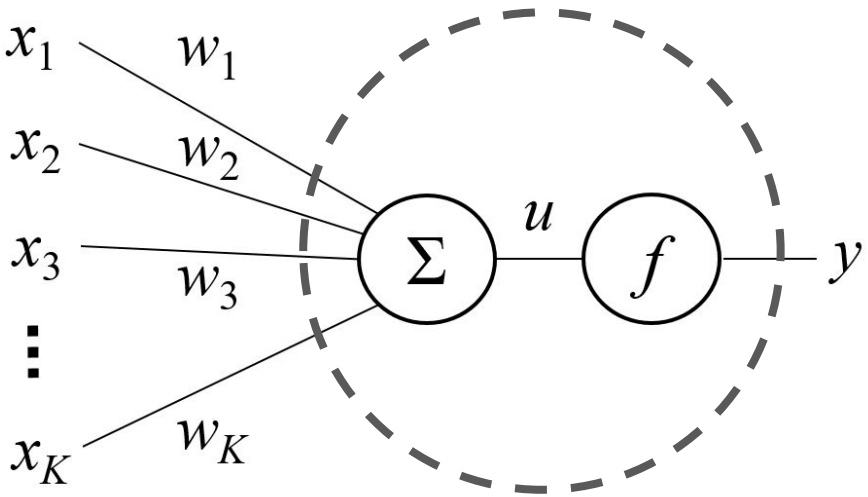


word2vec decomposed



wevi
demo
bit.ly/wevi-online

Basic Neuron Structure



a neuron

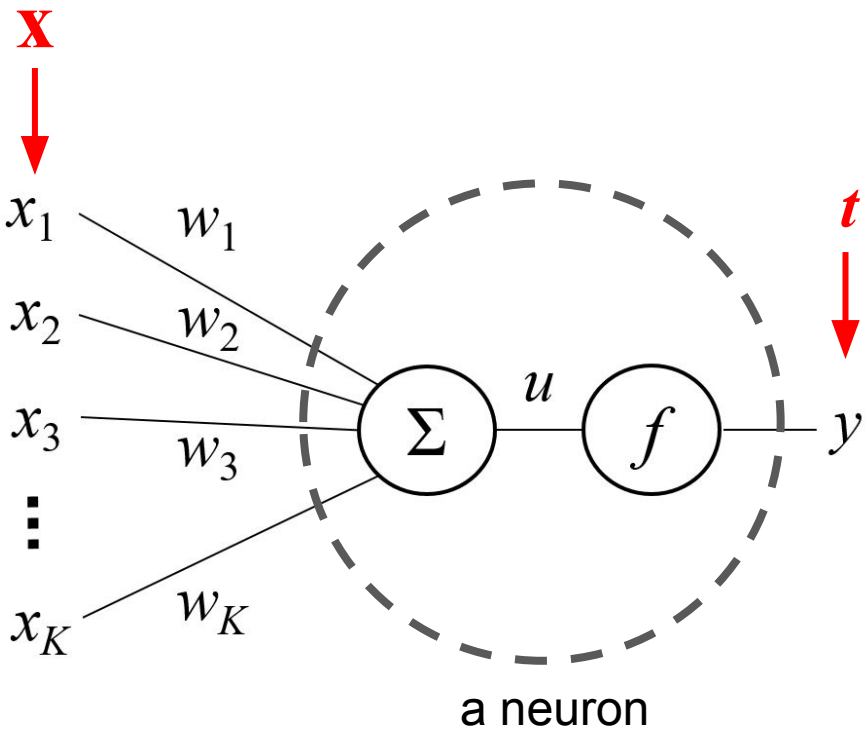
$$u = \sum_{i=0}^{K} w_i x_i$$

$$y = f(u)$$

$$f(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$f(u) = \frac{1}{1 + e^{-u}}$$

Training a Single Neuron



a neuron

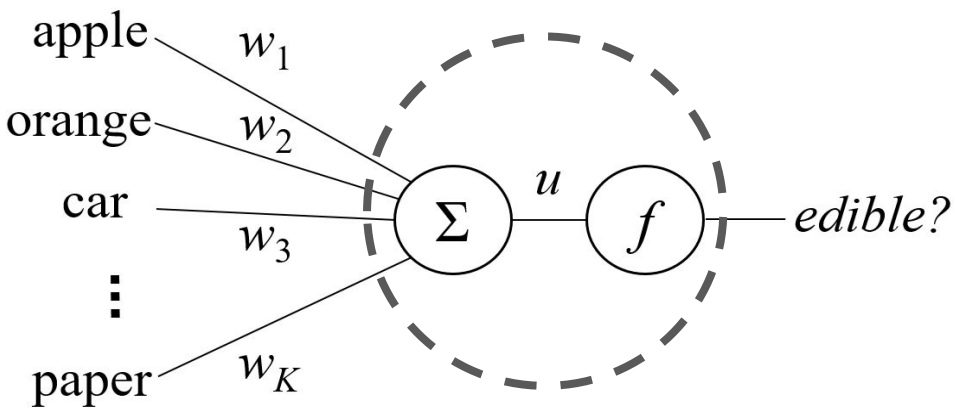
$$E = \frac{1}{2}(t - y)^2$$

$$\frac{\partial E}{\partial y} = y - t$$

$$\begin{aligned} \frac{\partial E}{\partial u} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u} \\ &= (y - t)y(1 - y) \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial w_i} \\ &= (y - t) \cdot y(1 - y) \cdot x_i \end{aligned}$$

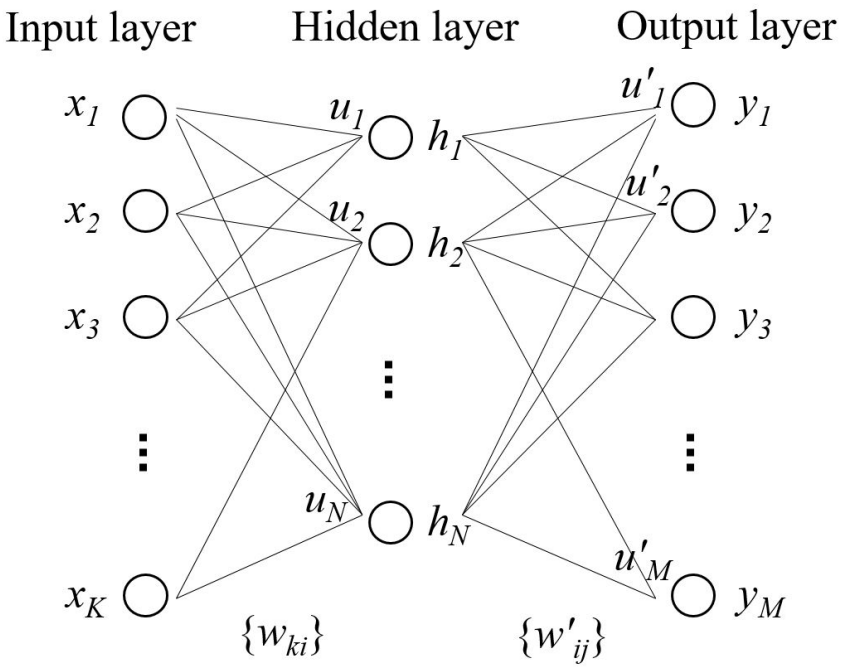
Task Afforded by a Single Neuron



A lookup table

Item	Edible?
apple	Y
orange	Y
car	N
...	...
paper	N

Multilayer Neural Network



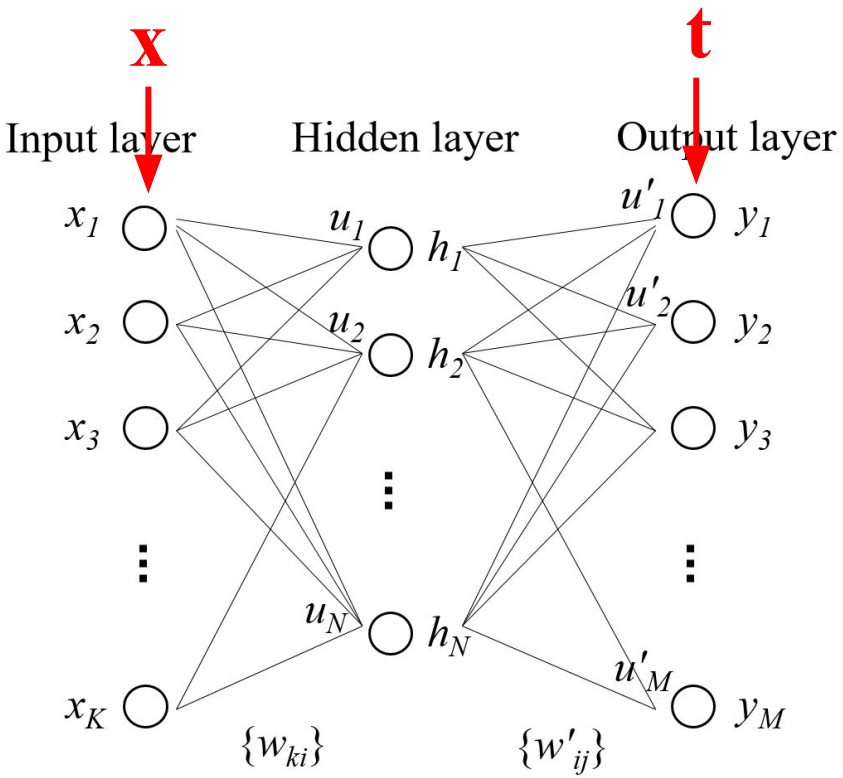
$$u_i = \sum_{k=1}^{K} w_{ki} x_k$$

$$h_i = f(u_i)$$

$$u'_j = \sum_{i=1}^{N} w'_{ij} h_i$$

$$y_j = f(u'_j)$$

Backpropagation



$$E = \frac{1}{2}\sum_{j=1}^{M}(y_j - t_j)^2$$

$$\frac{\partial E}{\partial y_j} = y_j - t_j$$

$$\frac{\partial E}{\partial u'_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial u'_j}$$

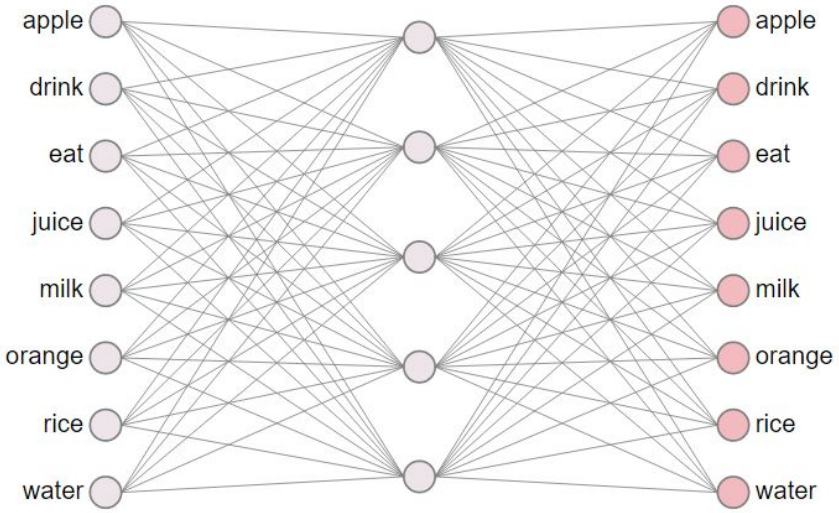
$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u'_j} \cdot \frac{\partial u'_j}{\partial w'_{ij}}$$

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{M} \frac{\partial E}{\partial u'_j} \frac{\partial u'_j}{\partial h_i}$$

$$\frac{\partial E}{\partial u_i} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial u_i}$$

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial u_i} \cdot \frac{\partial u_i}{\partial w_{ki}}$$

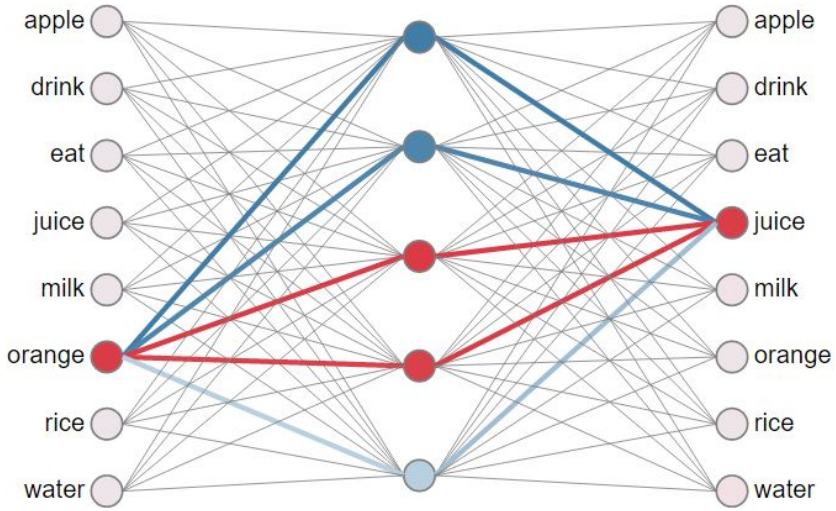
word2vec network

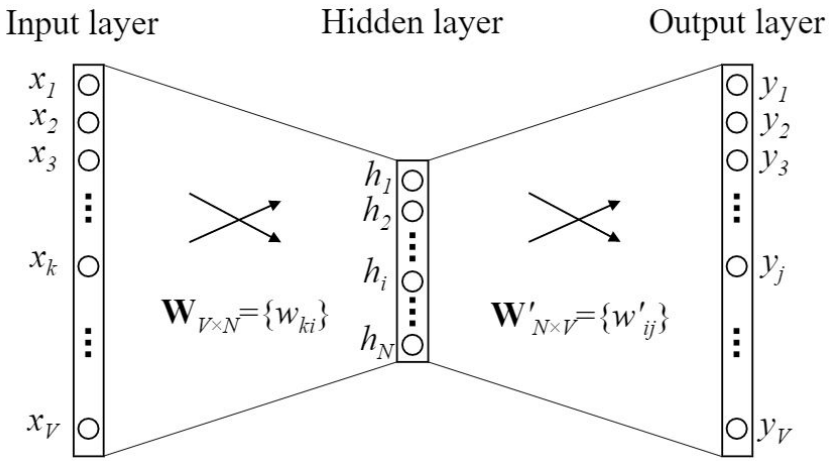


Structure Highlights:

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

word2vec network



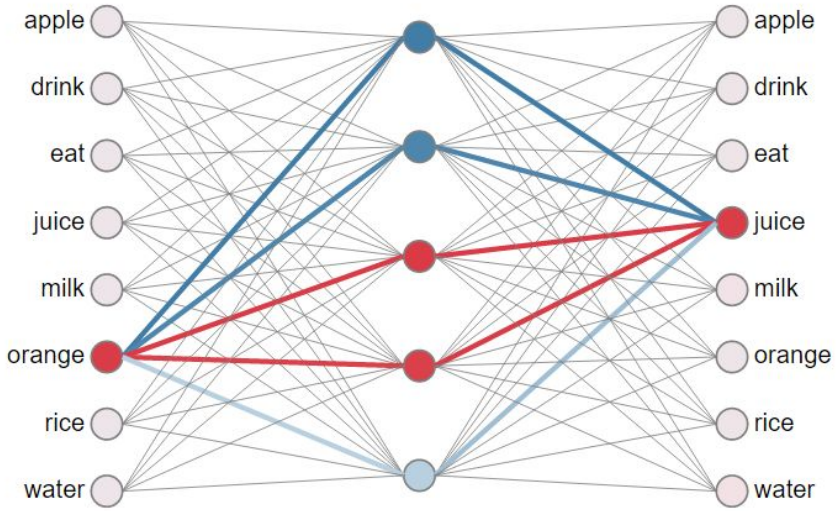


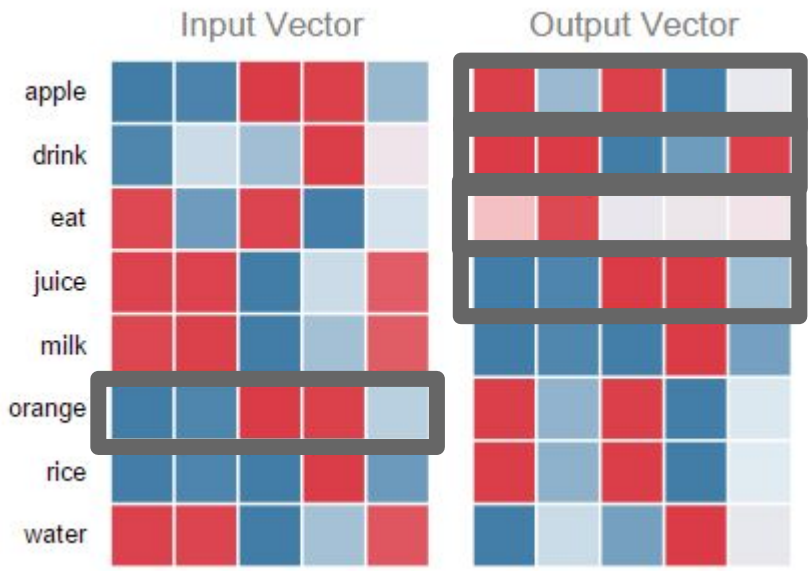
$$\mathbf{h} = \mathbf{x}^T\mathbf{W} := \mathbf{v}_{w_I}$$

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

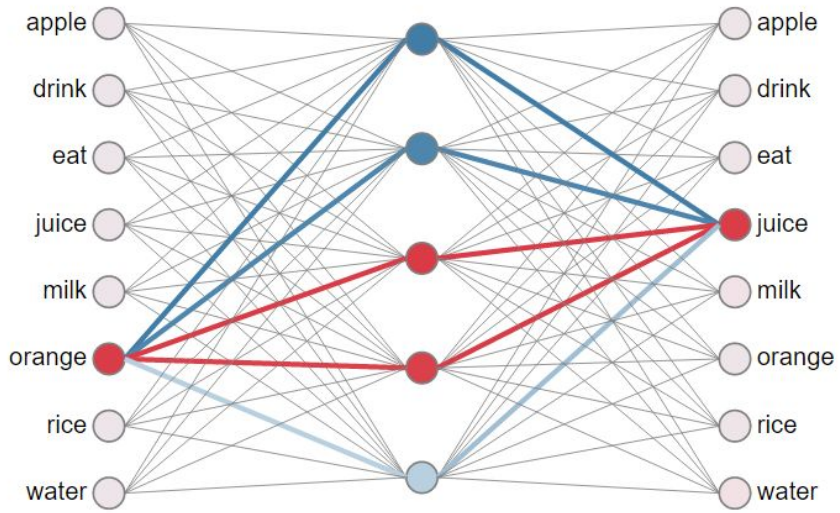
$$p(w_j|w_I) = \frac{\exp\left(\mathbf{v}'^{T}_{w_O}\mathbf{v}_{w_I}\right)}{\sum_{j'=1}^{V}\exp\left(\mathbf{v}'^{T}_{w'_j}\mathbf{v}_{w_I}\right)}$$

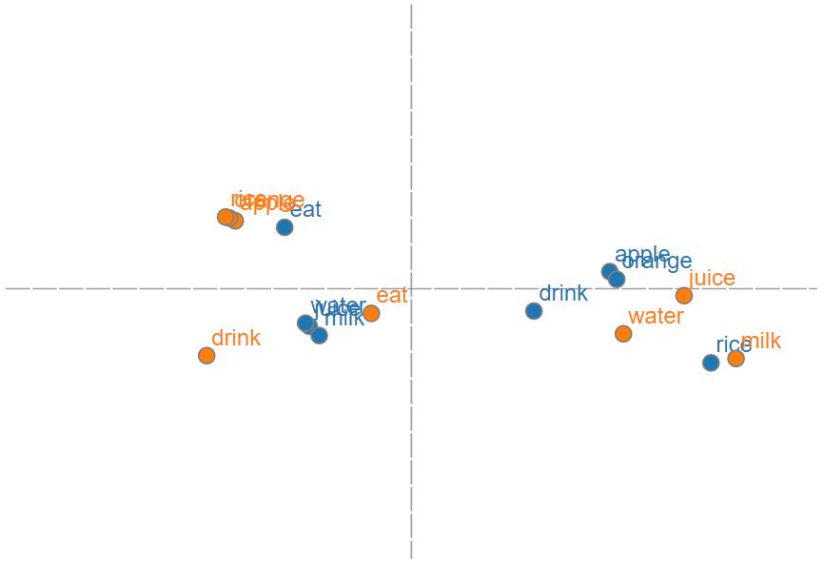
neural network and weight matrices



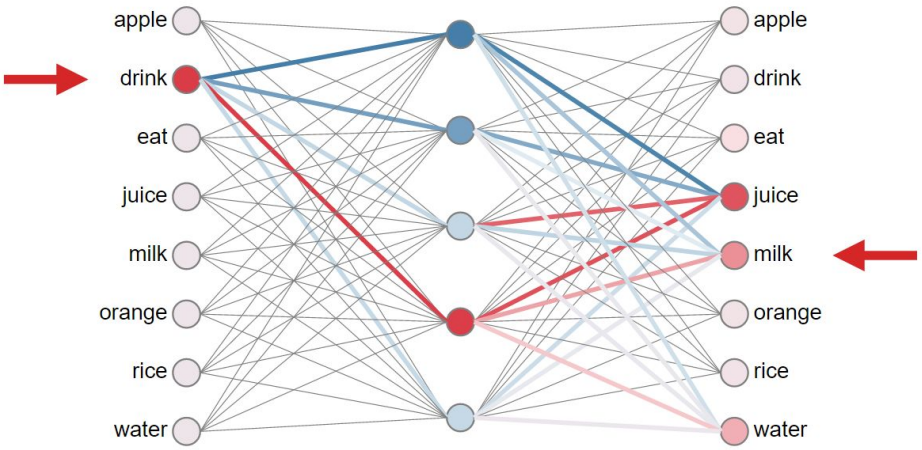


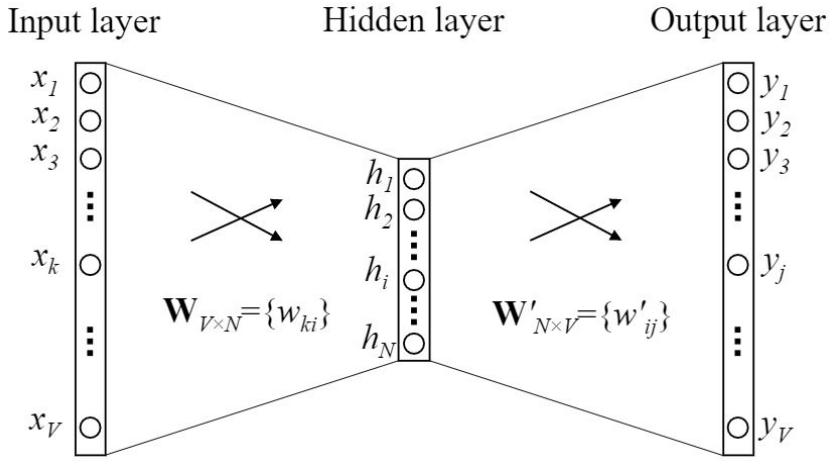
neural network and word vectors





Training: updating word vectors





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

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

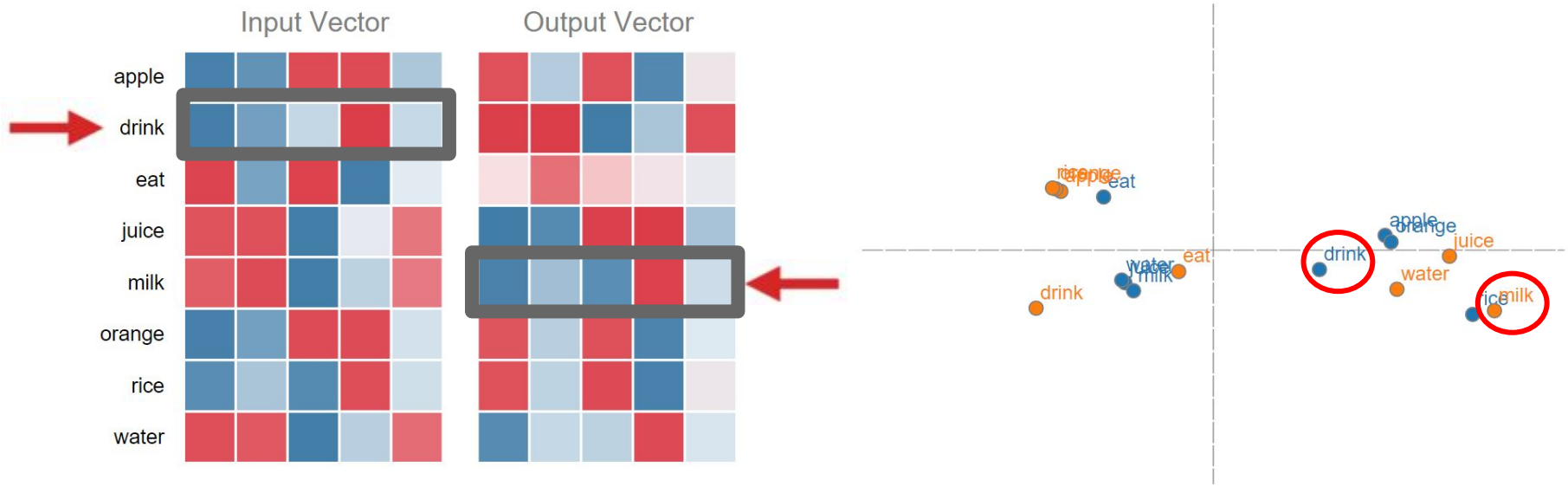
$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{u_j}{\partial w'_{ij}}$$

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

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}}$$

Intuition of **output** vector update rule

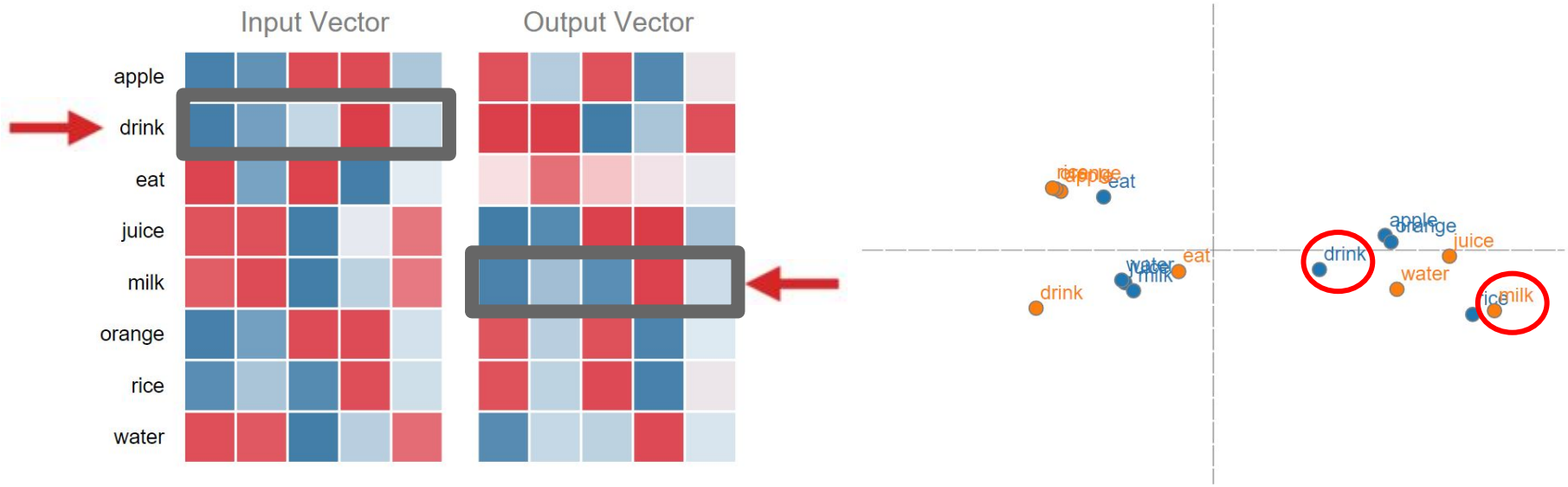
$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot e_j \cdot \mathbf{h}$$



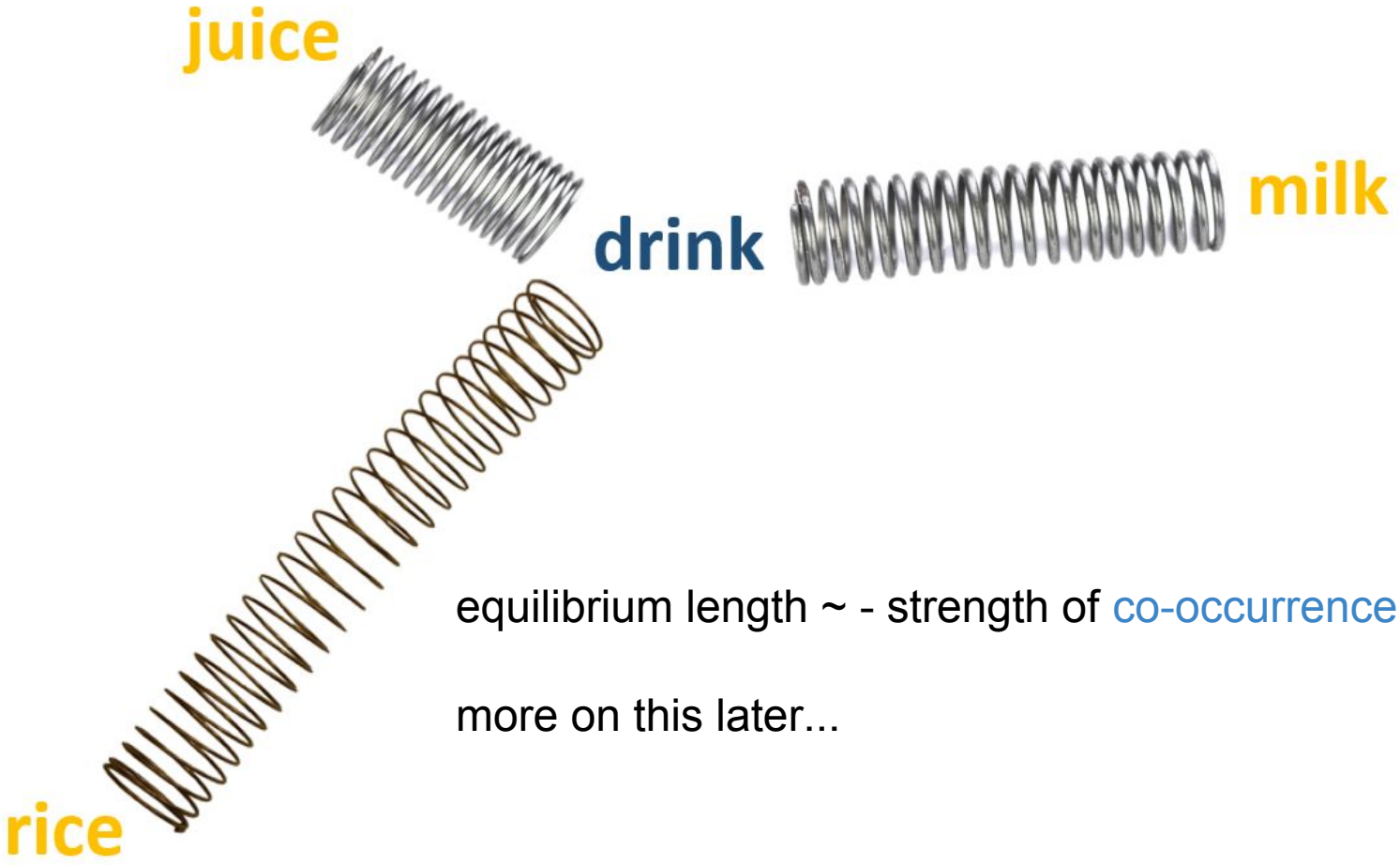
Intuitive Understanding of **Input** Vectors

$$\mathrm{EH}_i := \frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij}$$

$$\mathbf{v}_{w_I}^{(\mathrm{new})} = \mathbf{v}_{w_I}^{(\mathrm{old})} - \eta \cdot \mathrm{EH}$$



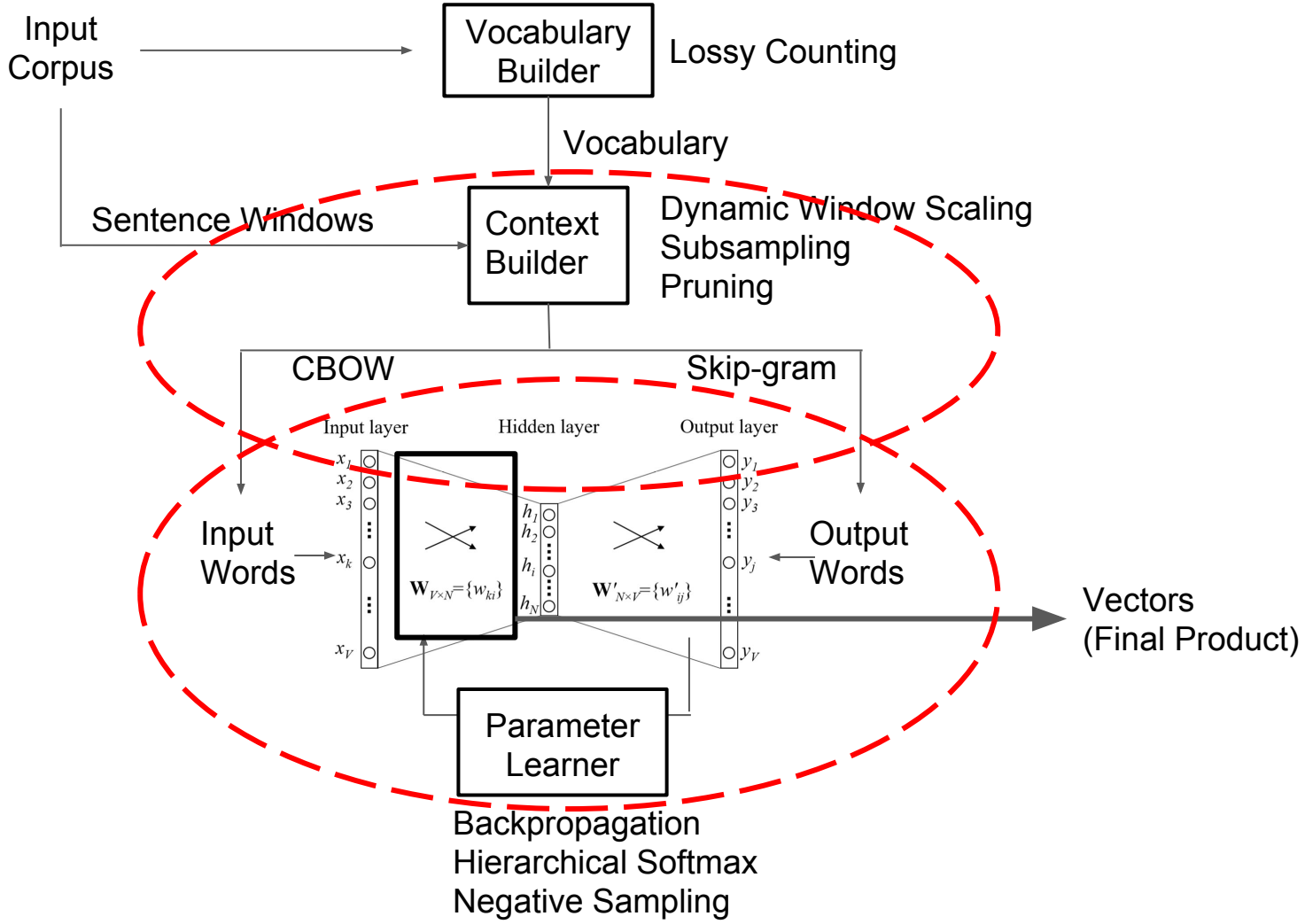
Resemblance to a force-directed graph



equilibrium length ~ - strength of co-occurrence

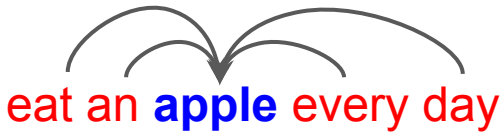
more on this later...

word2vec decomposed

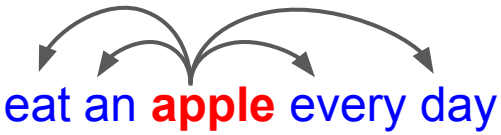


How do we select input and output words?

Method 1: continuous bag-of-word (CBOW)

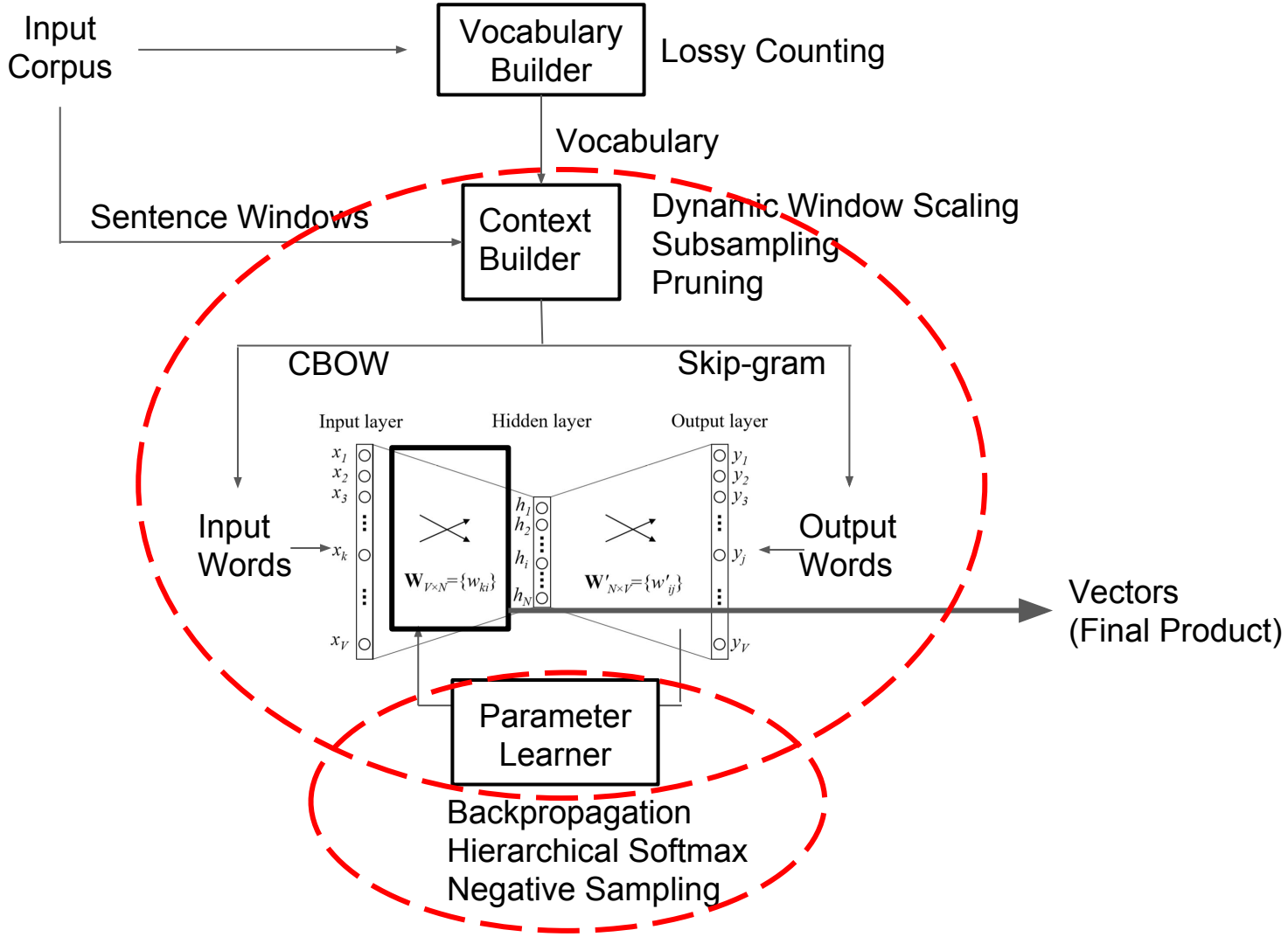


Method 2: skip-gram (SG)

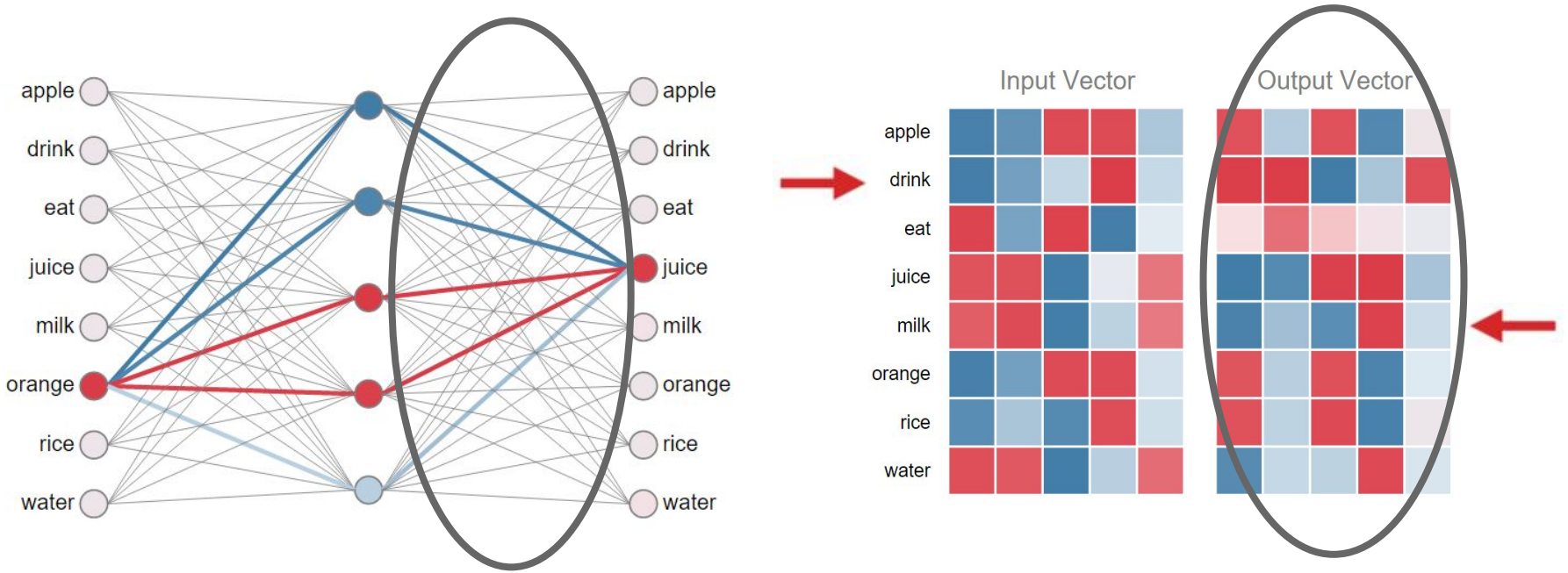


wevi
demo
(CBOW and SG)

word2vec decomposed

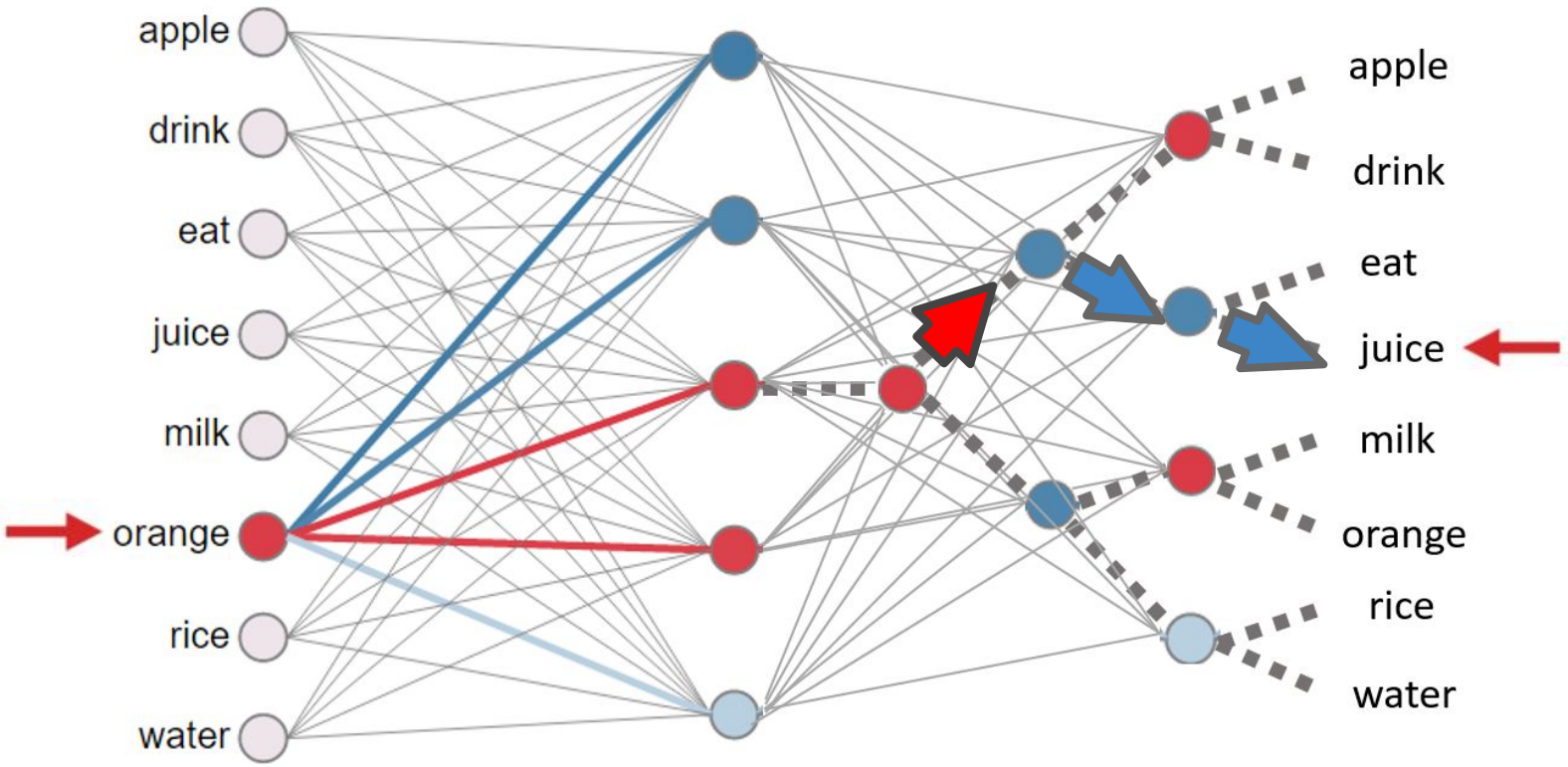


Training Generic Softmax is Intractable

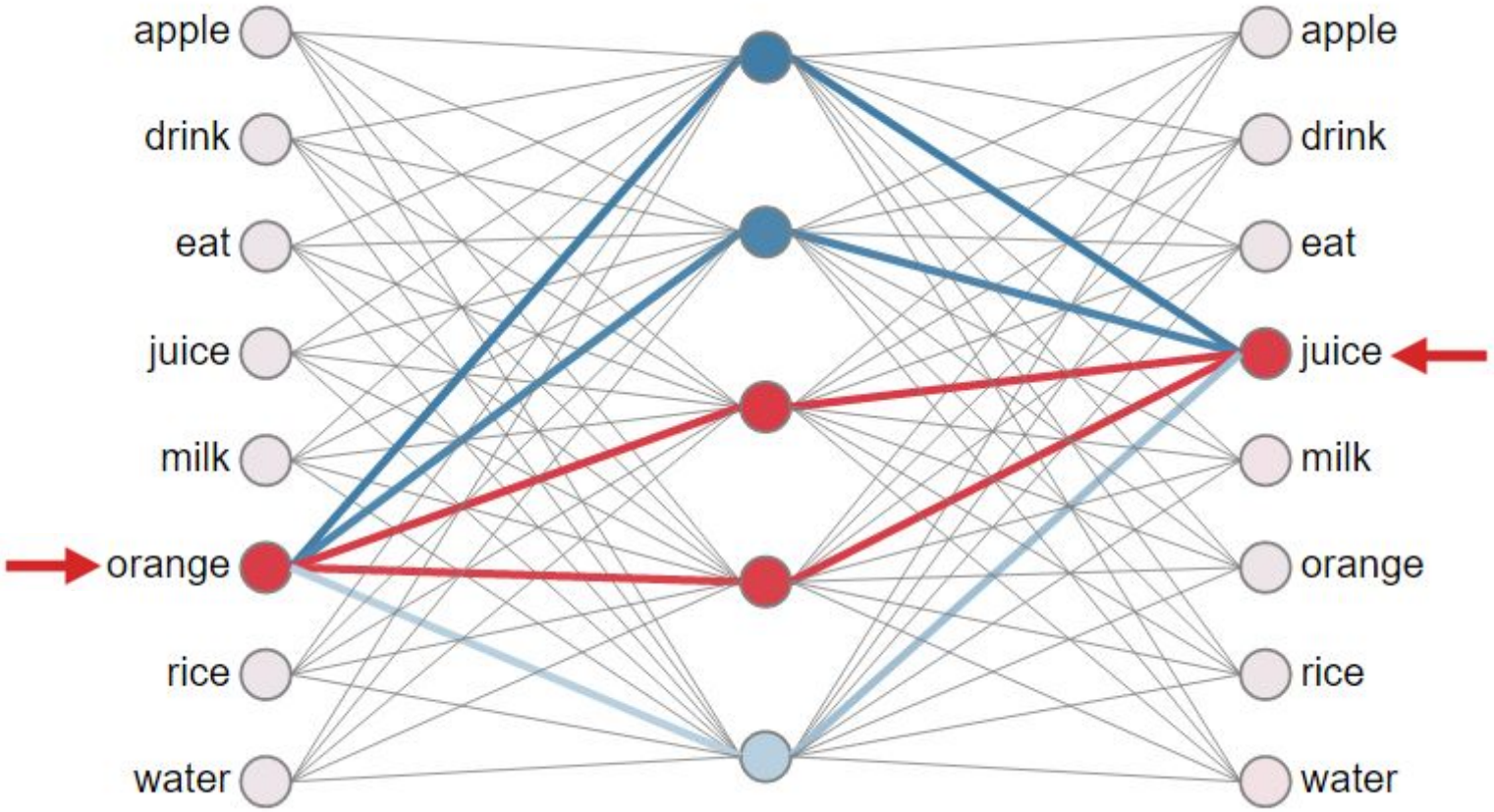


need to update every single output vector!

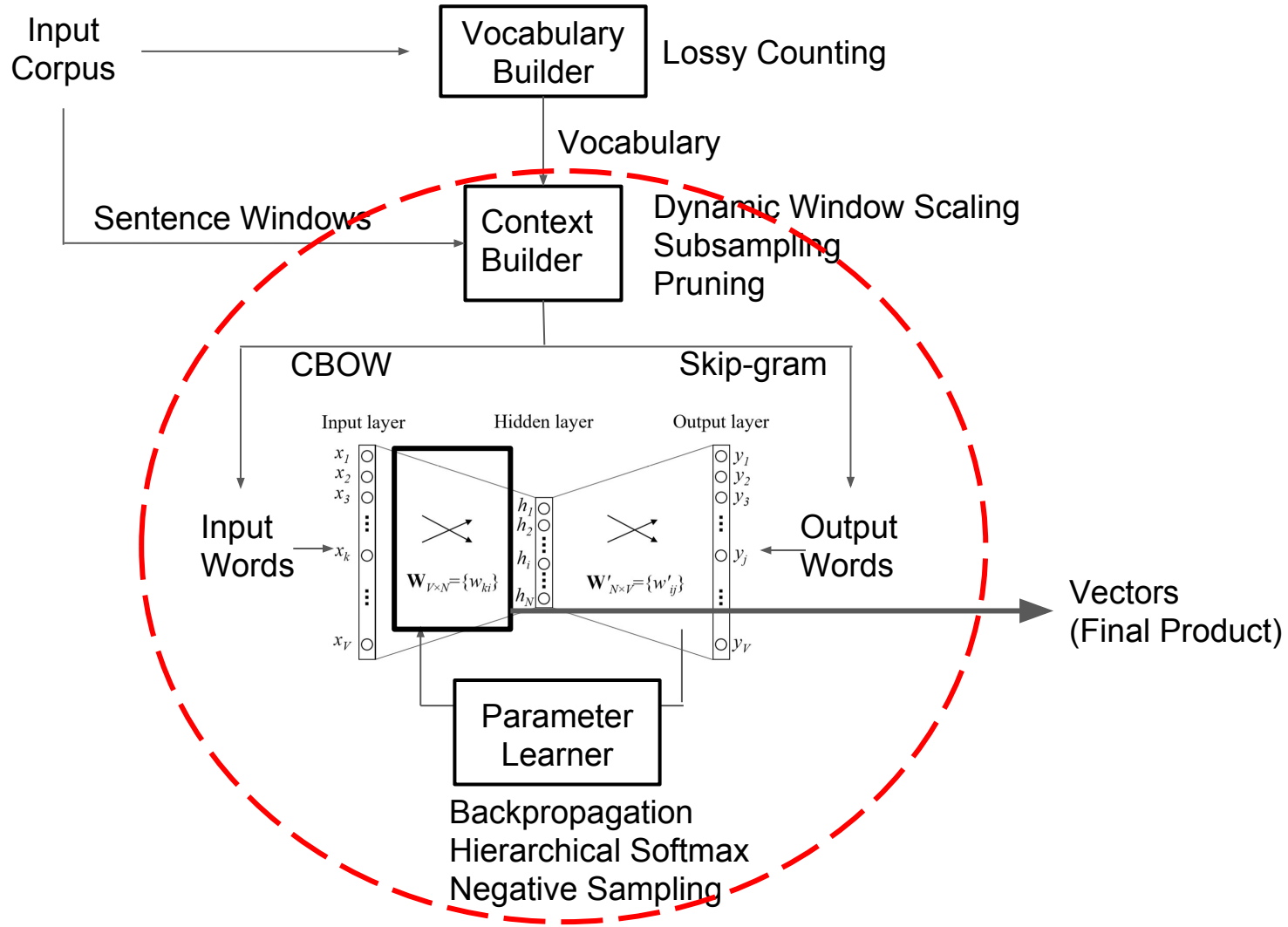
Hierarchical Softmax



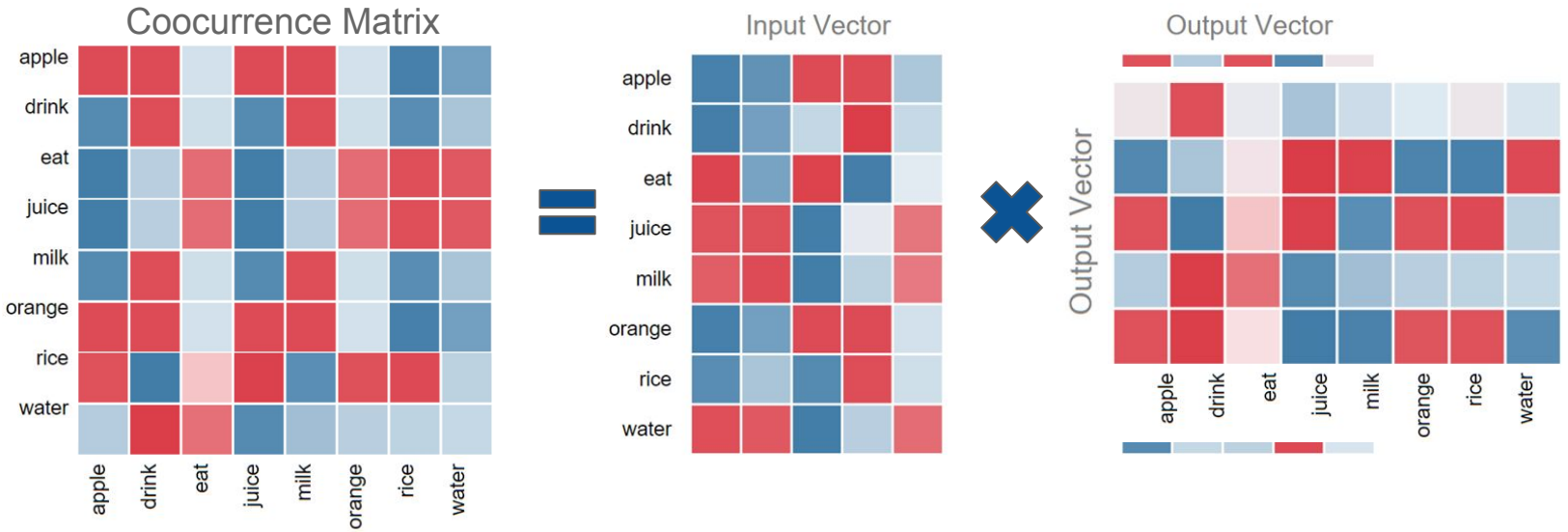
Negative Sampling



word2vec decomposed



Interpreting word embedding model



Neural Word Embedding as Implicit Matrix Factorization

Omer Levy
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

Yoav Goldberg
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

Abstract

We analyze skip-gram with negative-sampling (SGNS), a word embedding method introduced by Mikolov et al., and show that it is implicitly factorizing a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant. We find that another embedding method, NCE, is implicitly factorizing a similar matrix, where each cell is the (shifted) log conditional probability of a word given its context. We show that using a sparse *Shifted Positive PMI* word-context matrix to represent words improves results on two word similarity tasks and one of two analogy tasks. When dense low-dimensional vectors are preferred, exact factorization with SVD can achieve solutions that are at least as good as SGNS's solutions for word similarity tasks. On analogy questions SGNS remains superior to SVD. We conjecture that this stems from the weighted nature of SGNS's factorization.

1 Introduction

Most tasks in natural language processing and understanding involve looking at words, and could benefit from word representations that do not treat individual words as unique symbols, but instead

Levy & Goldberg. "Neural word embedding as implicit matrix factorization." *NIPS* 2014.

Improving Distributional Similarity with Lessons Learned from Word Embeddings

Omer Levy **Yoav Goldberg** **Ido Dagan**
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
{omerlevy,yogo,dagan}@cs.biu.ac.il

Abstract

Recent trends suggest that neural-network-inspired word embedding models outperform traditional count-based distributional models on word similarity and analogy detection tasks. We reveal that much of the performance gains of word embeddings are due to certain system design choices and hyperparameter optimizations, rather than the embedding algorithms themselves. Furthermore, we show that these modifications can be transferred to traditional distributional models, yielding similar gains. In contrast to prior reports, we observe mostly local or insignificant performance differences between the methods, with no global advantage to any single approach over the others.

A recent study by Baroni et al. (2014) conducts a set of systematic experiments comparing `word2vec` embeddings to the more traditional distributional methods, such as pointwise mutual information (PMI) matrices (see Turney and Pantel (2010) and Baroni and Lenci (2010) for comprehensive surveys). These results suggest that the new embedding methods consistently outperform the traditional methods by a non-trivial margin on many similarity-oriented tasks. However, state-of-the-art embedding methods are all based on the same bag-of-contexts representation of words. Furthermore, analysis by Levy and Goldberg (2014c) shows that `word2vec`'s SGNS is implicitly factorizing a word-context PMI matrix. That is, the mathematical objective and the sources of information available to SGNS are in fact very similar to those employed by the more traditional methods.

What, then, is the source of superiority (or per-

Levy, Goldberg, and Dagan. *ACL* (2015)

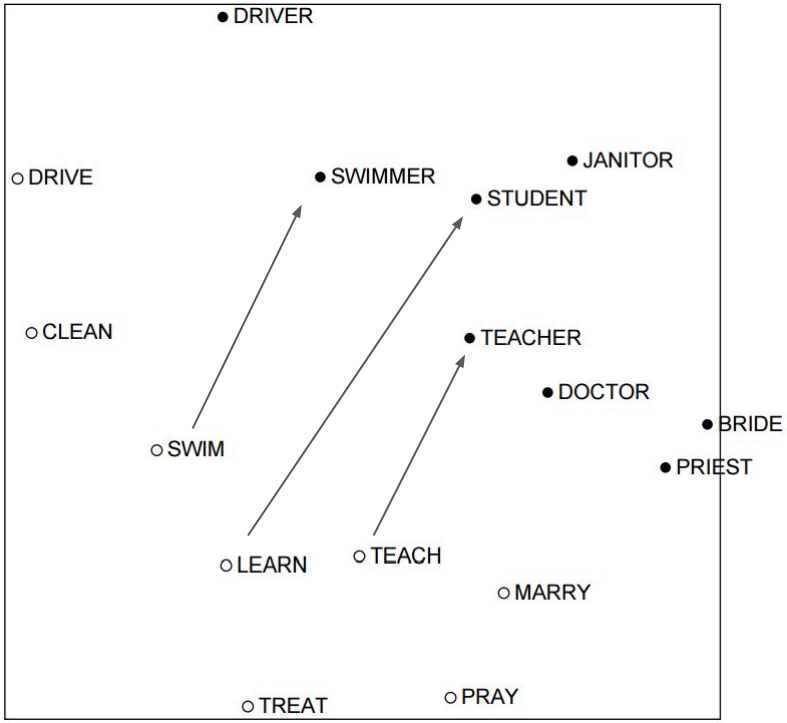
Other neural embedding models

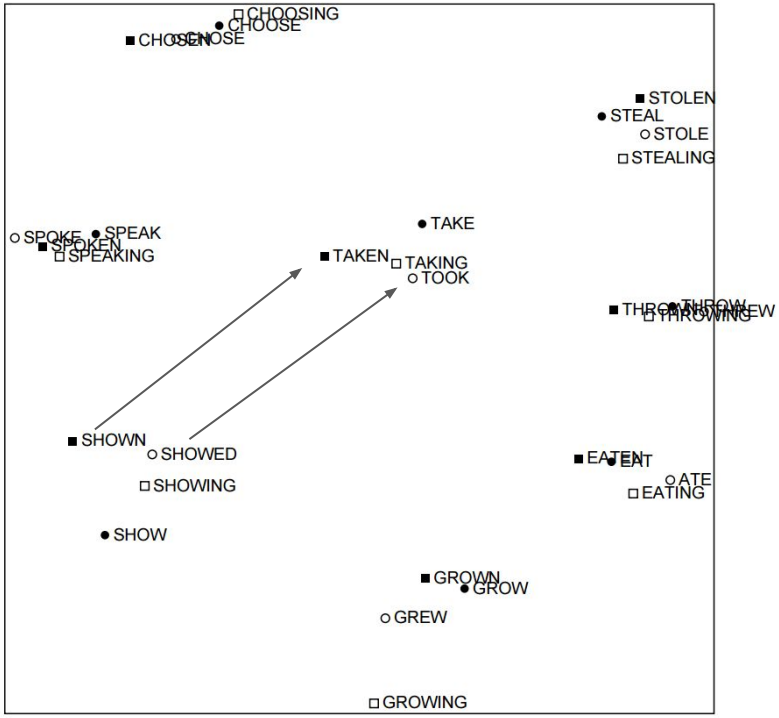
- Bengio (2003)
- Mnih & Hinton (2008)
- Collobert & Weston (2008)
- Minh et a. (2013)
- GloVe by Pennington et al. (2014)
- DeepWalk by Perozzi (2014)
- LINE by Tang et al. (2015)

More on Word Analogy

Rohde, Gonnerman & Plaut (2005)

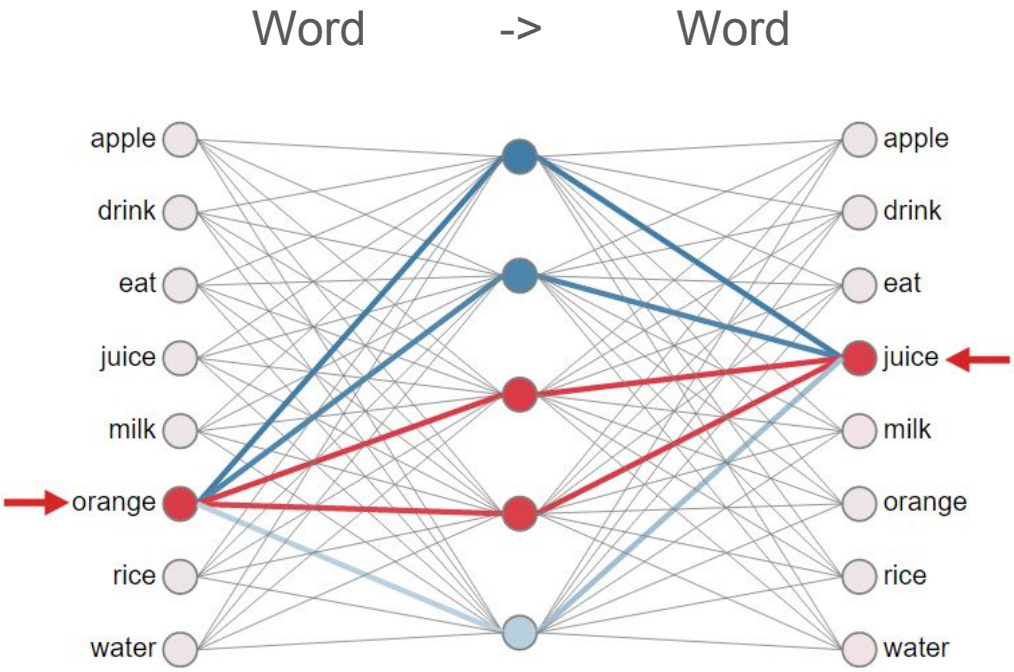
"An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence"





wevi
demo
(word analogy)

"Miscellaneous" Embedding



Alternative input/output setting:

- User -> Tweet
- Customer -> Product
- Word -> Parse-tree Neighbors
- Entity -> Neighbor in a list
- Node -> Random walk path in a graph

Limitations

- Word ambiguity
- Debuggability
- Sequence

Take-aways

- Word embedding techniques perform either explicit or implicit matrix factorization to word co-occurrence matrix.
- The neural network structure of word2vec is a feedforward network with one hidden layer with a linear activation function.
- The training method of word2vec is backpropagation with stochastic gradient descent.
- The loss function is cross entropy.
- Training can be made feasible by using either hierarchical softmax or negative sampling.
- Word embeddings of equal quality as word2vec can be obtained via count-based methods with carefully tuned co-occurrence metrics.

Thanks!

Source code: github.com/ronxin/wevi

Online demo: ronxin.github.io/wevi