Distributed Representations of Words and Phrases and their Compositionality

- **Mikolov T., Sutskever I., Chen K., Corado G., Dean J.**

Advances in neural information processing systems. 2013.

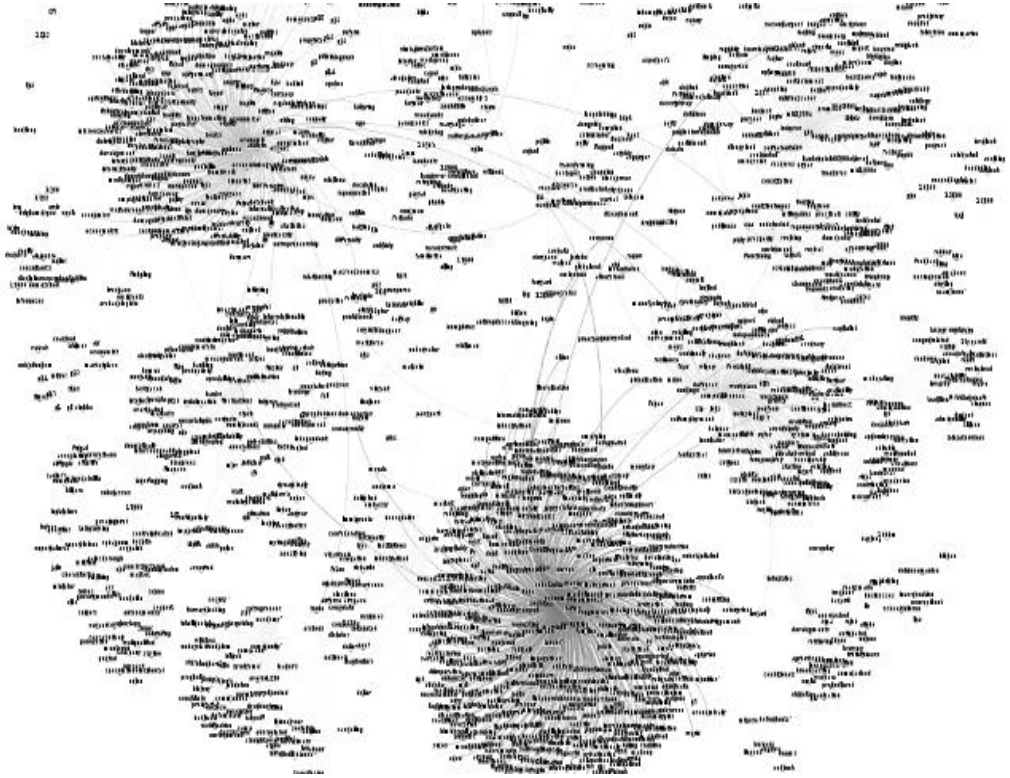
Presented by - Shubham Mittal

Outline

- Language Modelling:
  - History
  - Motivation
  - Word2Vec Architecture
  - Optimization Techniques
  - Phrase2Vec
  - Advantages
  - Linguistic Analogies
- Graph Modelling:
  - Learning Latent Representations
  - Motivation
  - Connection between graphs and language modelling
  - DeepWalk Architecture
  - Advantages

A Brief History

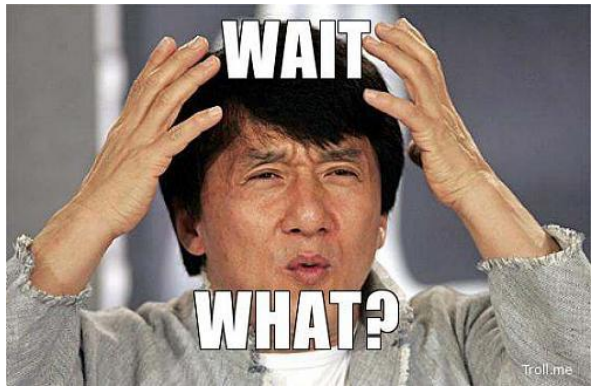
- Introduced in 2001 and re-introduced in 2003 by Bengio et al.
- The idea of Distributed Representations is even older - 1986
- Collobert and Weston - pre-trained word embeddings in 2008. Propose a neural network architectural model
- **Word2Vec** - a toolkit enabling the training and use of pre-trained embeddings proposed by Mikolov et al. brings embedding to the fore-front in NLP in 2013



“A word is known by the company it keeps”

- Word Embeddings has its roots in the above quote by Firth’54.

- *“Word Embeddings aim at quantifying and categorizing semantic similarities between linguistic items based on their distributional properties in large samples of language data”* – Wikipedia



Co-occurrence

…he curtains open and the moon shining in on the barely…

…ars and the cold , close moon " . And neither of the w…

…rough the night with the moon shining so brightly , it…

…made in the light of the moon . It all boils down , wr…

…surely under a crescent moon , thrilled by ice-white…

…sun , the seasons of the moon ? Home , alone , Jay pla…

…m is dazzling snow , the moon has risen full and cold…

…un and the temple of the moon , driving out of the hug…

…in the dark and now the moon rises , full and amber a…

…bird on the shape of the moon over the trees in front…

Demystifying Word Embeddings

- What's a word ?
  - An item from the Vocabulary, indexed by 1…V
  - One-hot encoding – represented by unit basic vectors with one single component as one and all other as zero.

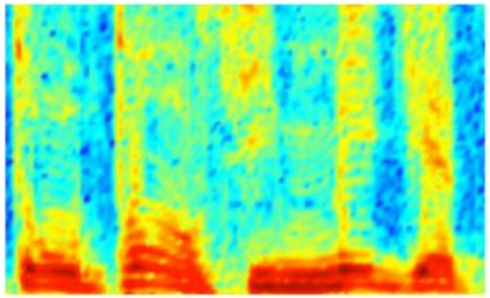
- What's a word embedding ?
  - A parametrized function **W**: *words* -> $\mathbf{R}^n$ mapping words in some language to a high dimensional vector (typically 200-500)

W("*cat*") = (0.2, −0.4, 0.7, ...)

W("*mat*") = (0.0, 0.6, −0.1, ...)

Why Learn Word Embeddings?

AUDIO



Audio Spectrogram

DENSE

IMAGES

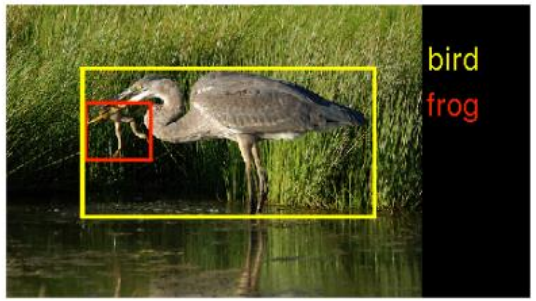


Image pixels

DENSE

TEXT

0	0	0	0.2	0	0.7	0	0	0	...	...

Word, context, or document vectors

SPARSE

Atomic Representation – Problem?

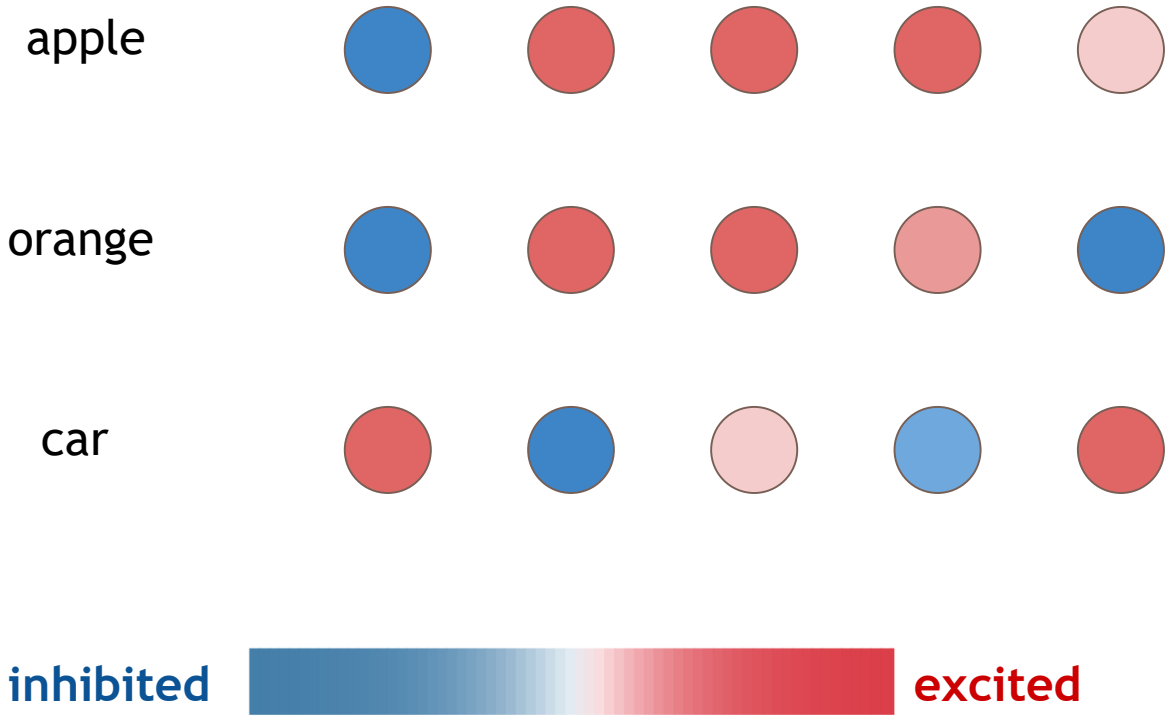
Word is represented by identity

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]

Problem ?
Motel [0 0 0 0 0 1 0 0 … 0 0 0] AND *Hotel* [0 0 0 0 0 0 0 0 0 … 0 1 0] = 0

Distributed Representation

Word is represented as continuous level of activations



Contextual Representation

Word is represented by context in use

How to use context?

- Solution – Create a cooccurrence matrix !

- 2 Ways ---
  - Use Full Document
  - Use Windows

- Word – document cooccurrence matrix -> Latent Semantic Analysis

- Windowed Approach -> Capture both semantic and syntactic (part-of-seech) information.

Window-based cooccurrence matrix

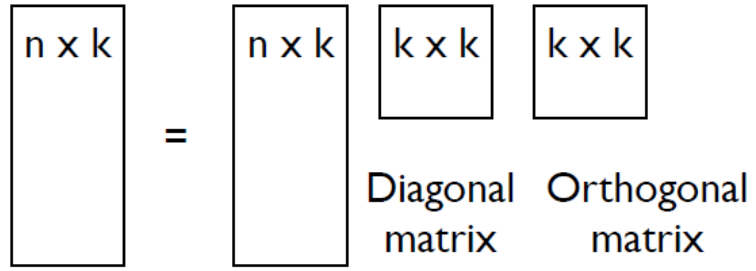
Corpus = {“I like deep learning”
“I like NLP”
“I enjoy flying”}

Context = previous word
and next word

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Co-occurrence Matrix with Singular Value Decomposition

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$$



$$\underset{m \times n}{\hat{X}} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix} \approx \underset{m \times r}{U} \begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix} \underset{r \times r}{S} \begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix} \underset{r \times n}{V^{\mathsf{T}}} \begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}$$

Problems?

- Increase in size with vocabulary
- Very high dimensional – requires HUGE storage
- Subsequent classification models have sparsity needs
- Models are less robust

Demystifying Word Embeddings -- Revisited

- What's a word ?
  - An item from the Vocabulary, indexed by 1…V
  - One-hot encoding – represented by unit basic vectors with one single component as one and all other as zero.

- What's a word embedding ?
  - A parametrized function **W**: *words* -> $\mathbf{R}^n$ mapping words in some language to a high dimensional vector (typically 200-500)

W("*cat*") = (0.2, −0.4, 0.7, ...)

W("*mat*") = (0.0, 0.6, −0.1, ...)

Word Representations

Traditional Method - Bag of Words Model	Word Embeddings
Uses one hot encoding - Each word in the vocabulary is represented by one bit position in a HUGE vector	Stores each word in as a point of space represented by a high dimensional vector
Difficult to scale	Unsupervised learning, can be built by simply reading a large corpus
Context information is not utilized	Has a deep dependence on context

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

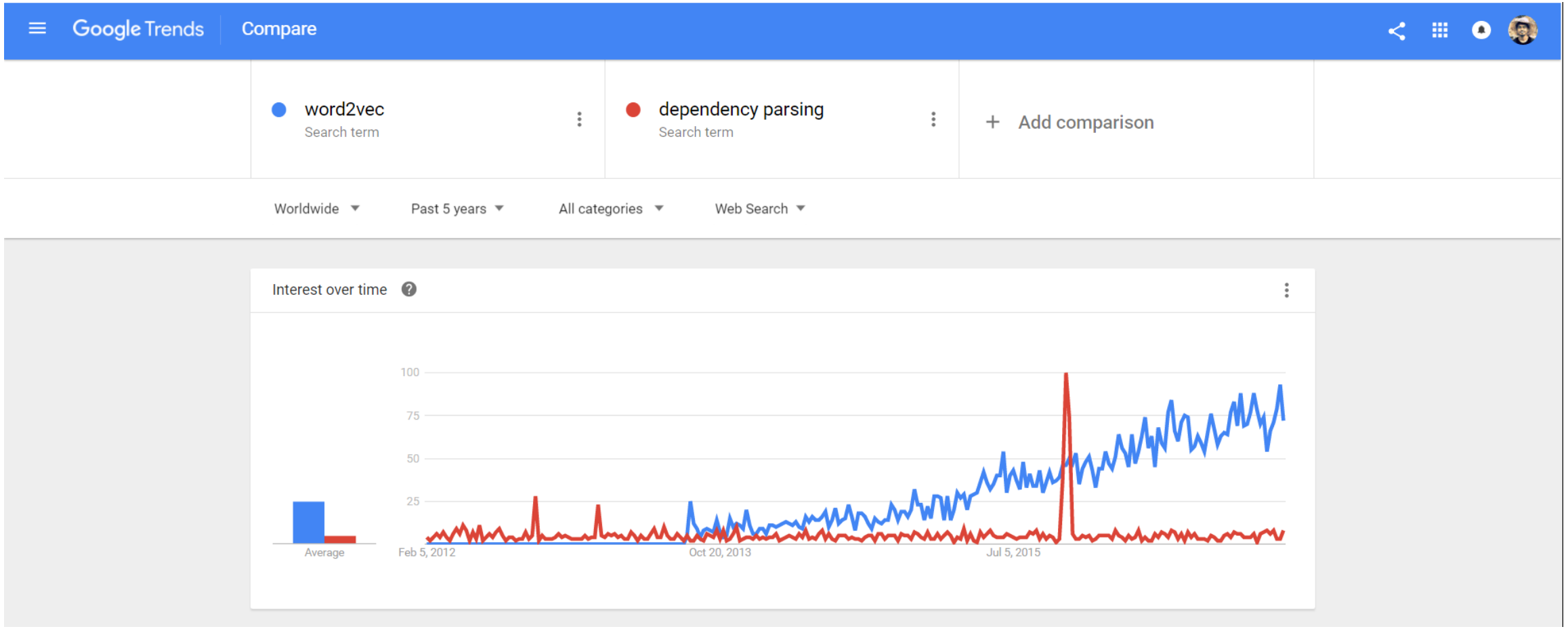
However Things are not that Simple …

- Extremely slow !
- Problems with scalability
- Poor quality embeddings

Enter Word2Vec: Neural Word Embeddings

- Learn word embeddings directly from the input data
- Use a neural network architecture
- Online, scalable to large data sets
- Implicitly factorizes the co-occurance matrix

Word2Vec Explosion



Word2Vec – As a black box

feed in text

dog = (0.13, -0.34, -0.92...)
cat = (0.15, -0.29, -0.90...)

chair = (-0.35, 0.77, -0.19...)

Wikipedia

Output is a |V| x d matrix W where each row is a vector for the word

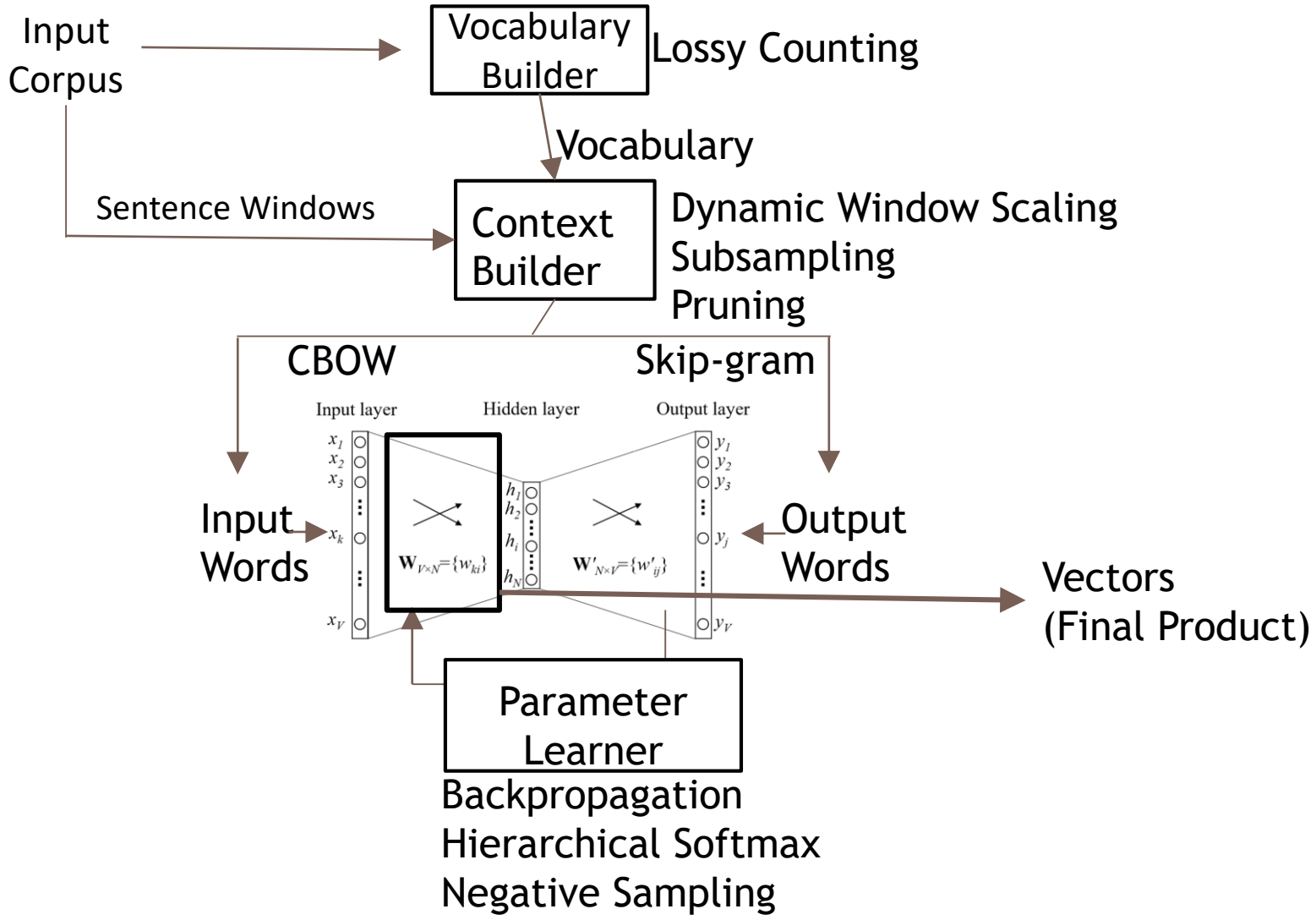
Word2Vec

- Dog
  - cat, dogs, dachshund, rabbit, puppy, poodle, rottweiler, mixed-breed, doberman, pig
- Sheep
  - cattle, goats, cows, chickens, sheeps, hogs, donkeys,herds, shorthorn, livestock
- November
  - october, december, april, june, february, july, september,january, august, march

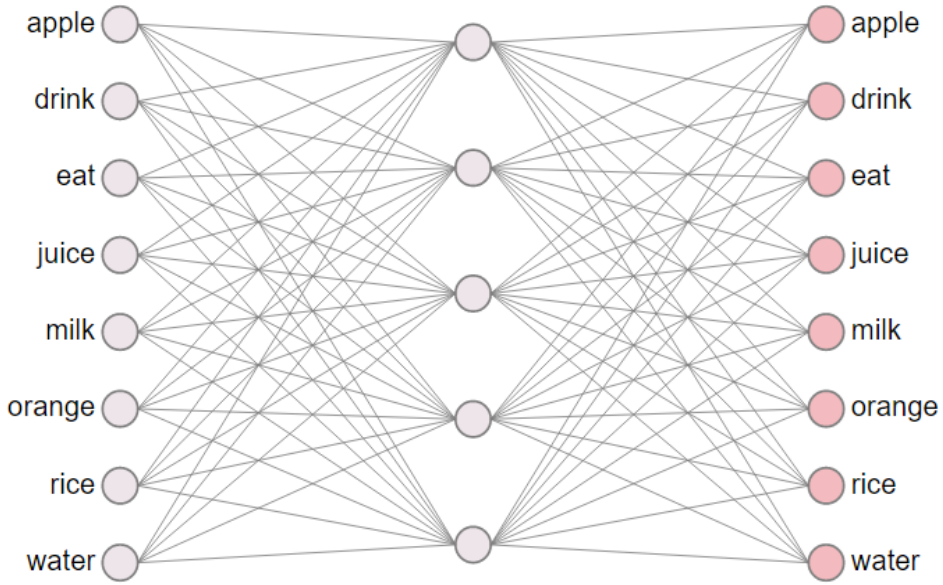
Main Idea

- Predict surrounding words of every word instead of capturing cooccurrence counts directly !
  - Eventually captures the cooccurrence metrics (but differently)
- Extremely fast !
- Can easily incorporate a new sentence/document or add a new word to the vocabulary

Word2Vec Decomposed



word2vec network



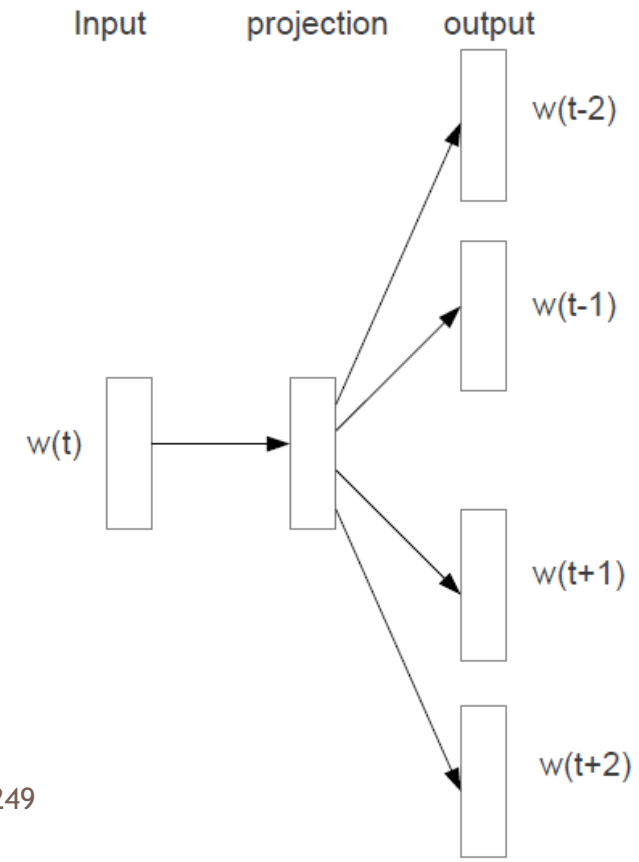
Structure Highlights:

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

Demo - Wevi

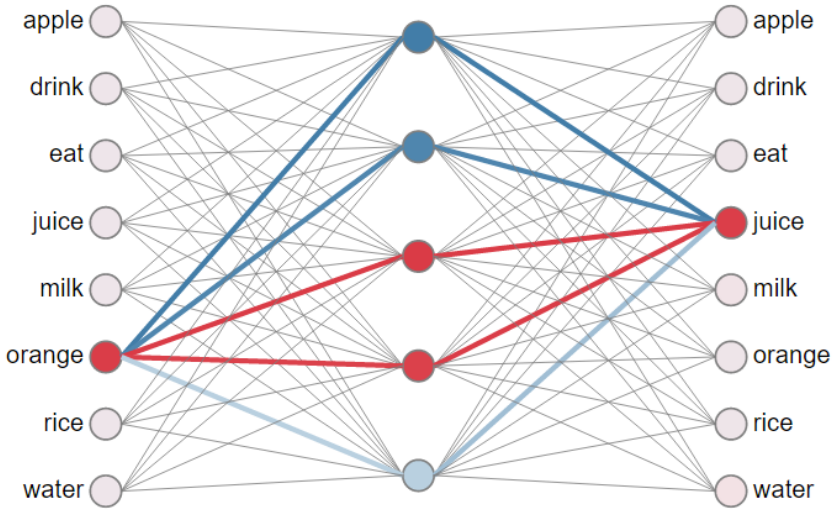
Skip-gram Architecture

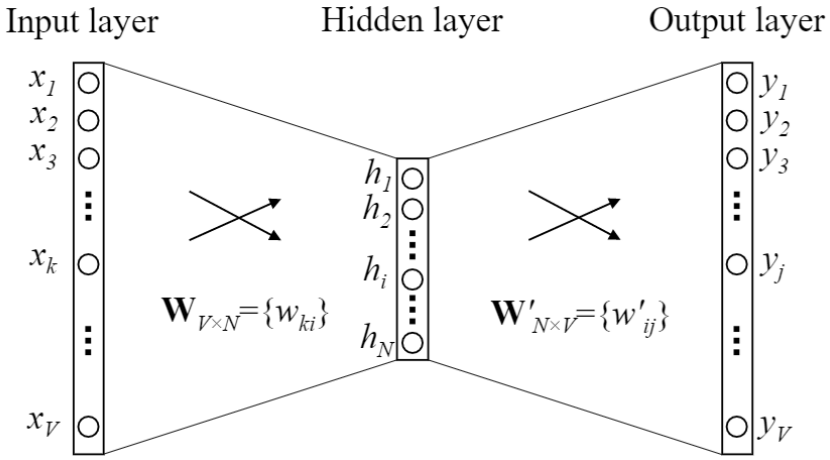
- Predicts the surrounding words with the current word



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

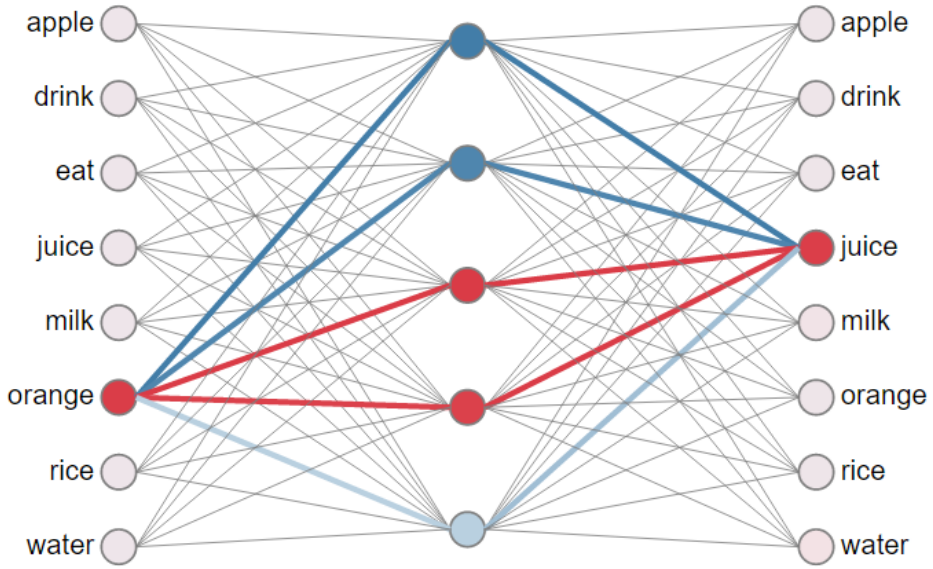
word2vec network

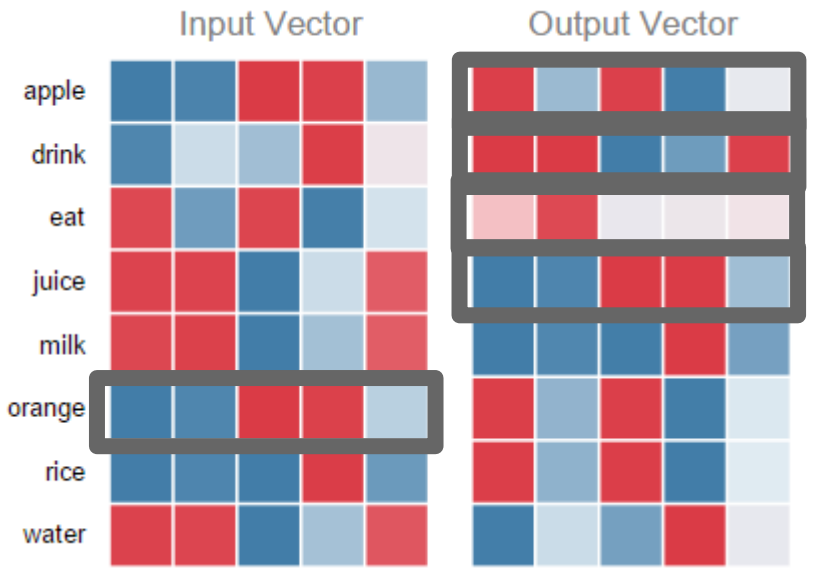




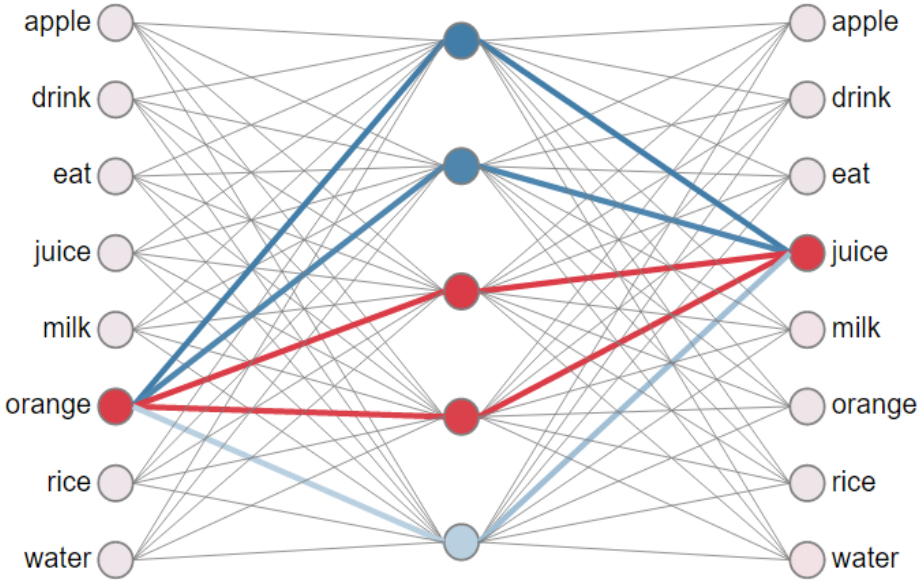
$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

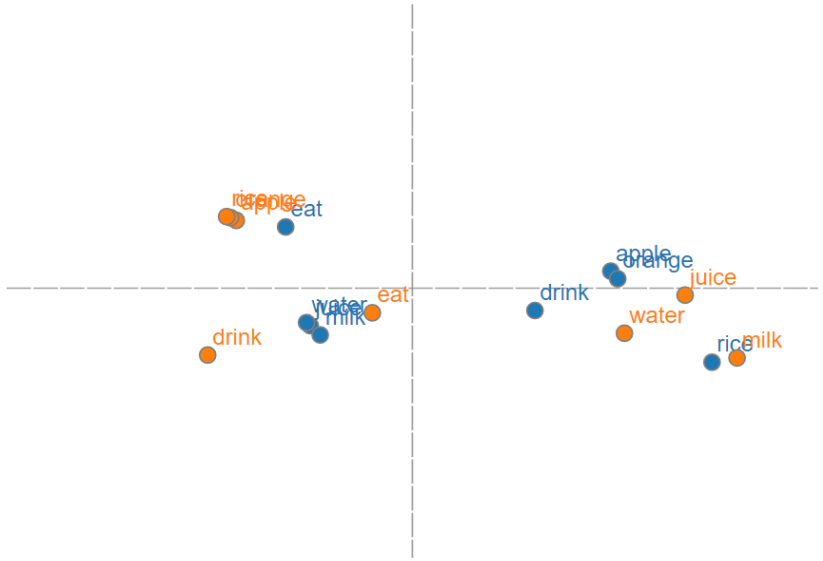
NN & Weight Matrices



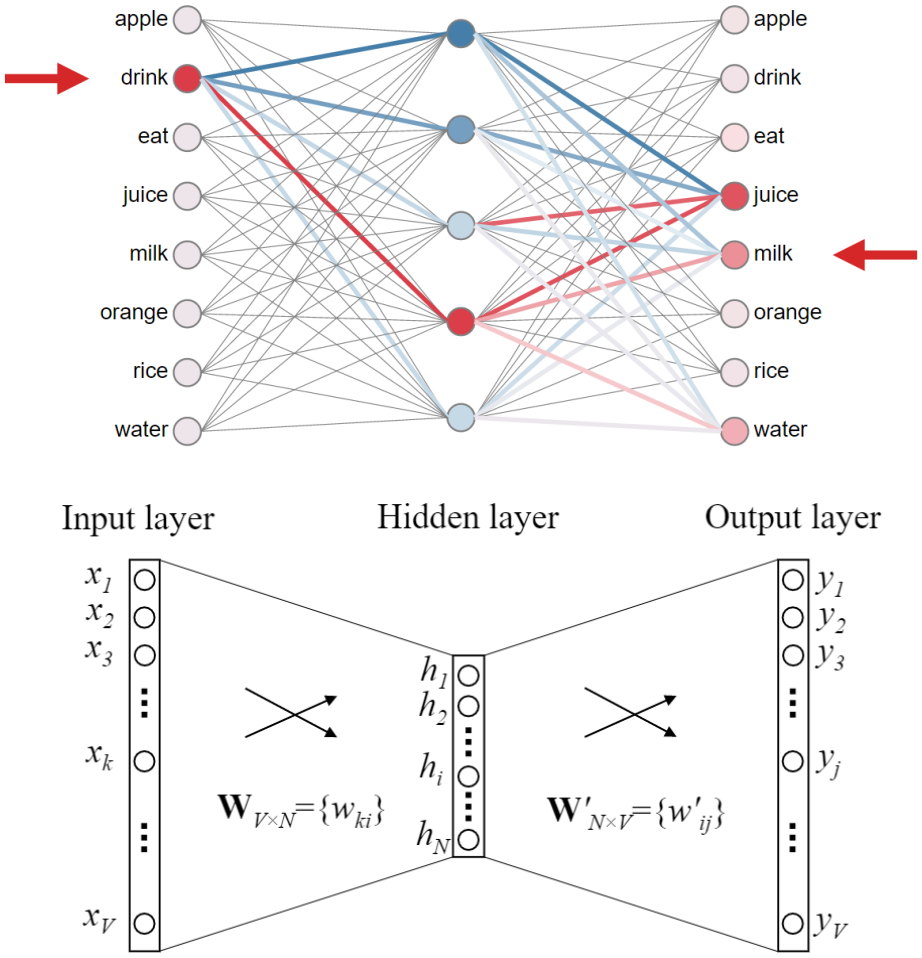


NN & Word Vectors





Training: updating word vectors



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

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

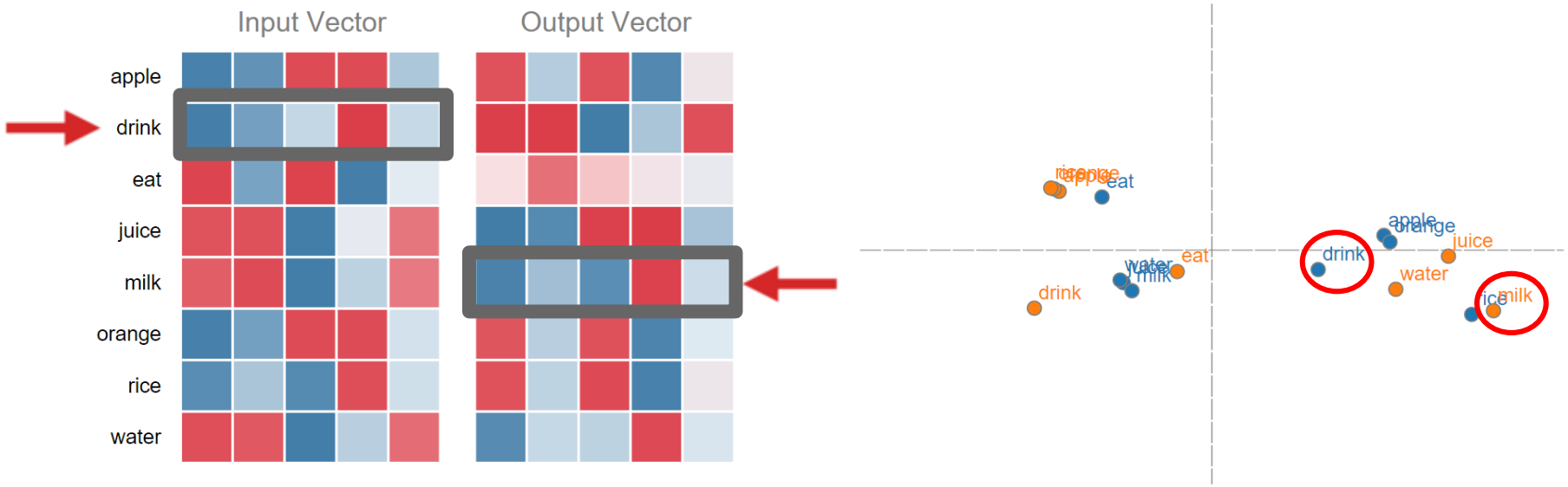
$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{u_j}{\partial w'_{ij}}$$

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

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

Intuition of output vector update rule

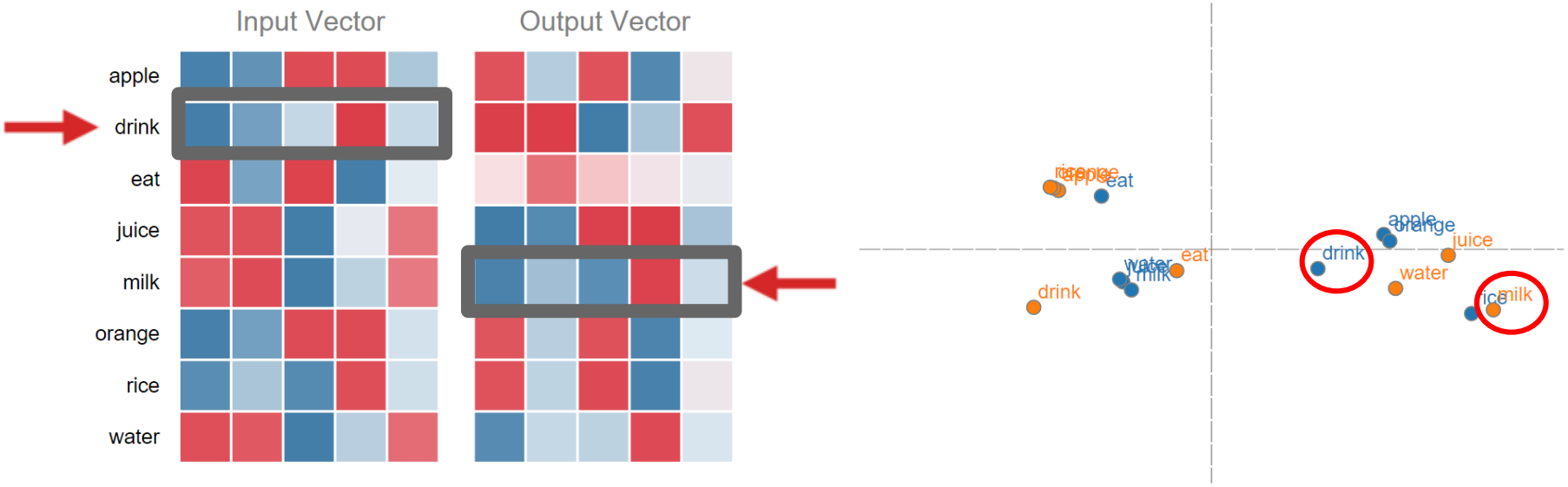
$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot e_j \cdot \mathbf{h}$$



Intuitive Understanding of Input Vectors

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

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

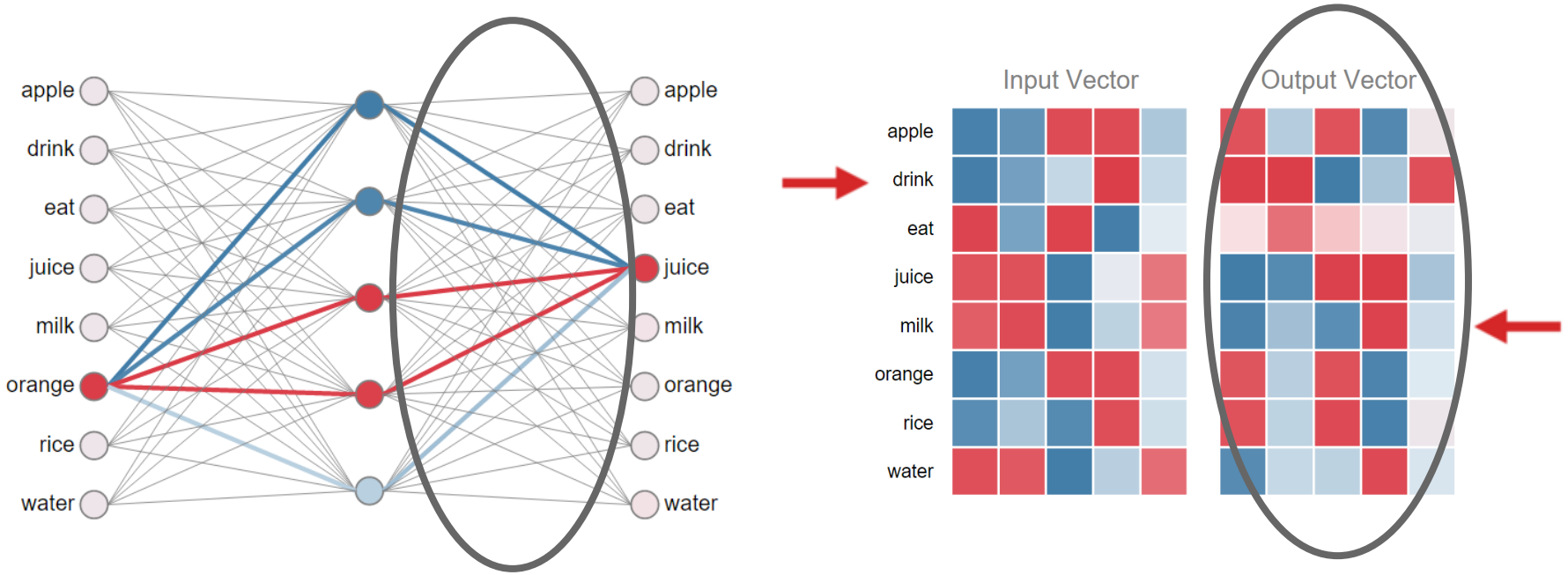


Optimizations:

- Softmax-based Approaches
  - Hierarchical Softmax
  - Differentiated Softmax
  - CNN-Softmax

- Sampling-based Approaches
  - Importance Sampling
  - Adaptive Importance Sampling
  - Target Sampling
  - Noise Contrastive Estimation
  - Negative Sampling
  - Self-Normalisation
  - Infrequent Normalisation

Training General Softmax Word2Vec is Intractable

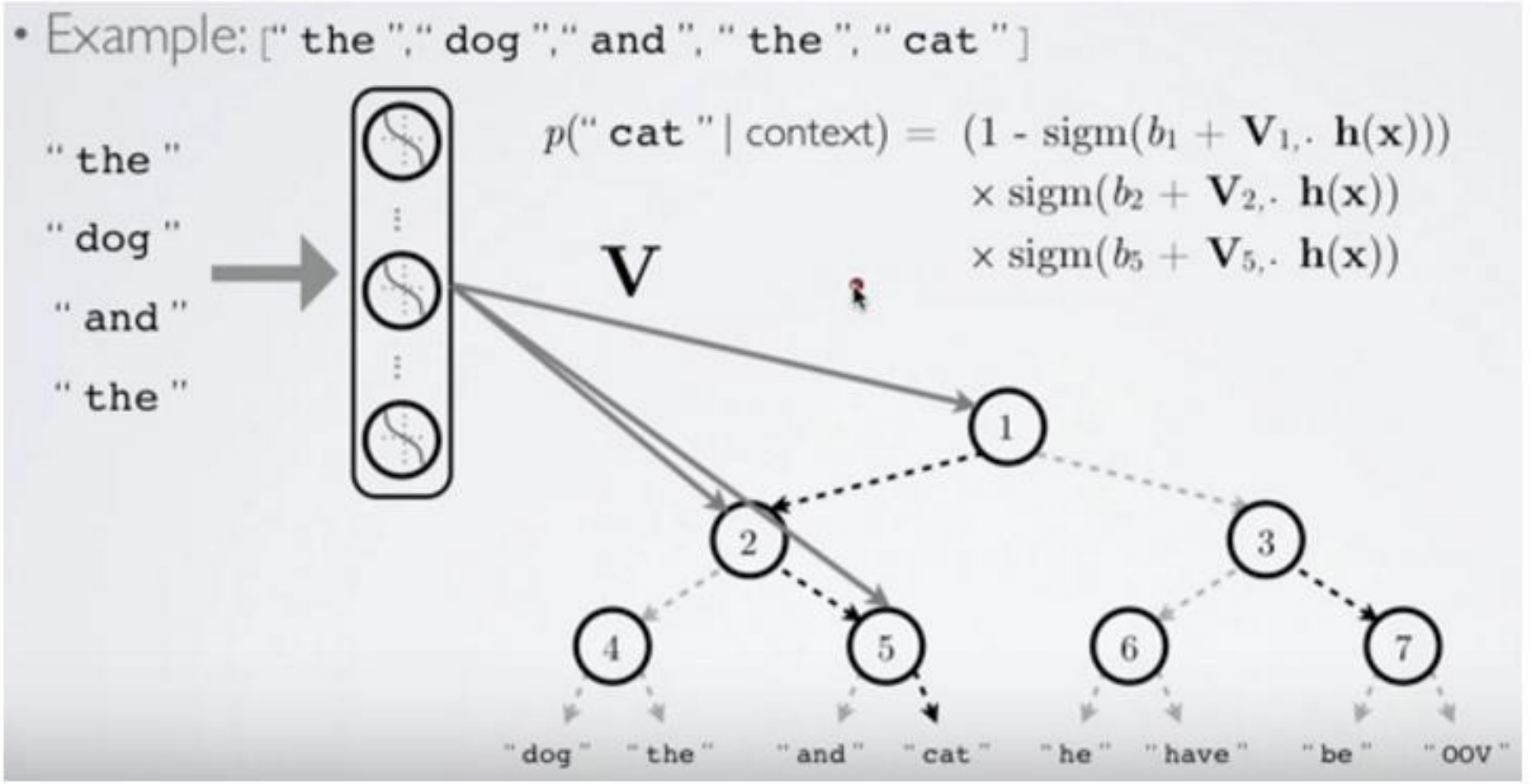


need to update every single output vector!

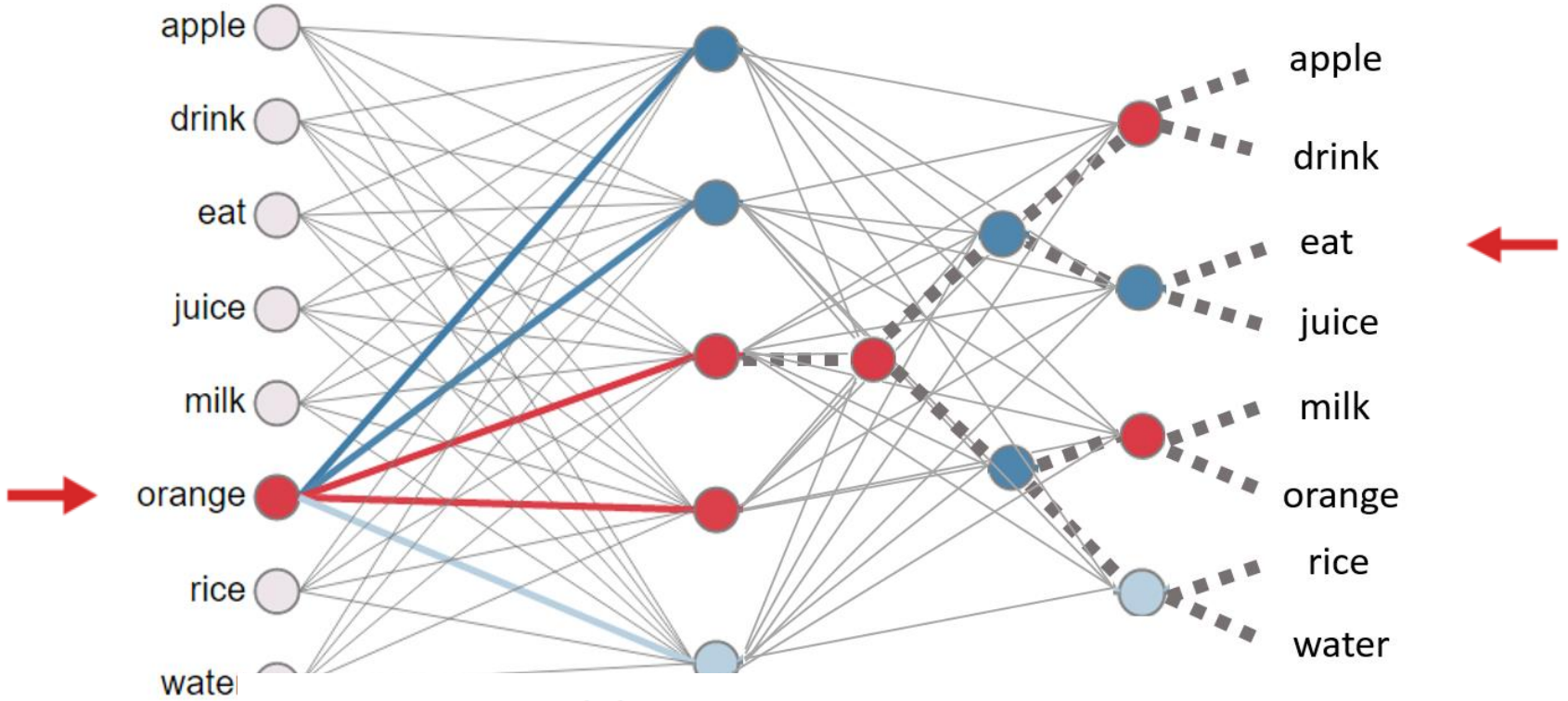
Hierarchical Softmax

- Proposed by Morin & Bengio'05
- Replaces the flat softmax layer with a hierarchical layer with words as leaves
- Yields speedups for word prediction tasks of at least 50x
- Critical for low-latency tasks such as real-time communication in Google's new messenger app Allo

Hierarchical Softmax



Hierarchical Softmax



$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma\left(\left[\!\left[n(w, j+1) = \mathrm{ch}(n(w, j))\right]\!\right] \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h}\right)$$

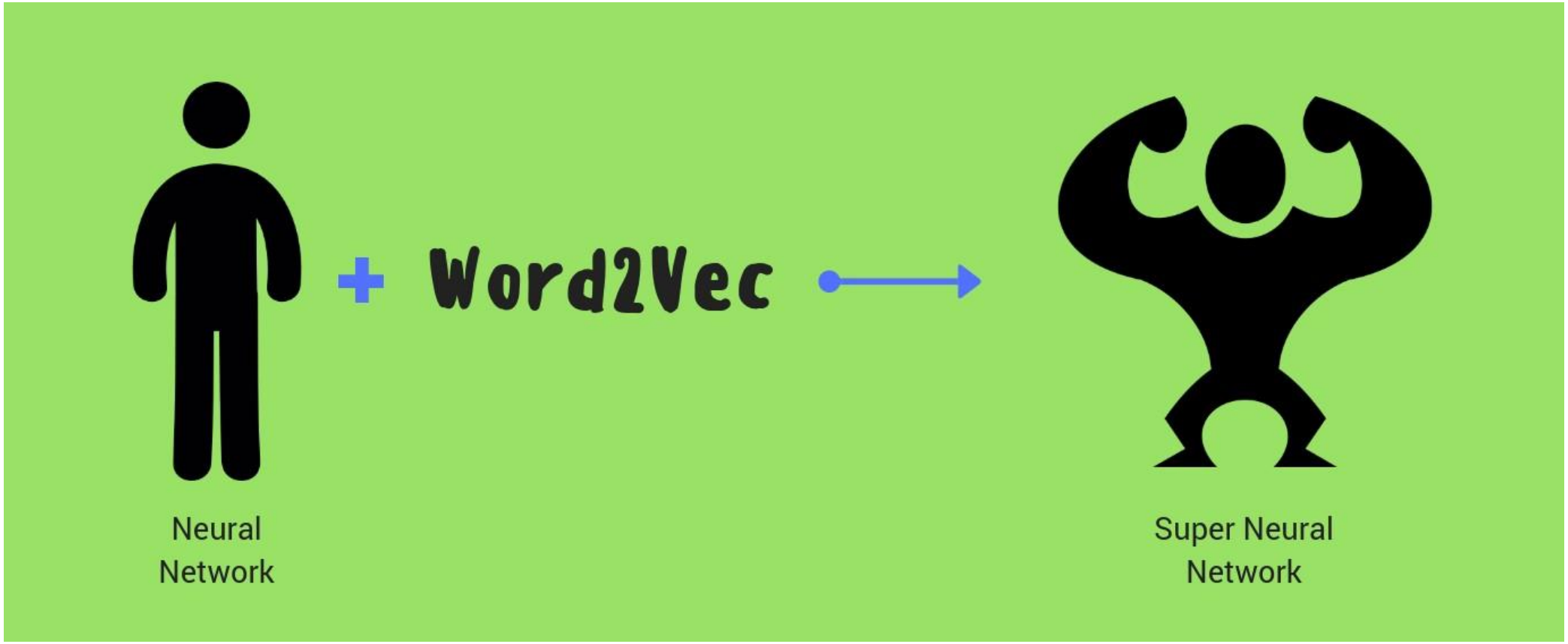
Negative Sampling

- Increase positive samples' probability while decrease negative samples' probability
  - Hidden assumption: decrease negative samples' probability means Increase positive samples' probability

- Use a probabilistic distribution ?

- Word2Vec uses -- $$E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h})$$

Word2Vec - the Steroids for Natural Language Processing



From Words to Phrases and Beyond

- Example query:
  - restaurants in mountain view that are not very good
- Forming the phrases:
  - restaurants in (mountain view) that are (not very good)
- Adding the vectors:
  - restaurants + in + (mountain view) + that + are + (not very good)
- Very simple and efficient
- Will not work well for long sentences or documents

Compositionality by Vector Addition

Expression	Nearest tokens
Czech + currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa, flag carrier Lufthansa
Russian + river	Moscow, Volga River, upriver, Russia
French + actress	Juliette Binoche, Vanessa Paradis, Charlotte Gainsbourg

Advantages

- Because word embeddings are a key element of deep learning models for NLP, it is generally assumed to belong to the same group. word2vec is not technically not be considered a component of deep learning, with the reasoning being that its architecture is neither deep nor uses non-linearities

- Here are two key benefits that these architectures have over Bengio's and the C&W model
  - They forgo the costly hidden layer.
  - They allow the language model to take additional context into account.
  - Preserves not only contextual relations, but also semantic relations between words

Applications

1. Word Similarity

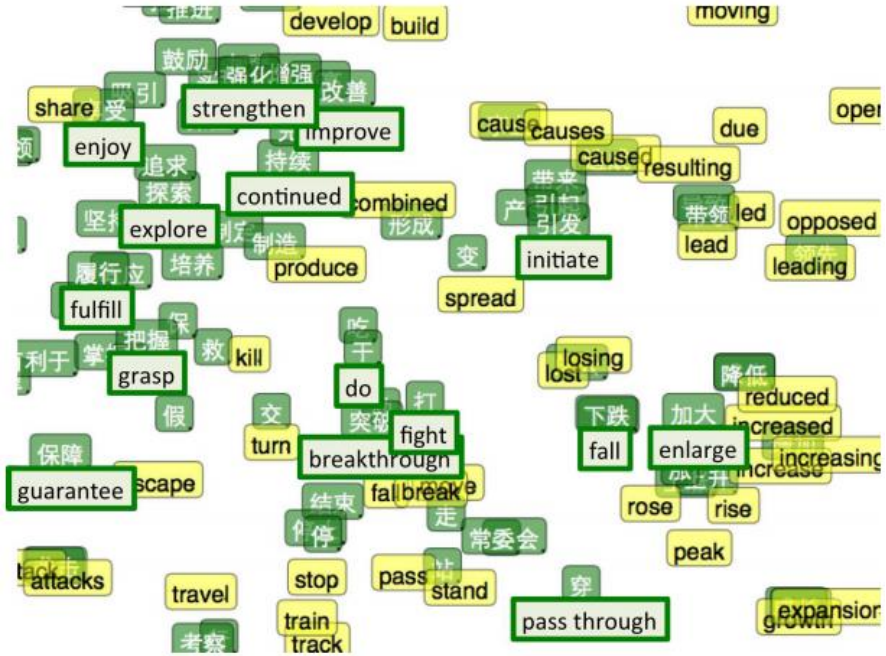
Classic Methods : Edit Distance, WordNet, Porter's Stemmer, Lemmatization using dictionaries

- Easily identifies similar words and synonyms since they occur in similar contexts
- Stemming (thought -> think)
- Inflections, Tense forms
- *eg. Think, thought, ponder, pondering,*
- *eg. Plane, Aircraft, Flight*

Applications

2. Machine Translation

Classic Methods : Rule-based machine translation, morphological transformation



Applications

3. Part-of-Speech and Named Entity Recognition

Classic Methods : Sequential Models (MEMM , Conditional Random Fields), Logistic Regression

	POS WSJ (acc.)	NER CoNLL (F1)
State-of-the-art*	97.24	89.31
Supervised NN	**96.37**	**81.47**
Unsupervised pre-training followed by supervised NN**	**97.20**	**88.87**
+ hand-crafted features***	97.29	89.59

Applications

4. Relation Extraction

Classic Methods : OpenIE, Linear programing models, Bootstrapping

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Applications

5. Sentiment Analysis

Classic Methods : Naive Bayes, Random Forests/SVM

- Classifying sentences as positive and negative
- Building sentiment lexicons using seed sentiment sets
- No need for classifiers, we can just use cosine distances to compare unseen reviews to known reviews.

```
Enter word or sentence (EXIT to break): sad

Word: sad  Position in vocabulary: 4067

                                              Word       Cosine distance
--------------------------------------------------------------------------
                                         saddening                 0.727309
                                               Sad                 0.661083
                                          saddened                 0.660439
                                      heartbreaking                0.657351
                                      disheartening                0.650732
                                      Meny_Friedman                0.648706
                                parishioner_Pat_Patello            0.647586
                                         saddens_me                0.640712
                                       distressing                 0.639909
                                  reminders_bobbing                0.635772
                                  Turkoman_Shiites                 0.635577
                                           saddest                 0.634551
                                       unfortunate                 0.627209
                                             sorry                 0.619405
                                       bittersweet                 0.617521
                                            tragic                 0.611279
                                         regretful                 0.603472
```

Applications

6. Co-reference Resolution

- Chaining entity mentions across multiple documents - can we find and unify the multiple contexts in which mentions occurs?

7. Clustering

- Words in the same class naturally occur in similar contexts, and this feature vector can directly be used with any conventional clustering algorithms (K-Means, agglomerative, etc). Human doesn't have to waste time hand-picking useful word features to cluster on.

8. Semantic Analysis of Documents

- Build word distributions for various topics, etc.

Linguistic Analogies through Word2Vec

- Word2Vec & other neural embeddings can actually lead to interesting geometries
- These patterns capture “relational similarities”
- Can be used to solve analogies:

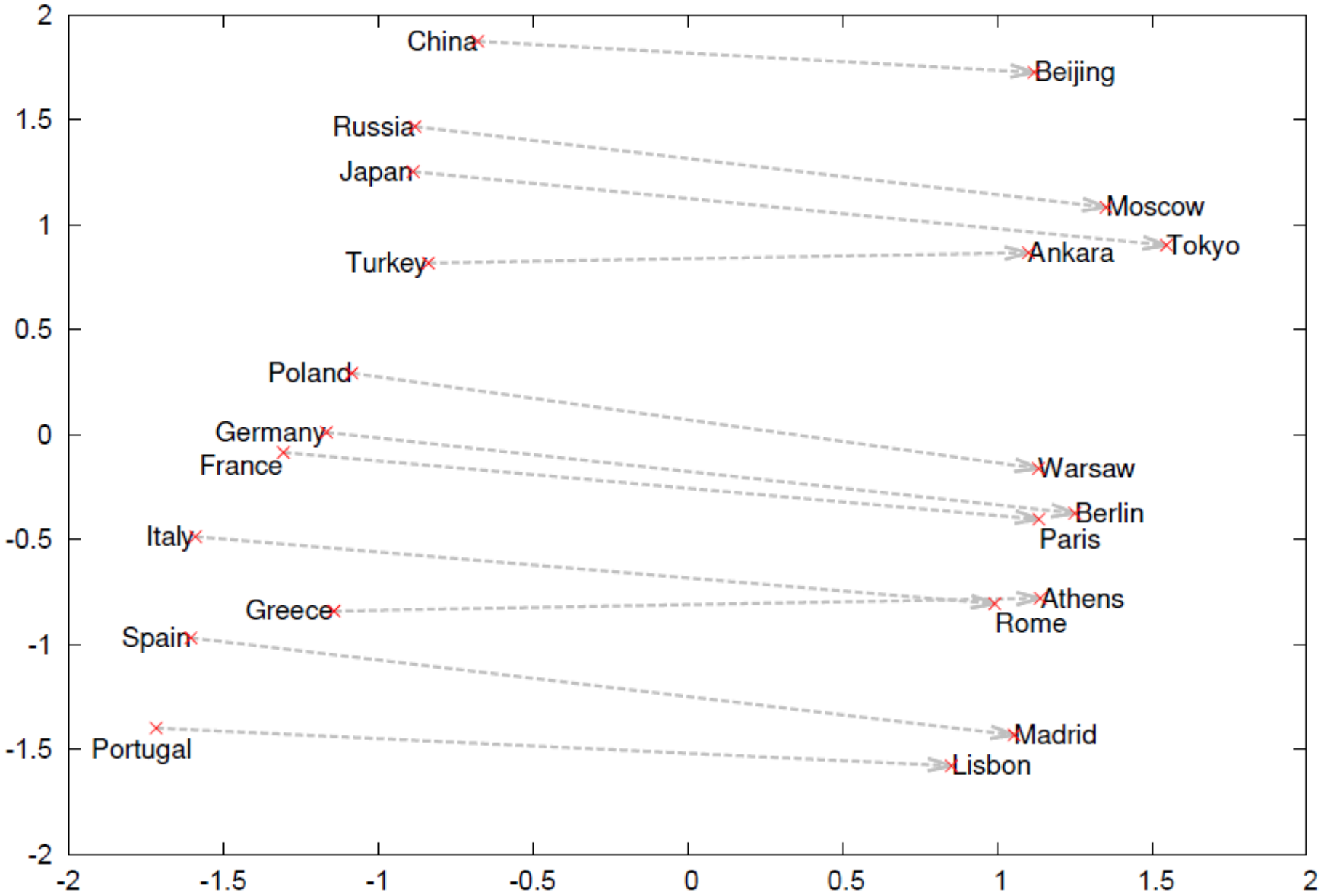
man is to woman as king is to queen

Or to generalize

a is to a^* as b is to b^*

- With simple vector arithmetic:

$$a - a^* = b - b^*$$


China
Beijing
Russia
Moscow
Japan
Tokyo
Turkey
Ankara
Poland
Warsaw
Germany
Berlin
France
Paris
Italy
Athens
Rome
Greece
Spain
Madrid
Portugal
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Some interesting examples

Tokyo − Japan + France = Paris

king − man + woman = queen

best − good + strong = strongest

Why does vector arithmetic reveal analogies ?

- We wish to find the closest x to $king - man + woman$
- This is done with cosine similarity:

$$\arg\max_x(\cos(x, king - man + woman)) =$$

$$\arg\max_x(\cos(x, king) - \cos(x, man) + \cos(x, woman))$$

vector arithmetic = similarity arithmetic

word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method

Yoav Goldberg and Omer Levy
{yoav.goldberg,omerlevy}@gmail.com

February 14, 2014

Why does this produce good representations ?

Good question. We don't really know.

The distributional hypothesis states that words in similar contexts have similar meanings. The objective above clearly tries to increase the quantity $v_w \cdot v_c$ for good word-context pairs, and decrease it for bad ones. Intuitively, this means that words that share many contexts will be similar to each other (note also that contexts sharing many words will also be similar to each other). This is, however, very hand-wavy.

Can we make this intuition more precise? We'd really like to see something more formal.

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)

Deep Walk: Online Learning of Social Representations

- **Perozzi B., Al-Rfou R., & Skiena S.,**

Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014

56

Presented by - Shubham Mittal

Data Sparsity – A Problem ?

- Data Sparsity is a double edged sword !
  - Enables the design of efficient discrete algorithms, but
  - Makes generalization in statistical learning a lot harder
- ML Applications in networks (like network classification, anomaly detection etc.) are continuously challenged with increasing data sparseness

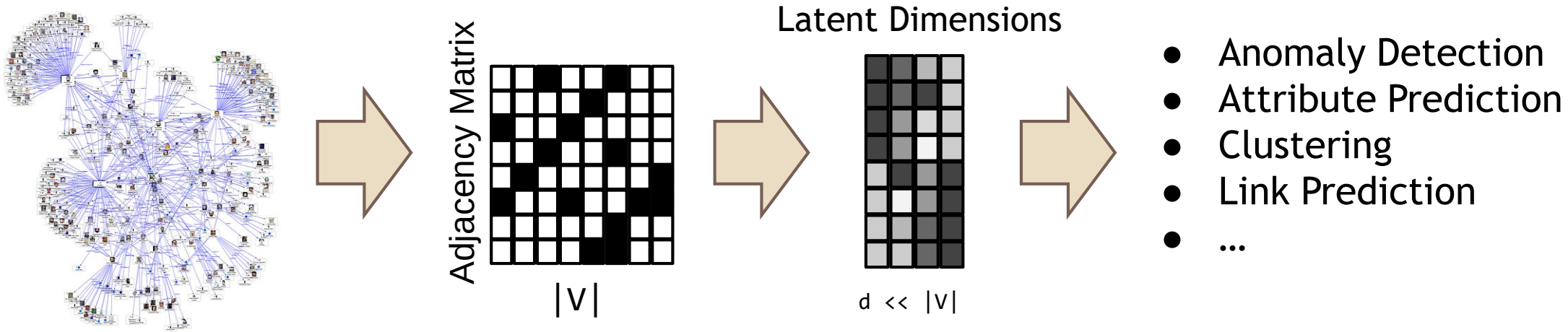
Motivation

- Learn social representations with the following characteristics:
  - Adaptability
  - Community Aware
  - Low Dimensional
  - Continuous

DeepWalk to the rescue

- Learns a latent representation of adjacency matrices using deep learning techniques developed for language modeling

- Get features from the graph by transforming the graph into a lower dimensional latent representation



Intuition

- How to extract information from the network?
- Solution – Use a stream of Random Walks
  - Local Exploration is now parallelizable
  - Accommodate small changes in the graphs without global re-computation
- Okay, but how do we represent this information ?

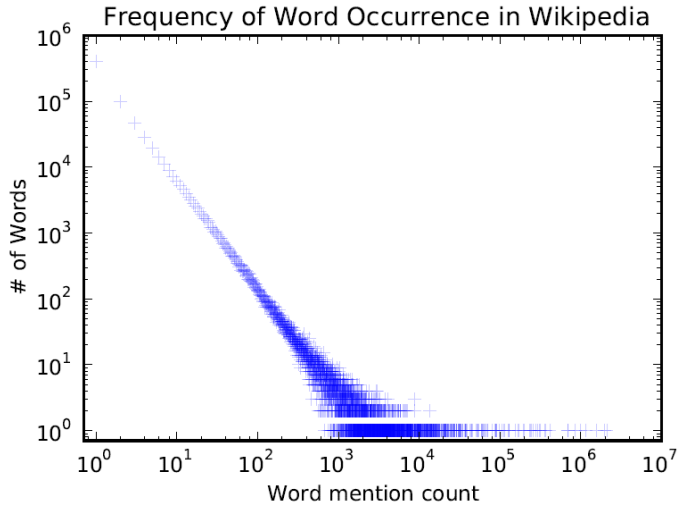
Word Frequency - Revisited

ctains open and the moon shining in on the
nd the cold , close moon " . And neither o
the night with the moon shining so bright
in the light of the moon . It all boils do
ly under a crescent moon , thrilled by ice
the seasons of the moon ? Home , alone ,
dazzling snow , the moon has risen full an
d the temple of the moon , driving out of

- Words frequency in a natural language corpus follows a power law.

Co-Occurrence Matrix

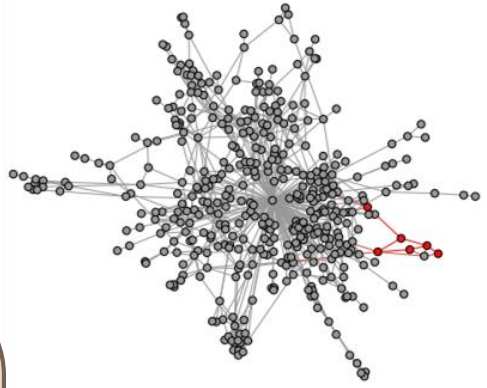
	planet	night	full	shadow	shine	crescent
moon	10	22	43	16	29	12
sun	14	10	4	15	45	0
dog	0	4	2	10	0	0



(b) Wikipedia Article Text

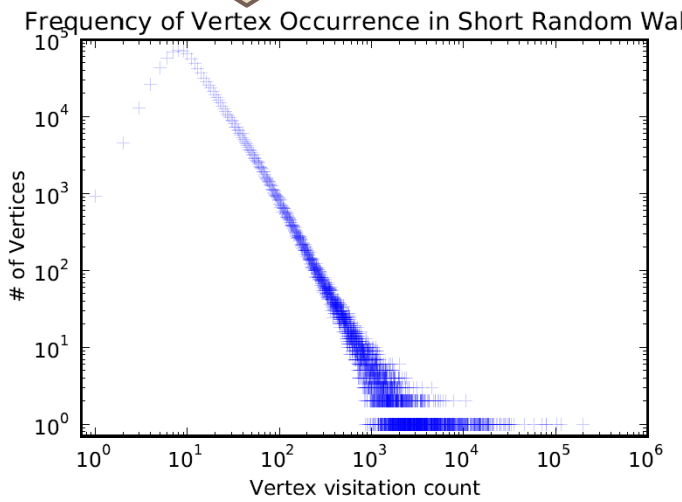
Connection: Power Laws

$$
\begin{array}{cccccc}
v_{71} \rightarrow & v_{24} \rightarrow & v_5 \rightarrow & v_1 \rightarrow & v_{17} \rightarrow & v_{80} \rightarrow \\
v_{92} \rightarrow & v_2 \rightarrow & v_3 \rightarrow & v_1 \rightarrow & v_{12} \rightarrow & v_{73} \rightarrow \\
v_{37} \rightarrow & v_{34} \rightarrow & v_9 \rightarrow & v_1 \rightarrow & v_{10} \rightarrow & v_{94} \rightarrow \\
v_{73} \rightarrow & v_{64} \rightarrow & v_5 \rightarrow & v_1 \rightarrow & v_{12} \rightarrow & v_1 \rightarrow \\
v_{75} \rightarrow & v_{14} \rightarrow & v_6 \rightarrow & v_1 \rightarrow & v_{13} \rightarrow & v_{61} \rightarrow
\end{array}
$$



Scale Free Graph

- Vertex frequency in random walks on scale free graphs also follows a ***power law***.
- Short truncated random walks are sentences in an artificial language!
- Random walk distance is known to be good features for many problems



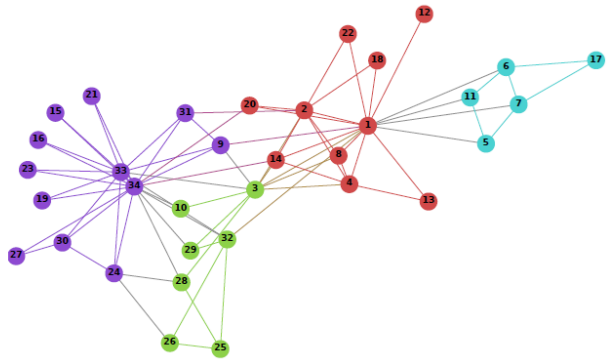
(a) YouTube Social Graph

The Core Idea

Short random walks in graphs = sentences in natural language

DeepWalk - The Big Picture

Input: Graph

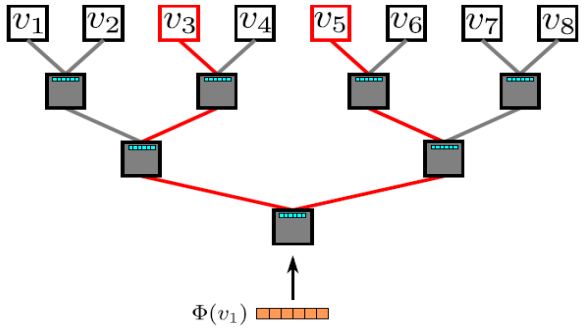


Random Walks

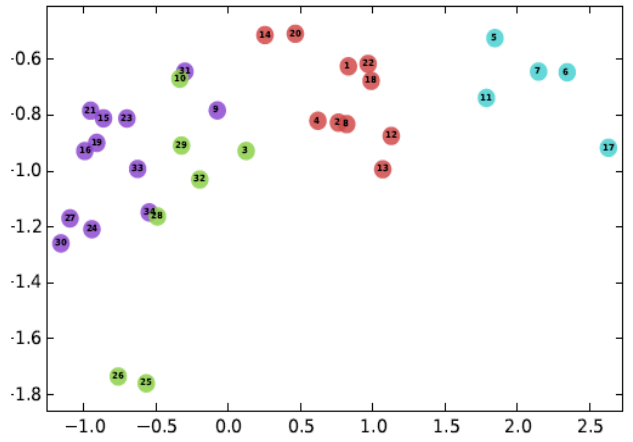
$\mathcal{W}_{v_4} = 4$

$u_k \begin{bmatrix} 3 \\ 1 \\ 5 \\ 1 \\ \vdots \end{bmatrix} v_j \rightarrow \Phi$ (Φ: d × j)

Representation Mapping



Hierarchical Softmax



Output: Representation

Results: BlogCatalog

Name	BlogCatalog
$\|V\|$	10,312
$\|E\|$	333,983
$\|\mathcal{Y}\|$	39
Labels	Interests

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	39.97	40.99	**41.66**	**42.42**	**42.62**
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wvRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
Macro-F1(%)	DeepWalk	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpectralClustering	19.14	23.57	**25.97**	**27.46**	**28.31**	**29.46**	**30.13**	**31.38**	**31.78**
	EdgeCluster	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

Table 2: Multi-label classification results in BlogCatalog

- DeepWalk performs well, especially when labels are sparse.

Results: Flickr

Name	Flickr
$\|V\|$	80,513
$\|E\|$	5,899,882
$\|\mathcal{Y}\|$	195
Labels	Groups

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**32.4**	**34.6**	**35.9**	**36.7**	**37.2**	**37.7**	**38.1**	**38.3**	**38.5**	**38.7**
	SpectralClustering	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
	EdgeCluster	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Modularity	22.75	25.29	27.3	27.6	28.05	29.33	29.43	28.89	29.17	29.2
	wvRN	17.7	14.43	15.72	20.97	19.83	19.42	19.22	21.25	22.51	22.73
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71
Macro-F1(%)	DeepWalk	**14.0**	17.3	**19.6**	**21.1**	**22.1**	**22.9**	**23.6**	**24.1**	**24.6**	**25.0**
	SpectralClustering	13.84	**17.49**	19.44	20.75	21.60	22.36	23.01	23.36	23.82	24.05
	EdgeCluster	10.52	14.10	15.91	16.72	18.01	18.54	19.54	20.18	20.78	20.85
	Modularity	10.21	13.37	15.24	15.11	16.14	16.64	17.02	17.1	17.14	17.12
	wvRN	1.53	2.46	2.91	3.47	4.95	5.56	5.82	6.59	8.00	7.26
	Majority	0.45	0.44	0.45	0.46	0.47	0.44	0.45	0.47	0.47	0.47

Table: Multi-label classification results in Flickr

Results: Youtube

Name	YouTube
$\|V\|$	1,138,499
$\|E\|$	2,990,443
$\|\mathcal{Y}\|$	47
Labels	Groups

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
Micro-F1(%)	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	26.79	29.18	33.1	32.88	35.76	37.38	38.21	37.75	38.68	39.42
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
	DeepWalk	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
Macro-F1(%)	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	13.15	15.78	19.66	20.9	23.31	25.43	27.08	26.48	28.33	28.89
	Majority	6.12	5.86	6.21	6.1	6.07	6.19	6.17	6.16	6.18	6.19

Key Take-Away from DeepWalk

Language Modeling techniques can be used for online learning of network representations.

Advantages of DeepWalk

- Effective - Encodes structural regularities in the graph
- Scalable - An online algorithm that does not use entire graph at once
- Generalization of language modelling - Walks as sentences metaphor
- Implementation available: bit.ly/deepwalk

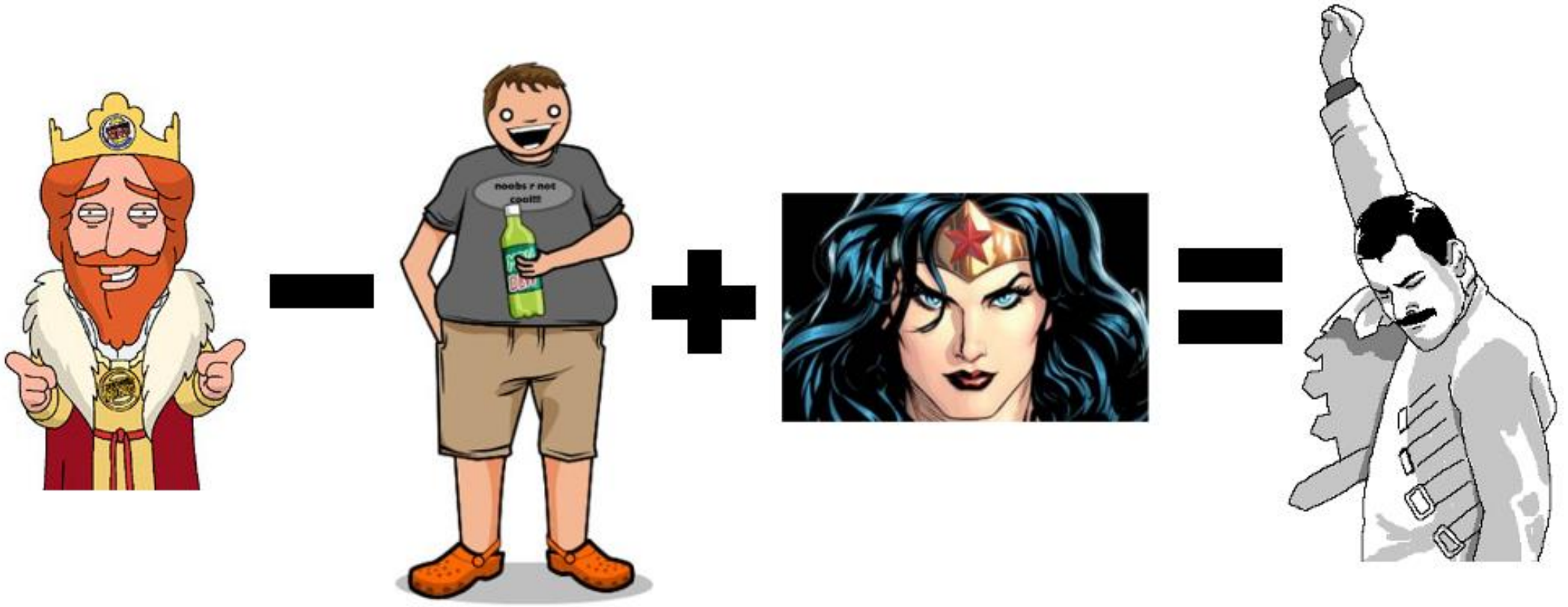
References

- Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.
- https://buss_jan.gitbooks.io/word2vec/content/chapter2.html
- https://medium.com/@mukulmalik/word2vec-part-1-fe2ec6514d70#.t2cqy1cky
- http://blog.aylien.com/overview-word-embeddings-history-word2vec-cbow-glove/
- Goldberg, Yoav, and Omer Levy. "word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method." arXiv preprint arXiv:1402.3722 (2014). APA
- Rong, Xin. "word2vec parameter learning explained." arXiv preprint arXiv:1411.2738 (2014).
- Hugo Lachorelle's Hierarchical Softmax – https://www.youtube.com/watch?v=B95LTf2rVWM

References

- Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014.
- Blog - http://sebastianruder.com/word-embeddings-1/
- http://mathinsight.org/scale_free_network

Thanks − for + listening = you :)


noobs r not cool!!!