NATURAL LANGUAGE PROCESSING

UNIT-2

Vector Semantics

feedback/corrections: vibha@pesu.pes.edu

VIBHA MASTI

Define Words by their Usage

- Words defined by environment (distributionalist)
- Synonyms: identical environments
  - unknown concept 'Ong choi'
    - Ong choi is delicious **sautéed with garlic**.
    - Ong choi is superb **over rice**
    - Ong choi **leaves** with salty sauces
  - Environments seen before
    - ...spinach **sautéed with garlic over rice**
    - Chard stems and **leaves** are **delicious**
    - Collard greens and other **salty** leafy greens
  - Conclusion: ong choi is a leafy green

Vector Semantics

- Word: point in multi-dimensional space
- Vector for representing word: embedding

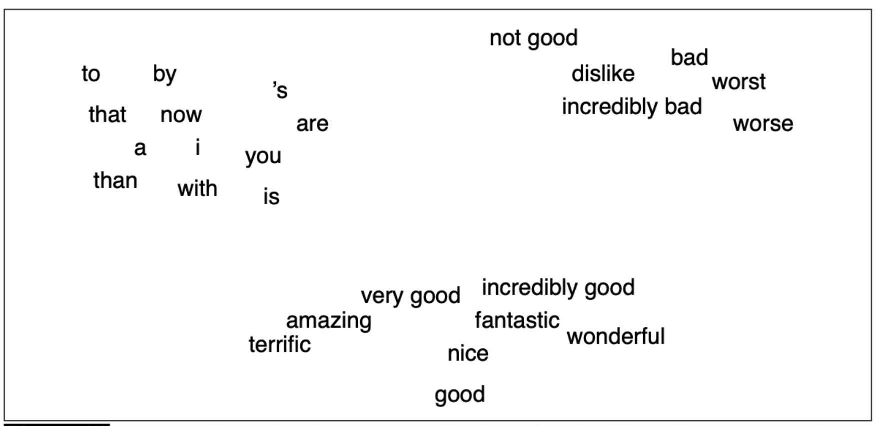


Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015).

- Words with similar meaning: closer in space
- Sentiment analysis works better with unseen words
- ICLR workshop 2016 – Li et al.

EMBEDDINGS

1. TF-IDF (Term Frequency-Inverse Document Frequency)

- Words represented as function of the counts of nearby words
- Sparse vectors

2. Word2Vec

- Train classifier to predict whether a word is likely to appear nearby
- Dense vectors

TF-IDF

- t : term / word
- d : document
- N: size of corpus / no. of documents
- corpus: set of documents

https://arxiv.org/pdf/1512.08183.pdf

- TF = frequency of words in a document, normalized by total words in document

  (range − [0, 1])

$$tf(t, d) = \frac{\text{count of } t \text{ in } d}{\text{no. of words in } d}$$

also $= \log_{10}(\text{count}(t, d) + 1)$ use this

- DF = document frequency − no. of documents in which a term appears (normalized by total no. of documents − optional)

$df(t)$ = documents with t in it

$$df_{norm}(t) = \frac{df}{N}$$

- IDF = dampened inverse of DF

$$idf(t) = \log\left(\frac{N}{df + 1}\right)$$

- TF-IDF

$$\text{tf-idf}(t, d) = tf(t, d) * idf(t)$$

Q: consider the following 3 documents

D1: text mining is to find useful information from text

D2: useful information from text is mined

D3: dark came

Document-word matrix

	text	mining	is	to	find	useful	information	from	text	mined	dark	came
D1	1	1	1	1	1	1	1	1	1	0	0	0
D2	0	0	1	0	0	1	1	1	1	1	0	0
D3	0	0	0	0	0	0	0	0	0	0	1	1

$tf(text, D1) = 1$

$df(text) = 2$

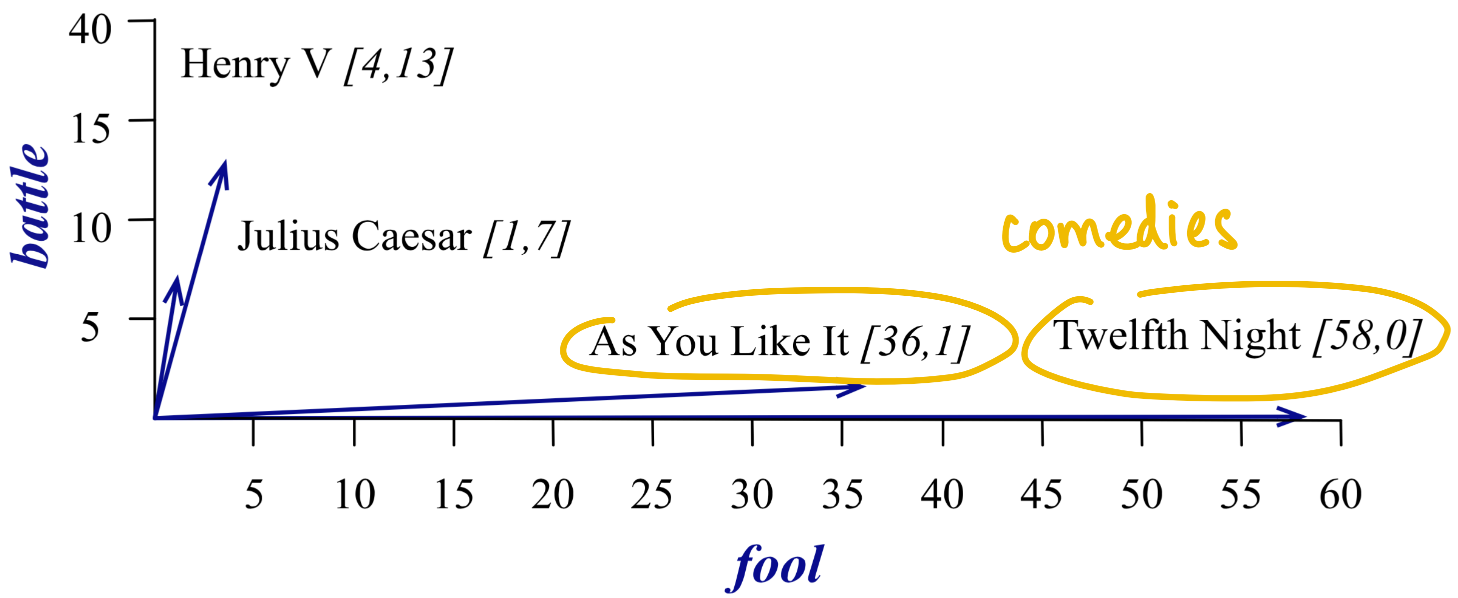
$$idf(text) = \log\left(\frac{3}{df+1}\right) = \log\left(\frac{3}{3}\right) = 0$$

$\therefore$ $tf\text{-}idf(text, D1) = 0$

Term-Document Matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.



battle = [1, 0, 7, 13]
good = [114, 80, 62, 89]

Word-Word Matrix

- No. of times row word and column word occur in some context
- Window size of W (context)

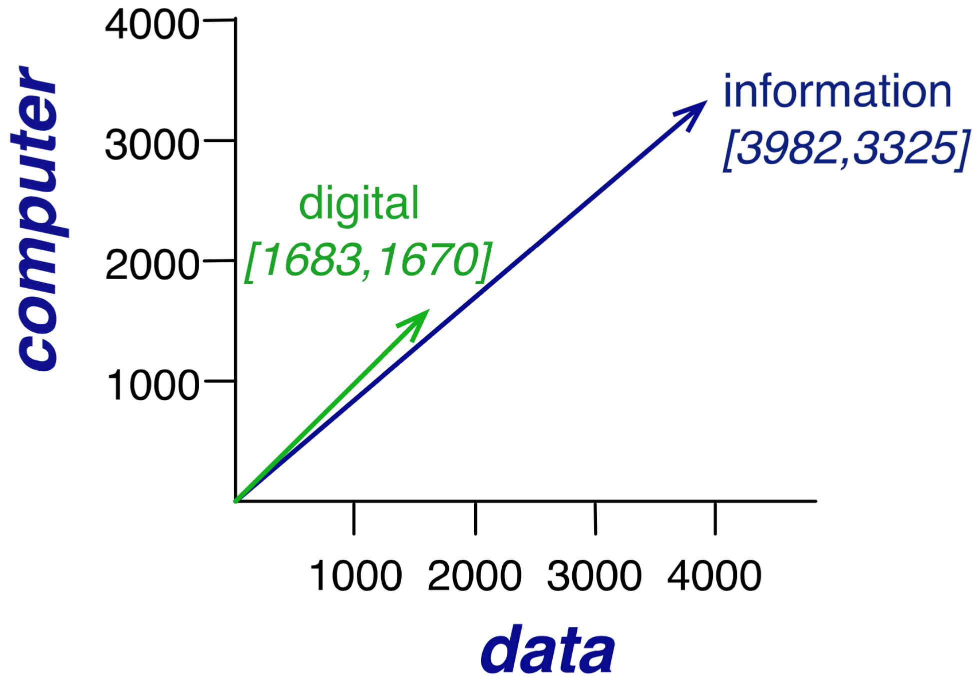
- No. of times column word occurs in a ±W word window around row word

- Eg: wikipedia corpus

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for digital is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.



A spatial visualization of word vectors for digital and information, showing just two of the dimensions, corresponding to the words data and computer.

Cosine Similarity

- Between 2 words

$$\cos(\vec{w}, \vec{v}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \, \|\vec{w}\|}$$

PMI

- Pointwise mutual information
- $\frac{\text{Probability of co-occurrence}}{\text{Independent occ. probabilities}}$
- Do words x & y co-occur more than if they were independent?

$$PMI(x,y) = \log_2\left(\frac{P(x,y)}{P(x)\,P(y)}\right)$$

range $(-\infty, \infty)$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

PMI(digital, computer) = ?

$$P(\text{digital, computer}) = \frac{1670}{11716} = 0.1425$$

$$P(\text{digital}) = \frac{3447}{11716} = 0.2942$$

$$P(\text{computer}) = \frac{4997}{11716} = 0.4265$$

$$PMI = \log_2\left(\frac{0.1425}{0.2942 \times 0.4265}\right) = 0.1836$$

$$PPMI = \max(0, PMI)$$

Weighting PMI

$$PPMI_\alpha(w, c) = \max\left(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0\right)$$

$$P_\alpha(c) = \frac{\text{count}(c)^\alpha}{\sum_x \text{count}(x)^\alpha} \qquad (\alpha = 0.75)$$

Dense Vectors

DISTRIBUTIONAL METHODS

- word's meaning related to distribution of words around it

- A bottle of *tezgüino* is on the table.

Consider the following sentences which are there with the above sentence.

> Everybody likes *tezgüino.*
>
> *Tezgüino makes you drunk.*
>
> *We make tezgüino out of corn.*

So from above sentences, the meaning of tezgüino can be " A fermented alcoholic drink made of corn"

- Word w : vector of features
- feature f_i : for a word v_i, does the word w occur in its neighbourhood/context?

w = tezgüino $\qquad \vec{w} = (f_1, f_2, f_3, \ldots, f_N)$

v_1 = bottle $\Rightarrow f_1 = 1$
v_2 = drunk $\Rightarrow f_2 = 1$
v_3 = matrix $\Rightarrow f_3 = 0$
$\vdots$

$\therefore \vec{w} = (1, 1, 0, \ldots)$

Sparse vs Dense Vectors

- tf-idf : long, sparse
- alternative : short, dense

TYPES of VECTOR MODELS

1. Sparse vector representations

(a) Word co-occurrence matrices

2. Dense vector representations

(b) SVD and LSA

(c) Neural net inspired models – skip-grams, CBOW, word2vec, glove

(d) Brown clusters

- Distributional hypothesis: similarity of meaning correlates with similarity of distribution

DISTRIBUTIONAL SEMANTIC MODELS (DSMs)

- General purpose semantic models that can be applied to various tasks
- Co-occurrence matrix
- Eg:
  1. Hyperspace Analogue to Language
  2. LSA
  3. Syntax-based DSM using dependency relation

Co-Occurrence Matrix

- Row: word
- Column: context (window/paragraph/doc)

	d_1	d_2	d_3	d_4	d_5
dog	88	92	11	1	2
lion	57	28	3	0	0
bark	80	62	10	0	1
car	0	1	0	93	97
tire	2	0	2	80	72
drive	0	1	0	90	45

Association

- $PMI(w, c) = \log_2\left(\frac{P(w,c)}{P(w)P(c)}\right)$
- word with context

Similarity

- cosine similarity

DIMENSIONALITY REDUCTION

- SVD

$$M_{m \times n} = U_{m \times m} \Sigma_{m \times n} (V_{n \times n})^T$$

- Select k top singular values and obtain lower dimensional vectors

- Truncated M

$$M_{reduced} = U_{m \times k} \Sigma_{k \times k} (V_{n \times k})^T$$

LATENT SEMANTIC ANALYSIS

- Most effective if count matrix transformed before SVD (PMI)

- Bullinaria and Levy, 2007 : PPMI-based LSA vectors solve TOEFL MCQs (word similarity) with 85% accuracy

- Each word represented as vector (relating to contexts / documents)

Corpus - titles of 9 technical memoranda:

(1) 5 about HCI
(2) 4 about graph theory

Example of text data: Titles of Some Technical Memos

c1: *Human* machine *interface* for ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user* perceived *response time* to error measurement

m1: The generation of random, binary, ordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

- Term-document matrix

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

sim(human, user) = 0
sim(human, minors) = 0

(can also do for doc similarity)

•

```python
import numpy as np
term_doc_counts = np.array([
    [1,0,0,1,0,0,0,0,0],
    [1,0,1,0,0,0,0,0,0],
    [1,1,0,0,0,0,0,0,0],
    [0,1,1,0,1,0,0,0,0],
    [0,1,1,2,0,0,0,0,0],
    [0,1,0,0,1,0,0,0,0],
    [0,1,0,0,1,0,0,0,0],
    [0,0,1,1,0,0,0,0,0],
    [0,1,0,0,0,0,0,0,1],
    [0,0,0,0,0,1,1,1,0],
    [0,0,0,0,0,0,1,1,1],
    [0,0,0,0,0,0,0,1,1]
])
u, s, vt = np.linalg.svd(term_doc_counts, full_matrices=True)

# Taking only the k=2 most important features
up, sp, vtp = u[:, 0:2], np.diag(s[0:2]), vt[0:2, :]

# Estimate the new term-document matrix
lsa_term_doc = up @ sp @ vtp
```

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.4	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.4	0.16	-0.03	-0.07	-0.1	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.7	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.2	-0.11
survey	0.1	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.3	0.2	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.1	-0.21	0.15	0.22	0.5	0.71	0.62

sim(human, user) = 0.89
sim(human, minors) = -0.28

https://colab.research.google.com/drive/1qqBkvhaQjkBI70H9W2b8cU4gISjXZdZb?usp=sharing

Documents with 2 concepts:

- Computer Science (CS)
- Medical Documents (MD)

$$
\begin{array}{c} \\ \text{CS} \\ \\ \\ \text{MD} \\ \\ \end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{data} & \text{inf.} & \text{retrieval} & \text{brain} & \text{lung} \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

Term-Document Matrix
Row: 1 Document
Columns: 1 Term

Document-Concept Similarity Matrix
Row: 1 Document
Columns: 1 Concept

Concept Strength Matrix
Row: 1 Concept

Term-Concept Matrix
Row: 1 Concept
Column: 1 Term

Distributional Representation

- captures linguistic distribution of each word in form of a high-dimensional numeric vector
- typically based on co-occurrence counts (count models)
- based on distributional hypothesis: similar distribution ~ similar meaning (similar distribution = similar representation)

Distributed Representation

- sub-symbolic, compact representation of words as dense numeric vectors
- meaning is captured in different dimensions and it is used to predict words (predict models)
- similarity of vectors corresponds to similarity of of the words
- word embeddings

WORD EMBEDDINGS

1. One-hot encodings

- Each vector is almost entirely 0's
- Eg: vocabulary size = 4

  words: apple (1, 0, 0, 0)

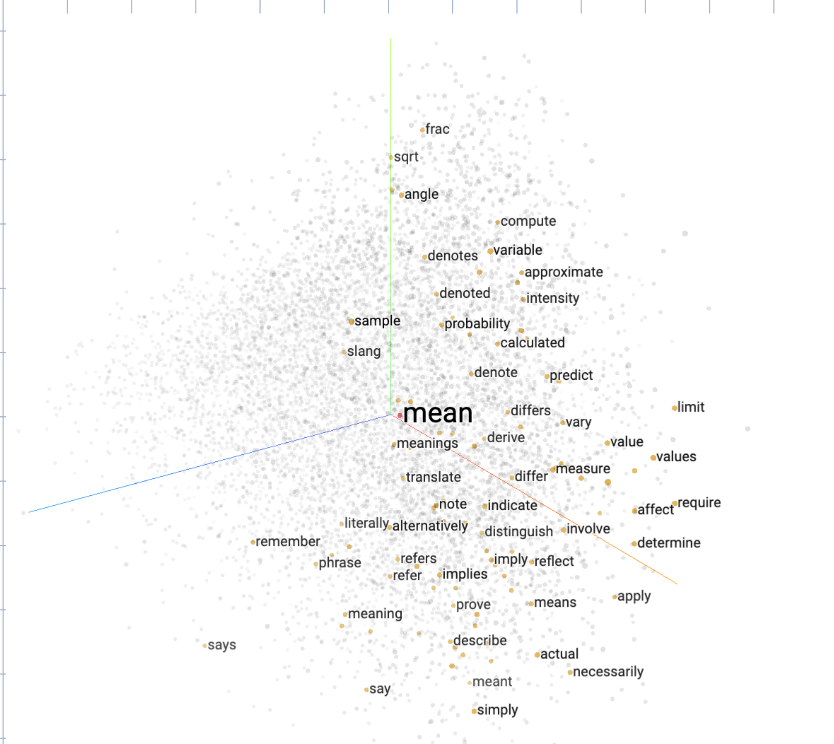
  car (0, 1, 0, 0)

  fruit (0, 0, 1, 0)

  doll (0, 0, 0, 1)

2. Distributional representation

- Could use context / documents



http://projector.tensorflow.org

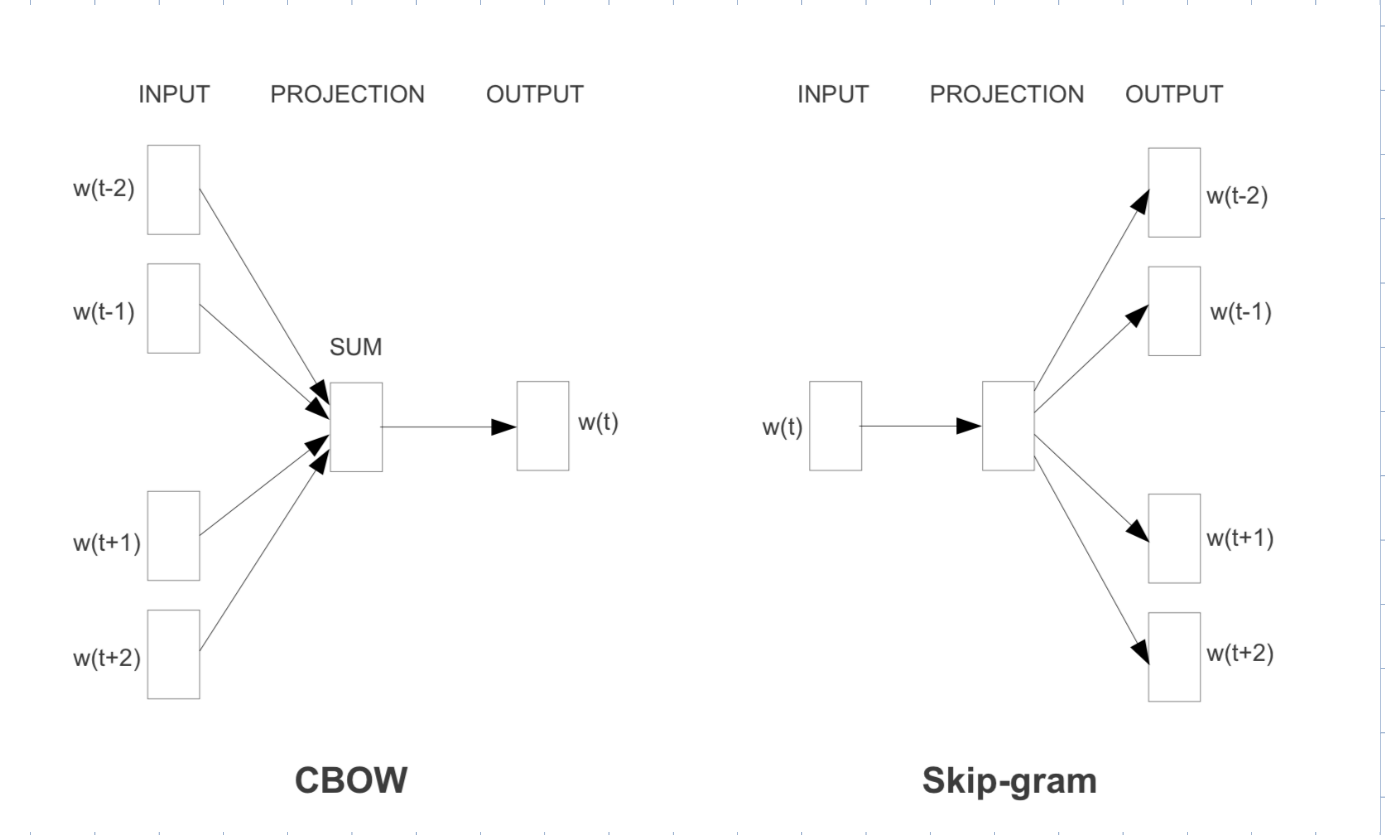
3. word2vec

- Method to compute vector representations (open source version is there)

- 2 diff/possible techniques:
  1. Continuous Bag of Words (CBOW)
  2. Skip-gram model

- Both are shallow NNs

- Static embeddings



- No activation functions in hidden layers

CBOW

- Assume we use 4 context words (2 before, 2 after)

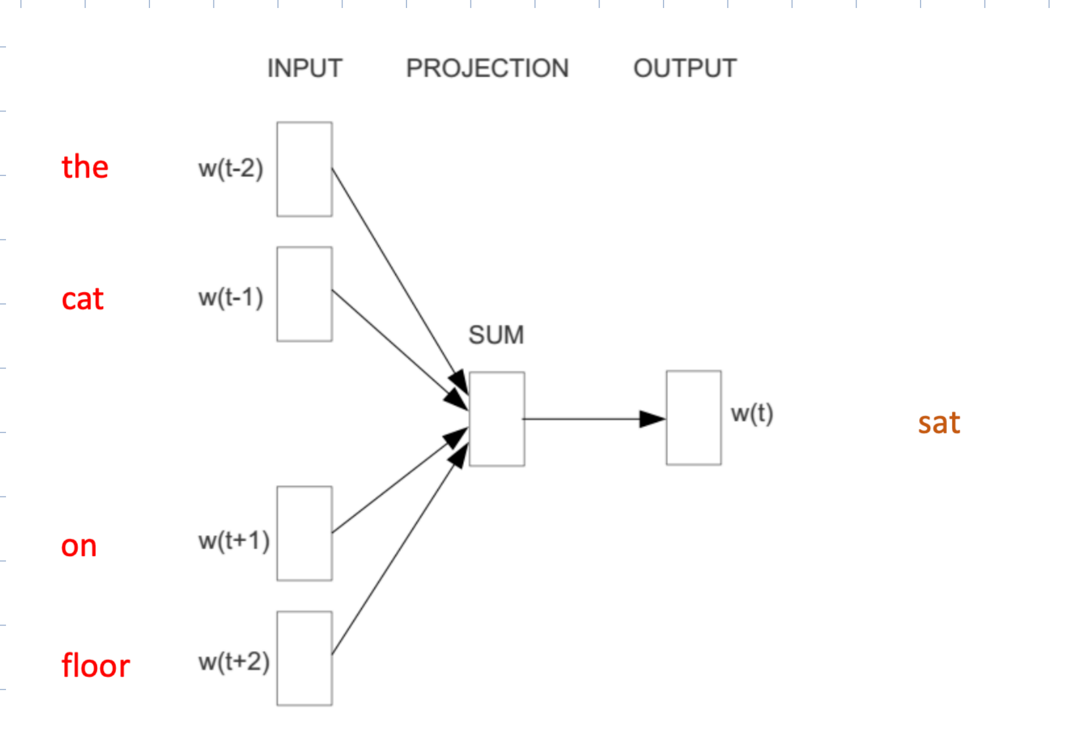
  sentence:

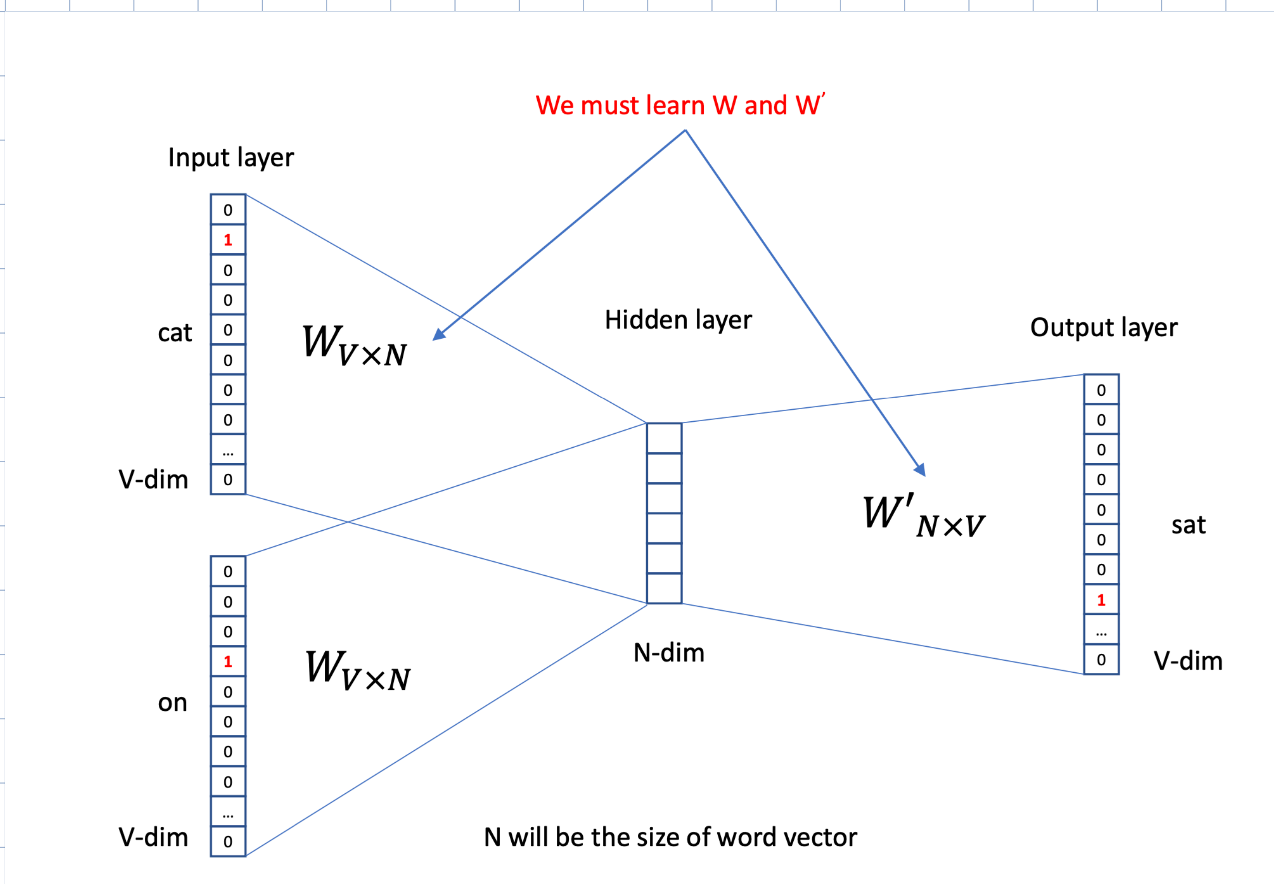
  we eat pancakes every morning

- Let target = pancakes
- context words = [we, eat, every, morning]
- Pass each word as vector to NN (think sequentially and avg) (as one-hot encoding)
- Multiply by weight matrix to hidden layer and multiply again by weight matrix 2 to get output layer
- Element-wise summation for all 4 context word outputs
- Final softmax activation
- Multiplying just selects one column (if words are one-hot columns)

Eg: "the cat sat on floor"

window = ± 2 (4)





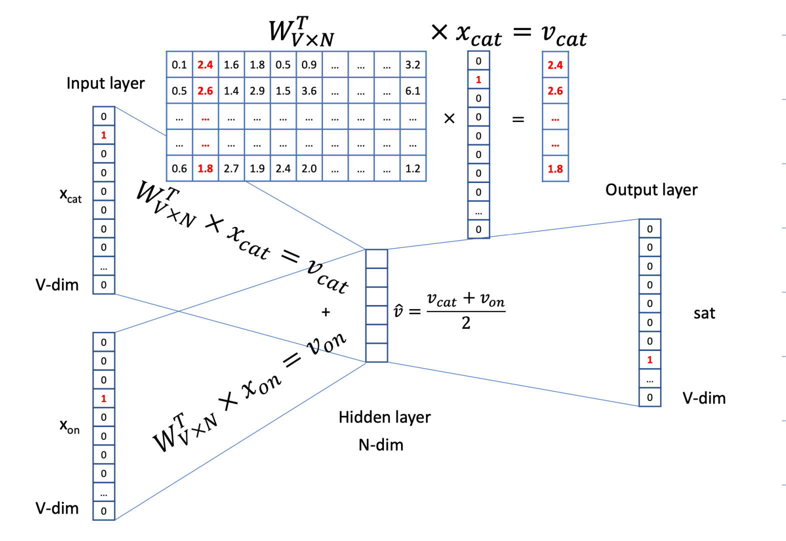
v_{cat} represents embedding for cat

v_{on} represents embedding for on

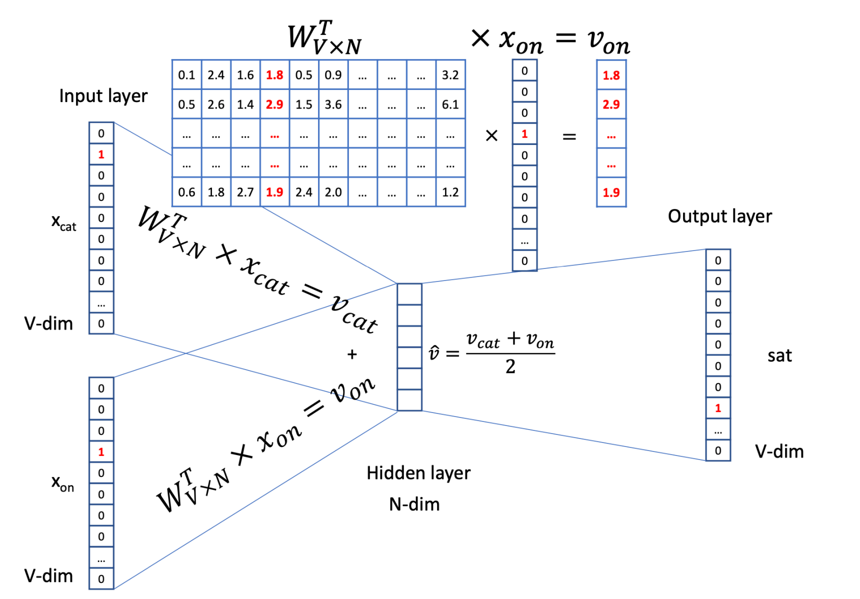
N: dimensions of embedding

Input to Hidden Layer

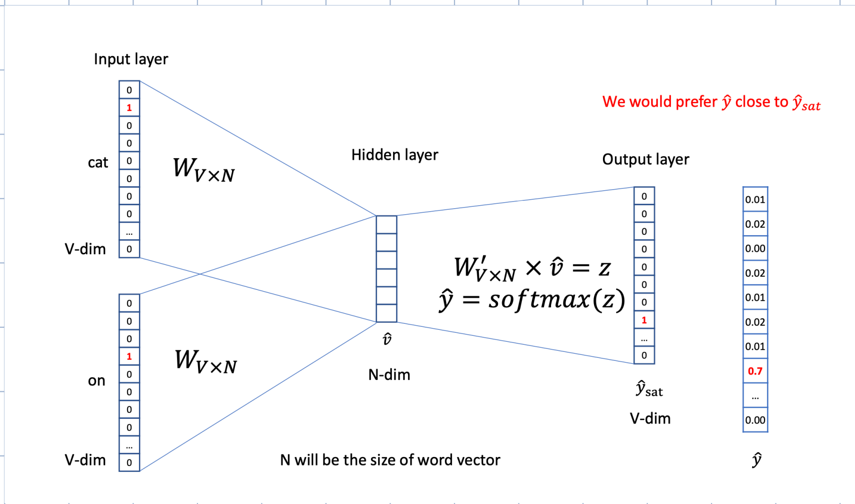
(a) cat



(b) on

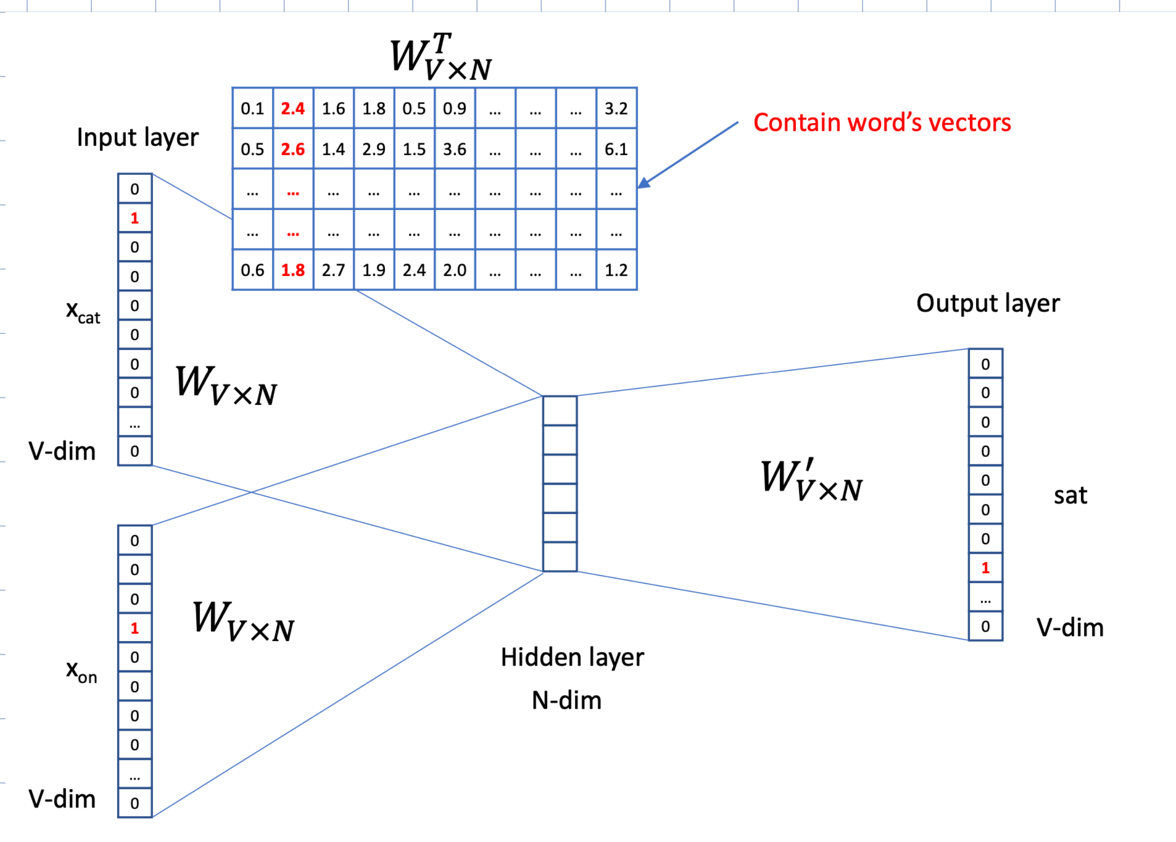


Hidden to Output Layer



- Backpropagate

Word Embeddings – Rows of $W_{V\times N}$ (Context)



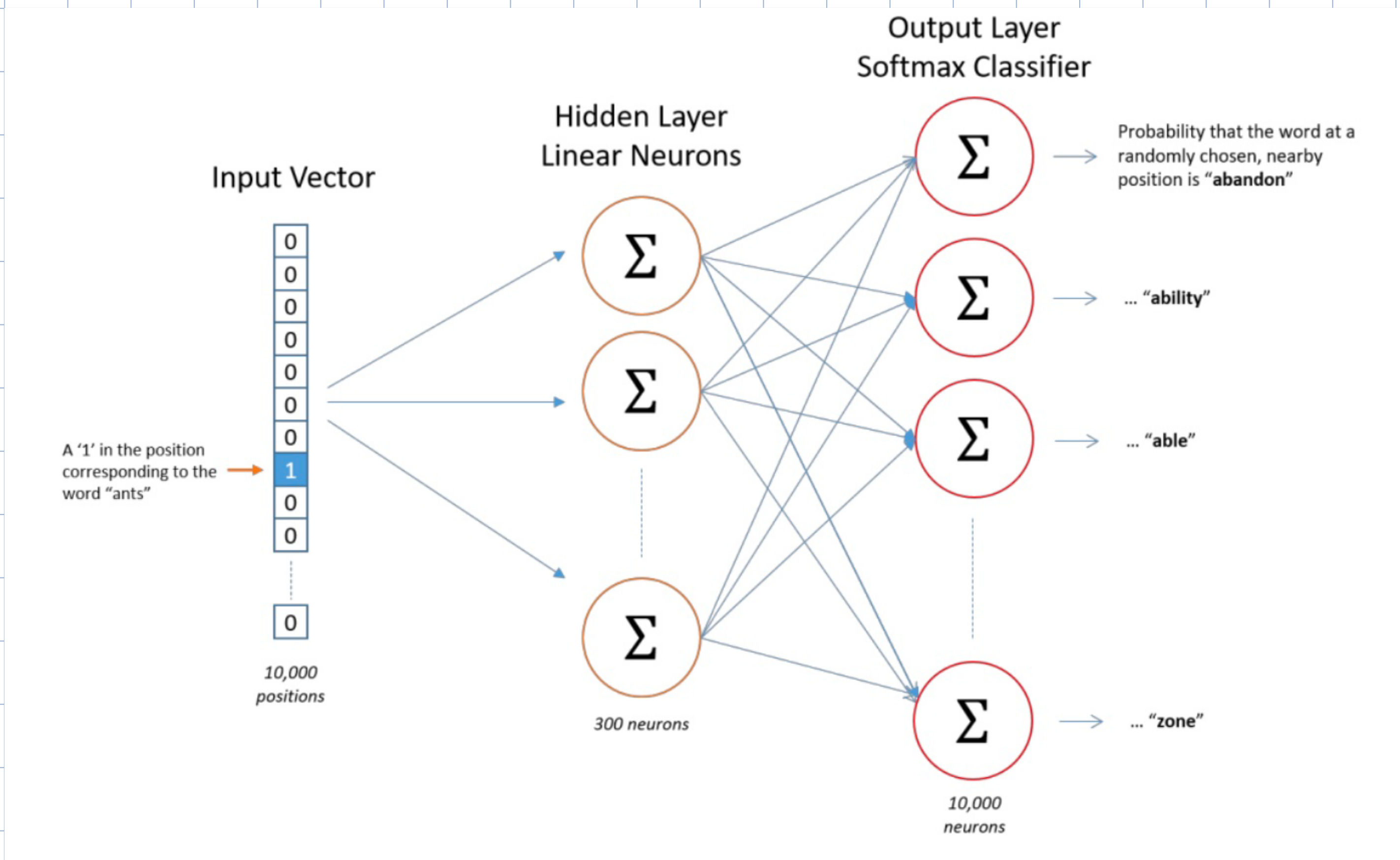
We can consider either W or W' as the word's representation. Or even take the average.

Word Embedding of Target word

- Average of embeddings of context words

Skip Gram

- Alternative to CBOW
- Input : target word one-hot encoding
  Output : surrounding words
- Scales better



- Assigns probability based on similarity of context window to target word

...lemon, a [tablespoon of apricot jam, a] pinch...

c1 c2 [target] c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

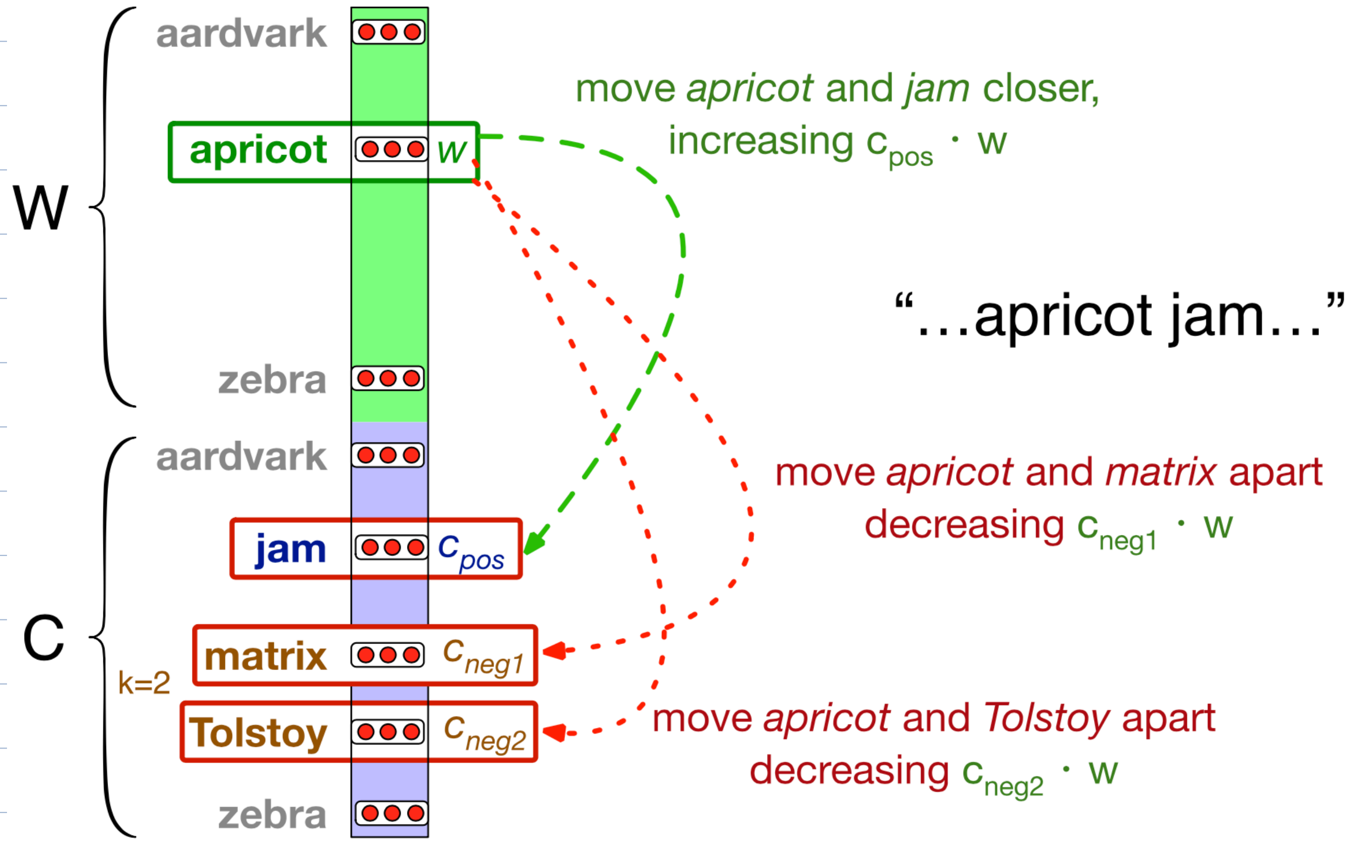
negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

- SGNS (skip-gram with negative sampling) uses more negative than positive samples
- Called noise words (chosen using weighted unigram frequency)
- Unweighted: words like the, and etc chosen most often
- weighted: set $\alpha = 0.75$

$$P_\alpha(w) = \frac{[\text{count}(w)]^\alpha}{\sum_{w'} [\text{count}(w')]^\alpha}$$

- Maximize similarity between (target, context) word pairs from pos data (and minimize for neg)



Loss function

output of pos

$$L_{CE} = -\left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w) \right]$$

- learns two sets of embeddings for each word (2 matrices)

1. Target embeddings matrix W
2. Context embeddings matrix C

Word2Vec Shortcomings

1. Slow to train, large no. of weights
2. Need lots of data

Improvements

1. Word pairs and phrases
2. Subsample frequent words
3. Selective updates (negative inputs)

4. GloVe

- Combines
  - global matrix factorization (LSA)
  - local context window (skip-gram)

Count based or Global Matrix Factorization Methods	Prediction based or Local Context window based Methods
Advantages: 1. Fast Training 2. Efficiently leverage statistical information.	1. Generates improved performance on tasks like POS tags or NER. 2. Can capture complex patterns beyond word similarity.
Disadv: 1. Primarily captures word similarity. 2. *Relatively perform poorly on the word analogy tasks,* *3. Disproportionate importance given to large counts.*	*Disadv:* *1. poorly utilize the statistics of the corpus since they train on separate local context windows instead of on global co-occurrence counts.* *2. Scales with corpus size* *3. Inefficient use of statistics of the dataset.*

- Learn word vectors ST dot product equals log of the probability of co-occurrence

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Co-occurrence probabilities for target words ice and steam with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like water and fashion cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

- water, fashion non-discriminative (cancel)

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij}) (u_i^T - \log P_{ij})^2$$

objective function

fast Text

- Character n-grams
- Slides