

word2vec

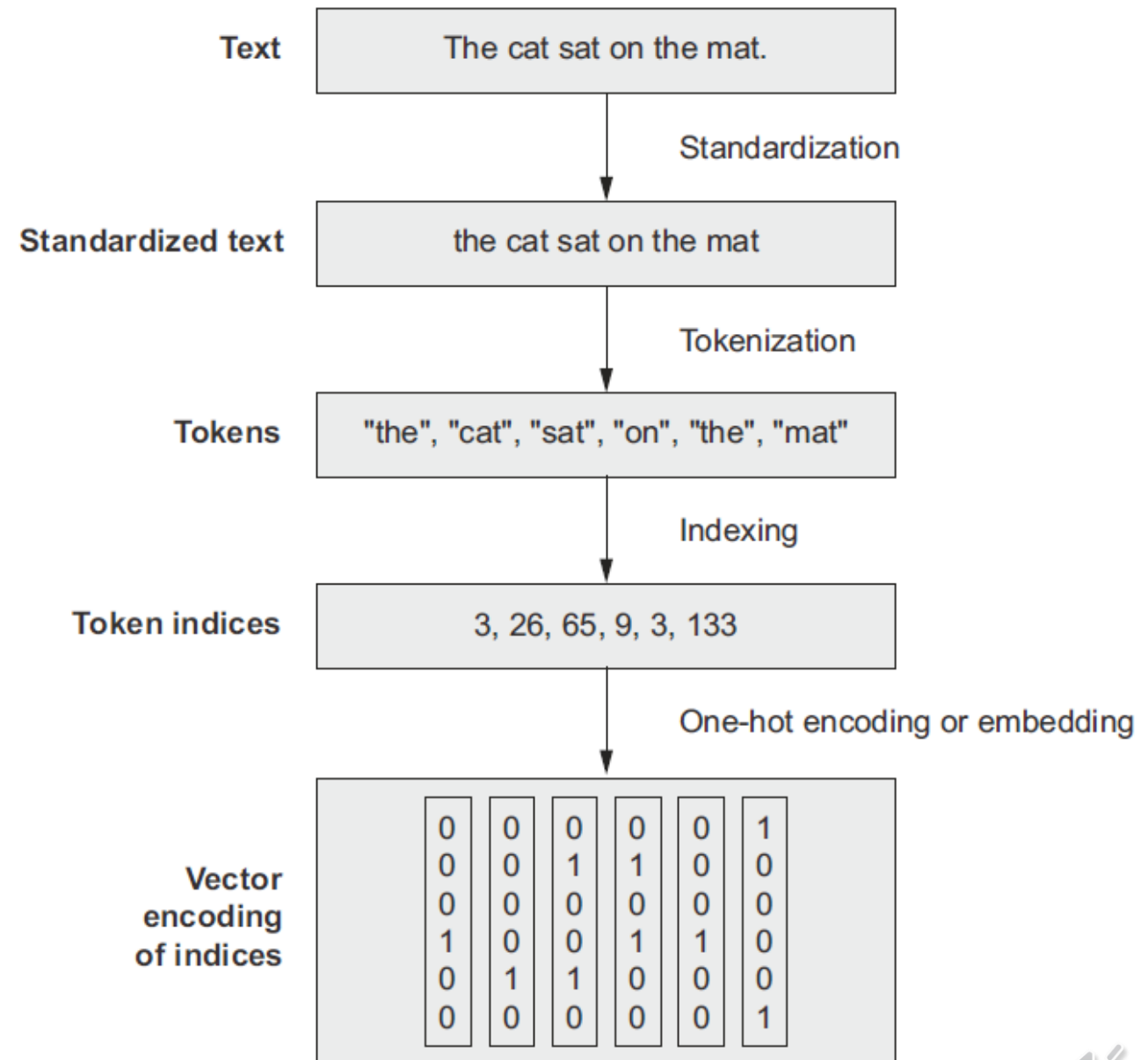
Kuan-Ting Lai

2022/4/25



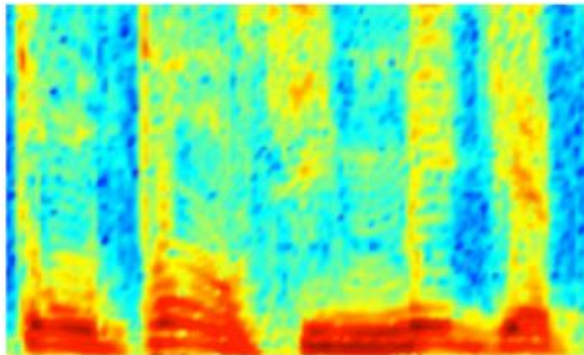
Natural Language Processing (NLP)

- Preprocessing
 - Convert words into vectors first



Why Word Embeddings?

AUDIO



Audio Spectrogram

DENSE

IMAGES

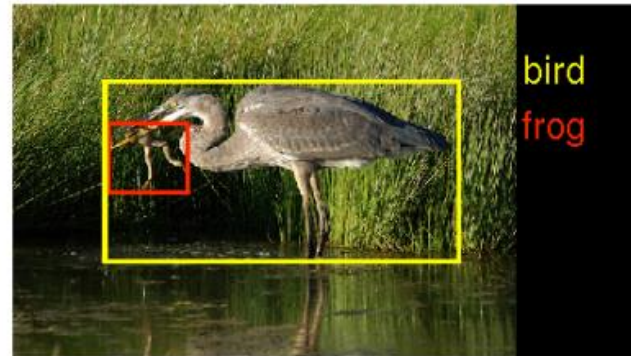
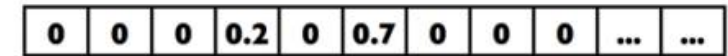


Image pixels

DENSE

TEXT



Word, context, or document vectors

SPARSE



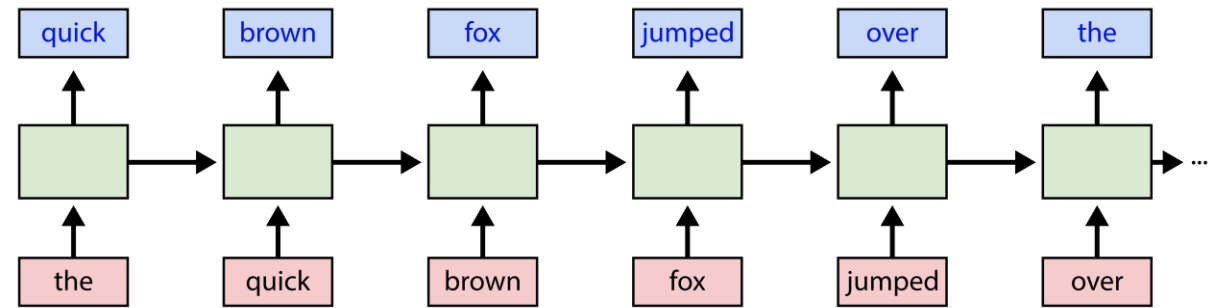
Word2vec (Word Embeddings)

- Embed one-hot encoded word vectors into dense vectors
- Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In *Advances in neural information processing systems*, pp. 3111-3119. 2013.



Bag-of-words vs. Sequence Model

The quick brown fox jumped over the lazy dog

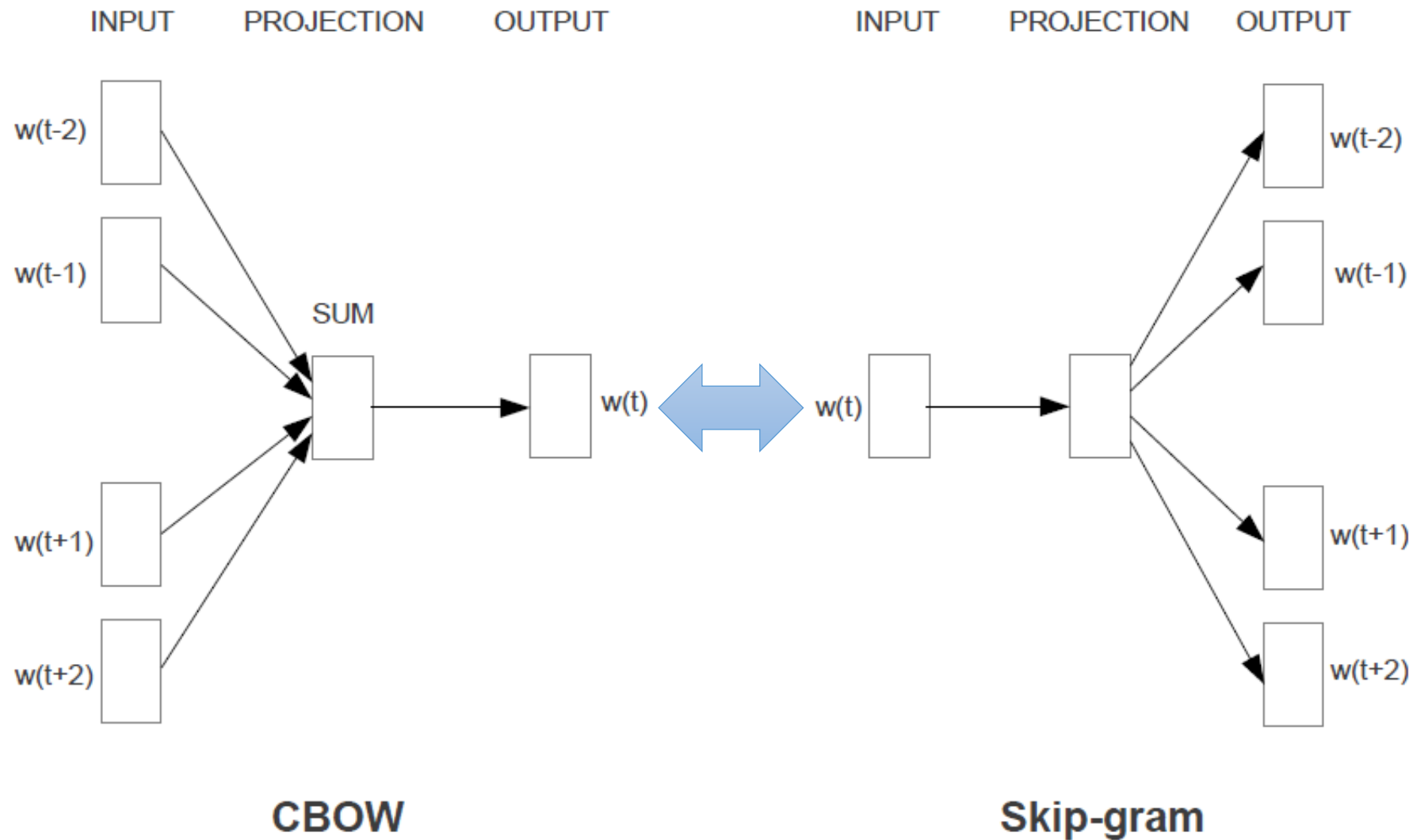


Bag-of-words

- **Count-based methods:**
 - how often some word co-occurs with its neighbor words
 - Latent Semantic Analysis
- **Predictive methods:**
 - Predict a word from its neighbors
 - Continuous Bag-of-Words model (CBOW) and Skip-Gram model

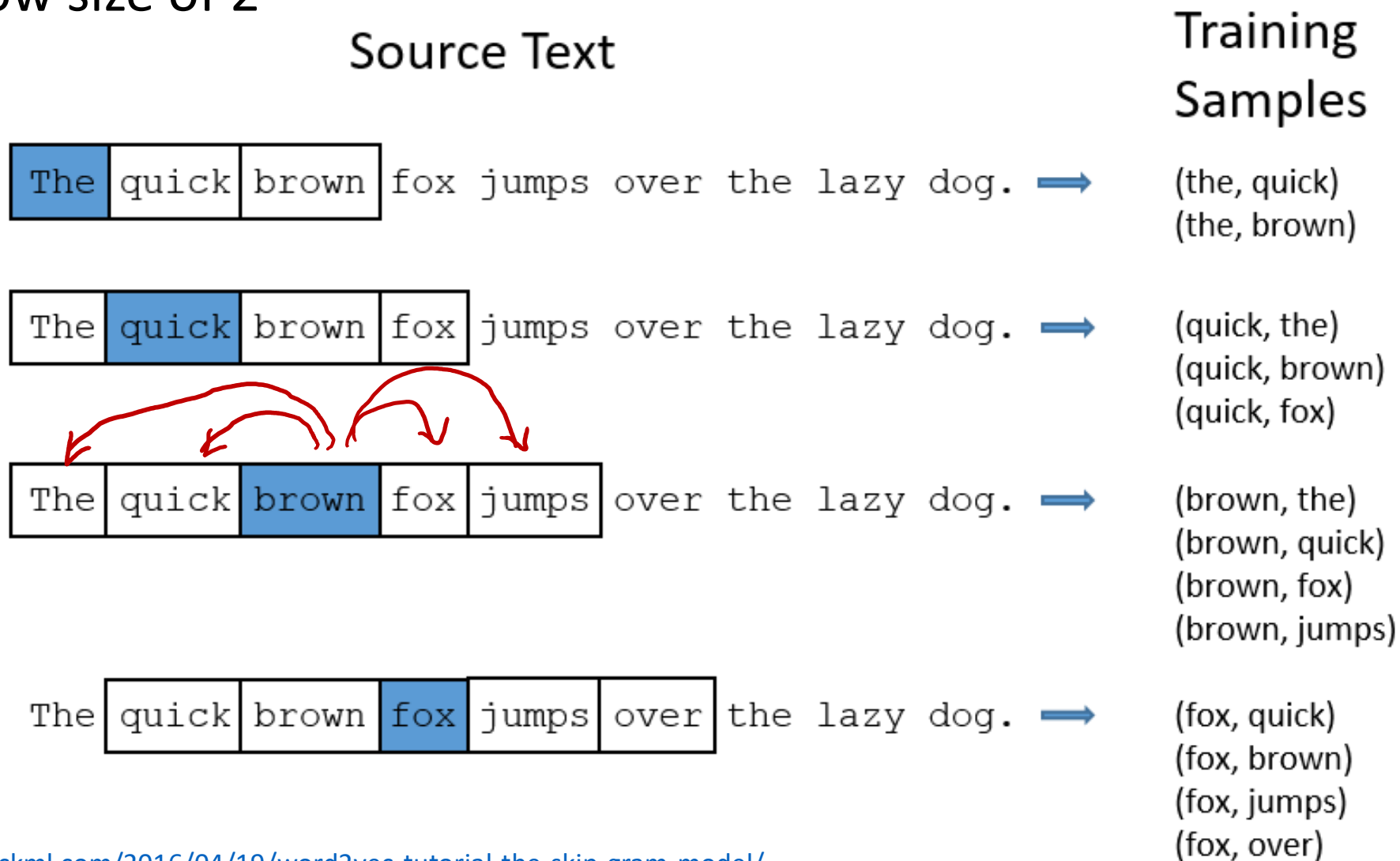


Continuous Bag-of-Words vs. Skip-Gram



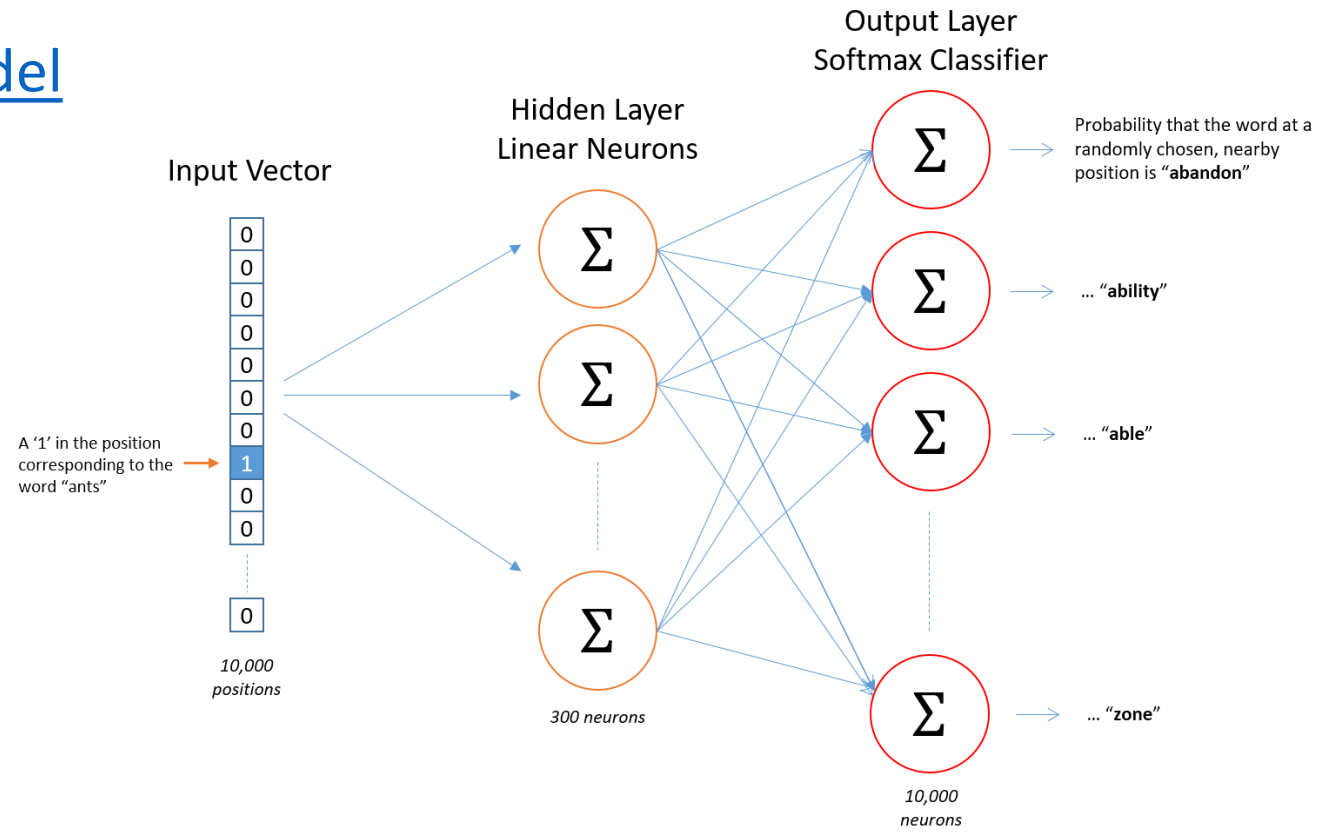
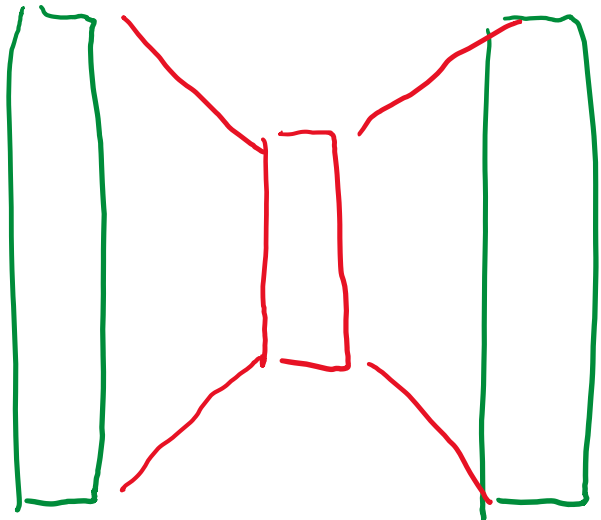
Skip-Gram Model

- Window size of 2

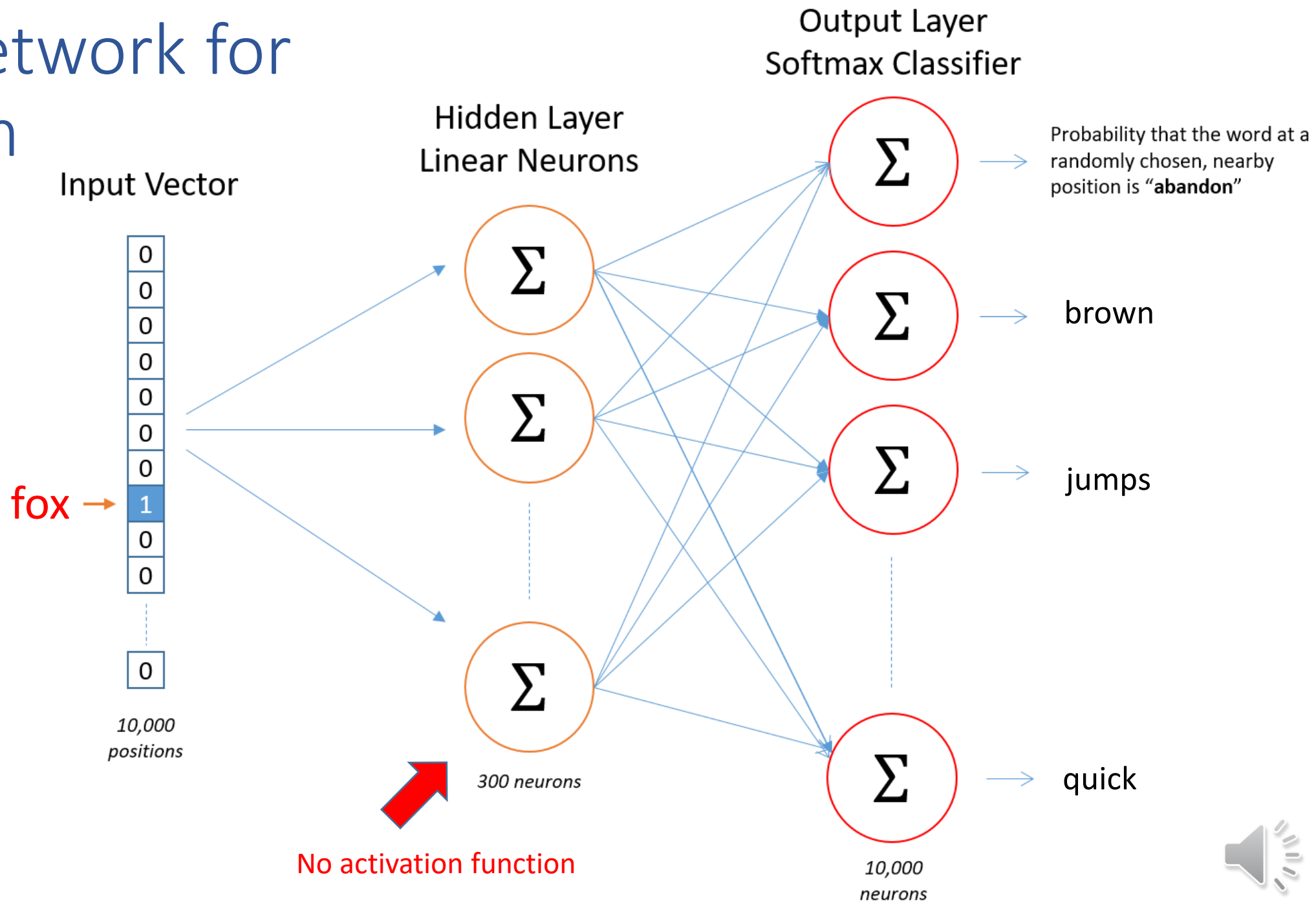


Word2Vec Tutorial

- [Word2Vec Tutorial - The Skip-Gram Model](#)
- [Word2Vec Tutorial - Negative Sampling](#)



Neural Network for Skip-Gram

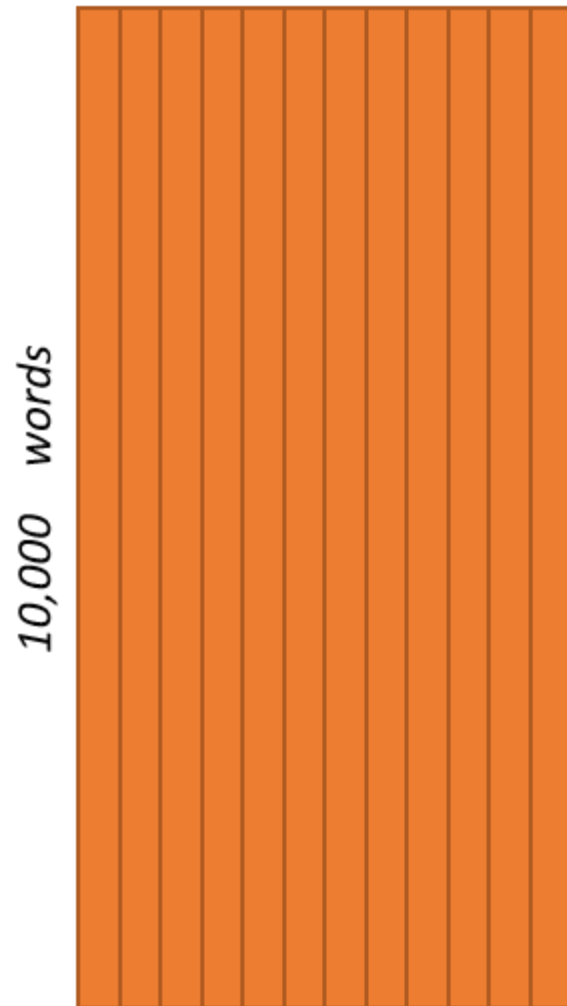


Hidden Layer
Weight Matrix

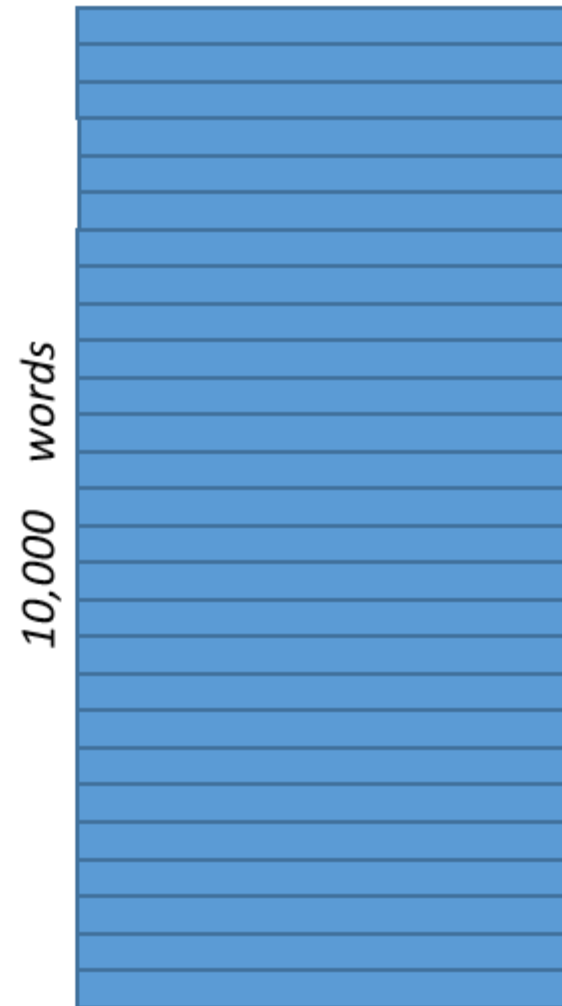


*Word Vector
Lookup Table!*

300 neurons



300 features



Hidden Layer as Look-up Table

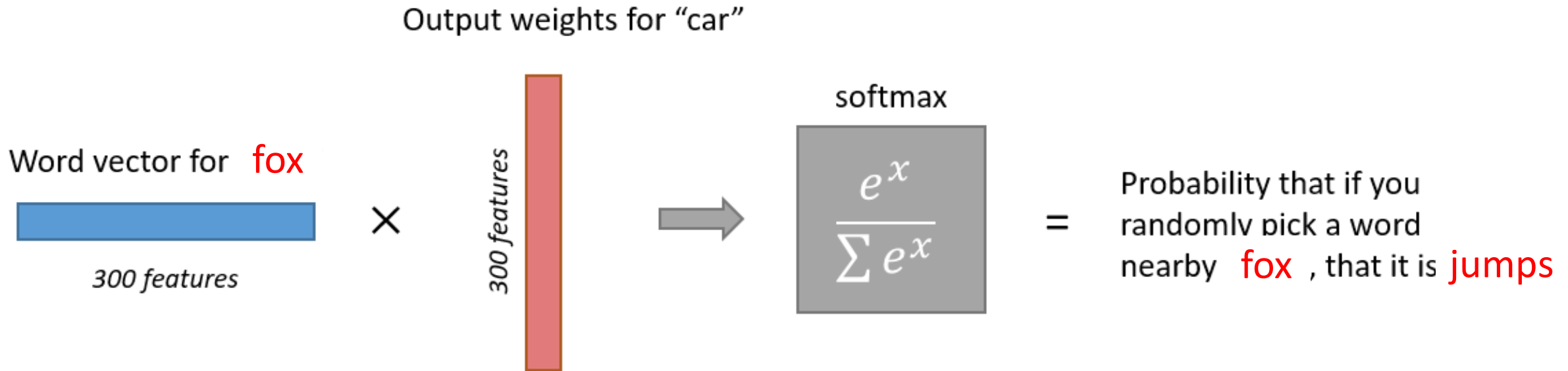
- One-hot vector *selects* the matrix row corresponding to the “1”

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$



The Output Layer (Softmax)

- Output probability of nearby words (e.g., “jumps” next to “fox”)
- Sum of all outputs is equal to 1



Softmax Function

- $P(w_t|h) = \text{softmax}(\text{score}(w_t, h)) = \frac{e^{\{\text{score}(w_t, h)\}}}{\sum_{\text{word } w' \text{ in vocab.}} e^{\{\text{score}(w', h)\}}}$
- $\text{score}(w_t, h)$ computes compatibility of word w_t with the context h (dot-product is used)
- Train the model by maximizing its log-likelihood:
$$-\log P(w_t|h) = \text{score}(w_t, h) - \log \left(\sum_{\text{word } w' \text{ in vocab.}} e^{\{\text{score}(w', h)\}} \right)$$



Sampling Important Words

- Remove non-informative word “the”

Source Text	Training Samples			
<table border="1"><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 border="1"><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 border="1"><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 border="1"><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		

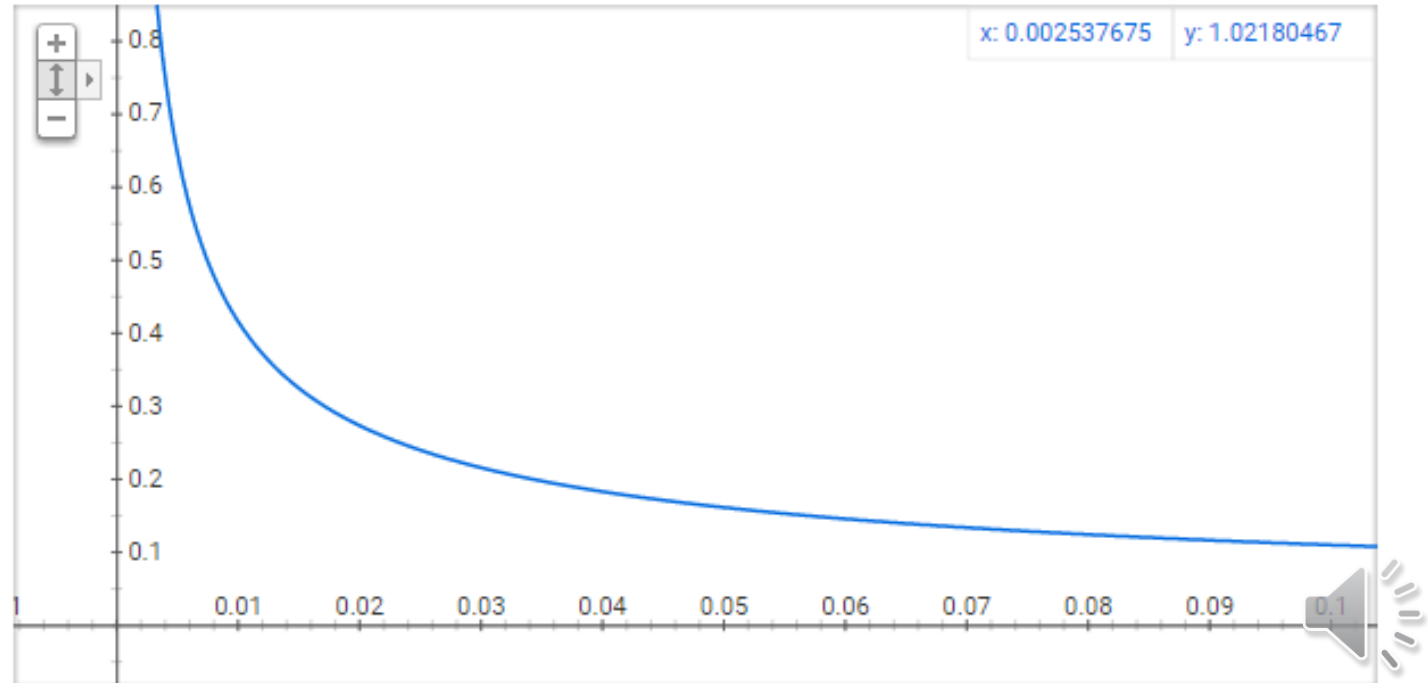


Probability of Keeping the Word

- $z(w_i)$ is the occurrence rate of word w_i
- $P(w_i)$ is the keeping probability

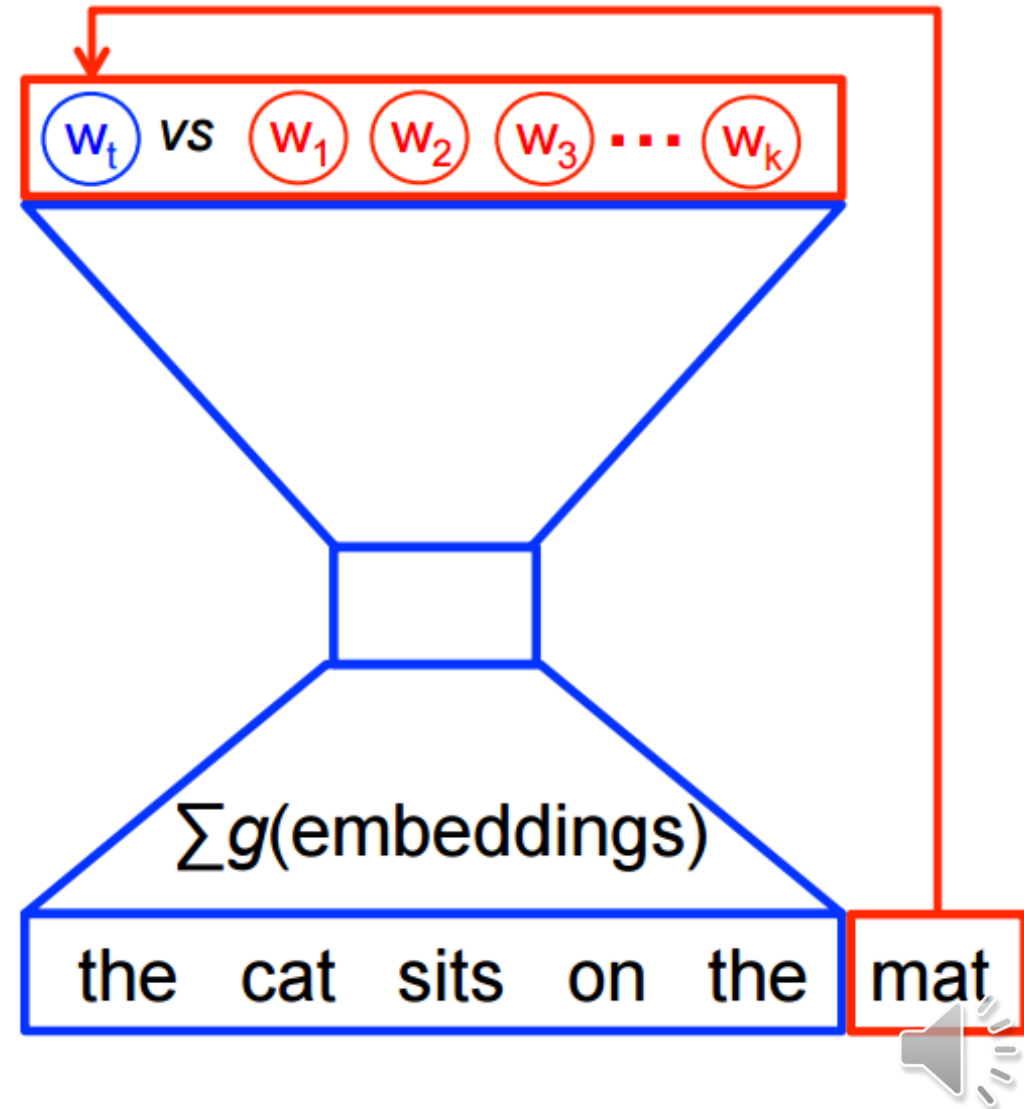
$$P(w_i) = \left(\sqrt{\frac{z(w_i)}{0.001}} + 1 \right) \cdot \frac{0.001}{z(w_i)}$$

Graph for $(\sqrt{x/0.001}+1)*0.001/x$



Negative Sampling

- Problem: too many parameters to learn at training
- Solution
 - Select only few other words as negative samples (output prob. = “0”)
 - Original paper selected 5 – 20 words for small datasets. 2 – 5 words work for large datasets

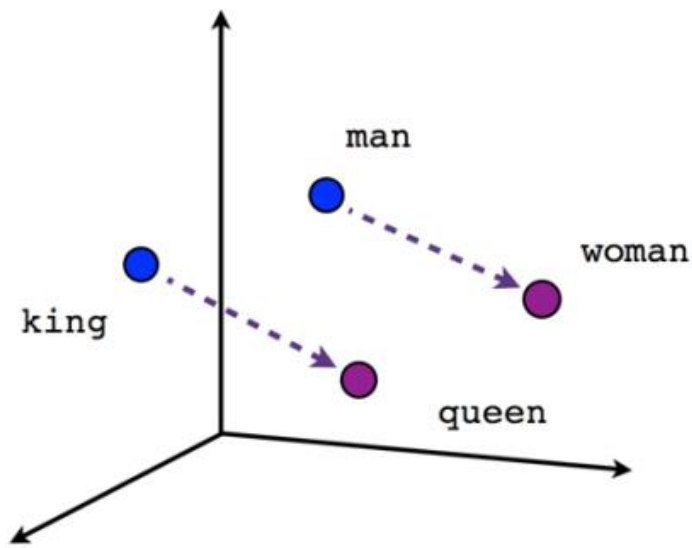


Negative Sampling

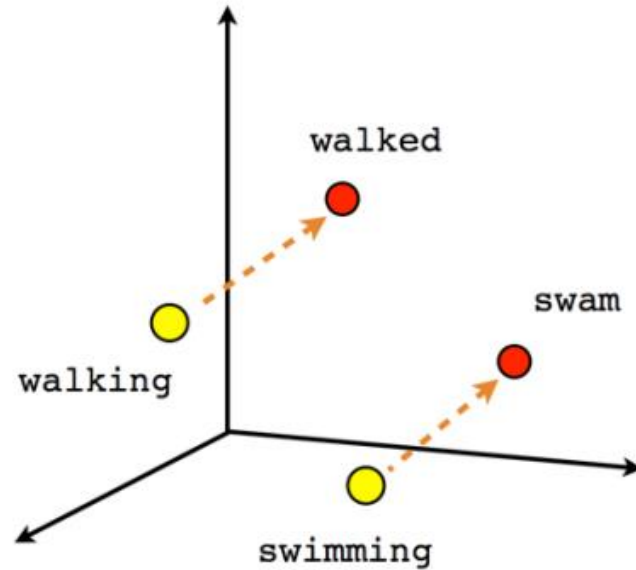
- $P(w_t|h) = \text{softmax}(\text{score}(w_t, h)) = \frac{e^{\{\text{score}(w_t, h)\}}}{\sum_{\text{word } w' \text{ in vocab.}} e^{\{\text{score}(w', h)\}}}$
- $\log P(w_t|h) = \text{score}(w_t, h) - \log \left(\sum_{\text{word } w' \text{ in vocab.}} e^{\{\text{score}(w', h)\}} \right)$
- Negative sampling reduces the number of words in the second terms



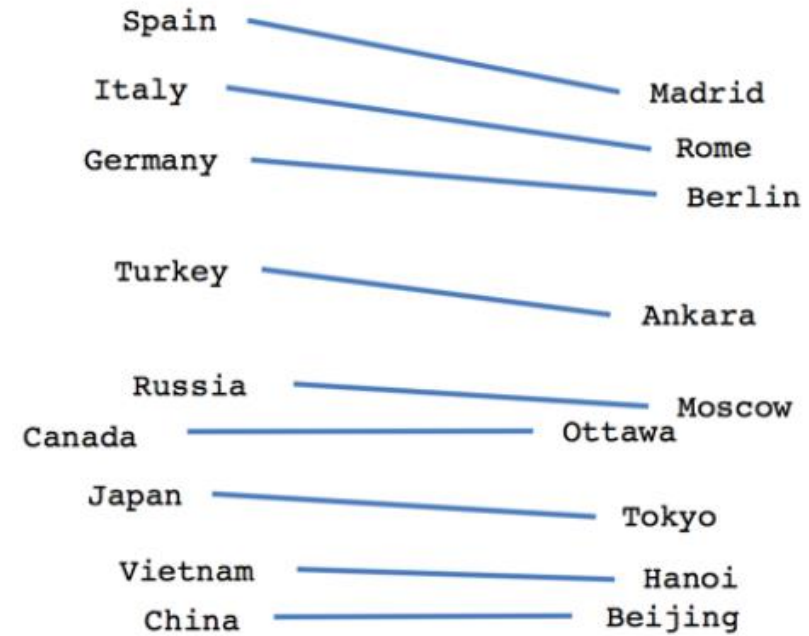
Evaluate Word2Vec



Male-Female



Verb tense



Country-Capital



Vector Addition & Subtraction

- $\text{vec}(\text{"Russia"}) + \text{vec}(\text{"river"}) \approx \text{vec}(\text{"Volga River"})$
- $\text{vec}(\text{"Germany"}) + \text{vec}(\text{"capital"}) \approx \text{vec}(\text{"Berlin"})$
- $\text{vec}(\text{"King"}) - \text{vec}(\text{"man"}) + \text{vec}(\text{"woman"}) \approx \text{vec}(\text{"Queen"})$



Embedding in Keras

- Input dimension: Dimension of the one-hot encoding, e.g. number of word indices
- Output dimension: Dimension of embedding vector

```
from keras.layers import Embedding  
  
embedding_layer = Embedding(1000, 64)
```



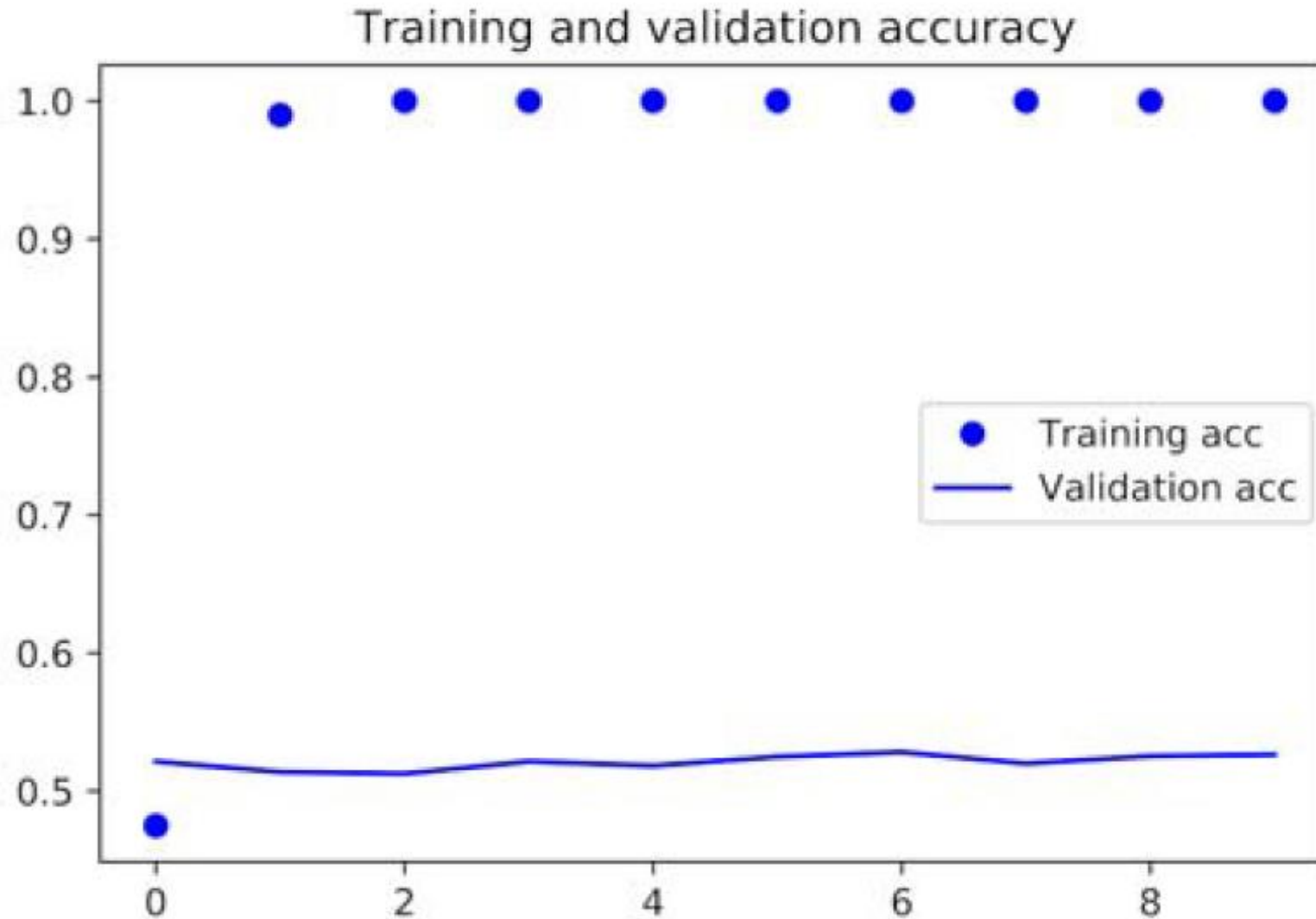
Using Embedding to Classify IMDB Data

```
from keras.datasets import imdb
from keras import preprocessing
from keras.models import Sequential
from keras.layers import Flatten, Dense, Embedding
max_features = 10000 # Number of words
maxlen = 20 # Select only 20 words in a text for demo
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
# Turn the lists of integers into a 2D integer tensor of shape (samples, maxlen)
x_train = preprocessing.sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = preprocessing.sequence.pad_sequences(x_test, maxlen=maxlen)

model = Sequential()
# Specify the max input length to the Embedding layer so we can later flatten the embedded
# inputs. After the Embedding layer, the activations have shape (samples, maxlen, 8).
model.add(Embedding(10000, 8, input_length=maxlen))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```



Training Embedding Model on IMDB



GloVe: Global Vectors for Word Representation

- Developed by Stanford in 2014
- Based on Matrix Factorization of Word Co-occurrence
- <https://nlp.stanford.edu/projects/glove/>
- Assumption
 - Ratios of word-word co-occurrence probabilities encode some form of meaning

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



Using Pretrained Word Embedding Vectors (2-1)

```
# Preprocessing the embeddings
glove_dir = './glove/'
embeddings_index = {}
f = open(os.path.join(glove_dir, 'glove.6B.100d.txt'))
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Found %s word vectors.' % len(embeddings_index))# 400000 word vectors.

# Create a word embedding tensor
embedding_dim = 100
embedding_matrix = np.zeros((max_words, embedding_dim))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if i < max_words:
        if embedding_vector is not None:
            # Words not found in embedding index will be all-zeros.
            embedding_matrix[i] = embedding_vector
```



Using Pretrained Word Embedding Vectors (2-2)

```
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense

model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=maxlen))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()

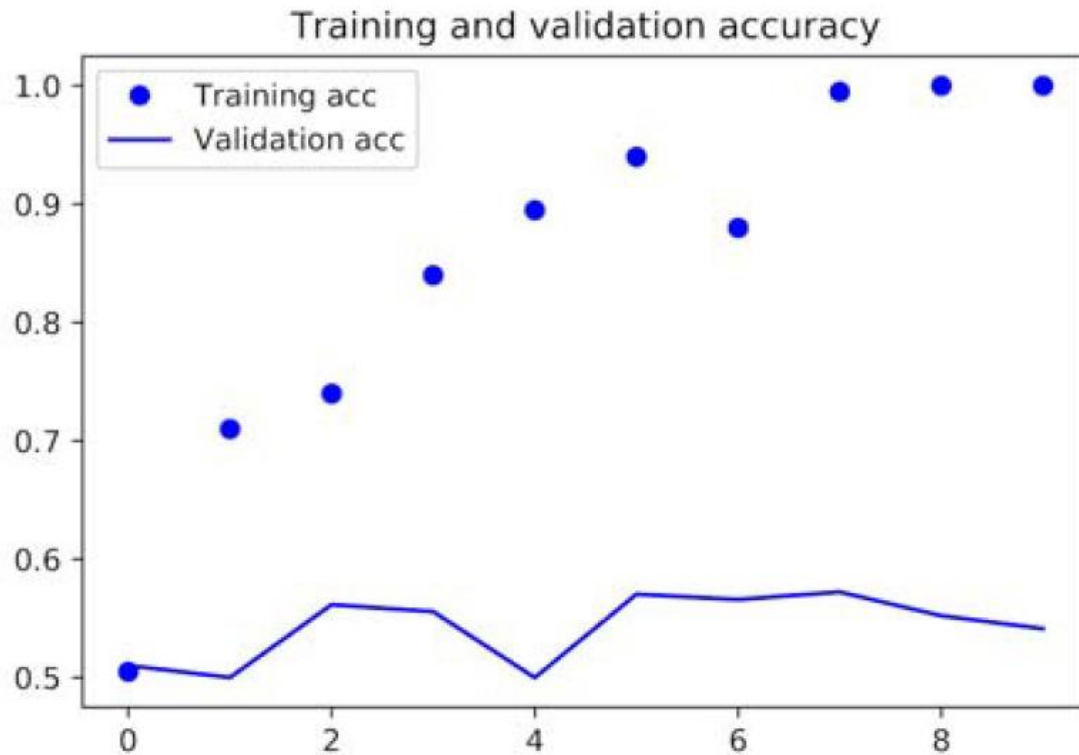
# Load the GloVe embeddings in the model
model.layers[0].set_weights([embedding_matrix])
model.layers[0].trainable = False

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10, batch_size=32, validation_data=(x_val, y_val))
model.save_weights('pre_trained_glove_model.h5')
```

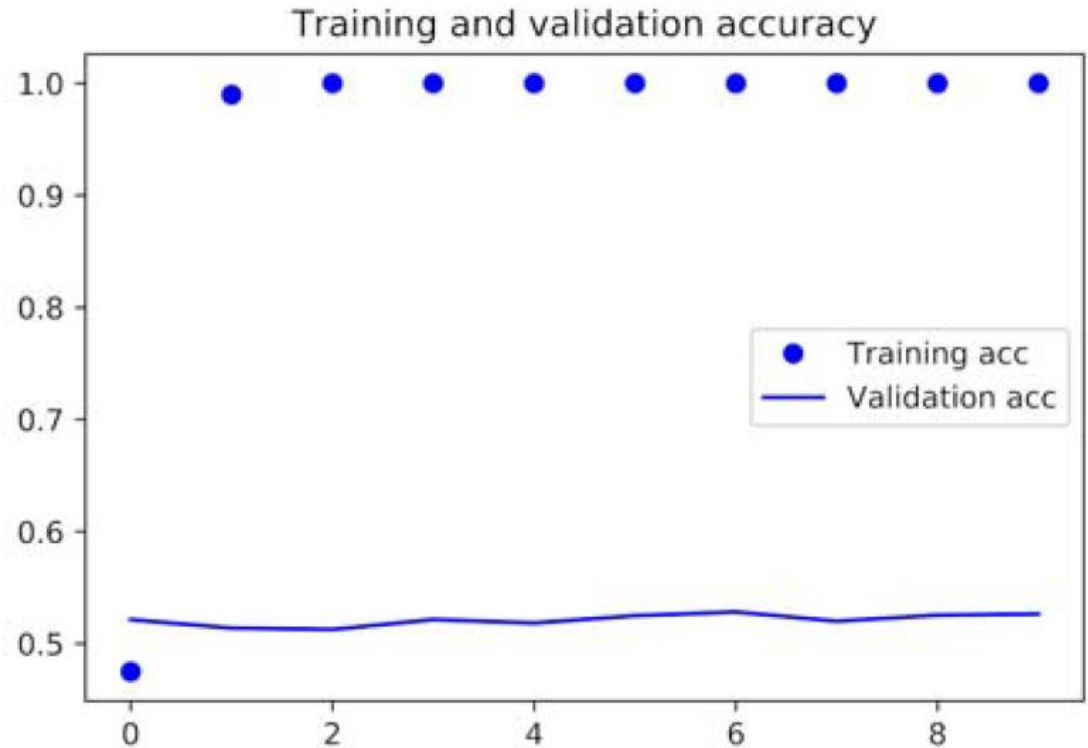


Classifying IMDB Reviews

With pretrained Glove vectors



Without pretrained embedding model



Embedding Project (projector.tensorflow.org/)

Embedding Projector



DATA

Points: 10000 | Dimension: 200

5 tensors found
Word2Vec 10K

Label by word
Color by No color map

Edit by word
Tag selection as

Load Publish Download Label

Sphereize data

Checkpoint: Demo datasets

Metadata: oss_data/word2vec_10000_200d_labels.tsv

UMAP T-SNE **PCA** CUSTOM

x Component #1
y Component #2

z Component #3

PCA is approximate.

Total variance described: 8.5%.



Show All Data

Isolate selection

Clear selection

Search

by word

BOOKMARKS (0)



Embedding Projector

DATA

Points: 10000 | Dimension: 200

5 tensors found
Word2Vec 10K

Label by: **word** | Color by: **No color map**

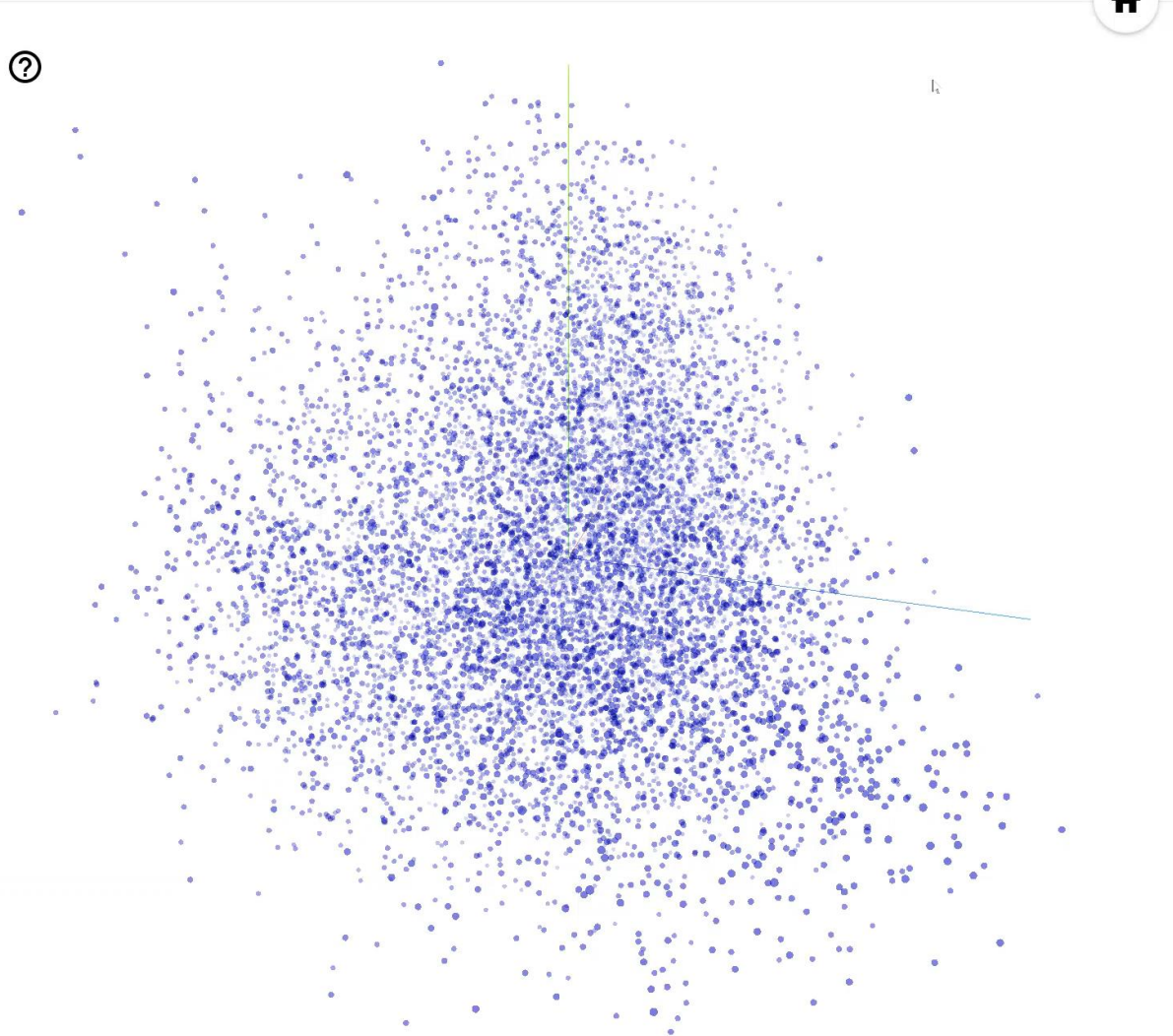
Edit by: **word** | Tag selection as

Load **Publish** **Download** Label

Sphereize data

Checkpoint: Demo datasets

Metadata: oss_data/word2vec_10000_200d_labels.tsv



UMAP T-SNE **PCA** CUSTOM

X: **Component #1** | Y: **Component #2**

Z: **Component #3**

PCA is approximate.

Total variance described: 8.5%.

Show All Data Isolate selection Clear selection

Search by **word**



Neighbors of "Learning"

Embedding Projector



DATA

Points: 10000 | Dimension: 200 | Selected 264 points

5 tensors found
Word2Vec 10K

Label by word
Color by No color map

Edit by word
Tag selection as

Load Publish Download Label

Sphereize data

Checkpoint: Demo datasets

Metadata: oss_data/word2vec_10000_200d_labels.tsv

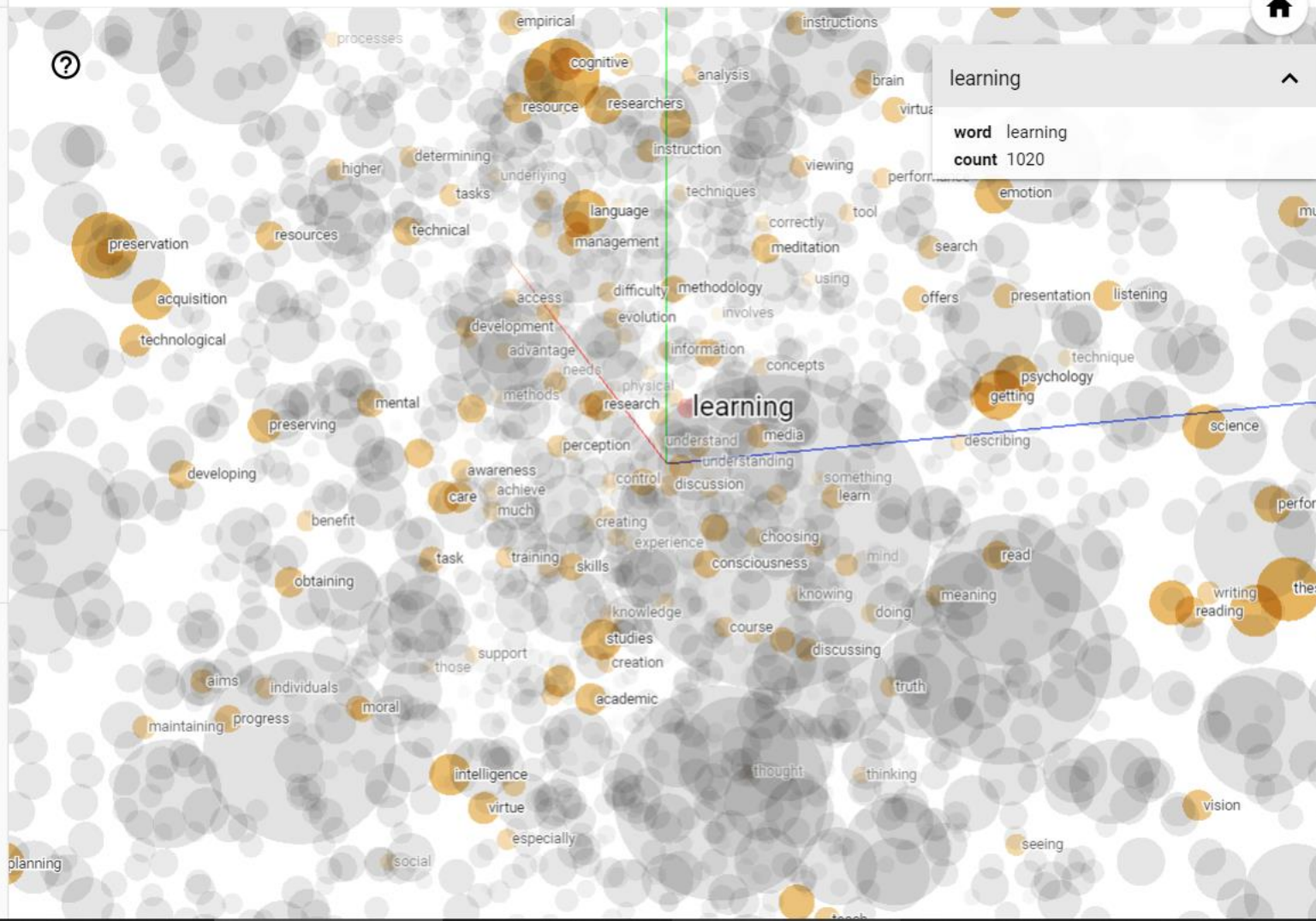
UMAP T-SNE **PCA** CUSTOM

X Component #1
Y Component #2

Z Component #3

PCA is approximate.

Total variance described: 8.5%.



Show All Data Isolate 264 points Clear selection

Search learning by word

neighbors 263

distance COSINE EUCLIDEAN

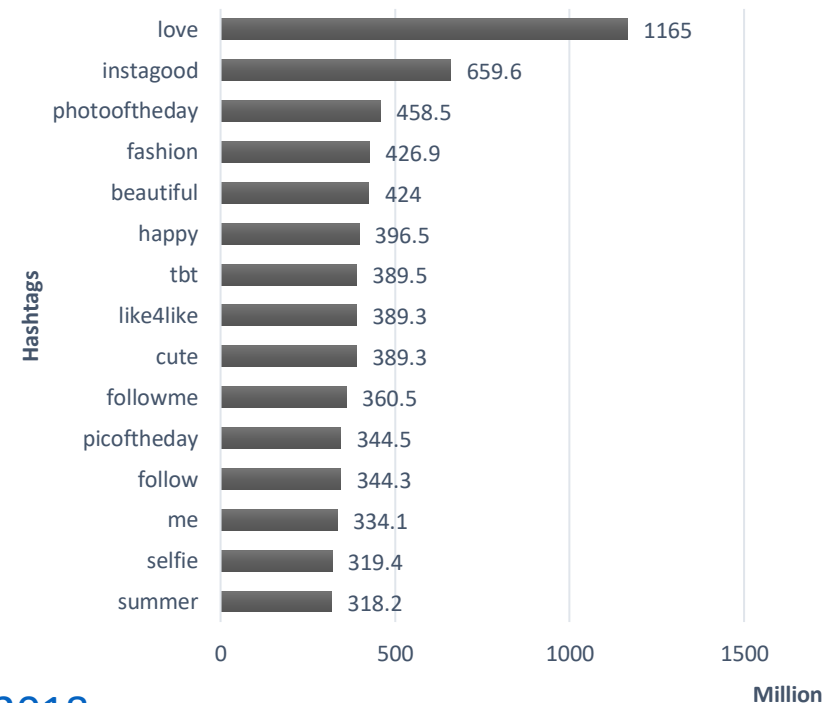
Nearest points in the original space:

teaching	0.451
knowledge	0.556
cognitive	0.566
educational	0.568
education	0.574
thinking	0.604
learn	0.613
training	0.615
understanding	0.626
communication	0.630
learned	0.632
instruction	0.633
mathematics	0.636
study	0.637

BOOKMARKS (0)

Image Hashtag Recommendation

- Hashtag => a word or phrase preceded by the symbol # that categorizes the accompanying text
- Created by Twitter, now supported by all social networks
- Instagram hashtag statistics (2017):



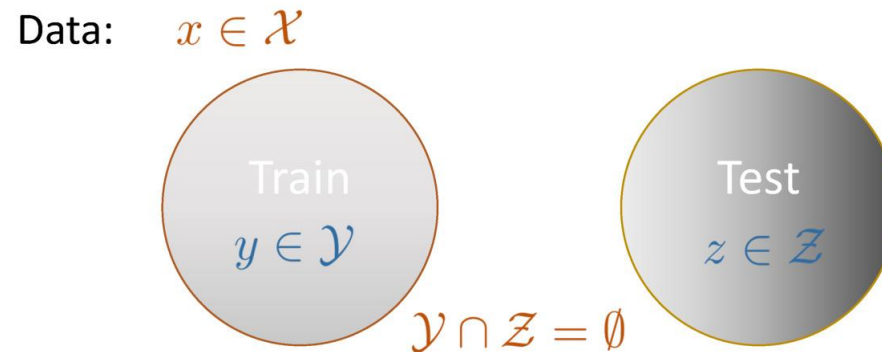
Difficulties of Predicting Image Hashtag

- Abstraction: #love, #cute,...
- Abbreviation: #ootd, #ootn,...
- Emotion: #happy,...
- Obscurity: #motivation, #lol,...
- New-creation: #EvaChenPose,...
- No-relevance: #tbt, #nofilter, #vscocam
- Location: #NYC, #London



Zero-Shot Learning

- Identify object that you've never seen before
- More formal definition:
 - Classify test classes Z with zero labeled data (Zero-shot!)

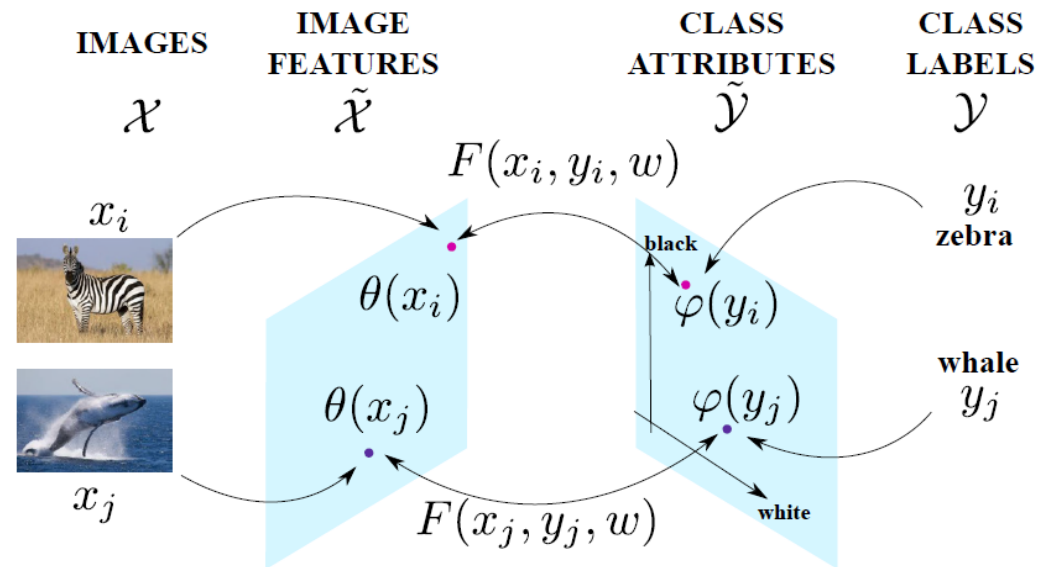


Objective: $f : \mathcal{X} \rightarrow \mathcal{Z}$



Zero-Shot Formulation

- Describe objects by words
 - Use attributes (semantic features)



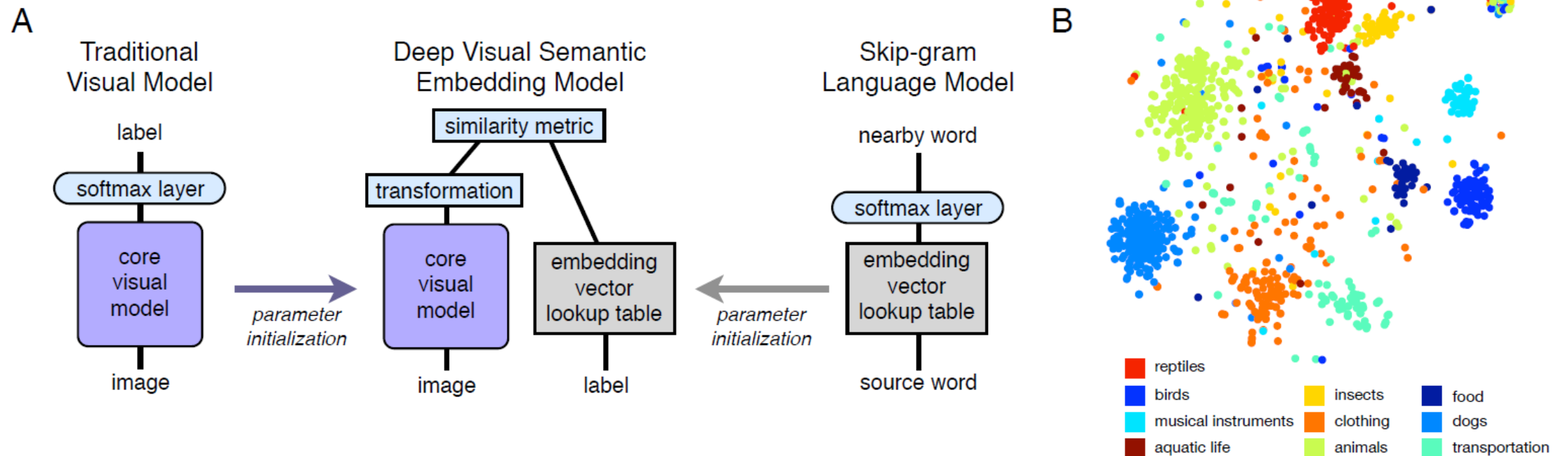
$$F(x, y; W) = \theta(x)^T W \phi(y)$$



DeViSE – Deep Visual Semantic Embedding

- Google, NIPS, 2013

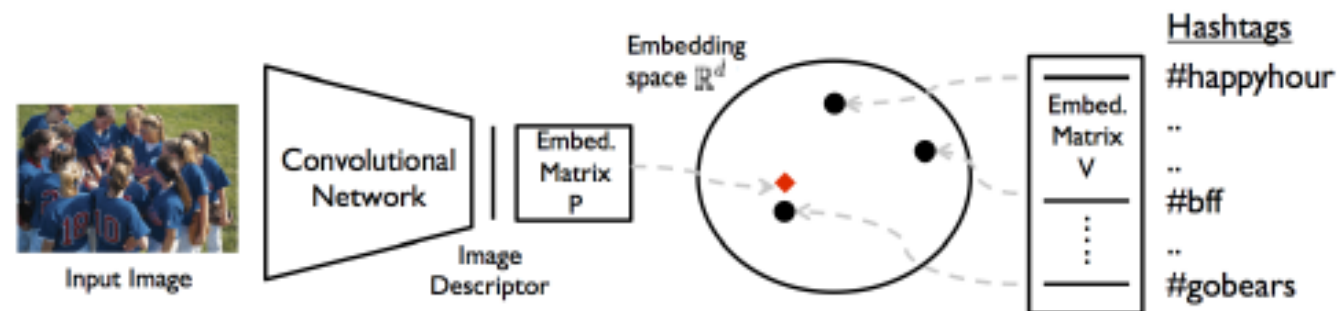
Model	200 labels	1000 labels
DeViSE	31.8%	9.0%
Mensink et al. 2012 [12]	35.7%	1.9%
Rohrbach et al. 2011 [17]	34.8%	-



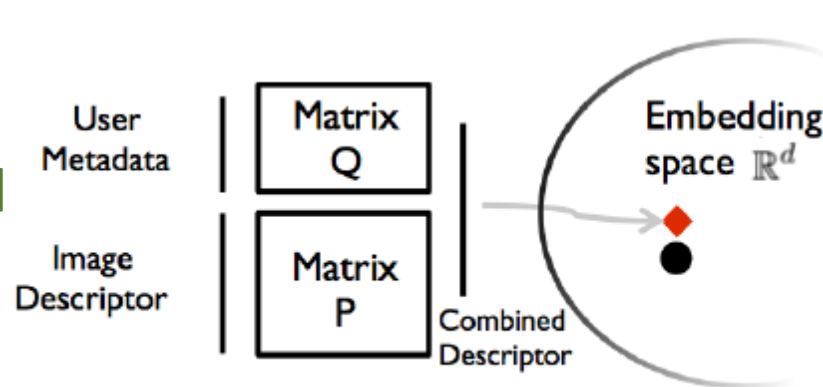
User Conditional Hashtag Prediction for Images

- E. Denton, J. Weston, M. Paluri, L. Bourdev, and R. Fergus, "User Conditional Hashtag Prediction for Images," *ACM SIGKDD*, 2015 (Facebook)
- Hashtag Embedding: $f(x, y) = \Phi_I(x)^T \Phi_H(y)$
- Proposed 3 models:

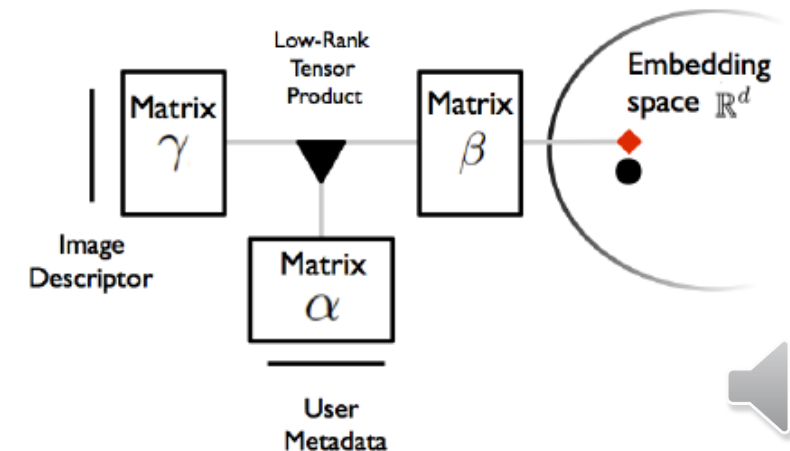
1. Bilinear Embedding Model



2. User-biased model



3. User-multiplicative model



User Profile and Locations



Age	Females	Males
13-17	#mcm	#like
	#bestfriend	#lmp
	#love	#throwback
	#lovehim	#squad
	#mce	#wce
	#latepost	#throwback-
	#bestfriends	#thursday
	#boyfriend	#family
	#loveher	#workflow
	#loveyou	#selfie
43-47	#100happydays	#photoshop-
	#mcm	#express
	#love	#wcv
	#sisters	#goodtimes
	#cousins	#prouddad
	#lovehim	#throwback-
	#latergram	#thursday
	#loveher	#selfie
	#bff	#salute
	#youcampperfect	#blessed
	#zijasummit14	
	#familyfirst	

Gender	15-17 years old	43-47 years old
Female	#wcv	#100happydays
	#mcm	#blessed
	#bestfriend	#goodtimes
	#tb	#family
	#ss	#love
	#bestfriends	#photogrid
	#throwback	#latergram
	#latepost	#cousins
	#like	#sundayfunday
	#selfiesunday	#friends
Male	#wcv	#goodtimes
	#like	#blessed
	#throwback	#love
	#squad	#family
	#tb	#photoshop-
	#lmp	#express
	#mcm	#photogrid
	#ss	#sundayfunday
	#wce	#friends
	#selfiesunday	#zijasummit14
	#prouddad	

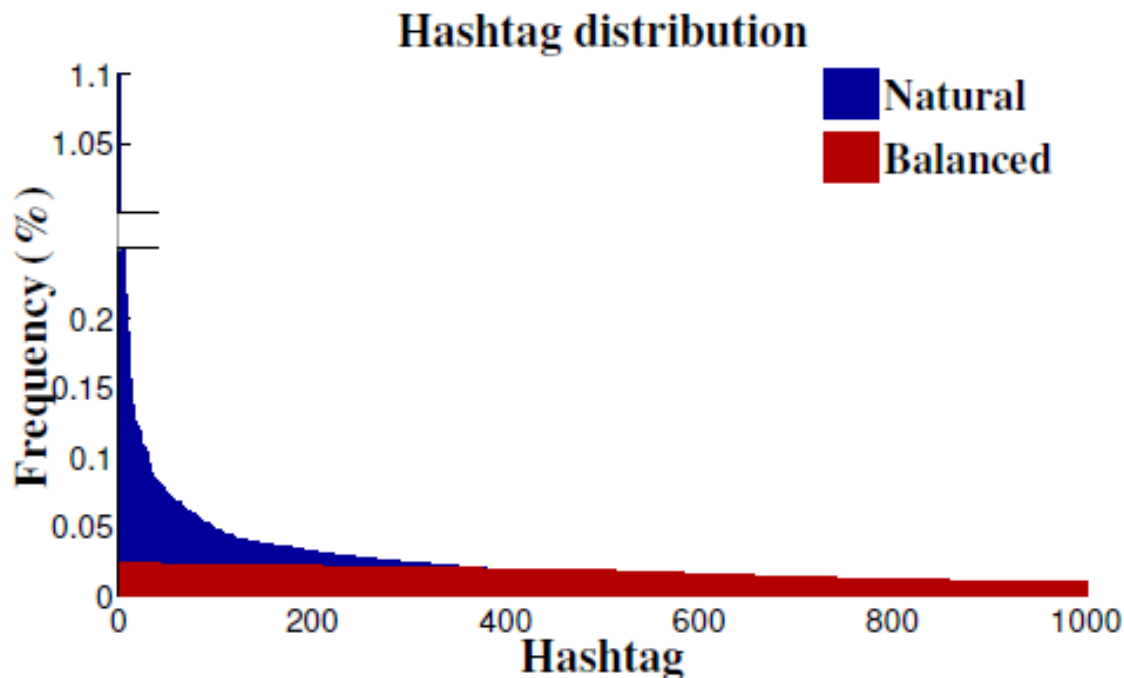
Sydney	Toronto
#melbourne	#toronto
#sydney	#tbt
#australia	#canada
#spring	#vancouver
#beach	#fall
#grandfinal	#throwback
#sunshine	#blessed
#sun	#ilovethiscity
#nz	#vancity
#newzealand	#vscocam
#bali	#tb
#happy	#cntower
#nofilter	#goodmorning
#wellington	#hoco
#springbreak	#montreal
#bondi	#wcv
#afl	#tdot
#thailand	#lateupload
#stkilda	#downtown
#city	#beautiful

Meta data	Possible values
Age	13 – 114
Gender	Male, Female, Unknown
Home City	GPS coordinates
Country	United States, Canada, Great Britain, Australia, New Zealand



Facebook's Experiments

- 20 million images
- 4.6 million hashtags, average 2.7 tags per image
- Result



Method	d	K	P@1	R@10	A@10
Freq. baseline	-	-	3.04%	5.63%	9.45%
Bilinear	64	-	7.37%	11.71%	18.69%
Bilinear	128	-	7.37%	11.69%	18.44%
Bilinear	256	-	6.75%	10.84%	17.25%
Bilinear	512	-	6.50%	10.83%	17.17%
User-biased	64	-	9.02%	13.63%	21.88%
User-biased	128	-	9.00%	13.67%	21.83%
User-biased	256	-	8.48%	13.03%	20.96%
User-biased	512	-	7.98%	12.51%	20.05%
3-way mult.	64	50	8.95%	13.66%	21.82%
3-way mult.	64	100	9.03%	13.81%	22.04%
3-way mult.	64	200	8.96%	13.81%	22.05%
3-way mult.	64	300	9.00%	13.74%	21.96%
3-way mult.	64	400	8.96%	13.65%	21.82%



Real World Applications

mccormickml.com/2018/06/15/applying-word2vec-to-recommenders-and-advertising/

VIEWING HISTORY



Context

Input

Context



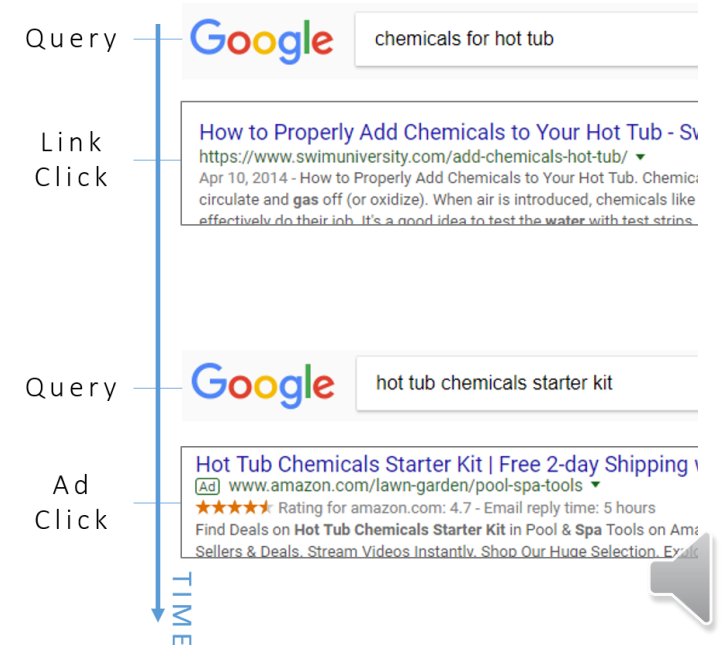
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	Amazon.com	Inbox	Amazon Orders	Your Amazon.com	View order ↗	Jun 12
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	Target	Inbox		Thanks for shopping Target! Here's your order #: 90		Jun 10
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	Amazon.com (3)	Inbox	Amazon Orders	Your Amazon.com	View order ↗	Jun 10
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	Amazon.com	Inbox	Amazon Orders	Your Amazon.com order of "Da	Track package ↗	Jun 6
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	Amazon.com	Inbox	Amazon Orders	Your Amazon.com	View order ↗	Jun 4
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	orderstatus@costco.com	Inbox		Your Costco.com Order Number :	View order ↗	Jun 4
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	Amazon.com	Inbox	Amazon Orders	Your Amazon.com order of "Fr	Track package ↗	Jun 3
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	orderstatus@costco.com	Inbox		Your Costco.com Order Number :	View order ↗	Jun 2
<input type="checkbox"/>	<input type="star"/>	<input type="trash"/>	Amazon.com	Inbox	Amazon Orders	Your Amazon.com	View order ↗	Jun 2

Context

Input

Context

User Activity Log



References

1. Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.
2. Goldberg, Yoav, and Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method." *arXiv preprint arXiv:1402.3722* (2014).
3. <https://www.tensorflow.org/tutorials/representation/word2vec>
4. <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>
5. <https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/>

