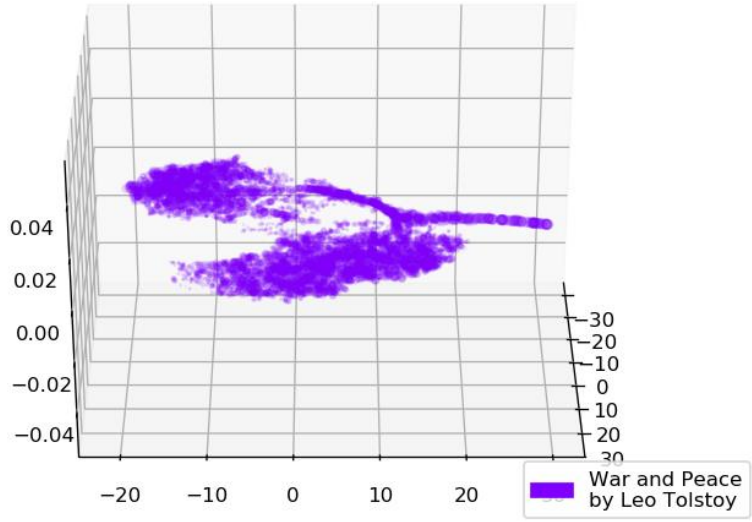


word2vec

Kuan-Ting Lai

2020/4/7

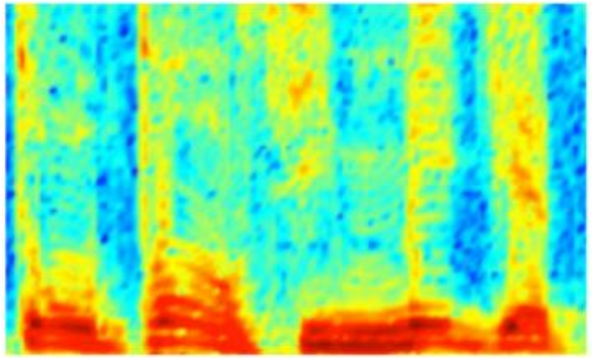
Word2vec (Word Embeddings)

- 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.

Why Word Embeddings?

AUDIO



Audio Spectrogram

DENSE

IMAGES

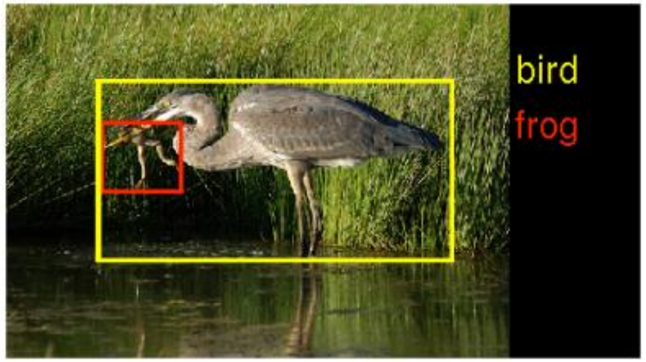


Image pixels

DENSE

TEXT

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

Word, context, or document vectors

SPARSE

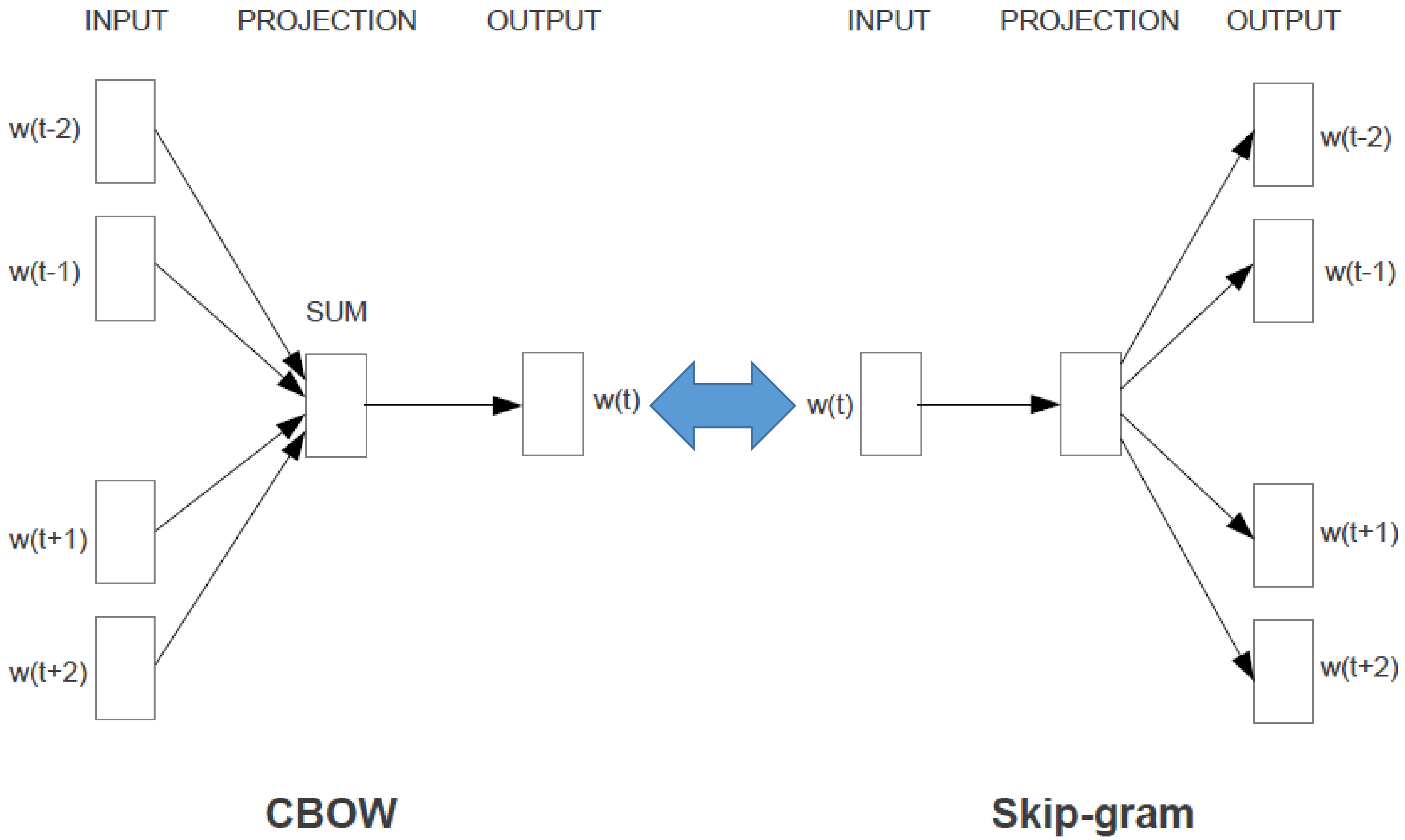
https://www.tensorflow.org/tutorials/representation/word2vec

Vector Space Models for Word Embedding

- 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

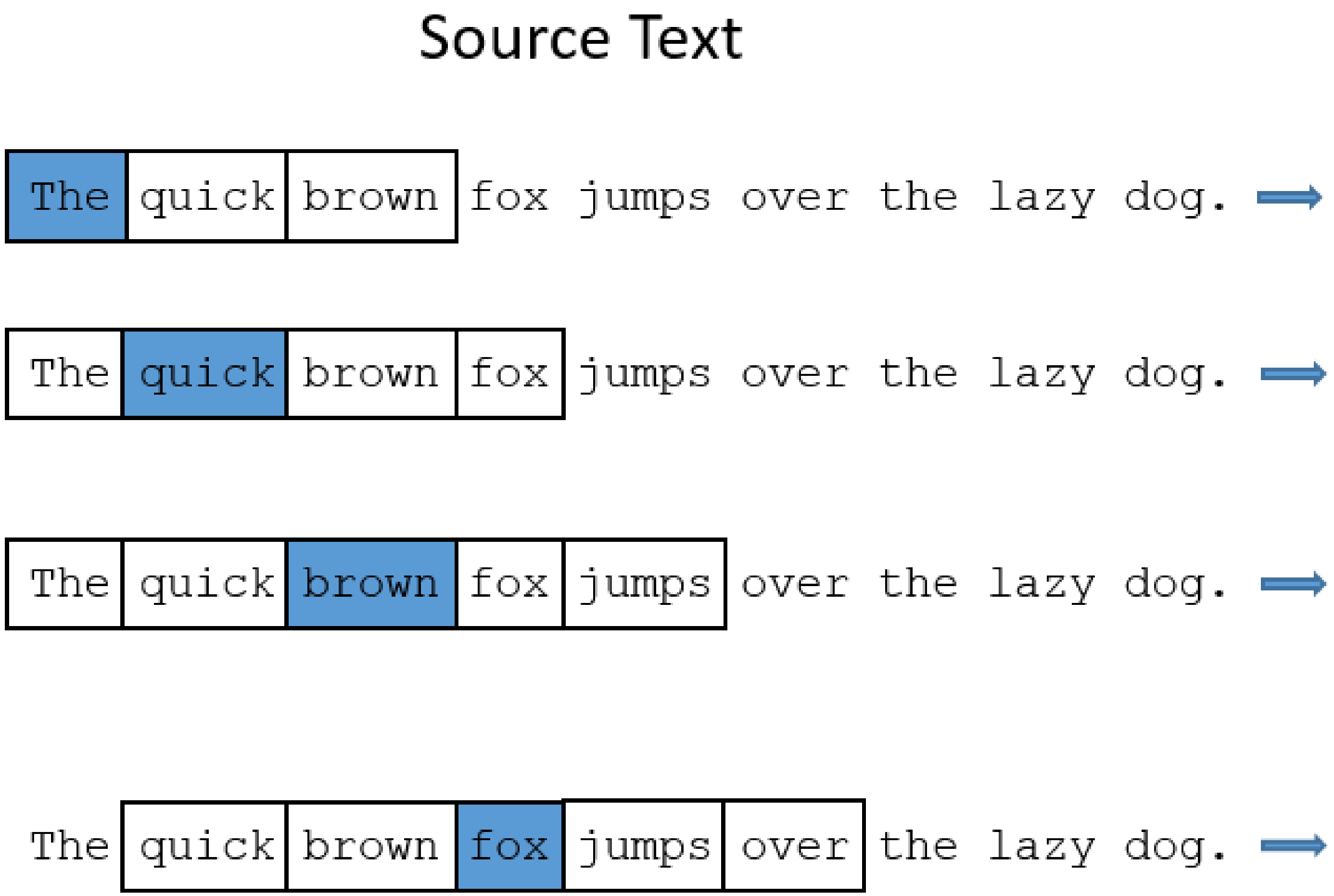


N-Gram Model

- Use a sequence of N words to predict next word
- Example N=3
  - (The, quick, brown) -> fox

Skip-Gram Model

- Window size of 2



Training Samples

(the, quick)
(the, brown)

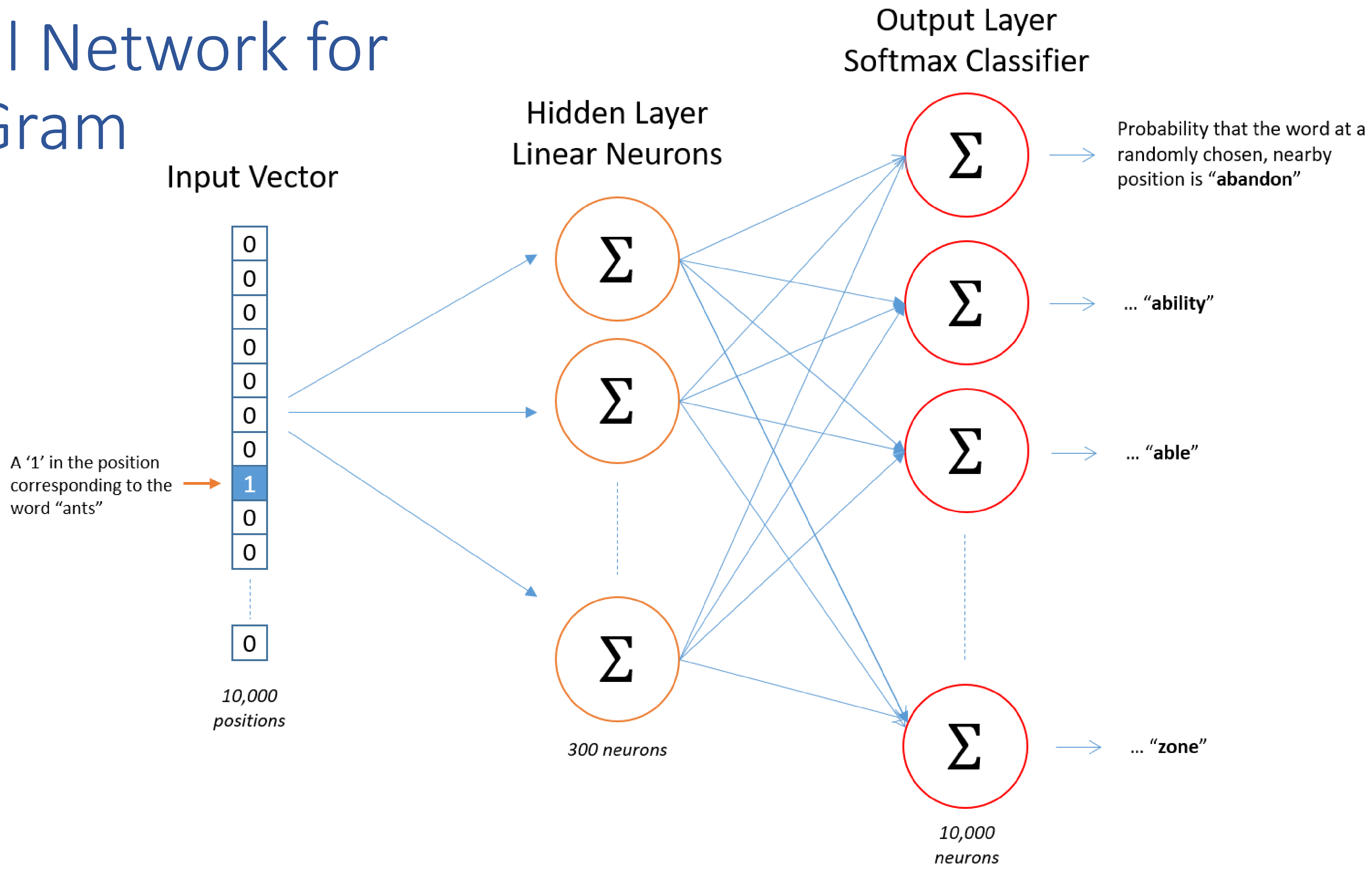
(quick, the)
(quick, brown)
(quick, fox)

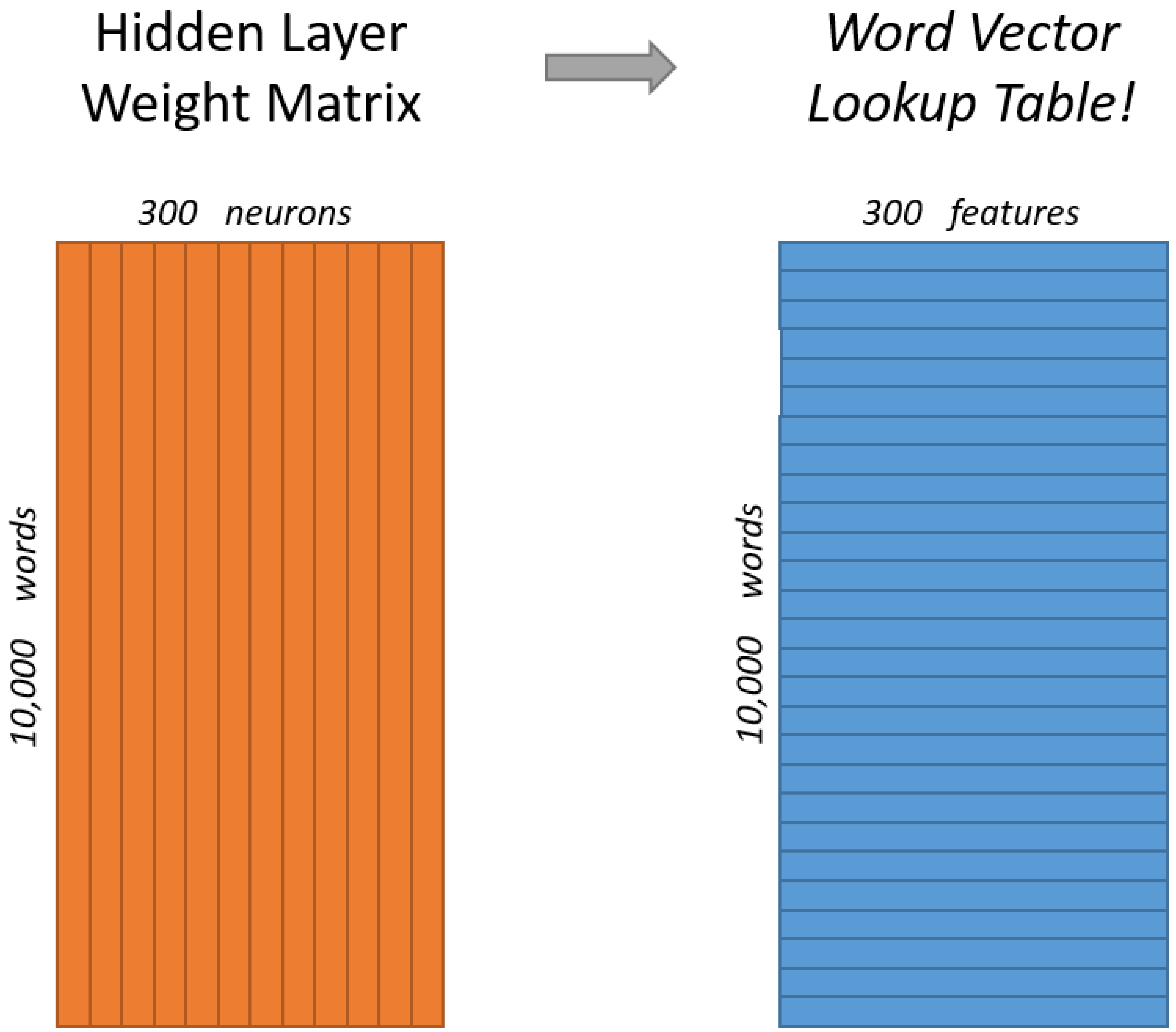
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)

(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Neural Network for Skip-Gram




Hidden Layer
Weight Matrix
Word Vector
Lookup Table!
300 neurons
10,000 words
300 features
10,000 words

Hidden Layer as Look-up Table

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

Softmax Function

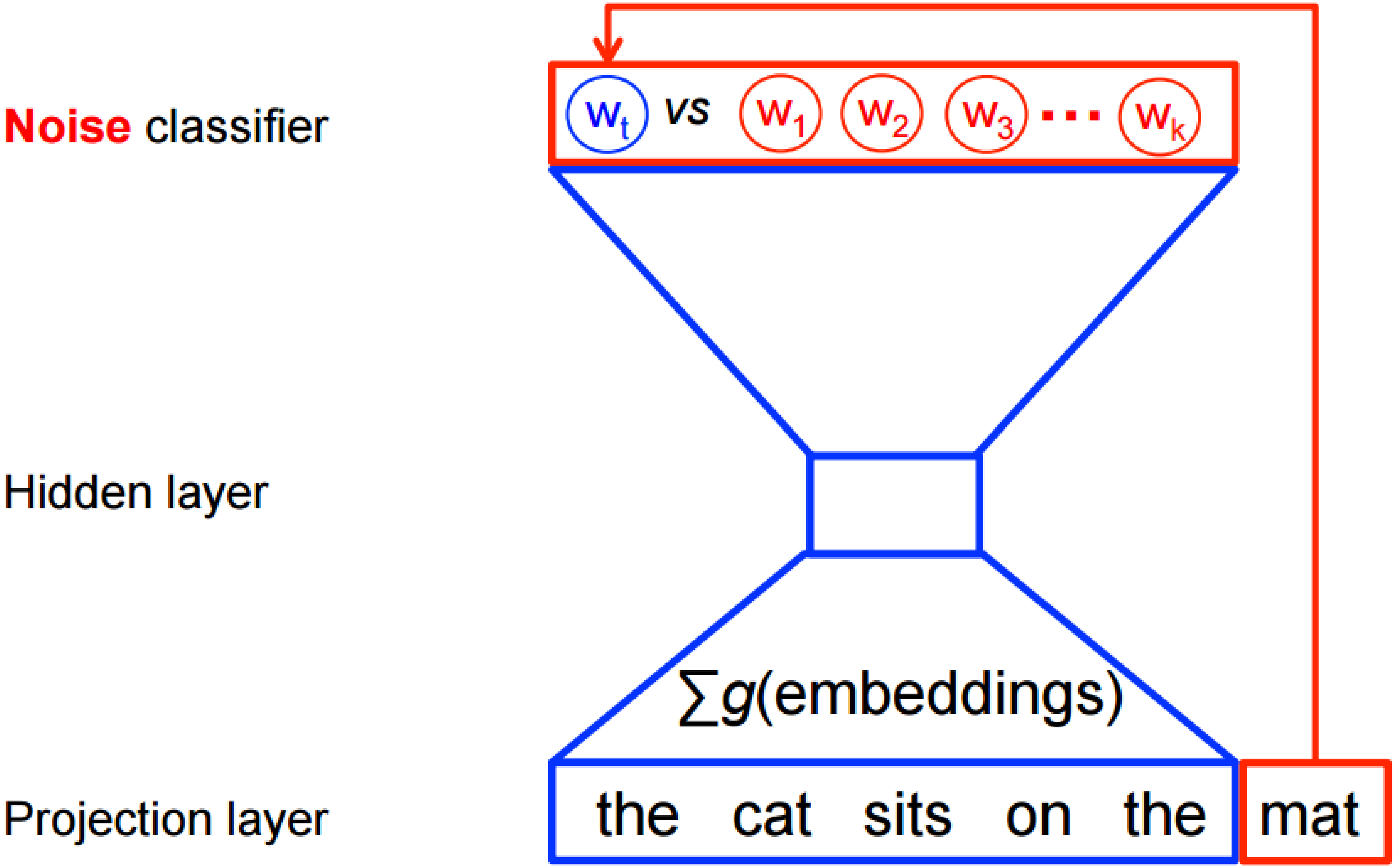
- $P(w_t|h) = softmax\left(score(w_t, h)\right) = \frac{e^{\{score(w_t,h)\}}}{\sum_{word\ w'\ in\ vocab.} e^{\{score(w',h)\}}}$

- $score(w_t, h)$ computes compatibility of word w_t with the context h (dot-product is used)

- Train the model by maximizing its log-likelihood:

  $- logP(w_t|h) = score(w_t, h) - log\left(\sum_{word\ w'\ in\ vocab.} e^{\{score(w',h)\}}\right)$

Simplified Softmax Problem as Classification



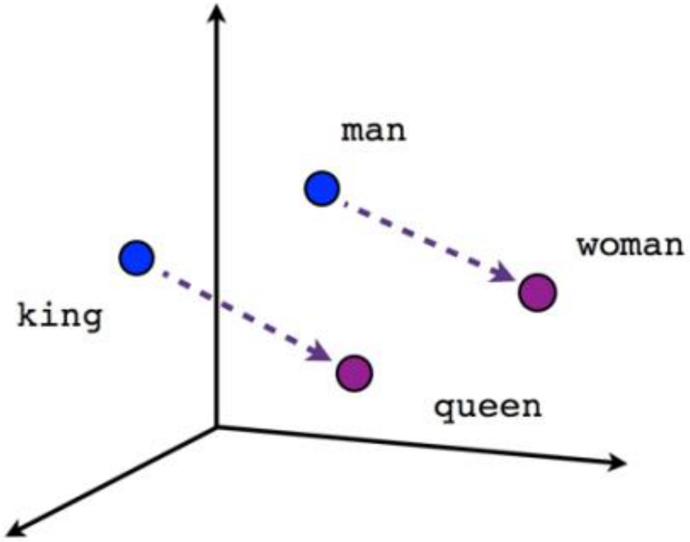
Negative Sampling

- $J_{NEG} = \log Q_\theta(D = 1|w_t, h) + k\mathbb{E}[\log Q_\theta(D = 0|\widetilde{w}, h)]$
- where

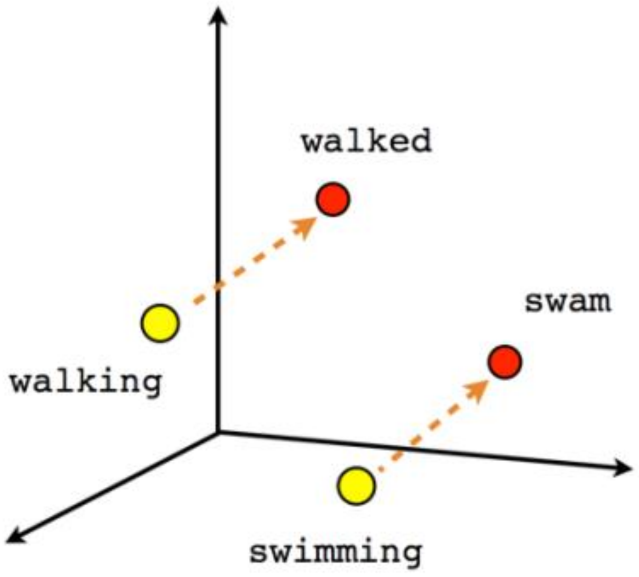
$\widetilde{w} \in P_{noise}$

$Q_\theta(D = 1|w_t, h)$ is binary logistic regression probability

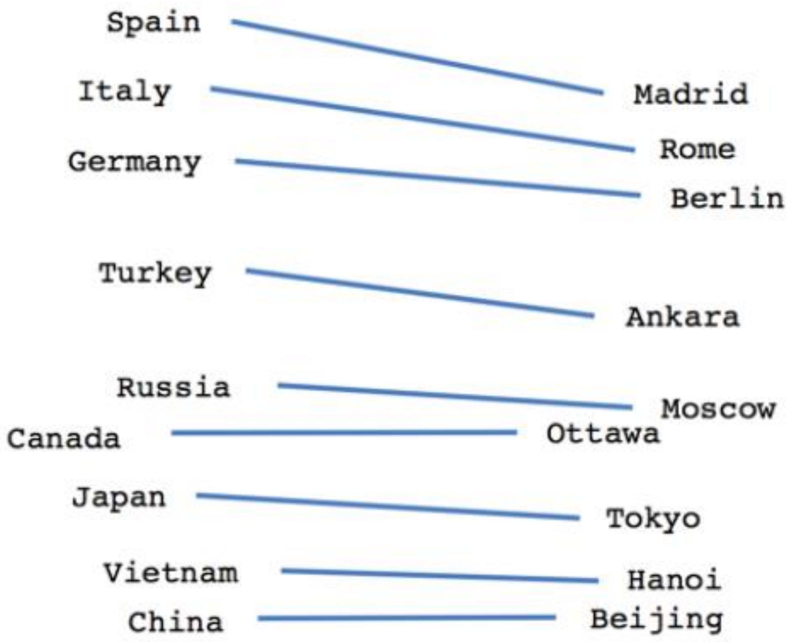
Evaluate Word2Vec



Male-Female



Verb tense



Country-Capital

Vector Addition & Subtraction

- vec(“Russia”) + vec(“river”) $\approx$ vec(“Volga River”)
- vec(“Germany”) + vec(“capital”) $\approx$ vec(“Berlin”)
- vec(“King”) - vec(“man”) + vec(“woman”) $\approx$ vec(“Queen”)

Embedding in Keras

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

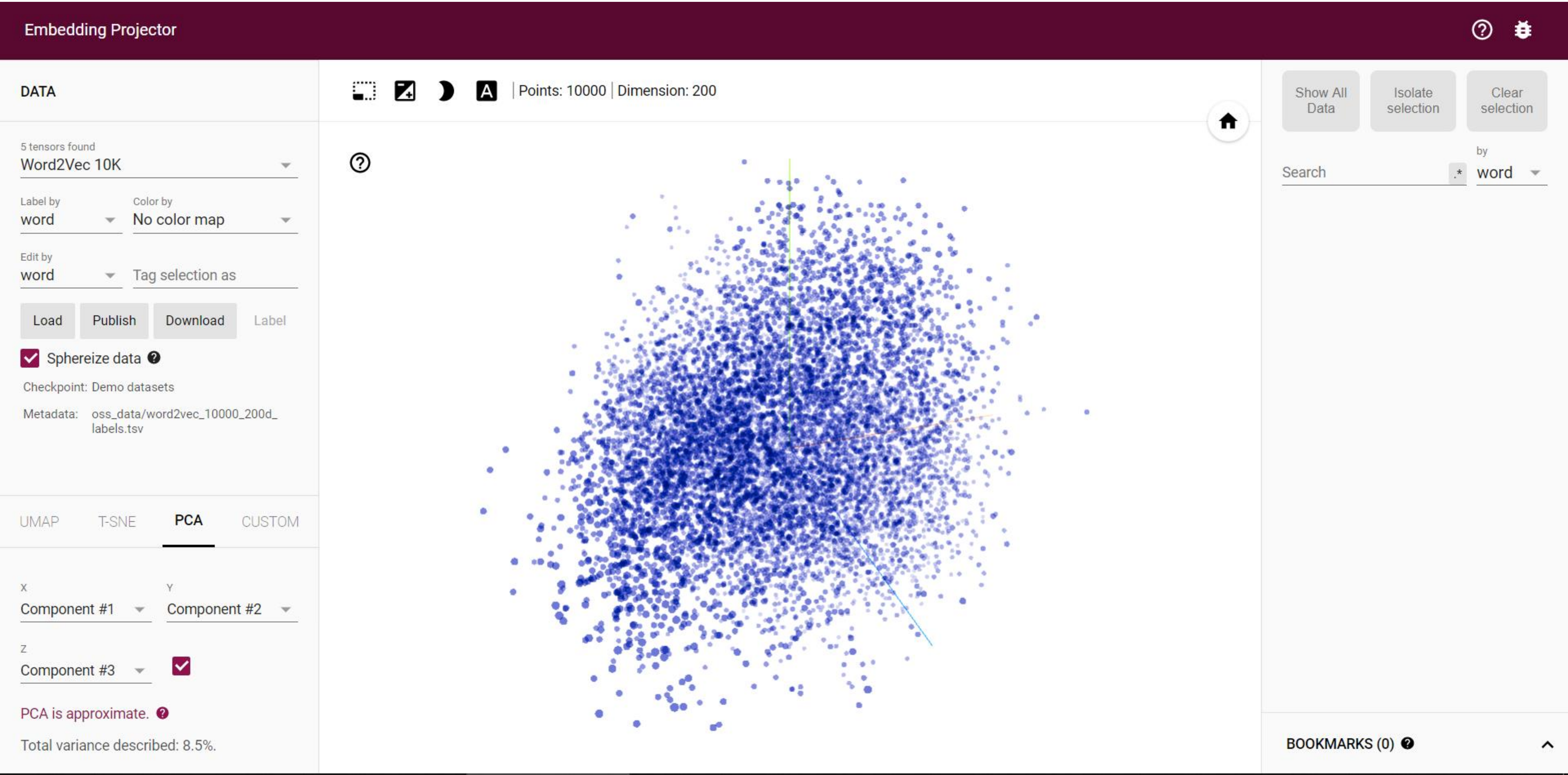
- Output dimension: Dimension of embedding vector

```
from keras.layers import Embedding

embedding_layer = Embedding(1000, 64)
```

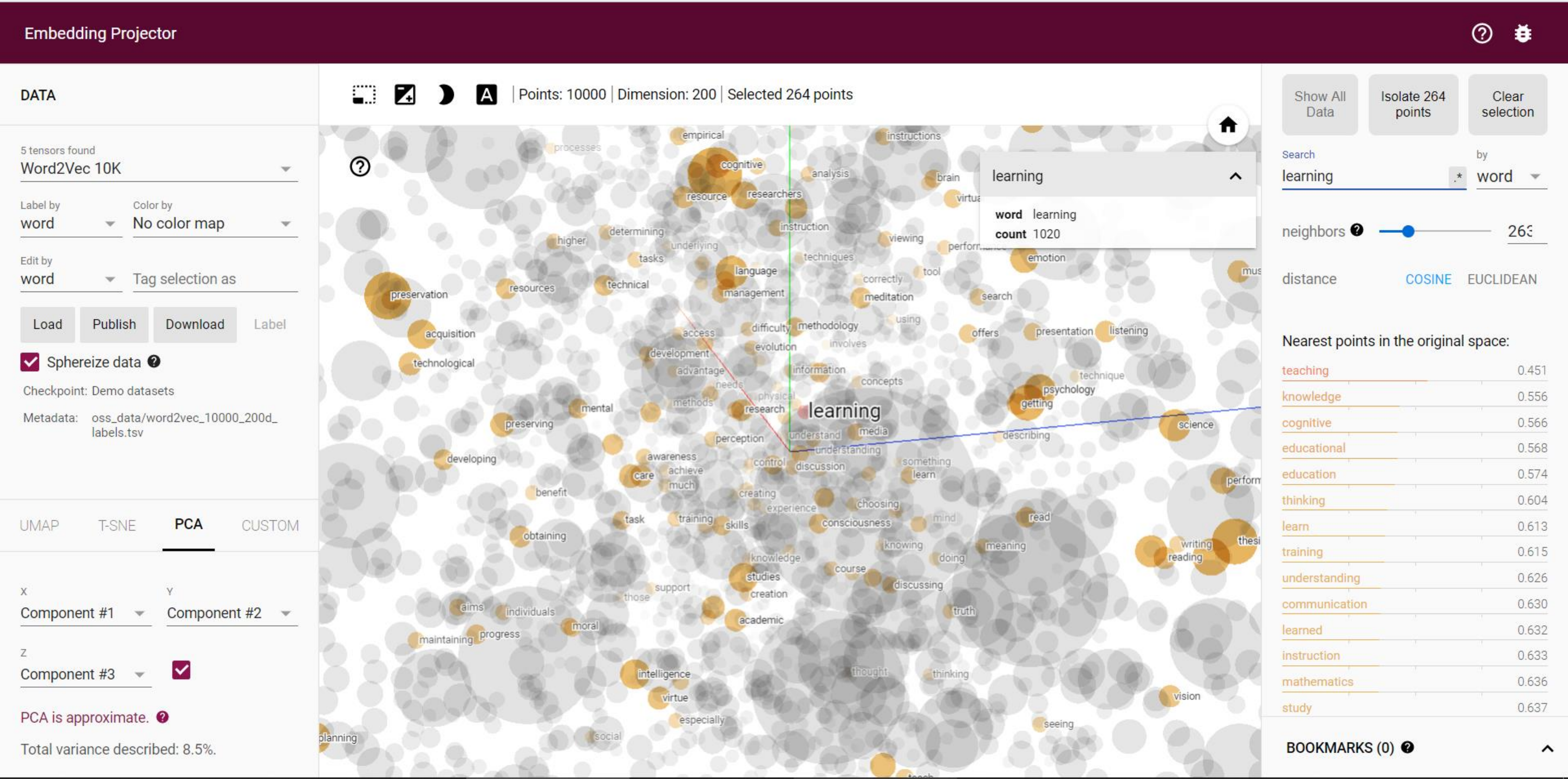
Using Embedding to Classify IMDB Data

```
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)
```

Embedding Project (projector.tensorflow.org/)



Neighbors of “Learning”



References

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