Word Embeddings

and its application in legal data feature extraction

Xu Xiao

Apr 15, 2019

Representation of linguistic data

Hierarchy

- Document
- Paragraph
- Sentence
- Words

Vector Space

- Represent an item (e.g. a word) with a vector (list) of numbers.
- Computers can't recognize a word

Word vectors

	emotional strength	positiveness	...
good	2	1	...
bad	2	0	...
great	4	1	...
terrible	4	0	...

More dimension means more information

- The properties of words can be endless
- Cannot generate automatically
- Different types of words require different properties

Representation of linguistic data

"Words that occur in the same contexts tend to have similar meanings."

(Harris, 1954)

"You shall know a word by the company it keeps."

(Firth, 1957)

Counting over context

	can	doesn't	hurt	...
eat	1	0	0	...
glass	0	0	0	...
it	0	1	0	...
me	0	0	1	...

Raw counting method uses numbers of occurrence as vector value

Given the sentence *I can eat glass and it doesn't hurt me*, suppose we take 2 adjacent words as context for each word, the vector space is shown to the left.

- Vector dimension equals to the length of vocabulary
- The vectors are sparse
- Frequent words may characterize vectors

tf-idf

$$\mathrm{tf}(t, d) = f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$$

$$\mathrm{idf}(t, D) = \log \frac{N}{n_t} = -\log \frac{n_t}{N}$$

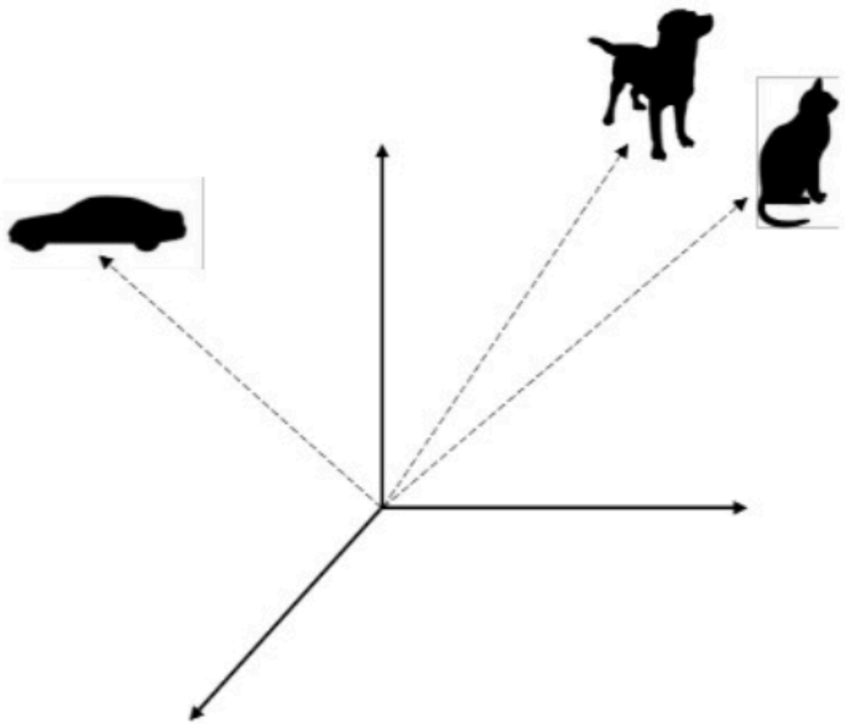
$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

Tf-idf adjust the weights with frequency in document

- Vector dimension equals to the length of vocabulary
- The vectors are sparse
- ~~Frequent words may characterize vectors~~

We need to embedded the information into lower-dimension word vectors.

Word embeddings



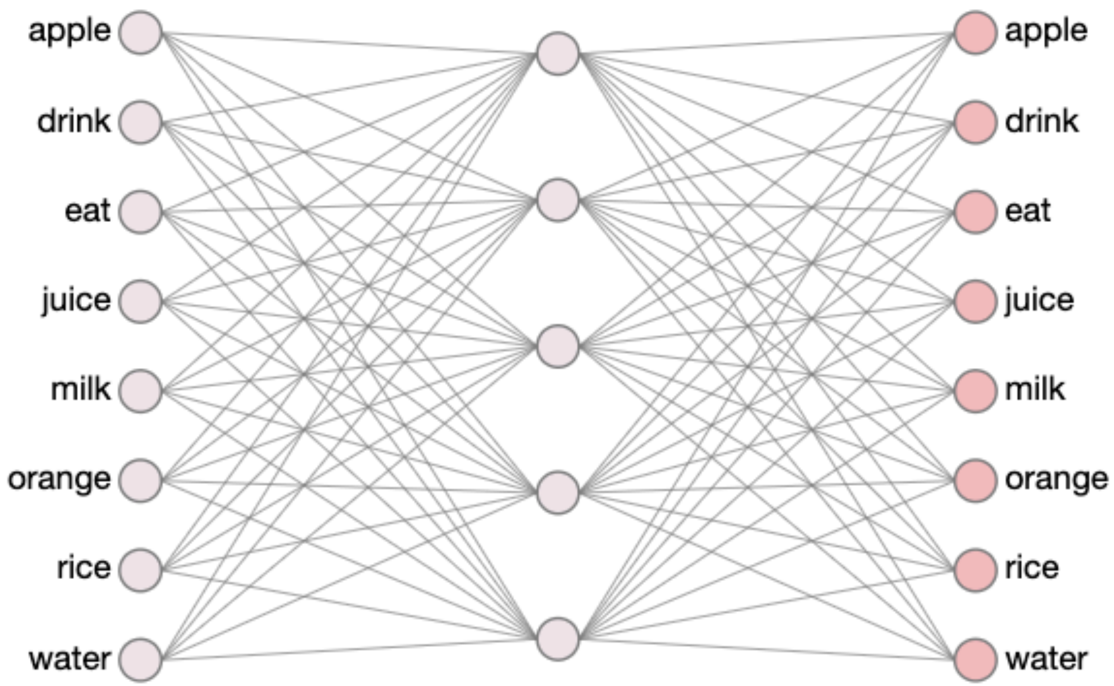
The low-dimension and dense word vectors are called word embeddings.

It involves a mathematical embedding from a space with one dimension per word to a continuous vector space with a much lower dimension.

Several famous models to produce embeddings:

- word2vec
- GloVe
- LDA

word2vec

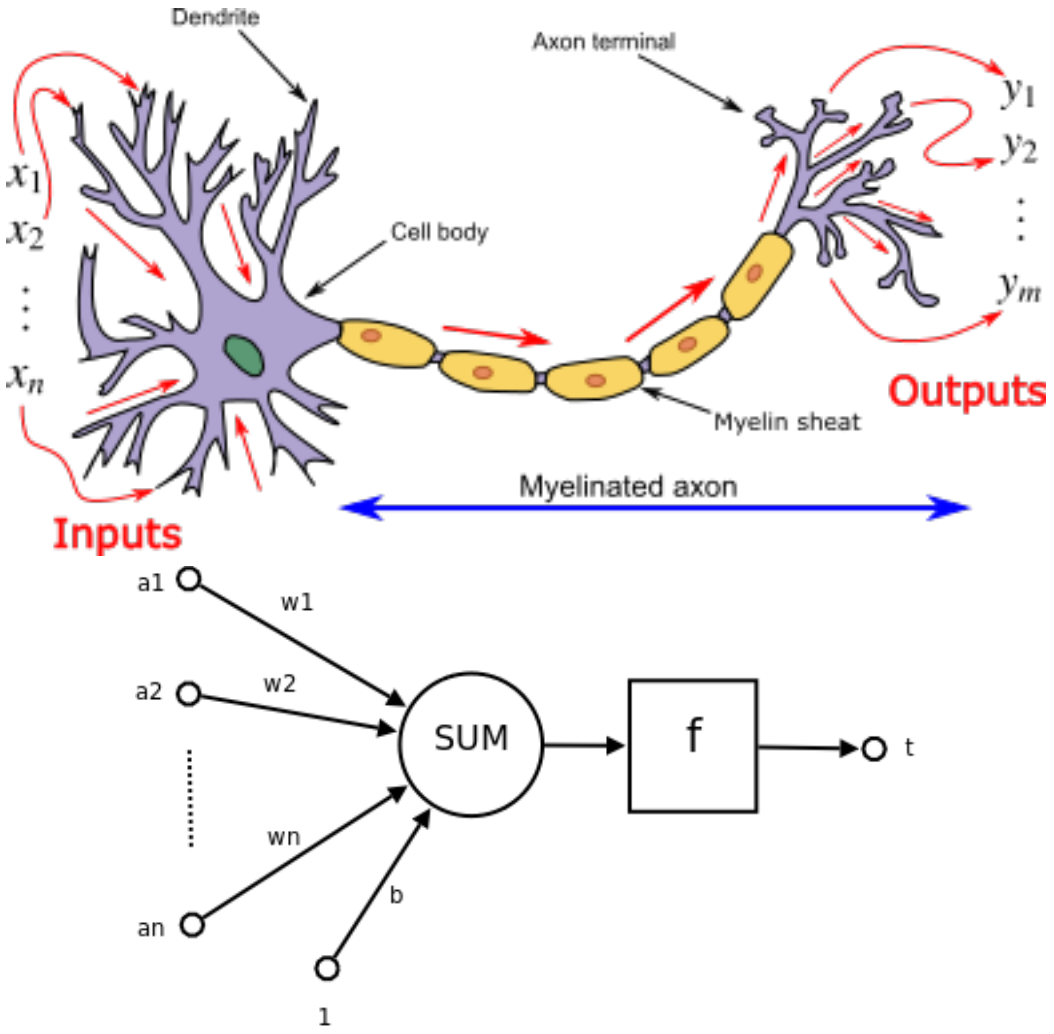


word2vec uses a Neural Network with a "bottleneck" to achieve lower dimension

The Neural Network takes word and content as input and output, and has a hidden layer with set number of nodes.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

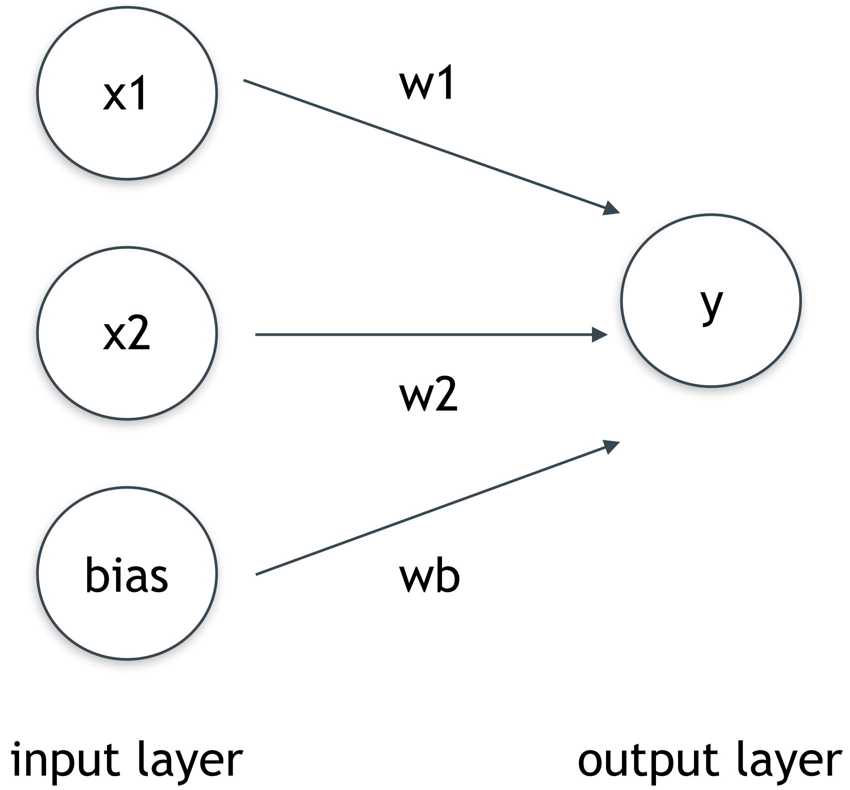
A brief review of Neural Network



A neural network consists of “neurons” which is based on neurons in human brain.

A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns.

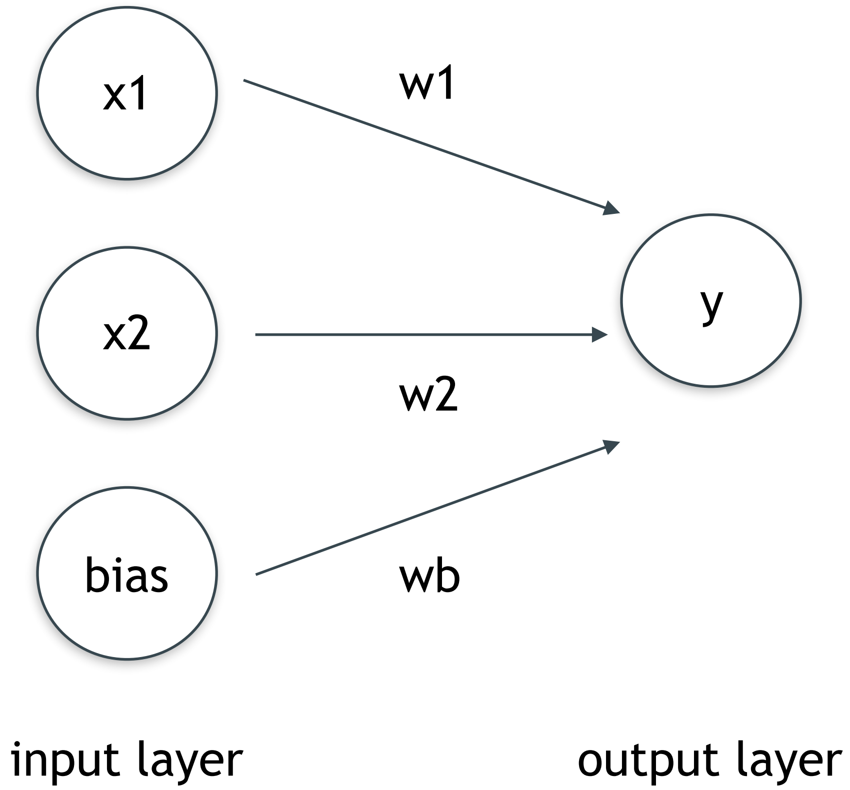
Example: single layer perceptron



Suppose we have a simple network and we want to solve ‘OR’ problem

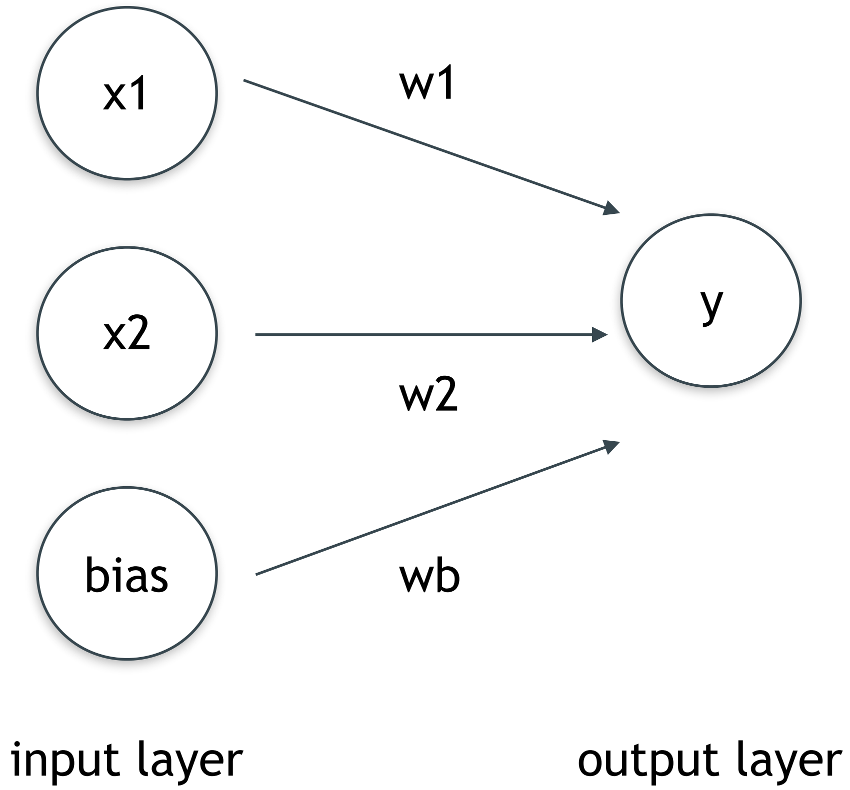
We set learning rate to 0.5, initial weights w1=w2=wb=0.5.

Example: single layer perceptron



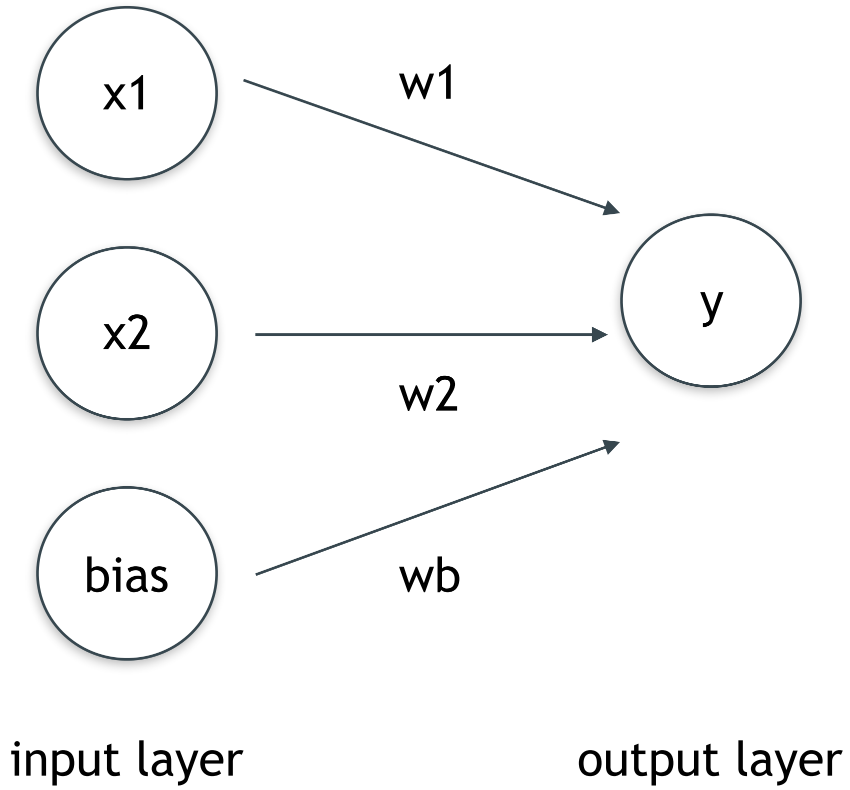
x1	x2	w1	w2	wb	y	t	a(t-y)
0	0	0.5	0.5	0.5	0.5	0	-0.25

Example: single layer perceptron



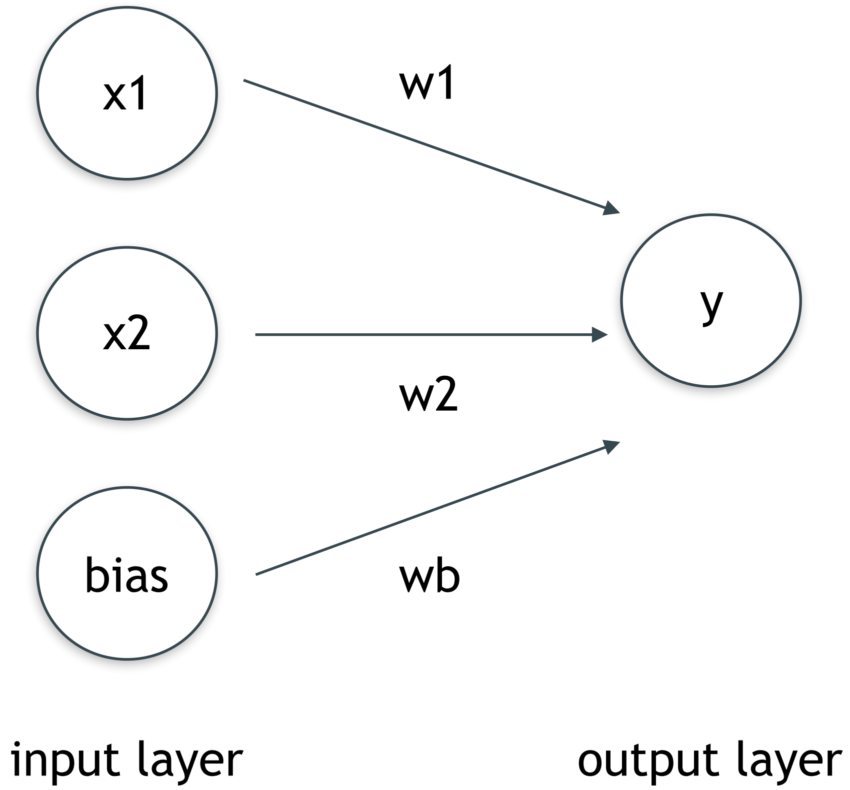
x1	x2	w1	w2	wb	y	t	a(t-y)
0	0	0.5	0.5	0.5	0.5	0	-0.25
1	0	0.5	0.5	0.25	0.75	1	0.125

Example: single layer perceptron



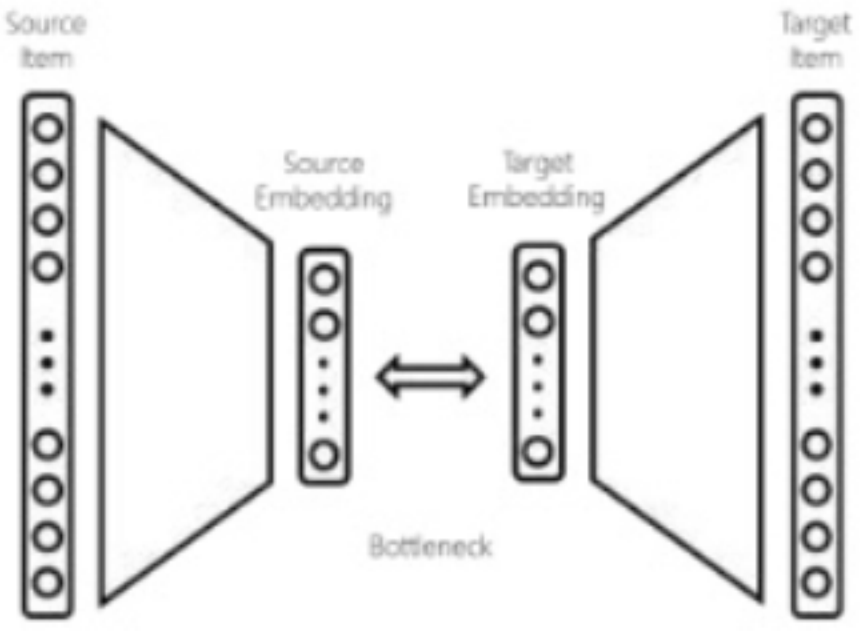
x1	x2	w1	w2	wb	y	t	a(t-y)
0	0	0.5	0.5	0.5	0.5	0	-0.25
1	0	0.5	0.5	0.25	0.75	1	0.125
0	1	0.625	0.5	0.375	0.875	1	0.0625

Example: single layer perceptron



x1	x2	w1	w2	wb	y	t	a(t-y)
0	0	0.5	0.5	0.5	0.5	0	-0.25
1	0	0.5	0.5	0.25	0.75	1	0.125
0	1	0.625	0.5	0.375	0.875	1	0.0625
1	1	0.625	0.5625	0.4375	1.625	1	0.3125

Structure of word2vec models

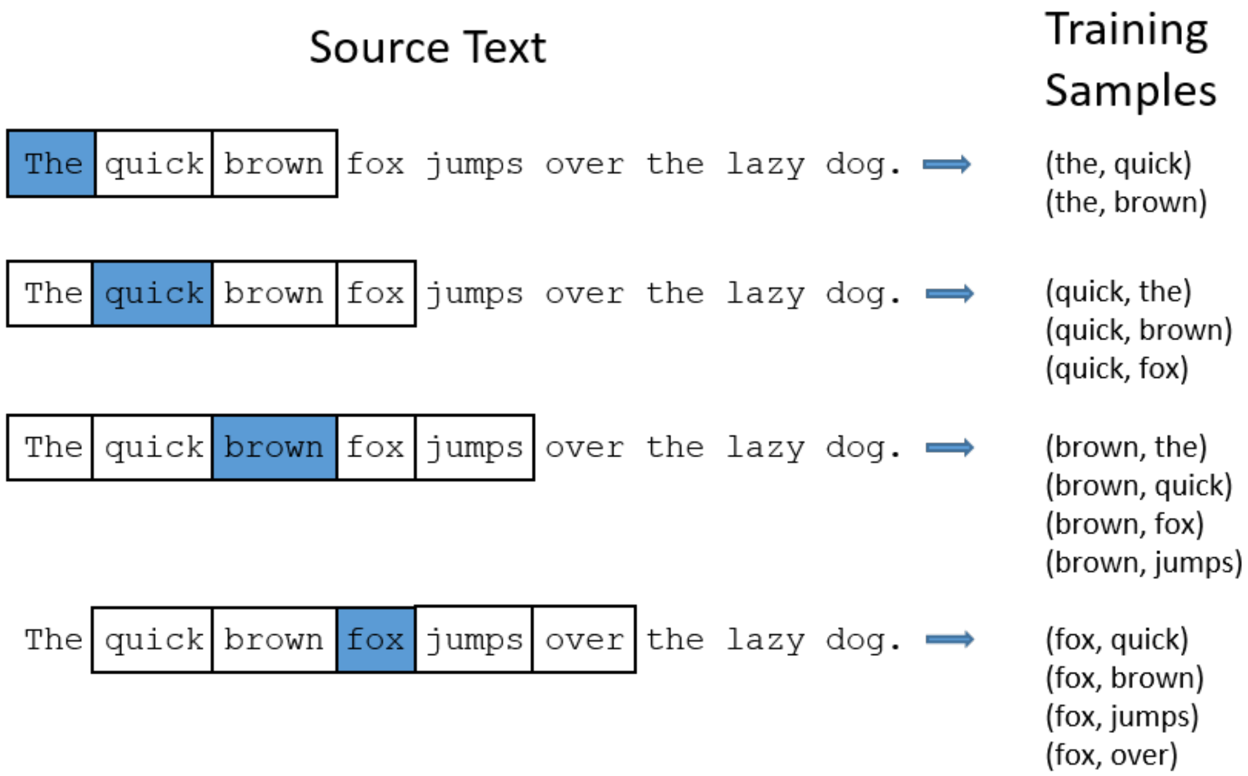


Inputs: One-hot vectors (vectors with only one non-zero element, which is 1)

Example: *I can eat glass and it doesn't hurt me*

I: [1 0 0 0 0 0 0 0 0]
can: [0 1 0 0 0 0 0 0 0]
eat: [0 0 1 0 0 0 0 0 0]
...

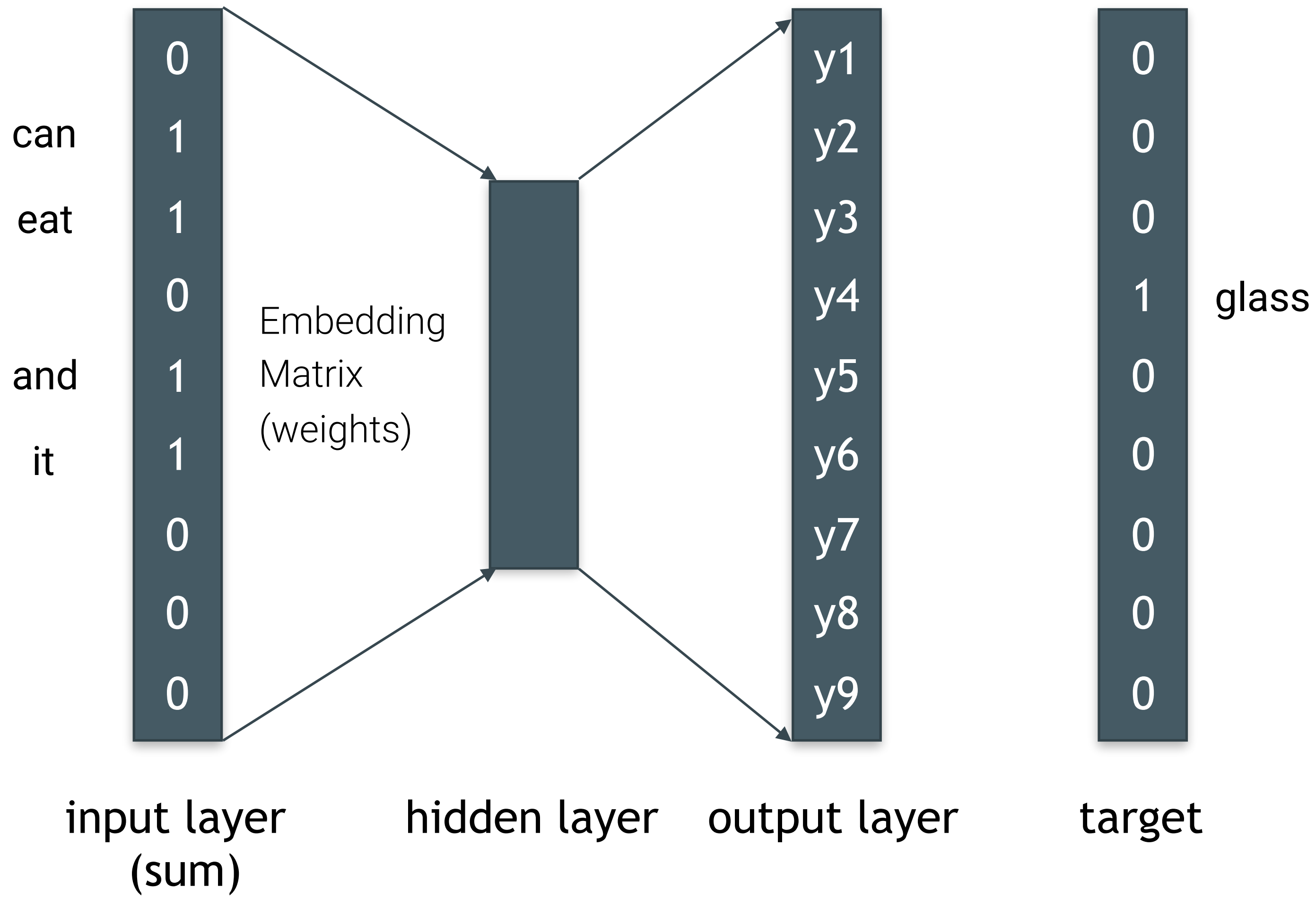
Structure of word2vec models



Two models: Skipgram vs CBOW

Skipgram: Using the middle word to predict its context
CBOW (continuous bag-of-words): Using the context to predict the middle word

Continuous bag-of words

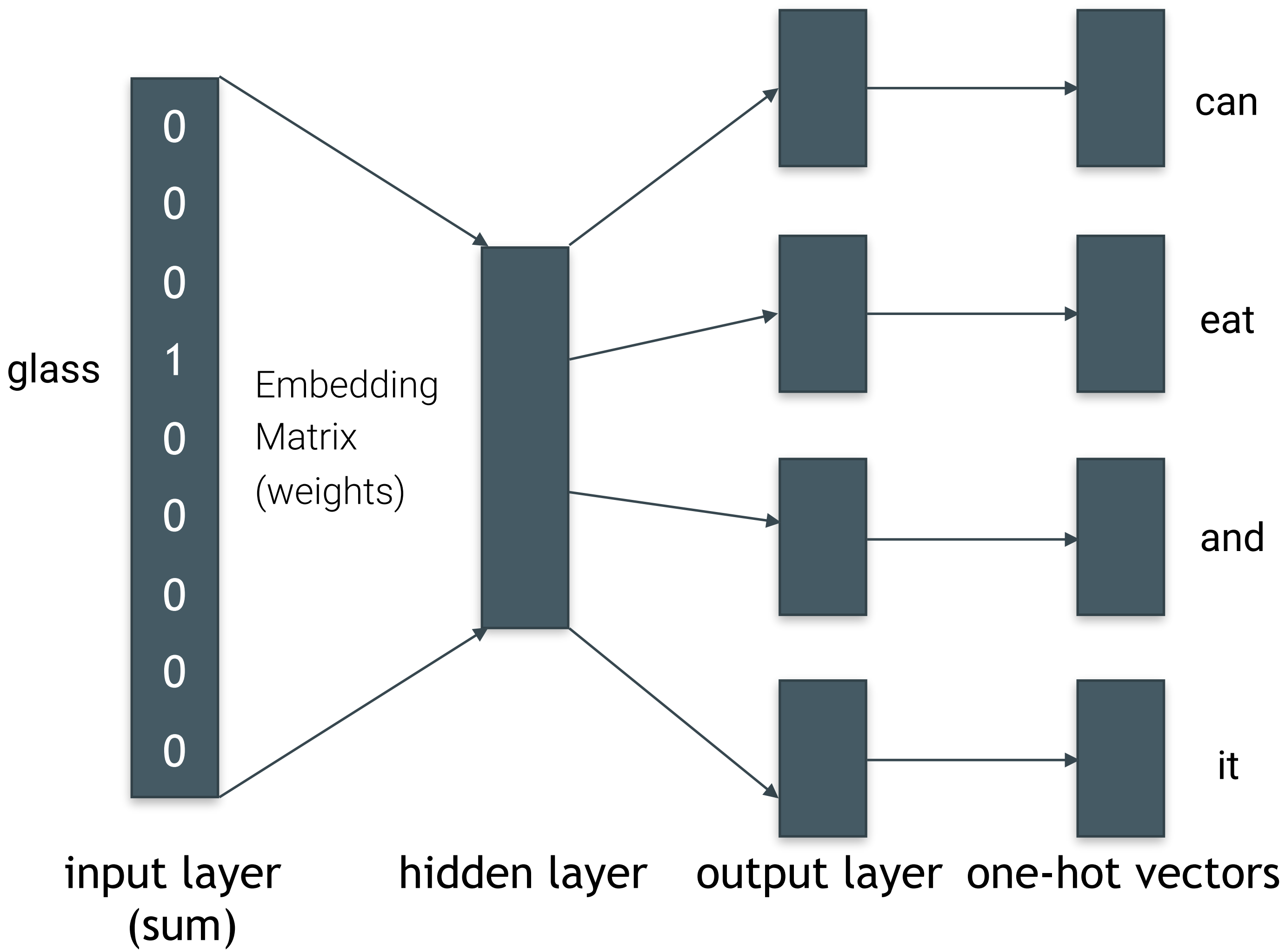


CBOW takes multiple one-hot vectors as input to predict middle word

Example: *I can eat **glass** and it doesn't hurt me*

Here the model takes 4 one-hot input vectors and we aim to have the output corresponds to *glass*. The values from weights matrix after training are the embeddings we're looking for.

Skipgram

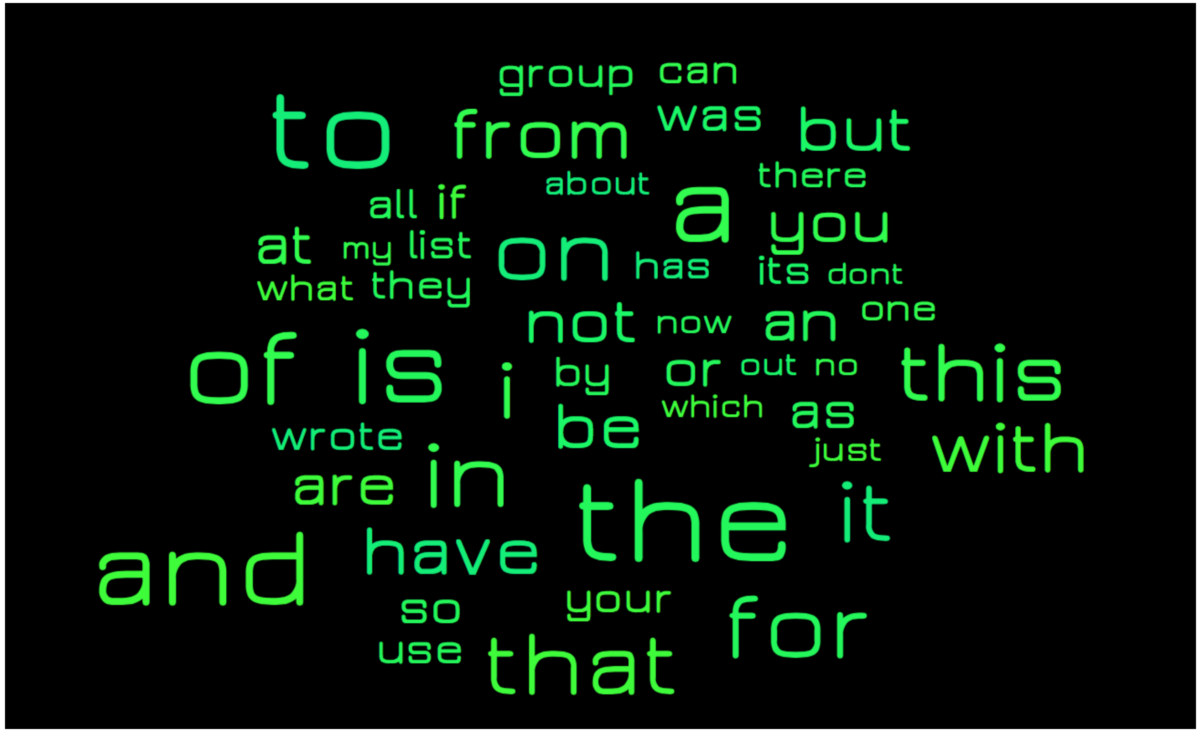


Skipgram takes the middle word as input to predict its context

Example: *I can eat* ***glass*** *and it doesn't hurt me*

Here the model takes an one-hot input vectors representing *glass* and we aim to have the output corresponds to its context. The values from weights matrix after training are the embeddings we're looking for.

Subsampling of Frequent Words



To counter the imbalance between the rare and frequent words we need subsampling

Each word w_i in the training set is discarded with probability computed by the formula

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

where $f(wi)$ is the frequency of word wi and t is a chosen threshold typically around 10^-5.

Using word2vec package

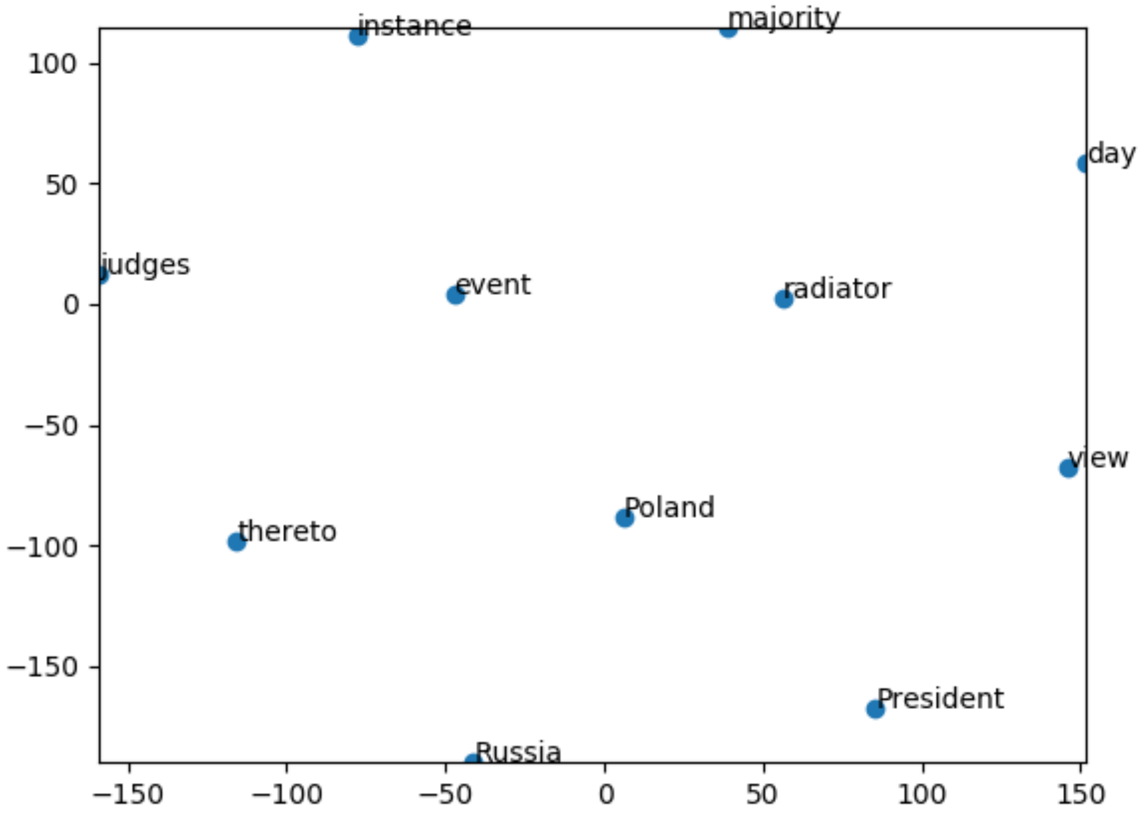
Binary package

- https://github.com/tmikolov/word2vec
- Compile and download corpus
- Train your model

Using gensim

- Python package, install with pip
- Optimization over years
- Integration to your code
- https://radimrehurek.com/gensim/models/word2vec.html

Similarity

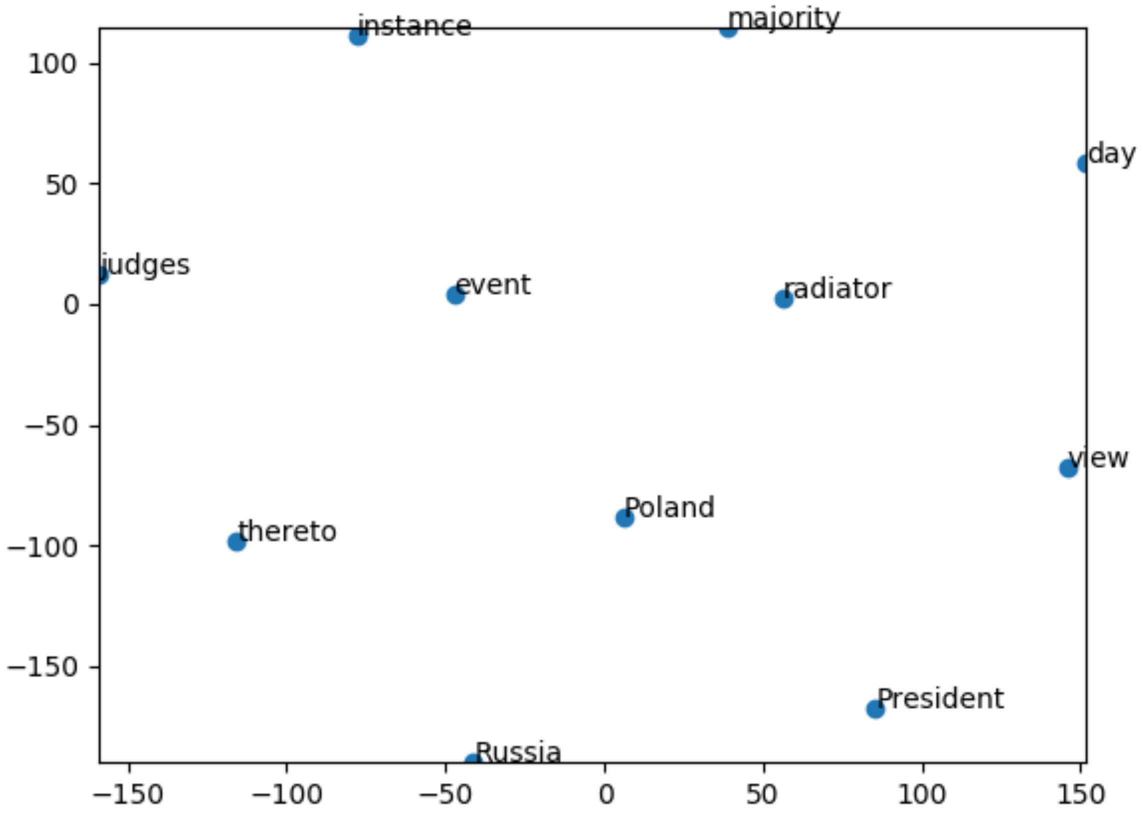


One of the most direct application of word embeddings is to calculate similarity

The most common similarity measure of vectors is **Cosine similarity**.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

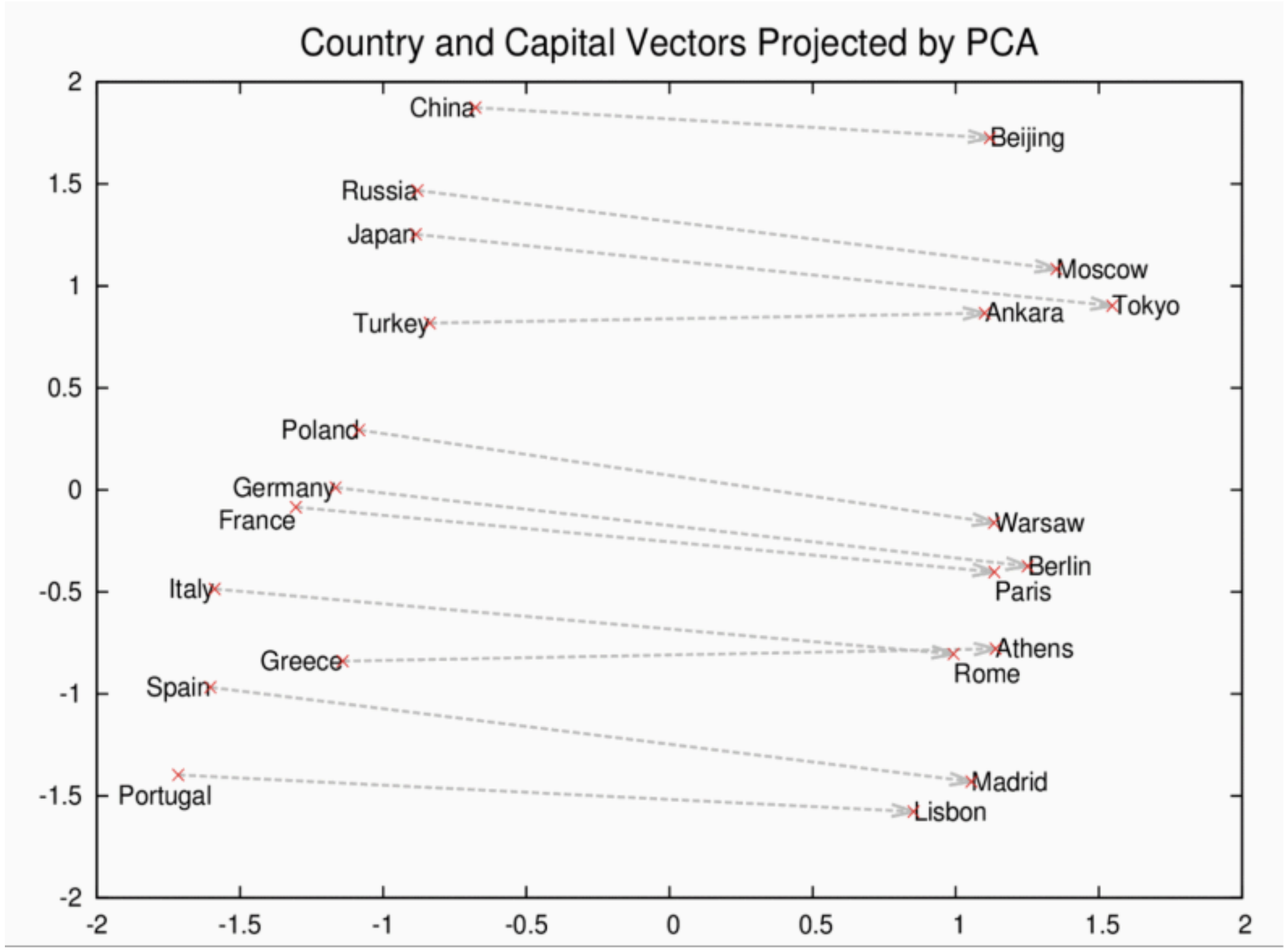
TSNE graph



You can use t-SNE graph to visualize high-dimension vectors

Visualizing Data using t-SNE, Maaten and Hinton, 2008

Analogies



Word embeddings are capable of learning finding analogies

Suppose A, B, C and D are vectors representing words, If B - A = D - C, we could say the relations between AB and CD are similar.

Example:
vector("king") - vector("man") = vector("queen") - vector("woman")

Feature extraction

Table 4: 10-fold ross-validation results: F1-score (macro) per article

Model	art3	art5	art6	art8
N-grams	0.78	0.68	0.61	0.50
Word embeddings	**0.85**	**0.81**	**0.75**	**0.70**

Word embeddings are usually combined with other classifiers

Using the vectors created for words in the word2vec dataset we determine the vectors for our training set and test set. However, as we do not want values per word, but per document, we average the vectors for each word in the entire document, and normalize them by using tf-idf weighting, which take into account account the number of documents (i.e. cases) in which each word occurs.

Conclusions

Advantages

- A powerful word representation
- Incorporate well with other models

Disadvantages

- Need to custom parameters accordingly
- Need specified dataset
- Require large dataset and heavy computing

Questions

For lab session, please go to
https://github.com/WillSkywalker/word-embedding

Xu Xiao
Apr 15, 2019