Vector Representation of Text

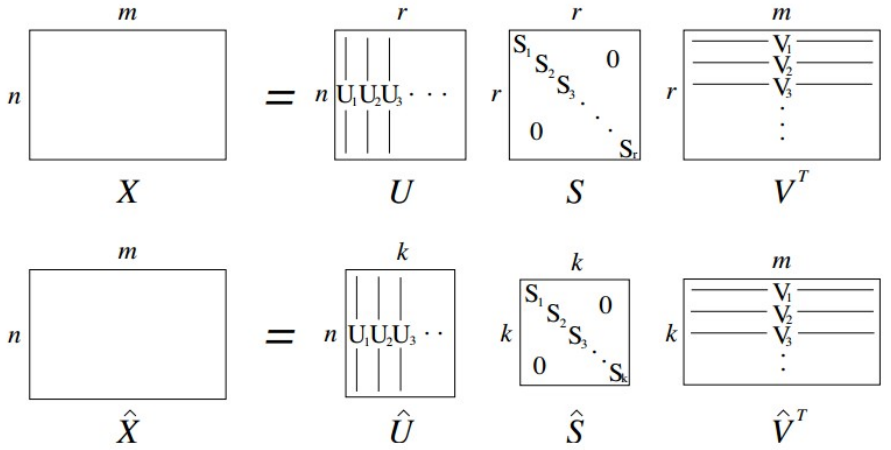
Deep Learning for NLP

To compare pieces of text

- We need effective representation of :
  - Words
  - Sentences
  - Text
- Approach 1: Use existing thesauri or ontologies like WordNet and Snomed CT (for medical). Drawbacks:
  - Manual
  - Not context specific
- Approach 2: Use co-occurrences for word similarity. Drawbacks:
  - Quadratic space needed
  - Relative position and order of words not considered

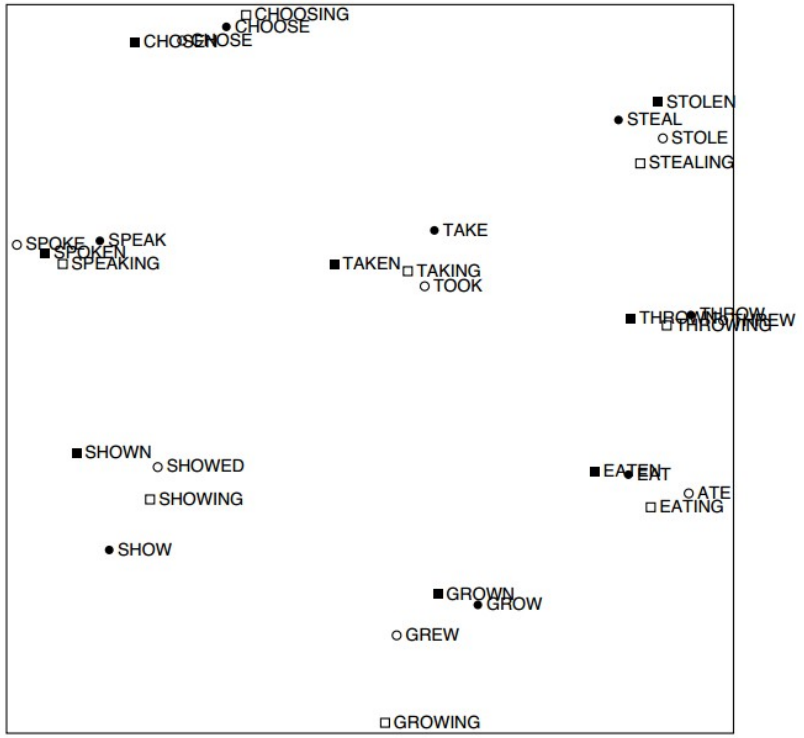
Approach 3: low dimensional vectors

- Store only “important” information in fixed, low dimensional vector.
- Singular Value Decomposition (SVD) on co-occurrence matrix
  - is the best rank *k* approximation to *X* , in terms of least squares
  - Motel = [0.286, 0.792, -0.177, -0.107, 0.109, -0.542, 0.349, 0.271]
- m = n = size of vocabulary
- is the same matrix as S except that it contains only the top largest singular values



Approach 3: low dimensional vectors

- An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence



Problems with SVD

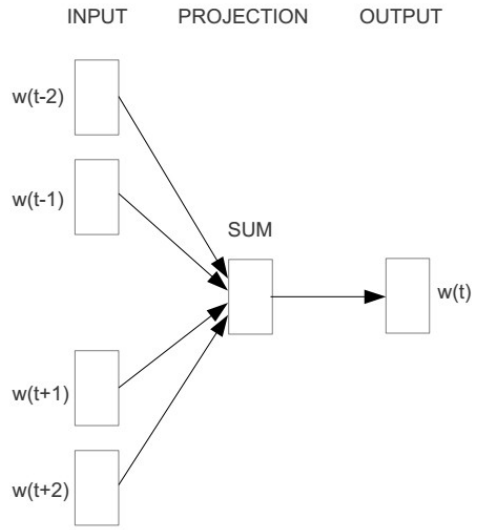
- Computational cost scales quadratically for n x m matrix: O(mn^2) flops (when n<m)
- Hard to incorporate new words or documents
- Does not consider order of words

word2vec approach to represent the meaning of word

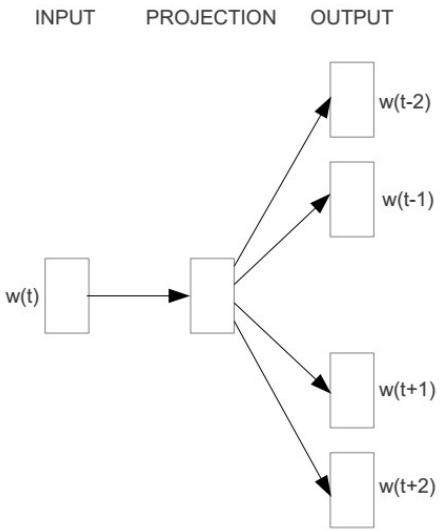
- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key idea: Predict surrounding words of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

Represent the meaning of **word** – word2vec

- 2 basic neural network models:
  - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.



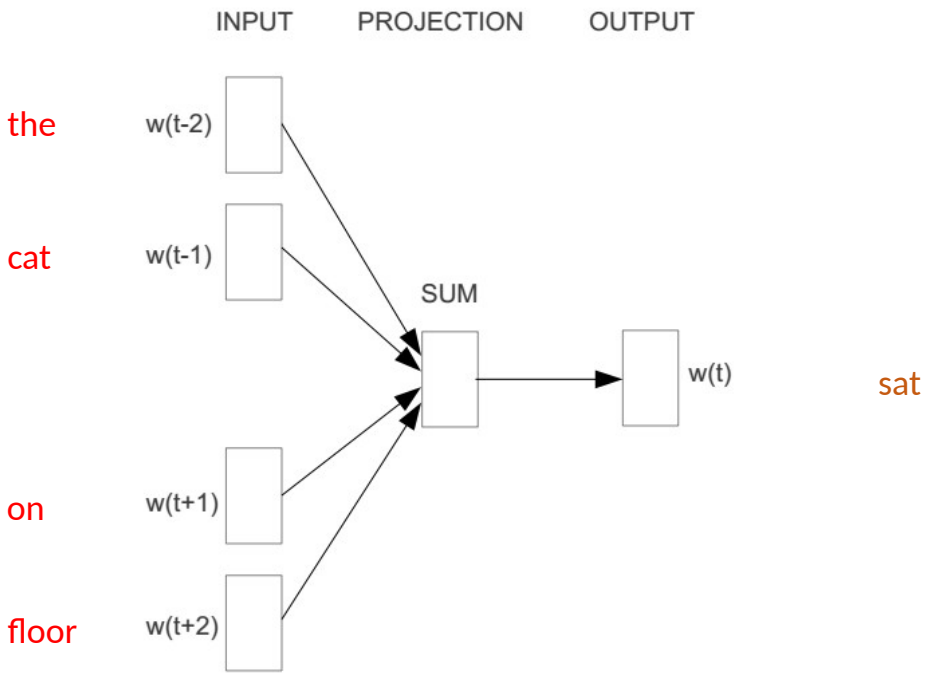
CBOW

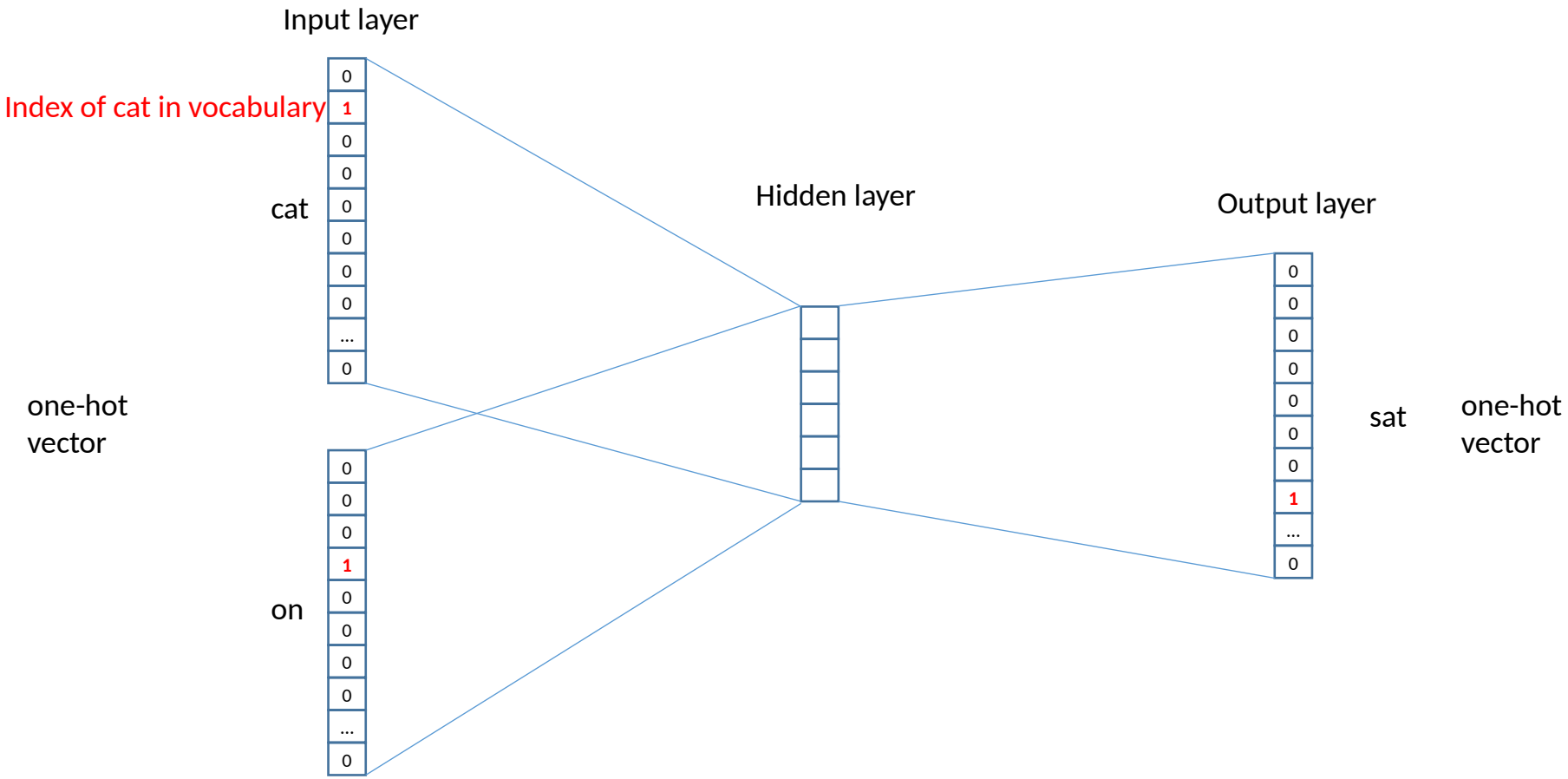


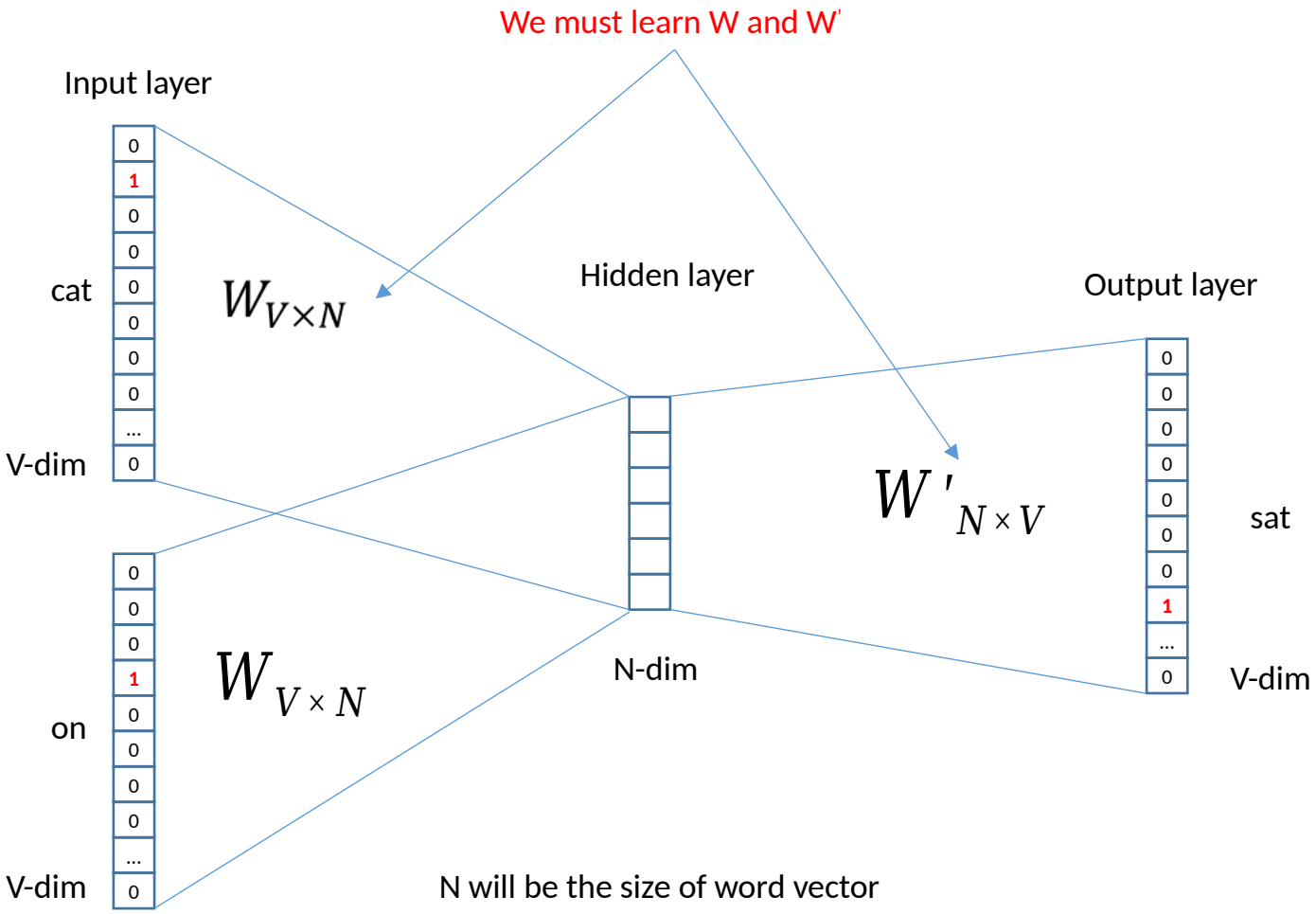
Skip-gram

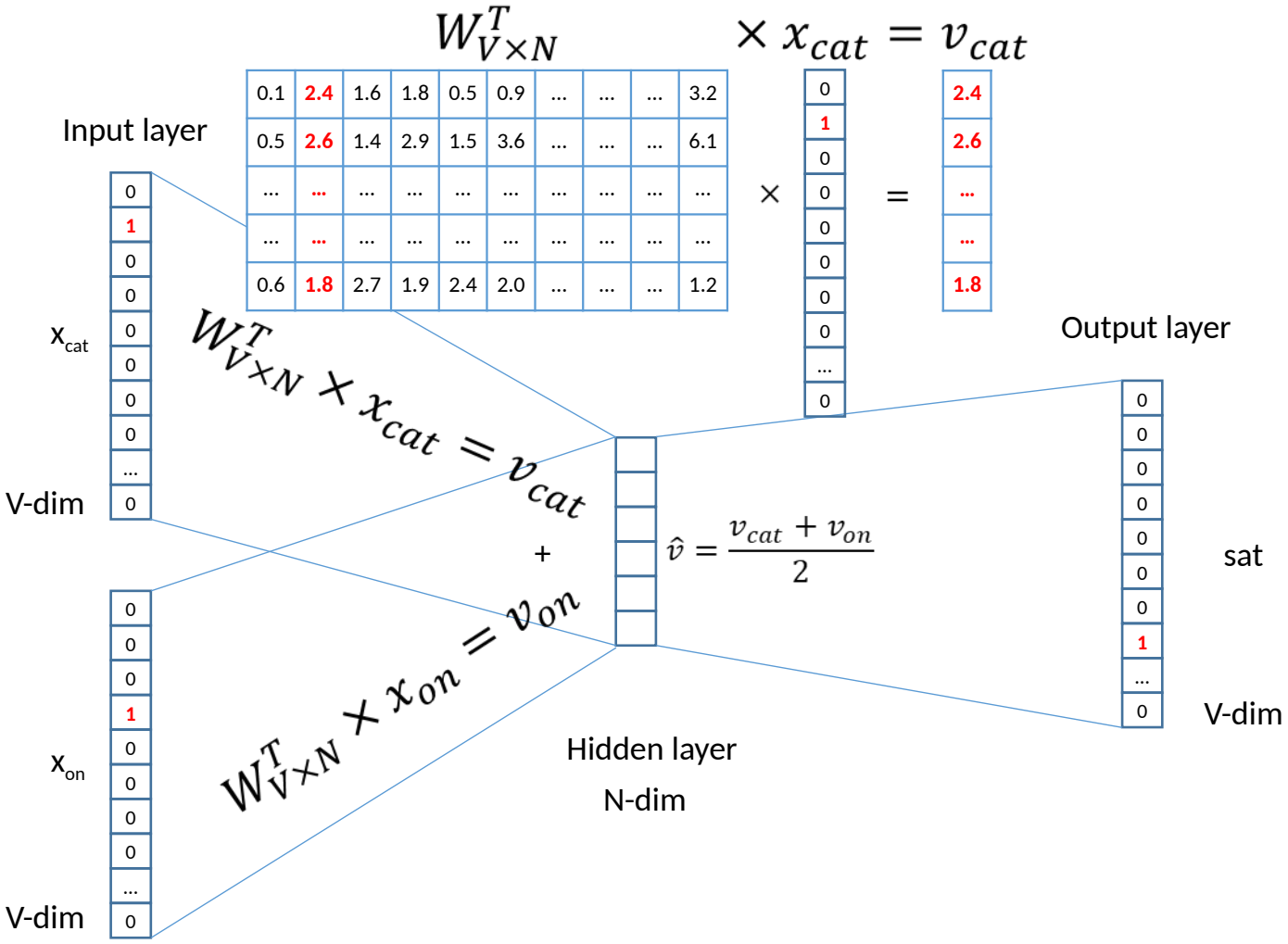
Word2vec – Continuous Bag of Word

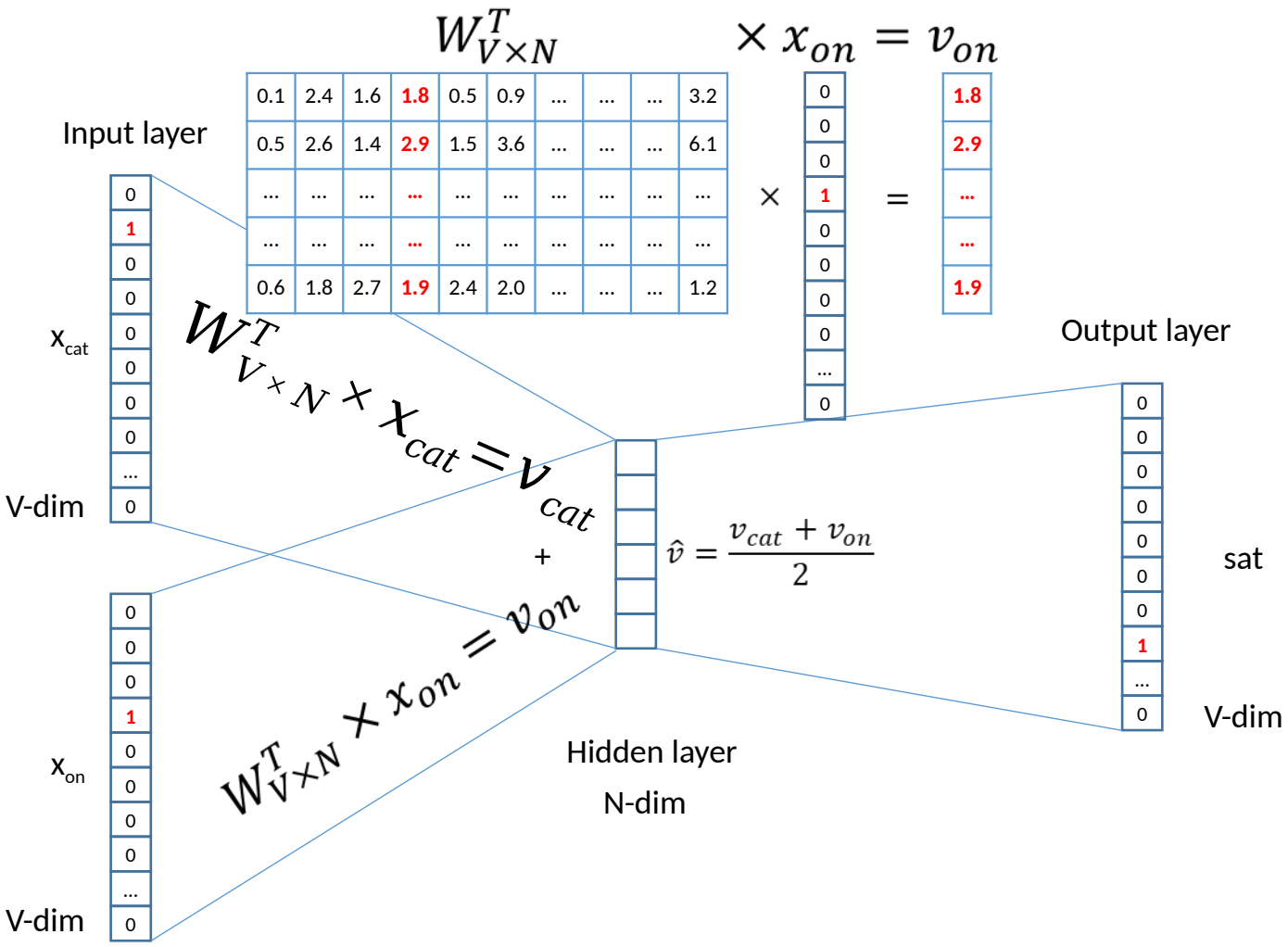
- E.g. “The cat sat on floor”
  - Window size = 2

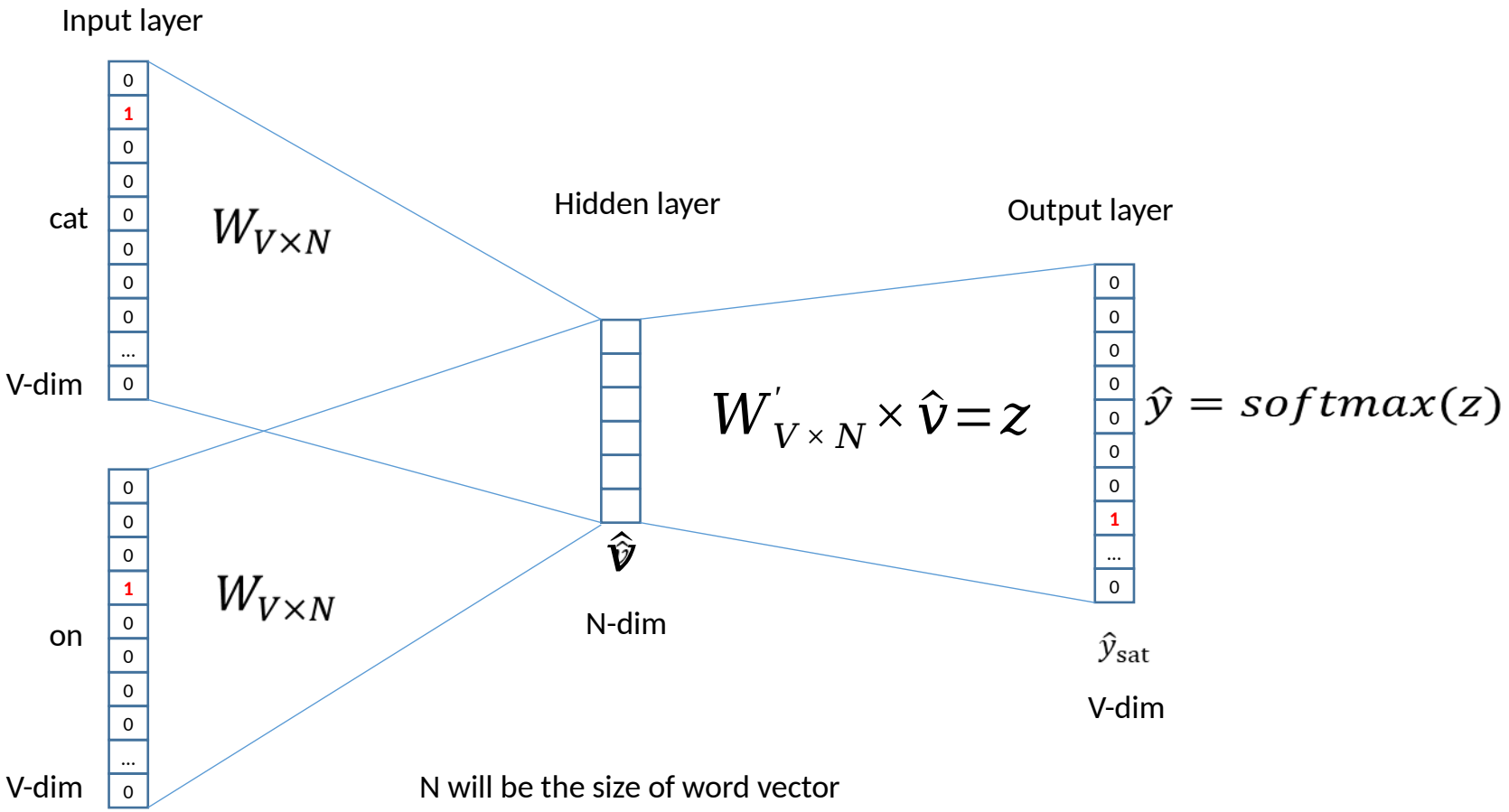


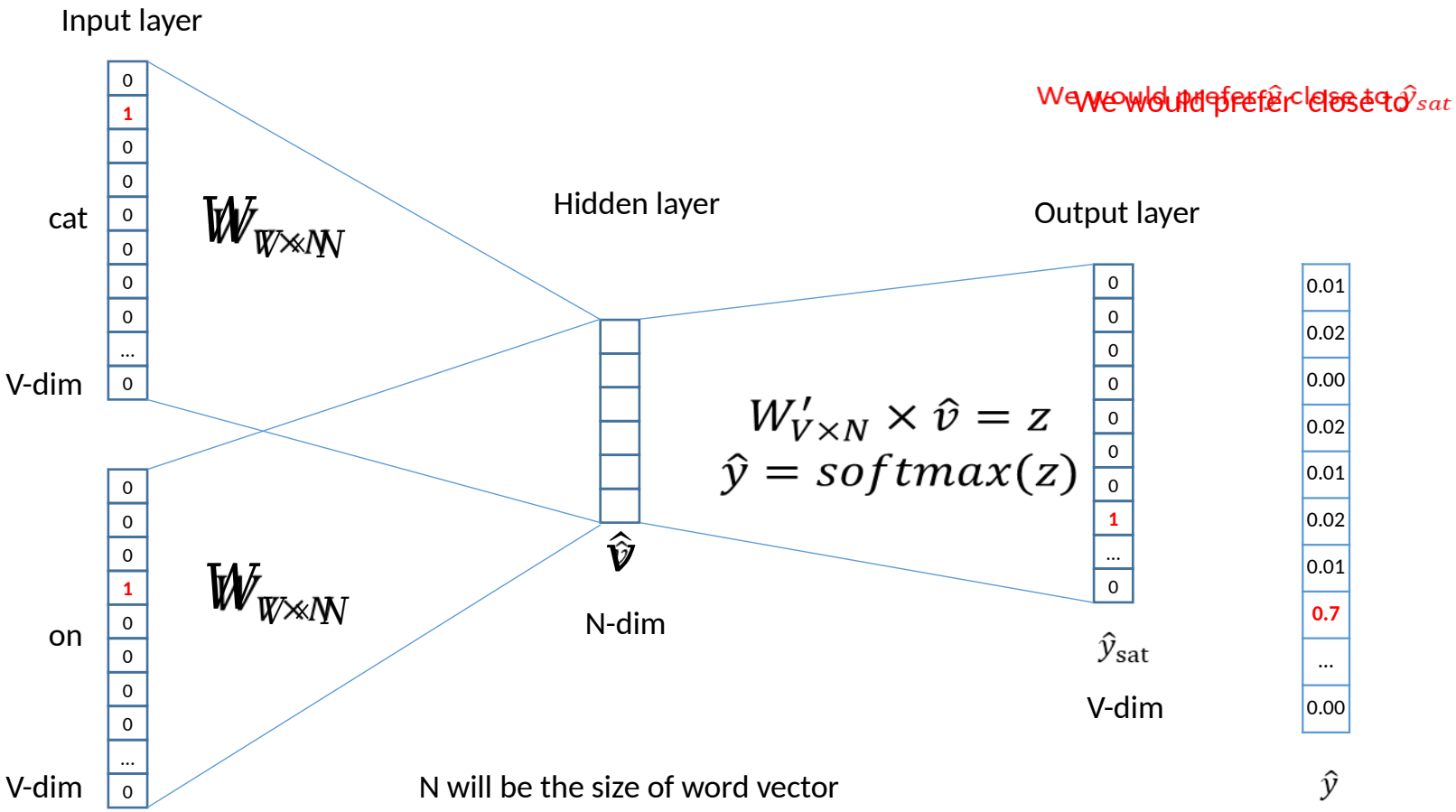

Input layer
Index of cat in vocabulary
cat
one-hot vector
on
Hidden layer
Output layer
sat
one-hot vector

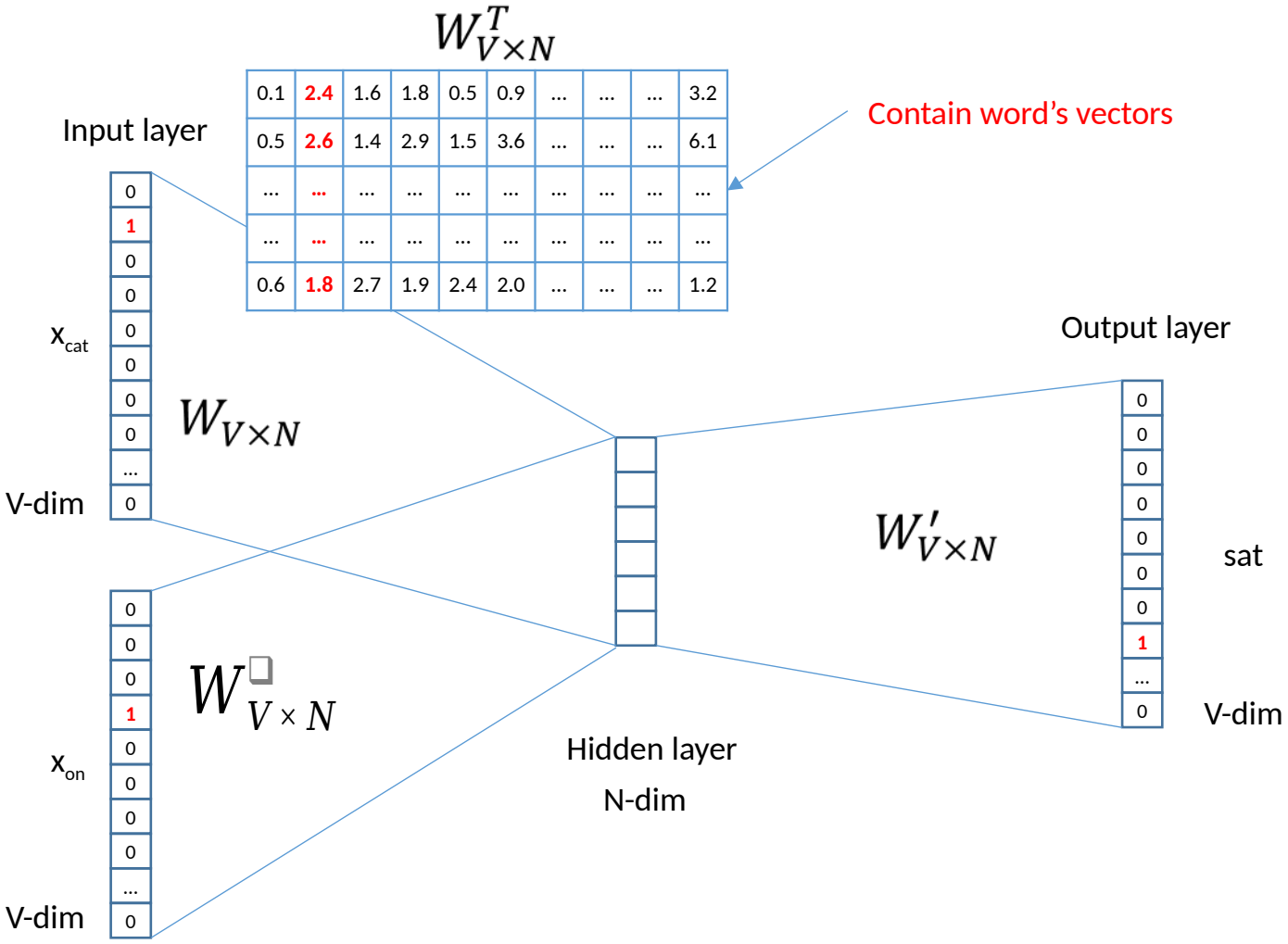

We must learn W and W'
Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
N-dim
$W'_{N\times V}$
Output layer
sat
V-dim
N will be the size of word vector


$W^T_{V\times N}$
$\times x_{cat} = v_{cat}$
Input layer
0.1 2.4 1.6 1.8 0.5 0.9 ... ... 3.2
0.5 2.6 1.4 2.9 1.5 3.6 ... ... 6.1
... ...
... ...
0.6 1.8 2.7 1.9 2.4 2.0 ... ... 1.2
0 1 0 0 0 0 0 ... 0
×
=
2.4 2.6 ... ... 1.8
Output layer
x_{cat}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$W^T_{V\times N} \times x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
Hidden layer
N-dim
sat
V-dim
x_{on}
V-dim


$W^T_{V\times N}$
$\times x_{on} = v_{on}$
Input layer
x_{cat}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$W^T_{V\times N} \times x_{on} = v_{on}$
x_{on}
V-dim
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
Hidden layer
N-dim
Output layer
sat
V-dim


Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
$W'_{V\times N} \times \hat{v} = z$
Output layer
$\hat{y} = softmax(z)$
$\hat{y}_{sat}$
V-dim
N will be the size of word vector


Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
$W'_{V\times N} \times \hat{v} = z$
$\hat{y} = softmax(z)$
Output layer
$\hat{y}_{sat}$
V-dim
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
$\hat{y}$
N will be the size of word vector



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

Some interesting results

Word Analogies

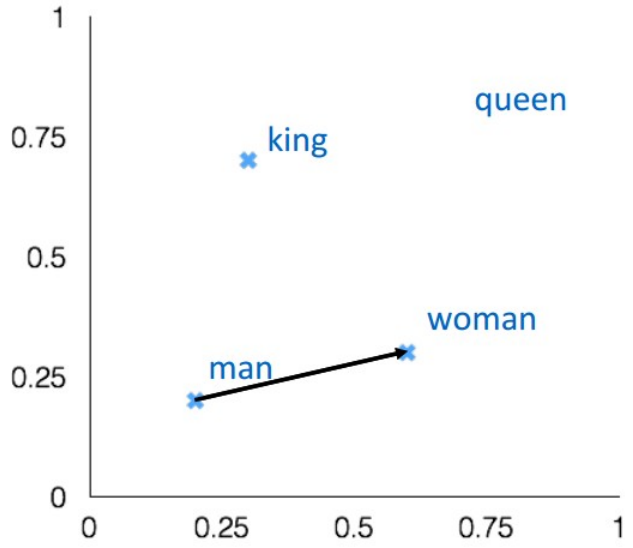
Test for linear relationships, examined by Mikolov et al. (2014)

a:b :: c:?

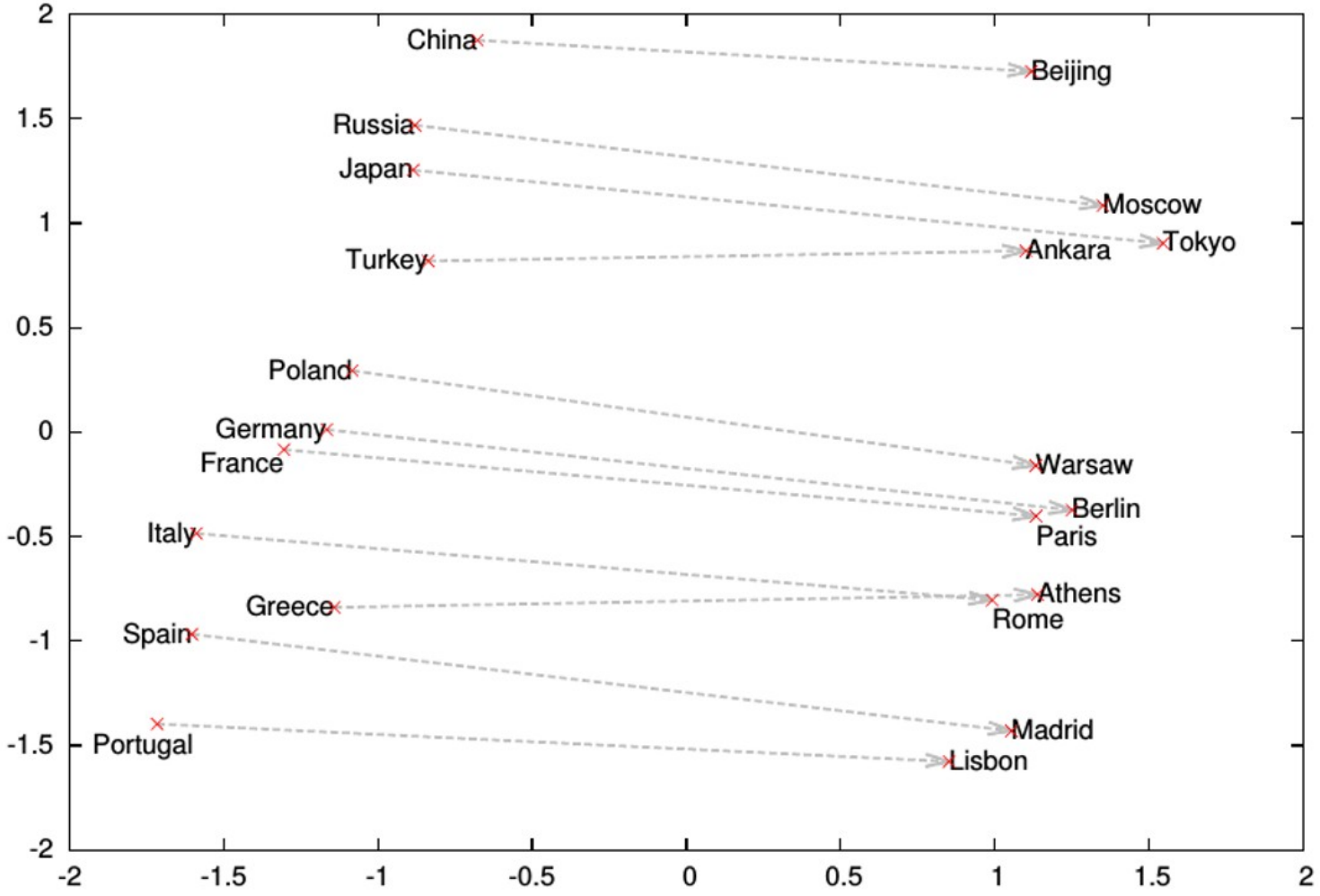
$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

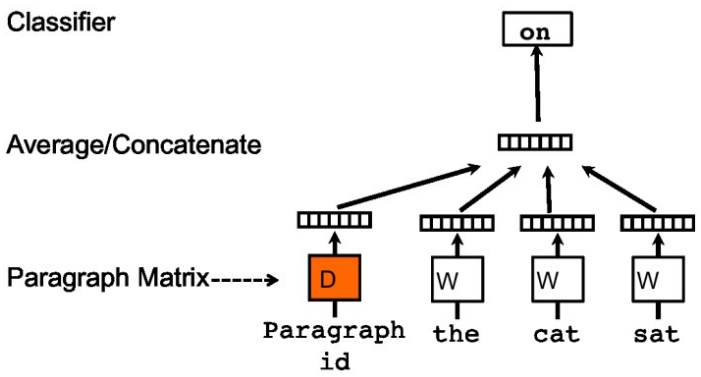


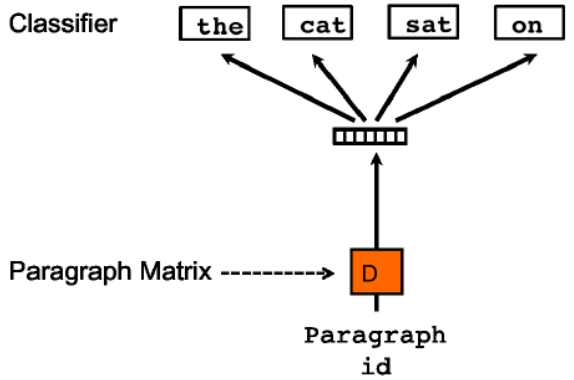
Word analogies



Represent the meaning of **sentence/text**

- Simple approach: take avg of the word2vecs of its words
- Another approach: Paragraph vector (2014, Quoc Le, Mikolov)
  - Extend word2vec to text level
  - Also two models: add paragraph vector as the input





Applications

- Search, e.g., query expansion
- Sentiment analysis
- Classification
- Clustering

Resources

- Stanford CS224d: Deep Learning for NLP
  - http://cs224d.stanford.edu/index.html
  - The best
- “word2vec Parameter Learning Explained”, Xin Rong
  - https://ronxin.github.io/wevi/
- Word2Vec Tutorial - The Skip-Gram Model
  - http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
- Improvements and pre-trained models for word2vec:
  - https://nlp.stanford.edu/projects/glove/
  - https://fasttext.cc/ (by Facebook)