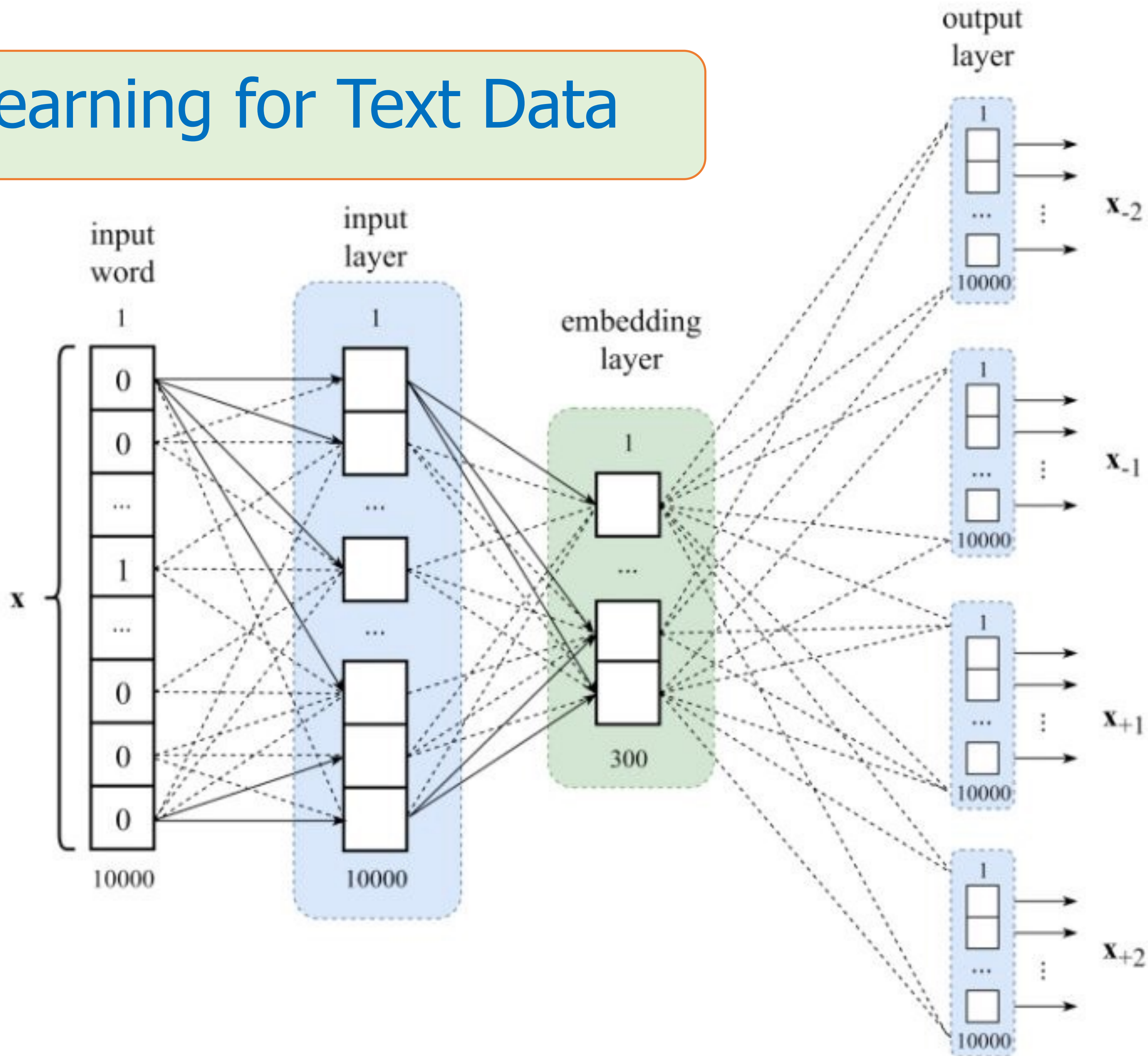
Deep Learning for Text Data



Agenda

- Word Vectors
- Word Embedding
- Word2vec
  - Continuous BoW
  - Skip-Gram
- Concluding Remarks

Word Vectors:
Word Similarity & Relatedness

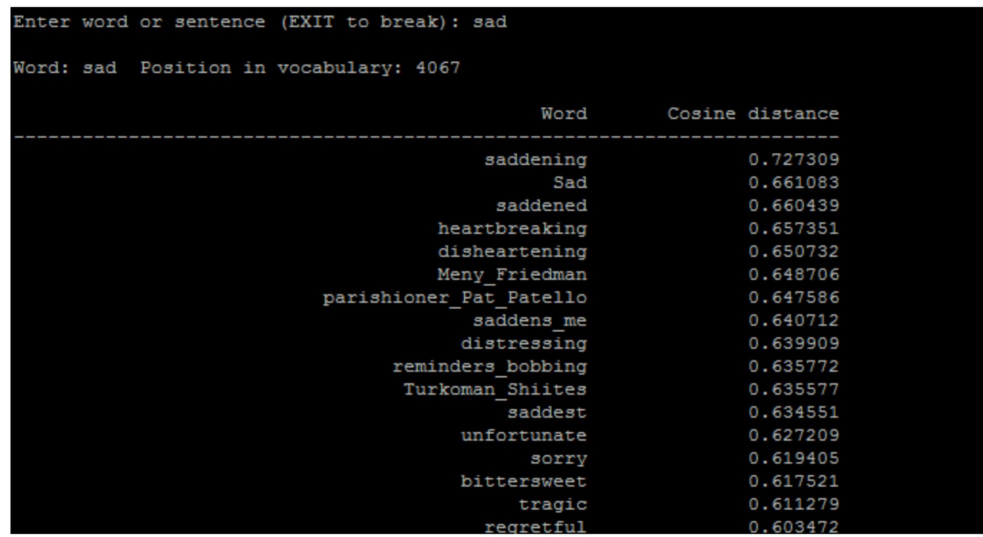
- **Representing words as vectors** allows easy computation of similarity
  - Measure the **semantic similarity** between words
  - How similar is **pizza** to **pasta**?
  - How related is **pizza** to **Italy**?

- **As features** for various supervised NLP tasks such as document classification, named entity recognition, and sentiment analysis

Application of Word Vectors: Sentiment Analysis

Classic Methods : Random Forests, Naive Bayes, SVM

- Classifying sentences as positive and negative
- Building sentiment lexicons using seed sentiment sets
- No need for classifiers, we can just use cosine distances to compare unseen reviews to known reviews.



Enter word or sentence (EXIT to break): sad

Word: sad Position in vocabulary: 4067

Word	Cosine distance
saddening	0.727309
Sad	0.661083
saddened	0.660439
heartbreaking	0.657351
disheartening	0.650732
Meny_Friedman	0.648706
parishioner_Pat_Patello	0.647586
saddens_me	0.640712
distressing	0.639909
reminders_bobbing	0.635772
Turkoman_Shiites	0.635577
saddest	0.634551
unfortunate	0.627209
sorry	0.619405
bittersweet	0.617521
tragic	0.611279
regretful	0.603472

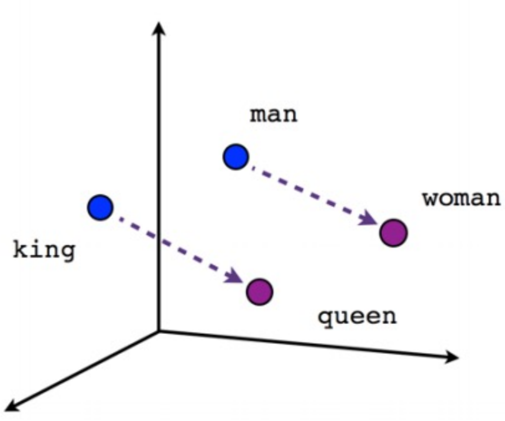
Word Representations

Traditional Method – Bag of Words (BoW) Model	Deep Method – **Word Embeddings**
• Uses one hot encoding • Each word in the vocabulary is represented by one bit position in a HUGE vector. • For example, if we have a vocabulary of 10000 words, and "Hello" is the 4^{th} word in the dictionary, it would be represented by: 0 0 0 1 0 0 . . . . 0 0 0 0 • Context information is not utilized	• Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300) • Unsupervised, built just by reading huge corpus • For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02] • Dimensions are basically projections along different axes, more of a mathematical concept.

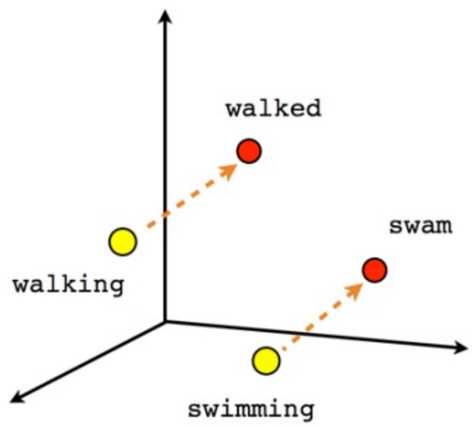
The Power of Word Vectors

- They provide a fresh perspective to ***ALL*** problems in NLP, and not just solve one problem.
- Technological Improvement
  - Rise of deep learning since 2006 (Big Data + GPUs + Work done by Andrew Ng, Yoshua Bengio, Yann Lecun and Geoff Hinton)
  - Application of Deep Learning to NLP – led by Yoshua Bengio, Christopher Manning, Richard Socher, Tomas Mikalov
- The need for unsupervised learning . (Supervised learning tends to be excessively dependant on hand-labelled data and often does not scale)

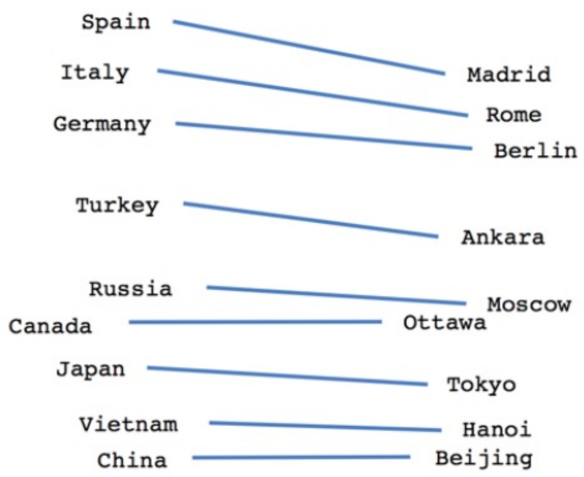
Examples



Male-Female



Verb tense



Country-Capital

vector[Queen] = vector[King] - vector[Man] + vector[Woman]

Word Embedding

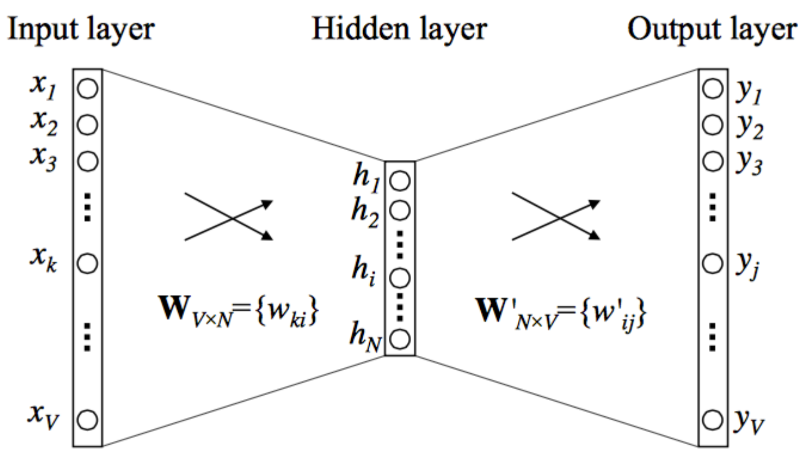
- **Idea:** learn an embedding from words into vectors
- A very famous method (from Google) to build lower-dimension vector representations for words based on their context
- Need to have a function $\mathcal{E}mb(\text{word})$ that returns a vector encoding that word.

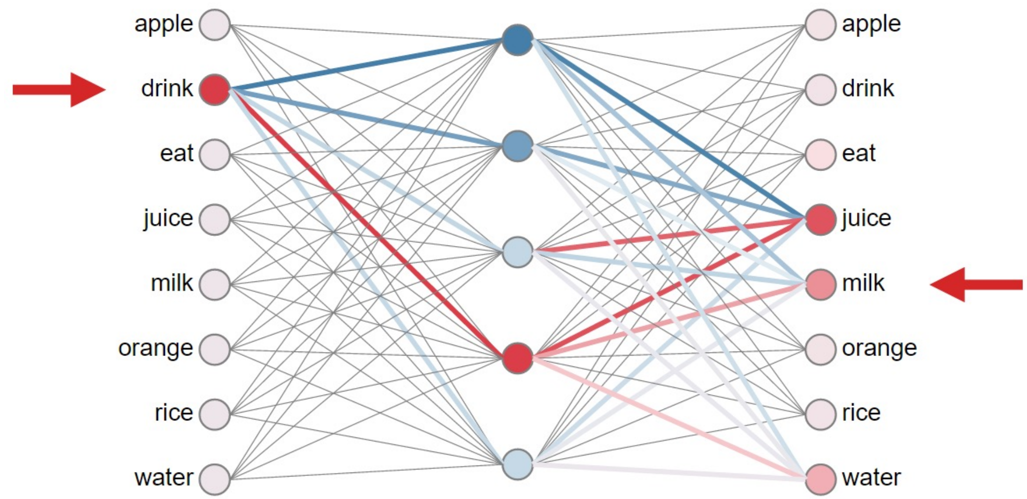
Word embeddings: questions

- How big should the embedding space be?
  - Trade-offs like any other machine learning problem – greater capacity versus efficiency and overfitting.
  - E.g. how many hidden nodes do we need for a MLP application?

- How do we find the embedding function $\mathcal{E}mb(\text{word})$?
  - Often as part of a prediction or classification task involving neighboring words.

Intuitive Idea





1. eat|apple
2. eat|orange
3. eat|rice
4. drink|juice
5. drink|milk
6. drink|water
7. orange|juice
8. apple|juice
9. rice|milk
10. milk|drink
11. water|drink
12. juice|drink

<u>Concept :</u>
1. Milk and Juice are drinks
2. Apple, Orange and Rice can be eaten
3. Apple and Orange are also juices
4. Rice milk is a actually a type of milk!

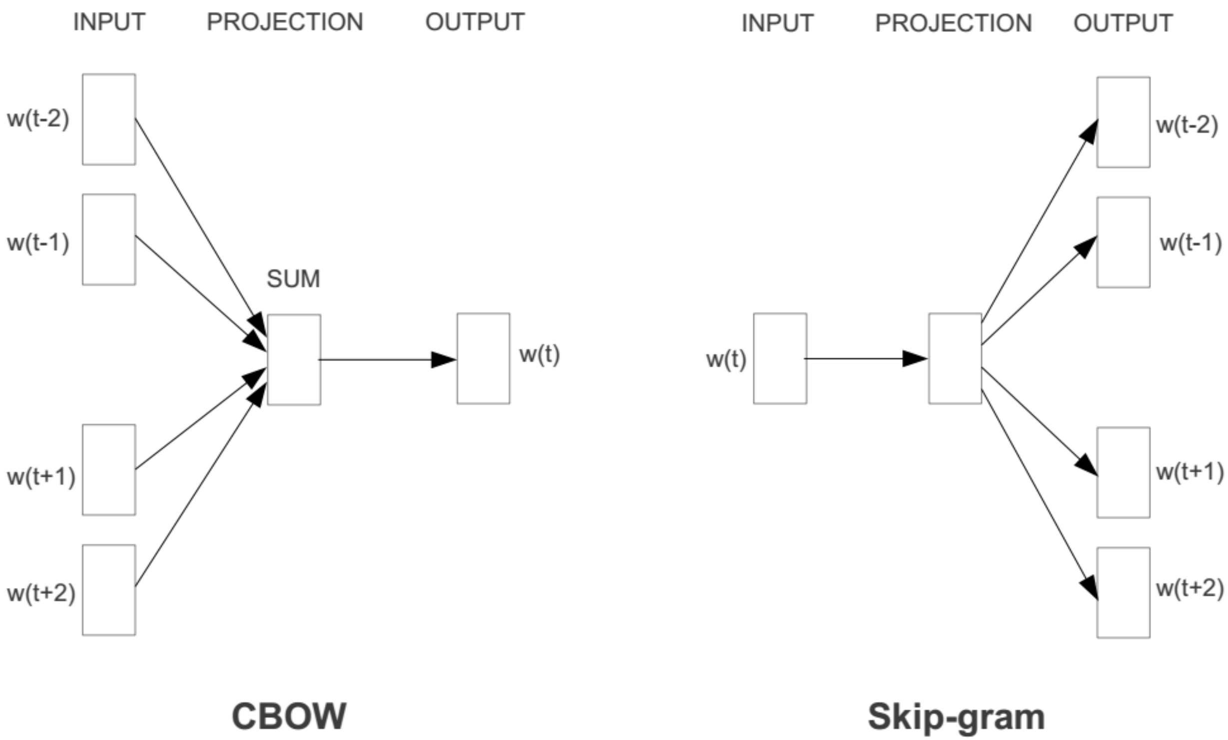
Word Embedding Visualization
http://ronxin.github.io/wevi/

word2vec:
An 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

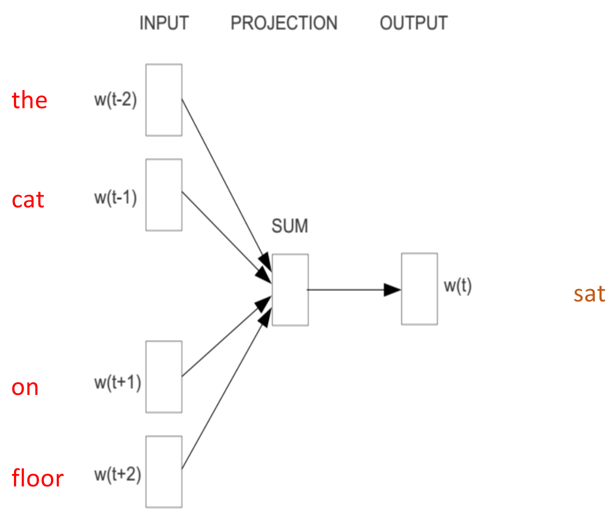
word2vec

- Involves 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.

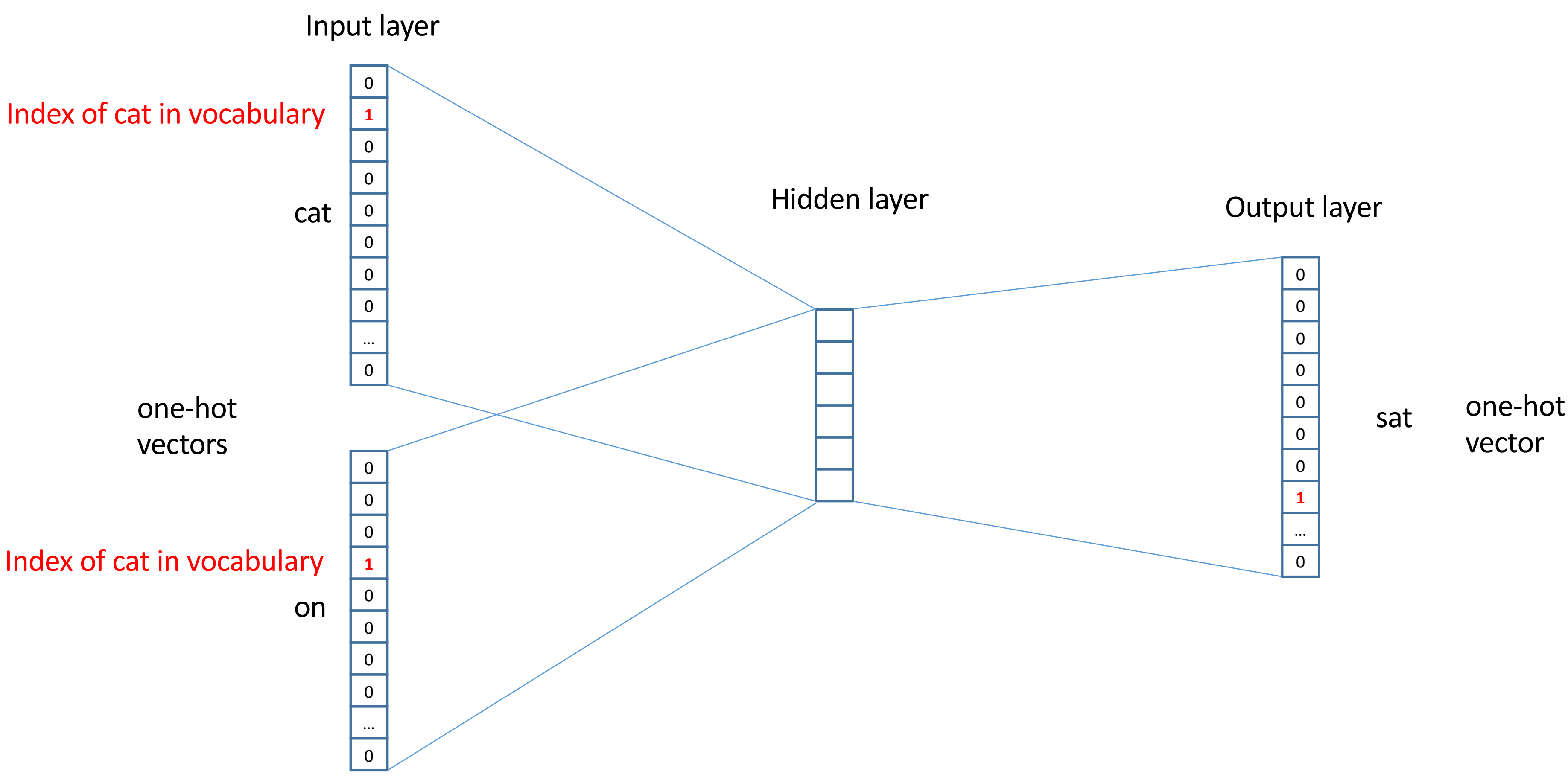


word2vec:
Continuous Bag of Word (CBoW) Module

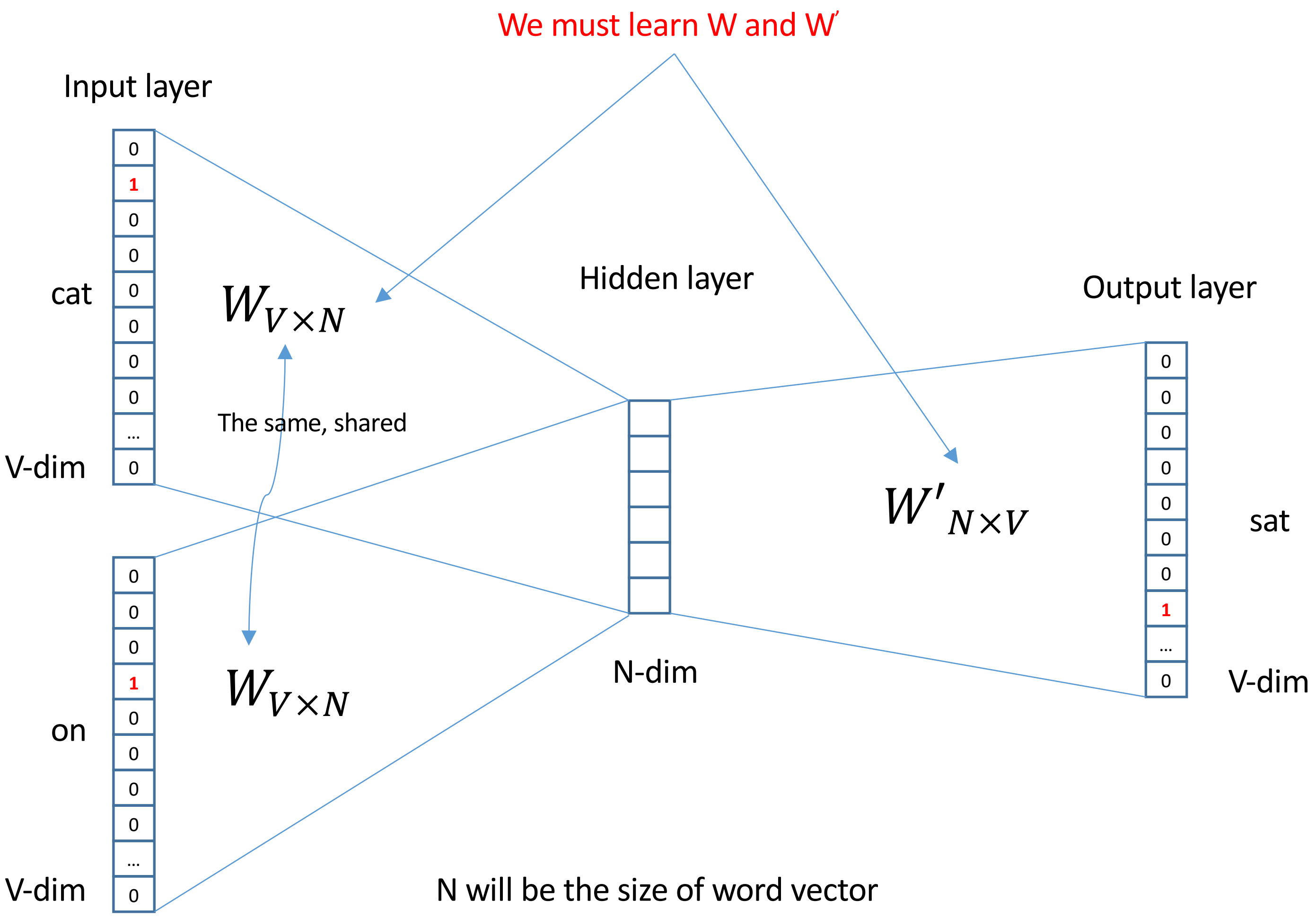
- Bag of words (BoW)
  - Get rid of word order (c.f. tfidf). Used in discrete case using counts of words that appear.
- CBoW
  - Takes vector embeddings of n words before target and n words after and adds them (as vectors).
  - Also removes word order, but the vector sum is meaningful enough to deduce missing word.
- E.g. “The cat sat on floor”
  - Window size = 2



word2vec:
Continuous Bag of Word (CBoW) Module (cont.)

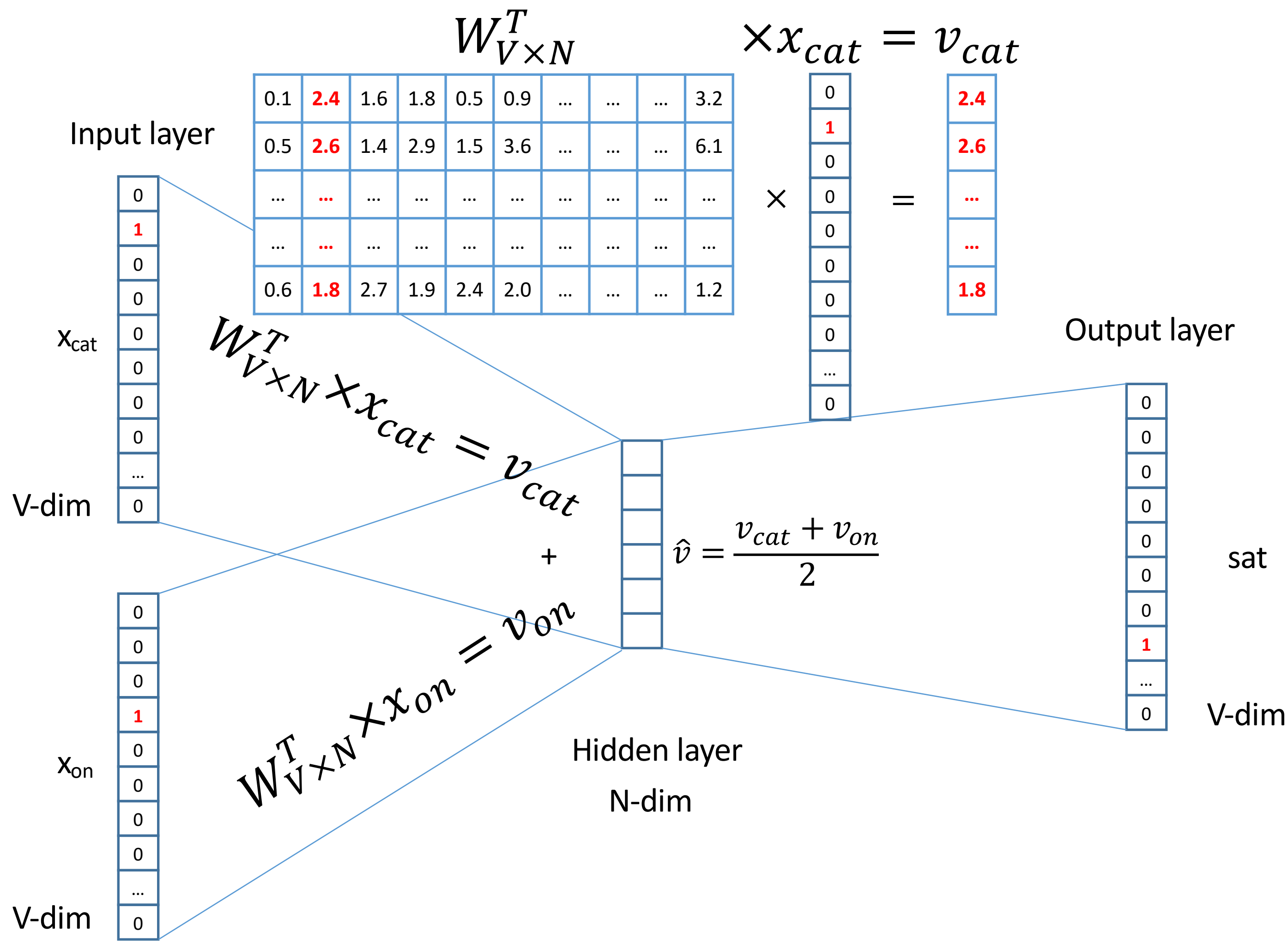


word2vec:
Continuous Bag of Word (CBoW) Module (cont.)

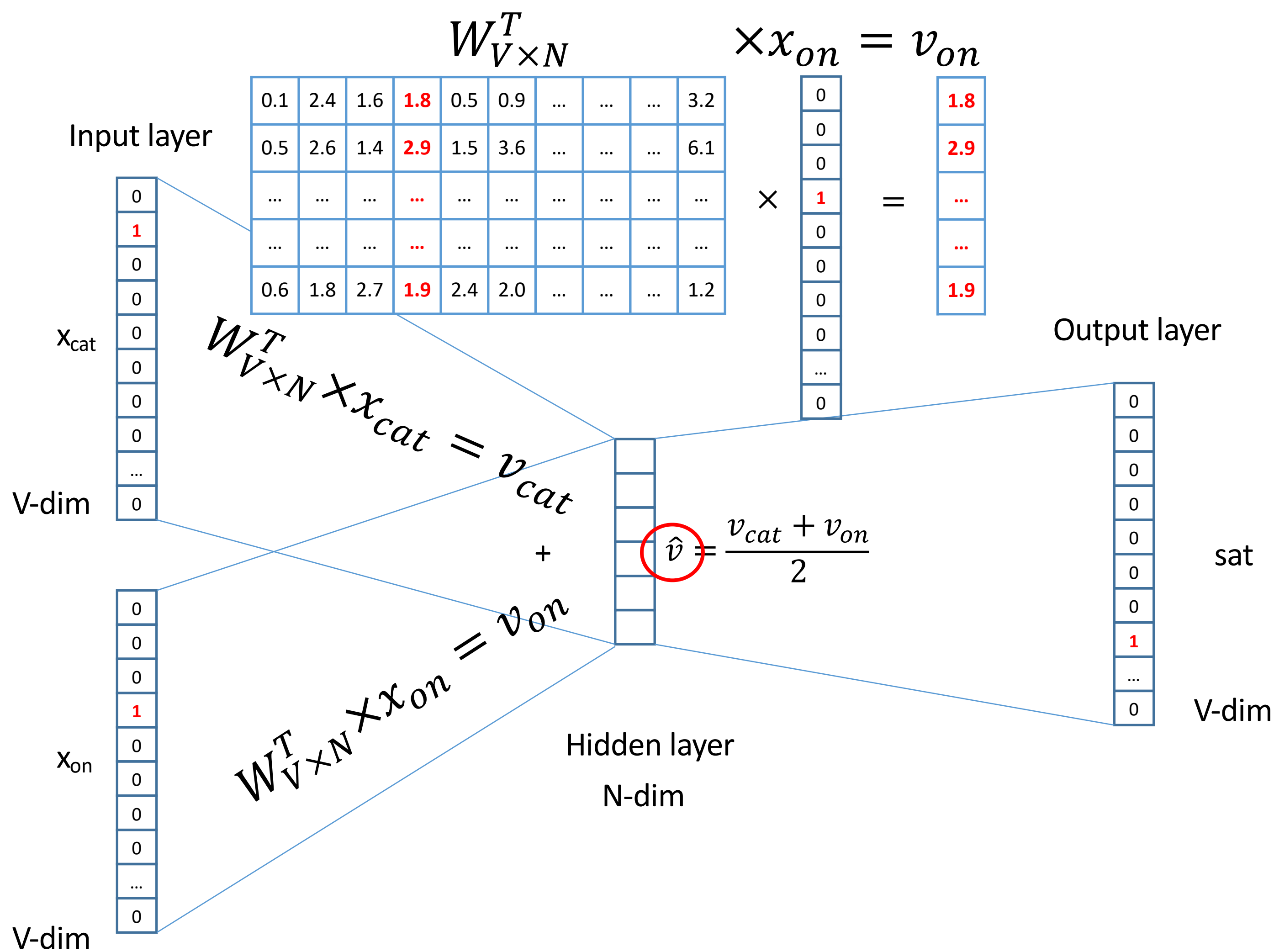


N will be the size of word vector

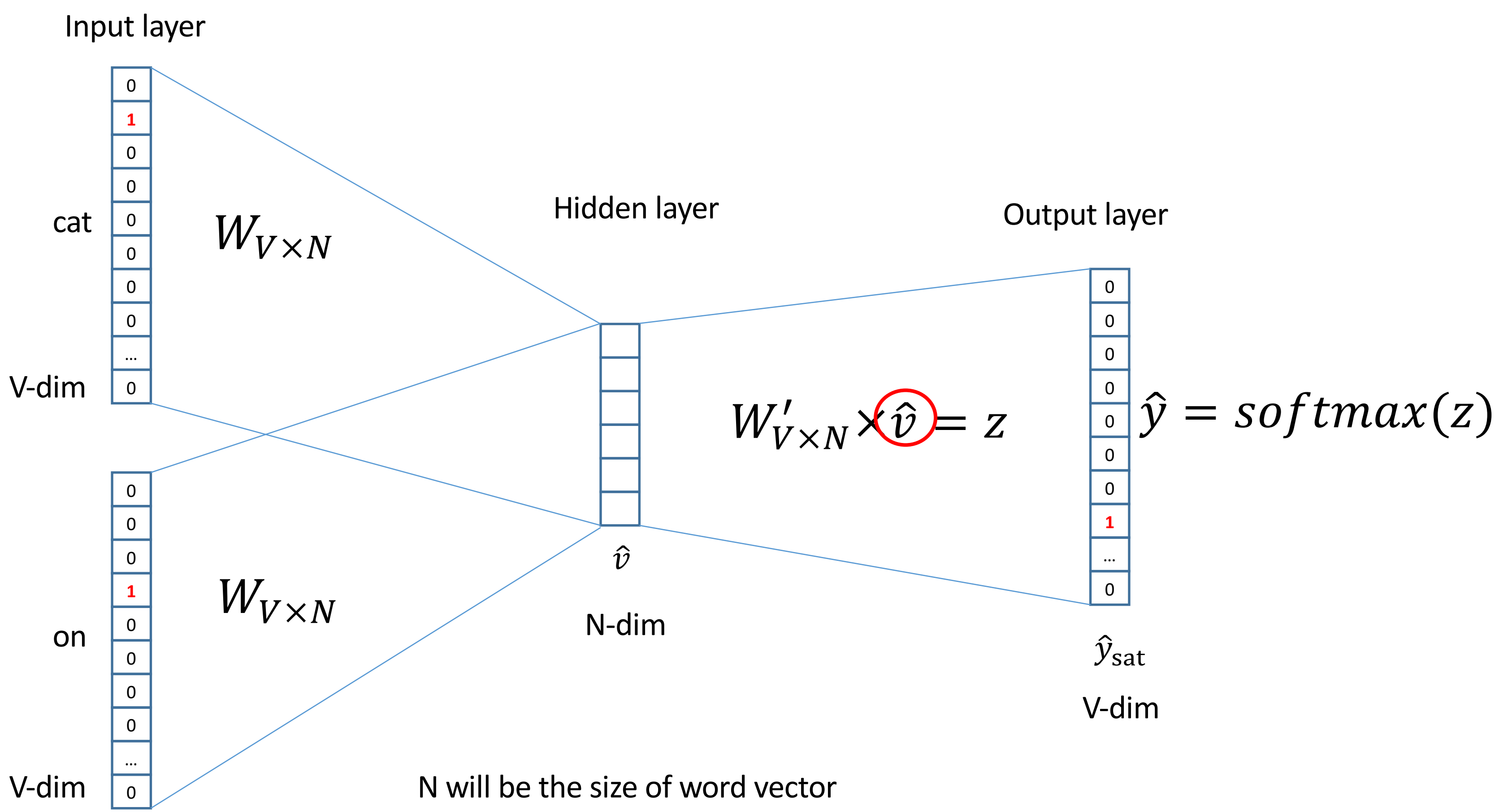
word2vec:
Continuous Bag of Word (CBoW) Module (cont.)



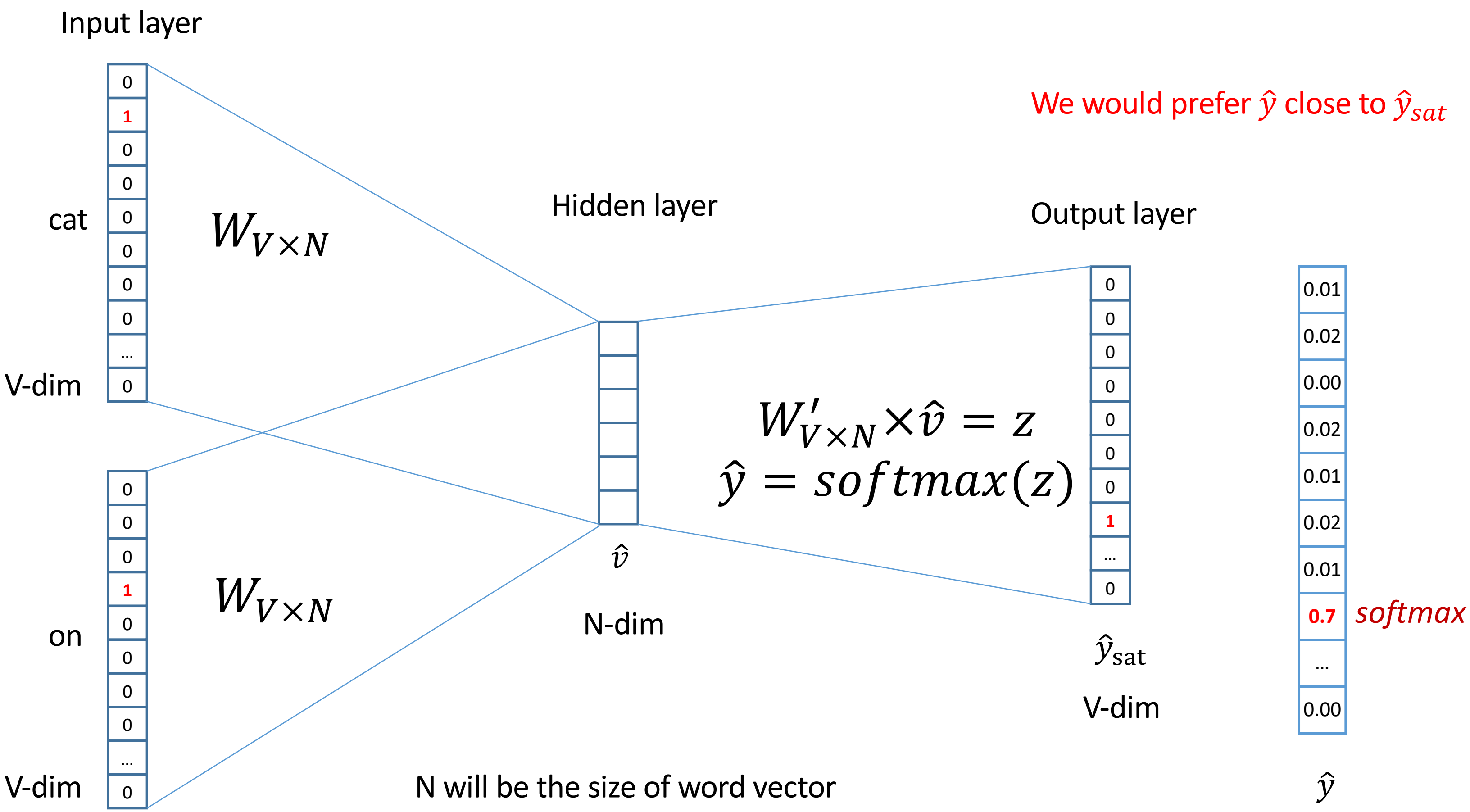
word2vec: Continuous Bag of Word (CBoW) Module (cont.)



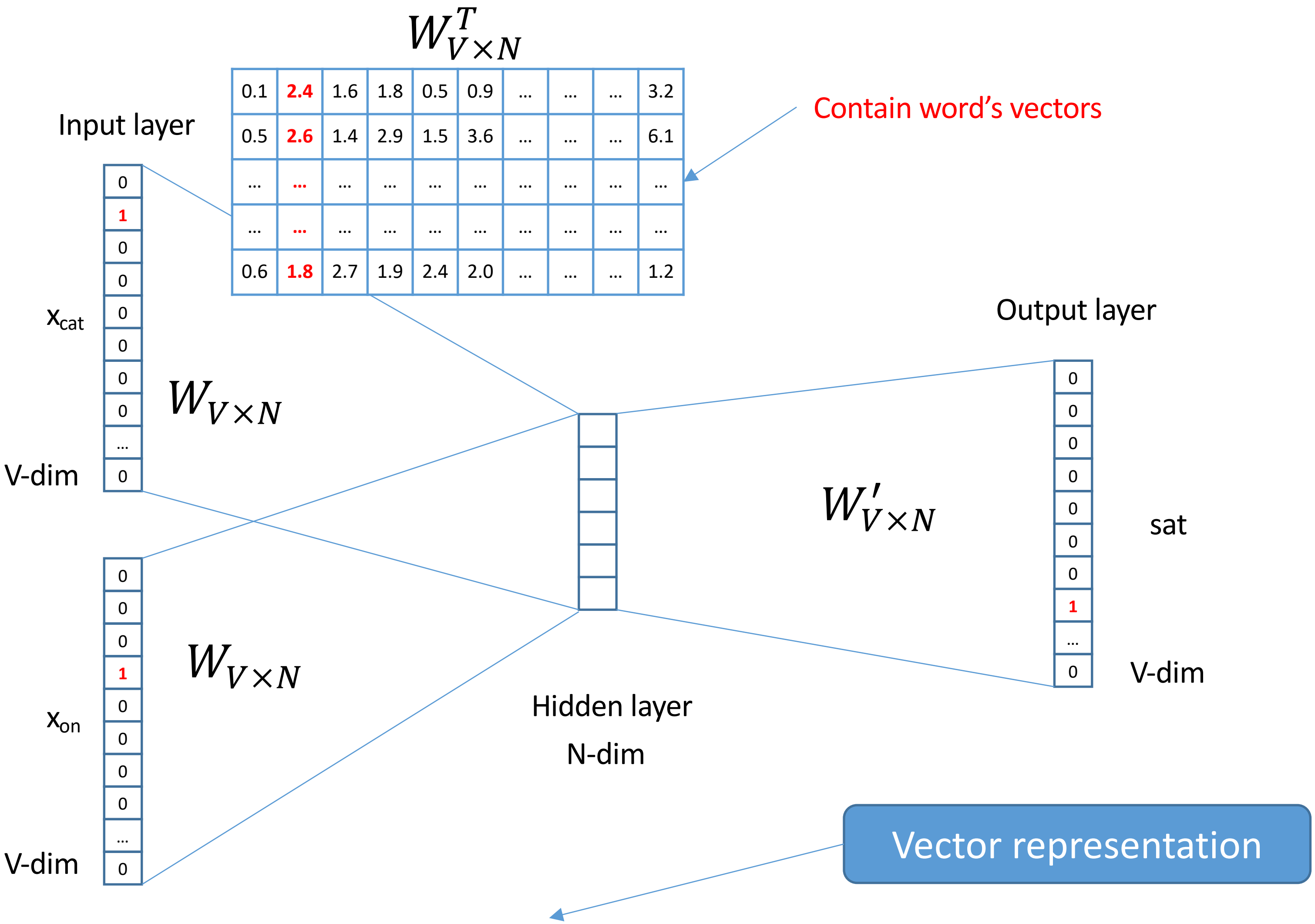
word2vec:
Continuous Bag of Word (CBoW) Module (cont.)



word2vec: Continuous Bag of Word (CBoW) Module (cont.)



word2vec: Continuous Bag of Word (CBoW) Module (cont.)



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

Some interesting results

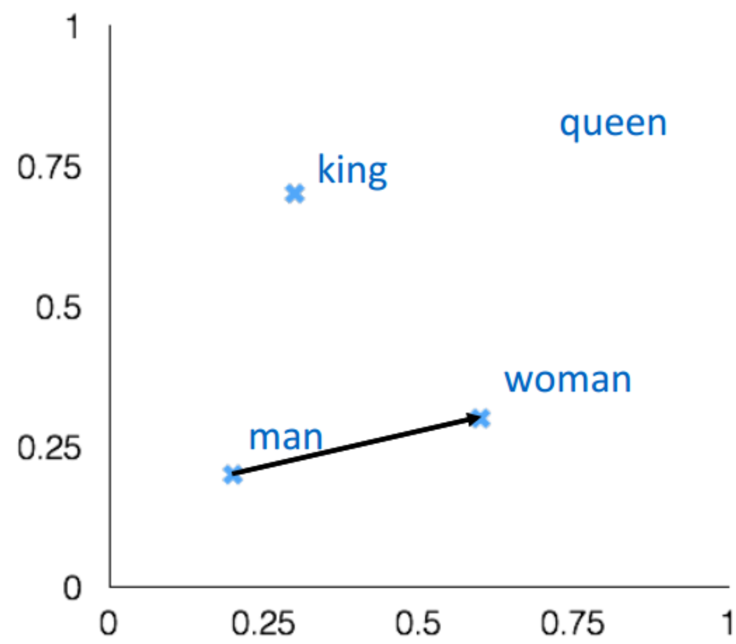
Word Analogies

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

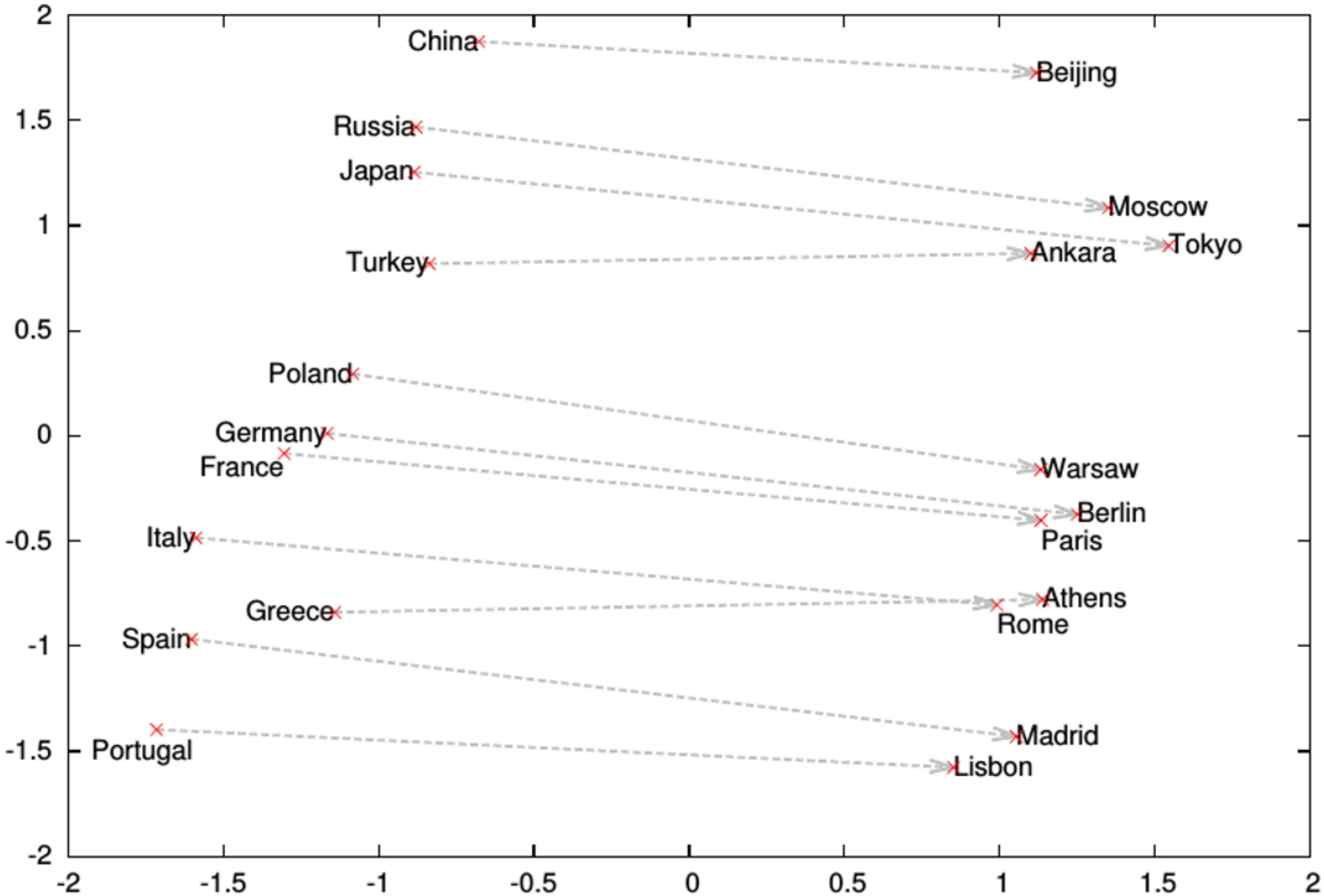
a:b :: c:? → $$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

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

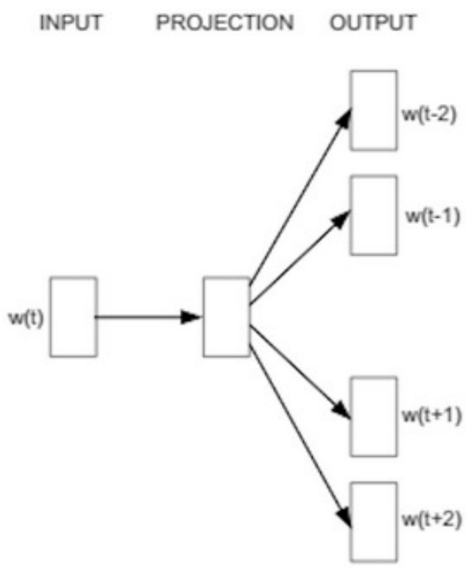


Word analogies



word2vec: Skip-gram

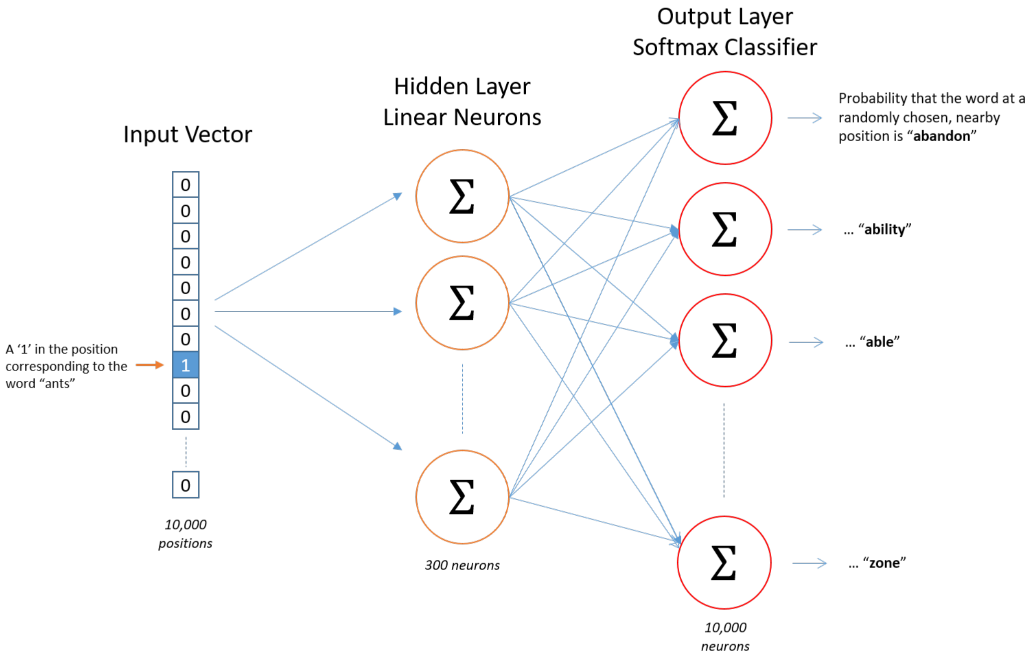
- Skip-gram – alternative to CBOW
  - Start with a single word embedding and try to predict the surrounding words.
  - Much less well-defined (difficult) problem, but works better in practice (scales better).



Skip-gram

Skip-gram

- Map from center word to probability on surrounding words. One input/output unit below.
    - There is no activation function on the hidden layer neurons, but the output neurons use softmax.

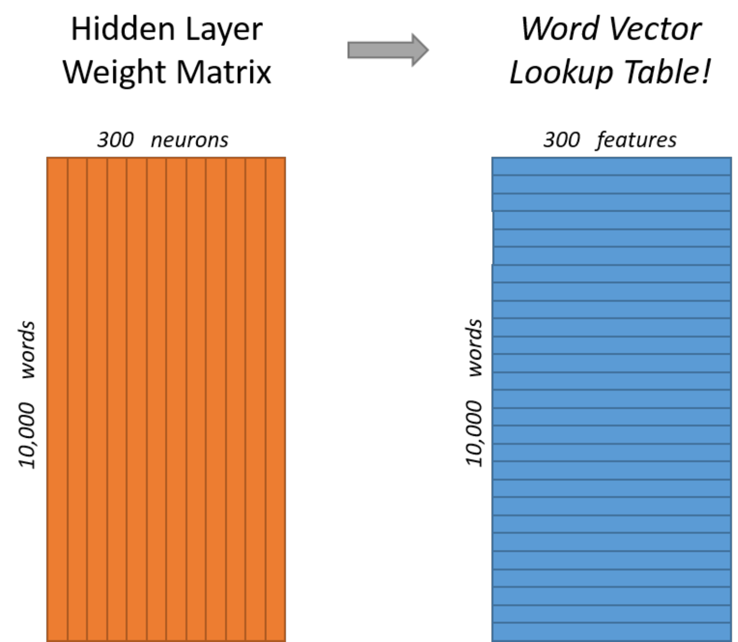


http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Skip-gram example

- Vocabulary of 10,000 words.
- Embedding vectors with 300 features.
- So the hidden layer is going to be represented by a weight matrix with 10,000 rows (multiply by vector on the left).

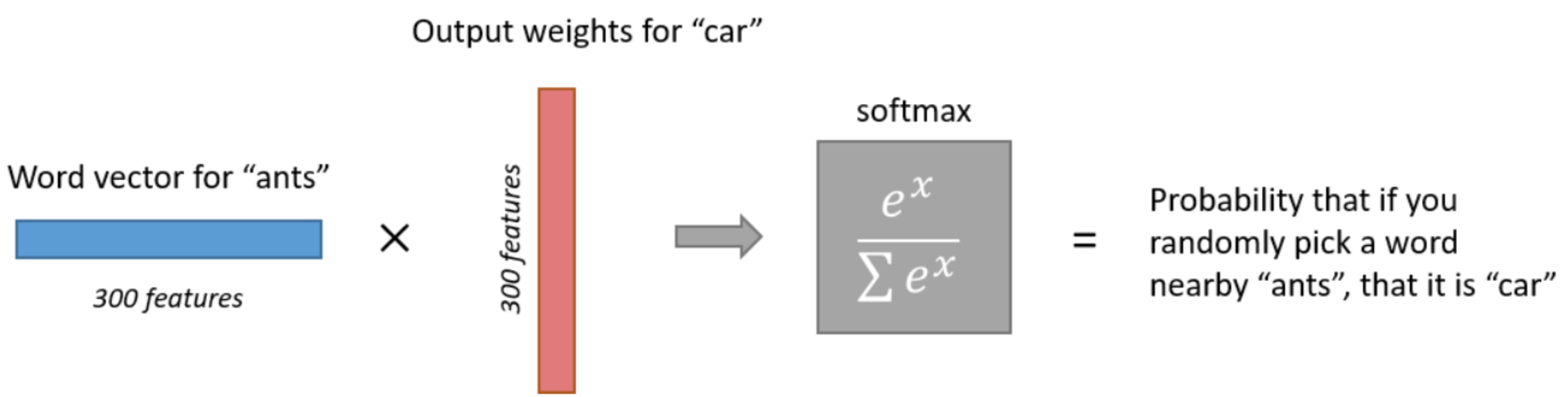
$$\begin{bmatrix} 0 & 0 & 0 & \boxed{1} & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$



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

The output layer of skip-gram

- The 1x300 word vector gets fed to the output layer which is a softmax regression classifier
- Here is an example:



Skip gram/CBOW intuition

- Similar "contexts" (that is, what words are likely to appear around them), lead to similar embeddings for two words.
- One way for the network to output similar context predictions for these two words is if *the word vectors are similar*. So, if two words have similar contexts, then the network is motivated to learn similar word vectors for these two words!

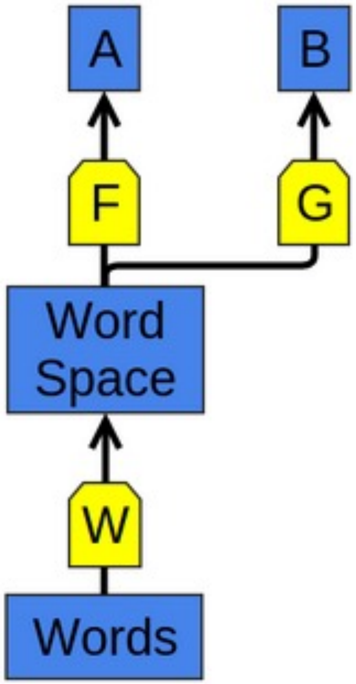
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

word2vec shortcomings

- **Problem:** 10,000 words and 300 dim embedding gives a large parameter space to learn. And 10K words is minimal for real applications.
- Slow to train, and need lots of data, particularly to learn uncommon words.
- Very vulnerable, and not a robust concept
- Non-uniform results
- Hard to understand and visualize

An important milestone

- The use of word representations… has become a key “secret sauce” for the success of many NLP systems in recent years, across tasks including named entity recognition, part-of-speech tagging, parsing, and semantic role labeling. ([Luong *et al.* (2013)](#))
- Learning a good representation on a task A and then using it on a task B is one of the major tricks in the Deep Learning toolbox.
  - Pretraining, transfer learning, and multi-task learning.
  - Can allow the representation to learn from more than one kind of data.



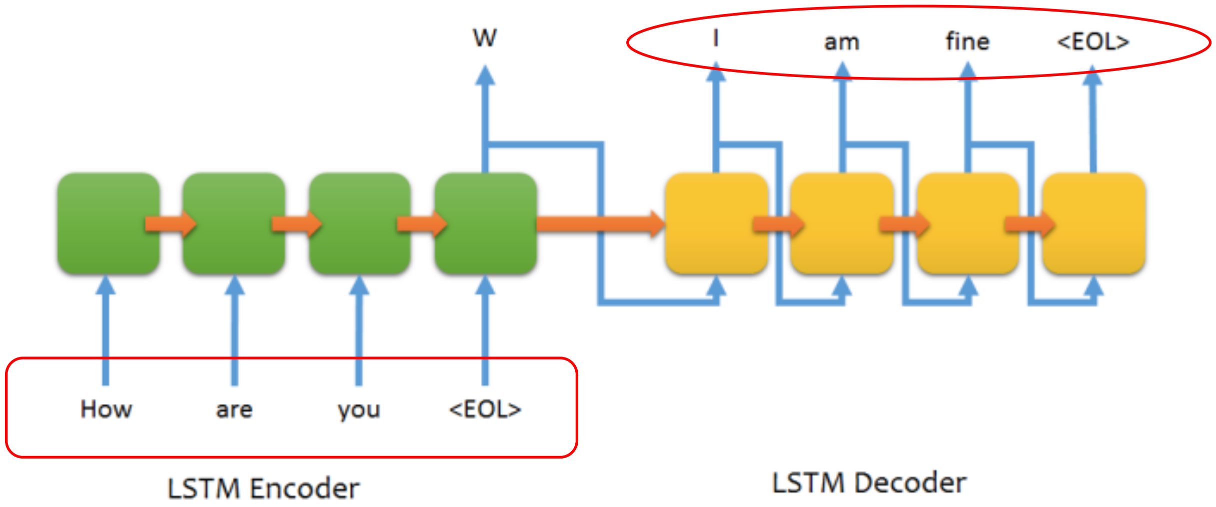
W and F learn to perform task A. Later, G can learn to perform B based on W.

http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

Leading to Chatting, Transformer and GPT

- Given “The cat sat on", predict the next word.
- Given “The cat sat on floor.”, predict the next sentence.

Encoder-Decoder LSTM (Long Short Term Memory) structure for chatting



Final Words

- Effective word representation is an important milestone of deep learning, leading to the state-of-the-art ChatGPT storm.
- Important concepts include embedding, similarity, relatedness, etc.
- Yet more important concepts like transfer learning and attention are waiting for you to further study.
- We have already seen the dramatic success of learning image/media representation and word representation. So, what’s next?

Acknowledgement

- Girish K, Texas A&M University
- Vagelis Hristidis, UCRiverside
- Greg Buzzard, Purdue University

- Stanford CS224d: Deep Learning for NLP
  - http://cs224d.stanford.edu/index.html
- “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/