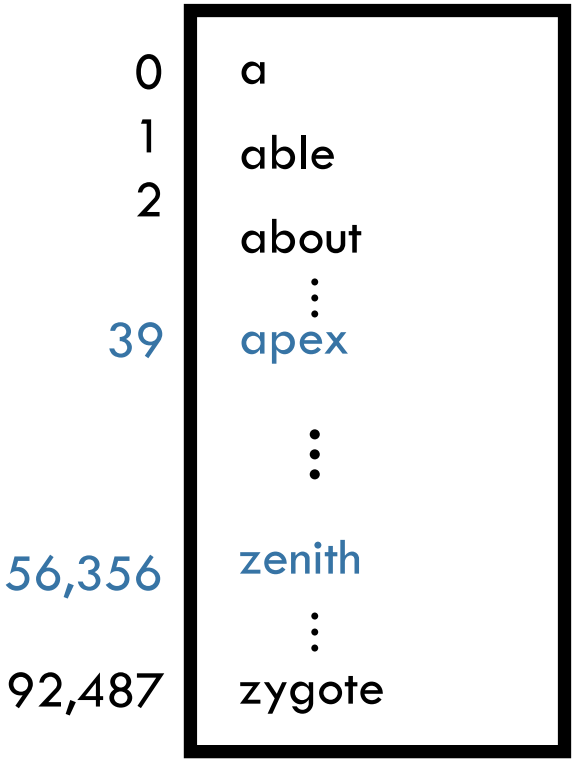
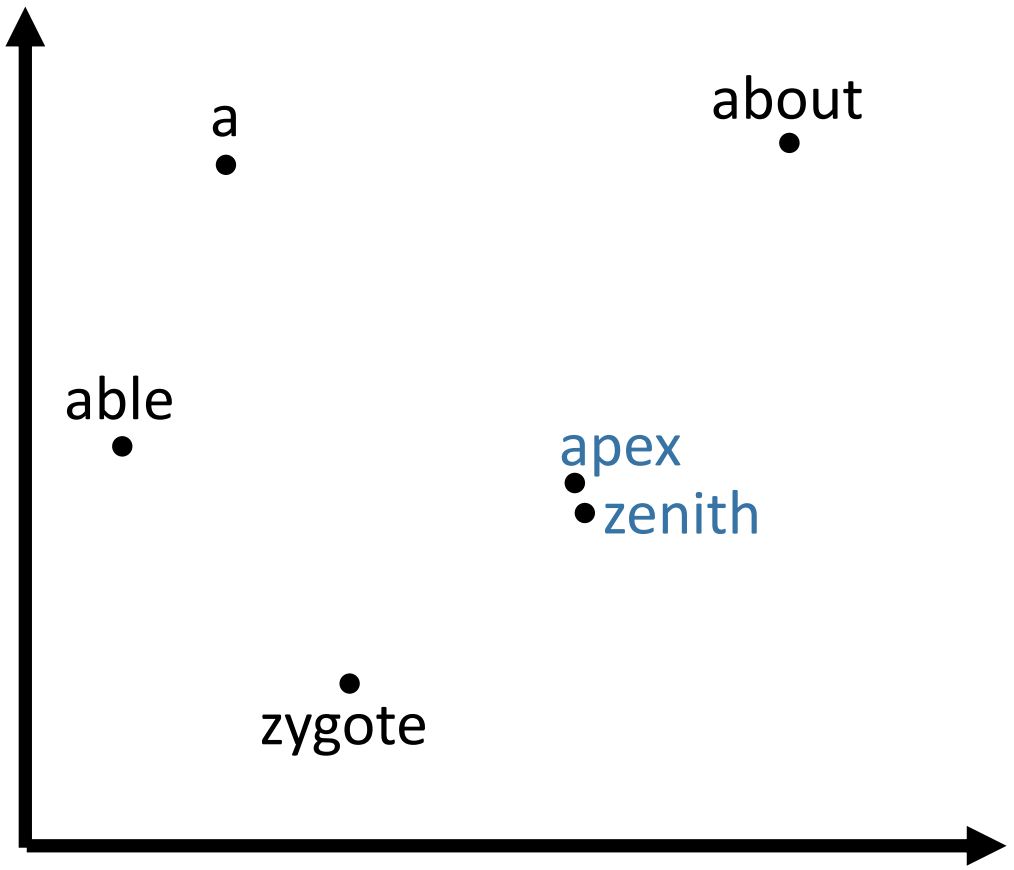
EE542 Lecture 23: Embeddings LLM Part 2

Young Cho

Department of Electrical Engineering

University of Southern California


0
a
1
able
2
about
39
apex
56,356
zenith
92,487
zygote



Word Embedding (e.g., word2Vec, GloVe)

Word embeddings

Text processing with current Machine Learning requires encoding into vectors.

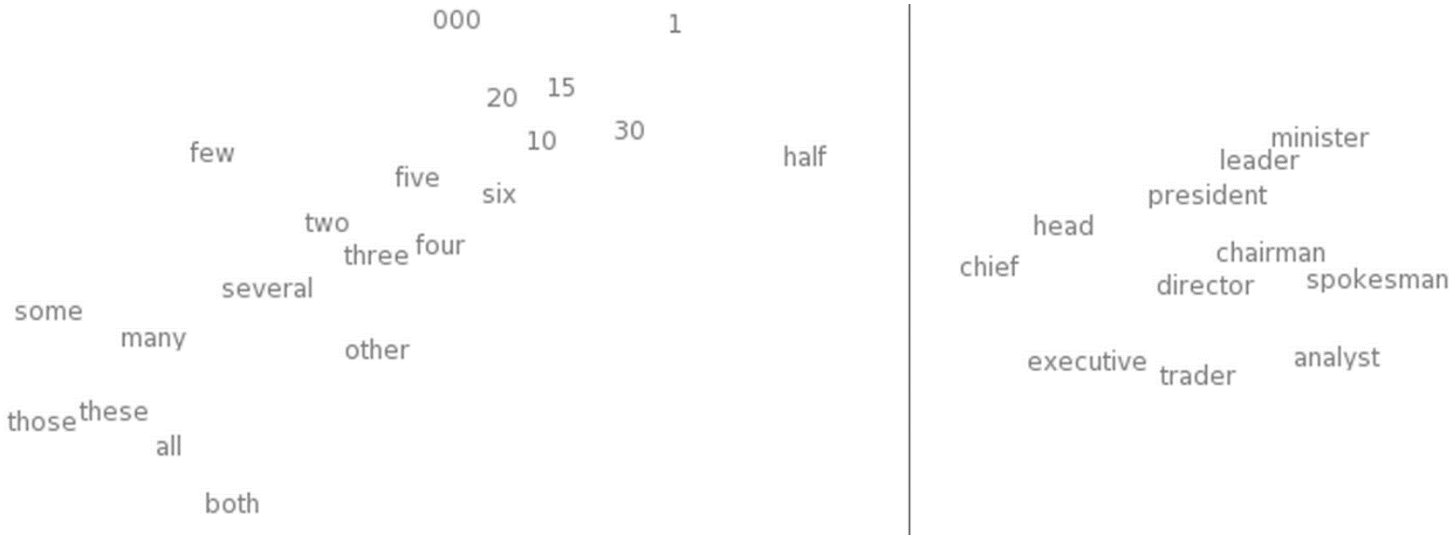
One-hot encoding:

- N words encoded by length N vectors.
- A word gets a vector with exactly one entry = 1, others 0.
- Very space inefficient, no natural language structure.

Bag of words:

- Collection of words (along with number of occurences).
- No word order, no natural language structure.

Similarity



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

Word embeddings: relationships

- Hope to preserve some language structure (relationships between words).

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

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

Word embeddings

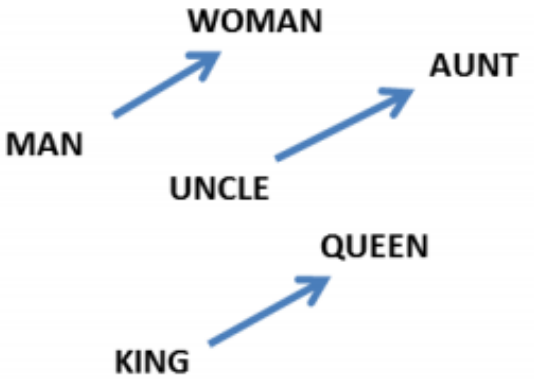
Need to have a function W(word) that returns a vector encoding that word.

Similarity of words corresponds to nearby vectors.

Relationships between words correspond to difference between vectors.

Properties

- Relationships between words correspond to difference between vectors.



$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“aunt”}) - W(\text{“uncle”})$$

$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“queen”}) - W(\text{“king”})$$

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

Determining Parameters

How big should the embedding space be?

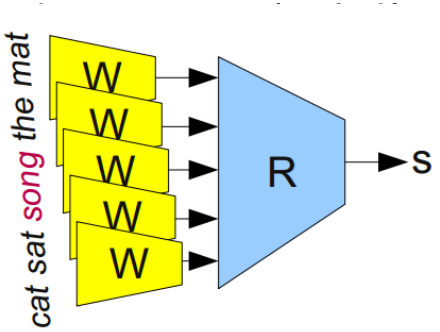
- Trade-offs like any other machine learning problem – greater capacity versus efficiency and overfitting.

How do we find W?

- Often as part of a prediction or classification task involving neighboring words.

Learning word embeddings

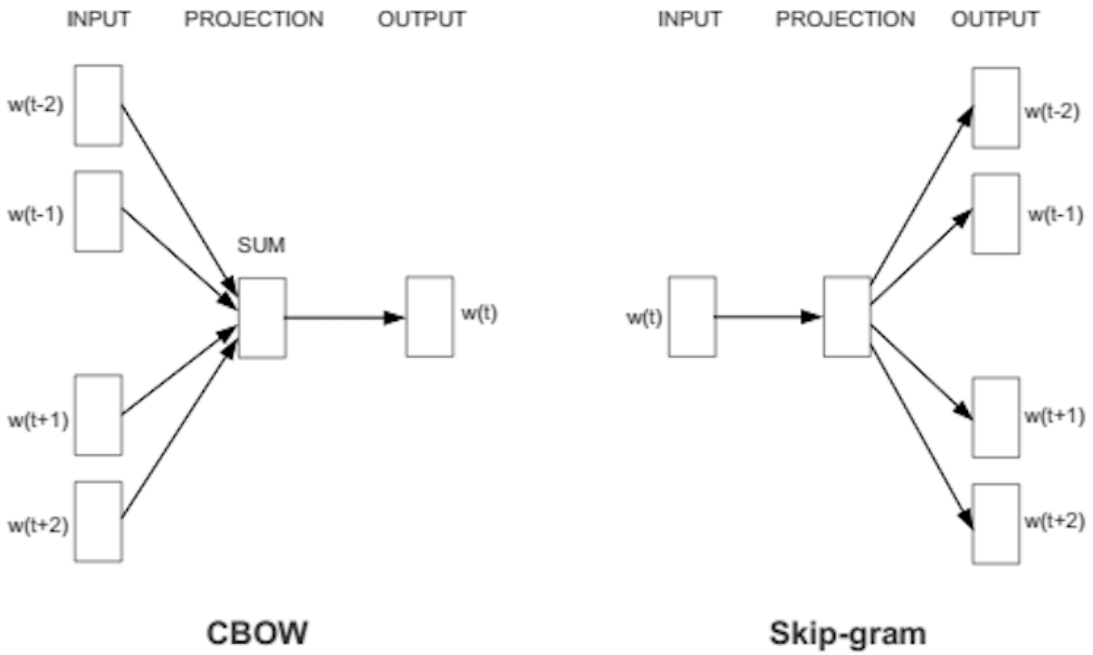
- First attempt: Bottou, “From Machine Learning to Machine Reasoning,” 2011.
  - Input data is sets of 5 words from a meaningful sentence. E.g., “one of the best places”.
  - Modify half of them by replacing middle word with a random word. “one of function best places”
  - W is a map (depending on parameters, Q) from words to 50 dim’l vectors. E.g., a look-up table or an RNN.
  - Feed 5 embeddings into a module R or ‘invalid’
  - Optimize over Q to predict better



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

Newer: Mikolov, Word2vec, 2013

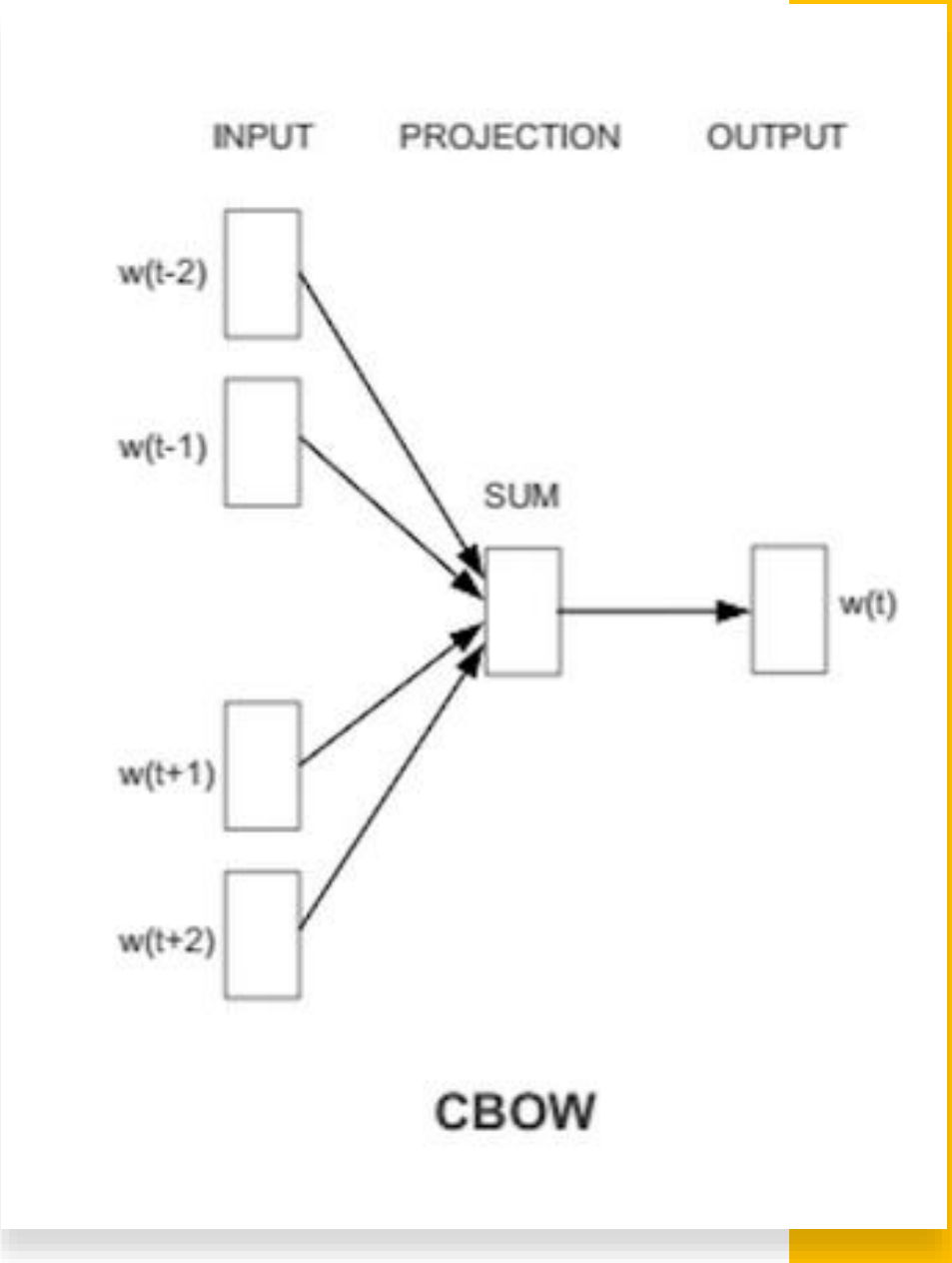
- Predict words using context
- Continuous Bag of Words and Skip-gram



https://skymind.ai/wiki/word2vec

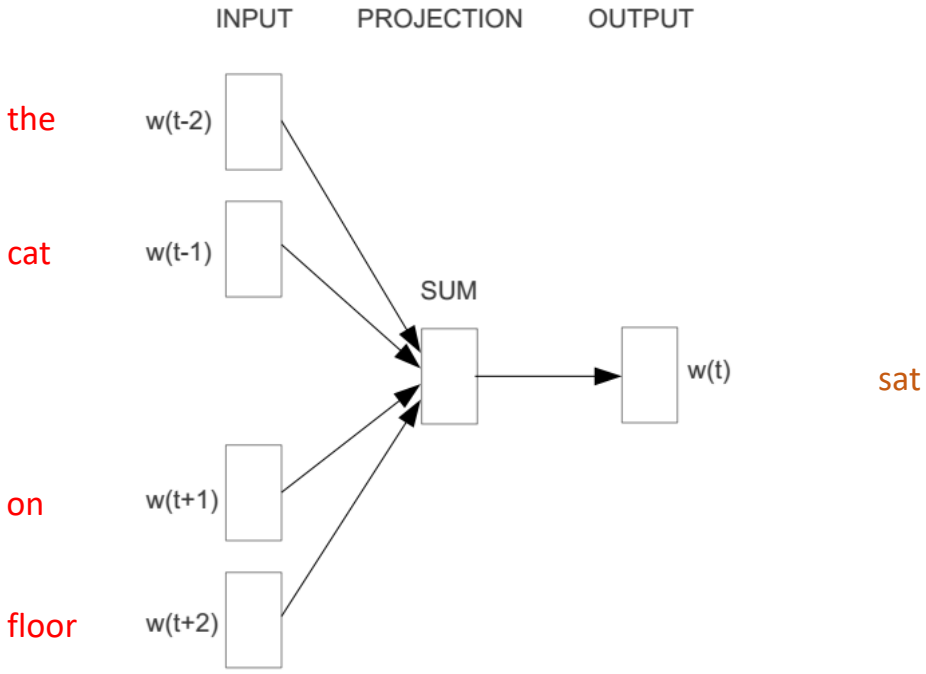
Continuous Bag of Words

- Bag of words (BOW)
  - Gets rid of word order. Used in discrete case using counts of words that appear.
- Continuous BOW
  - 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.

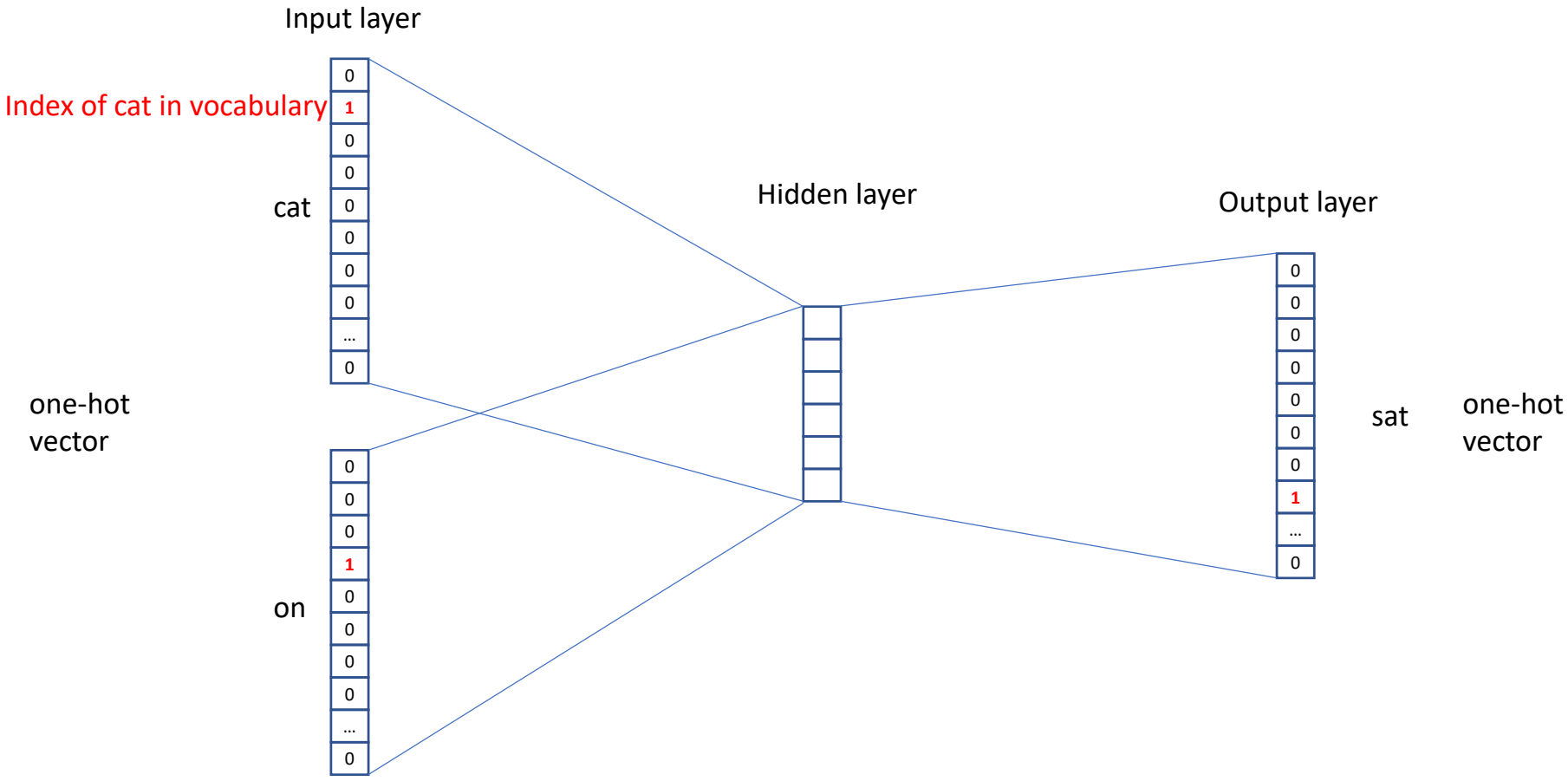


Continuous Bag of Word

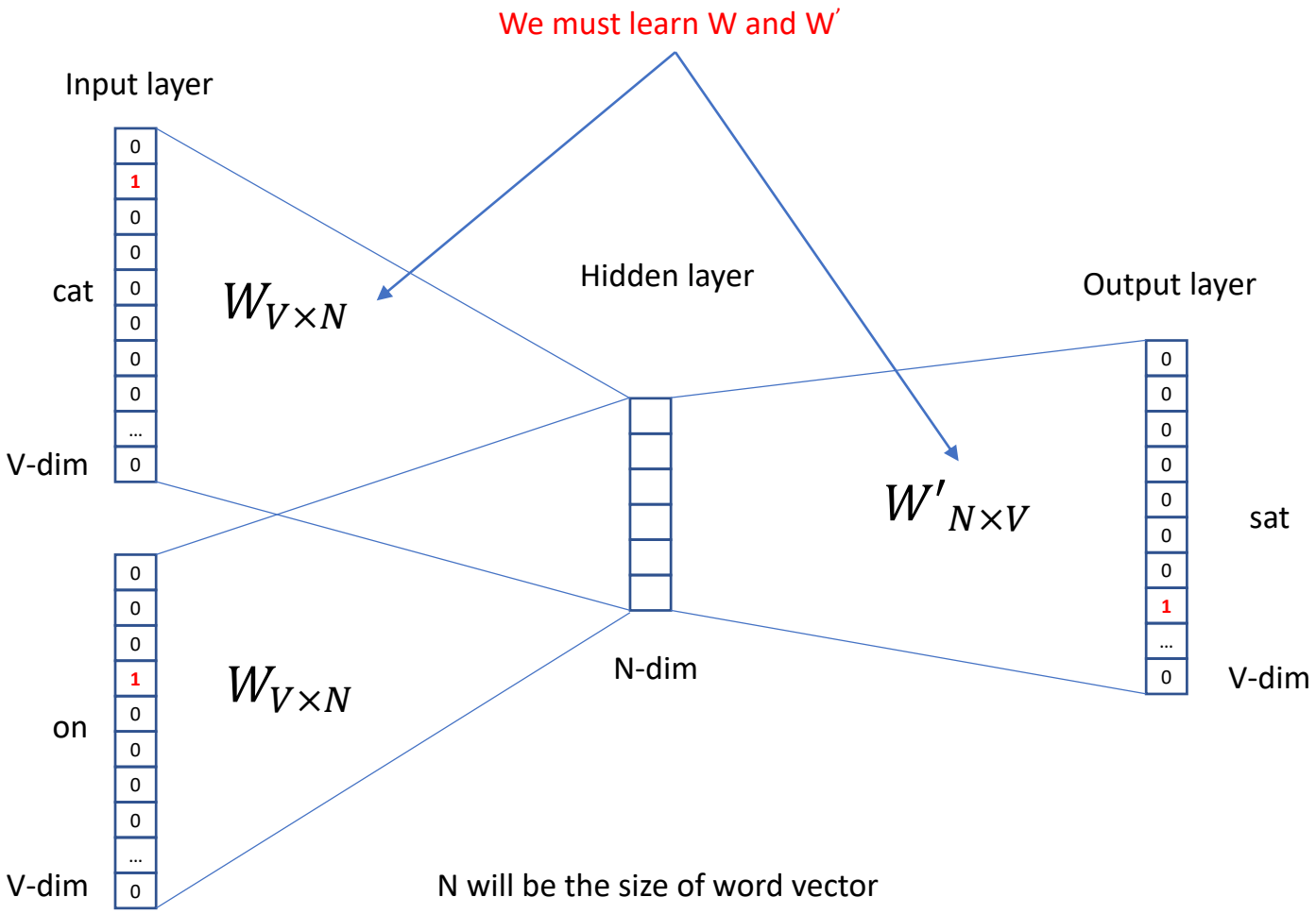
- E.g. “The cat sat on floor”
  - Window size = 2



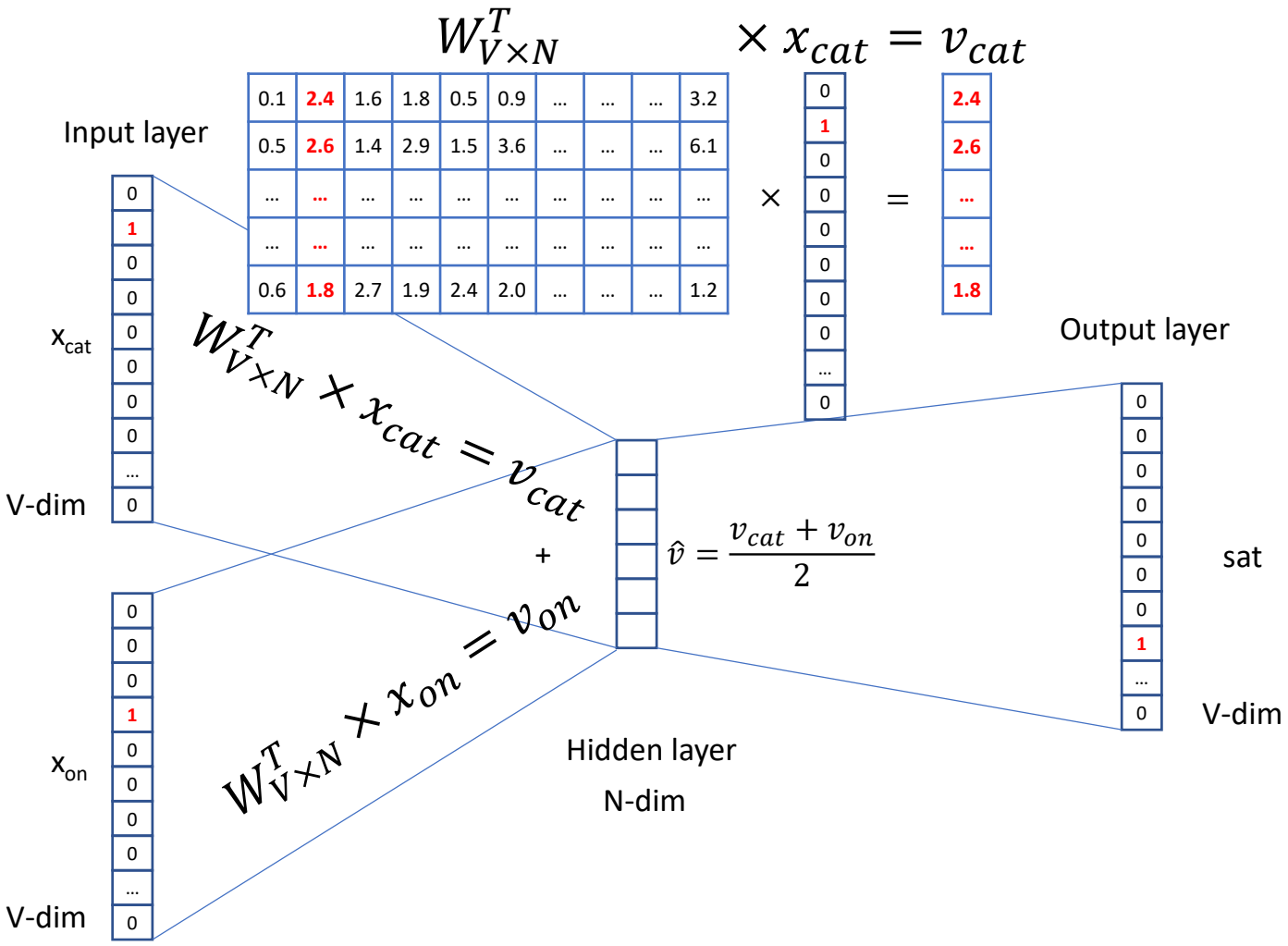
www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



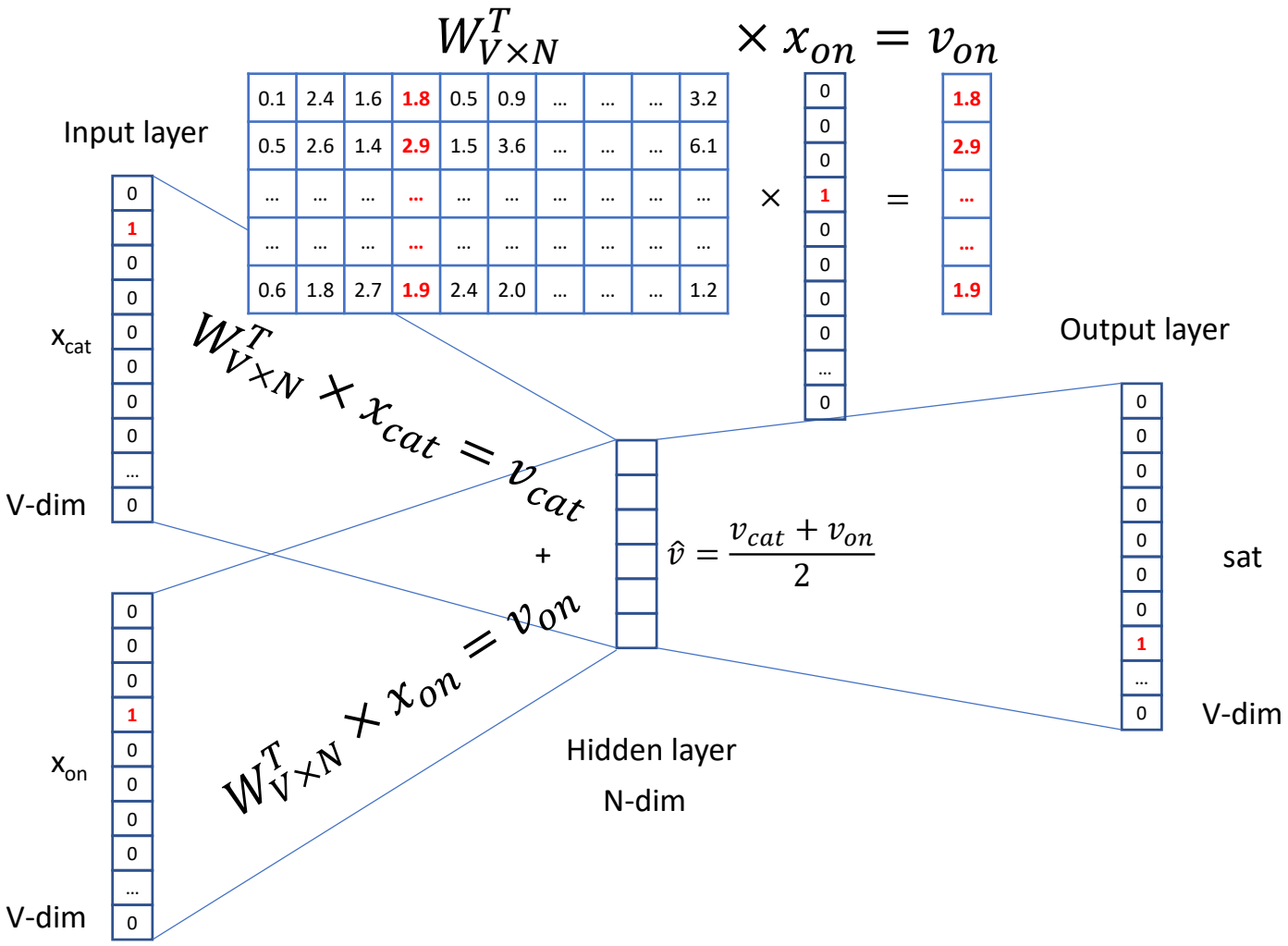
www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



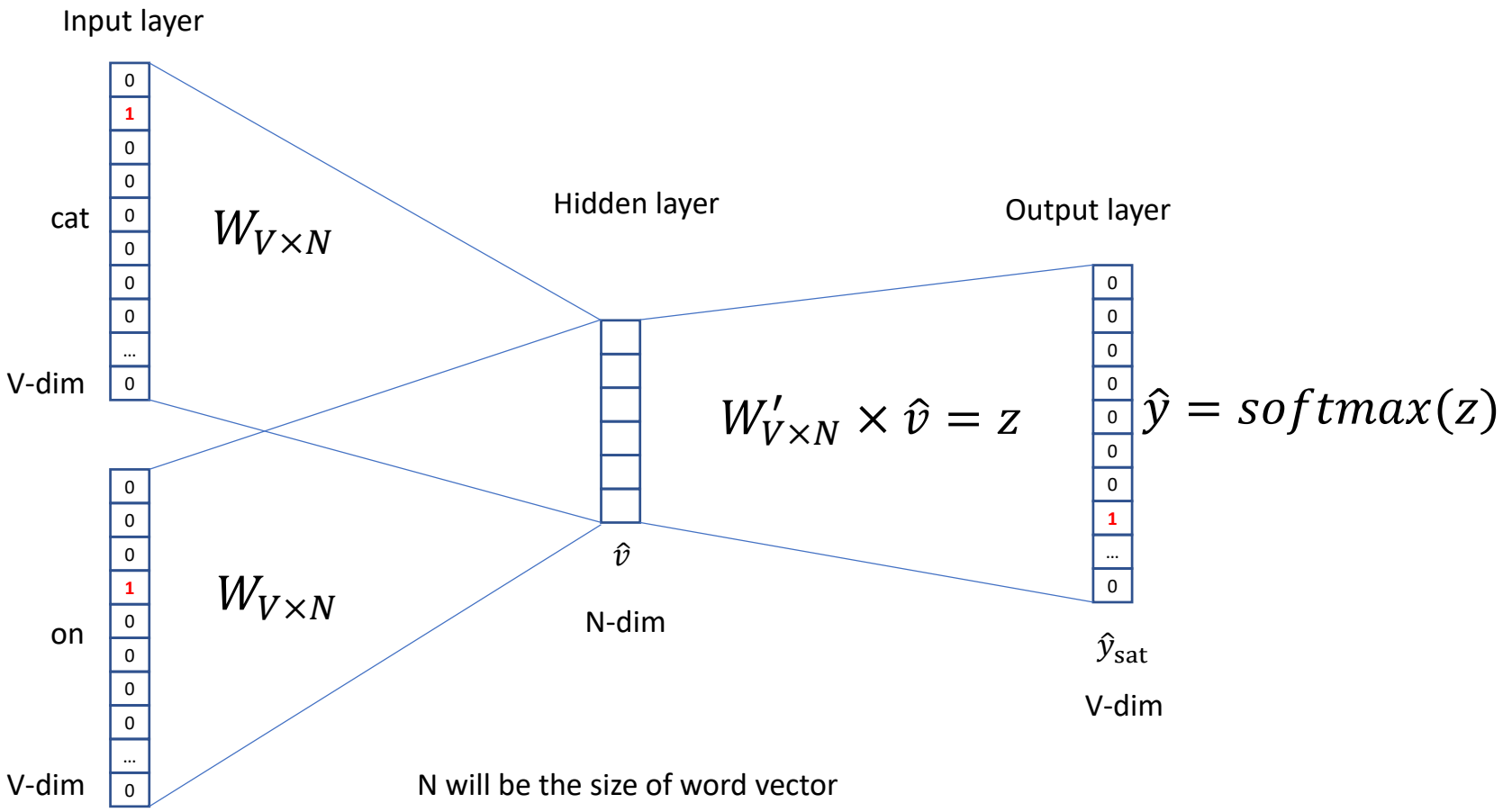
www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



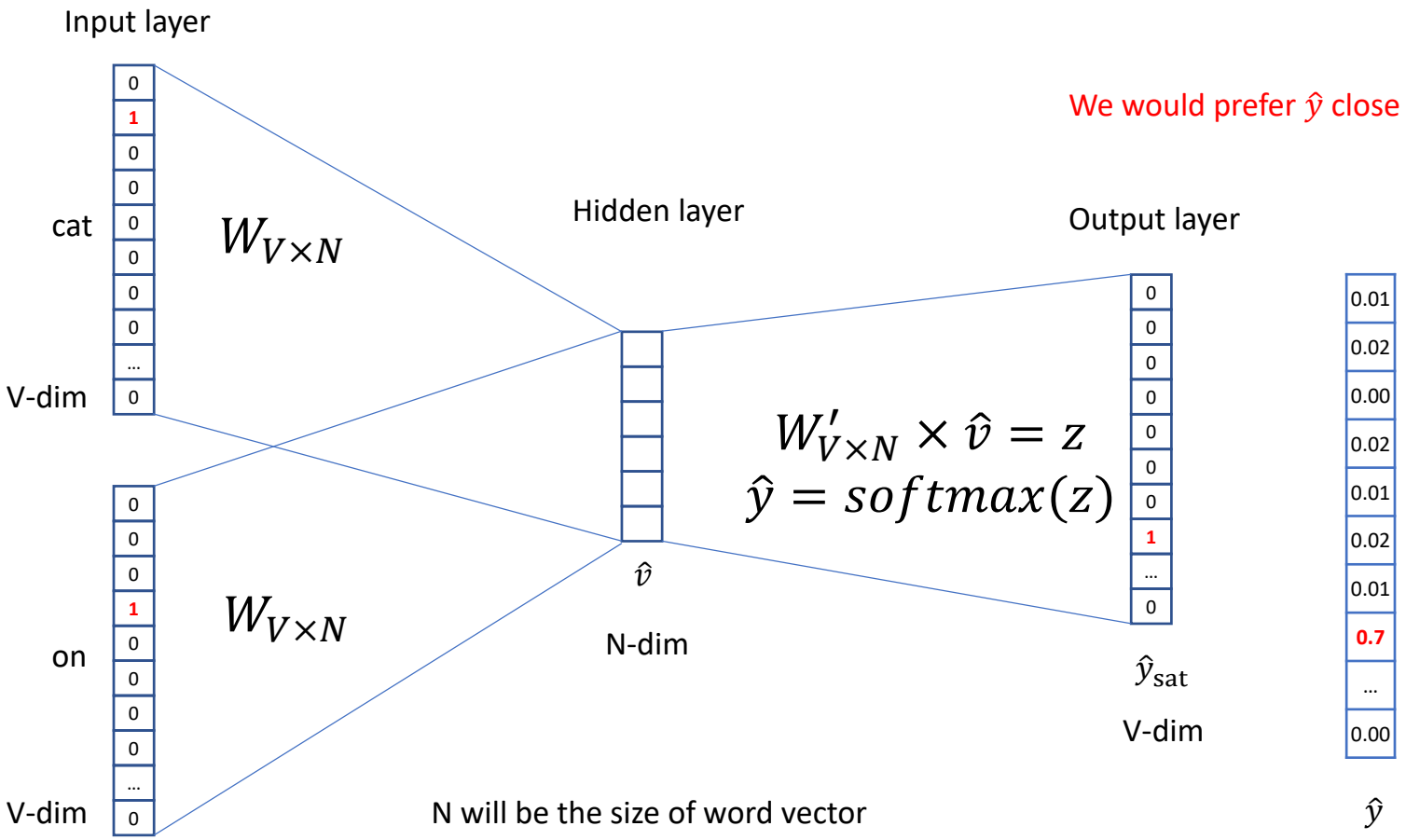
www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



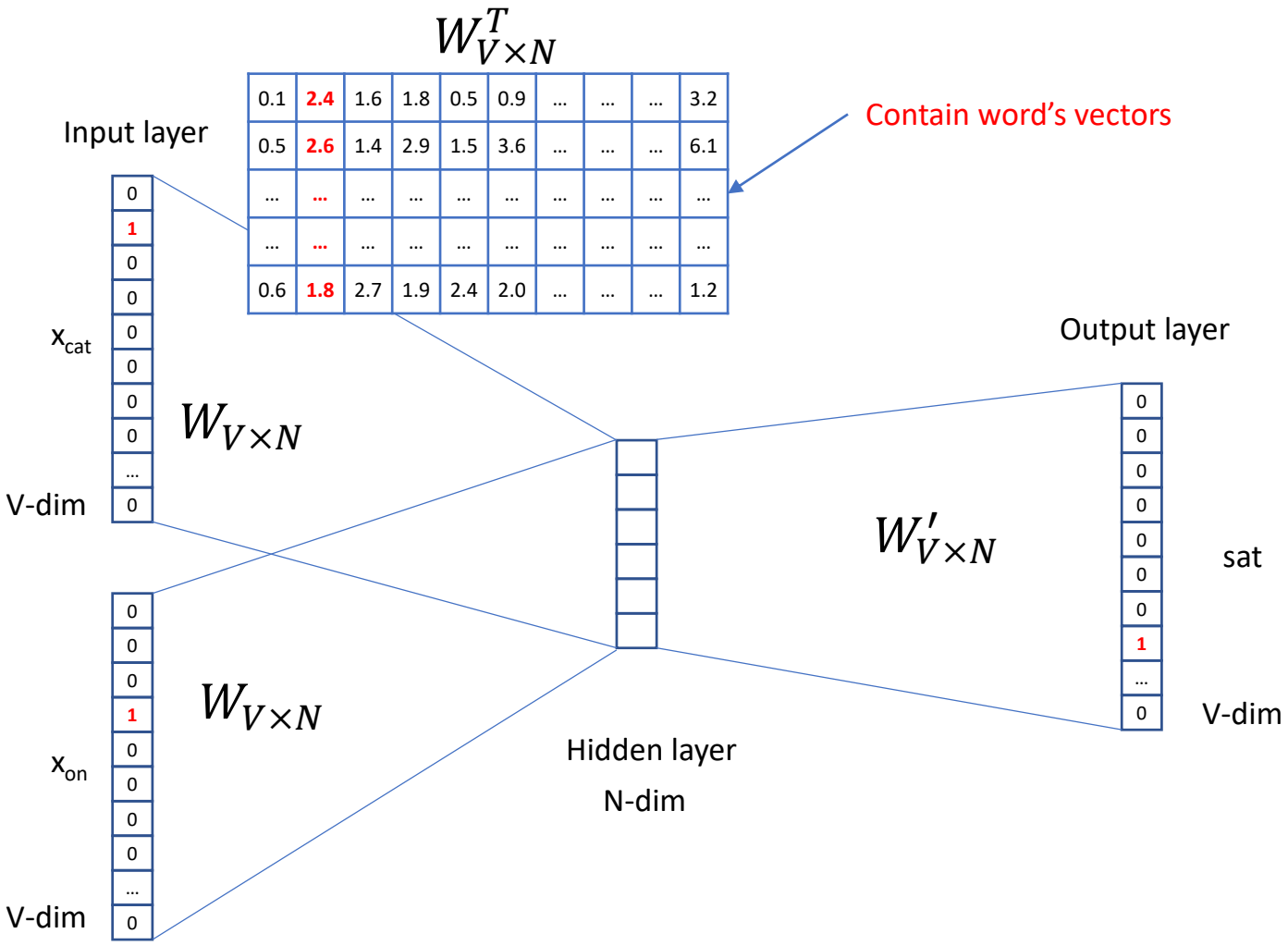
www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



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

www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

Some interesting results

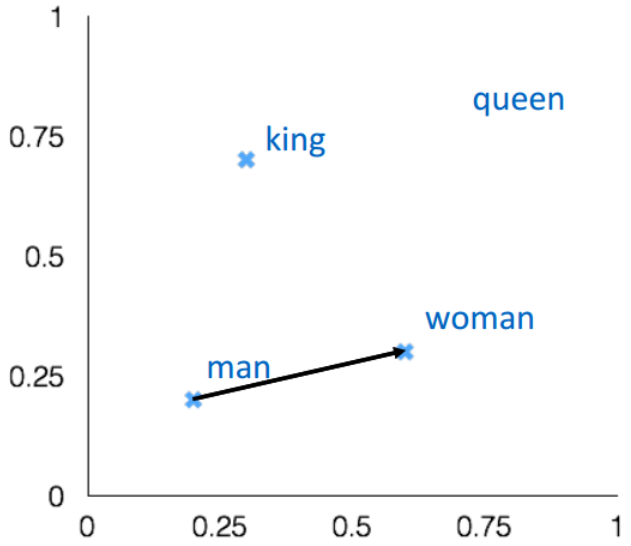
Word Analogies

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

a:b :: c:?

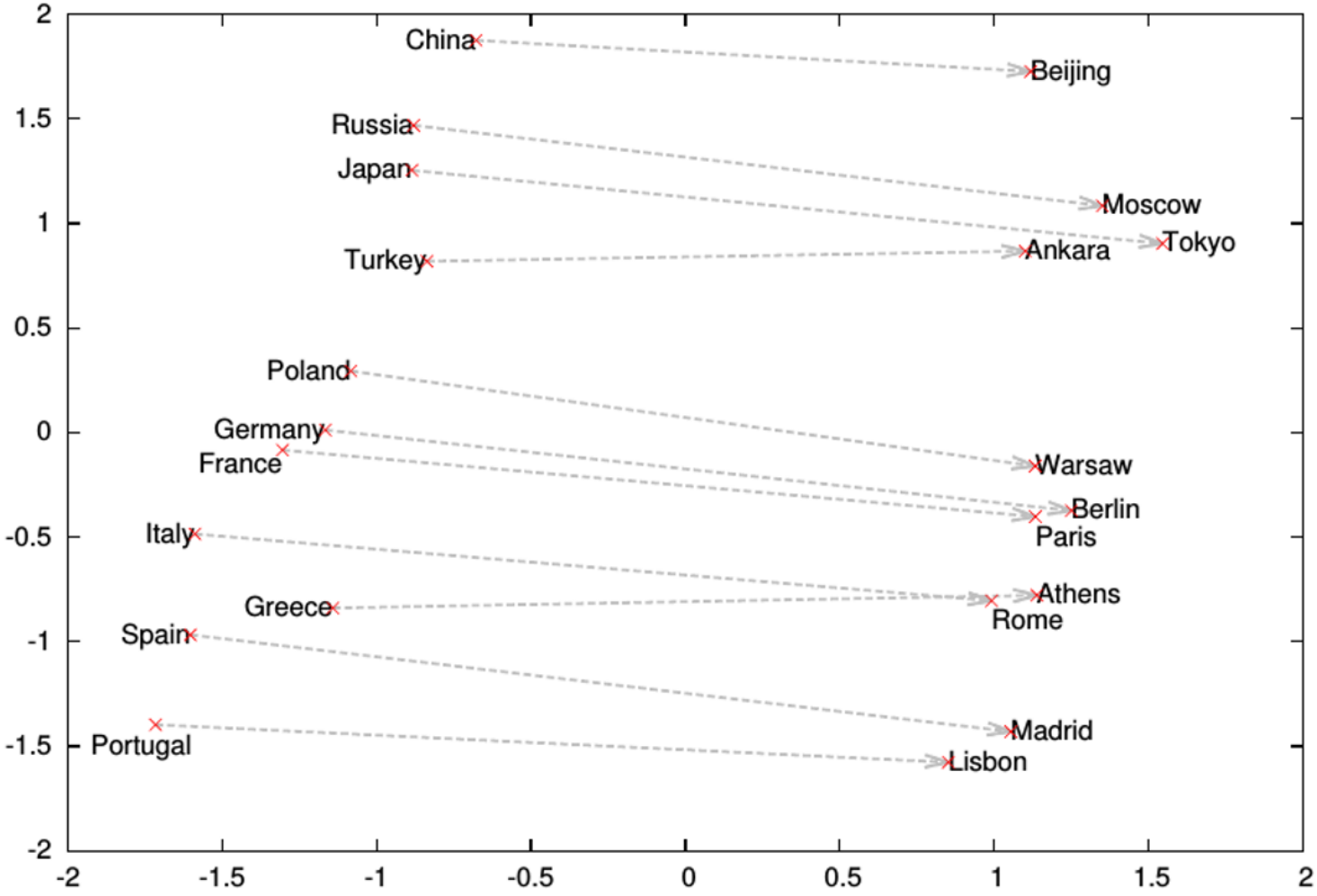
man:woman :: king:?

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



www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

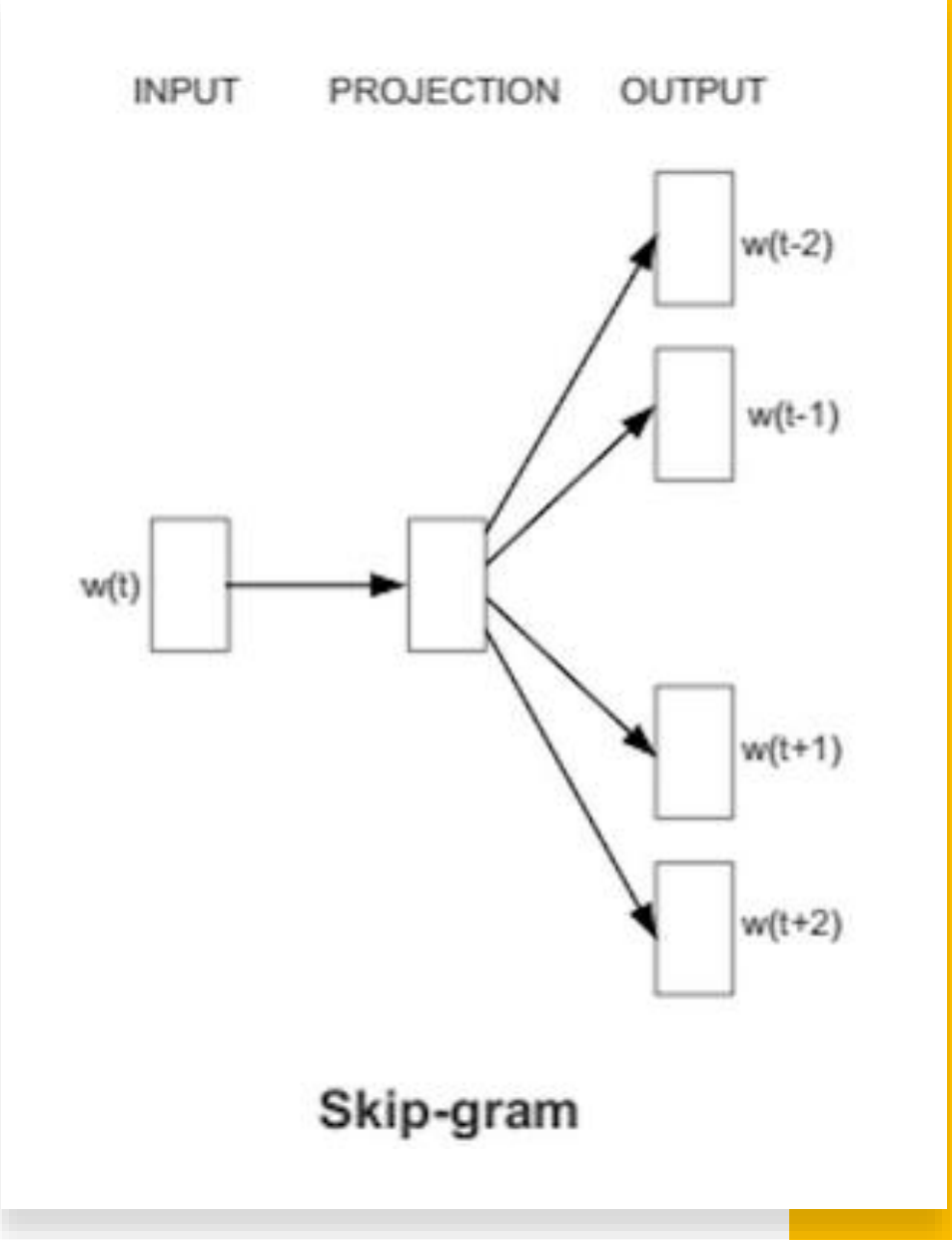
Word analogies



www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

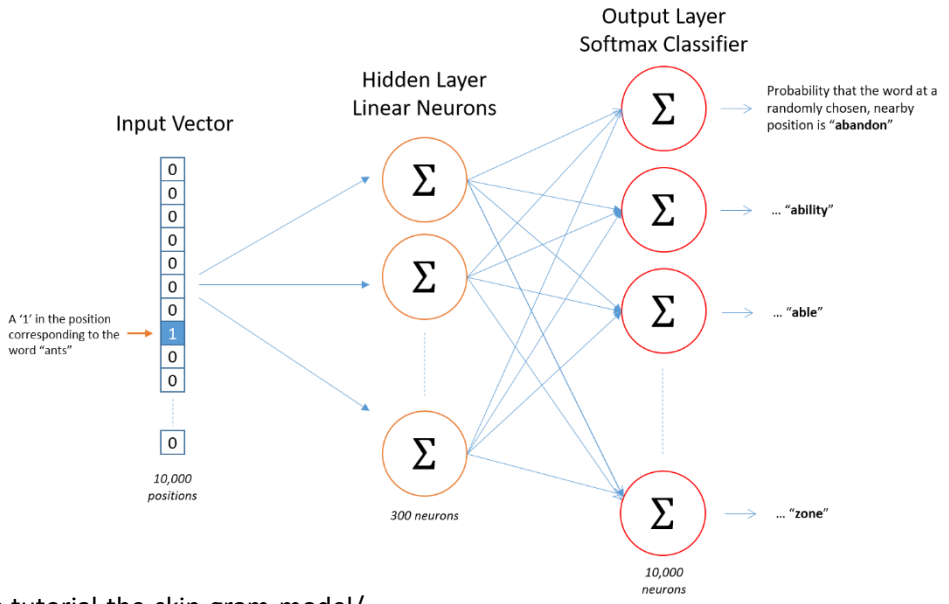
Skip gram

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



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.

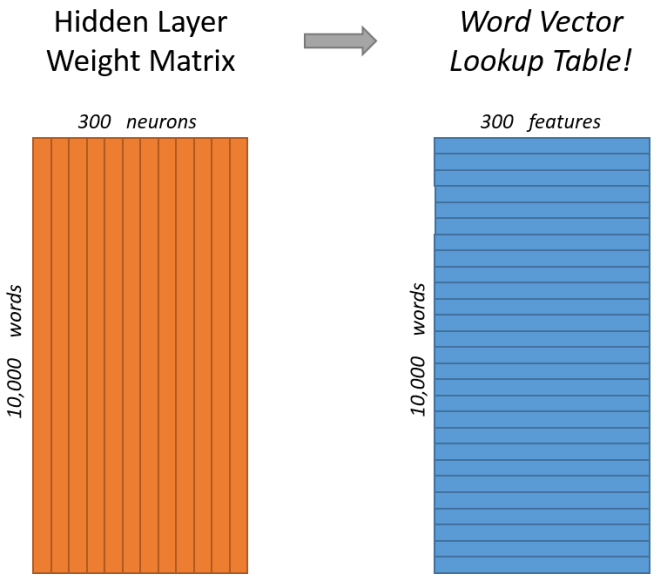


http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

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).

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



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

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.

Word2vec improvements: word pairs and phrases

Concept: Treat common word pairs or phrases as single "words." E.g., Boston Globe (newspaper) is different from Boston and Globe separately. Embed Boston Globe as a single word/phrase.

Method: make phrases out of words which occur together often relative to the number of individual occurrences. Prefer phrases made of infrequent words in order to avoid making phrases out of common words like "and the" or "this is".

Pros/cons: Increases vocabulary size but decreases training expense.

Word2vec improvements: subsample frequent words

Concept: Subsample frequent words to decrease the number of training examples.

Method: For each word, cut the word with probability related to the word's frequency.

Benefits: If we have a window size of 10, and we remove a specific instance of "the" from our text:

Word2vec improvements: selective updates

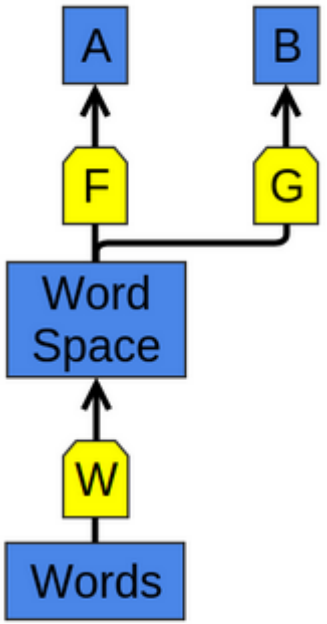
Concept: Use “Negative Sampling”, which causes each training sample to update only a small percentage of the model’s weights.

Observation: A “correct output” of the network is a one-hot vector. That is, one neuron should output a 1, and *all* of the other thousands of output neurons to output a 0.

Method: With negative sampling, randomly select just a small number of “negative” words (let’s say 5) to update the weights for. (In this context, a “negative” word is one for which we want the network to output a 0 for). We will also still update the weights for our “positive” word.

Applications

- 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.

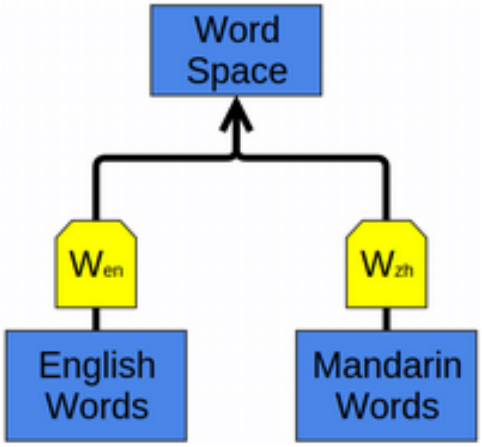


W and F learn to perform task A. Later, G can learn to perform B based on W.

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

Applications

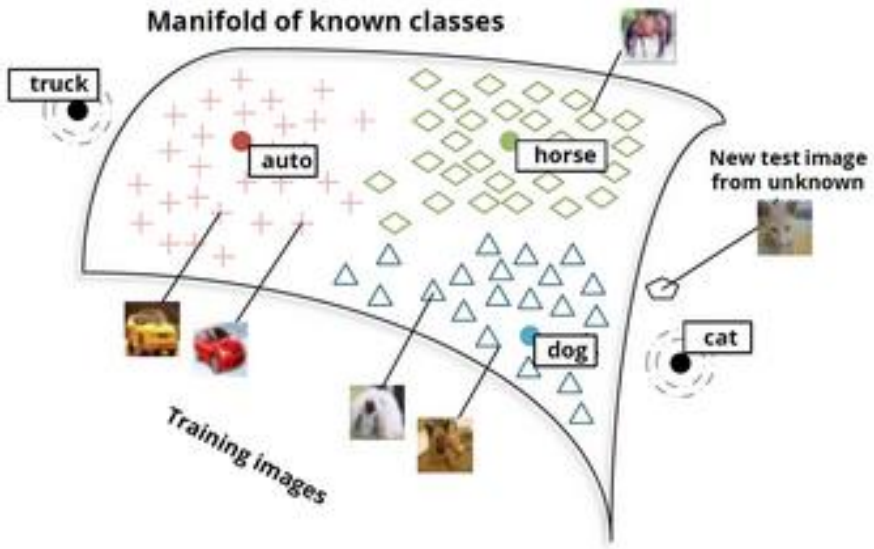
- Can learn to map multiple kinds of data into a single representation.
  - E.g., bilingual English and Mandarin Chinese word-embedding as in Socher *et al.* (2013a).
- Embed as above, but words that are known as close translations should be close together.
- Structures of two languages get pulled into alignment.



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

Applications

- Can apply to get a joint embedding of words and images or other multi-modal data sets.
- New classes map near similar existing classes: e.g., if ‘cat’ is unknown, cat images map near dog.



(Socher *et al.* (2013b))