Natural Language Processing

Natural Language Processing

Subfield of Artificial Intelligence that deals with enabling computers to understand human languages

- Speech
- Text

We will limit ourselves to text

Natural Language Processing

Some sub-problems in NLP for text:

- Machine translation
- Information retrieval
  - Question-answering
    - Factual: Watson in Jeopardy
    - Interpretation:
  - Text classification
    - By author
    - By topic
    - By sentiment
  - Summarization
    - Extract key sentences
    - 'Understand' and rephrase
  - Data extraction
    - Named entity recognition and identification
    - Word sense disambiguation

In the last 10 years or so, virtually all approached to these problems have relied on machine learning

Natural Language Processing

How can we represent text? – The One-hot representation

Use a vocabulary V, a list containing all allowed words, plus a special code for ‘out-of-vocabulary’.

For word i, the representation of would consist of all 0’s except for a 1 at position i

A document with n words is thus represented by a sequence of n word representations, requiring n*|V| space

This is not just memory intensive; it is hard for learning models to process so much information.

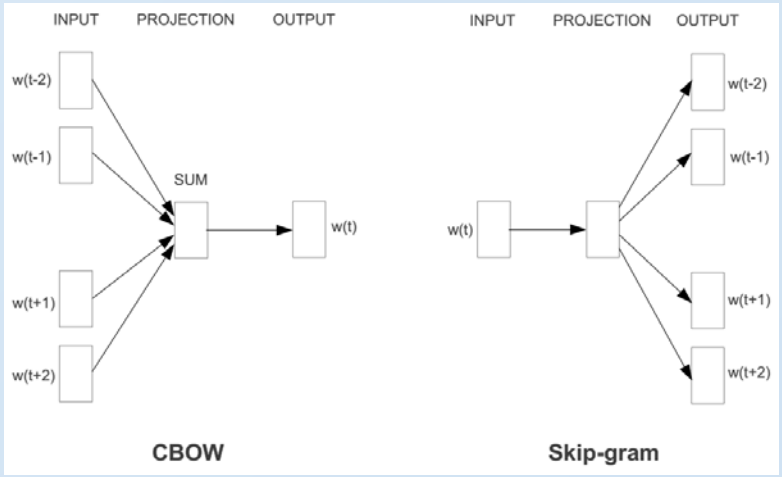
However, successful application to several information retrieval task using this approach were presented

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

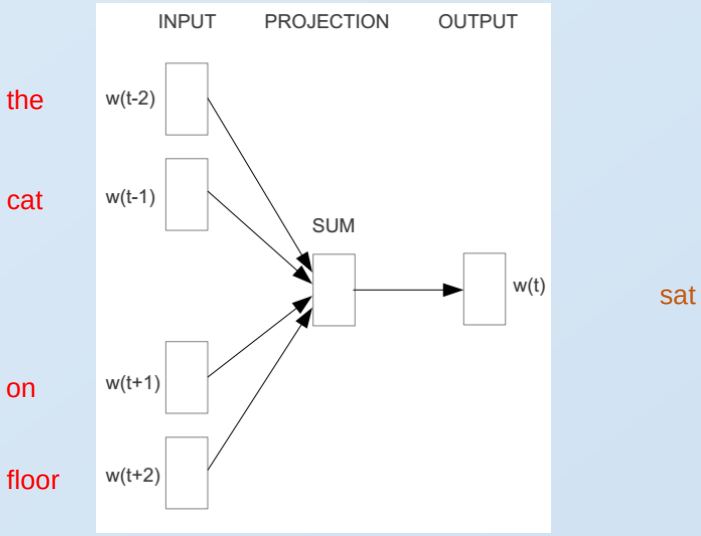
Represent the meaning of **word** – word2vec

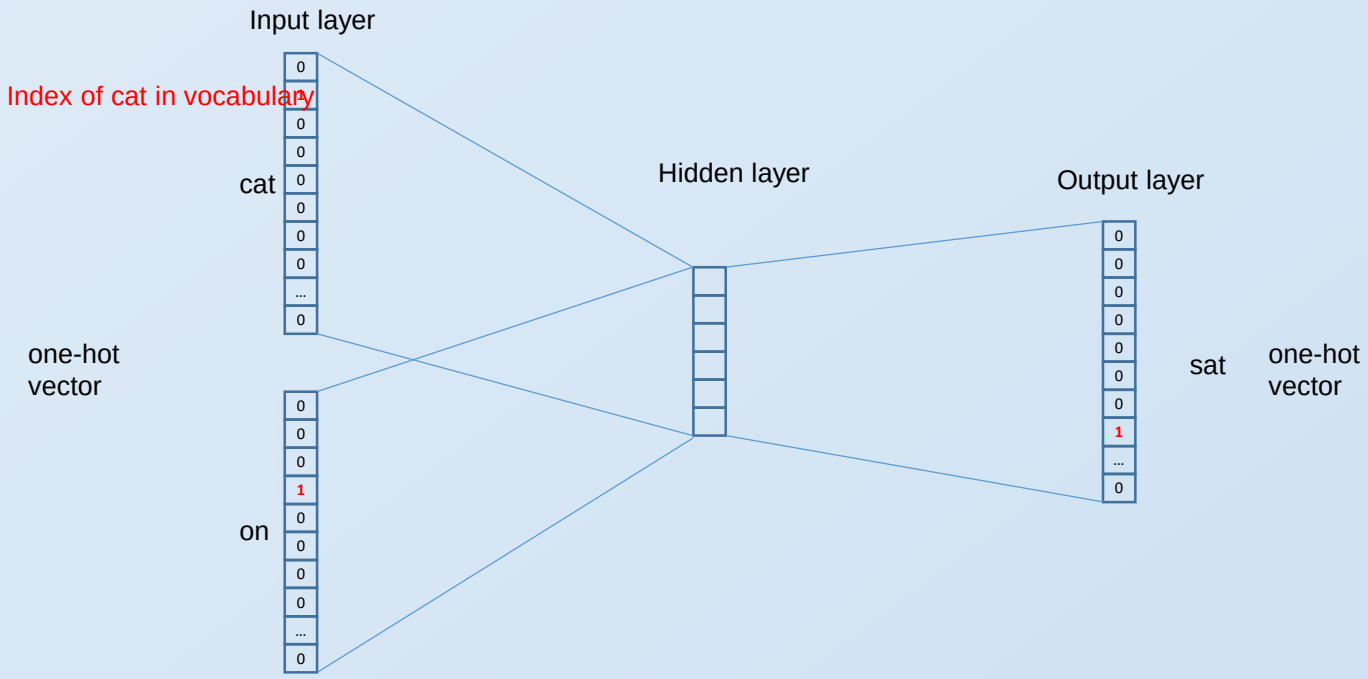
- 2 basic neural network models:
  - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.

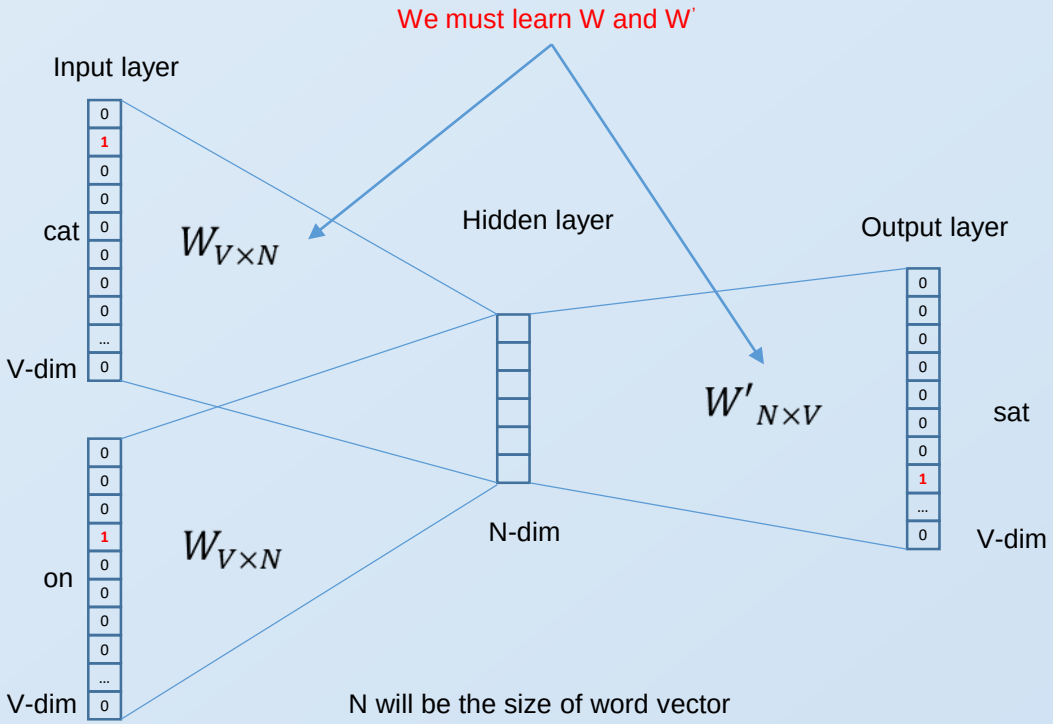


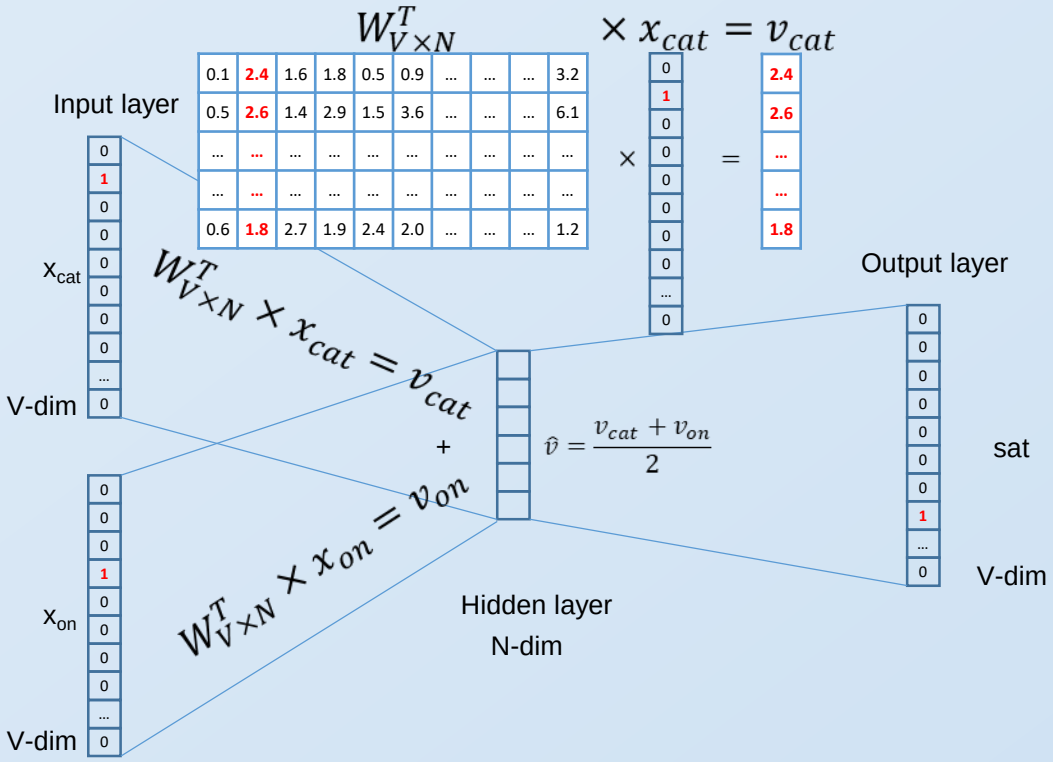
Word2vec – Continuous Bag of Word

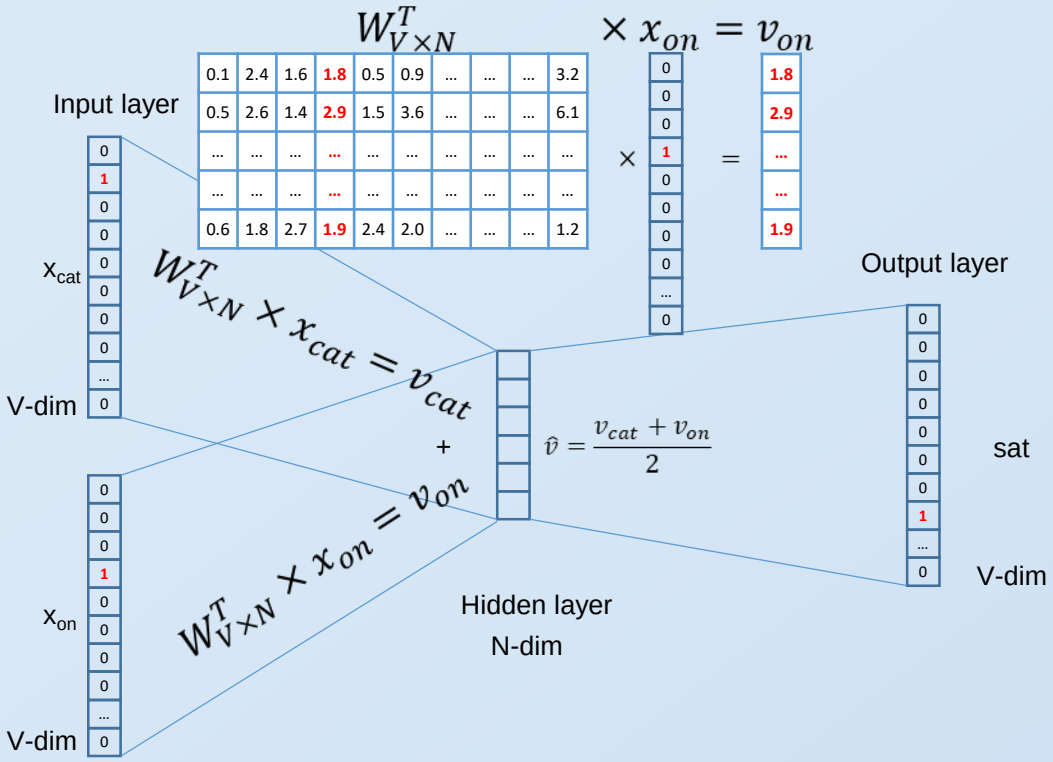
- E.g. “The cat sat on floor”
  - Window size = 2

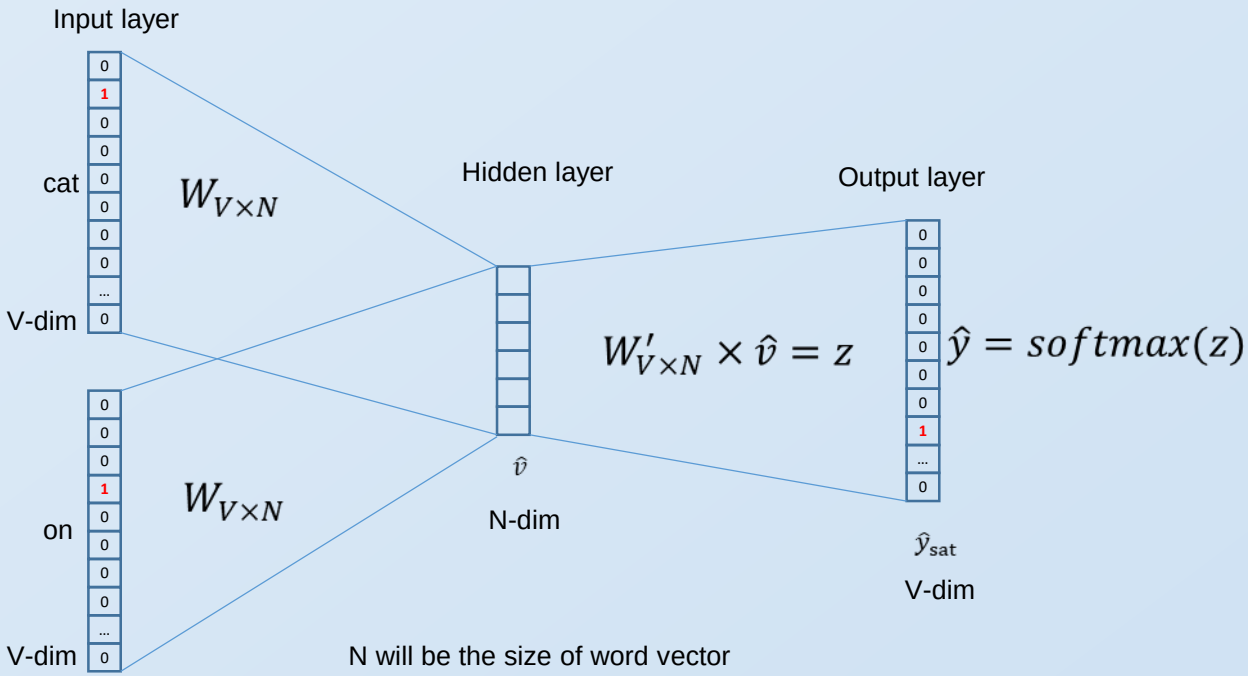


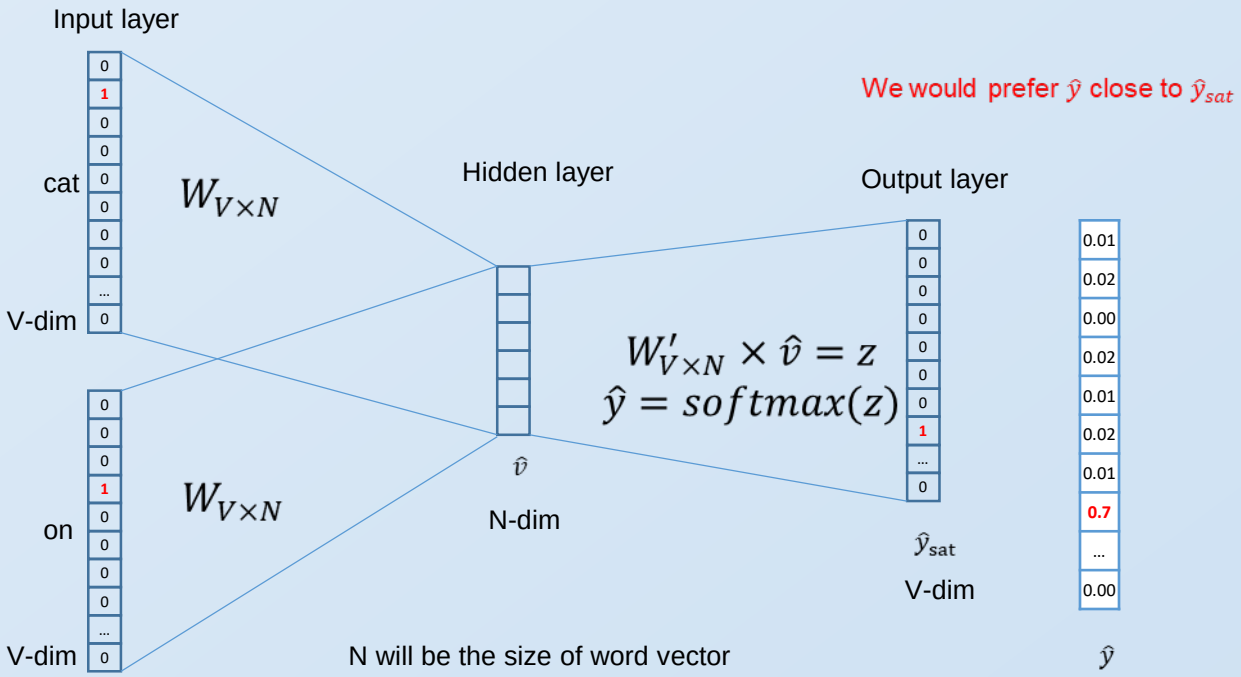

Input layer
Index of cat in vocabulary
cat
one-hot vector
on
Hidden layer
Output layer
sat
one-hot vector

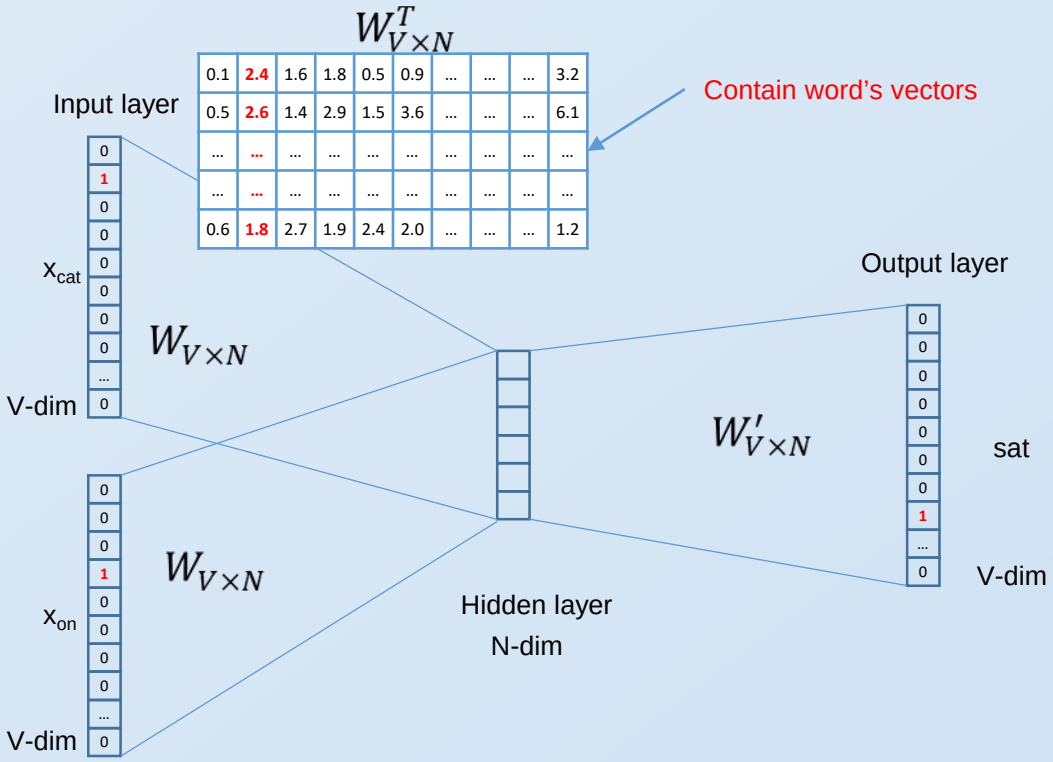

We must learn W and W'
Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
N-dim
$W'_{N\times V}$
Output layer
sat
V-dim
N will be the size of word vector


$W^T_{V\times N}$
$\times x_{cat} = v_{cat}$
Input layer
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


$W^T_{V\times N}$
$\times x_{on} = v_{on}$
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 0 0 1 0 0 0 0 ... 0
1.8 2.9 ... ... 1.9
x_{cat}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$W^T_{V\times N} \times x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
Output layer
0 0 0 0 0 0 0 0 1 ... 0
sat
V-dim
x_{on}
V-dim
Hidden layer
N-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

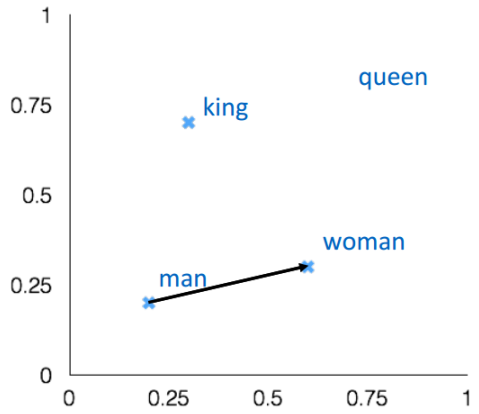
Test for linear relationships, examined by Mikolov et al. (2014)

a:b :: c:?

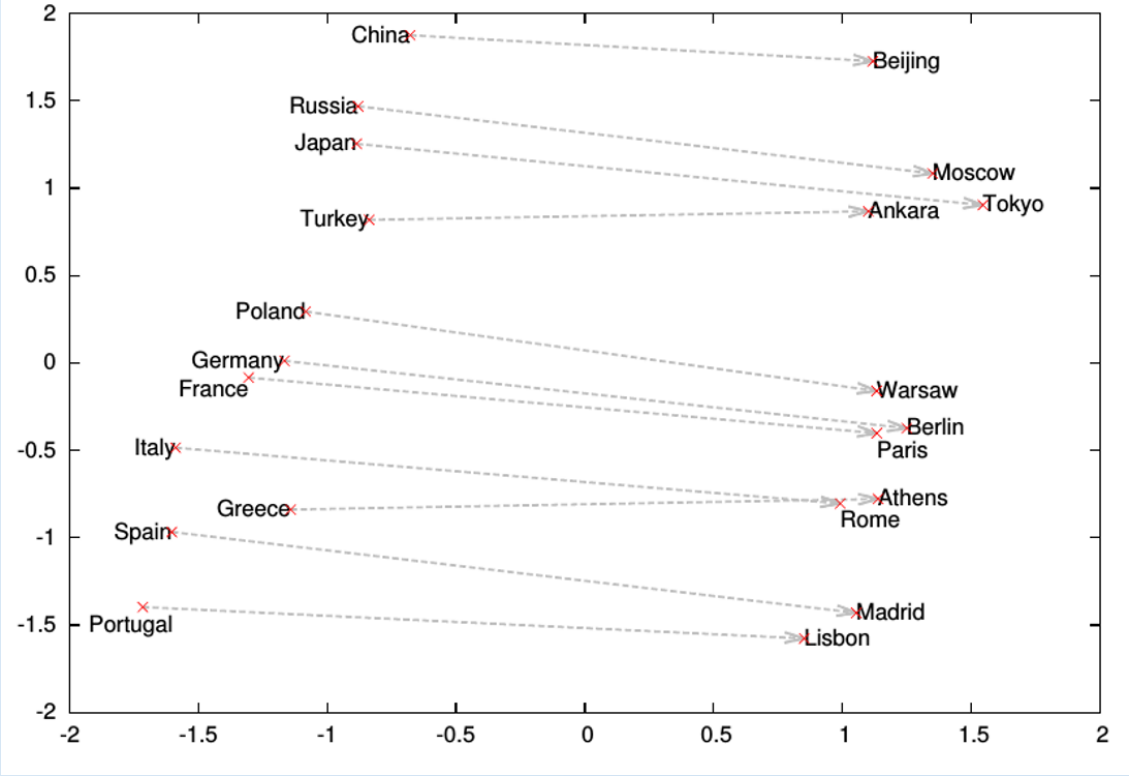
$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

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

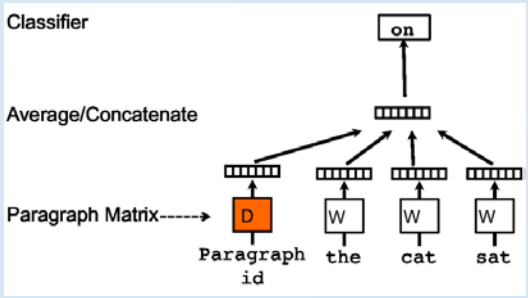


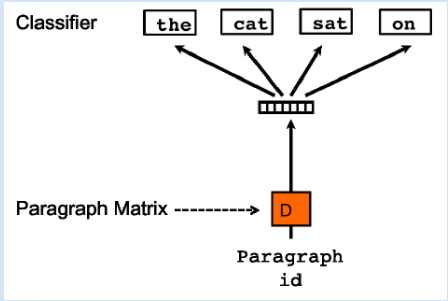
Word analogies



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





Represent the meaning of **sentence/text**

Better approaches:

- Use a 1D Convolutional neural network, where the input is a 2D array containing the embeddings of the text in the sequence
  - Convolutions can be applied across the rows (difference words) but not across the columns (different dimensions in the embedding)
  - We pad the arrays with zeros as necessary to deal with text documents of different lengths, since CNNs require inputs of constant size

- Use a **RECURRENT NEURAL NETWORK** – a NN architecture designed to deal with sequences
  - RNNs can be used to
    - Classify sequences
    - Perform sequence to sequence transformations -> Machine translation

Applications

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