Deep Learning

Natural Language Processing (NLP)

Lionel Fillatre

April 2021

Outline

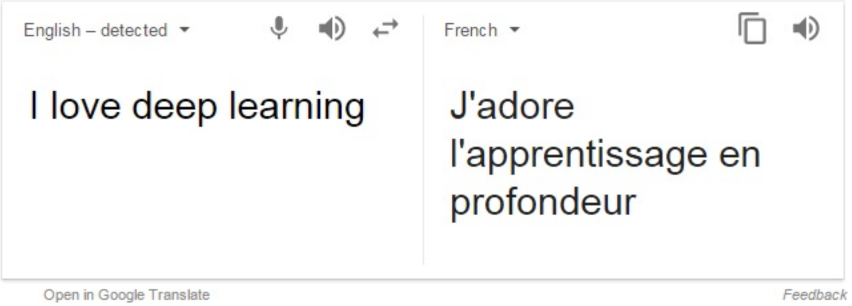
- Introduction
- Word Embedding
- Word2Vec
- Conclusion

Introduction

Natural Language Processing (NLP) Applications

- Sentiment Analysis
- Email Filters
- Voice Recognition
- Information Extraction
- Translation
- …

	Sentence	Class	index
0	So there is no way for me to plug it in here i...	0	0
1	Good case, Excellent value.	1	1
2	Great for the jawbone.	1	2
3	Tied to charger for conversations lasting more...	0	3
4	The mic is great.	1	4
5	I have to jiggle the plug to get it to line up...	0	5
6	If you have several dozen or several hundred c...	0	6
7	If you are Razr owner...you must have this!	1	7
8	Needless to say, I wasted my money.	0	8
9	What a waste of money and time!.	0	9
10	And the sound quality is great.	1	10



Language Model

Language Models

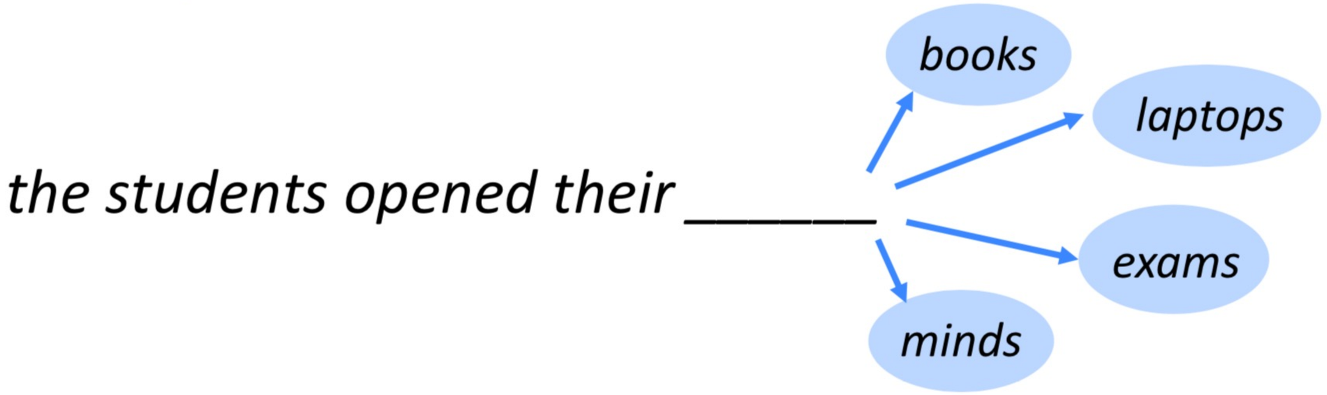
- Formal grammars (e.g. regular, context free) give a hard “binary” model of the legal sentences in a language.
- For NLP, a probabilistic model of a language that gives a probability that a string is a member of a language is more useful.
- To specify a correct probability distribution, the probability of all sentences in a language must sum to 1.

Uses of Language Models

- Speech recognition
  - “I ate a cherry” is a more likely sentence than “Eye eight uh Jerry”
- OCR & Handwriting recognition
  - More probable sentences are more likely correct readings.
- Machine translation
  - More likely sentences are probably better translations.
- Generation
  - More likely sentences are probably better Natural Language generations.
- Context sensitive spelling correction
  - “Their are problems wit this sentence.”

Language Modeling

- Langage Modeling is the task of predicting what word comes next



- Given a sequence of words $w_1, w_2, \ldots, w_t$, compute the probability distribution of the next word w_{t+1}

$$\Pr(w_{t+1} | w_t, w_{t-1}, \ldots, w_1)$$

where w_{t+1} can be any word in the vocabulary $V = \{v_1, v_2, \cdots, v_{|V|}\}$

- A system that does this is called a Language Model.

Completion Prediction

- A language model also supports predicting the completion of a sentence.
  - Please turn off your cell _____
  - Your program does not ______

- Predictive text input systems can guess what you are typing and give choices on how to complete it.

Language Modeling

- You can also think of a Language Model as a system that assigns probability to a piece of text.

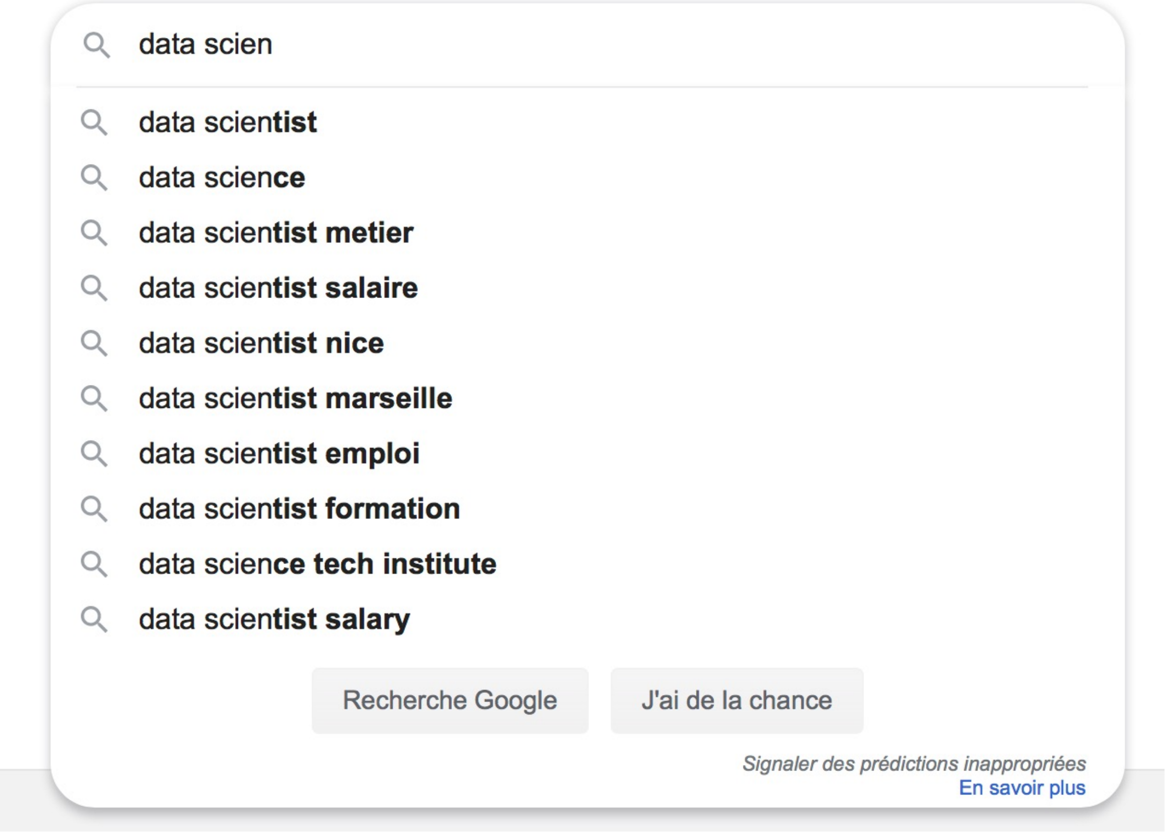
- For example, if we have some text $w_1, w_2, \ldots, w_T$, then the probability of this text (according to the Language Model) is:

$$\Pr(w_1, w_2, \cdots, w_T) = \prod_{t=1}^{N} \Pr(w_t | w_{t-1}, \ldots, w_1)$$

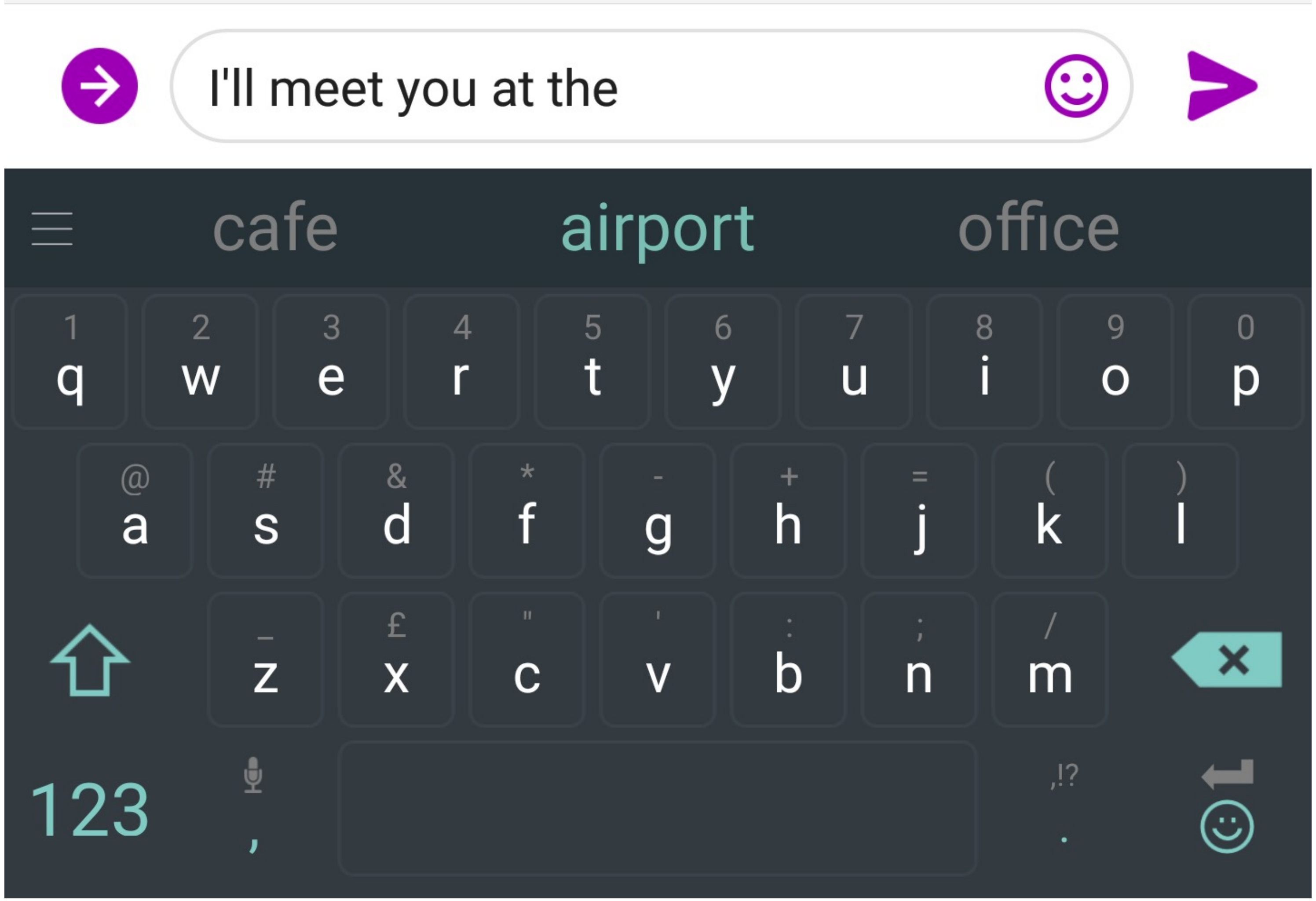
- Example: $\Pr$(its water was so transparent) = $\Pr$(its)* $\Pr$(water|its)* $\Pr$(was|its water)* $\Pr$(so|its water was)* $\Pr$(transparent|its water was so)

- Language Modeling provides $\Pr(w_t | w_{t-1}, \ldots, w_1)$

You use Language Models every day!



You use Language Models every day!



N-Gram

N-gram Language Models

- Question: How to learn a Language Model?

- Answer (pre-Deep Learning): Learn a n-gram Language Model!

- Definition: A n-gram is a chunk of n consecutive words.
  - unigrams: “the”, “students”, “opened”, ”their”
  - bigrams: “the students”, “students opened”, “opened their”
  - trigrams: “the students opened”, “students opened their”
  - 4-grams: “the students opened their”

- Idea: Collect statistics about how frequent different n-grams are, and use these to predict next word.

N-Gram Models

- Estimate probability of each word given prior context.
  - P(phone | Please turn off your cell)

- Number of parameters required grows exponentially with the number of words of prior context.

- An N-gram model uses only $N{-}1$ words of prior context.
  - Unigram: P(phone)
  - Bigram: P(phone | cell)
  - Trigram: P(phone | your cell)

- Markov model
  - The Markov assumption is the presumption that the future behavior of a dynamical system only depends on its recent history.
  - In particular, in a kth-order Markov model, the next state only depends on the k most recent states, therefore an N-gram model is a $(N{-}1)$-order Markov model.

N-gram Language Models

- First we make a simplifying assumption: preceding $n-1$ words

$$\Pr\left(w_{t+1} \middle| w_t, \ldots, w_1\right) = \Pr\left(w_{t+1} \middle| w_t, \ldots, w_{t-n+2}\right)$$

- Conditional probabilities:

$$\Pr\left(w_1 \middle| w_t, \ldots, w_{t-n+2}\right) = \frac{\Pr(w_{t+1}, w_t, \ldots, w_{t-n+2})}{\Pr(w_t, \ldots, w_{t-n+2})}$$

- **Question:** How do we get these n-gram and $(n-1)$-gram probabilities?

- **Answer:** By counting them in some large corpus of text!
  - Statistical approximation:

$$\Pr\left(w_{t+1} \middle| w_t, \ldots, w_{t-n+2}\right) \approx \frac{\text{count}(w_{t+1}, w_t, \ldots, w_{t-n+2})}{\text{count}(w_t, \ldots, w_{t-n+2})}$$

Estimating Probabilities

- N-gram conditional probabilities can be estimated from raw text based on the relative frequency of word sequences

  - Bigram: $\Pr\left(w_t \middle| w_{t-1}\right) = \frac{\mathrm{count}(w_{t-1} w_t)}{\mathrm{count}(w_{t-1})}$

  - N-gram: $\Pr\left(w_t \middle| w_{t-N+1}^{t-1}\right) = \frac{\mathrm{count}\left(w_{t-N+1}^{t-1} w_t\right)}{\mathrm{count}\left(w_{t-N+1}^{t-1}\right)}$

- To have a consistent probabilistic model, append a unique start (<s>) and end (</s>) symbol to every sentence and treat these as additional words, e.g.,
  - <s> I am Sam </s>
  - <s> Sam I am </s>

An Example

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$$

N-gram Language Models: Example

- Suppose we are learning a 4-gram Language Model

~~as the proctor started the clock~~, the students opened their ________

$$\Pr(w|\text{students opened their}) = \frac{\text{count}(\text{students opened their } w)}{\text{count}(\text{students opened their})}$$

- For example, suppose that in the corpus:
  - "students opened their" occurred 1000 times
  - "students opened their books" occurred 400 times
    - → $\Pr$(books | students opened their) = 0.4
  - "students opened their exams" occurred 100 times
    - → $\Pr$(exams | students opened their) = 0.1

Sparsity Problems with n-gram Language Models

Problem: What if "students opened their w" never occurred in data? Then w has probability 0!

(Partial) Solution: Add small δ to the count for every $w \in V$. This is called smoothing.

$$\Pr(w|\text{students opened their}) = \frac{\text{count(students opened their } w)}{\text{count(students opened their)}}$$

Problem: What if "students opened their" never occurred in data? Then we can not calculate probability for any w !

(Partial) Solution: Just condition on "opened their" instead. This is called backoff.

Note: Increasing n makes sparsity problems worse. Typically we can't have n bigger than 5.

Storage Problems with n-gram Language Models

Storage: Need to store count for all n-grams you saw in the corpus.

$$\Pr(w|\text{students opened their}) = \frac{\text{count}(\text{students opened their } w)}{\text{count}(\text{students opened their})}$$

Increasing n or increasing corpus increases model size!

n-gram Language Models in practice

- You can build a simple trigram Language Model over a 1.7 million word corpus (Reuters) in a few seconds on your laptop

- Example:
  - *today the* _______

  - The probability distribution is

Company	0.153
Bank	0.153
price	0.077
italian	0.039
emirate	0.039

  - It seems reasonable but sparsity problem: not much granularity in the probability distribution to decide between « Company » and « Bank »

Generating text with a n-gram Language Model

- You can use a Language Model to generate text.

today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share.

- Surprisingly grammatical!
- ...but incoherent. We need to consider more than three words at a time if we want to model language well.
- But increasing n worsens sparsity problem, and increases model size...

Word Embedding

How do we represent the meaning of a word?

- Definition: meaning
  - The idea that is represented by a word, phrase, etc.
  - The idea that a person wants to express by using words, signs, etc.
  - The idea that is expressed in a work of writing, art, etc.

- Commonest linguistic way of thinking of meaning:

signifier (symbol) ⟺ signified (idea or thing)

Representing words as discrete symbols

- In traditional NLP, we regard words as discrete symbols:
  - hotel, conference, motel – a localist representation

- Words can be represented by one-hot vectors:
  - motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
  - hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

- One-hot encoding
  - Vector dimension = number of words in vocabulary V (e.g., 500,000)
  - The number of words in the vocabulary is $|V|$
  - The vector consists of 0s in all cells with the exception of a single 1 in a cell used uniquely to identify the word.

Problem with words as discrete symbols

- **Example:** in web search, if user searches for “Seattle motel”, we would like to match documents containing “Seattle hotel”.

- But:
  - motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
  - hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
  - These two vectors are orthogonal.

- There is no natural notion of **similarity** for one-hot vectors!

- **Solution:**
  - Learn to encode similarity in the vectors themselves

Distributional Hypothesis

- The Distributional Hypothesis in linguistics is derived from the semantic theory of language usage, i.e.
  - Words that are used and occur in the same contexts tend to purport similar meanings.
  - « A word is characterized by the company it keeps ». (J. R. Firth, 1957)
  - « The complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously. »

- Use the many contexts of w to build up a representation of w
- Example:
  - These context words will represent banking

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

Word vectors

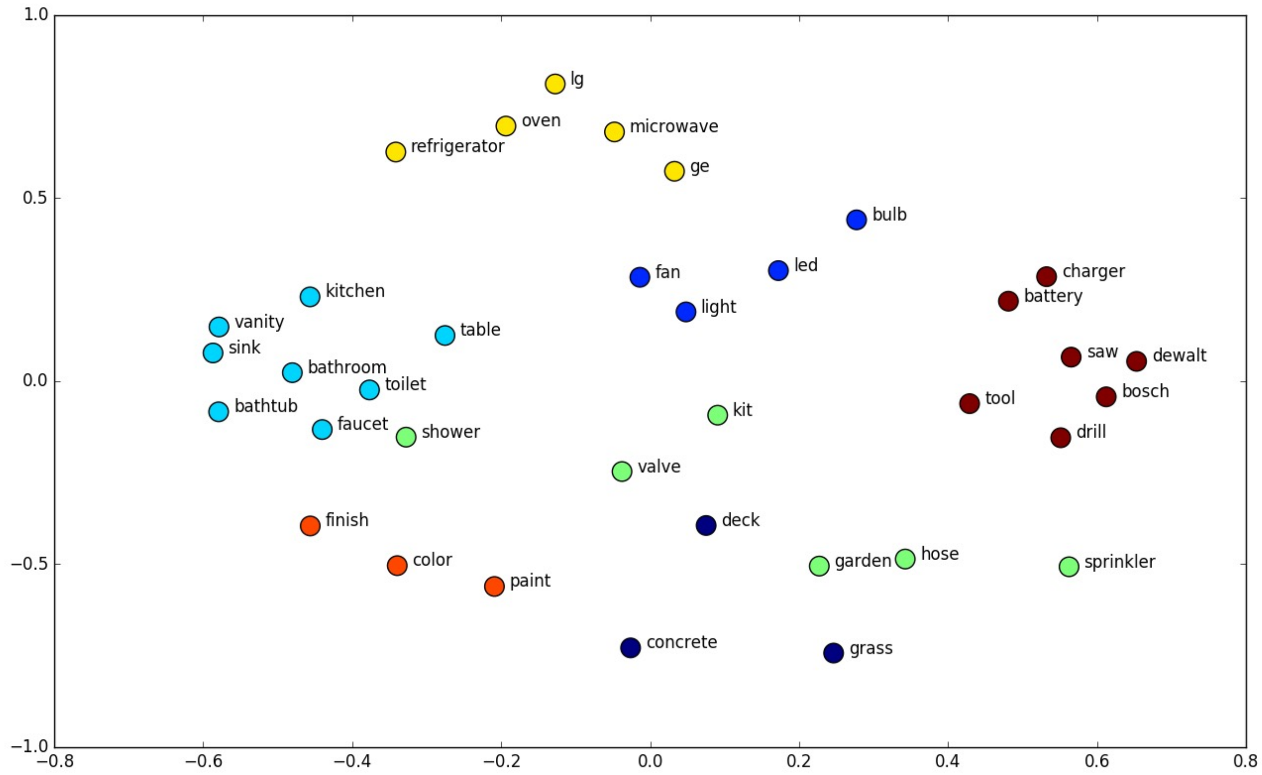
- We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

- Note: word vectors are sometimes called word embeddings or word representations. They are a distributed representation.

Representation Space

- Example 2D word embedding space, where similar words are found in similar locations



https://www.r-craft.org/r-news/get-busy-with-word-embeddings-an-introduction/

Word2Vec

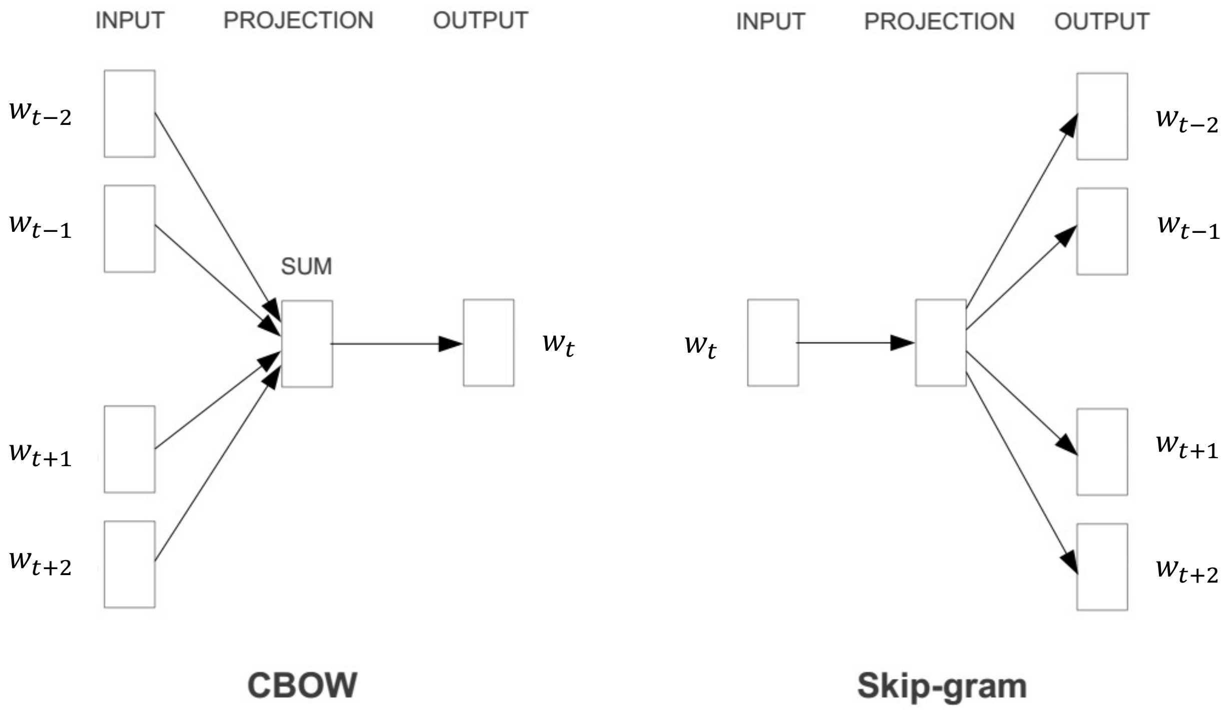
Word2vec: Overview

- Word2vec (Mikolov et al. 2013) is a framework for learning word vectors

- Idea:
  - We have a large corpus of text
  - Every word in a fixed vocabulary is represented by a vector
  - Go through each position t in the text, which has a center word c and context ("outside") words o
  - Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
  - Keep adjusting the word vectors to maximize this probability

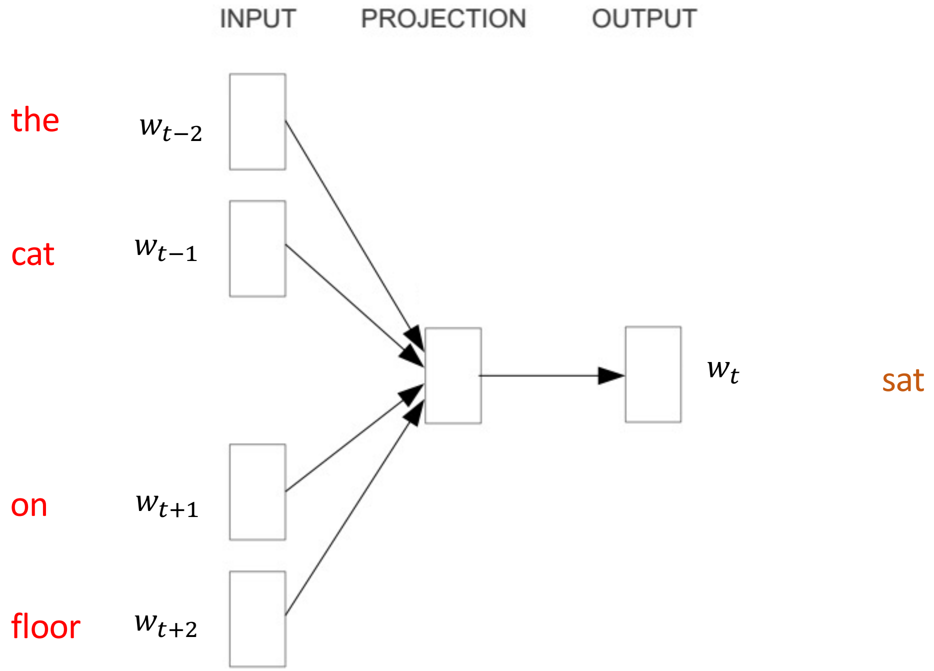
Represent the meaning of word – word2vec

- 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.

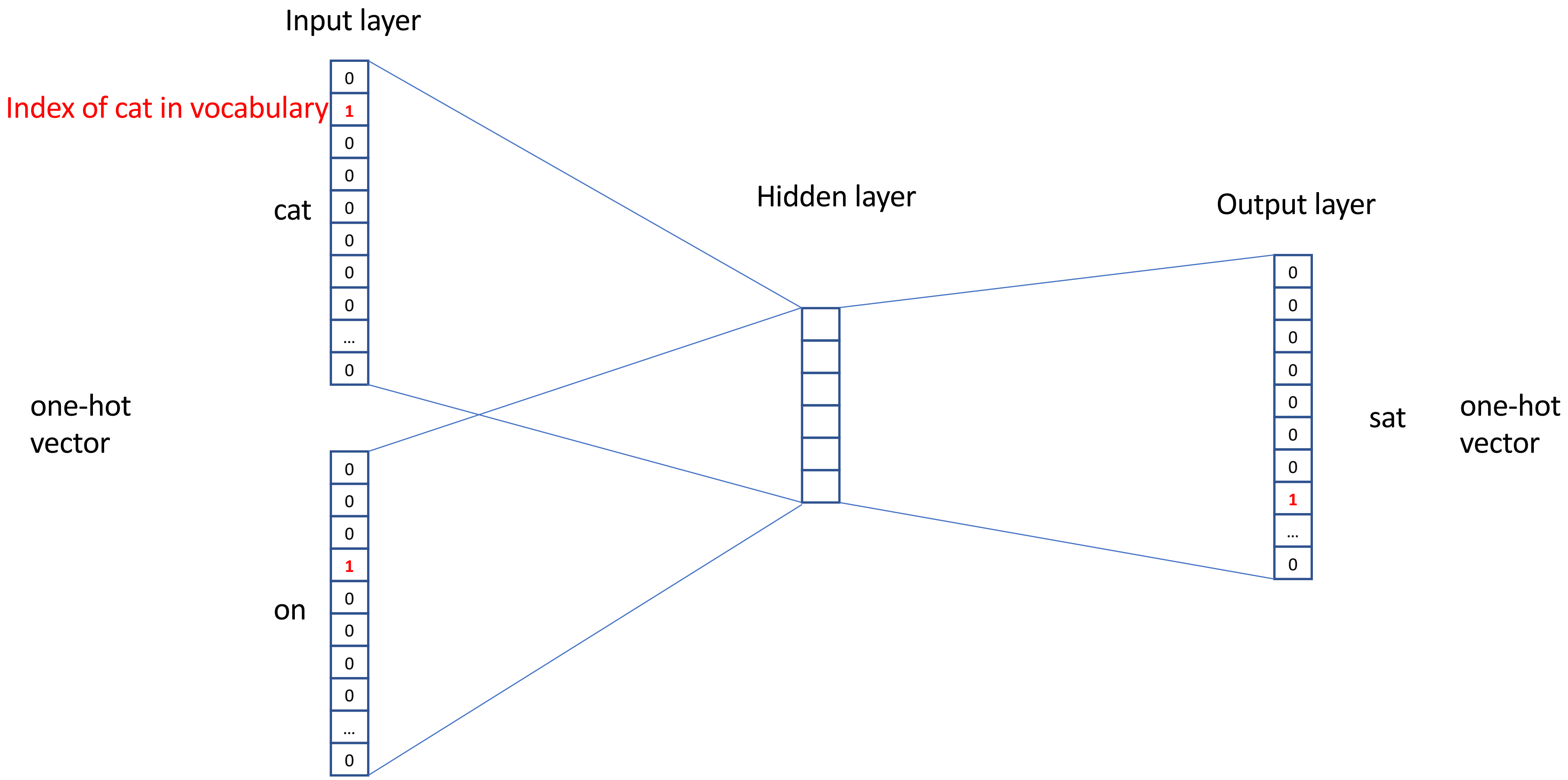


Word2vec – Continuous Bag of Word

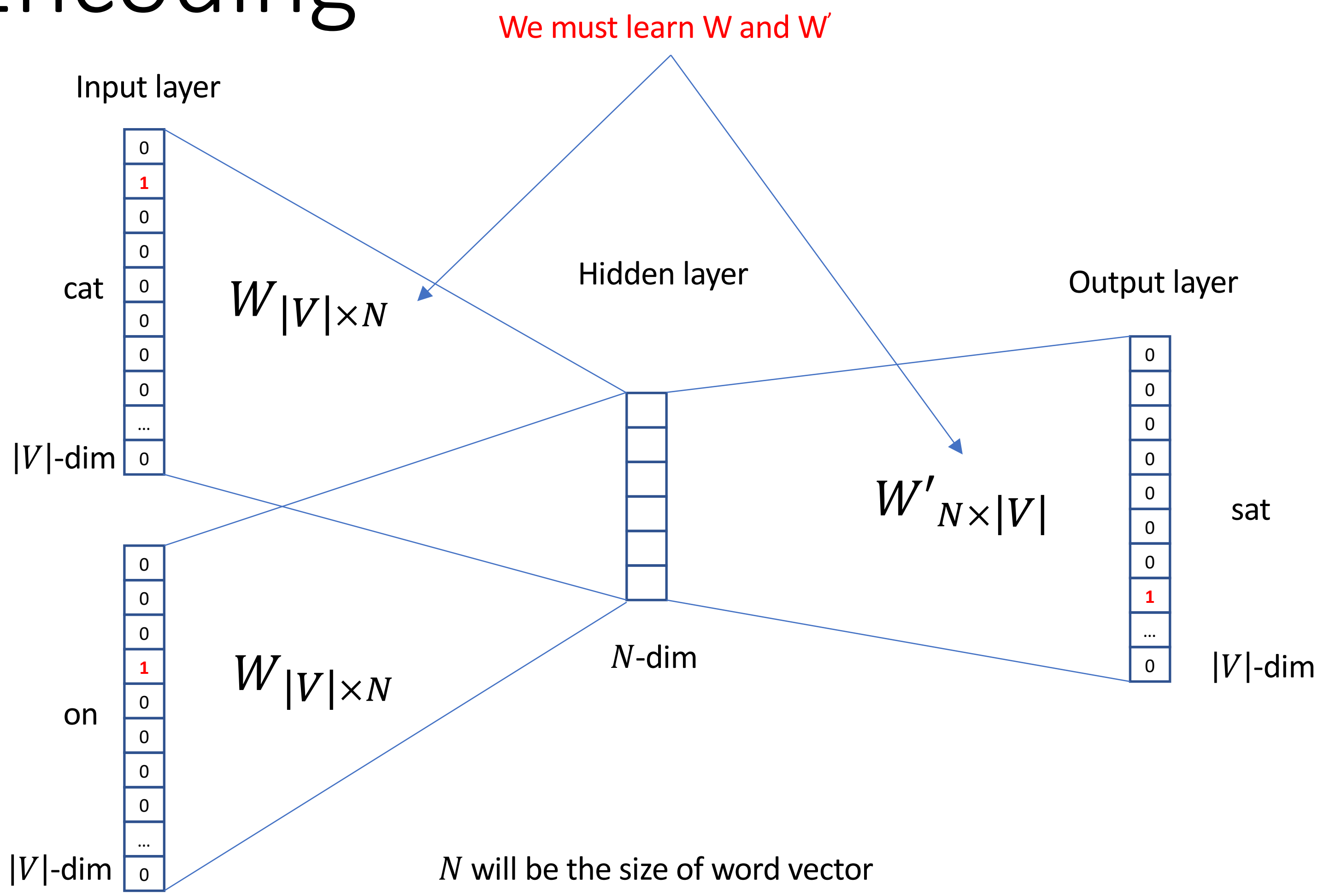
- E.g. “The cat sat on floor”
  - Window size = 2



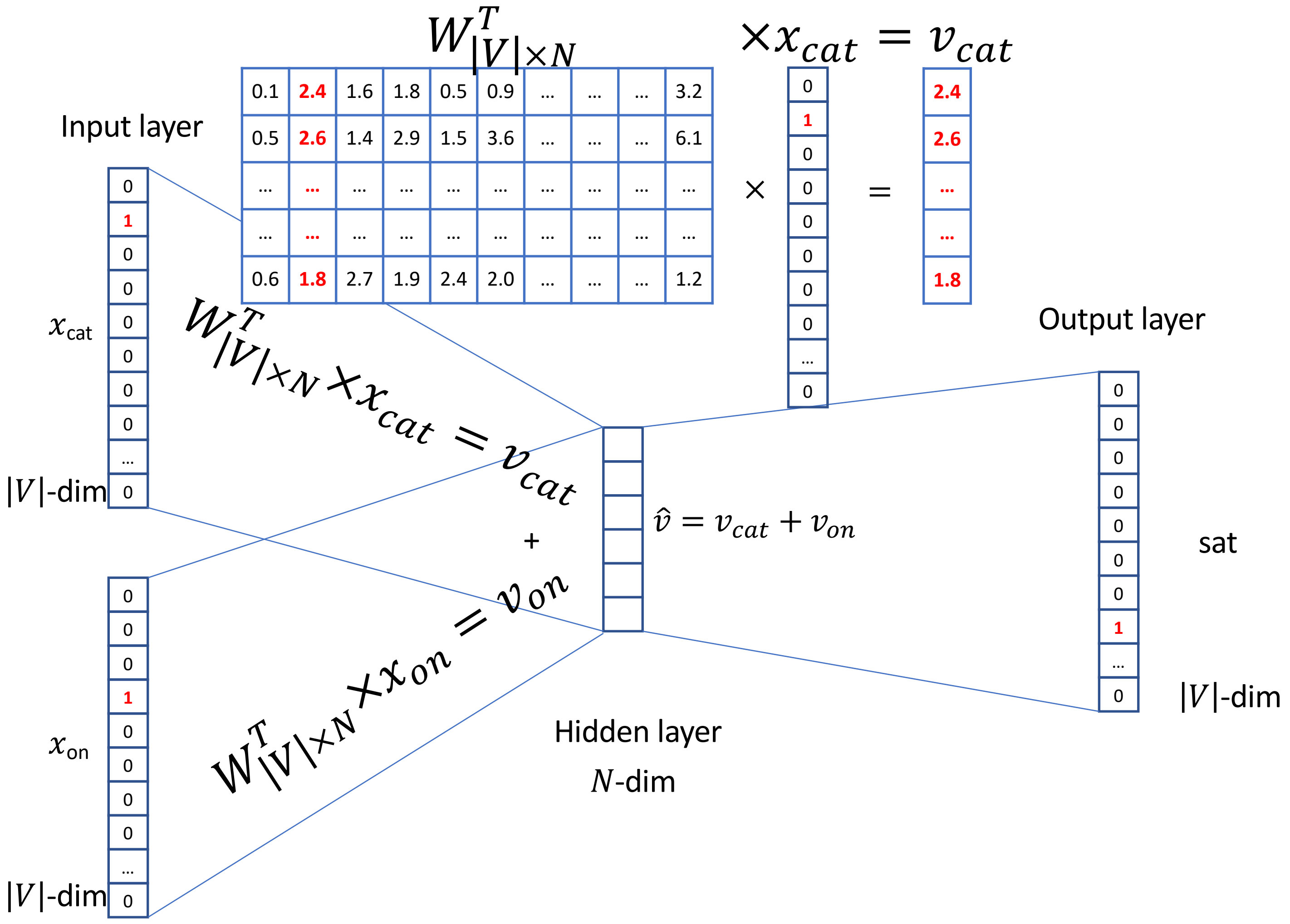
Input Encoding (window size = 1)



Matrix Encoding

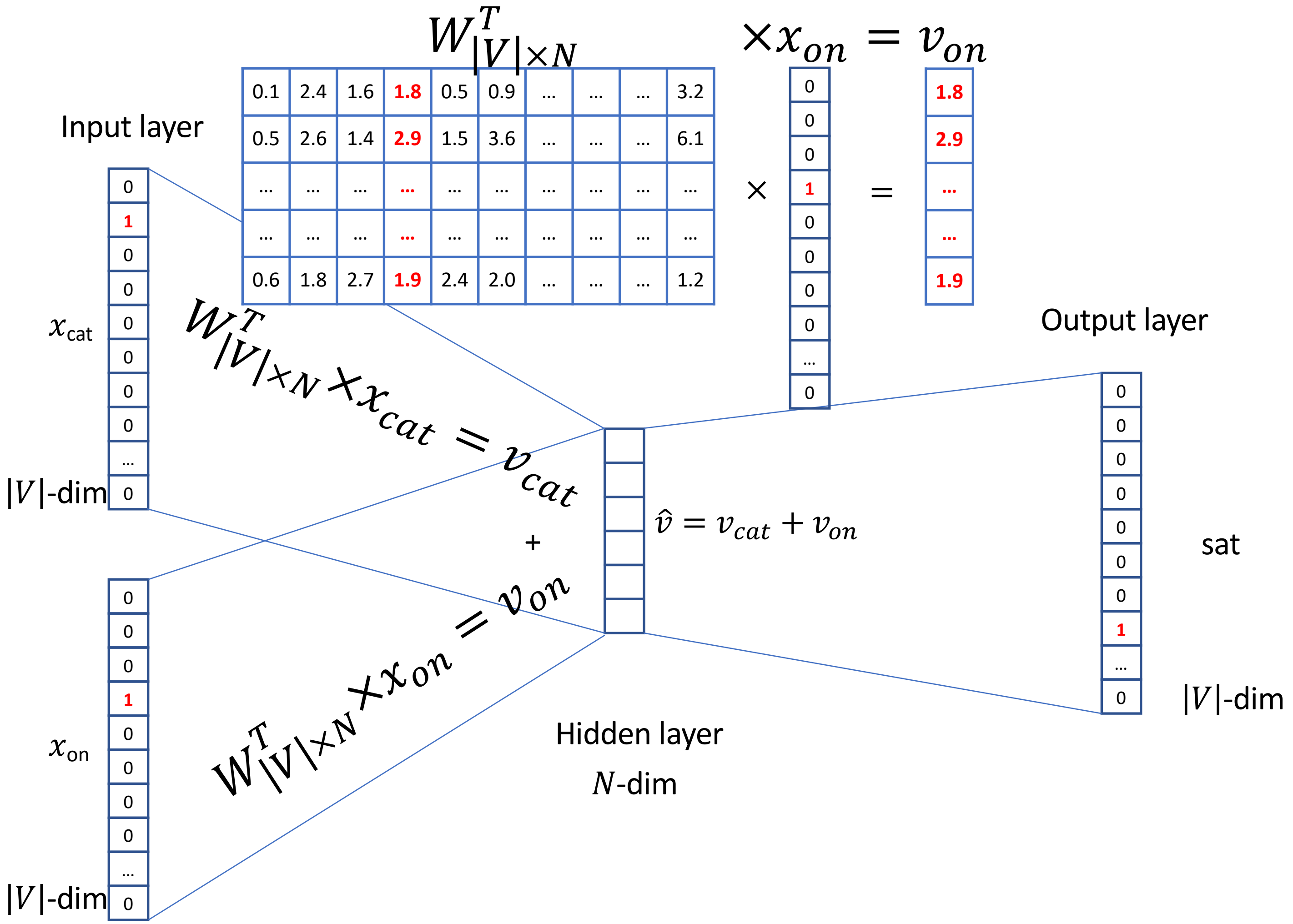


Example



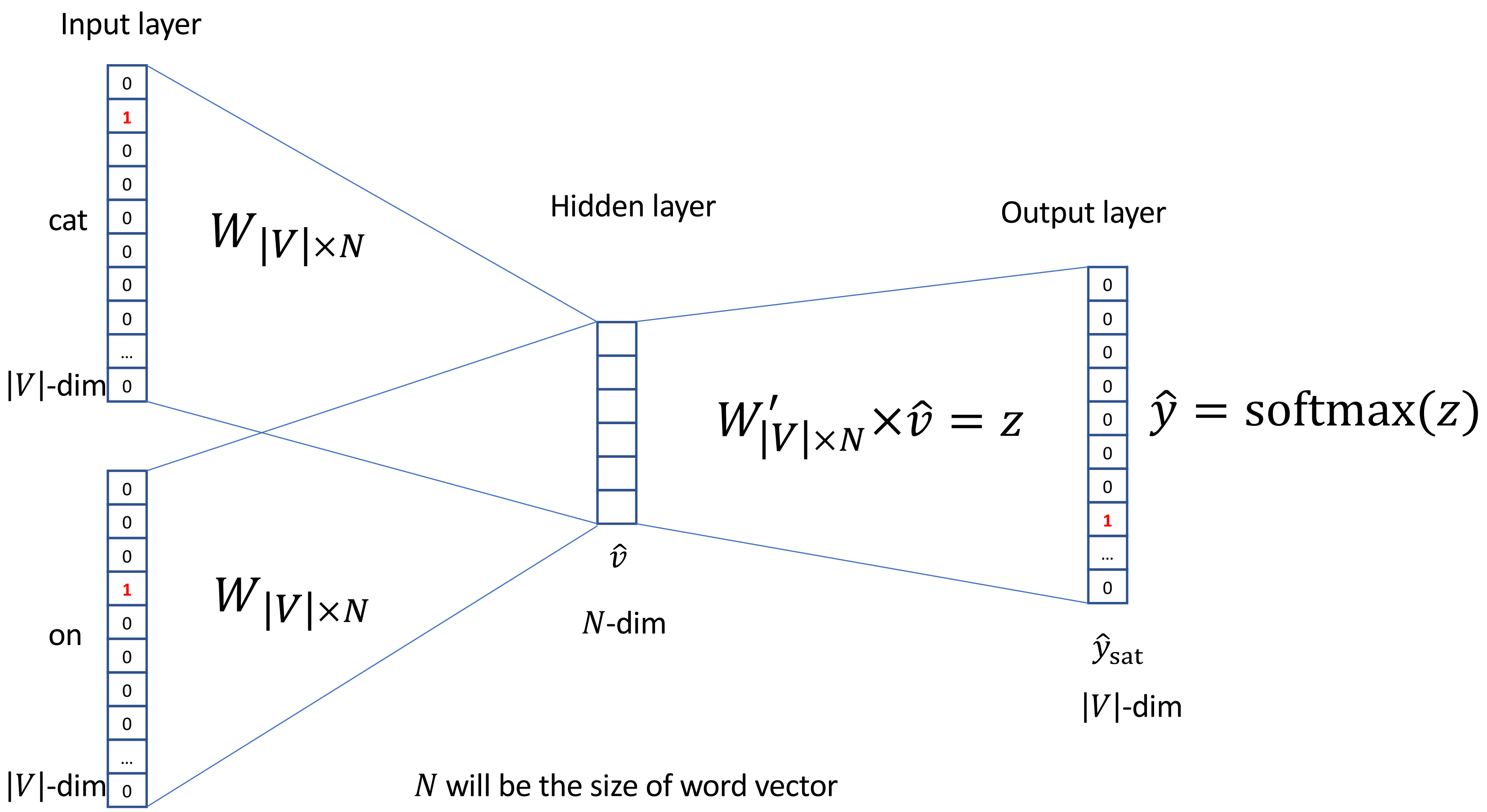
Note that we can take $\hat{v} = \frac{v_{cat}+v_{on}}{2}$ instead of $\hat{v} = v_{cat} + v_{on}$

Example

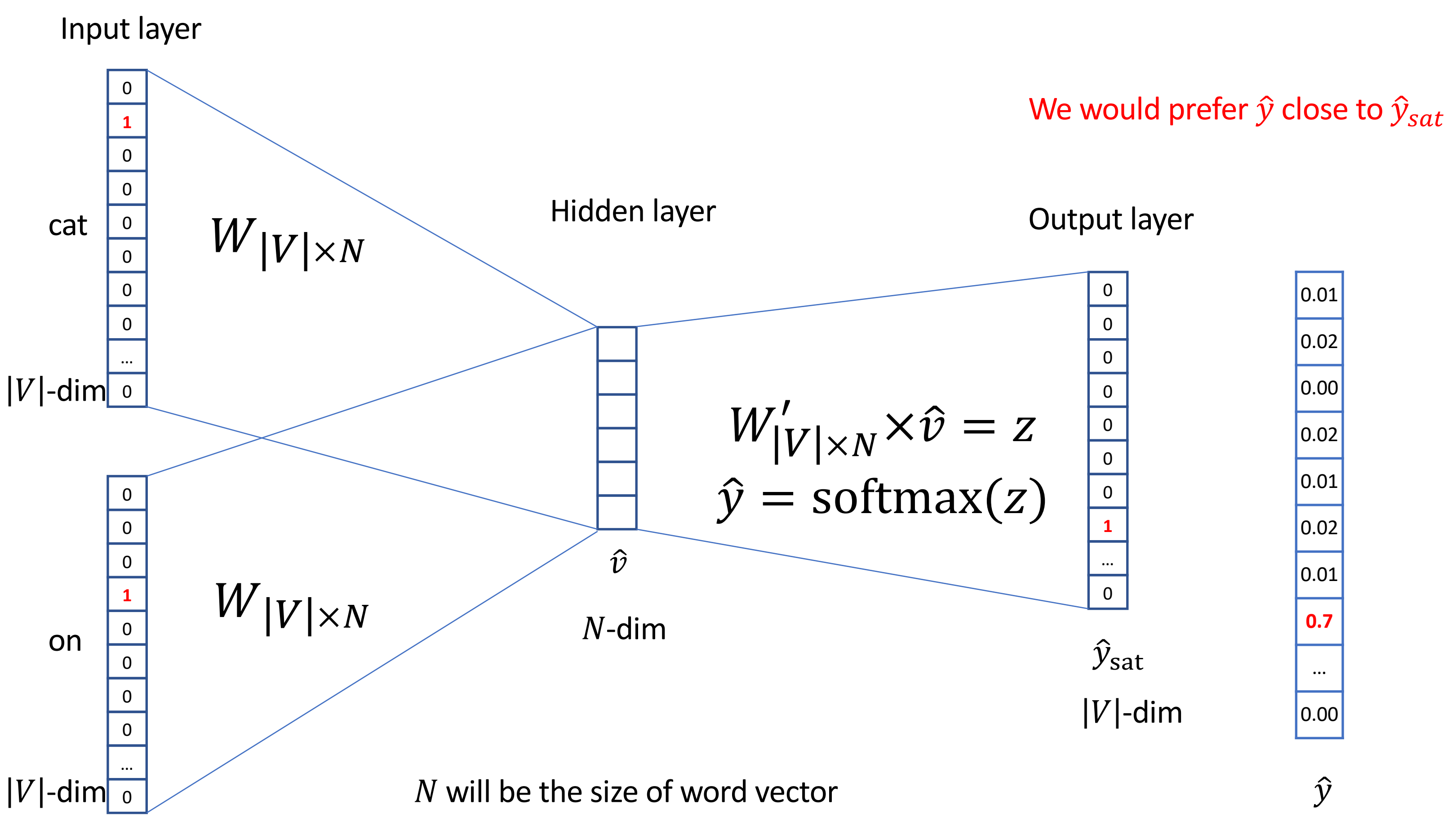


$W^T_{|V|\times N}$
$\times x_{on} = v_{on}$
0.1 2.4 1.6 1.8 0.5 0.9 … … … 3.2
0.5 2.6 1.4 2.9 1.5 3.6 … … … 6.1
0.6 1.8 2.7 1.9 2.4 2.0 … … … 1.2
1.8 2.9 … … 1.9
Input layer
x_{cat}
$|V|$-dim
$W^T_{|V|\times N} \times x_{cat} = v_{cat}$
+
$W^T_{|V|\times N} \times x_{on} = v_{on}$
x_{on}
$|V|$-dim
$\hat{v}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

Output Layer

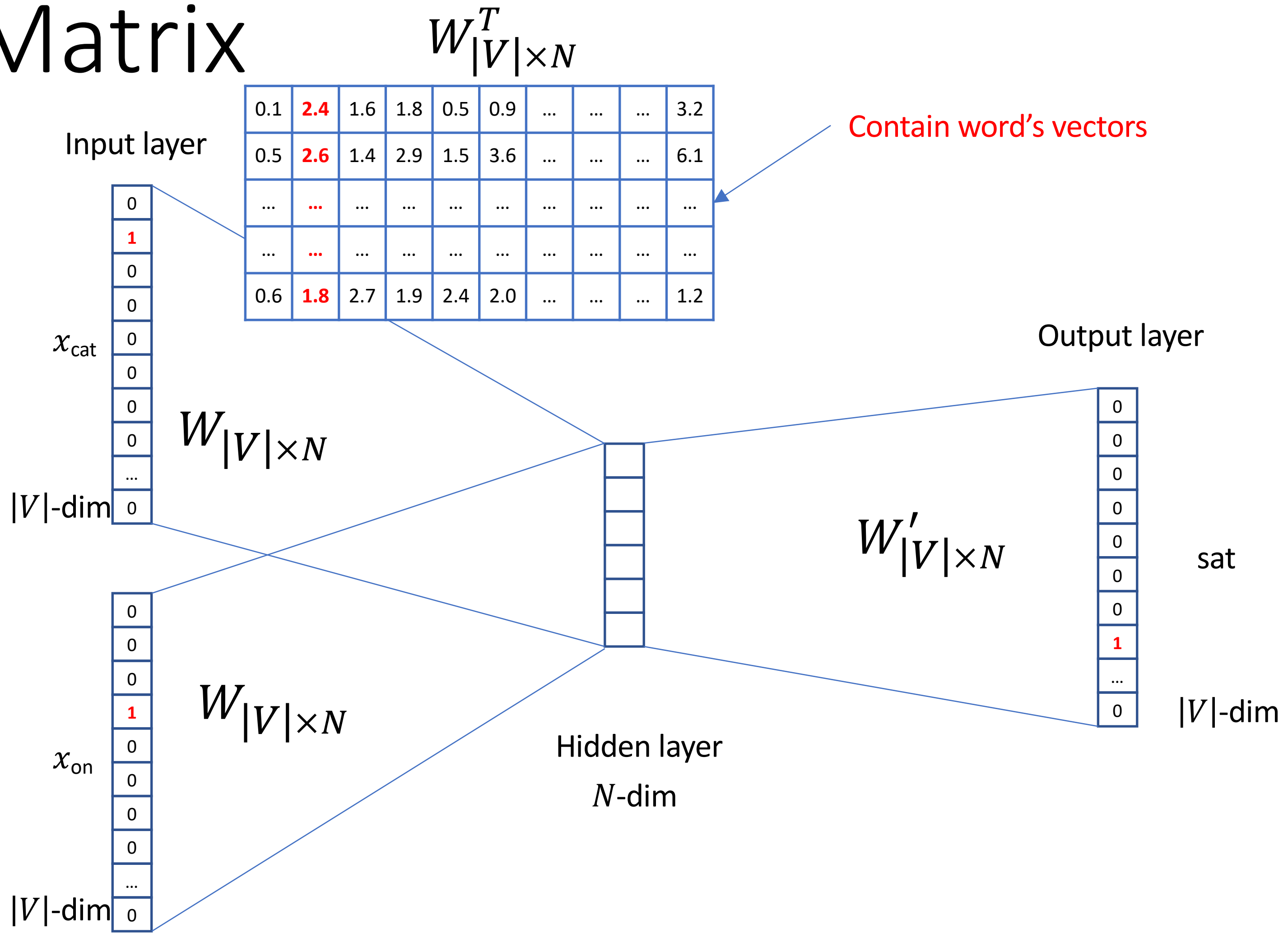


Output



Input layer
cat
$W_{|V|\times N}$
$|V|$-dim
on
$W_{|V|\times N}$
$|V|$-dim
Hidden layer
$\hat{v}$
N-dim
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
Output layer
$W'_{|V|\times N} \times \hat{v} = z$
$\hat{y} = \text{softmax}(z)$
$\hat{y}_{\text{sat}}$
$|V|$-dim
0.01
0.02
0.00
0.02
0.01
0.02
0.01
0.7
...
0.00
$\hat{y}$
N will be the size of word vector

Word's Matrix



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

Output: Similarity Computation

- Computed using the dot product of the two vectors

- To convert a similarity to a probability, use softmax

$$\Pr\left(w_k \middle| w_{k+j}, -m \leq j \leq m, j \neq 0\right) = \Pr(w_k|\hat{v}) = \frac{\exp(\hat{v}^T u_k)}{\sum_{j=1}^{|V|} \exp(\hat{v}^T u_j)}$$

for all word w_k

  - Reminder: $\hat{v}$ is the vector representation of the context and u_k is the vector representation of w_k
  - Let θ be the set of parameters u_k for all words w_k and $\hat{v}$ for all contexts $\left\{w_{k+j}, -m \leq j \leq m, j \neq 0\right\}$

- In practice, use negative sampling
  - Too many words in the denominator
  - The denominator is only computed for a few words

Word2vec: objective function

- It is the cross-entropy
- Likelihood is equivalent to cross-entropy when using one-hot encoding:

$$L(\theta) = \prod_{t=1}^{T} \Pr\left(w_t \middle| w_{t+j}, -m \leq j \leq m, j \neq 0; \theta\right)$$

$$-\log L(\theta) = -\sum_{t=1}^{T} \log \Pr\left(w_t \middle| w_{t+j}, -m \leq j \leq m, j \neq 0; \theta\right)$$

$$-\log L(\theta) = -\sum_{t=1}^{T} \sum_{x \in V} \log \Pr\left(w_t \middle| w_{t+j}, -m \leq j \leq m, j \neq 0; \theta\right) \Pr(w_t = x)$$

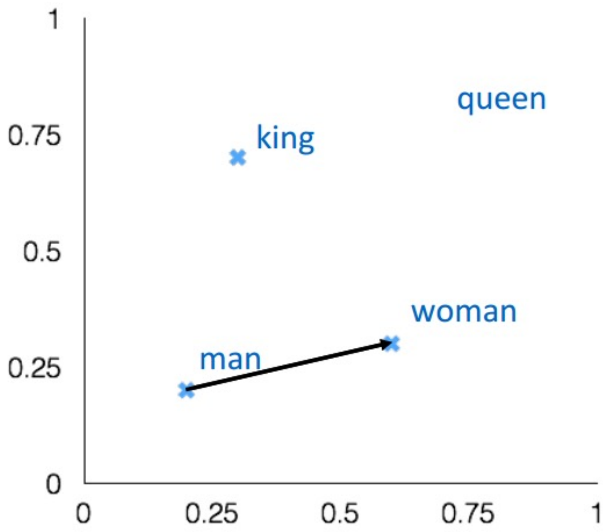
Some interesting results

Word Analogies

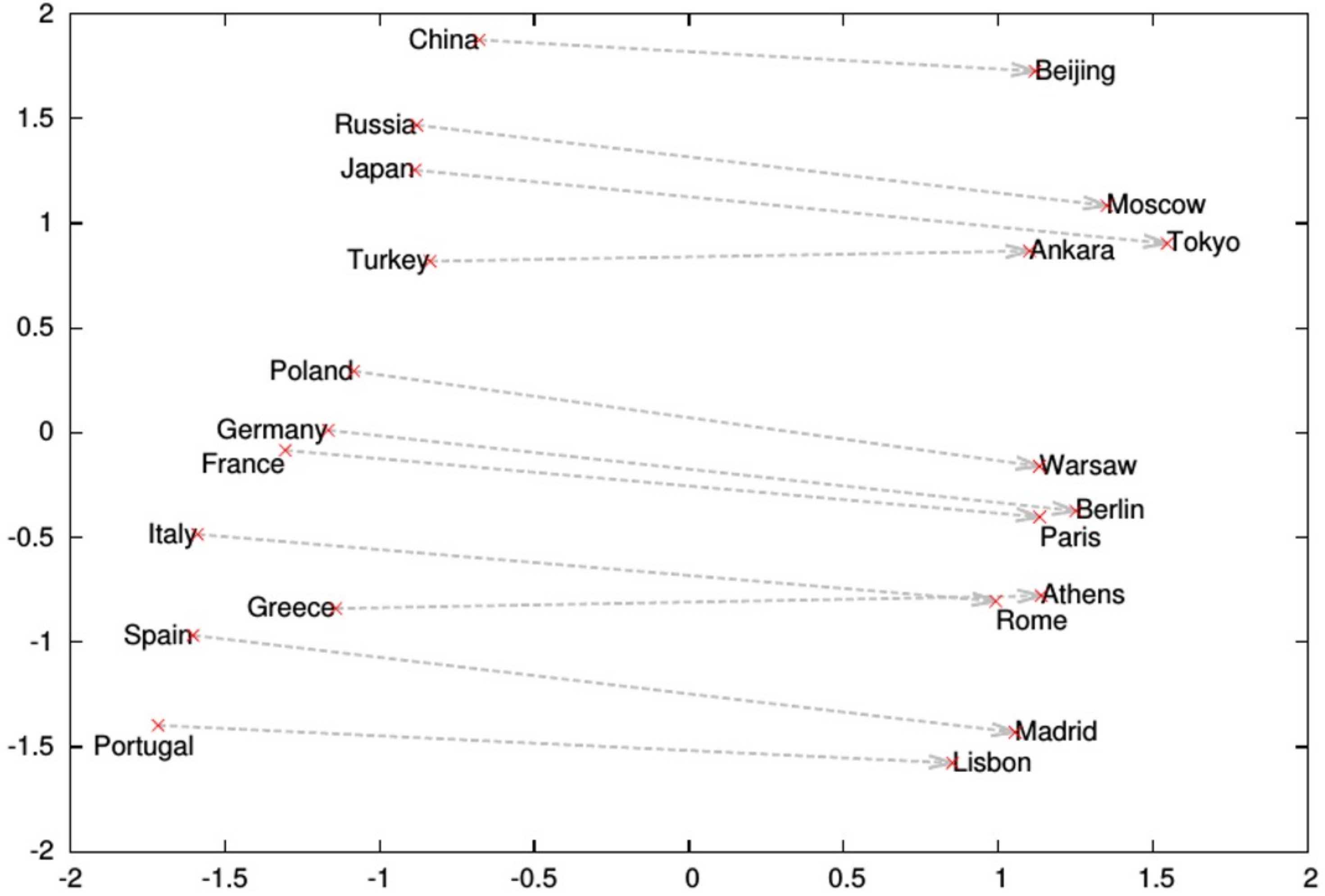
Test for linear relationships, examined by Mikolov et al. (2014)

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



Weakness of Word Embedding

- Very vulnerable, and not a robust concept
- Can take a long time to train
- Non-uniform results
- Hard to understand and visualize

Lab: The Continuous Bag of Words (CBOW) Model

- The CBOW model architecture tries to predict the current target word (the center word) based on the source context words (surrounding words).

- Considering a simple sentence, *“the quick brown fox jumps over the lazy dog”*
  - This can be pairs of *(context_window, target_word)* where if we consider a context window of size 2, we have examples like
    - *([quick, fox], brown),*
    - *([the, brown], quick),*
    - *([the, dog], lazy)*
    - and so on.

- Thus the model tries to predict the target_word based on the context_window words.

Conclusion

Conclusion

- Word embedding is a powerful preprocessing
- Used also in more advanced processing (graph processing for example)
- Embedding is not really deep learning but…
- Embedding is a very important step for deep neural network architecture dedicated to sequence analysis