output a

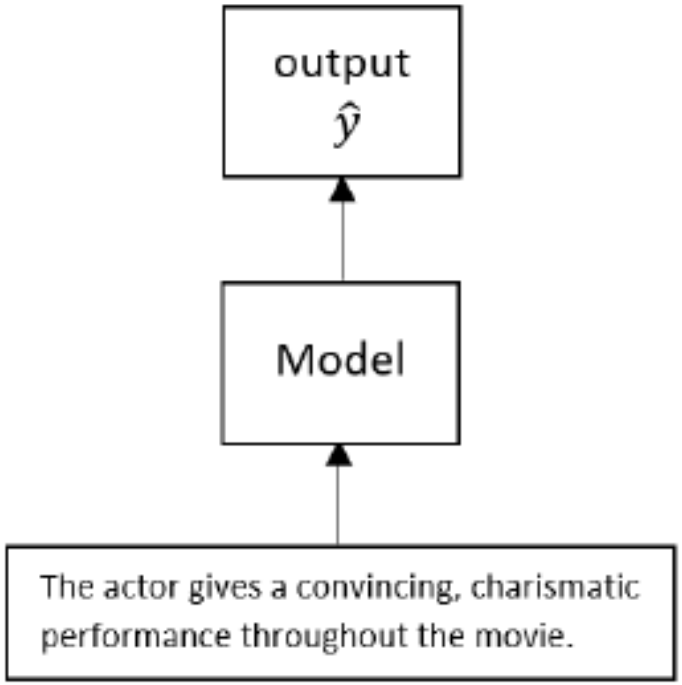
Word Representations

Introduction

The actor gives a convincing, charismatic performance throughout the movie.

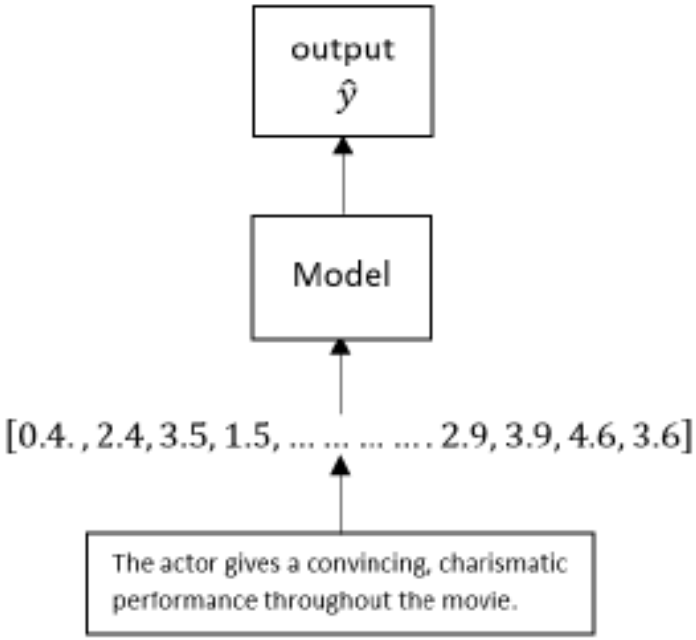
- Imagine we are given a set of words (sentence, document etc.) and we are interested in learning some function using it (e.g., $\hat{y} = sentiment(words)$)

Introduction



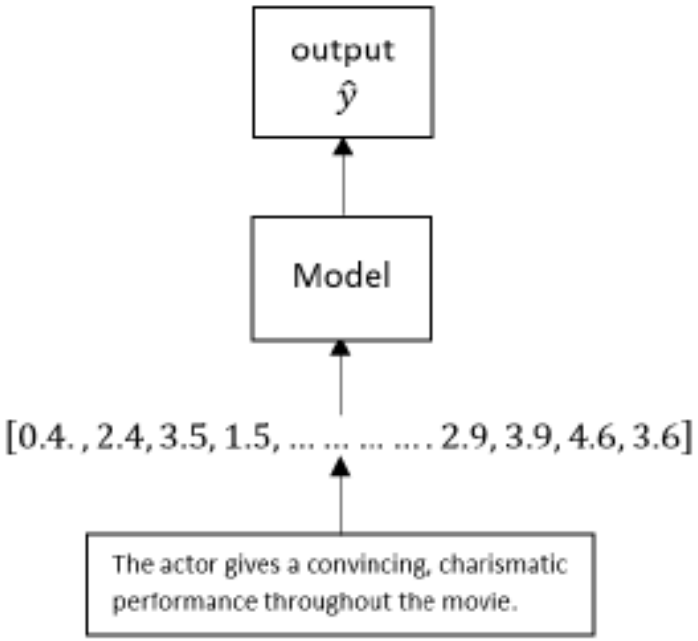
- Imagine we are given a set of words (sentence, document etc.) and we are interested in learning some function using it (e.g., $\hat{y} = sentiment(words)$)
- Goal would be to apply a model and learn the function $\hat{y} = f(x)$

Introduction



- Imagine we are given a set of words (sentence, document etc.) and we are interested in learning some function using it (e.g., $\hat{y} = sentiment(words)$)
- Goal would be to apply a model and learn the function $\hat{y} = f(x)$
- convert each input word to a vector **x** (mathematical representation)

Introduction



- Imagine we are given a set of words (sentence, document etc.) and we are interested in learning some function using it (e.g., $\hat{y} = sentiment(words)$)
- Goal would be to apply a model and learn the function $\hat{y} = f(x)$
- convert each input word to a vector **x** (mathematical representation)
- **How can we represent words using vectors?**

One-hot vectors (simple representation)

Corpus

- The actor gives a convincing performance.
- The movie is very powerful.
- The movie was awful.

$\mathcal{V}$ = [the, actor, gives, a, convincing performance, movie, is, very, powerful, was, awful]

- Given a corpus, consider the set $\mathcal{V}$ of all unique words in the input (all sentences or documents)
- $\mathcal{V}$ is called the **vocabulary** of the corpus
- Each word in $\mathcal{V}$ needs a representation.

one-hot vector representation

the :	[1	0	0	0	0	0	0	0	0	0	0	0]
actor :	[0	1	0	0	0	0	0	0	0	0	0	0]
awful :	[0	0	0	0	0	0	0	0	0	0	0	1]

$dimension = \mathbb{R}^{12}$, $\mathcal{V} = 12$

One-hot vector representation - problems

cat :	[1	0	0	0	0	0	0	0	0	0	0	0]
dog :	[0	0	0	0	0	1	0	0	0	0	0	0]
car :	[0	0	0	0	0	0	0	0	0	0	0	1]

- representation size increases significantly with the increase in the size of the corpus. (e.g., 50K for PTB, 13M for Google 1T corpus)

One-hot vector representation - problems

cat :	[1	0	0	0	0	0	0	0	0	0	0	0]
dog :	[0	0	0	0	0	1	0	0	0	0	0	0]
car :	[0	0	0	0	0	0	0	0	0	0	0	1]

- representation size increases significantly with the increase in the size of the corpus. (e.g., 50K for PTB, 13M for Google 1T corpus)
- results in sparse representation

One-hot vector representation - problems

cat :	[1	0	0	0	0	0	0	0	0	0	0	0]
dog :	[0	0	0	0	0	1	0	0	0	0	0	0]
car :	[0	0	0	0	0	0	0	0	0	0	0	1]

- representation size increases significantly with the increase in the size of the corpus. (e.g., 50K for PTB, 13M for Google 1T corpus)
- results in sparse representation
- each vector is equidistant from every other vector.

$$euclid\_dist(cat, dog) = \sqrt{2}; euclid\_dist(cat, car) = \sqrt{2}$$
$$cosine\_sim(cat, dog) = 0; cosine\_sim(cat, car) = 0$$

- ideally, *cat* and *dog* should be represented closer compared to *cat* and *car*

Why similarity between words matter?

- if we can learn:

  "fast" is similar to "rapid"

  "tall" is similar to "height"

- useful for **question answering task**

  Question: "How tall is Mount Everest?"

  Candidate answer: "The official height of Mount Everest is 29029 feet".

Why similarity between words matter?

Plagiarism Detection

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand

Do words have meanings?

Not really.

They just borrow meanings from their neighbours

Distributional Hypothesis

J. R. Firth

"You shall know a word by the company it keeps"

Image credit: www.odit.org

Quiz

- X is a device that is easy to carry around, you can speak using X, watch the Internet. It was manufactured by Apple. What could be X?
  - a dog
  - an airplane
  - an iPhone
  - a banana

But is that really true?

- Don't dictionaries define the meanings of words?
  - Dictionaries define the meanings of words using other words, in a recursive manner.
- Distributional hypothesis provides us with a practical method to learn the meanings of words using text corpora
- Distributional semantic representations have been successfully used in numerous NLP tasks reporting state-of-the-art performances.

  Therefore, it must be correct.

Two approaches ...

- Distributional Semantic Representations
  - Use the set of words that co-occur with X to represent the meaning of X
  - Sparse and high-dimensional
  - Classical approach for semantic representations
- Distributed Semantic Representations
  - Learn representations that can accurately predict the words that appear in the same context as X.
  - Limited dimensionality(10~1000)
  - Low dimensional and dense
  - Deep learning (to be precise representation learning) methods have been used
  - A more recent/modern approach

Two approaches ...

- Distributional Semantic Representations
  - Use the set of words that co-occur with X to represent the meaning of X
  - Sparse and high-dimensional
  - Classical approach for semantic representations
- Distributed Semantic Representations
  - Learn representations that can accurately predict the words that appear in the same context as X.
  - Limited dimensionality(10~1000)
  - Low dimensional and dense
  - Deep learning (to be precise representation learning) methods have been used
  - A more recent/modern approach

Use co-occurring words to represent meaning

Example sentences for "apple":

- S_2 : Apples are red
- S_2 : Red apples are delicious
- S_3 : Apples are produced in Washington

Use co-occurring words to represent meaning

Example sentences for "apple":

- S_2 : Apples are red
- S_2 : Red apples are delicious
- S_3 : Apples are produced in Washington

Representation for "apple":

apple = [(red,2), (delicious,1), (Washington,1), (produce,1)]

Use co-occurring words to represent meaning

Example sentences for "apple":

- S_2 : Apples are red
- S_2 : Red apples are delicious
- S_3 : Apples are produced in Washington

Example sentences for "oranges":

- S_4 : Oranges are yellow
- S_5 : Oranges are delicious
- S_6 : Oranges are produced in California

Representation for "apple":

apple = [(red,2), (delicious,1), (Washington,1), (produce,1)]

Use co-occurring words to represent meaning

Example sentences for "apple":

- S_2 : Apples are red
- S_2 : Red apples are delicious
- S_3 : Apples are produced in Washington

Representation for "apple":

apple = [(red,2), (delicious,1), (Washington,1), (produce,1)]

Example sentences for "oranges":

- S_4 : Oranges are yellow
- S_5 : Oranges are delicious
- S_6 : Oranges are produced in California

Representation for "oranges":

oranges = [(yellow,1), (delicious,1), (California,1), (produce,1)]

Use co-occurring words to represent meaning

- We have:

  apple = [(red,2), (delicious,1), (Washington,1), (produce,1)]
  oranges = [(yellow,1), (delicious,1), (California,1), (produce,1)]

- Similarity between two words can be measured using overlapping features/attributes in their respective semantic representations

  Jaccard Coefficienct =| apple AND orange | / | apple OR orange |
  sim(apple,orange) = 2/6 = 0.3333

Co-occurrence Matrix

- We can arrange the semantic representations
- We learn for all the words in a corpus as rows in a co-occurrence matrix
  - rows = semantic representation of words
  - various features/words that co-occur with words

[1.5]	red	delicious	Washington	produce	yellow	California
apple	2	1	1	1	0	0
orange	0	1	0	1	1	1

Issues of large co-occurrences

- How reliable are large co-occurrences?
- Consider Google hits (no. of pages) for,
  - (car, automobile) = 11,300,000
  - (car, apple) = 49,000,000
- apples are more similar to cars than automobiles ???
- We need proper weighting for co-occurrences (in particular when some words are very common)

Co-occurrence Weighting Measures

- combines scores of individual words, $h(x)$, $h(y)$ and co-occurrences between two words $h(x, y)$
  - weighting function $= f(h(x), h(y), h()x, y)$

- pointwise mutual information (PMI) (Church & Hanks, 1989)
  - Do words x and y co-occur more than if they were independent
  - $PMI(x, y) = \log\left(\frac{p(x,y)}{p(x),p(y)}\right) = \log\left(\frac{h(x,y)/N}{(h(x)/N)\times(h(y)/N)}\right)$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- negative values are problematic
  - Unreliable without enormous corpora
    - consider w_1 and w_2 whose probability is each 10^{-6}
    - hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}
  - replace negative PMI values by 0
  - Positive PMI (PPMI) between word1 and word2:
    $PMI(x, y) = max(\log_2 \frac{h(x,y)/N}{(h(x)/N)\times(h(y)/N)}, 0)$

PPMI - Example

Example contexts: 20 words (Brown corpus)

- equal amount of sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of clove and nutmeg,
- on board for their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened to that of
- of a recursive type well suited to programming on the **digital** computer. In finding the optimal R-stage policy from that of
- substantially affect commerce, for the purpose of gathering data and **information** necessary for the study authorised in the first section of this

PPMI - Example

Co-occurrence Matrix

	material	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pinepapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

- two words are considered to be similar if their context vectors are similar

PPMI - Example

Co-occurrence Matrix

	material	computer	data	pinch	result	sugar	$c(x)$
apricot	0	0	0	1	0	1	2
pinepapple	0	0	0	1	0	1	2
digital	0	2	1	0	1	0	4
information	0	1	6	0	4	0	11
$c(y)$	0	3	7	2	5	2	

c = count

$$PMI(x,y) = \log\left(\frac{p(x,y)}{p(x),p(y)}\right) = \log\left(\frac{h(x,y)/N}{(h(x)/N)\times(h(y)/N)}\right)$$

N = sum of all words in all contexts = 19
$h(x,y)$ = h(x=information, y=data) = 6; $h(x,y)/N = 6/19 = 0.32$
(x is the word, y is the context)
$h(x)$ = h(x=information) = 11; $h(x)/N = 11/19 = 0.58$
$h(y)$ = h(y=data) = 7; $h(y)/N = 7/19 = 0.37$

PPMI - Example

$$PMI(x,y) = \log\left(\frac{p(x,y)}{p(x),p(y)}\right) = \log\left(\frac{h(x,y)/N}{(h(x)/N)\times(h(y)/N)}\right)$$

N = sum of all words in all contexts = 19
$h(x,y)$ = h(x=information, y=data) = 6; $h(x,y)/N = 6/19 = 0.32$
(x is the word, y is the context)
$h(x)$ = h(x=information) = 11; $h(x)/N = 11/19 = 0.58$
$h(y)$ = h(y=data) = 7; $h(y)/N = 7/19 = 0.37$

Co-occurrence Matrix

	$h(x,y)$					
	computer	**data**	**pinch**	**result**	**sugar**	$h(x)$
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pinepapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
$h(y)$	0.16	0.37	0.11	0.26	0.11	

PPMI - Example

$h(x, y)$

	computer	data	pinch	result	sugar	$h(x)$
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pinepapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
$h(y)$	0.16	0.37	0.11	0.26	0.11	

$$PMI(x, y) = \log\left(\frac{p(x,y)}{p(x),p(y)}\right) = \log\left(\frac{h(x,y)/N}{(h(x)/N)\times(h(y)/N)}\right)$$

$$PMI(\text{information, data}) = \log_2(0.32/(0.37 \times 0.58)) = 0.57$$

$h(x, y)$

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pinepapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

PPMI - Example - add-2 smoothing

without smoothing

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pinepapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

with add-2 smoothing

	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pinepapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

$h(x, y)$ with add-2 smoothing

	computer	data	pinch	result	sugar	h(x)
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pinepapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.10	0.03	0.36
information	0.05	0.14	0.03	0.10	0.03	0.36
h(y)	0.19	0.25	0.17	0.22	0.17	

PPMI - Example - add-2 smoothing

$h(x, y)$ without smoothing

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pinepapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

$h(x, y)$ with add-2 smoothing

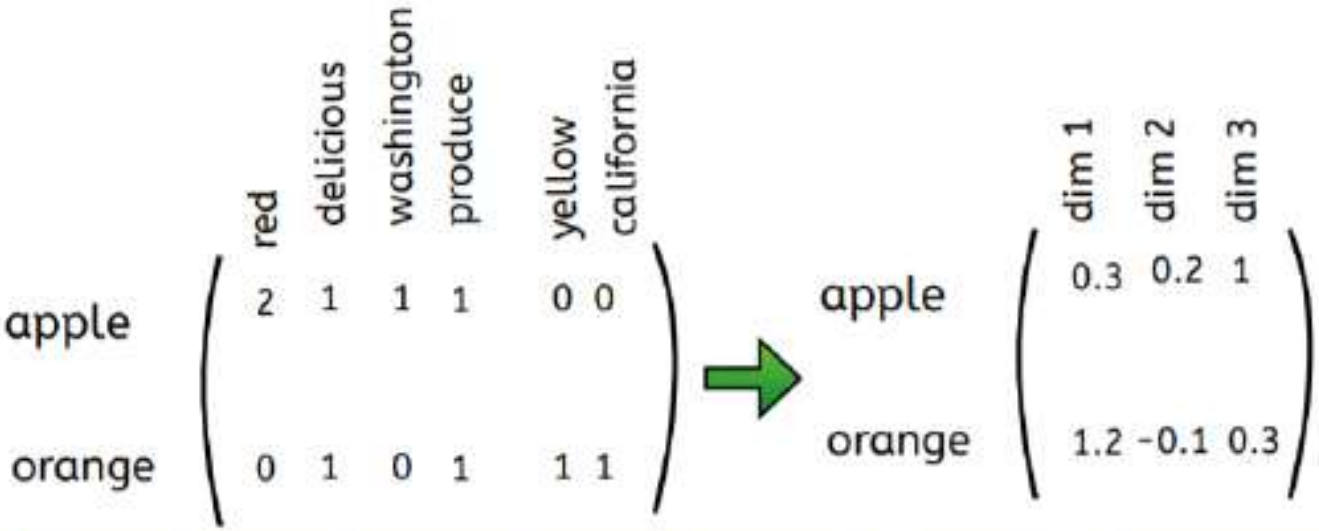
	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pinepapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.57	0.00	0.37	0.00

Issues of zero co-occurrences

- Some words never co-occur even in very large corpora.
- If words x and y do not co-occur, then
  - x and y might not be related OR
  - it could be that our corpus was too small and we did not observe their co-occurrences.
- If we have co-occurrence-based semantic representations that have many zeros, then we will have many zero similarity scores, which is not good.
- How can we reduce the number of zeros?

Dimensionality reduction / Low-dimensional projection

- We can reduce the number of features (columns) thereby collapsing similar dimensions.
- This process will reduce the number of zeros.



Note that the number of words (rows) does not change. Therefore, we have a representation for all the words that appear in the original co-occurrence matrix.

Dimensionality reduction methods

- Singular Value Decomposition (SVD)
  - $A = UDV^T$, use U or UD as the lower-dimensional projection
- Principal Component Analysis (PCA)
  - See the lecture on dimensionality reduction
- non-negative matrix factorization (NMF)
  - $A = WH$
- There are many libraries that implement the above (and many more)
  - In Python: numpy, scipy, sklearn

Selecting context

- How to select the "context"?
  - sentence-level co-occurrences
  - proximity window (n words before/after)
  - Words that are connected via dependency relations

Two approaches ...

- Distributional Semantic Representations
  - Use the set of words that co-occur with X to represent the meaning of X
  - Sparse and high-dimensional
  - Classical approach for semantic representations
- Distributed Semantic Representations
  - Learn representations that can accurately predict the words that appear in the same context as X.
  - Limited dimensionality(10~1000)
  - Low dimensional and dense
  - Deep learning (to be precise representation learning) methods have been used
  - A more recent/modern approach

Continuous bag-of-word and skip-gram

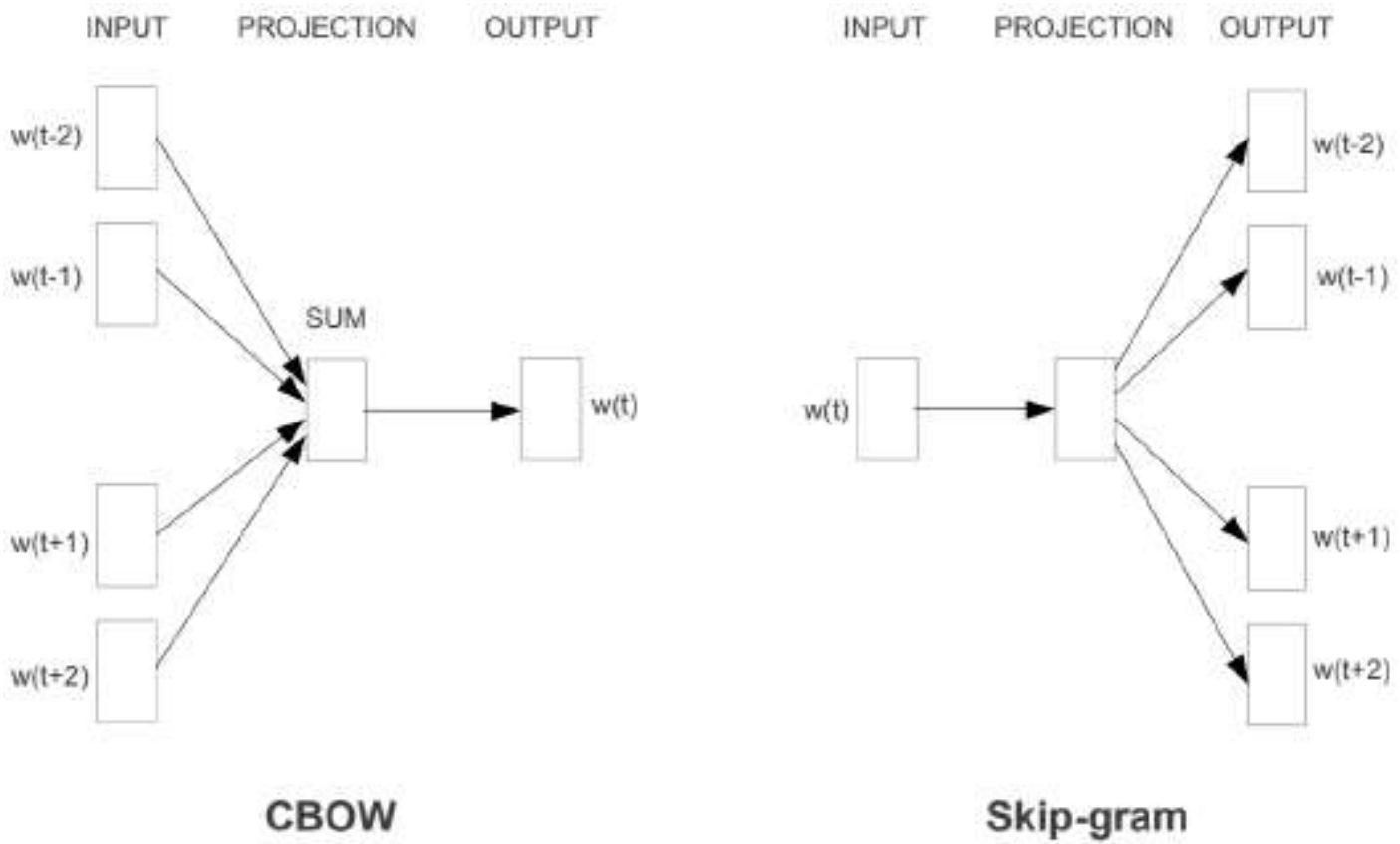


Figure: Source: Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov et al., 2013

CBOW Model

- given context (window of words), predict words

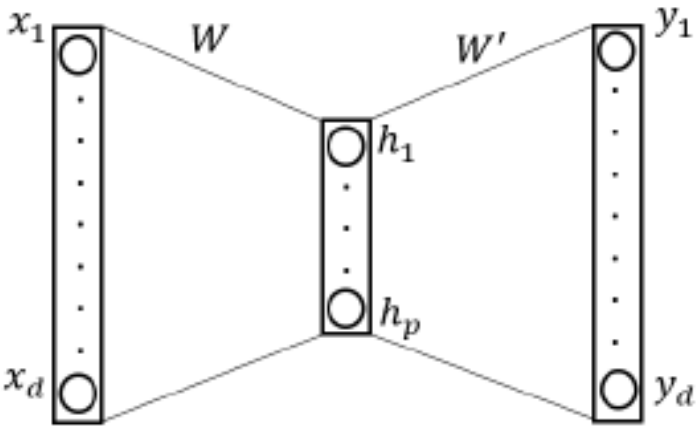


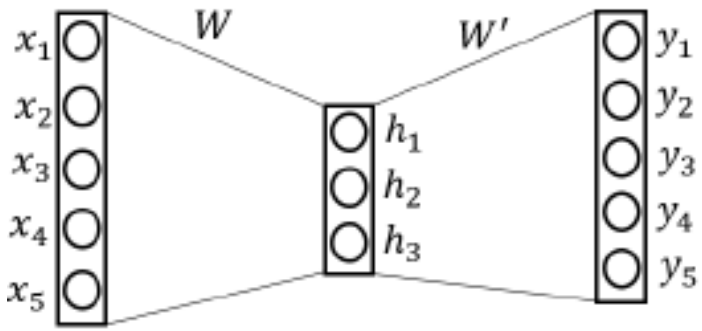
Figure: Architecture of CBOW model

- input: $\mathbf{x} \in \mathbb{R}^d$, $d = \mathcal{V}$, size of the vocabulary
- hidden layer: $\mathbf{h} = W^T x$
- output: $\mathbf{u} = W'^T \mathbf{h}$

  $\mathbf{y} = \phi(\mathbf{u}); \phi =$ activation function

  $y_i = \frac{exp(v'^T_w, v_c)}{\sum_{w=1}^{\mathcal{V}} exp(v'^T_w, v_c)}$
- parameters: $\{v_c \in W, v_w \in W'\}$

CBOW Model with one word in context



Toy corpus:

- The dog ran
- The cat screamed

$\mathcal{V} = [\text{the, dog, ran, can, screamed}]$

One-hot vector representation:

the	$[1, 0, 0, 0, 0]$
dog	$[0, 1, 0, 0, 0]$
ran	$[0, 0, 1, 0, 0]$
cat	$[0, 0, 0, 1, 0]$
screamed	$[0, 0, 0, 0, 1]$

CBOW Model with one word in context

- learn relation between *cat* and *screamed*, compute $P(word \mid context) = P(w \mid c)$
- input word would be *cat*
- hidden vector $\mathbf{h} = W^T\mathbf{x}$

$$\mathbf{h}_{3\times 1} = \begin{bmatrix} [1.5]h_1 \\ h_2 \\ h_3 \end{bmatrix}_{3\times 1} = \begin{bmatrix} [1.5]w_{11} & w_{21} & w_{31} & w_{41} & w_{51} \\ w_{12} & w_{22} & w_{32} & w_{42} & w_{52} \\ w_{13} & w_{23} & w_{33} & w_{43} & w_{53} \end{bmatrix}_{3\times 5} \begin{bmatrix} [1.5]0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}_{5\times 1}$$

$$\mathbf{h}_{3\times 1} = \begin{bmatrix} [1.5]w_{11}\times 0 + w_{21}\times 0 + w_{31}\times 0 + w_{41}\times 1 + w_{51}\times 0 + \\ w_{12}\times 0 + w_{22}\times 0 + w_{32}\times 0 + w_{42}\times 1 + w_{52}\times 0 + \\ w_{13}\times 0 + w_{23}\times 0 + w_{33}\times 0 + w_{43}\times 1 + w_{53}\times 0 + \end{bmatrix}_{3\times 1} = \begin{bmatrix} [1.5]w_{41} \\ w_{42} \\ w_{43} \end{bmatrix}$$

CBOW Model with one word in context

Remember, we are computing $\mathbf{h} = W^T\mathbf{x}$ (hidden representation of context)

$$\begin{bmatrix} [1.5]w_{11} \times 0 + w_{21} \times 0 + w_{31} \times 0 + w_{41} \times 1 + w_{51} \times 0 + \\ w_{12} \times 0 + w_{22} \times 0 + w_{32} \times 0 + w_{42} \times 1 + w_{52} \times 0 + \\ w_{13} \times 0 + w_{23} \times 0 + w_{33} \times 0 + w_{43} \times 1 + w_{53} \times 0 + \end{bmatrix} = \begin{bmatrix} [1.5]w_{41} \\ w_{42} \\ w_{43} \end{bmatrix}$$

$$W = \begin{bmatrix} [1.5]w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{bmatrix}$$

- note, the fourth column of W^T is $\mathbf{h}$
- note the fourth column of W^T is the fourth row of W
- in the toy example, $W^T\mathbf{x}$ is the 4th row of W
- to generalise, $W^T\mathbf{x}$ is the kth row of W

let us denote

$$h = W^T\mathbf{x} = v_c \tag{1}$$

v_c can also be regarded as the context vector in W

CBOW Model with one word in context

from hidden to output layer

we compute $\mathbf{u} = W'^{T}\mathbf{h}$

$$\mathbf{u} = \begin{bmatrix} [1.5]w'_{11} & w'_{21} & w'_{31} \\ w'_{12} & w'_{22} & w'_{32} \\ w'_{13} & w'_{23} & w'_{33} \\ w'_{14} & w'_{24} & w'_{34} \\ w'_{15} & w'_{25} & w'_{35} \end{bmatrix} \begin{bmatrix} [1.5]h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

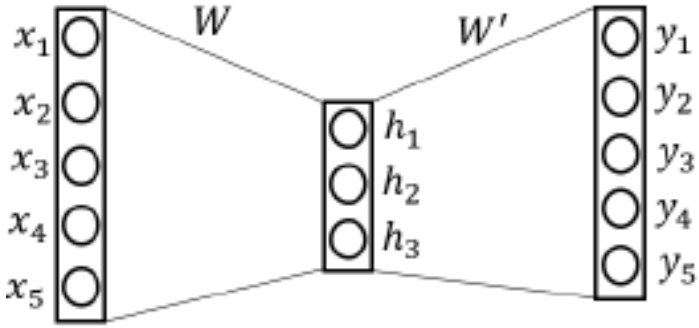
$u_i = {W_{:i}}'^{T}\mathbf{h}$

${W_{:i}}'^{T}$ is the ith row of W'^{T}, which is the ith column in W'

let us denote

$$ {W_{:i}}'^{T} = v_w \qquad (2)$$

v_w is also the word vector in W'



$$W' = \begin{bmatrix} [1.5]w'_{11} & w'_{12} & w'_{13} & w'_{14} & w'_{15} \\ w'_{21} & w'_{22} & w'_{23} & w'_{24} & w'_{25} \\ w'_{31} & w'_{32} & w'_{33} & w'_{34} & w'_{35} \end{bmatrix}$$

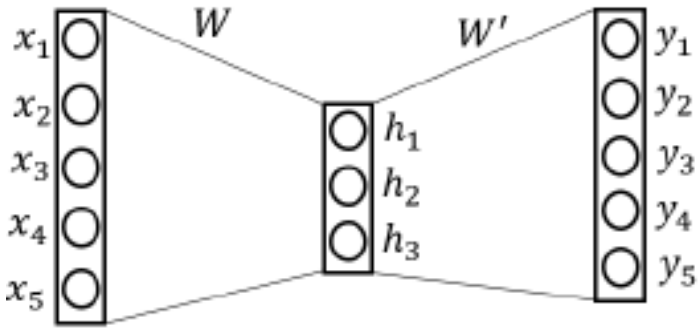
CBOW Model with one word in context

- $y_i = \phi(u_i)$
- *softmax* activation is used to compute probability for each y_i

$$y_i = \frac{exp(u_i)}{\sum_{i=1}^{\mathcal{V}} exp(u_i)}$$

$$y_i = \frac{exp({W_{:i}}'^T \mathbf{h})}{\sum_{i=1}^{\mathcal{V}} exp({W_{:i}}'^T \mathbf{h})}$$

- from (1) and (2): $y_i = \frac{exp(v_w^T v_c)}{\sum_{w=1}^{\mathcal{V}} exp(v_w^T v_c)}$
- To train the network, we learn v_w, v_c
- objective function: since we predict probabilities at each y_i, we can use maximum likelihood to maximise the likelihood of observing the data



CBOW Model with one word in context

- maximise log likelihood of the model

  parameters: $\theta = \{v_c, v_w\}$

  likelihood: $L(\theta) = \prod_{w \in V} P(w \mid c; \theta)$

  $L = \log(L(\theta)) = \sum_{w \in V} \log P(w \mid c; \theta)$

  $L = \sum_{w \in V} \log \frac{exp(v_w^T v_c)}{\sum_{w=1}^{\mathcal{V}} exp(v_w^T v_c)}$

  $L = \sum_{w \in V} v_w^T v_c - \log \sum_{w=1}^{\mathcal{V}} exp(v_w^T v_c)$
- compute $\frac{\partial L}{\partial v_w}$, $\frac{\partial L}{\partial v_c}$ to maximise the likelihood
- consider $\frac{\partial L}{\partial v_w} = v_c - \frac{1}{\sum_{w=1}^{\mathcal{V}} exp(v_w^T v_c)} exp(v_w^T v_c) v_c$
- $\frac{\partial L}{\partial v_w} = v_c - P(w \mid c) v_c$
- $\frac{\partial L}{\partial v_w} = v_c(1 - P(w \mid c))$

CBOW Model with one word in context

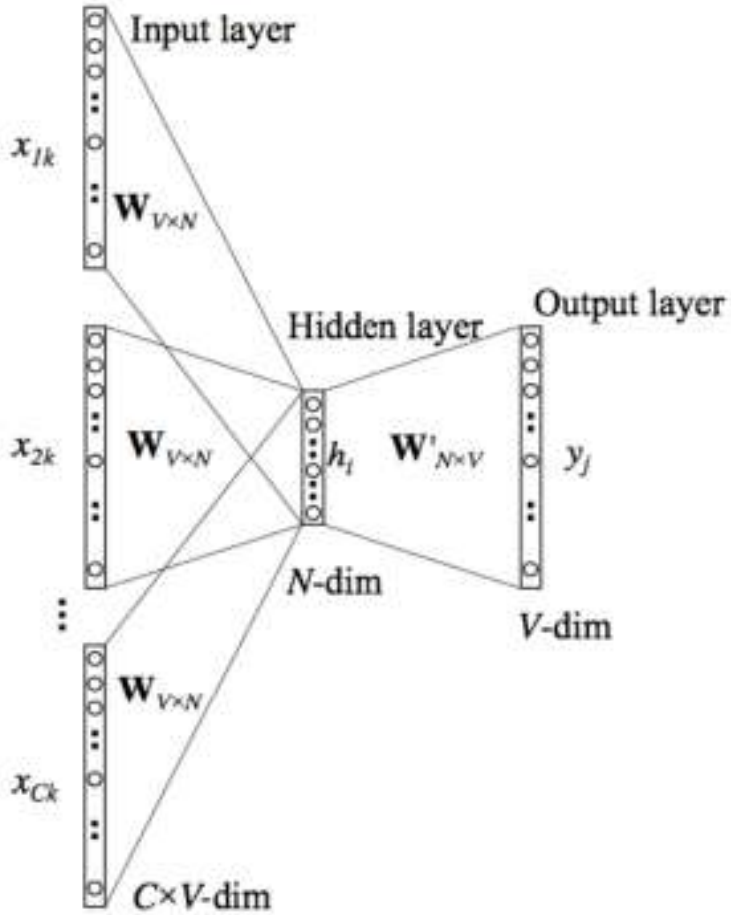
- update gradients using gradient descent.

  $v_w = v_w - \eta \frac{\partial L}{\partial v_w}$

  $v_w = v_w - \eta[v_c(1 - P(w \mid c))]$

- similarly $\frac{\partial L}{\partial v_c}$ can be computed to update v_c.

CBOW Model- multiple context words



Continuous bag-of-word and skip-gram

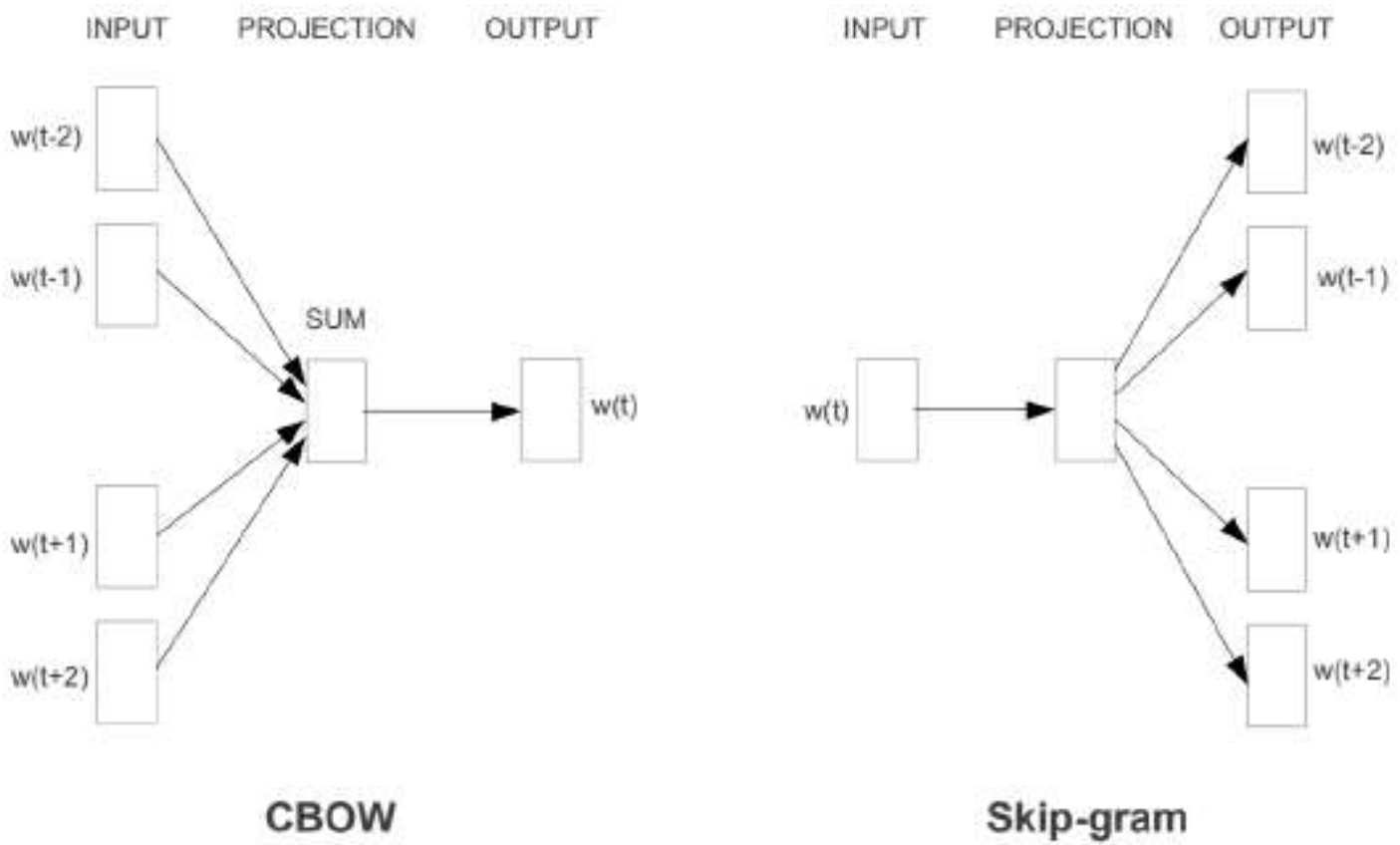
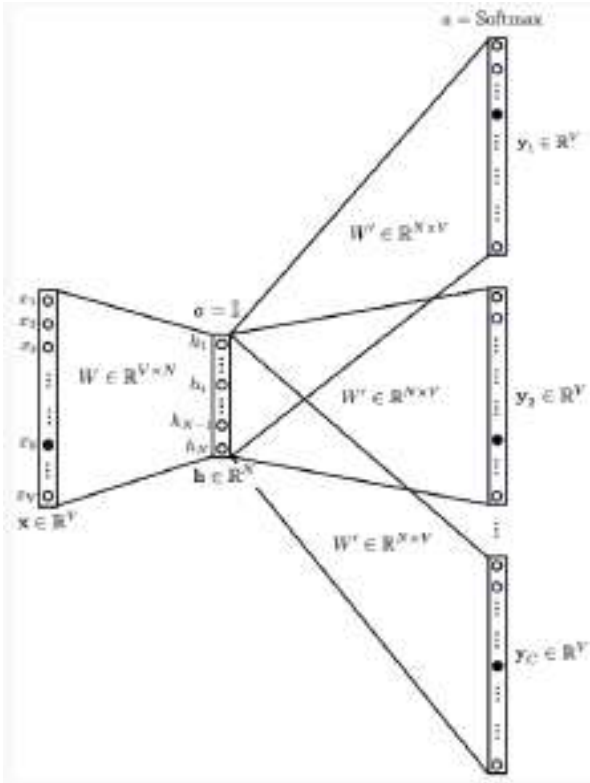


Figure: Source: Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov et al., 2013

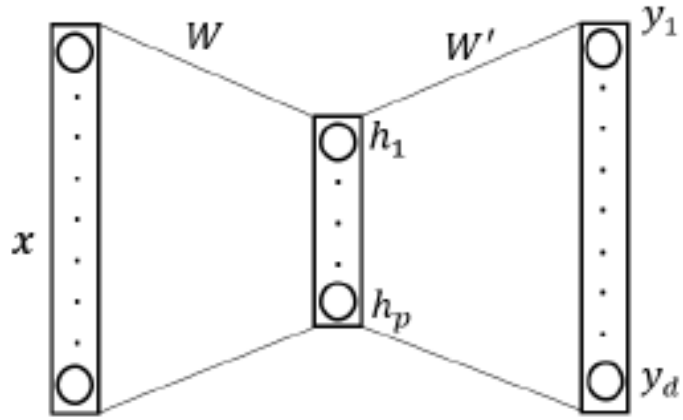
Skip-gram model

- architecture similar to CBOW model
- given input word (center word) predict context words
- CBOW model: one set of output probabilities; Skip-gram model: c set of output probabilities.
- output: $y_i = \frac{exp(v_w^T v_c)}{\sum_{w=1}^{\mathcal{V}} exp(v_w^T v_c)}$
- computationally expensive to compute softmax function ($10^5 - 10^7 terms$).
- solution: negative sampling



Negative Sampling

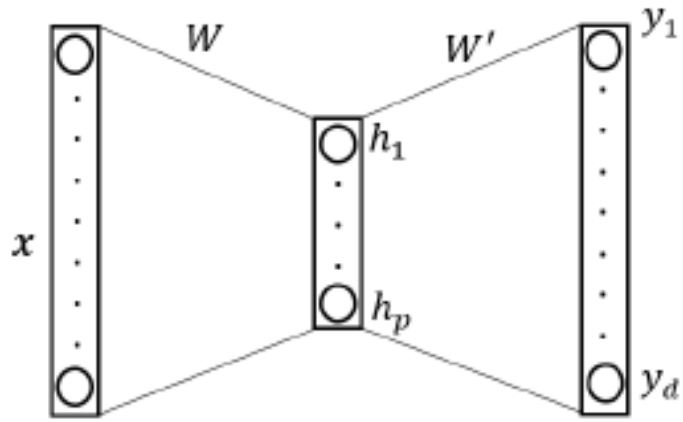
- turn unsupervised learning into supervised learning.
- create two sets of word pairs:
  - $\mathcal{D}$: set of all word pairs (w, c) present in the text
  - $\mathcal{D}'$: set of all word pairs (w, c) that are not present in the text
- we can now define:
  - $z = 1$ if $(w, c) \in \mathcal{D}$
  - $z = 0$ if $(w, c) \in \mathcal{D}'$



Negative Sampling

- instead, we can compute probabilities using logistic regression
  - $P(z = 1 \mid w, c)$ is the probability that (w, c) is in $\mathcal{D}$
  - $P(z = 0 \mid w, c)$ is the probability that (w, c) is in $\mathcal{D}'$
- we have turned the task into classification problem
- we can apply logistic function to compute probabilities

  thus,
  $P(z = 1 \mid w, c) = \sigma(v_c^T v_w) = \frac{1}{1+e^{-v_c^T v_w}}$



Negative Sampling

- if, $P(z = 1 \mid w, c) = \frac{1}{1+e^{-v_c^T v_w}}$
- then $P(z = 0 \mid w, c) = 1 - \frac{1}{1+e^{-v_c^T v_w}}$

  $P(z = 0 \mid w, c) = \frac{e^{-v_c^T v_w}}{1+e^{-v_c^T v_w}}$

  $P(z = 0 \mid w, c) = \frac{1}{1+e^{v_c^T v_w}} = \sigma(-v_c^T v_w)$
- probabilities can be written in a generalised form:

  $P(z \mid w, c; \theta) = \left(\frac{1}{1+e^{-v_c^T v_w}}\right)^z \left(\frac{1}{1+e^{v_c^T v_w}}\right)^{(1-z)}$, where $\theta = \{v_c, v_w\}$

Negative Sampling

- probabilities can be written in a generalised form:

  $P(z \mid w, c; \theta) = \left(\frac{1}{1+e^{-v_c^T v_w}}\right)^z \left(\frac{1}{1+e^{v_c^T v_w}}\right)^{(1-z)}$, where $\theta = \{v_c, v_w\}$

- to find v_c, v_w, we can use maximum likelihood for all (w, c) pairs in $\mathcal{D}, \mathcal{D}'$

  thus: $L(\theta) = \prod_{(w,c)\in\mathcal{D},\mathcal{D}'} \left(\left(\frac{1}{1+e^{-v_c^T v_w}}\right)^z \left(\frac{1}{1+e^{v_c^T v_w}}\right)^{(1-z)}\right)$

- to find the log likelihood:

  $\log(L(\theta)) = \sum_{(w,c)\in\mathcal{D},\mathcal{D}'} z \log\left(\frac{1}{1+e^{-v_c^T v_w}}\right) + (1-z)\log\left(\frac{1}{1+e^{v_c^T v_w}}\right)$

Negative Sampling

- to further simplify:

$$\log(L(\theta)) = \sum_{(w,c)\in\mathcal{D}} \log\left(\frac{1}{1+e^{-v_c^T v_w}}\right) + \sum_{(w,c)\in\mathcal{D}'} \log\left(\frac{1}{1+e^{v_c^T v_w}}\right)$$

$$\log(L(\theta)) = \sum_{(w,c)\in\mathcal{D}} \log\sigma(v_c^T v_w) + \sum_{(w,c)\in\mathcal{D}'} \log\sigma(-v_c^T v_w)$$

- loss function :

$$\log\sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i\sim P_n(w)}\left[\log\sigma(-{v'_{w_i}}^\top v_{w_I})\right]$$

Hierarchical Softmax

- provides computationally efficient approximation of the full softmax
- main advantage is that instead of evaluating W output nodes to obtain probability distribution, we evaluate about $\log_2(W)$

- the depth of the binary tree depends on the vocabulary
- the path to any given word in the vocabulary is known
- length to reach any word is $\log_2(\mathcal{V})$

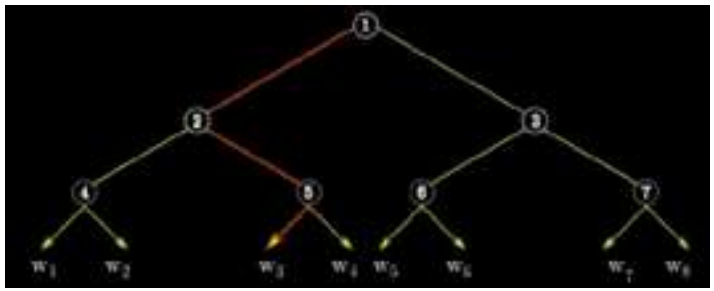


Figure: Balanced Binary Tree

Hierarchical Softmax

- We can arrange the words using:
  - random order
  - WordNet based rules
  - frequency

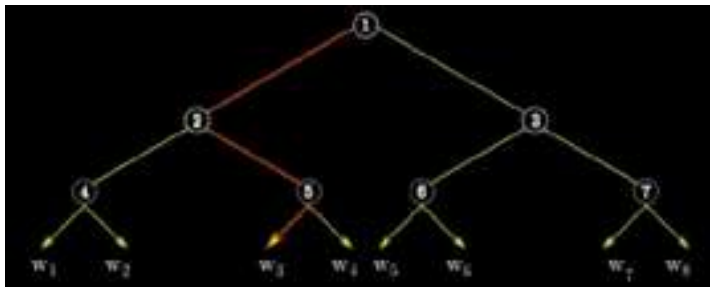
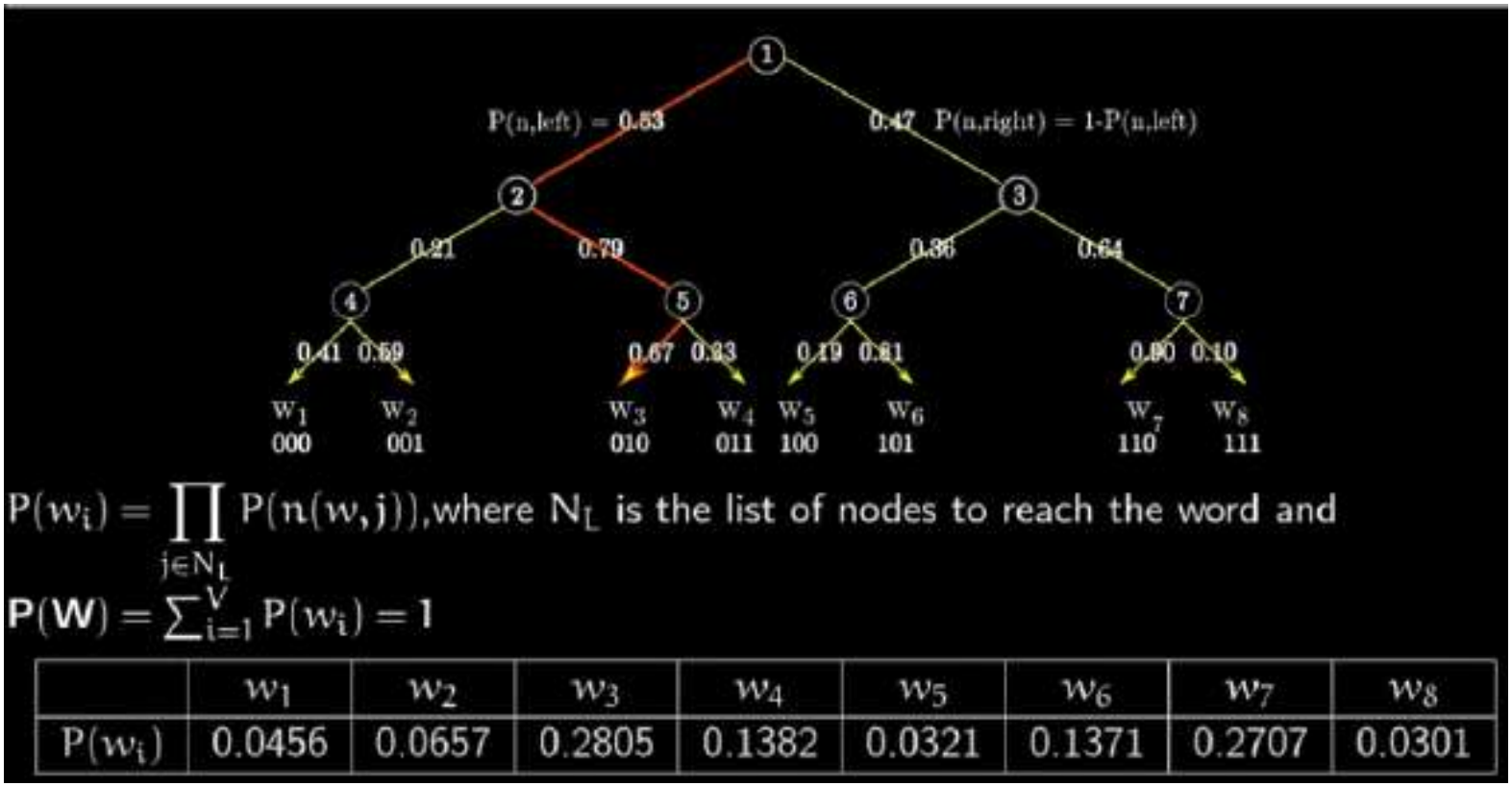


Figure: Balanced Binary Tree

Hierarchical Softmax

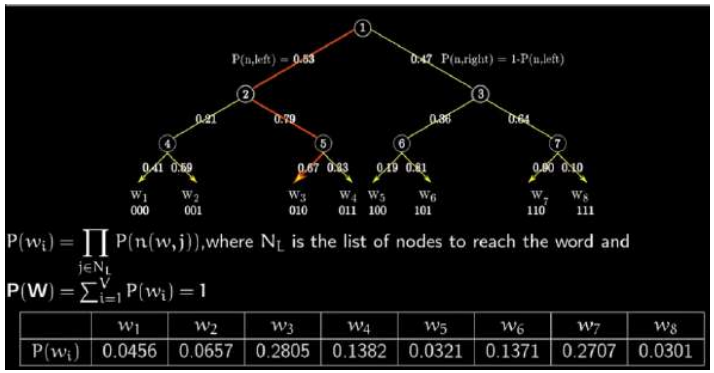
- provides a well-defined multinomial distribution among all words



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
$P(w_i)$	0.0456	0.0657	0.2805	0.1382	0.0321	0.1371	0.2707	0.0301

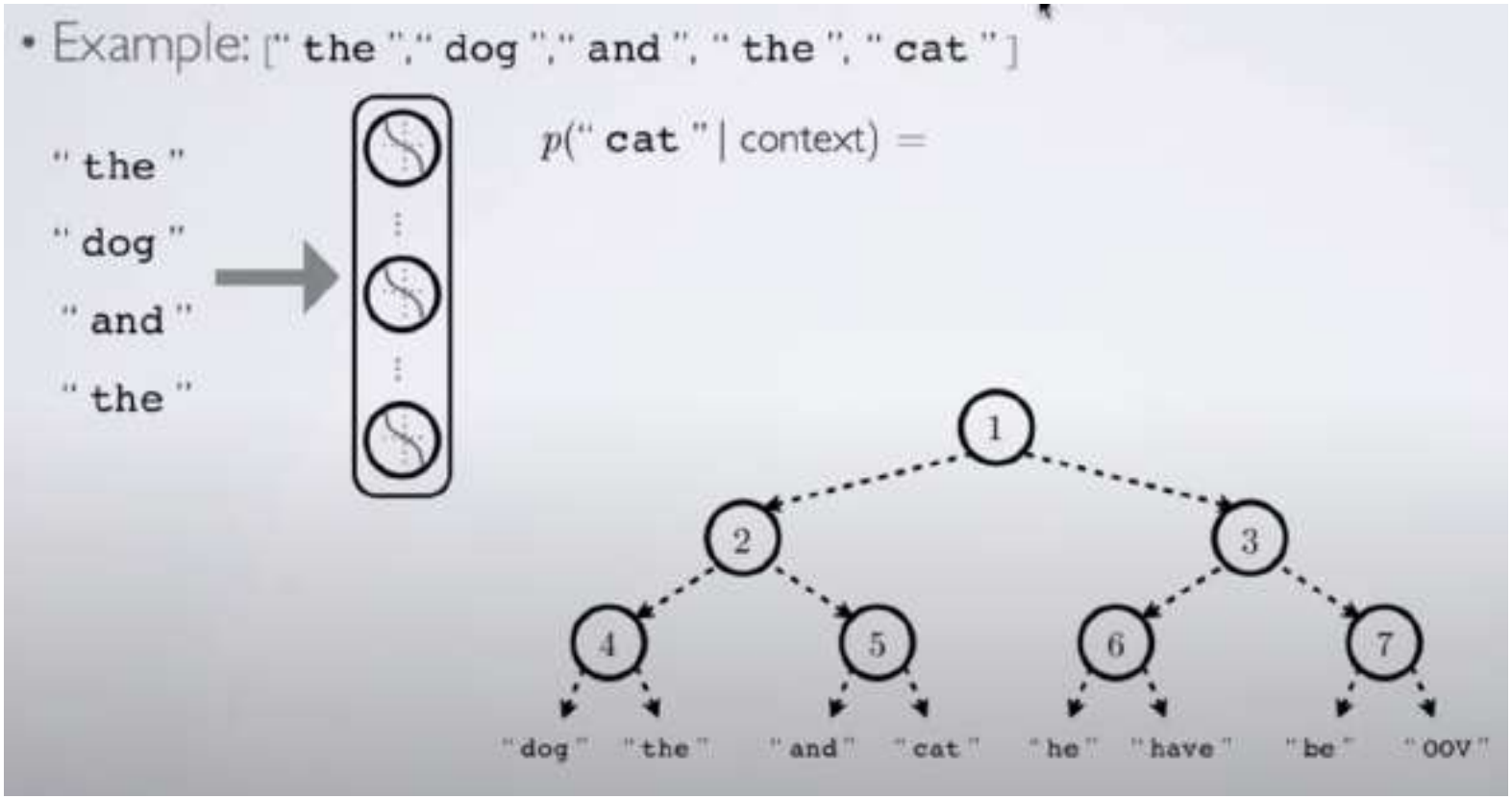
Hierarchical Softmax

- Advantages
  - a flat hierarchy can be decomposed into a binary tree
  - The path to each word (leaf) is unique
  - Each intermediate node provides the relative probabilities of its child nodes
  - every leaf represents the probability of the word

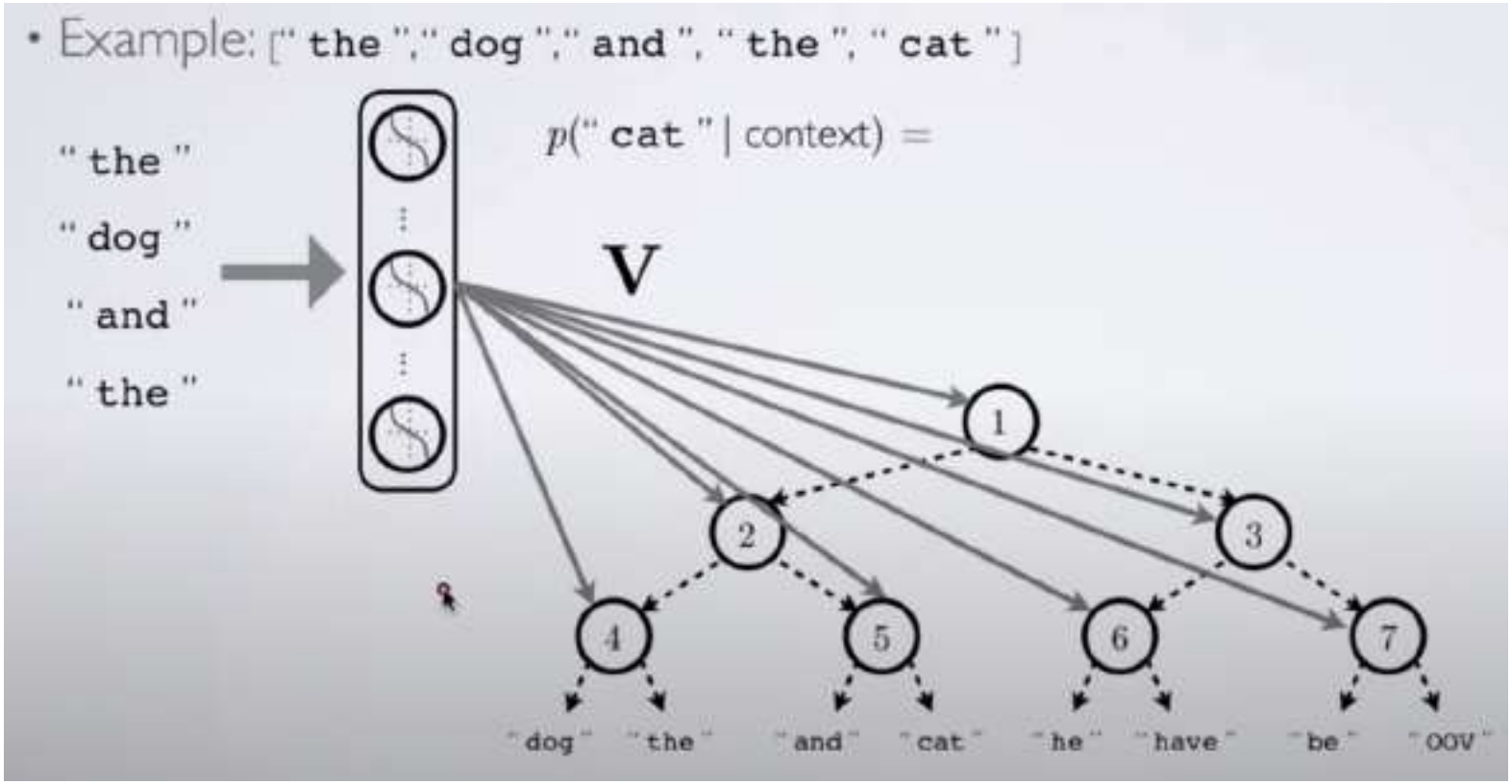


	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
$P(w_i)$	0.0456	0.0657	0.2805	0.1382	0.0321	0.1371	0.2707	0.0301

Hierarchical Softmax

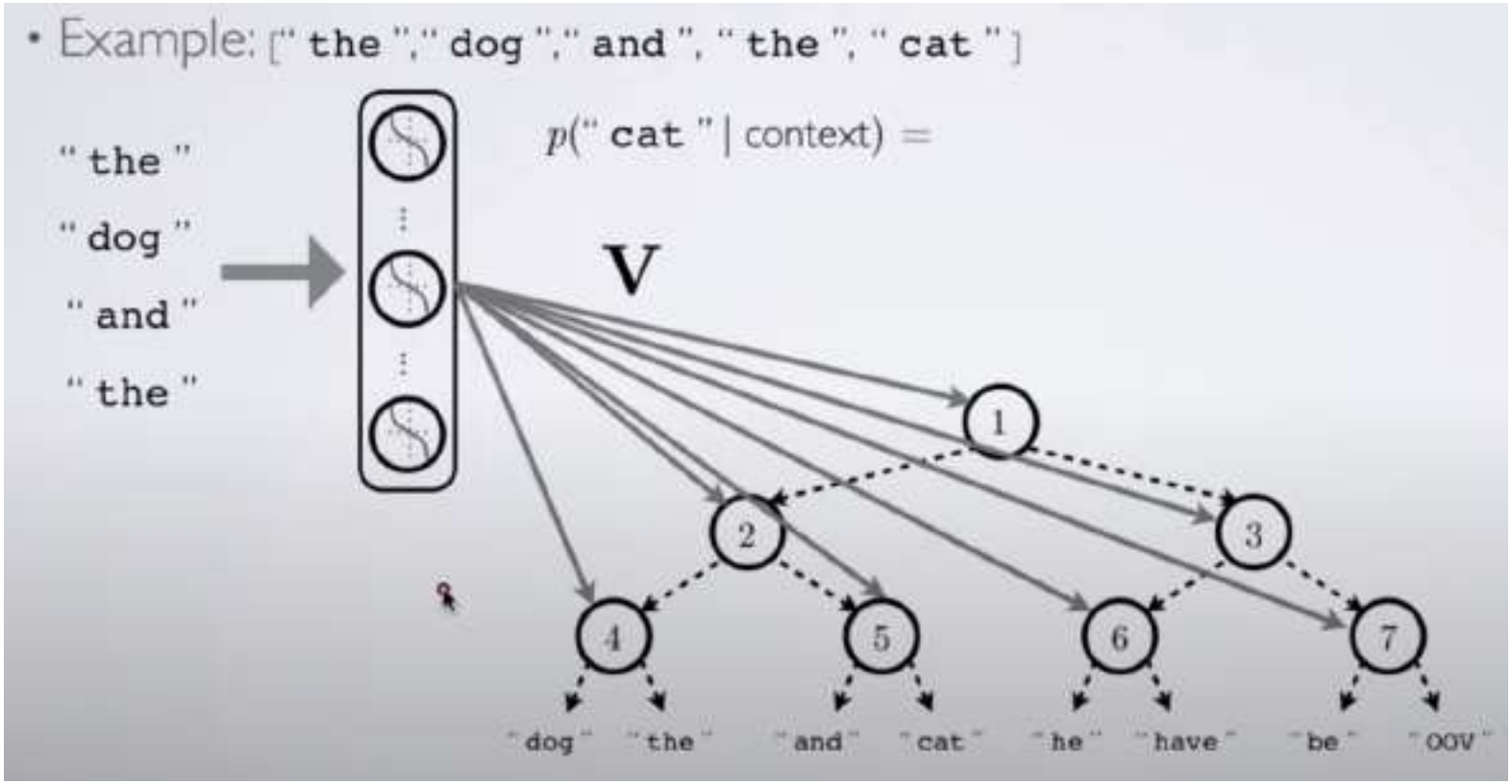


Hierarchical Softmax



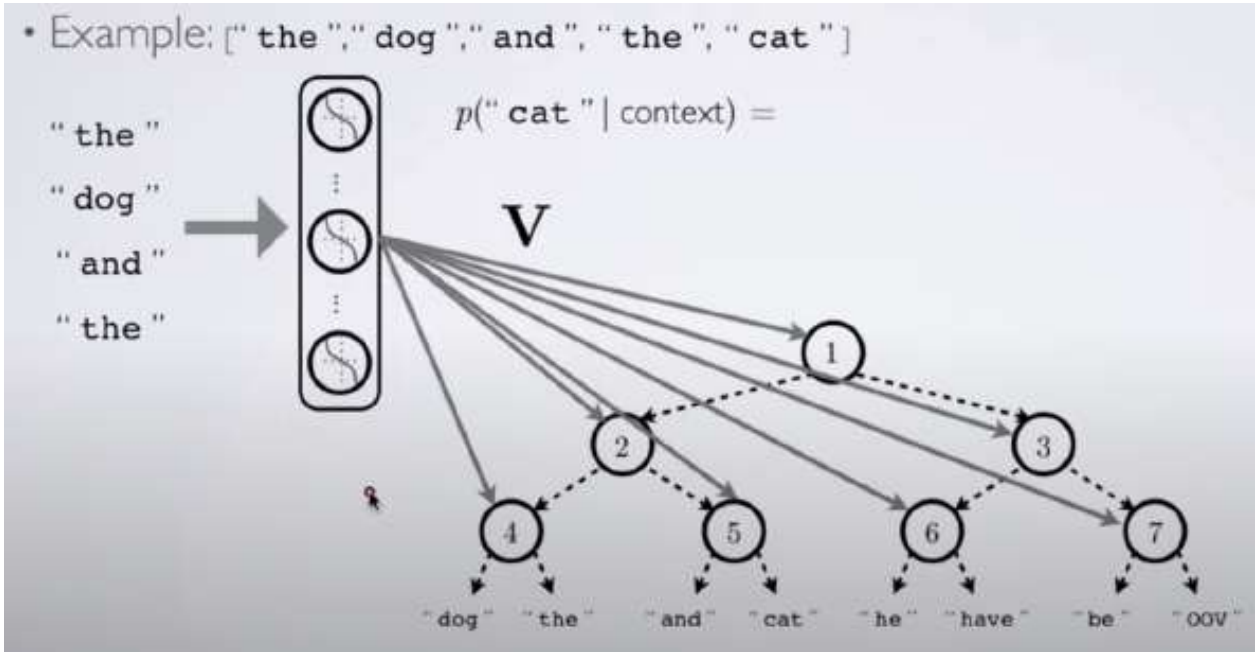
Hierarchical Softmax

- $P(cat \mid context) = p(\text{branch left at 1} \mid \text{ context}) p(\mid \text{branch right at 2} \mid\mid \text{ context}) p(\mid \text{branch right at 5} \mid\mid \text{ context})$



Hierarchical Softmax

- How to compute probabilities at inner nodes?
  - probability at a node to it's right child node $\sigma(v_w \mathbf{h})$
  - probability at a to to its left child node $1 - \sigma(v_w \mathbf{h})$



Hierarchical Softmax

item $P(cat \mid context) = (1 - \sigma(v_w \mathbf{h})(\sigma(v_w \mathbf{h})(\sigma(v_w \mathbf{h})$