

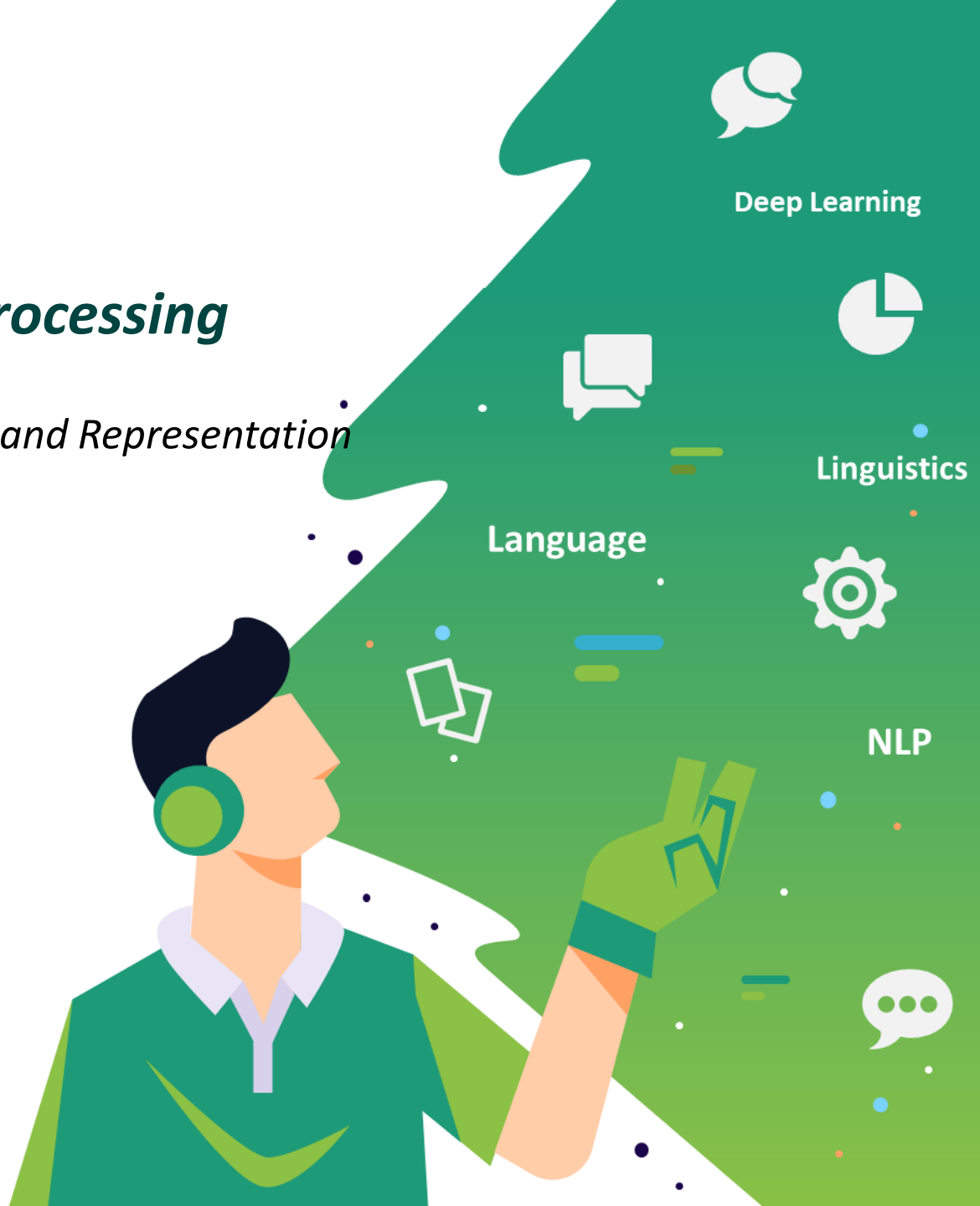
COMP5046

Natural Language Processing

Lecture 2: Word Embeddings and Representation

Dr. Caren Han

*Semester 1, 2021
School of Computer Science,
University of Sydney*



Lecture 2: Word Embeddings and Representation

1. **Lab Info**
2. Previous Lecture Review
 1. Word Meaning and WordNet
 2. Count based Word Representation
3. Prediction based Word Representation
 1. Introduction to the concept 'Prediction'
 2. Word2Vec
 3. FastText
 4. GloVe
4. Next Week Preview

Info: Lab Exercise

What do we do during Labs?

In Labs, Students will use Google Colab

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

Info: Lab Exercise

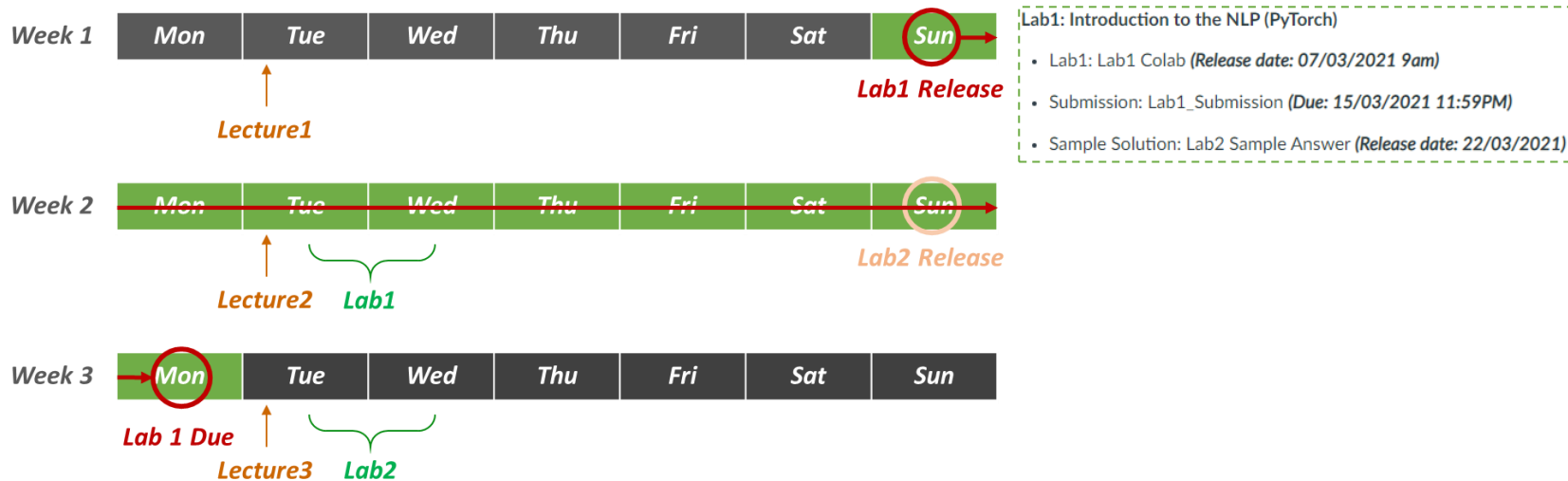
Submissions

How to Submit

Students should submit “**ipynb**” file (Download it from “File” > “Download .ipynb”) to Canvas.

When and Where to Submit

Students must submit the Lab 1(for Week2) by **Week 3 Monday 11:59PM**.



Lecture 2: Word Embeddings and Representation

1. Lab Info
2. **Count-based Word Representation**
 1. Word Meaning
 2. Limitations
3. **Prediction based Word Representation**
 1. Introduction to the concept 'Prediction'
 2. Word2Vec
 3. FastText
 4. GloVe
4. Next Week Preview

How to represent the meaning of the word?

Definition: meaning (Collins dictionary).

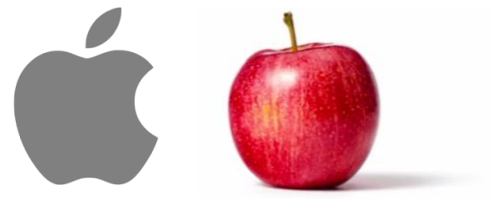
- the idea that it represents, and which can be explained using other words.
- the thoughts or ideas that are intended to be expressed by it.

signifier (symbol) \Leftrightarrow signified (idea or thing) = denotation

“Computer”



“Apple”



Problem with one-hot vectors

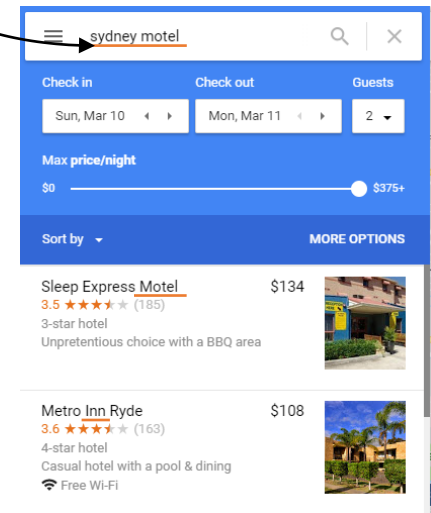
Problem #1. No word similarity representation

Example: in web search, if user searches for “Sydney motel”, we would like to match documents containing “Sydney Inn”

$$\begin{array}{l} \text{motel} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ \dots \ 0] \\ \text{hotel} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \dots \ 0] \\ \text{Inn} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \dots \ 1] \end{array}$$

hotel *motel* *Inn*

The diagram shows three one-hot vectors for the words 'motel', 'hotel', and 'Inn'. Each vector is a long sequence of 0s with a single 1 at a specific position. The positions of the 1s are highlighted by green dashed boxes. Arrows point from the labels 'hotel', 'motel', and 'Inn' to their respective vectors.



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

Problem #2. Inefficiency

Vector dimension = number of words in vocabulary

Each representation has only a single '1' with all remaining 0s.

Problem with BoW (Bag of Words)

- The intuition is that documents are similar if they have similar content. Further, that from the content alone we can learn something about the meaning of the document.
- **Discarding word order** ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged (“this is interesting” vs “is this interesting”).

*S1= I **love** you but you **hate** me*

*S2= I **hate** you but you **love** me*



Limitation of Term Frequency Inverse Document Frequency

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{df_i} \right) \longleftarrow 1+df_i$$

$w_{i,j}$ = weight of term i in document j

$tf_{i,j}$ = number of occurrences of term i in document j

N = total number of documents

df_i = number of documents containing term i

- It computes document similarity directly in the word-count space, which may be slow for large vocabularies.
- It assumes that the counts of different words provide independent evidence of similarity.
- It makes no use of ***semantic similarities between words***.

Sparse Representation

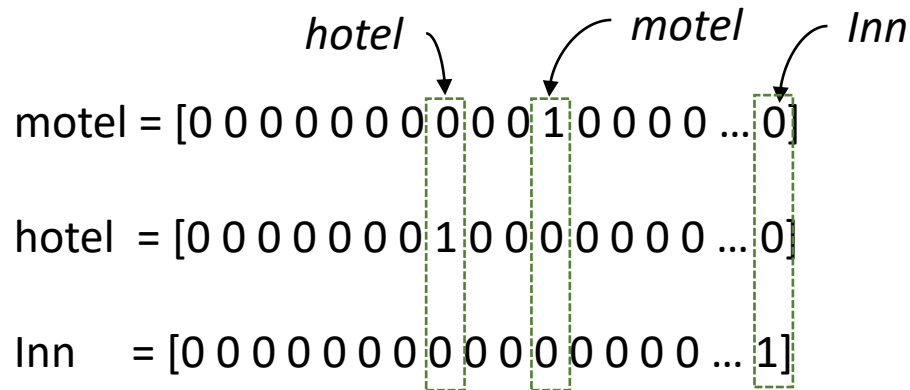
With **COUNT based word representation** (especially, one-hot vector), linguistic information was represented with **sparse representations** (high-dimensional features)

hotel *motel* *Inn*

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 ... 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 ... 0]

Inn = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 1]



Sparse Representation

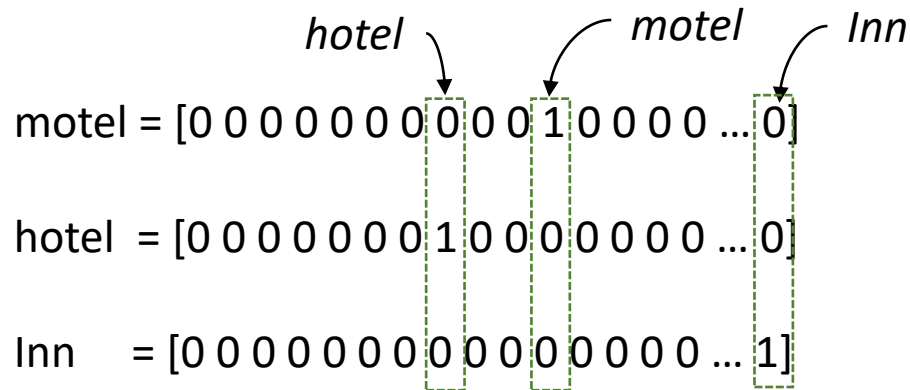
With **COUNT based word representation** (especially, one-hot vector), linguistic information was represented with **sparse representations** (high-dimensional features)

hotel *motel* *Inn*

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 ... 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 ... 0]

Inn = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 1]



A Significant Improvement Required!

1. How to get the low-dimensional vector representation
2. How to represent the word similarity

maybe a low-dimensional vector?

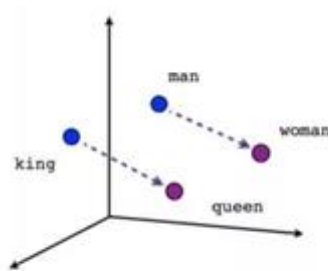
Can we use a list of fixed numbers (properties) to represent the word?

Lecture 2: Word Embeddings and Representation

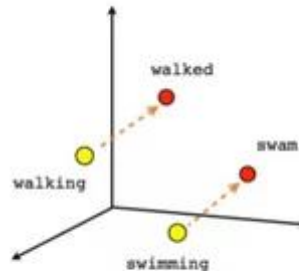
1. Lab Info
2. Previous Lecture Review
 1. Word Meaning and WordNet
 2. Count based Word Representation
3. **Prediction based Word Representation**
 1. Word Embedding
 2. Word2Vec
 3. FastText
 4. Glove
4. Next Week Preview

How to Represent the Word Similarity!

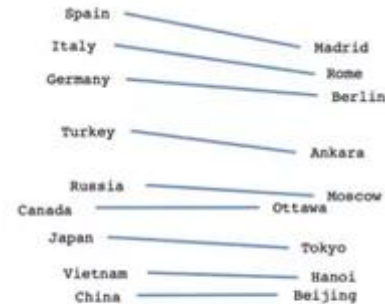
- How to represent the word similarity with dense vector



Male-Female



Verb tense



Country-Capital

- Try this with word2vec

Word Algebra

Enter all three words, the first two, or the last two and see the words that result.

+ (-) =
china 0.7477672216910414

3

Prediction based Word representation

Let's get familiar with using vectors to represent things

Assume that you are taking a personality test (the Big Five Personality Traits test)

1)Openness, 2)Agreeableness, 3)Conscientiousness, 4)Negative emotionality, 5)Extraversion



Jane



Openness

100

0



3

Prediction based Word representation

Let's get familiar with using vectors to represent things

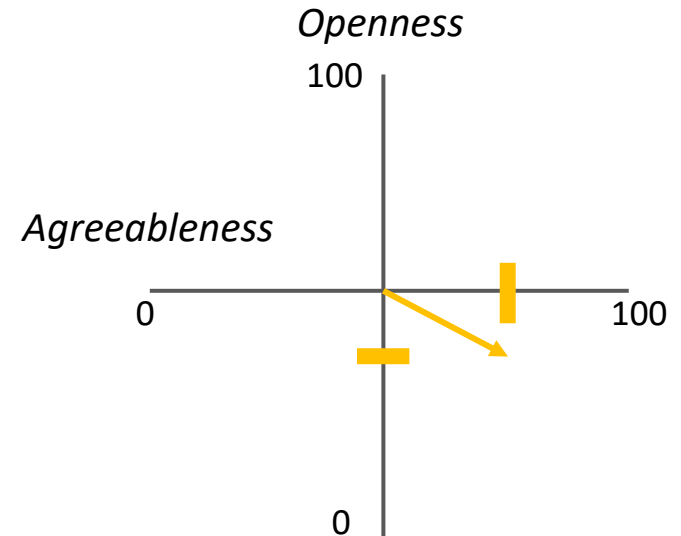
Assume that you are taking a personality test (the Big Five Personality Traits test)

1)Openness, 2)Agreeableness, 3)Conscientiousness, 4)Negative emotionality, 5)Extraversion



Jane

Openness		Agreeableness		
40	70			






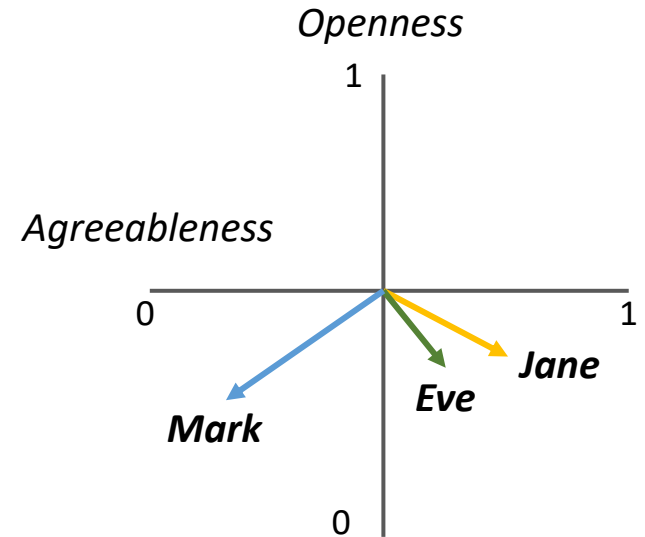
Prediction based Word representation

Let's get familiar with using vectors to represent things

Assume that you are taking a personality test (the Big Five Personality Traits test)

1)Openness, 2)Agreeableness, 3)Conscientiousness, 4)Negative emotionality, 5)Extraversion

		Openness	Agreeableness			
	Jane	0.4	0.7			
	Mark	0.3	0.2			
	Eve	0.4	0.6			

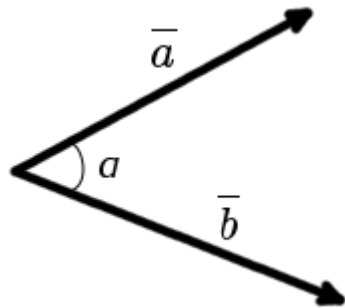


Let's get familiar with using vectors to represent things

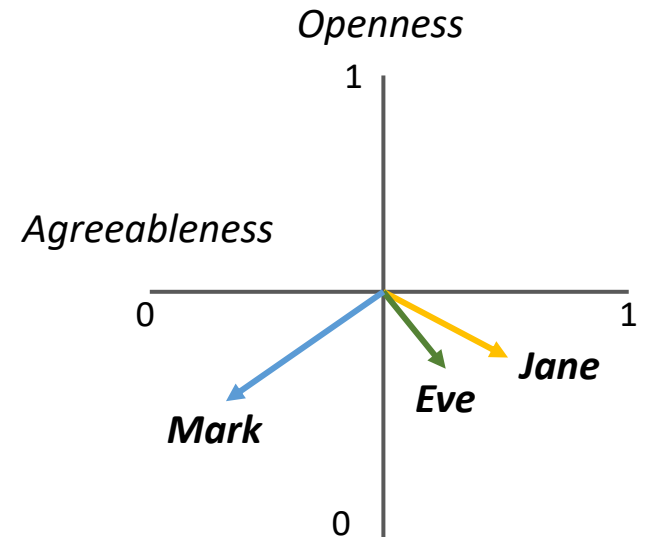
Which of two people (Mark or Eve) is more similar to Jane?

Cosine Similarity

Measure of similarity between two vectors of inner product space that measures the cosine of the angle between them



$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$






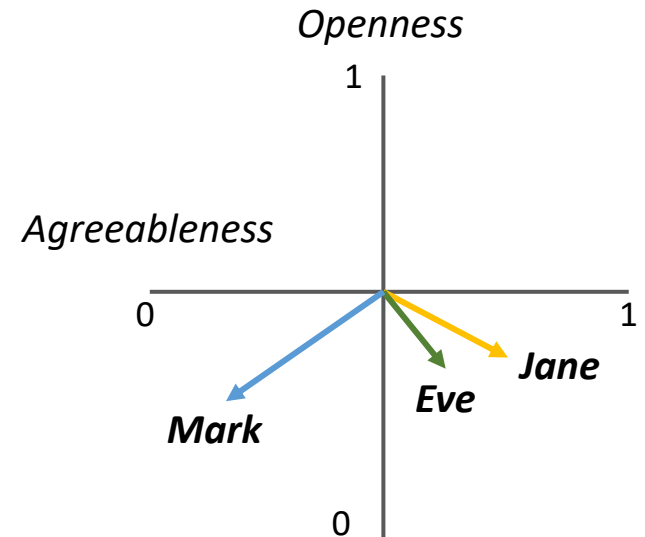
3

Prediction based Word representation

Let's get familiar with using vectors to represent things

Which of two people (Mark or Eve) is more similar to Jane?

		Openness		Agreeableness		
	Jane	0.4	0.7			
	Mark	0.3	0.2			
	Eve	0.4	0.6			



$$\cos\left(\begin{matrix} \text{Jane} \\ 0.4 & 0.7 \end{matrix}, \begin{matrix} \text{Mark} \\ 0.3 & 0.2 \end{matrix}\right) \approx 0.89$$




$$\cos\left(\begin{matrix} \text{Jane} \\ 0.4 & 0.7 \end{matrix}, \begin{matrix} \text{Eve} \\ 0.4 & 0.6 \end{matrix}\right) \approx 0.99$$

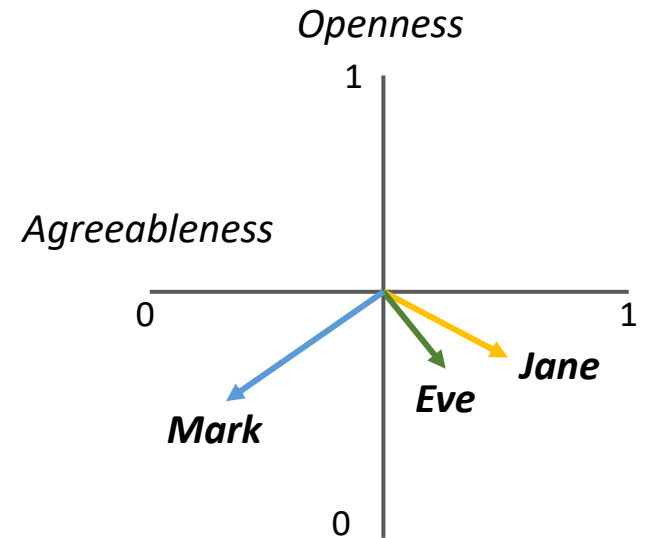
3

Prediction based Word representation

Let's get familiar with using vectors to represent things

We need all five major factors for represent the personality

		Openness	Agreeableness	Conscientiousness	NE	Extraversion
	Jane	0.4	0.7	0.5	0.2	0.1
	Mark	0.3	0.2	0.3	0.7	0.2
	Eve	0.4	0.6	0.4	0.3	0.5

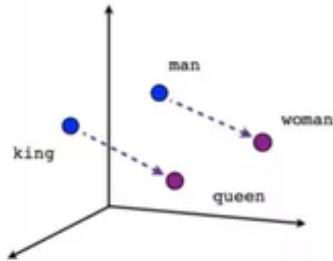


With these embeddings,

1. Represent things as vectors of fixed numbers!
2. Easily calculate the similarity between vectors

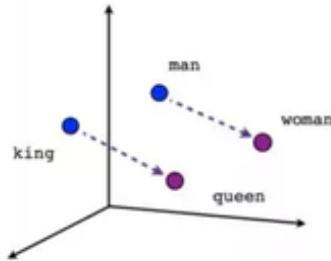
Prediction based Word representation

Remember? The Word2Vec Demo!



This is a word embedding for the word “king”

Remember? The Word2Vec Demo!



This is a word embedding for the word “king”

** Trained by Wikipedia Data, 50-dimension GloVe Vector*

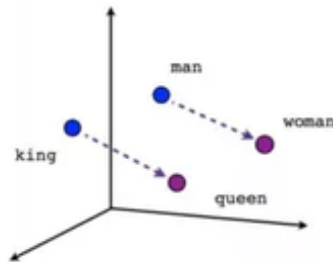
king

[0.50451, 0.68607, -0.59517, -0.022801, 0.60046, 0.08813, 0.47377, -0.61798, -0.31012, -0.066666, 1.493, -0.034173, -0.98173, 0.68229, 0.812229, 0.81722, -0.51722, -744.5.4 1503, -0.55809, 0.66421, 0.1961, -0.1495, -0.033474, -0.30344, 0.41177, -2.223, -1.0756, -0.343554, 0.33505, 1.9927, -0.042434, -0.64519, 0.72519, 0.71419, 0.714319, 0.71419 9159, 0.16754, 0.34344, -0.25663, -0.8523, 0.1661, 0.40102, 1.1685, -1.0137, -0.2155, 0.78321, -0.91241, -1.6626, -0.64426, -0.542102]

3

Prediction based Word representation

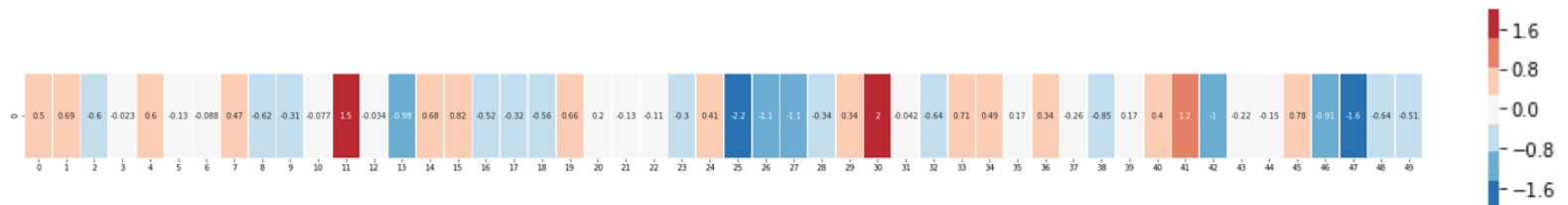
Remember? The Word2Vec Demo!



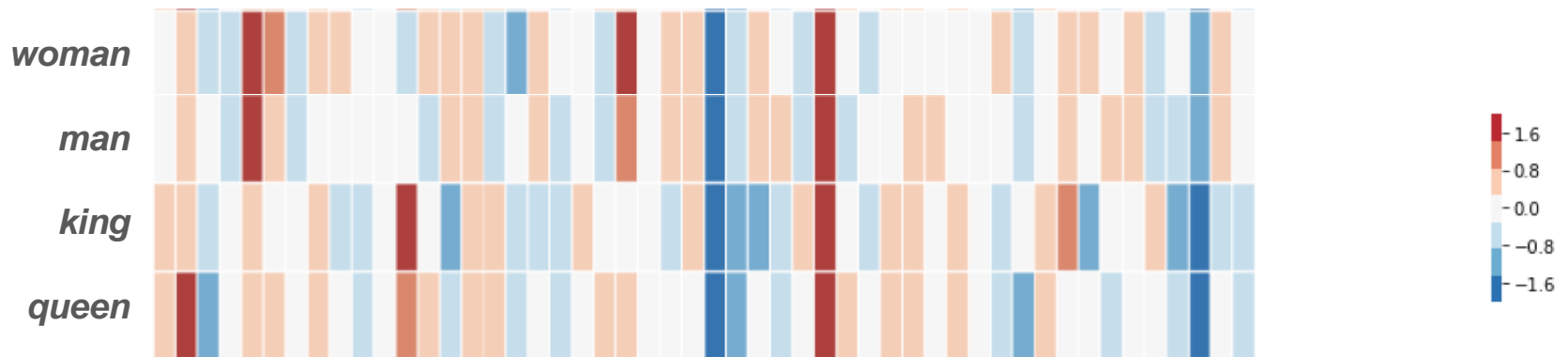
This is a word embedding for the word “king”

** Trained by Wikipedia Data, 50-dimension GloVe Vector*

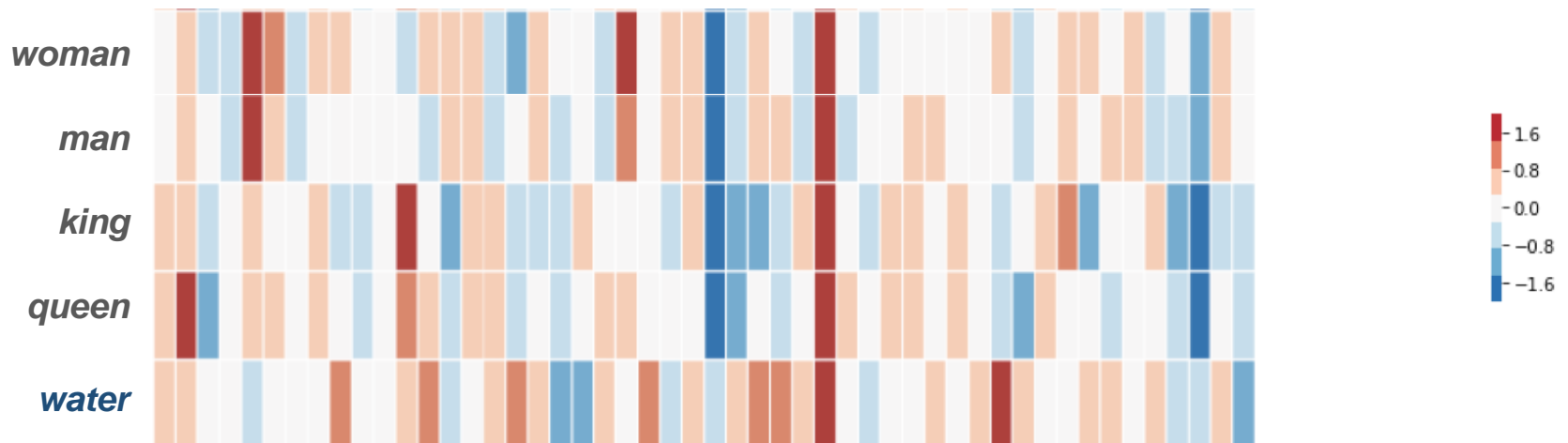
king



Remember? The Word2Vec Demo!



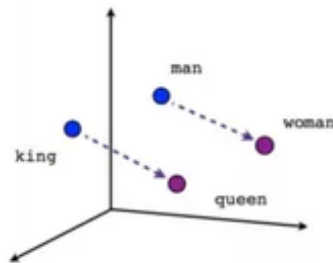
Remember? The Word2Vec Demo!



3

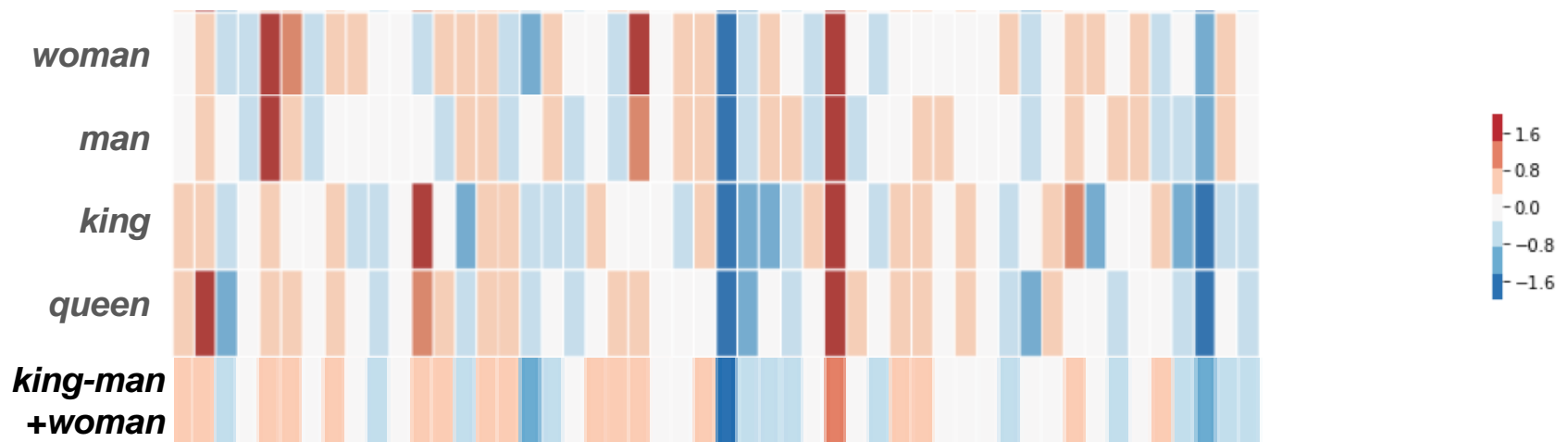
Prediction based Word representation

Remember? The Word2Vec Demo!



$king - man + woman \approx queen?$

Word Algebra



How to make dense vectors for word representation

How to make dense vectors for word representation



Prof. John Rupert Firth

“You shall know a word by the company it keeps”

— (Firth, J. R. 1957:11)

*Prof. Firth is noted for drawing attention to the **context-dependent nature of meaning** with his notion of '**context of situation**', and his work on collocational meaning is widely acknowledged in the field of **distributional semantics**.*

Prediction based Word representation

Word Representations in the context

When a *word* w appears in a text, its context is the set of words that appear nearby

- Use the surrounding contexts of w to build up a representation of w

Sydney

From Wikipedia, the free encyclopedia

This article is about the Australian metropolis. For the local government area, see [City of Sydney](#). For other uses, see [Sydney \(disambiguation\)](#).

Sydney (/ˈsɪdni/ (listen) *SID-nee*)^[7] is the state capital of New South Wales and the most populous city in Australia and Oceania.^[8] Located on Australia's east coast, the metropolis surrounds Port Jackson and extends about 70 km (43.5 mi) on its periphery towards the Blue Mountains to the west, Hawkesbury to the north, the Royal National Park to the south and Macarthur to the south-west.^[9] Sydney is made up of 658 suburbs, 40 local government areas and 15 contiguous regions. Residents of the city are known as "Sydneyiders".^[10] As of June 2017, Sydney's estimated metropolitan population was 5,131,326.^[11] and is home to approximately 65% of the state's population.^[12]

Indigenous Australians have inhabited the Sydney area for at least 30,000 years, and thousands of engravings remain throughout the region, making it one of the richest in Australia in terms of Aboriginal archaeological sites. During his first Pacific voyage in 1770, Lieutenant James Cook and his crew became the first Europeans to chart the eastern coast of Australia, making landfall at Botany Bay and inspiring British interest in the area. In 1788, the *First Fleet* of convicts, led by Arthur Phillip, founded Sydney as a British penal colony, the first European settlement in Australia. Phillip named the city Sydney in recognition of Thomas Townshend, 1st Viscount Sydney.^[13] Penal transportation to New South Wales ended soon after Sydney was incorporated as a city in 1842. A gold rush occurred in the colony in 1851, and over the next century, Sydney transformed from a colonial outpost into a major global cultural and economic centre. After World War II, it experienced mass migration and became one of the most multicultural cities in the world.^[3] At the time of the 2011 census, more than 250 different languages were spoken in Sydney.^[14] In the 2016 Census, about 35.8% of residents spoke a language other than English at home.^[15] Furthermore, 45.4% of the population reported having been born overseas, making Sydney the 3rd largest foreign born population of any city in the world after London and New York City, respectively.^{[16][17]}

How can we train the word representation to machine?

Neural Network! (Machine Learning)

Article
Talk
Read
Edit
View history
Se

Sydney

From Wikipedia, the free encyclopedia

This article is about the Australian metropolis. For the local government area, see [City of Sydney](#). For other uses, see [Sydney \(disambiguation\)](#).

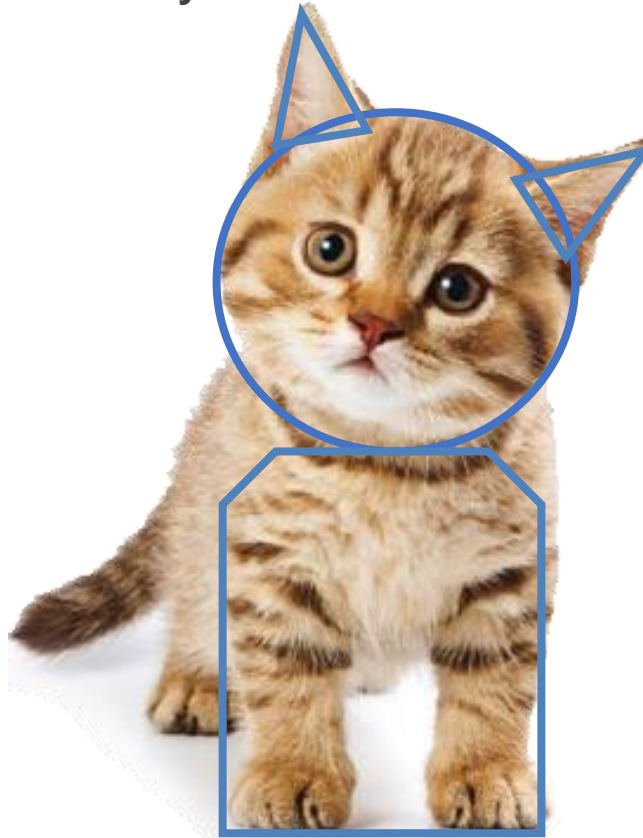
Sydney (/ˈsɪdni/ (listen) *SID*-nee)^[7] is the state capital of New South Wales and the most populous city in Australia and Oceania.^[8] Located on Australia's east coast, the metropolis surrounds [Port Jackson](#) and extends about 70 km (43.5 mi) on its periphery towards the [Blue Mountains](#) to the west, [Hawkesbury](#) to the north, the [Royal National Park](#) to the south and [Macarthur](#) to the south-west.^[9] [Sydney](#) is made up of 658 suburbs, 40 local government areas and 15 contiguous regions. Residents of the city are known as "[Sydney](#)siders".^[10] As of June 2017, [Sydney](#)'s estimated metropolitan population was 5,131,326.^[11] and is home to approximately 65% of the state's population.^[12]

Indigenous Australians have inhabited the [Sydney](#) area for at least 30,000 years, and thousands of engravings remain throughout the region, making it one of the richest in Australia in terms of [Aboriginal archaeological sites](#). During his first Pacific voyage in 1770, Lieutenant [James Cook](#) and his crew became the first Europeans to chart the eastern coast of Australia, making landfall at Botany Bay and inspiring British interest in the area. In 1788, the *First Fleet* of convicts, led by [Arthur Phillip](#), founded [Sydney](#) as a British penal colony, the first European settlement in Australia. Phillip named the city [Sydney](#) in recognition of [Thomas Townshend](#), 1st Viscount [Sydney](#).^[13] Penal transportation to New South Wales ended soon after [Sydney](#) was incorporated as a city in 1842. A gold rush occurred in the colony in 1851, and over the next century, [Sydney](#) transformed from a colonial outpost into a major global cultural and economic centre. After [World War II](#), it experienced mass migration and became one of the most multicultural cities in the world.^[3] At the time of the 2011 census, more than 250 different languages were spoken in [Sydney](#).^[14] In the 2016 Census, about 35.8% of residents spoke a language other than English at home.^[15] Furthermore, 45.4% of the population reported having been born overseas, making [Sydney](#) the 3rd largest foreign born population of any city in the world after London and New York City, respectively.^{[16][17]}

These context words will represent **Sydney**

Machine Learning

How to classify this with your machine?



Object: CAT

Computer System

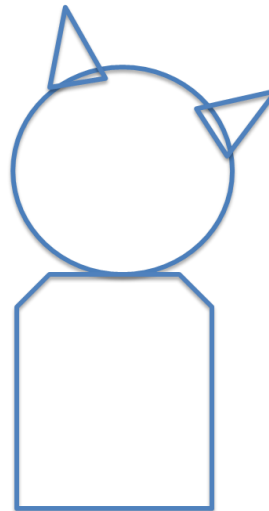


Data

```
def prediction(image as input):  
    ...program...  
    return result
```



Result

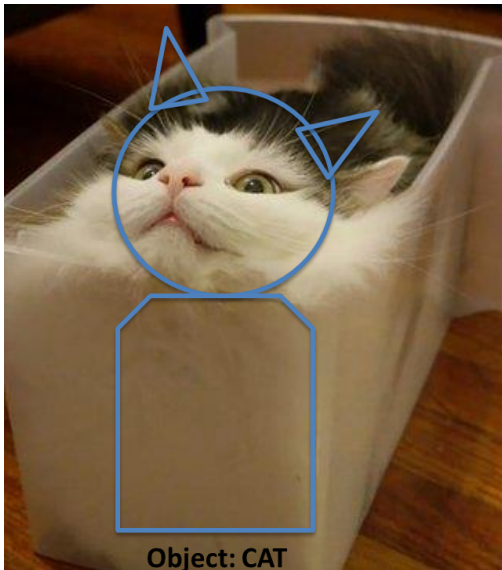


Object: CAT



Object: CAT

Can we classify this with the computer system?



Object: ???

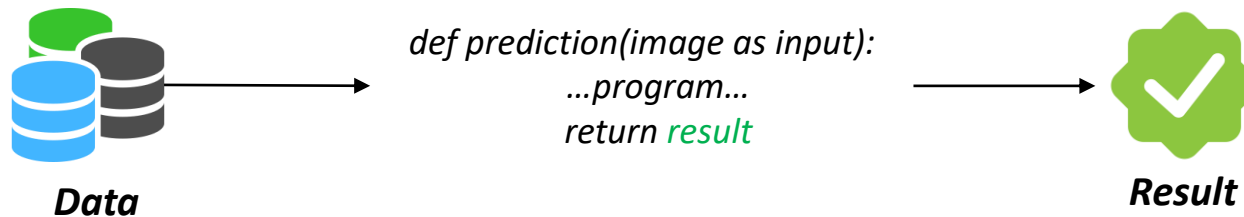


Object: ???



Object: ???

Computer System VS Machine Learning



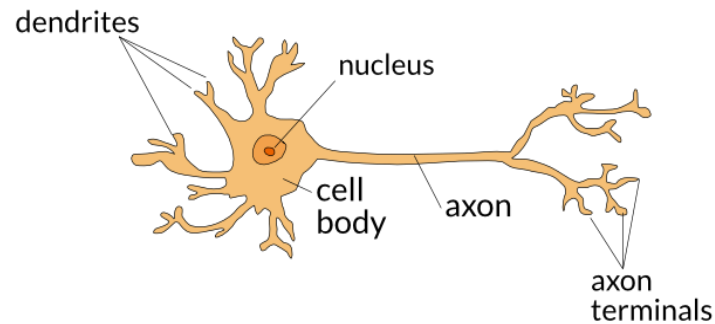
$$\{x_i, y_i\}_{i=1}^N$$

x_i	Input	words (indices or vectors), sentences, documents, etc.
y_i	class	What we try to classify/predict

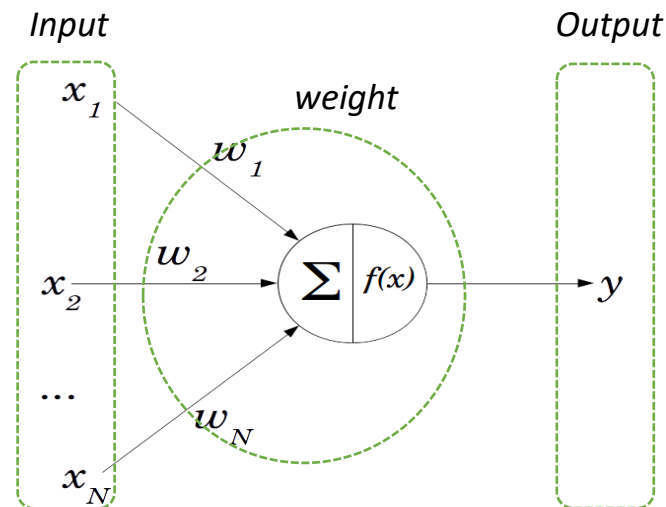
Neural Network and Deep Learning

Neuron and Perceptron

Neuron



Perceptron



The detailed neural network and deep learning concept will be covered in the Lecture 3

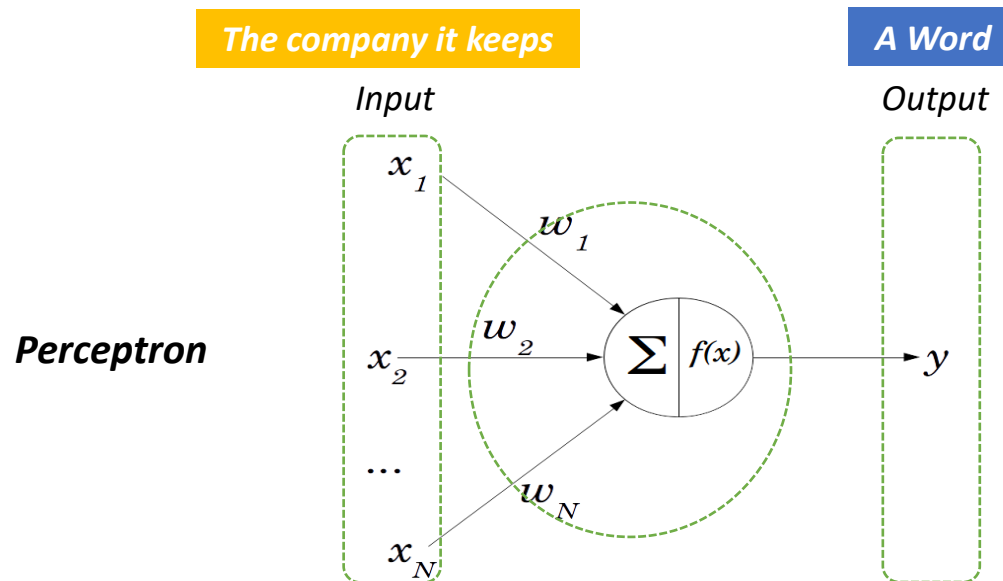
Prediction based Word representation

Neural Network and Deep Learning in Word Representation

“You shall know a word by the company it keeps” (Firth, J. R. 1957:11)

Why don't we train a word by the company it keeps?

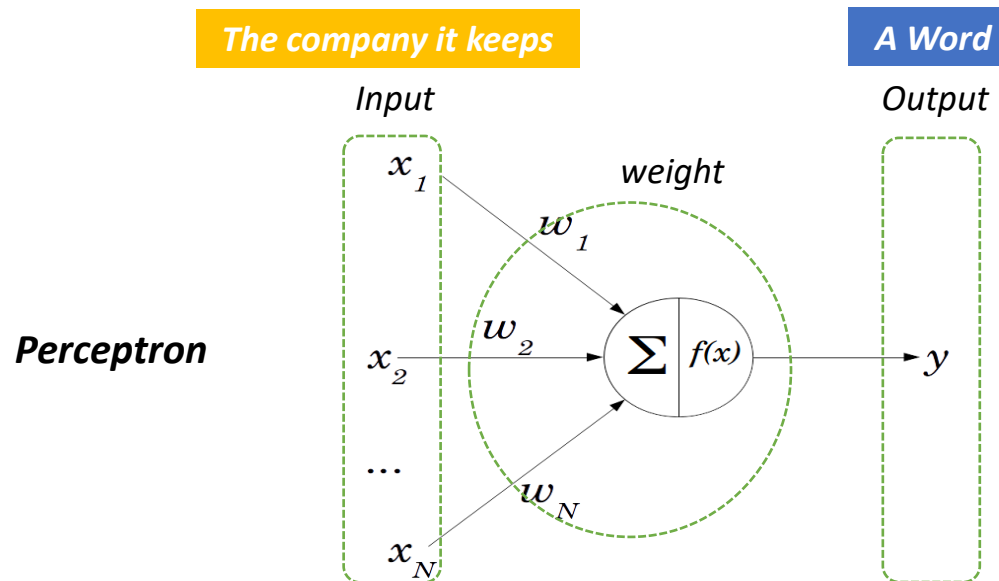
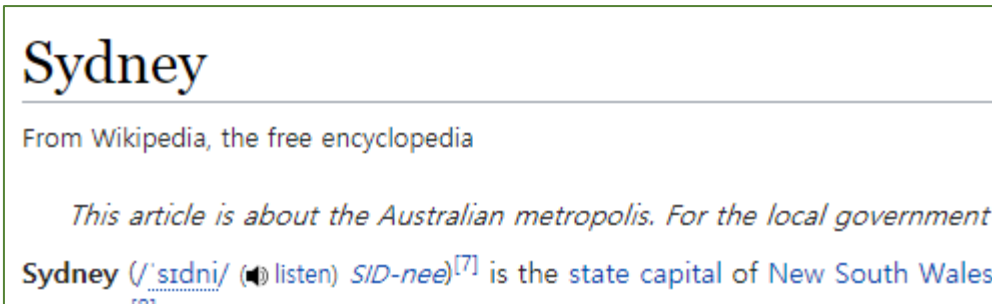
Why don't we represent a word by the company it keeps?



Prediction based Word representation

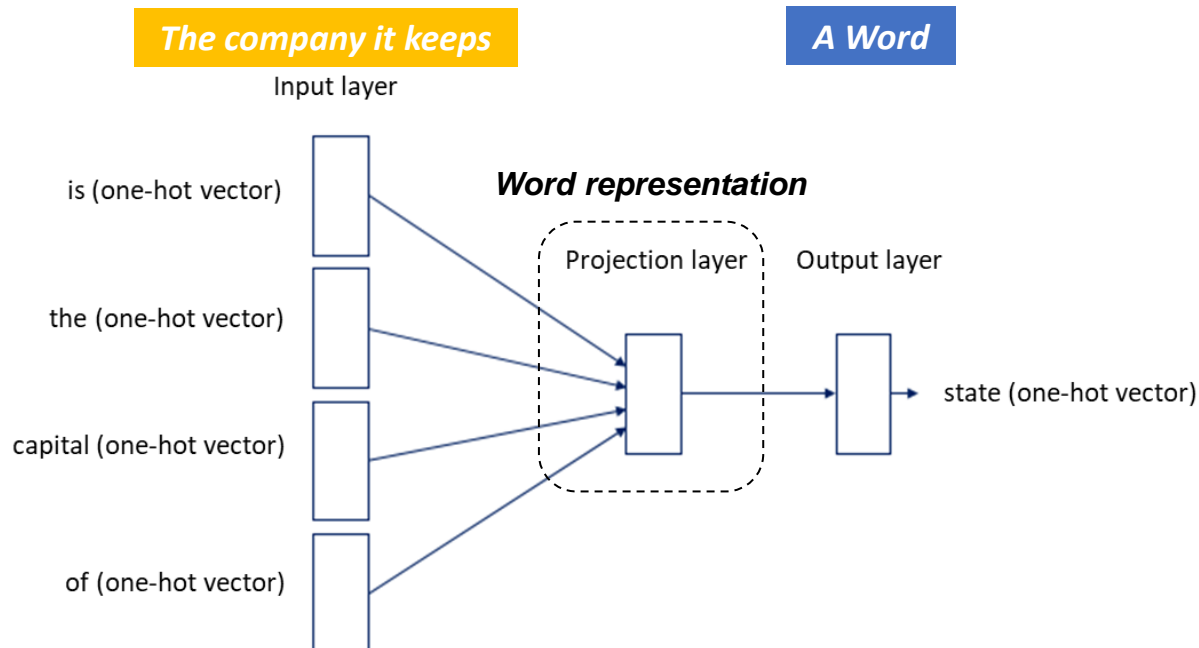
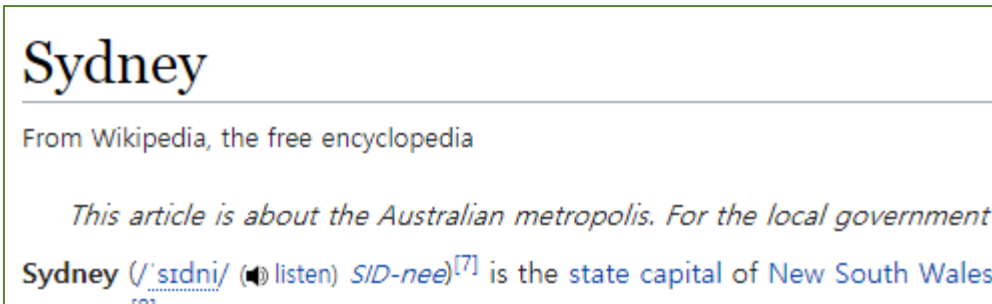
Neural Network and Deep Learning in Word Representation

Wikipedia: “Sydney is the state capital of NSW...”



Neural Network and Deep Learning in Word Representation

Wikipedia: “Sydney is the state capital of NSW...”



Neural Network and Deep Learning in Word Representation

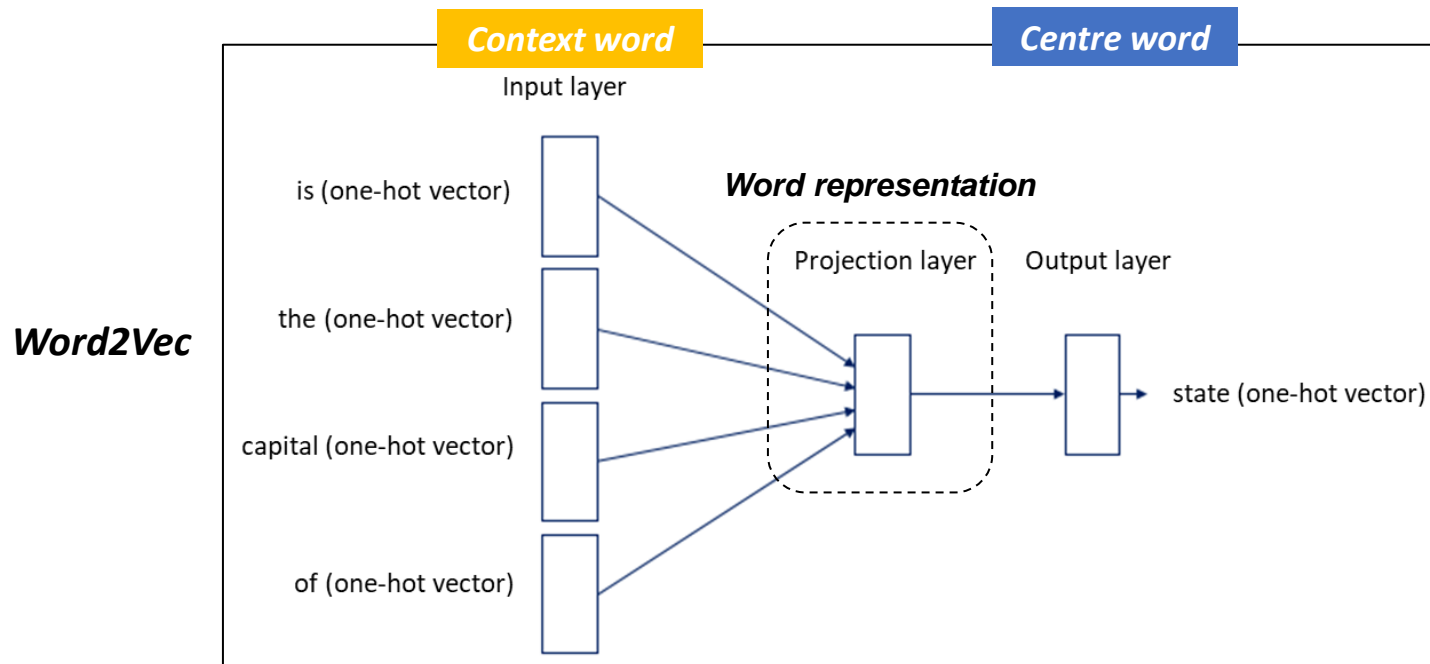
Wikipedia: “Sydney is the state capital of NSW...”

Sydney

From Wikipedia, the free encyclopedia

This article is about the Australian metropolis. For the local government area, see Sydney City Council.

Sydney (/ˈsɪdni/ (listen) *SID-nee*)^[7] is the state capital of New South Wales

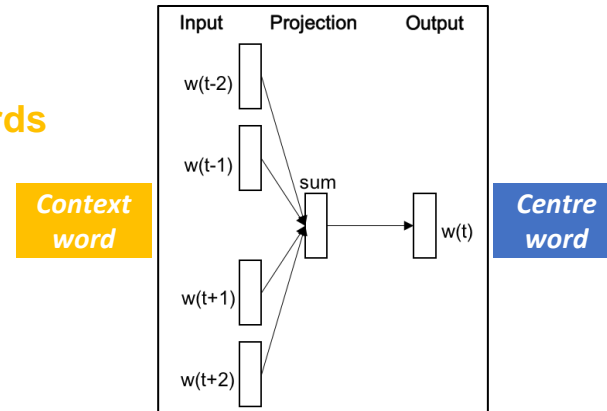


Word2Vec

Word2vec can utilize either of two model architectures to produce a distributed representation of words:

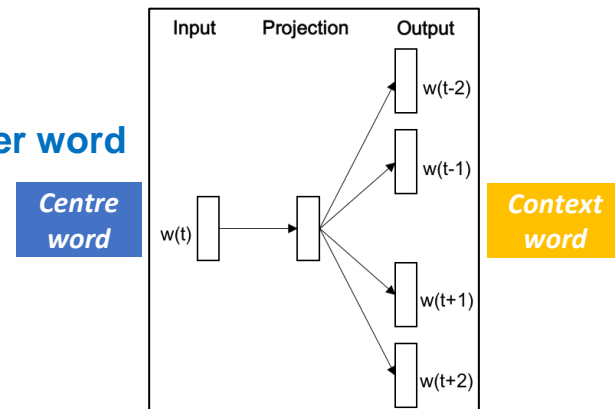
1. Continuous Bag of Words (CBOW)

Predict **center word** from (bag of) **context words**



2. Continuous Skip-gram

Predict **context ("outside") words** given **center word**



Prediction based Word representation

Word2Vec with Continuous Bag of Words (CBOW)

Predict center word from (bag of) context words

Sentence: “Sydney is the state capital of NSW”

Aim

- Predict the center word

Setup

- Window size
 - Assume that **the window size is 2**

Sydney	is	the	state	capital	of	NSW
Sydney	is	the	state	capital	of	NSW
Sydney	is	the	state	capital	of	NSW
Sydney	is	the	state	capital	of	NSW
Sydney	is	the	state	capital	of	NSW
Sydney	is	the	state	capital	of	NSW
Sydney	is	the	state	capital	of	NSW
Sydney	is	the	state	capital	of	NSW

Center word
Context (“outside”) word

Word2Vec with Continuous Bag of Words (CBOW)

Predict center word from (bag of) context words

Sentence: “Sydney is the state capital of NSW”

Using window slicing, develop the training data

Center word	Context (“outside”) word	
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0], [0,0,1,0,0,0,0]	Sydney is the state capital of NSW
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0], [0,0,1,0,0,0,0], [0,0,0,1,0,0,0]	Sydney is the state capital of NSW
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0], [0,1,0,0,0,0,0], [0,0,0,1,0,0,0], [0,0,0,0,1,0,0]	Sydney is the state capital of NSW
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0], [0,0,1,0,0,0,0], [0,0,0,1,0,0,0], [0,0,0,0,1,0,0]	Sydney is the state capital of NSW
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0], [0,0,0,1,0,0,0], [0,0,0,0,1,0,0], [0,0,0,0,0,1,0]	Sydney is the state capital of NSW
[0,0,0,0,0,1,0]	[0,0,0,1,0,0,0], [0,0,0,0,1,0,0], [0,0,0,0,0,1,0]	Sydney is the state capital of NSW
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0], [0,0,0,0,0,1,0]	Sydney is the state capital of NSW

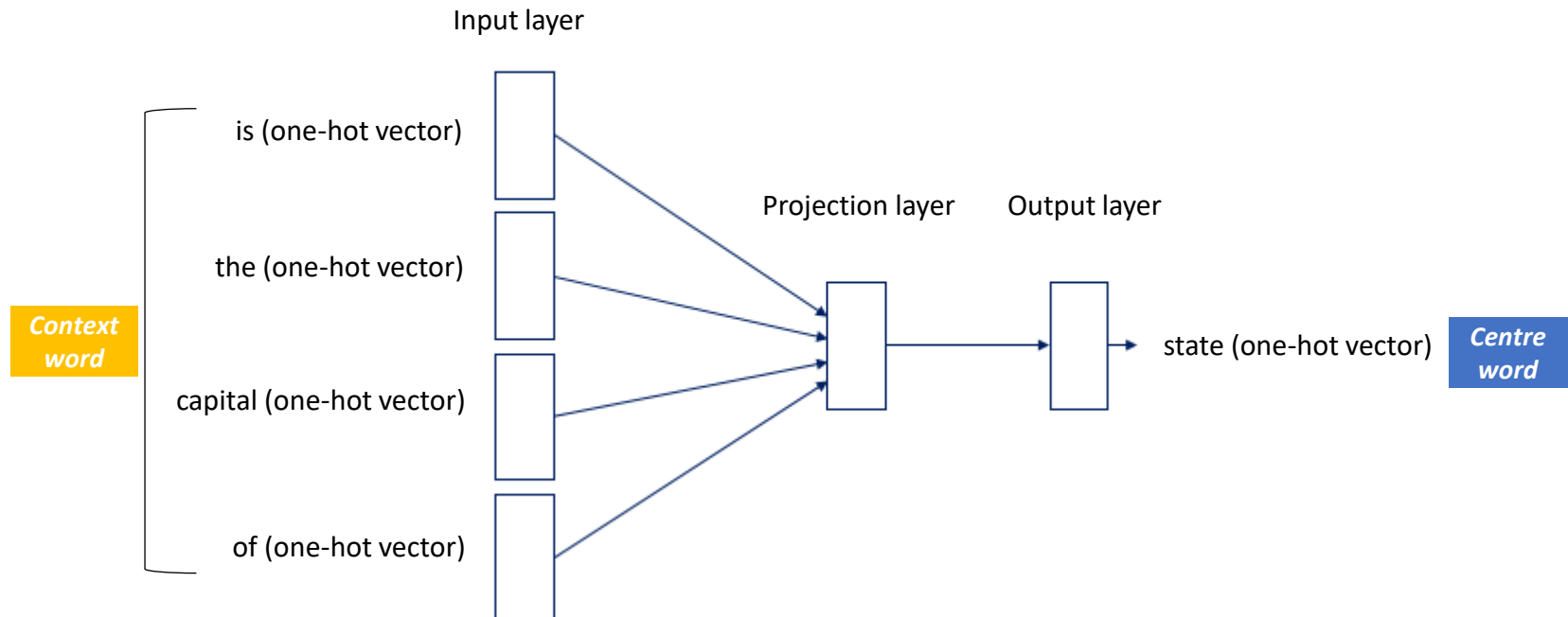
Center word
Context (“outside”) word

Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: “Sydney is the state capital of NSW”

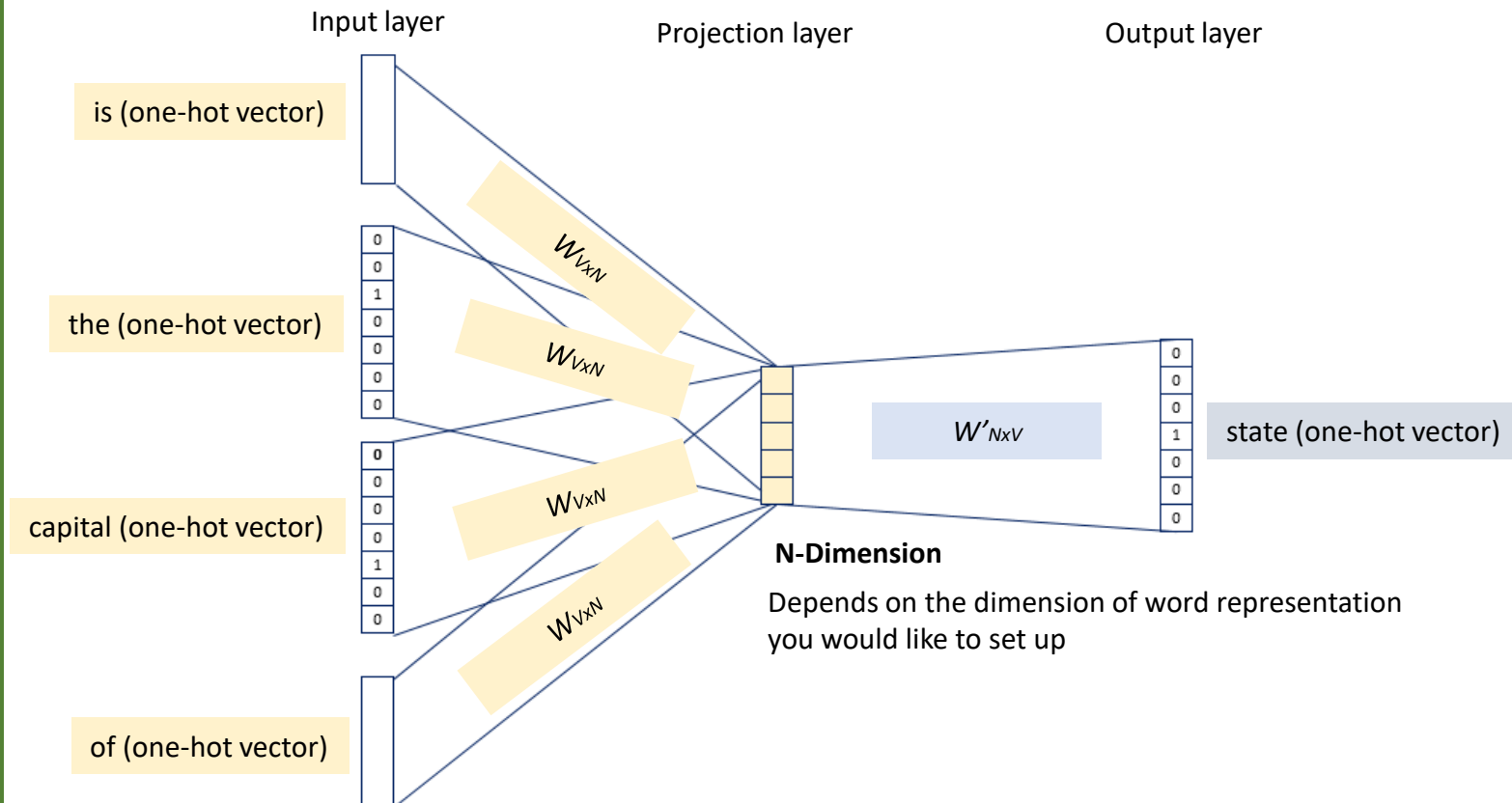


Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: “Sydney is the state capital of NSW”

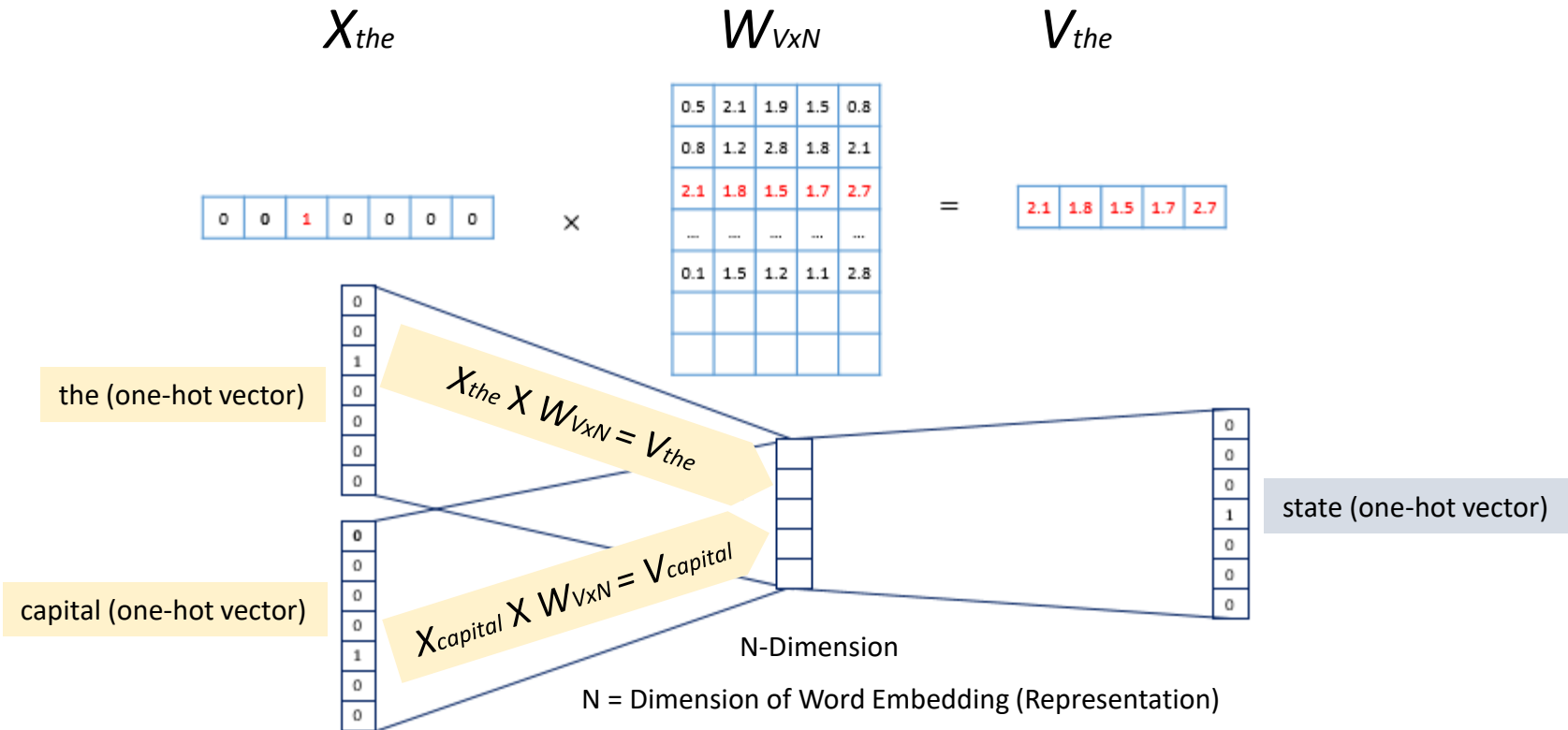


Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

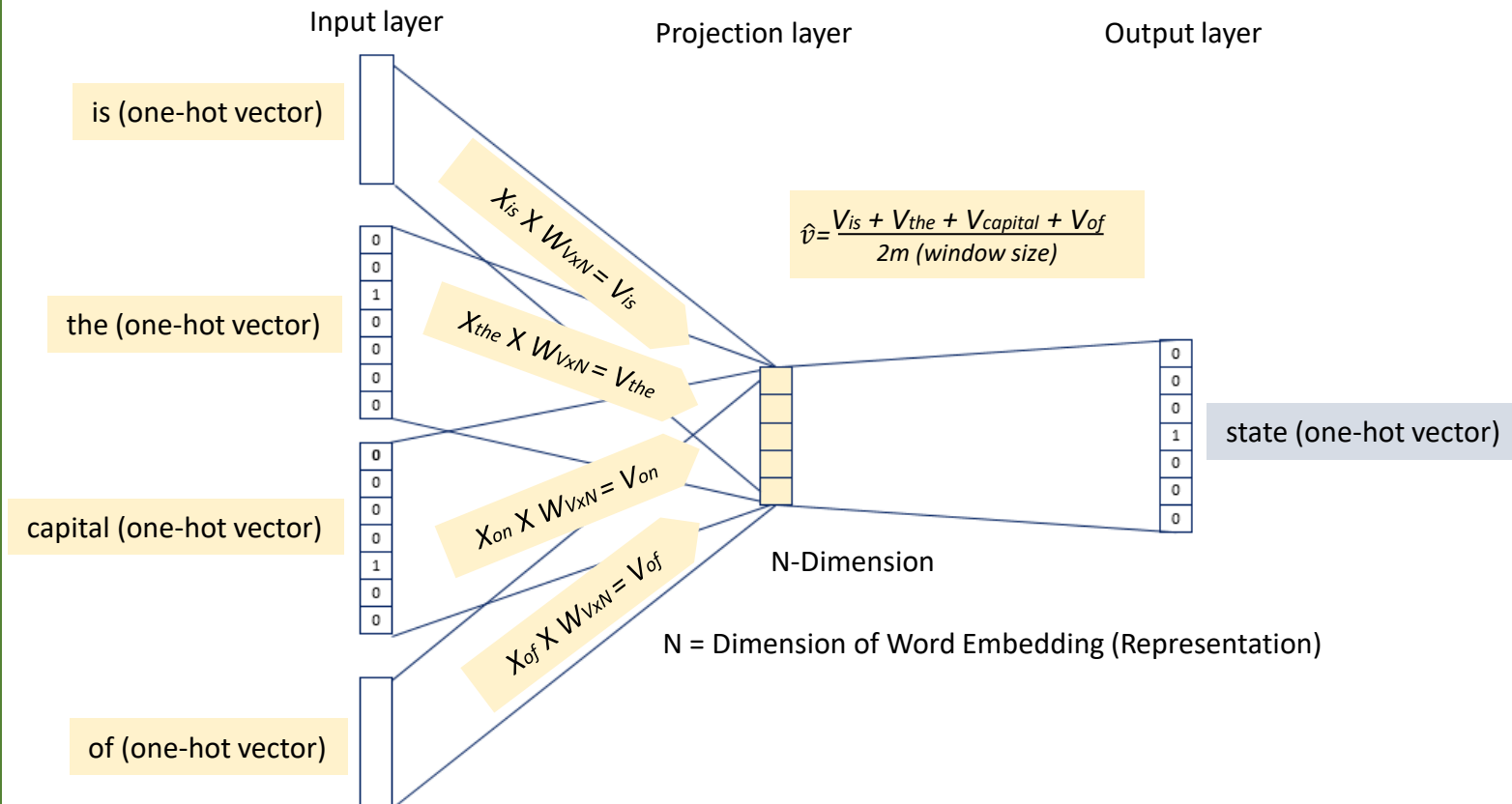
Sentence: “Sydney is the **state** capital of NSW”



CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: “Sydney is the **state** capital of NSW”

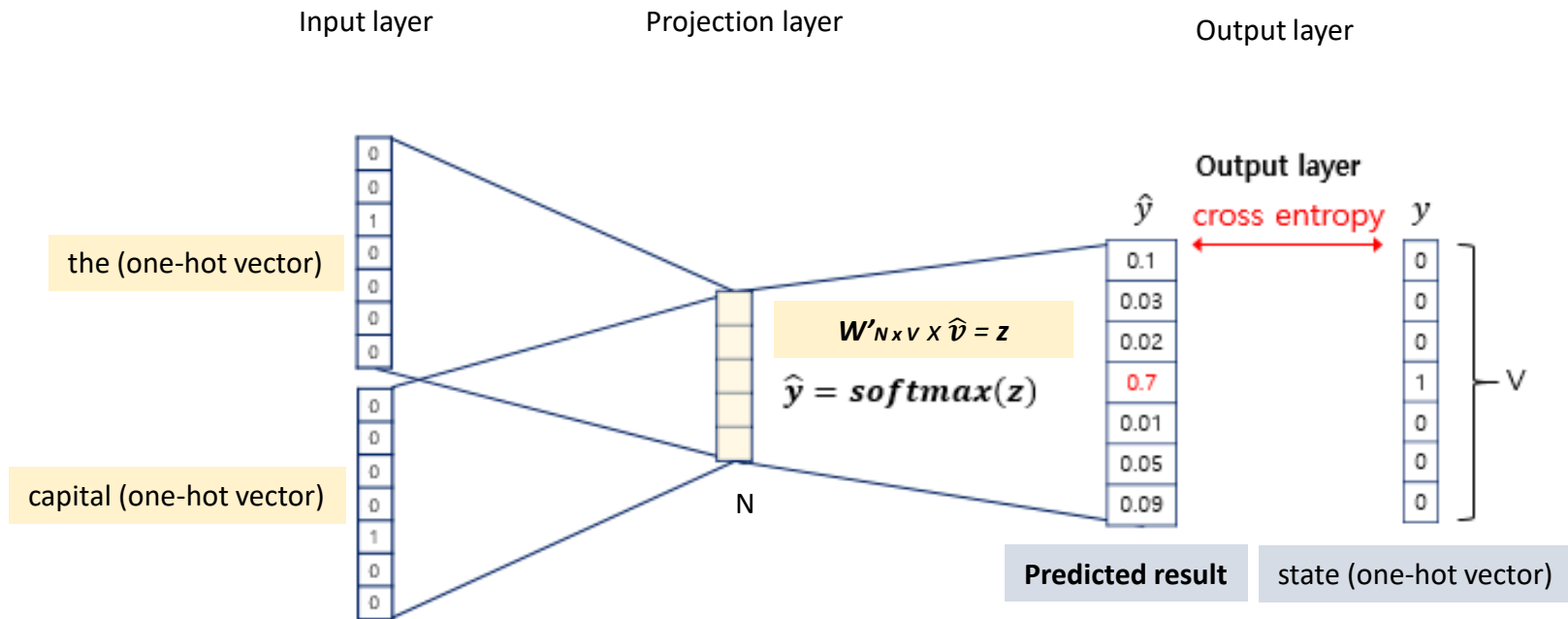


Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: “Sydney is the state capital of NSW”



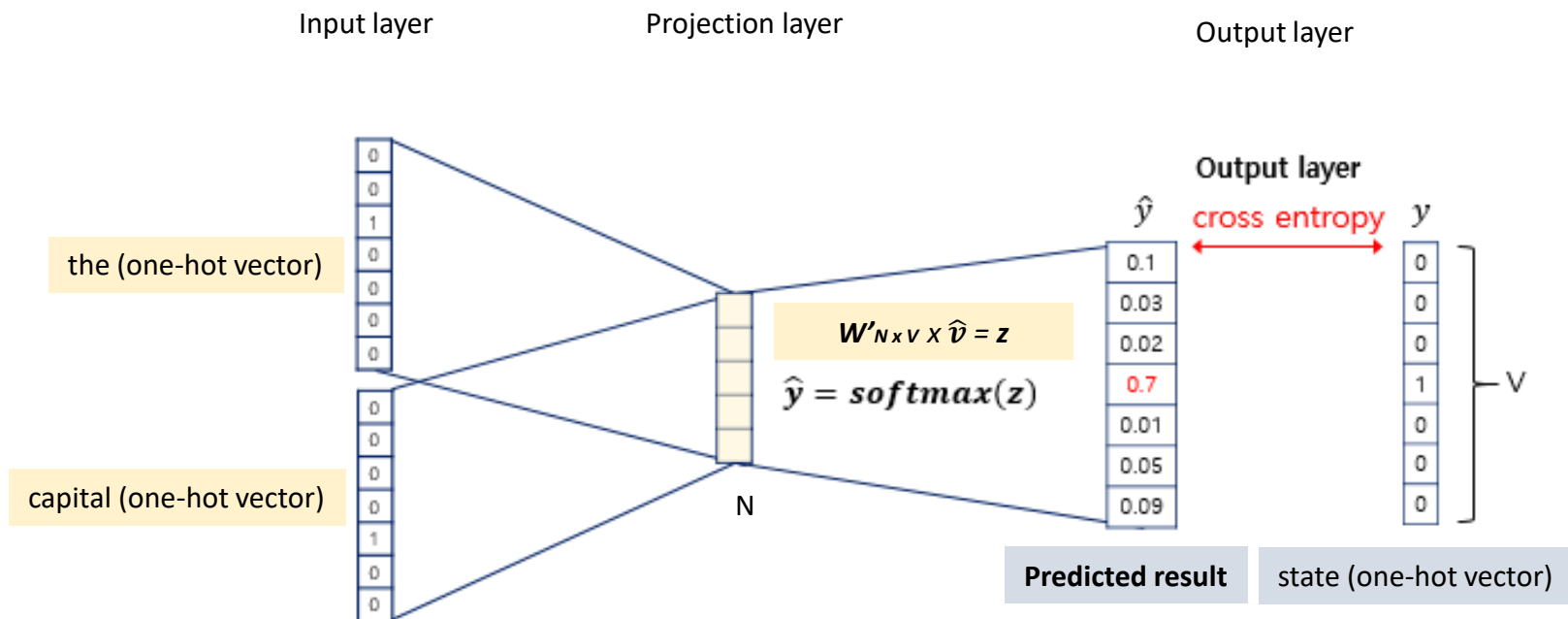
Softmax: outputs a vector that represents the probability distributions (sum to 1) of a list of potential outcome

Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: “Sydney is the state capital of NSW”



Cross Entropy: can be used as a loss function when optimizing classification

Loss Function (Cross Entropy)

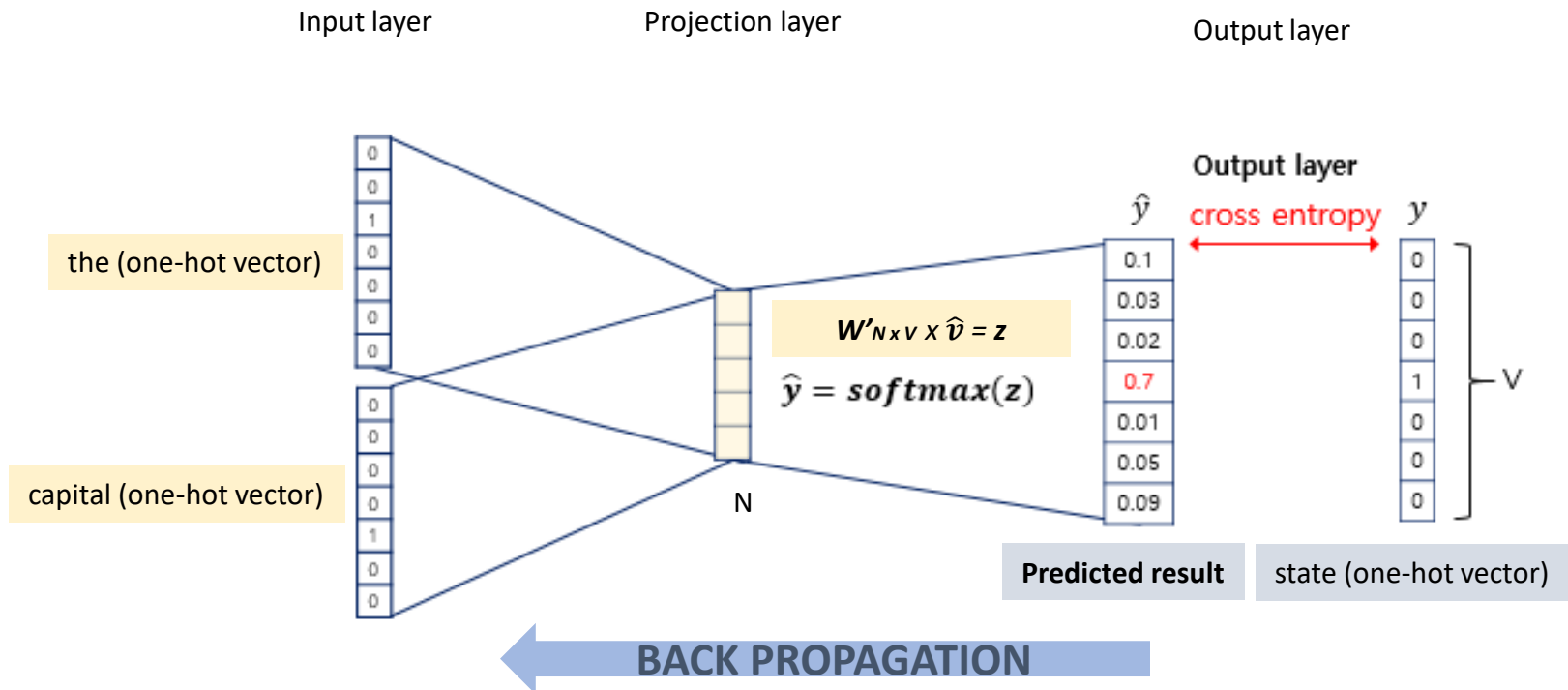
$$H(\hat{y}, y) = - \sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

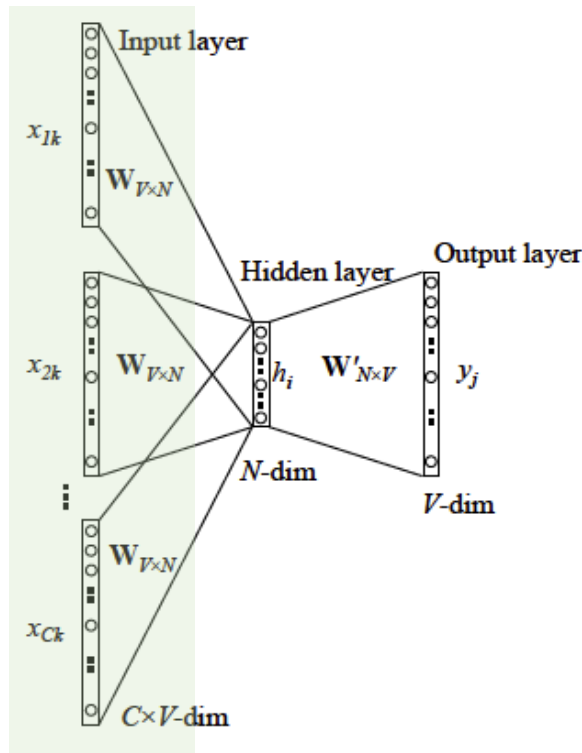
Sentence: “Sydney is the state capital of NSW”



CBOW – Neural Network Architecture

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



1. Initialise each word in a one-hot vector form.

$$x_k = [0, \dots, 0, 1, 0, \dots, 0]$$

2. Use context words ($2m$, based on window size $=m$) as input of the Word2Vec-CBOW model.

$$(x^{c-m}, x^{c-m+1}, \dots, x^{c-1}, x^{c+1}, \dots, x^{c+m-1}, x^{c+m}) \in \mathbb{R}^{|V|}$$

3. Has two Parameter Matrices:

1) Parameter Matrix (from Input Layer to Hidden/Projection Layer)

$$W \in \mathbb{R}^{V \times N}$$

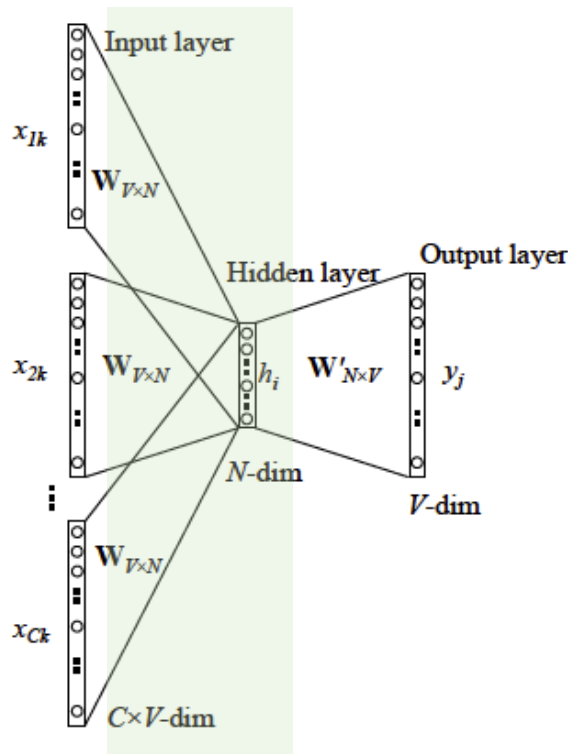
2) Parameter Matrix (to Output Layer)

$$W' \in \mathbb{R}^{N \times V}$$

CBOW – Neural Network Architecture

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



4. Initial words are represented in one hot vector so multiplying a **one hot vector** with $W_{V \times N}$ will give you a $1 \times N$ (embedded word) vector.

$$\text{e.g. } [0 \ 1 \ 0 \ 0] \times \begin{bmatrix} 10 & 2 & 18 \\ 15 & 22 & 3 \\ 25 & 11 & 19 \\ 4 & 7 & 22 \end{bmatrix} = [15 \ 22 \ 3]$$

$$(v_{c-m} = Wx^{c-m}, \dots, v_{c+m} = Wx^{c+m}) \in \mathbb{R}^n$$

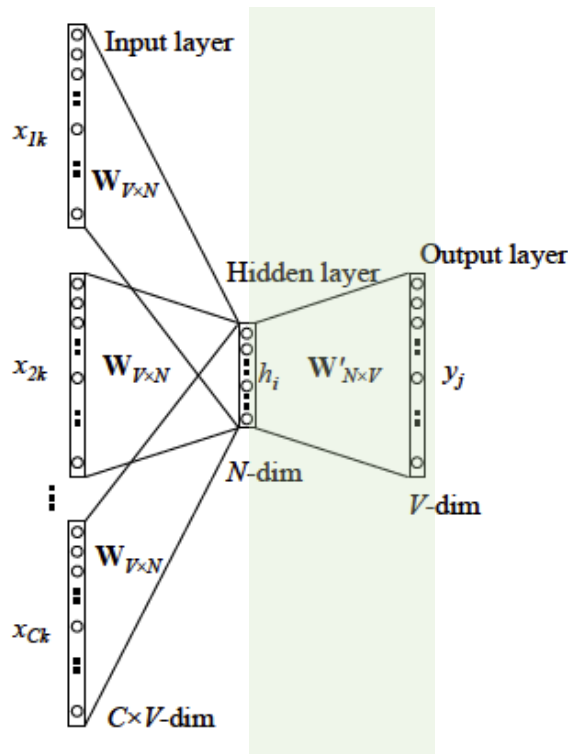
5. Average those $2m$ embedded vectors to calculate the value of the Hidden Layer.

$$\hat{v} = \frac{v_{c-m} + v_{c-m+1} + \dots + v_{c+m}}{2m}$$

CBOW – Neural Network Architecture

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



6. Calculate the score value for the output layer. The higher score is produced when words are closer.

$$\mathbf{z} = \mathbf{W}' \times \hat{\mathbf{v}} \in \mathbb{R}^{|V|}$$

7. Calculate the probability using softmax

$$\hat{y} = \text{softmax}(\mathbf{z}) \in \mathbb{R}^{|V|}$$

8. Train the parameter matrix using **objective function**.

$$H(\hat{y}, y) = - \sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

* Focus on minimising the value

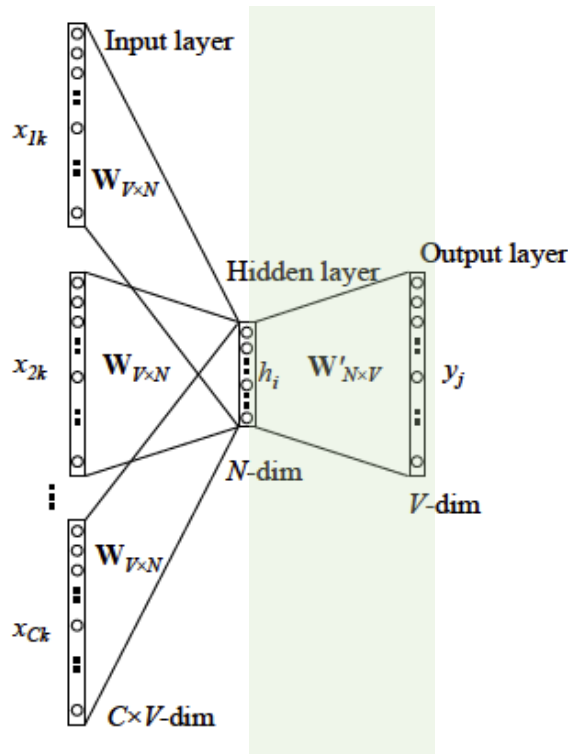
We use an one-hot vector (one 1, the rest 0) so it will be calculated in only one.

$$H(\hat{y}, y) = -y_j \log(\hat{y}_j)$$

CBOW – Neural Network Architecture

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



8-1. Optimization Objective Function can be presented:

$$\begin{aligned}
 \text{minimize } J &= -\log P(w_c | w_{c-m}, \dots, w_{c+m}) \\
 &= -\log P(u_c | v) \\
 &= -\log \frac{\exp(u_c^\top \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^\top \hat{v})} \\
 &= -u_c^{\text{intercal}} \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^\top \hat{v})
 \end{aligned}$$

**This optimization objective will be learned more details in the lecture 3.*

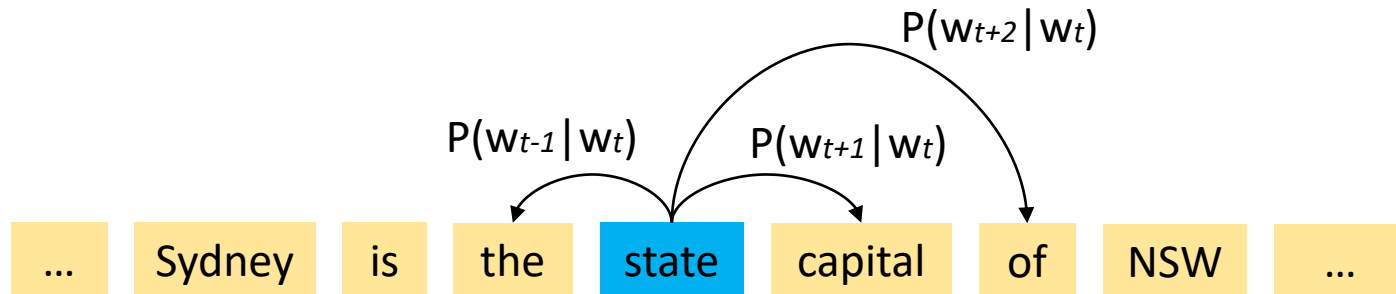
3

Prediction based Word representation

Skip Gram

Predict context (“outside”) words (position independent) given center word

Sentence: “Sydney is the state capital of NSW”

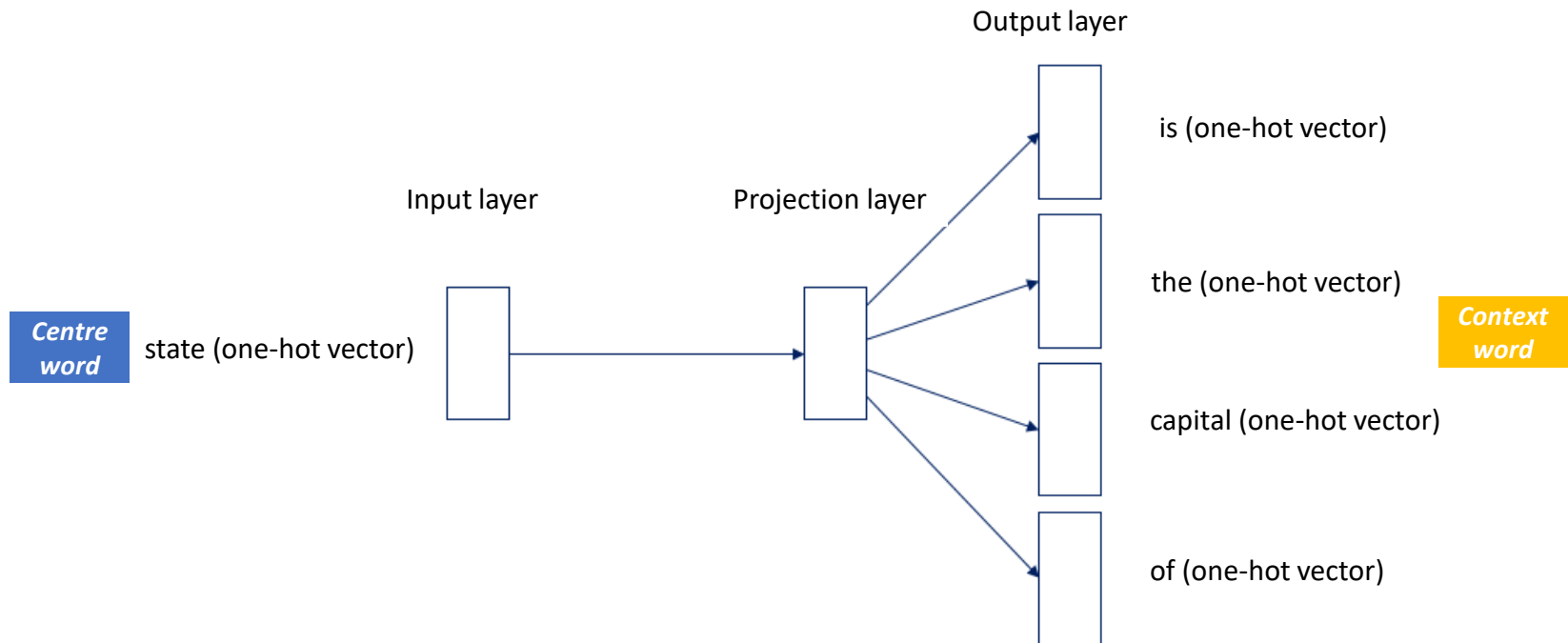


Prediction based Word representation

Skip Gram

Predict context (“outside”) words (position independent) given center word

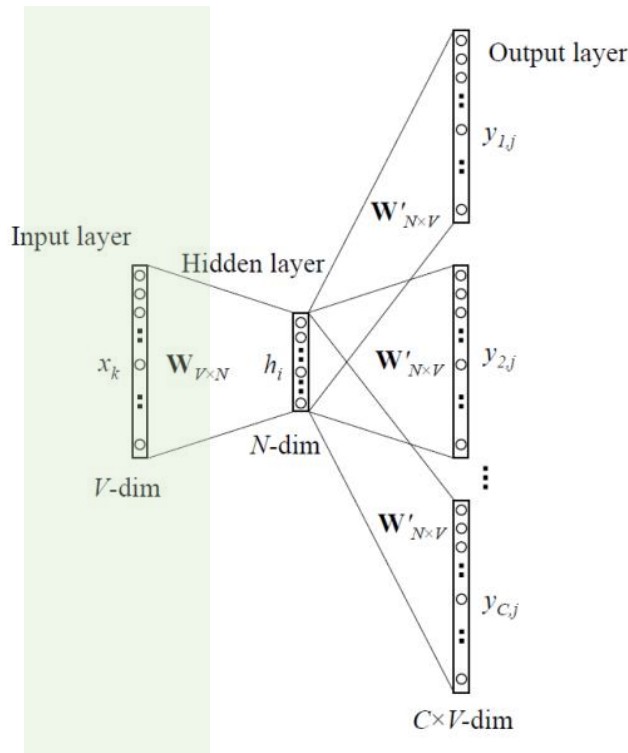
Sentence: “Sydney is the state capital of NSW”



Skip Gram – Neural Network Architecture

Predict context (“outside”) words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



1. Initialise the centre word in a one-hot vector form.

$$x_k = [0, \dots, 0, 1, 0, \dots, 0]$$

$$x \in \mathbb{R}^{|V|}$$

2. Has two Parameter Matrices:

1) Parameter Matrix (from Input Layer to Hidden/Projection Layer)

$$W \in \mathbb{R}^{V \times N}$$

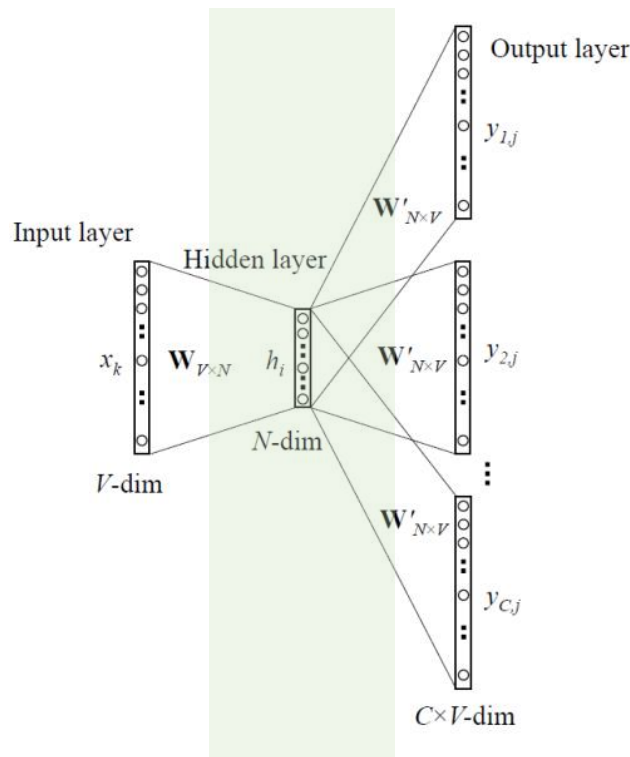
2) Parameter Matrix (to Output Layer)

$$W' \in \mathbb{R}^{N \times V}$$

Skip Gram – Neural Network Architecture

Predict context (“outside”) words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



3. Initial words are represented in one hot vector so multiplying a **one hot vector** with $W_{V \times N}$ will give you a $1 \times N$ (embedded word) vector.

$$\text{e.g. } [0 \ 1 \ 0 \ 0] \times \begin{bmatrix} 10 & 2 & 18 \\ 15 & 22 & 3 \\ 25 & 11 & 19 \\ 4 & 7 & 22 \end{bmatrix} = [15 \ 22 \ 3]$$

$$v_c = W_x \in \mathbb{R}^n \text{ (as there is only one input)}$$

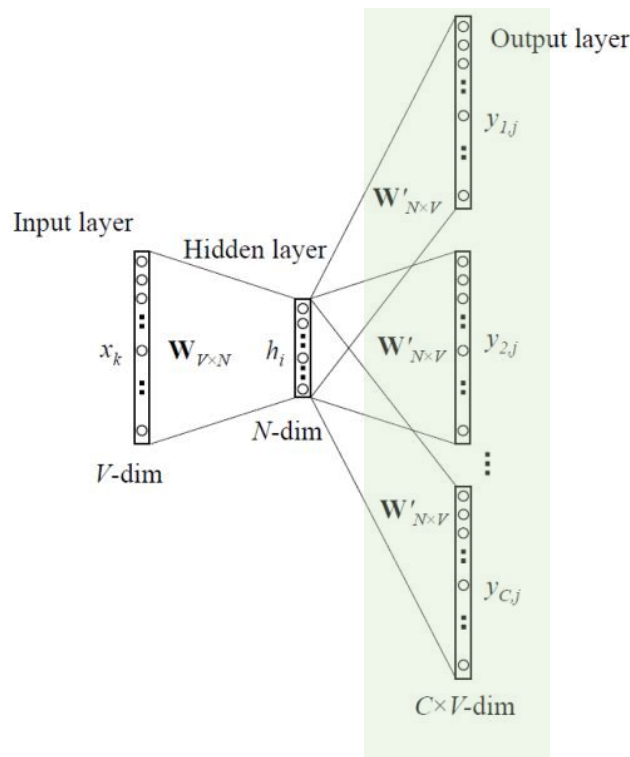
4. Calculate the score value for the output layer by multiplying the parameter matrix W'

$$z = W' v_c$$

Skip Gram – Neural Network Architecture

Predict context (“outside”) words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



5. Calculate the probability using softmax
 $\hat{y} = \text{softmax}(\mathbf{z})$

6. Calculate $2m$ probabilities as we need to predict $2m$ context words.

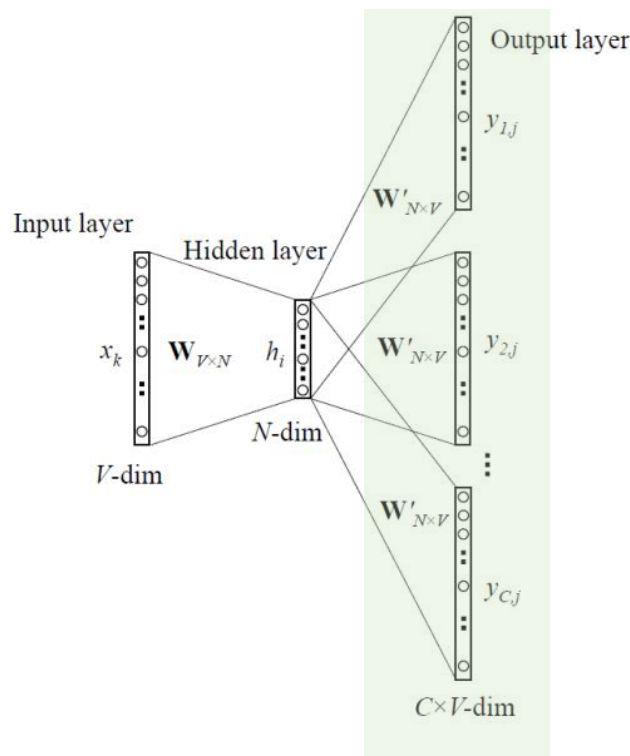
$$\hat{y}_{c-m}, \dots, \hat{y}_{c-1}, \hat{y}_{c+1}, \dots, \hat{y}_{c+m}$$

and compare with the ground truth (one-hot vector)
 $y^{(c-m)}, \dots, y^{(c-1)}, y^{(c+1)}, \dots, y^{(c+m)}$

Skip Gram – Neural Network Architecture

Predict context (“outside”) words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



8. As in CBOW, use an objective function for us to evaluate the model. A key difference is that we invoke a Naïve Bayes assumption to break out the probabilities. It is a strong conditional independence assumption. Given the centre word, all output words are completely independent.

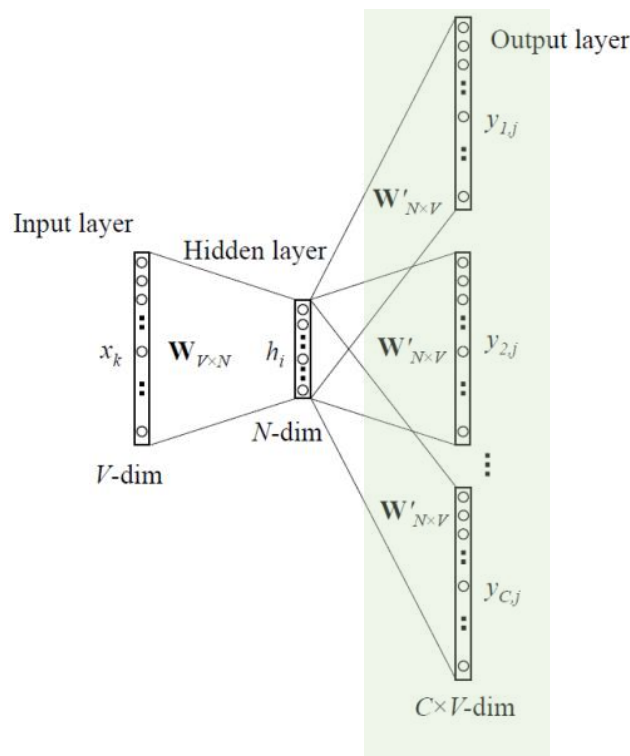
$$\text{minimize } J = -\log P(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m} | w_c)$$

$$\begin{aligned} &= -\log \prod_{j=0, j \neq m}^{2m} P(w_{c-m+j} | w_c) \\ &= -\log \prod_{j=0, j \neq m}^{2m} \frac{\exp(u_{c-m+j}^\top v_c)}{\sum_{k=1}^{|V|} \exp(u_k^\top v_c)} \\ &= -\sum_{j=0, j \neq m}^{2m} u_{c-m+j}^\top v_c + 2m \log \sum_{k=1}^{|V|} \exp(u_k^\top v_c) \end{aligned}$$

Skip Gram – Neural Network Architecture

Predict context (“outside”) words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



8-1. With this objective function, we can compute the gradients with respect to the unknown parameters and at each iteration update them via Stochastic Gradient Descent

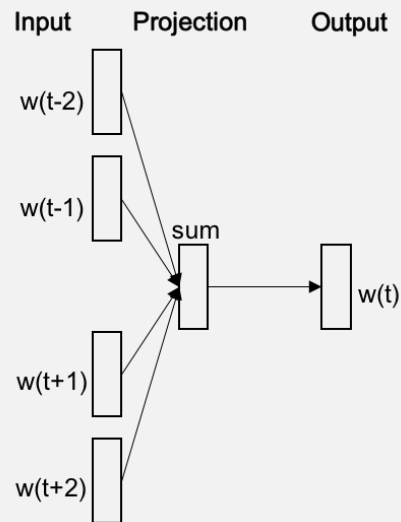
$$\begin{aligned}
 J &= - \sum_{j=0, j \neq m}^{2m} \log P(u_{c-m+j} | v_c) \\
 &= \sum_{j=0, j \neq m}^{2m} H(\hat{y}, y_{c-m+j})
 \end{aligned}$$

**This Stochastic Gradient Descent will be learned details in the lecture 3.*

CBOW vs Skip Gram Overview

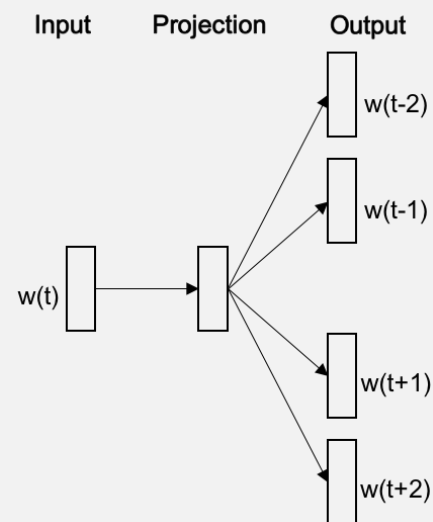
CBOW

*Predict center word
from (bag of) context words*



Skip-gram

*Predict context words
given center word*



Prediction based Word representation

Key Parameter (1) for Training methods: Window Size

Different tasks are served better by different window sizes.

Smaller window sizes (2-15) lead to embeddings where high similarity scores between two embeddings indicates that the words are interchangeable.

Larger window sizes (15-50, or even more) lead to embeddings where similarity is more indicative of relatedness of the words

Key Parameter (2) for Training methods: Negative Samples

The number of negative samples is another factor of the training process.

Negative samples to our dataset – samples of words that are not neighbors

Negative sample: 2

<i>Input word</i>	<i>Output word</i>	<i>Target</i>
eat	mango	1
eat	exam	0
eat	tobacco	0

**1= Appeared, 0=Not Appeared*

Negative sample: 5

<i>Input word</i>	<i>Output word</i>	<i>Target</i>
eat	mango	1
eat	exam	0
eat	tobacco	0
eat	pool	0
eat	supervisor	0

The original paper prescribes **5-20** as being a good number of negative samples. It also states that **2-5** seems to be enough when you have a large enough dataset.

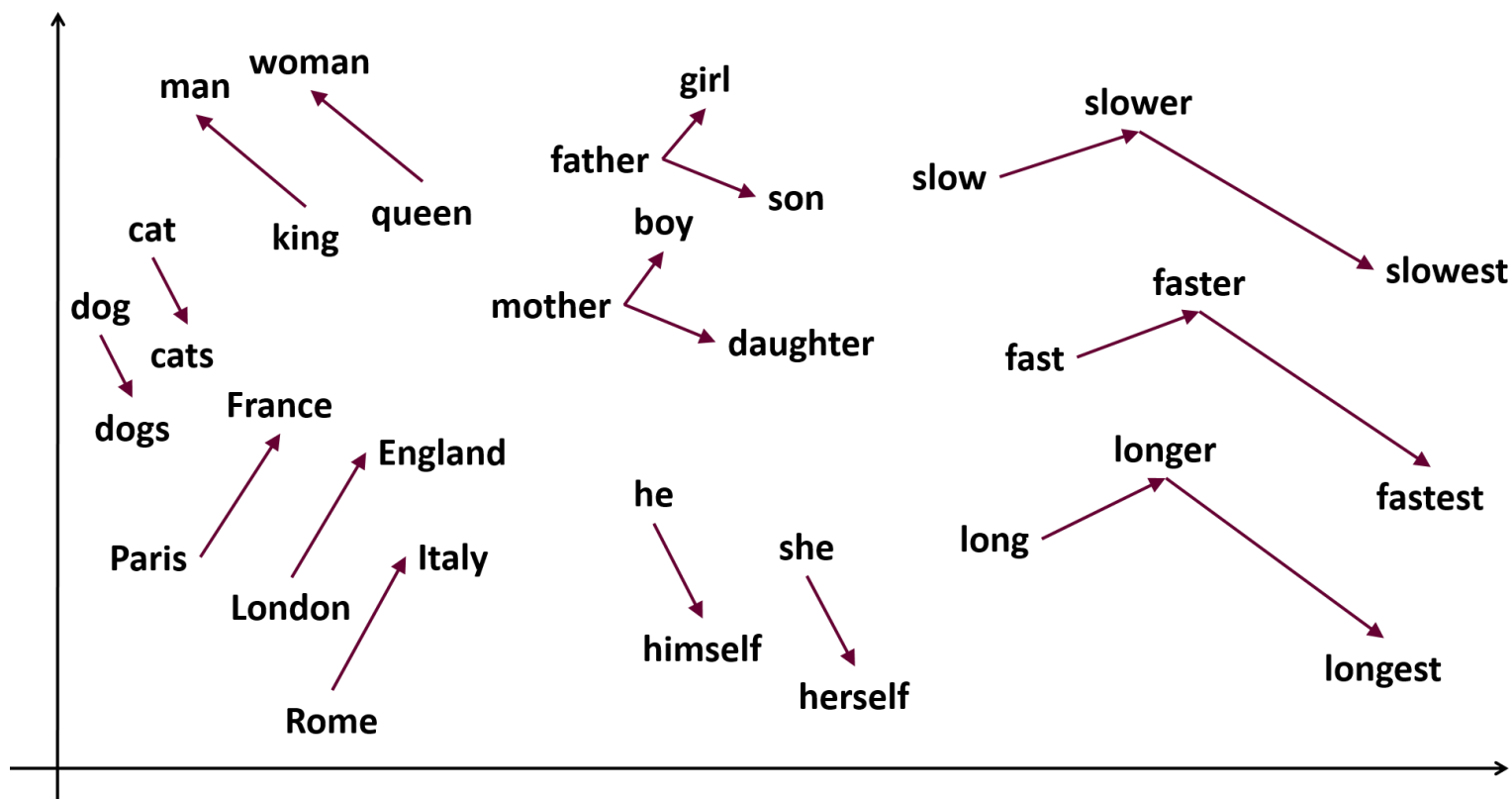
Word2Vec Overview

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

Idea:

- 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

Let's try some Word2Vec!



Gensim: <https://radimrehurek.com/gensim/models/word2vec.html>

Resources: <https://wit3.fbk.eu/>

<https://github.com/3Top/word2vec-api#where-to-get-a-pretrained-models>

Limitation of Word2Vec

Issue#1: Cannot cover the morphological similarity

- Word2vec represents every word as an independent vector, even though many words are morphologically similar, like: teach, teacher, teaching

Issue#2: Hard to conduct embedding for rare words

- Word2vec is based on the Distribution hypothesis. Works well with the frequent words but does not embed the rare words.

(same concept with the under-fitting in machine learning)

Issue#3: Cannot handle the Out-of-Vocabulary (OOV)

- Word2vec does not work at all if the word is not included in the Vocabulary

Prediction based Word representation

FastText

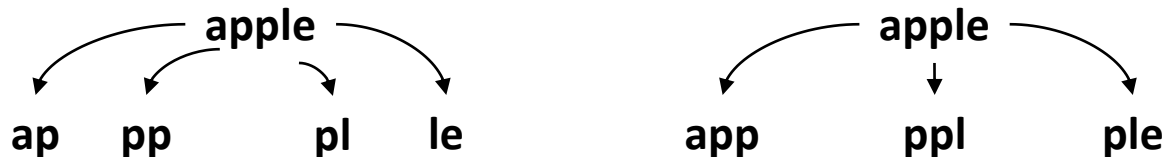
- Deal with this Word2Vec Limitation
- Another Way to transfer *WORDS* to *VECTORS*

fastText

- FastText is a library for learning of word embeddings and text classification created by Facebook's AI Research lab. The model allows to create an unsupervised learning or supervised learning algorithm for obtaining vector representations for words.
- Extension to Word2Vec
 - Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words)

FastText with N-gram Embeddings

- N-grams are simply all combinations of adjacent words or letters of length n that you can find in your source text. For example, given the word *apple*, all 2-grams (or “bigrams”) are ***ap***, ***pp***, ***pl***, and ***le***
- The tri-grams ($n=3$) for the word *apple* is ***app***, ***ppl***, and ***ple*** (ignoring the starting and ending of boundaries of words). The word embedding vector for *apple* will be the sum of all these n -grams.



- After training the Neural Network (either with skip-gram or CBOW), we will have word embeddings for all the n -grams given the training dataset.
- Rare words can now be properly represented since it is highly likely that some of their n -grams also appears in other words.

Word2Vec VS FastText

Find synonym with Word2vec

```
from gensim.models import Word2Vec
cbow_model = Word2Vec(sentences=result, size=100, window=5, min_count=5, workers=4, sg=0)

a=cbow_model.wv.most_similar("electrofishing")
pprint.pprint(a)
```

Find synonym with FastText

```
from gensim.models import FastText
FT_model = FastText(sentences=result, size=100, window=5, min_count=5, workers=4, sg=0)

a=FT_model.wv.most_similar("electrofishing")
pprint.pprint(a)
```

Global Vectors (GloVe)

- Deal with this Word2Vec Limitation

*“Methods like skip-gram may do better on the analogy task, but they poorly utilize the statistics of the corpus since they train on separate local context windows instead of on **global co-occurrence counts**.”*

(PeddingLon et al., 2014)

- Focus on the Co-occurrence

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k ice)/P(k steam)$	8.9	8.5×10^{-2}	1.36	0.96

Prediction based Word representation

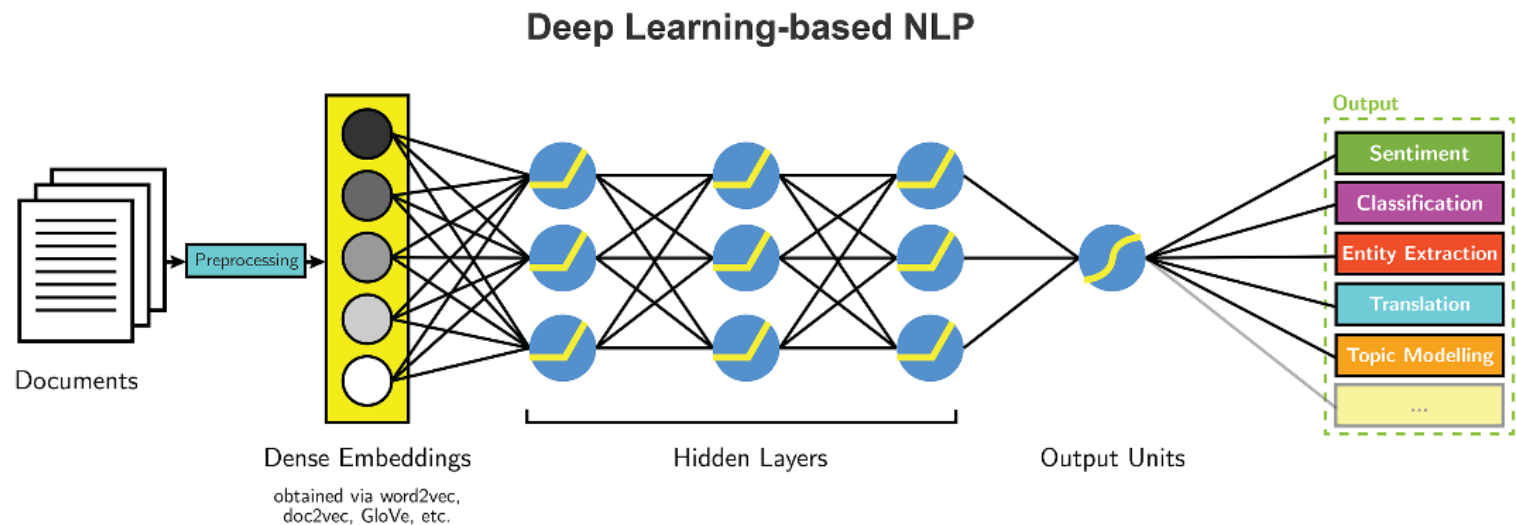
Limitation of Prediction based Word Representation

- I like _____
apple banana fruit
- Training dataset reflect the word representation result
 - The word similarity of the word 'software' the model learned by Google News corpus can be different from the one from Twitter.

Word Embeddings

- Finalisation!

Machine Learning/ Deep Learning for Natural Language Processing



Reference for this lecture

- Deng, L., & Liu, Y. (Eds.). (2018). Deep Learning in Natural Language Processing. Springer.
- Rao, D., & McMahan, B. (2019). Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning. " O'Reilly Media, Inc."
- Manning, C. D., Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT press.
- Manning, C 2017, Introduction and Word Vectors, Natural Language Processing with Deep Learning, lecture notes, Stanford University
- Images: <http://jalammar.github.io/illustrated-word2vec/>

Word2vec

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

FastText

- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135-146.
- Mikolov, T., Grave, E., Bojanowski, P., Puhersch, C., & Joulin, A. (2017). Advances in pre-training distributed word representations. arXiv preprint arXiv:1712.09405.