Session 3

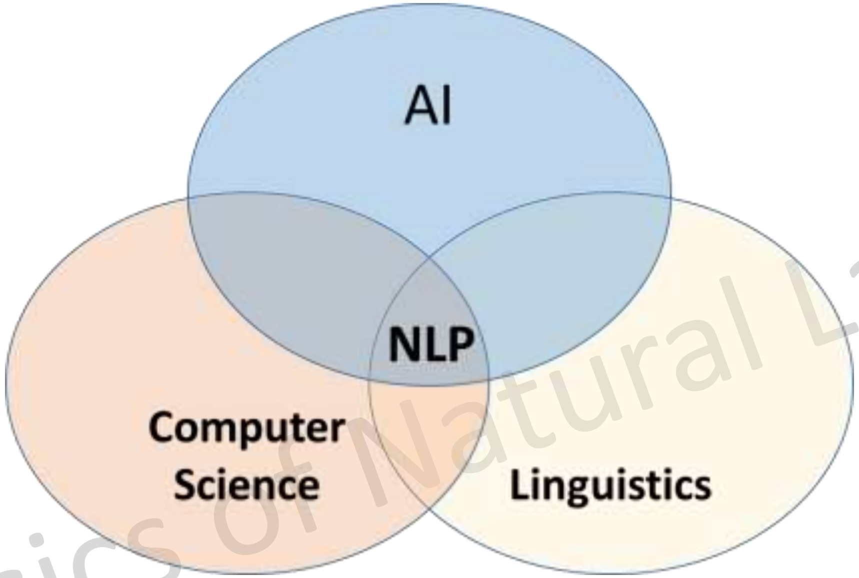
Basics of Machine Learning

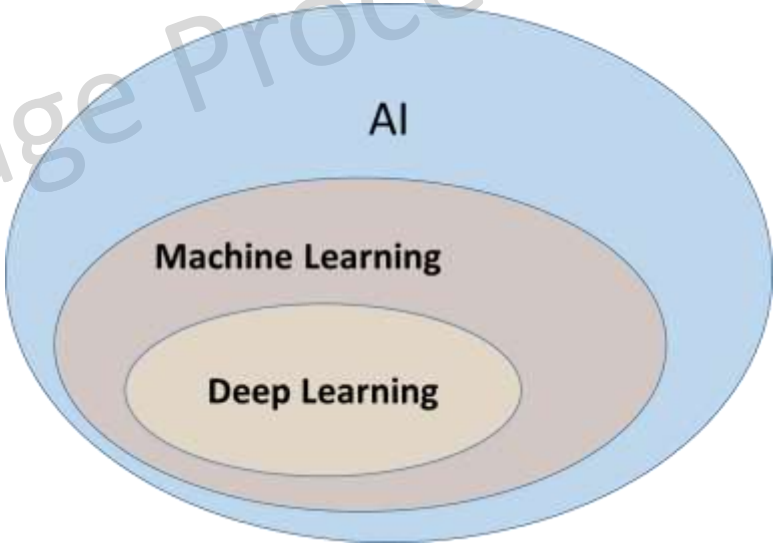
Azam Rabiee, PhD

September 6, 2020

Roadmap

We will review ML/DL methods for NLP





Outline

Session 1: Introduction

- Applications
- Tasks
- Approaches

Session 2. Basics of Linguistics

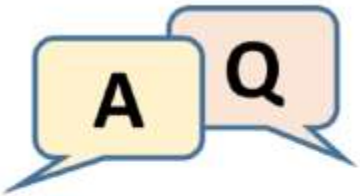
Session 3. Basics of ML

Session 4 (Lab). Effective Word Representation by python

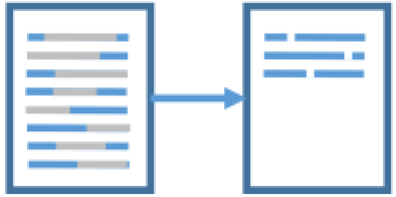
Review: Applications

Machine Translation

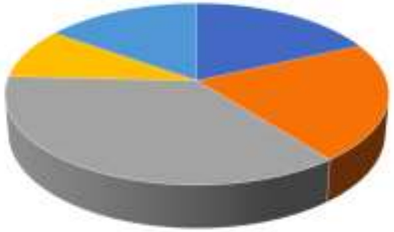
Sentiment Analysis



Question Answering



Automatic Summarization



AI Marketing

Text / Document Classification

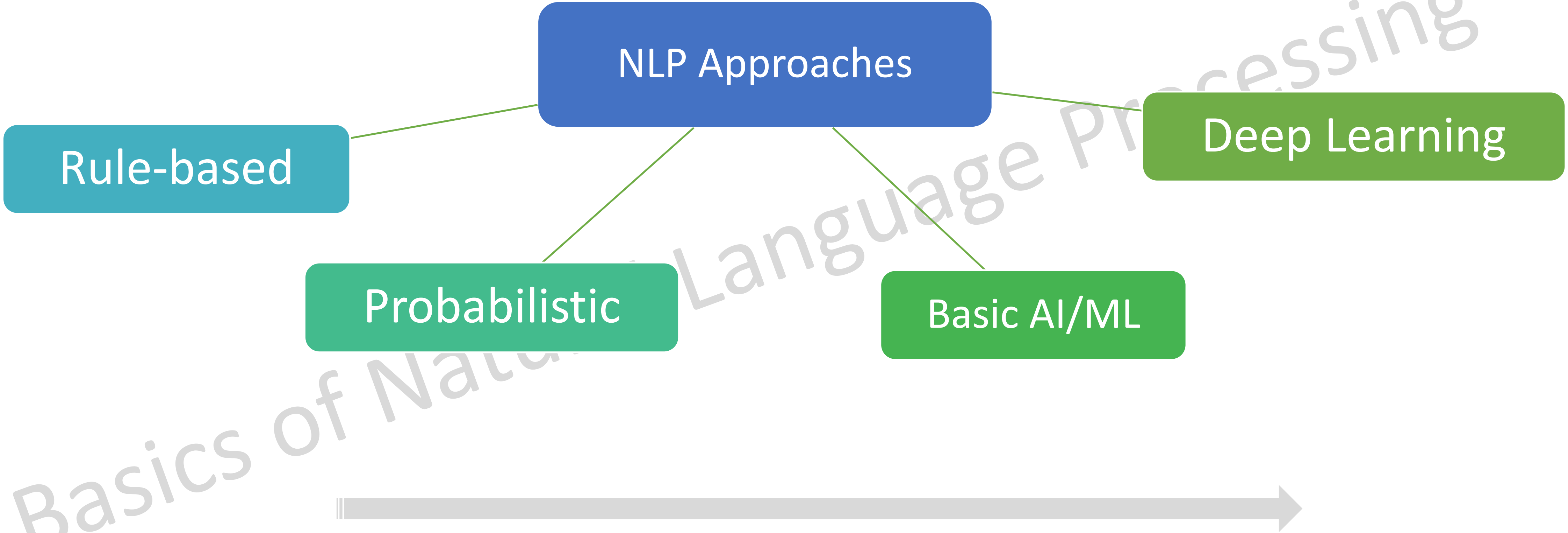
Speech Recognition

Handwritten Character Recognition

Spell Checking

Review: Tasks

Approaches


NLP Approaches
Rule-based
Probabilistic
Basic AI/ML
Deep Learning

Outline

Session 1: Introduction

Session 2. Basics of Linguistics

- Components
- Challenges
- Vectorization

Session 3. Basics of ML

Session 4 (Lab). Effective Word Representation by python

Components of NLP

Lexical Analysis

Tokenization	PoS tagging	Stemming/lemmatization

↓

Syntactic Analysis

Chunking / Parsing	Chinking

↓

Semantic Analysis

↓

Discourse Analysis

↓

Pragmatic Analysis

Current NLP Challenges

- Tokenization
- PoS Tagging
- Named Entity Recognition (NER)
- Context understanding
- Ambiguity in speech
- Semantic analysis
- Building an appropriate vocabulary
- Linking the components of a created vocabulary

Steps of ML/DL Projects with Text Data

Text Collection

Preprocessing

- Tokenization
- Stop words removal
- Stemming / Lemmatization
- …

Vectorization

- Convert to numbers

Modeling

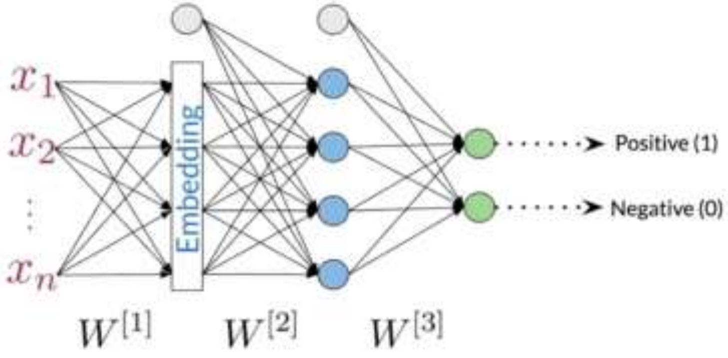
Vectorization

Some Vectorization Approaches

- Bag-of-Words
- TF-IDF
- Embedding Layer
- word2vec

	and	bus	by	expensive	flight	full	is	jim	late	pam	the	train	travel	was
S1	1	1	1	0	0	0	0	1	0	1	0	0	1	0
S2	0	0	0	0	0	0	0	0	1	0	1	1	0	1
S3	0	0	1	1	2	1	1	0	0	0	1	0	1	1

	and	bus	by	expensive	flight	full	is	jim	late	pam	the	train	travel	was
D1	0.4	0.4	0.3	0	0	0	0	0.4	0	0.4	0	0	0.3	0
D2	0	0	0	0	0	0	0	0	0.5	0	0.4	0.5	0	0.4
D3	0	0	0.2	0.3	0.6	0.3	0.3	0	0	0	0.2	0	0.2	0.2



Outline

Session 1: Introduction

Session 2. Basics of Linguistics

Session 3. Basics of ML

- word2vec
- Components
- Architectures
- Python

Session 4 (Lab). Effective Word Representation by python

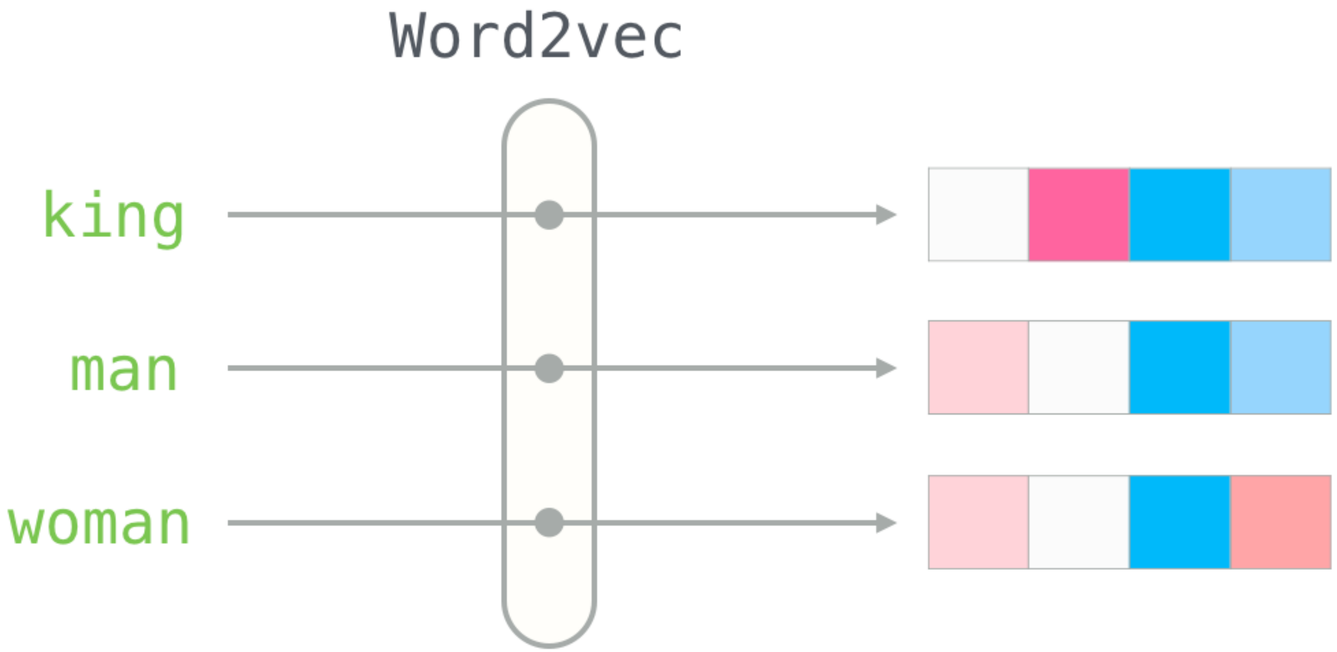
word2vec

Word2vec is a model provided by Google in 2013
for the **effective word embedding**.

[Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013]

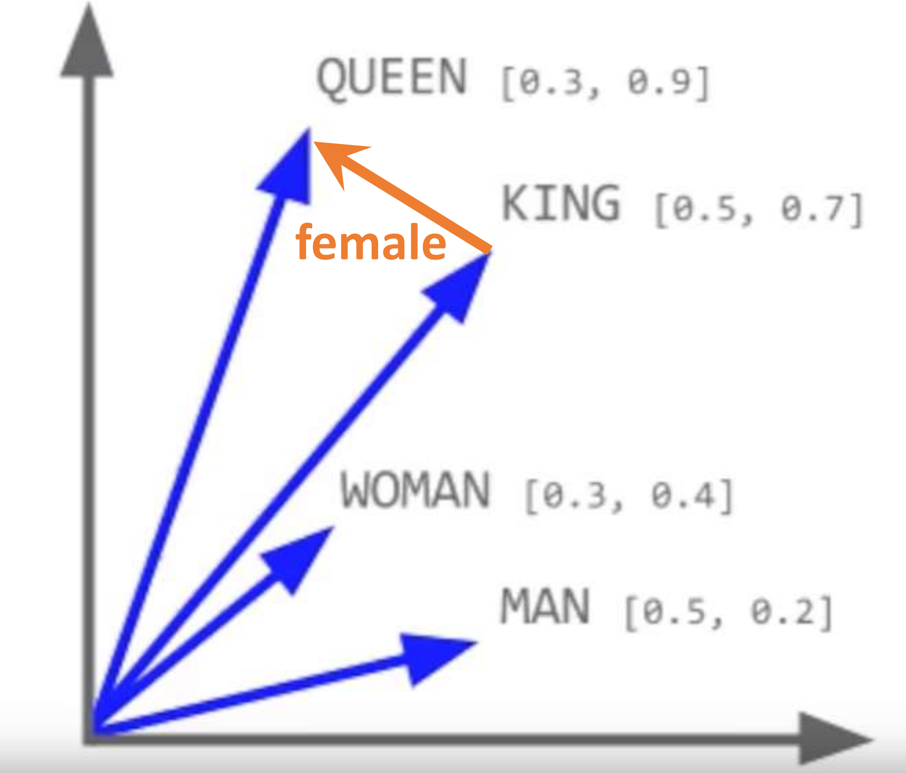
word2vec

Word2vec is a model provided by Google in 2013
for the **effective word embedding**.



[Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013]

Word Embedding



King + female = Queen

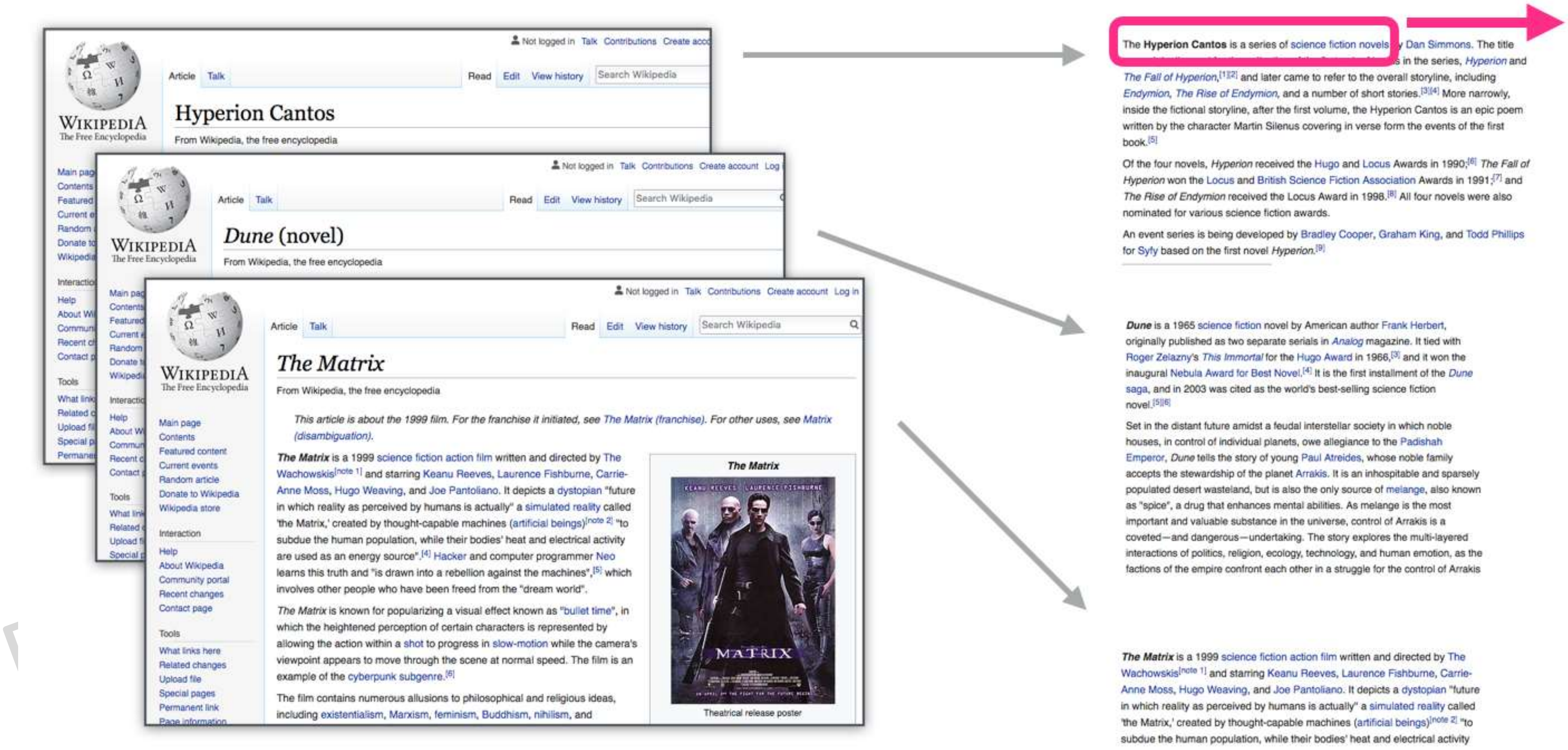
Man + female = Woman

Queen – royal = Woman

Steps of word2vec

Dataset

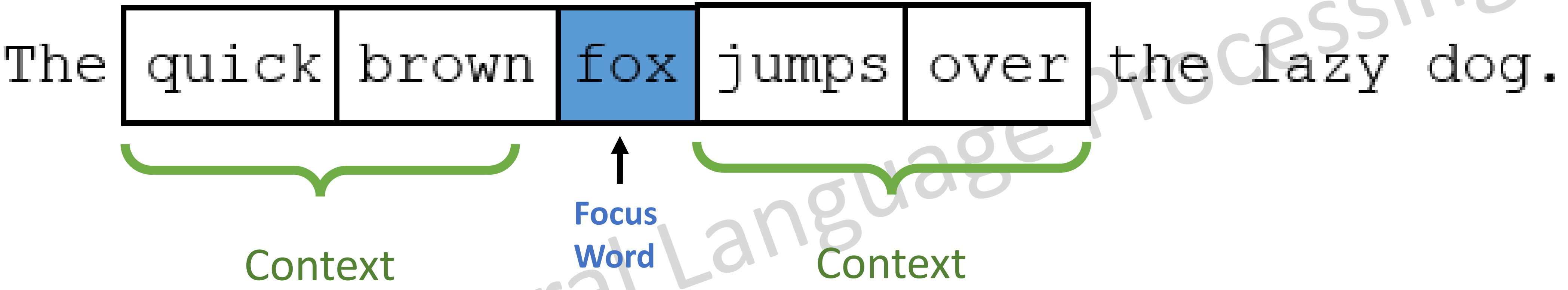
Sliding window across running text



[http://jalammar.github.io/illustrated-word2vec/]

Sliding Window

Window size=5



Example Dataset

Source Text		Training Samples
The quick brown fox jumps over the lazy dog.	⟹	(the, quick) (the, brown)
The **quick** brown fox jumps over the lazy dog.	⟹	(quick, the) (quick, brown) (quick, fox)
The quick **brown** fox jumps over the lazy dog.	⟹	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown **fox** jumps over the lazy dog.	⟹	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

[McCormick, C.: Word2vec Tutorial - The Skip-Gram Model. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/]

We are going to find the best vector representation for words (word embedding). Is it a supervised algorithm?

- Yes
- No

Now with the designed training example, is it a supervised algorithm?

1. Yes
2. No

Training Samples

(the, quick)
(the, brown)

(quick, the)
(quick, brown)
(quick, fox)

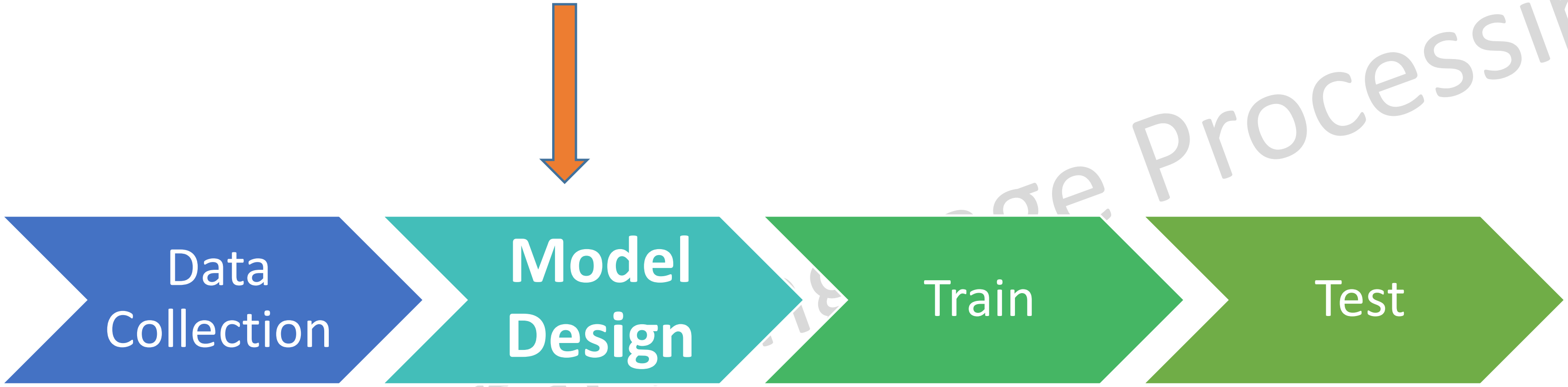
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)

(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

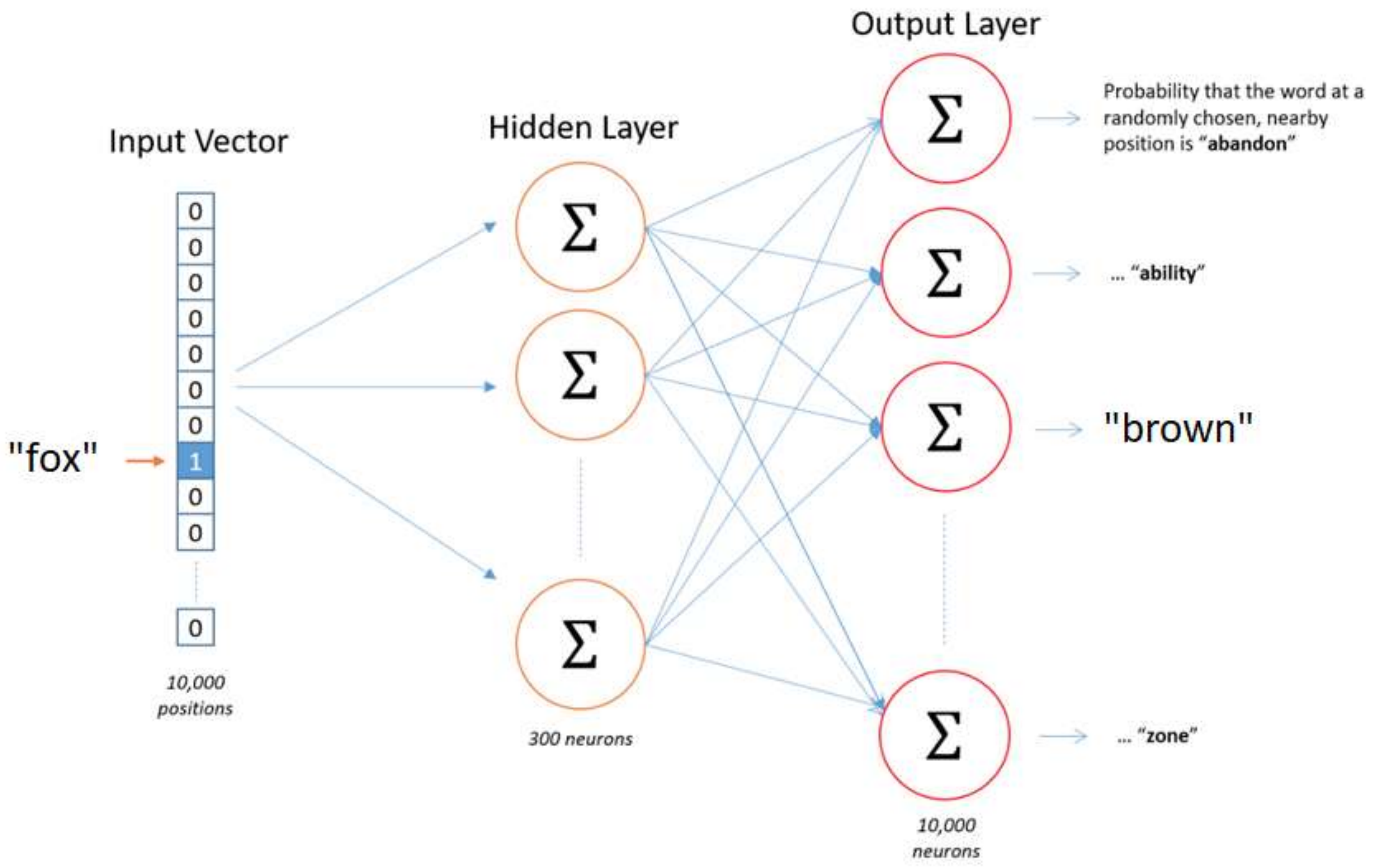
QUICK TIPS

In word2vec, we solve an unsupervised problem in a supervised way.

Steps of word2vec



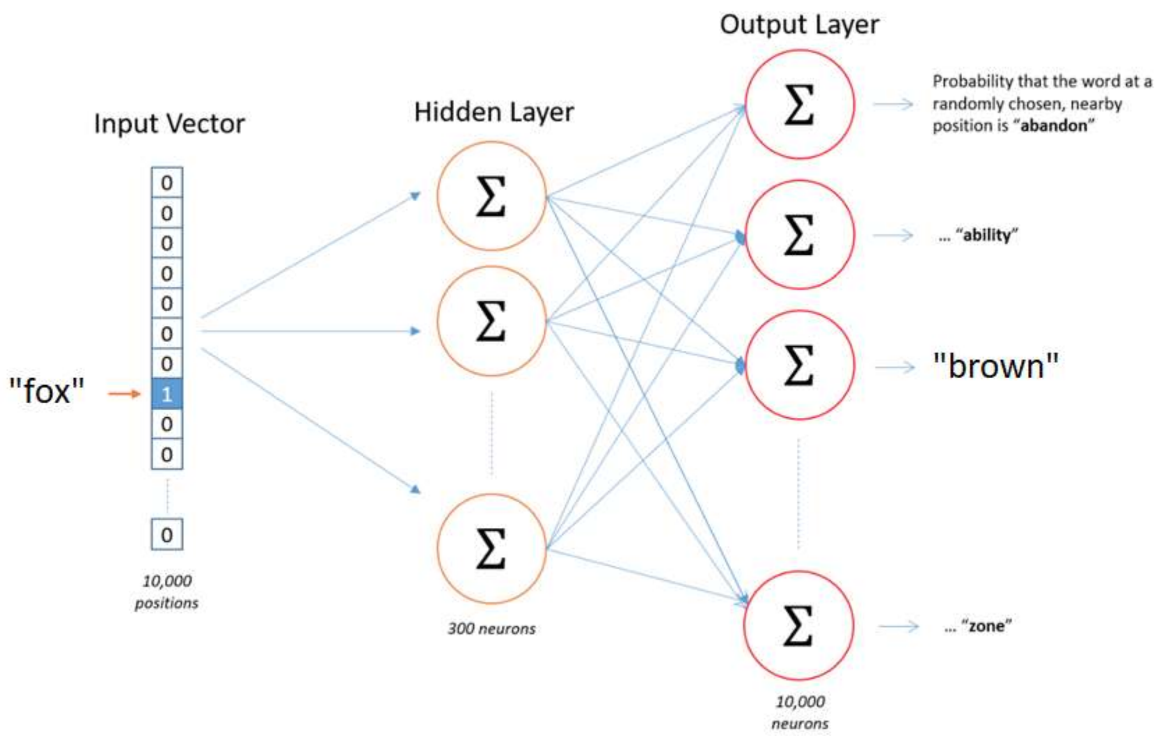
Network Architecture



Suppose that the list of words contains 10,000 unique words and we want to make 300-D word embedding

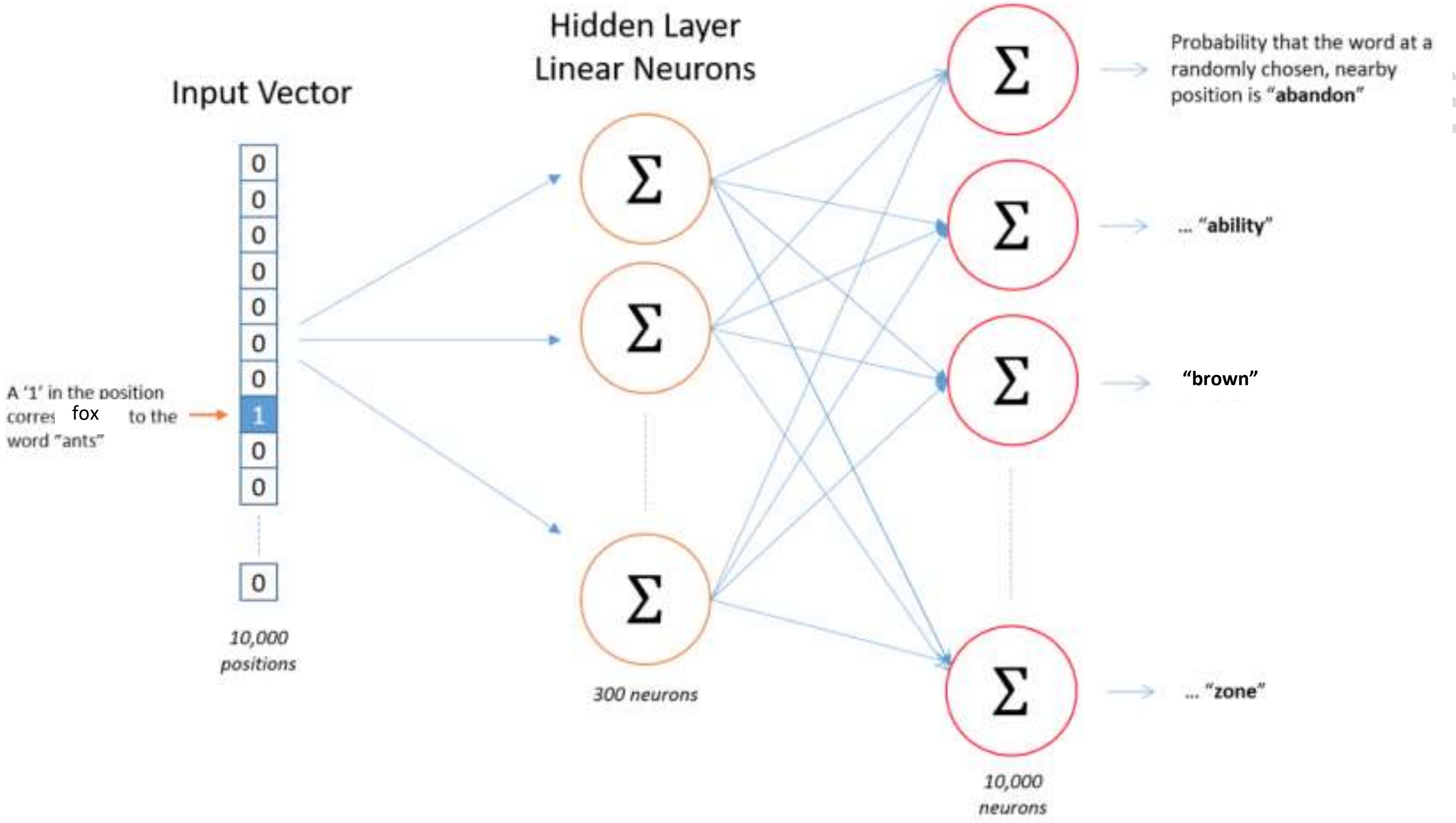
Q: How many weight parameters are in the network?

1. 10,000 x 300 x 2
2. 10,000 x 300 x 10,000
3. None of above



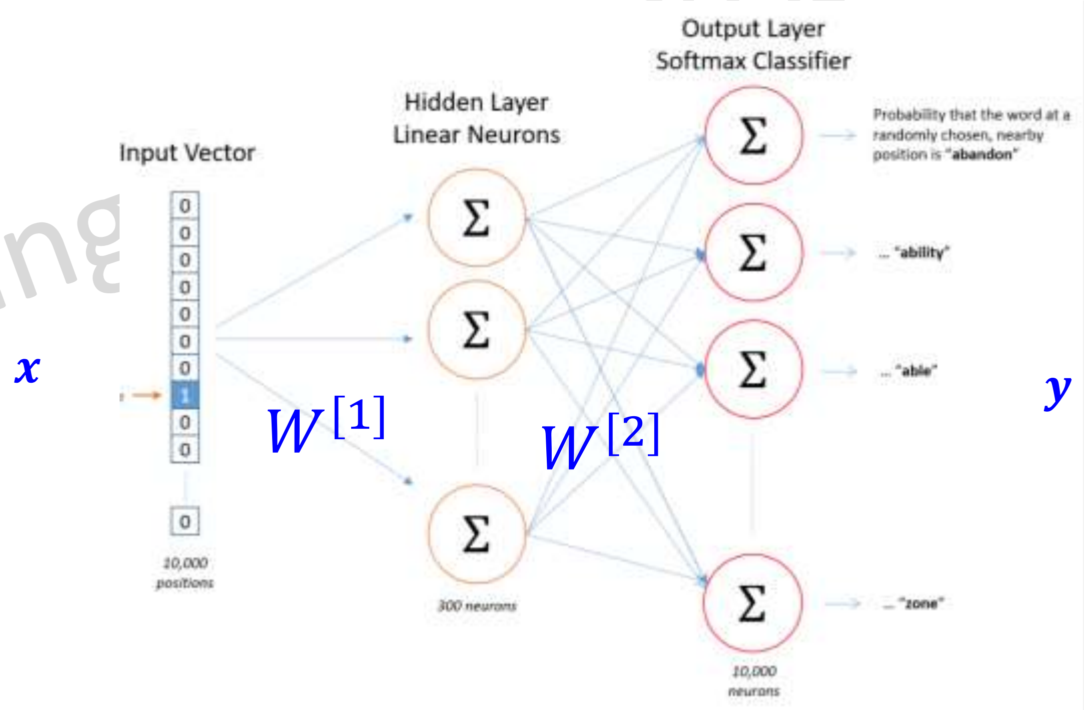
Network Architecture: Activation Functions

Sample: (fox, brown)



[McCormick, C.: Word2vec Tutorial - The Skip-Gram Model. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/]

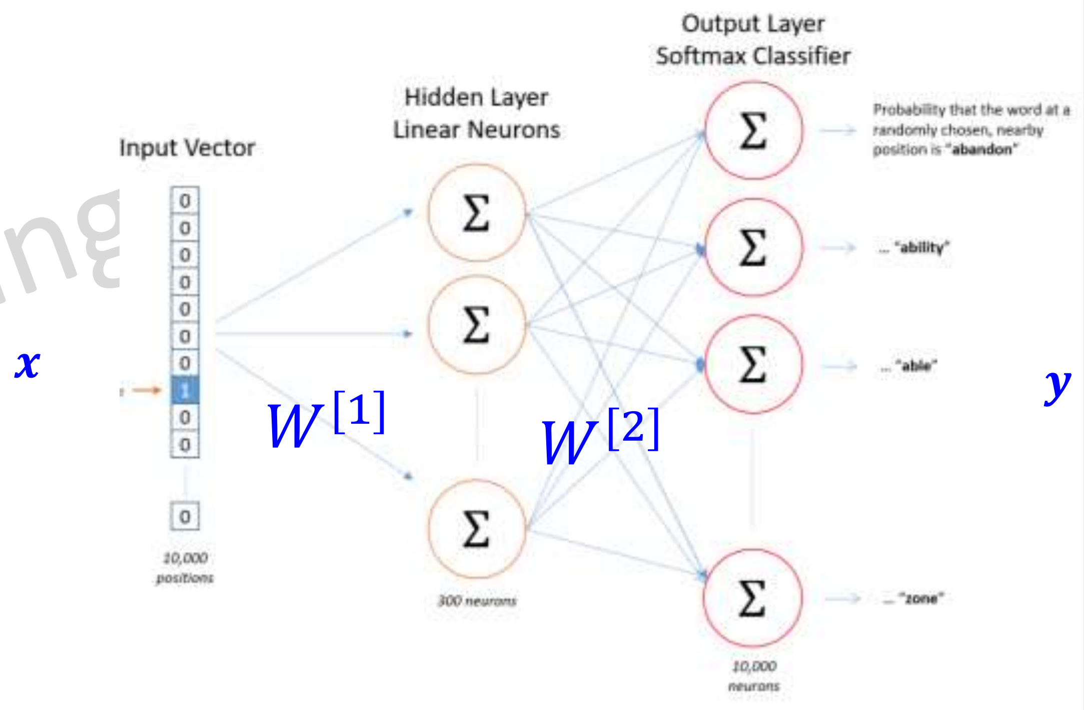
What is forward equations of this network?



What is forward equations of this network?

$$\boldsymbol{a}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{x}$$

$$\boldsymbol{y} = \boldsymbol{a}^{[2]} = softmax(\boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]})$$



Hidden Layer

Weights of the hidden layer

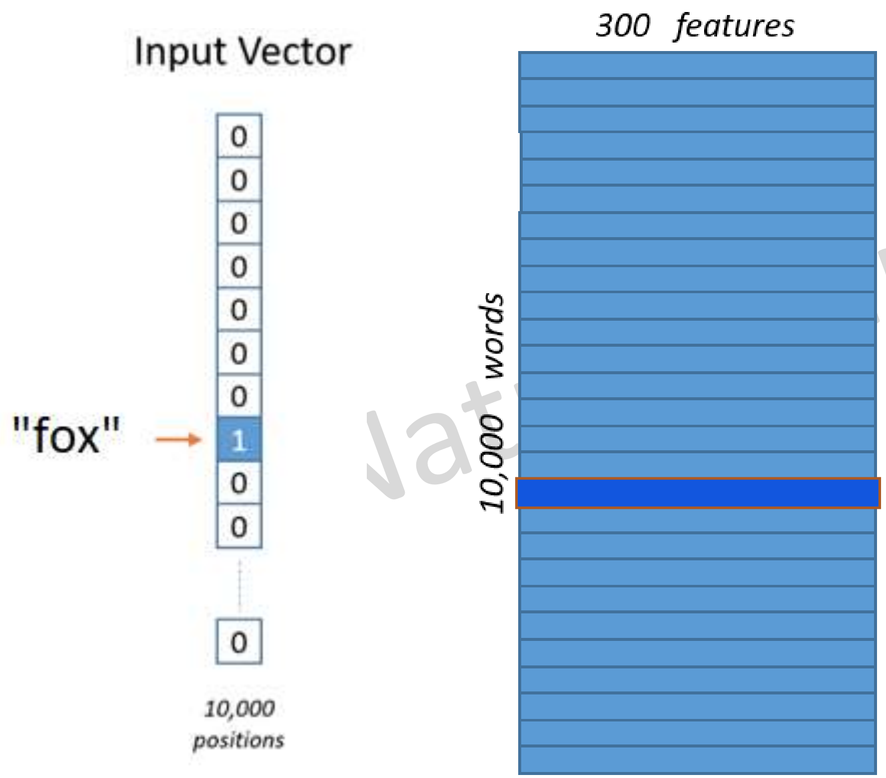
Onehot input

Output of the hidden layer

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

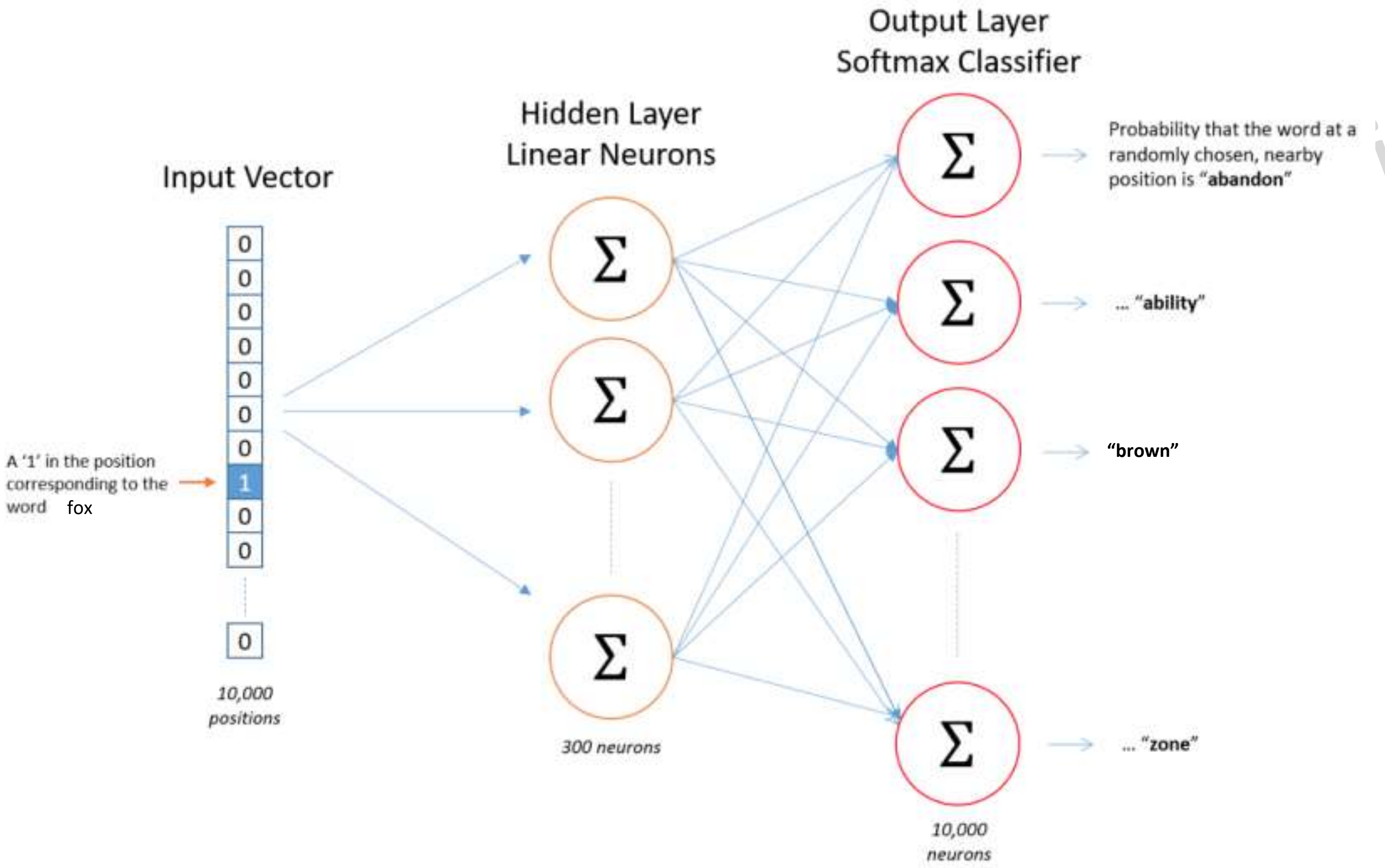
Hidden Layer

Word Vector
Lookup Table!



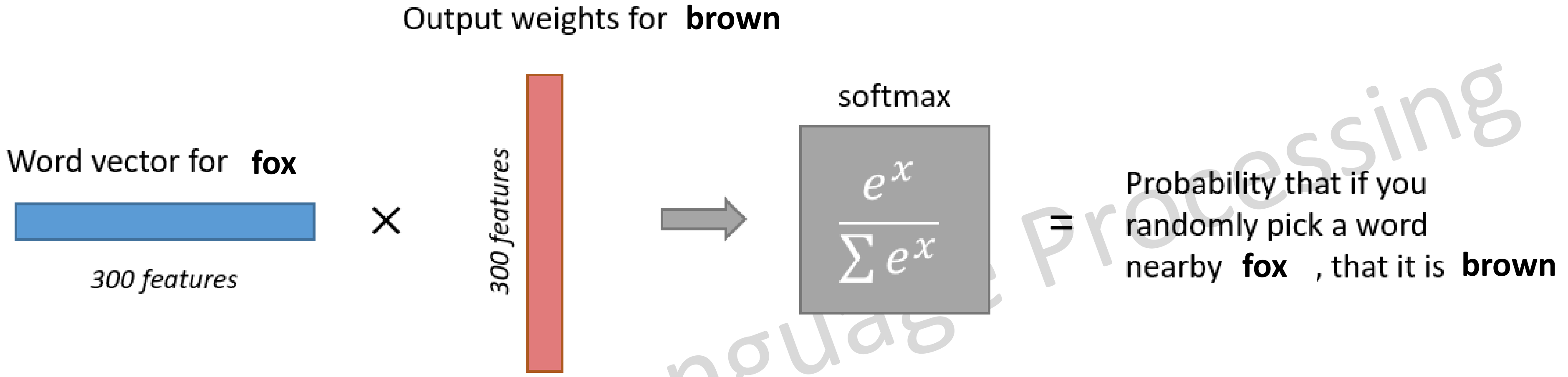
Output of the hidden layer:
embedding vector for “fox”

Network Architecture



[McCormick, C.: Word2vec Tutorial - The Skip-Gram Model. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/]

Output Layer

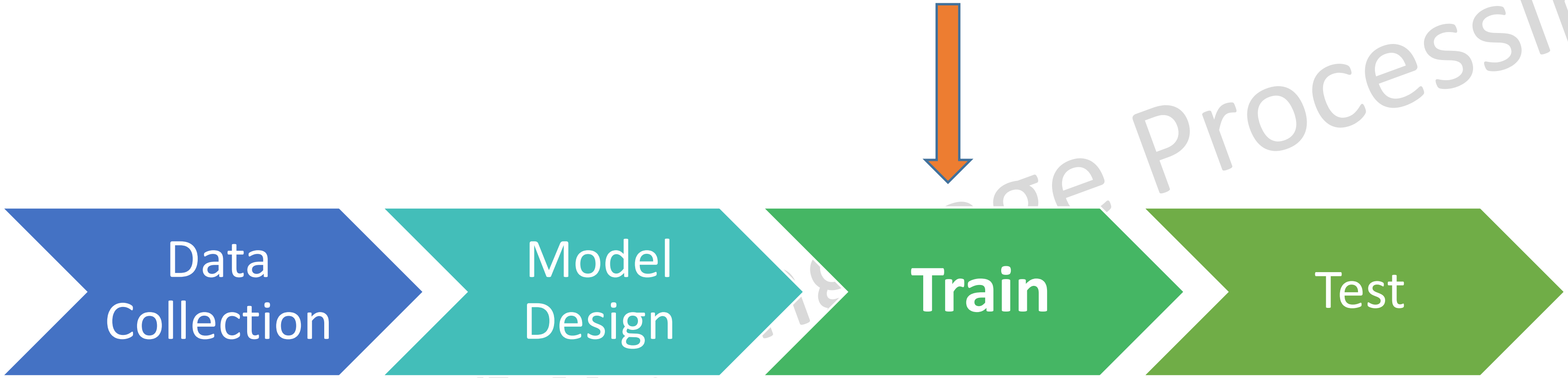


If words "**fox**" and "**brown**" appears many times in the dataset (meaning that they are close enough together in the text), then their embedding vector should be similar.

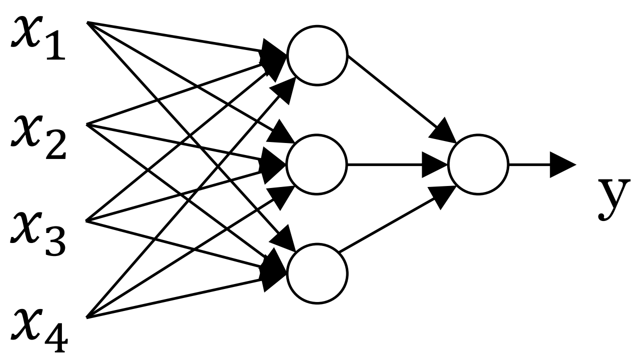
In word2vec, the **similarity** is obtained by maximizing the inner product.

Word2vec outputs very similar results for two words having very similar “contexts” (i.e. they are likely to appear around each other).

Steps of word2vec



Review: Training



$$y = f(x;\theta)$$

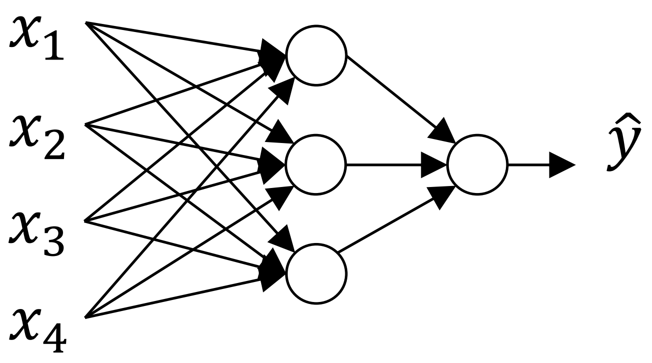
Aim of Training

Given neural network parameters $\boldsymbol{\theta}$, find the value of θ that minimizes cost function

$$J(\theta).$$

Review: Cost Function

Example: Sentiment Analysis



$\boldsymbol{x}$**: tweets**
y: positive (1) / negative (0)

Loss Function:

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

also known as **L2-norm**

Cost Function:

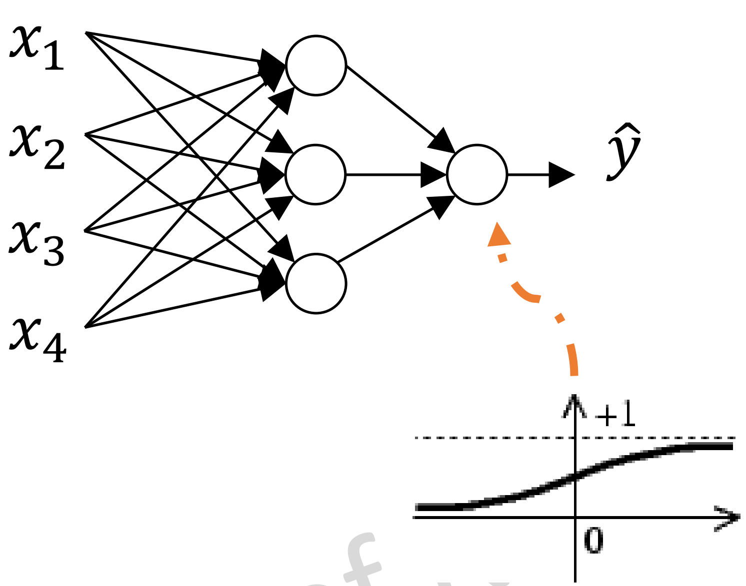
$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

The aim of training:

Minimizing the cost function

Review: Cost Function

Example: Sentiment Analysis



***x*: tweets**
y: positive (1) / negative (0)

<u>**Cross Entropy** Loss Function:</u>

$$\mathcal{L}(\hat{y}, y) = -\{y \log \hat{y} + (1-y)\log(1-\hat{y})\}$$

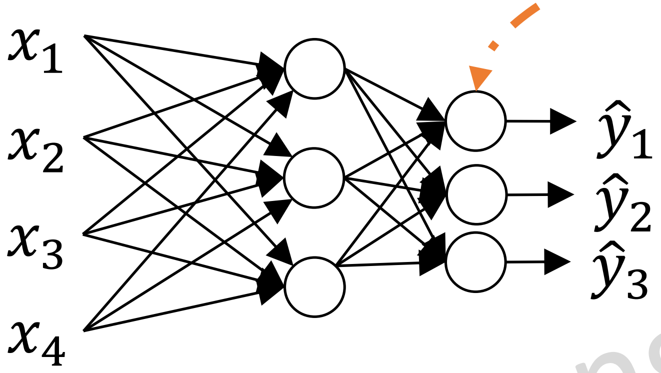
<u>Cost Function:</u>

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

Review: Cost Function

Example: Sentiment Analysis

$\boldsymbol{x}$**: tweets**
y_1: positive
y_2: negative
y_3: neutral



Softmax activation:

$$y_i = f(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

Review: Cost Function

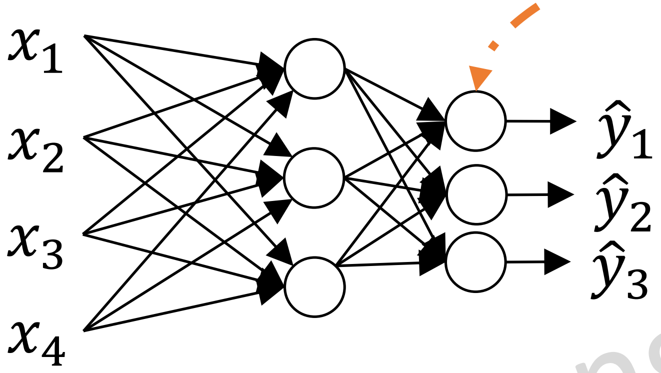
Example: Sentiment Analysis

***x*: tweets**

y_1: positive

y_2: negative

y_3: neutral



Softmax activation:

$$y_i = f(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

Multi-class Cross Entropy Loss Function:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{K} (y_k \log \hat{y}_k + (1 - y_k) \log(1 - \hat{y}_k))$$

We define the **cost function** according to:

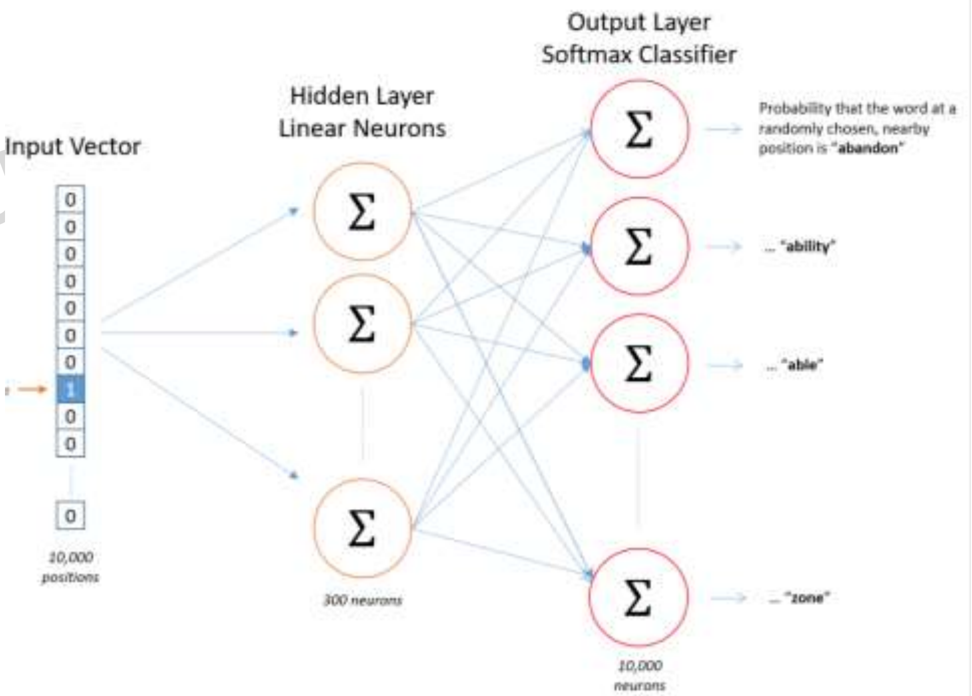
- the output type,
- the task
- the performance measure we expect from the model

Examples:

Output Type	Activation for Output Layer	Cost Function
Real / continuous value	Linear / ReLU	MSE / L1-norm / L2-norm
Binary	Sigmoid / Tanh	Cross-Entropy
Multi-class	Softmax	Multi-class Cross-Entropy

Which cost function is suitable for the word2vec model?

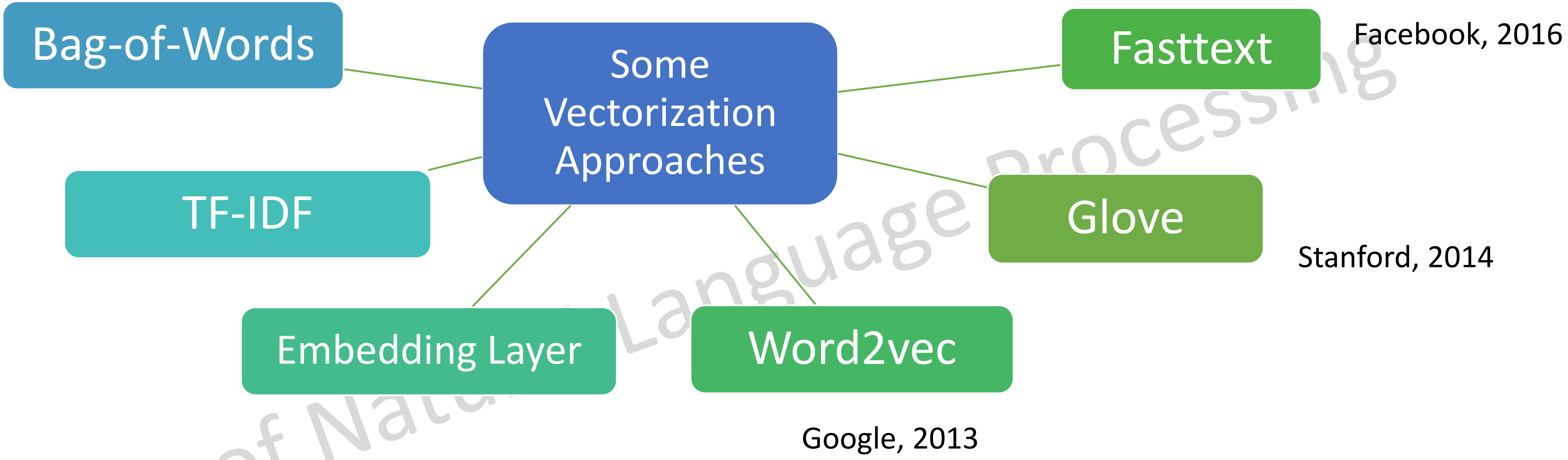
1. MSE / L2-norm
2. Cross entropy
3. Multi-class cross entropy



Steps of word2vec

Train and test phases will be discussed further in the next session via the **Python implementation**

Vectorization



Steps of ML/DL Projects with Text Data

Text Collection

Preprocessing

- Tokenization
- Stop words removal
- Stemming / Lemmatization
- …

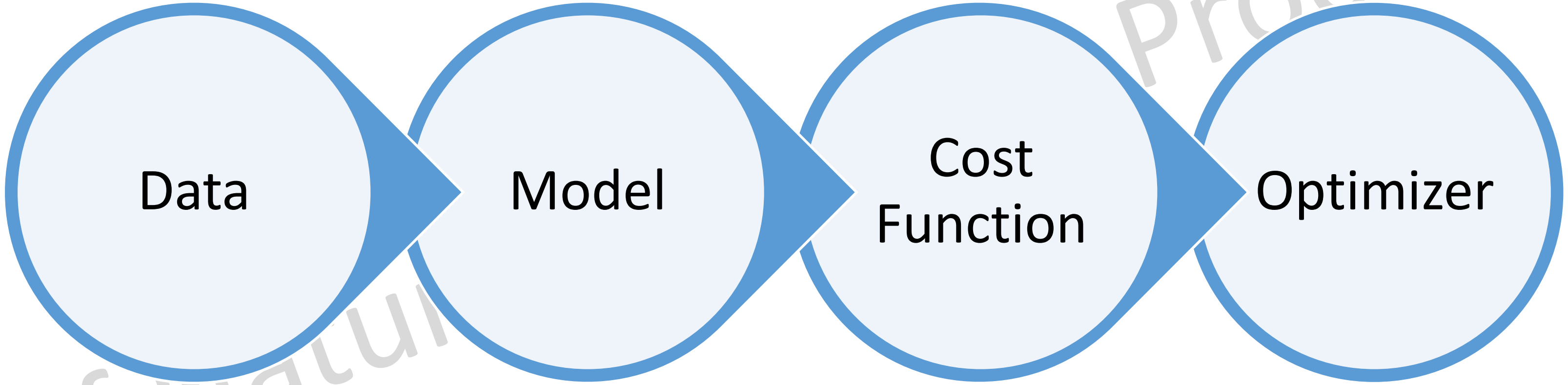
Vectorization

- Convert to numbers

Modeling

QUICK TIPS

Components of every ML Project



Outline

Session 1: Introduction

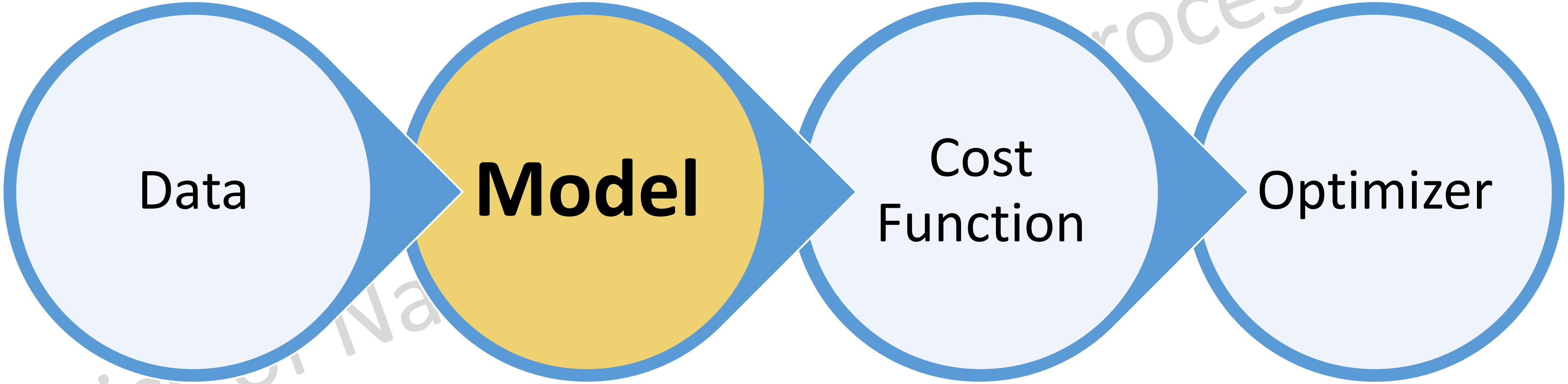
Session 2. Basics of Linguistics

Session 3. Basics of ML

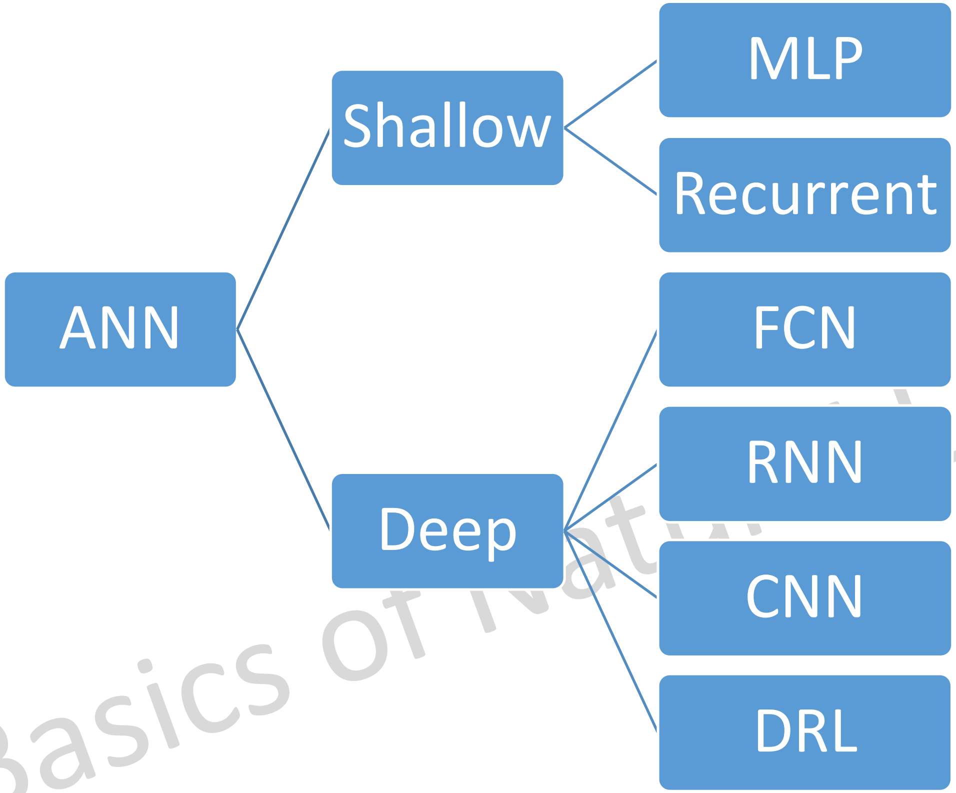
- word2vec
- Components
- **Architectures**
- Python

Session 4 (Lab). Effective Word Representation by python

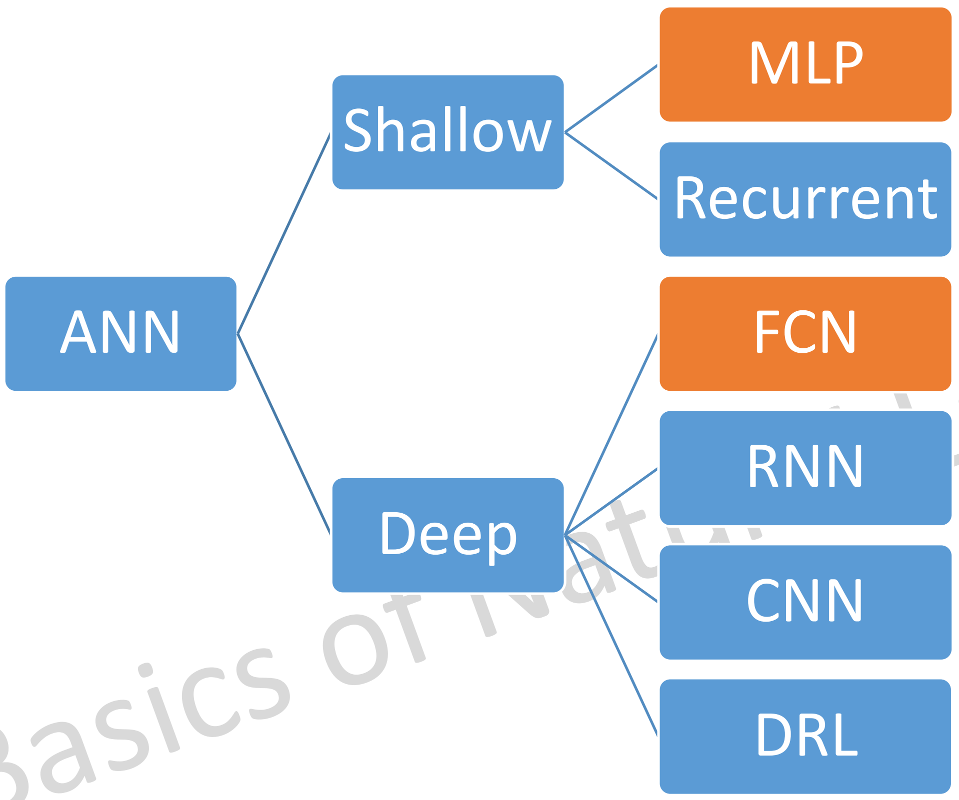
Neural Network Architectures

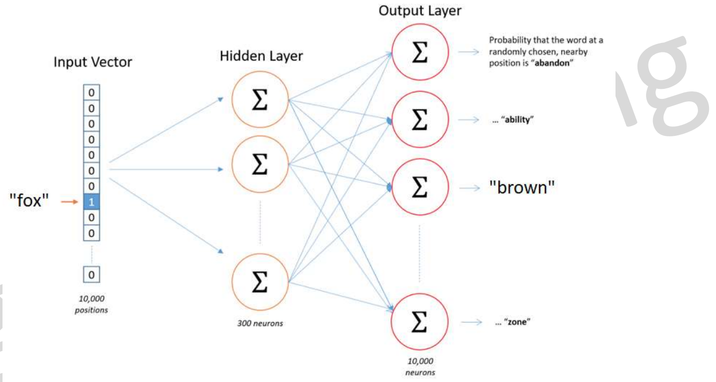


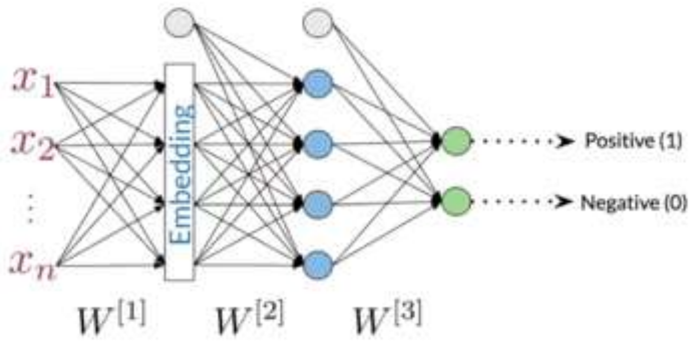
ANN Architectures



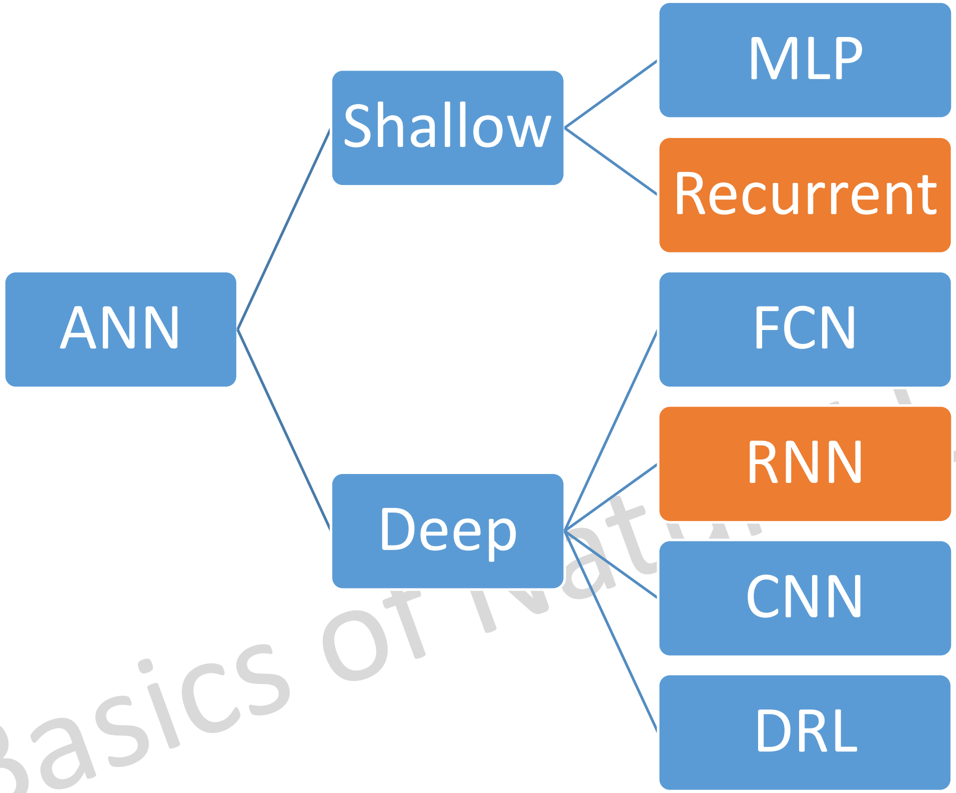
ANN Architectures: MLP / FCN

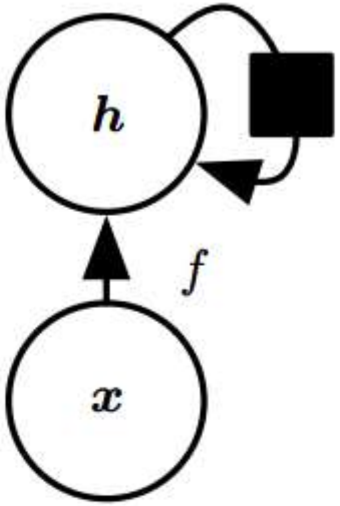






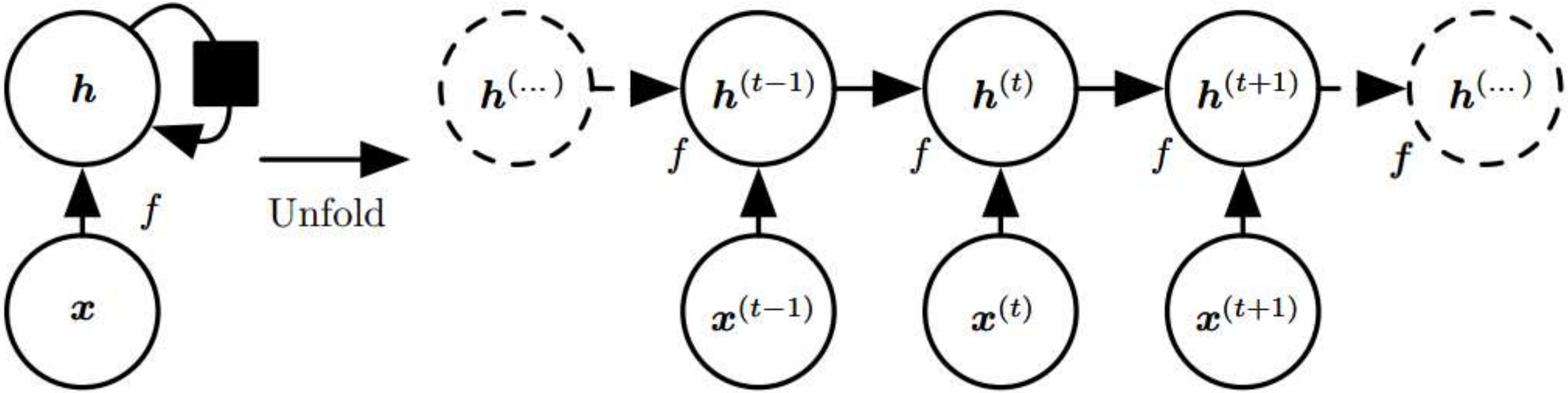
ANN Architectures: RNN





Sequence Modeling

Recurrent NNs are for processing <u>sequential</u> or <u>time-series</u> data, including text.



RNN usage in NLP

Word-level classification

- NER

Sentence-level classification

- Sentiment polarity

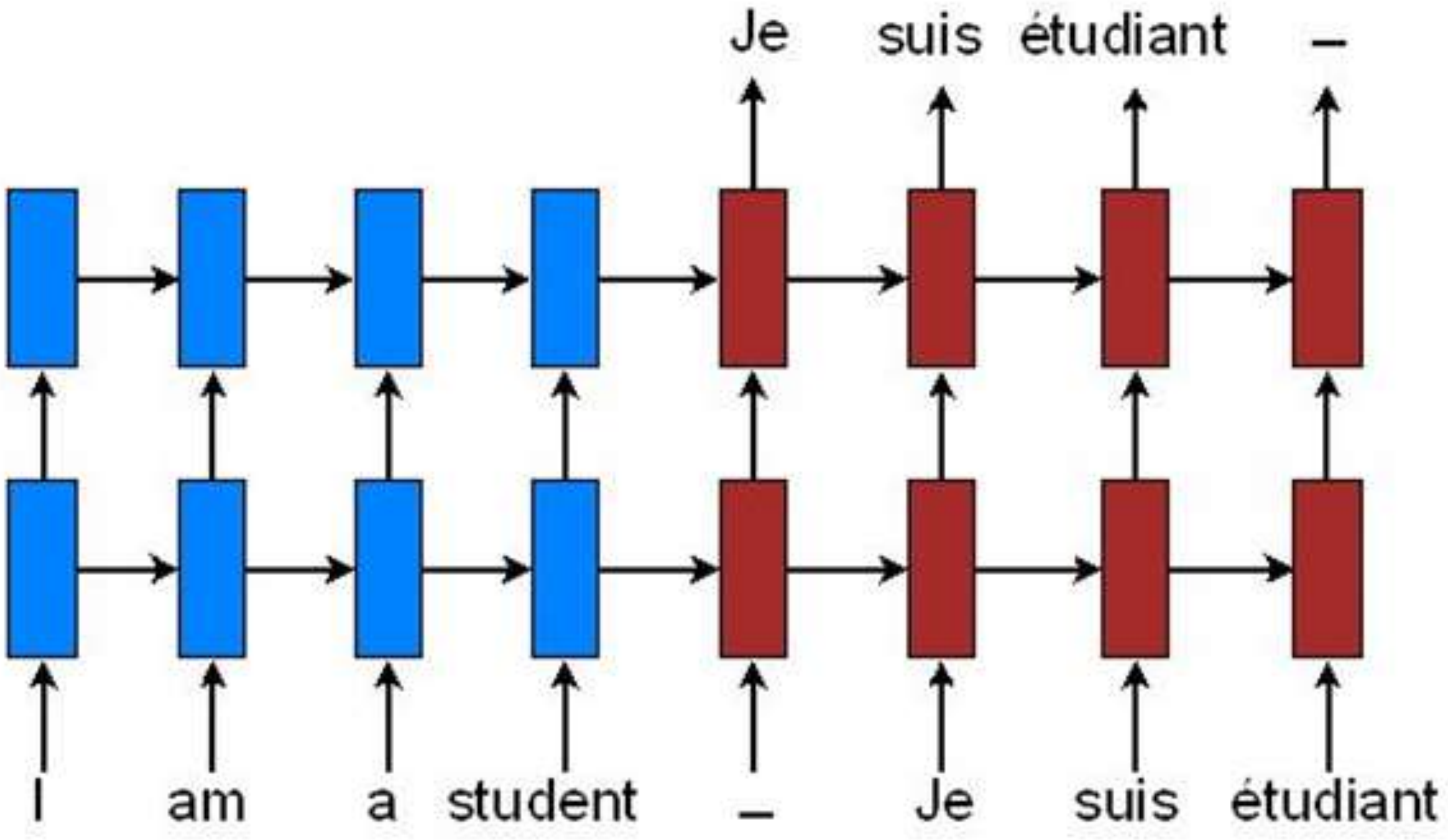
Semantic matching

- Match a message to candidate response in dialogue systems

NLG

- Machine translation
- QA
- Image captioning

Neural Machine Translation (NMT)



[Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473.*]

[https://slideplayer.com/slide/7710523/]

NLP Evolution on RNN

- RNN (LSTM, 1997; GRU, 2014)
- Transformer, 2017
- BERT, 2018; GPT, 2018
- GPT-2, 2019
- GPT-3, 2020

Next Digikala Academy Event will Review RNNs

Basic

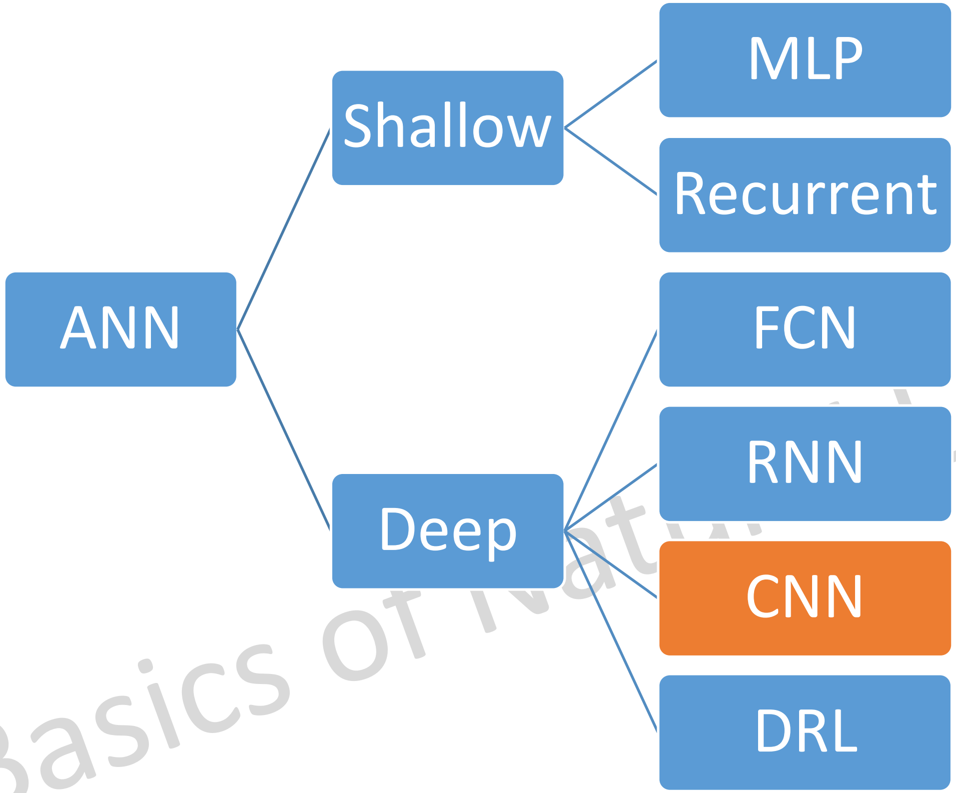
Intermediate

- **Session 1.** DL Models for NLP
- **Session 2.** RNN (GRU/LSTM/Transformer/attention)
- **Session 3.** BERT (Google, 2018)
- **Session 4 (Lab).** Language Model by Python

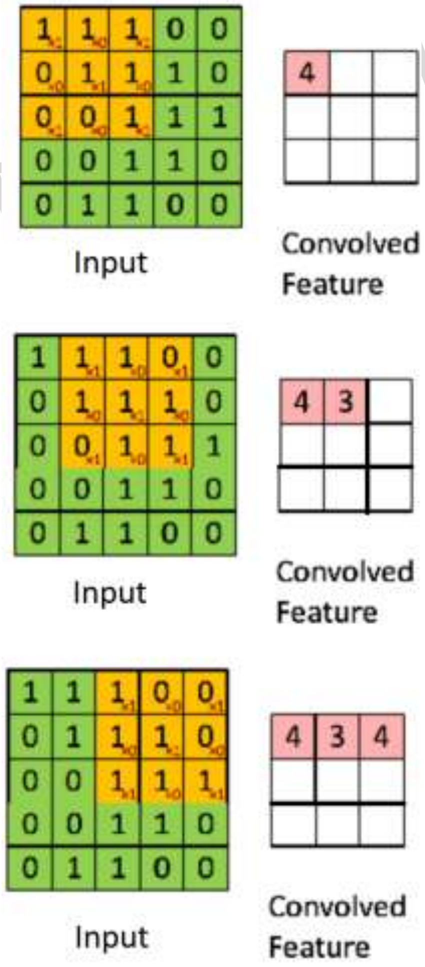
Advanced

- TBA

ANN Architectures: CNN



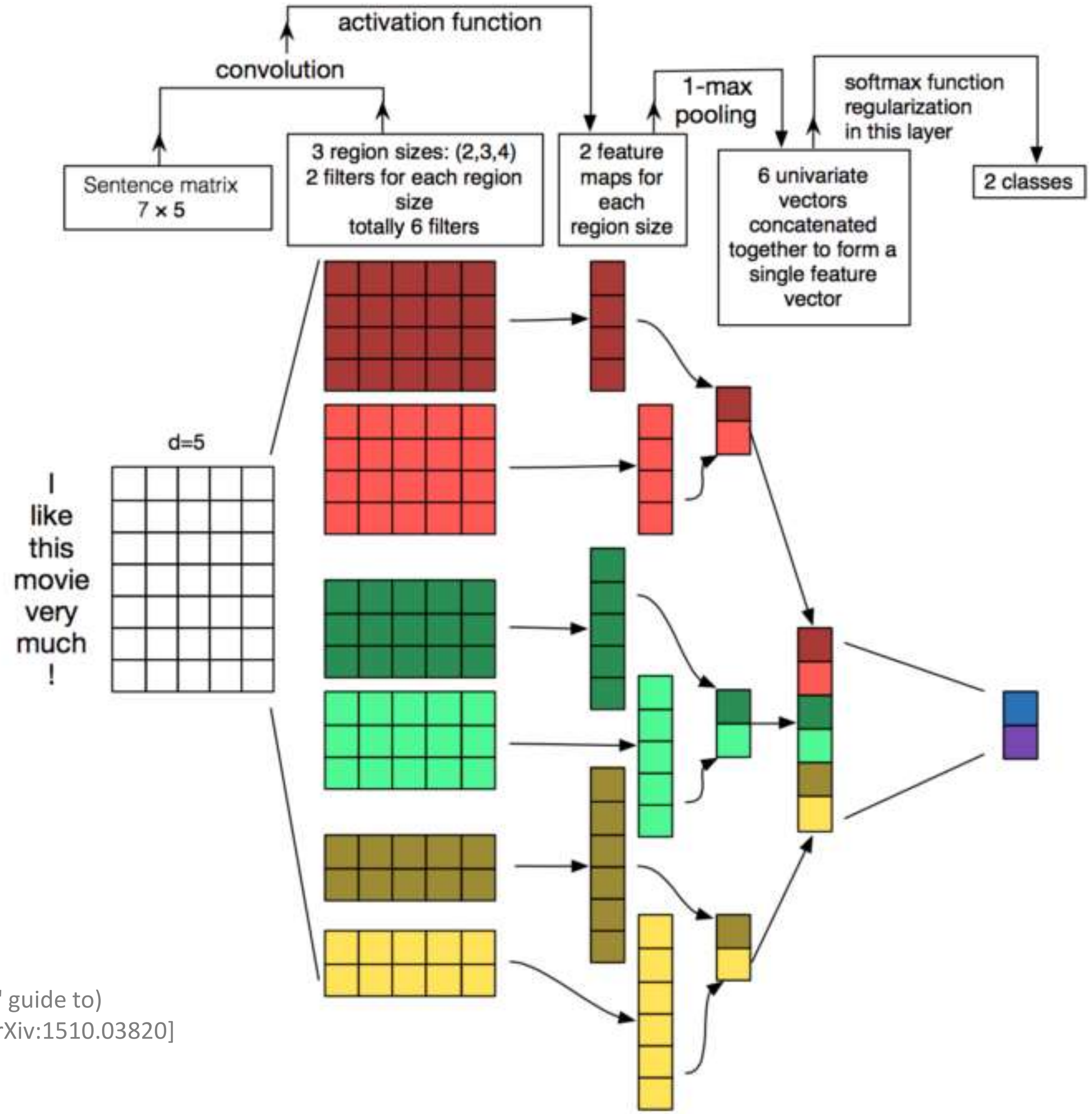
CNN uses <u>convolution</u> in place of general matrix multiplication in <u>at least one layer</u>.



CNN Example

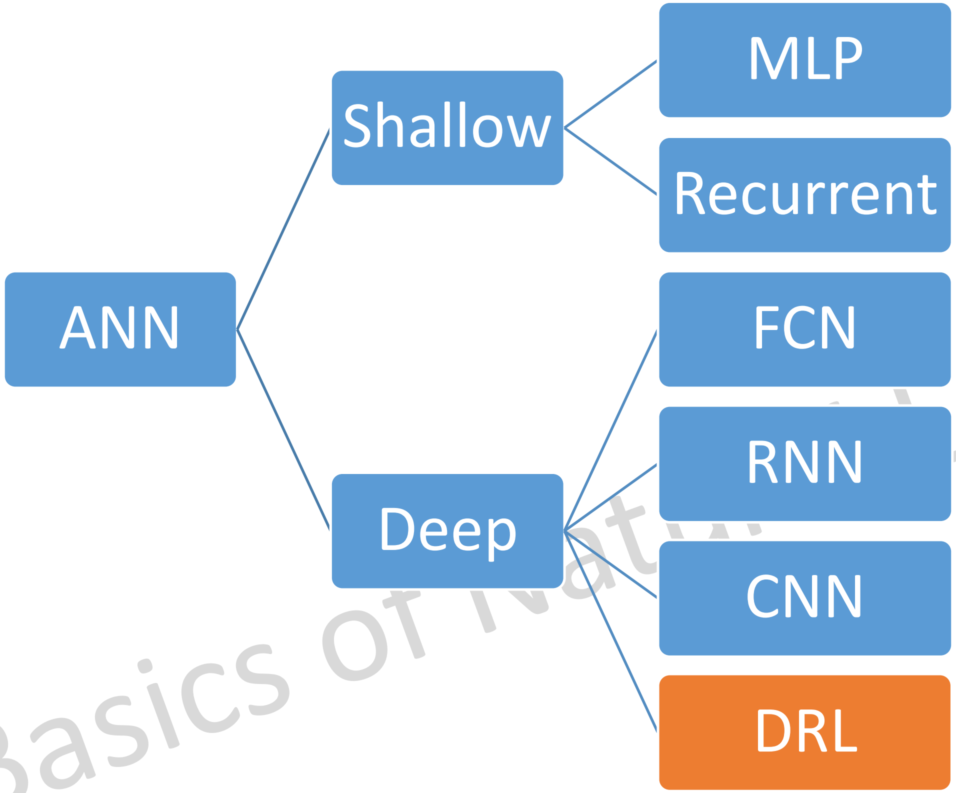
Sentiment Analysis

Q: Can you find N-gram analysis in the architecture?

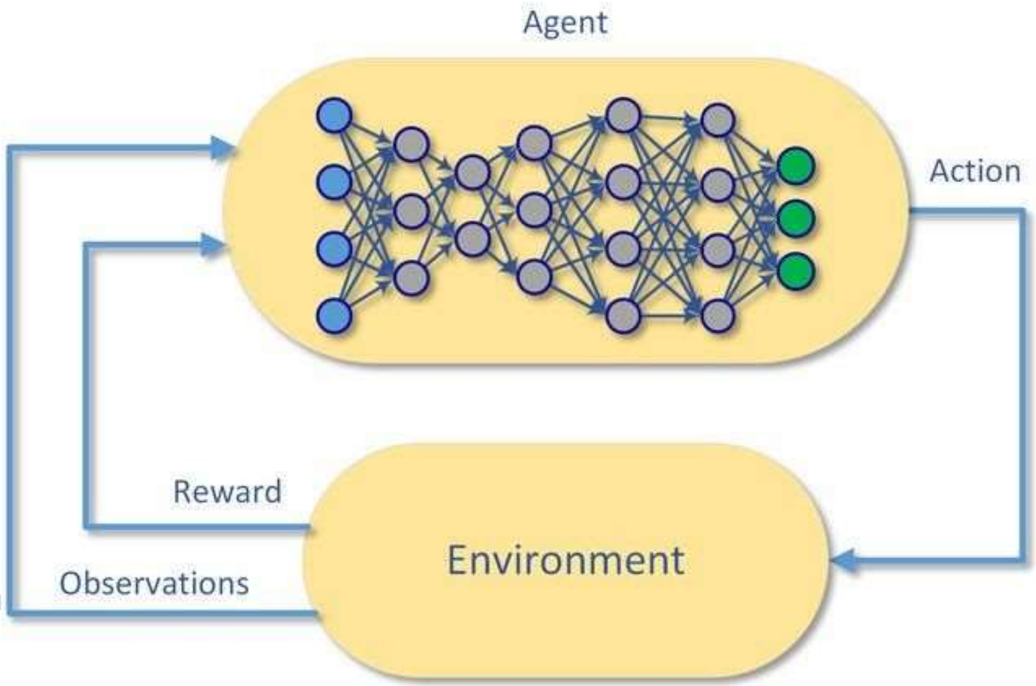


[Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820]

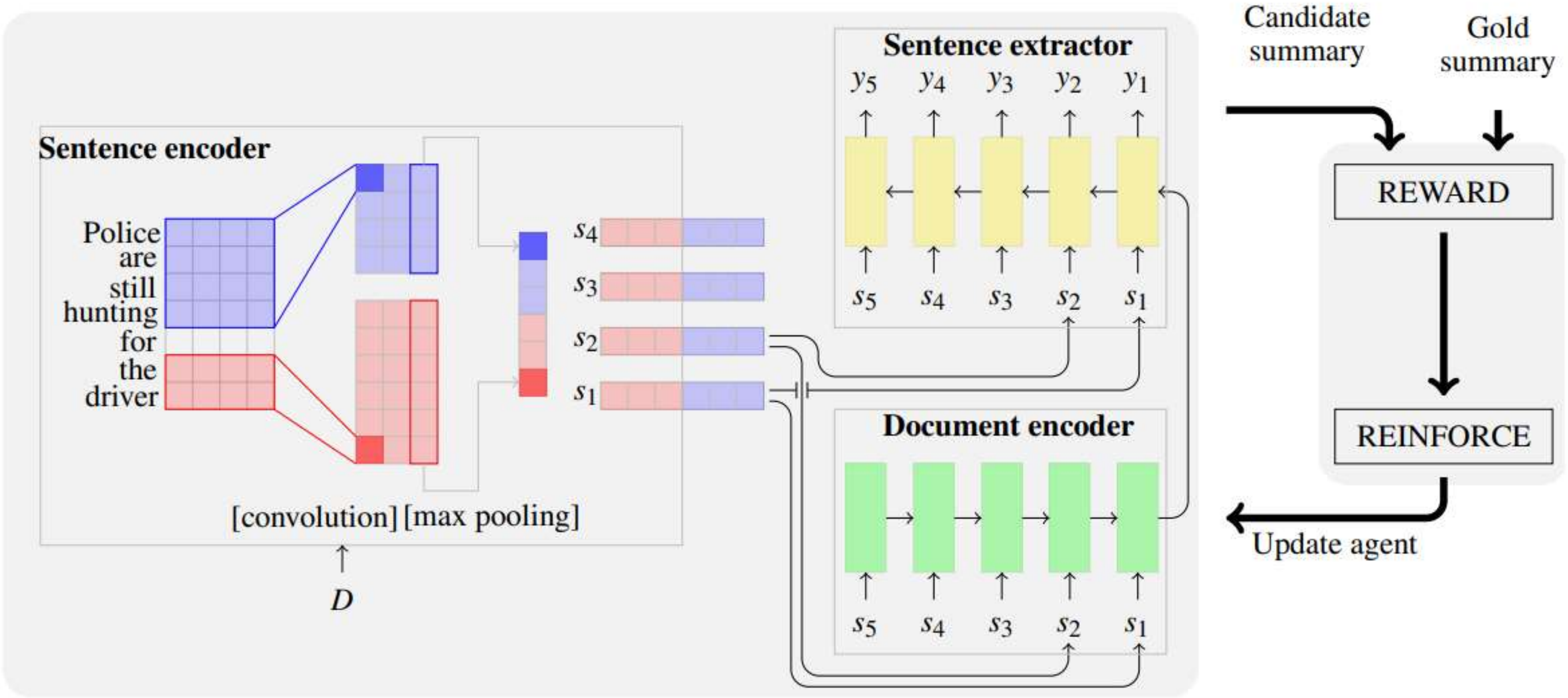
ANN Architectures



Deep Reinforcement Learning

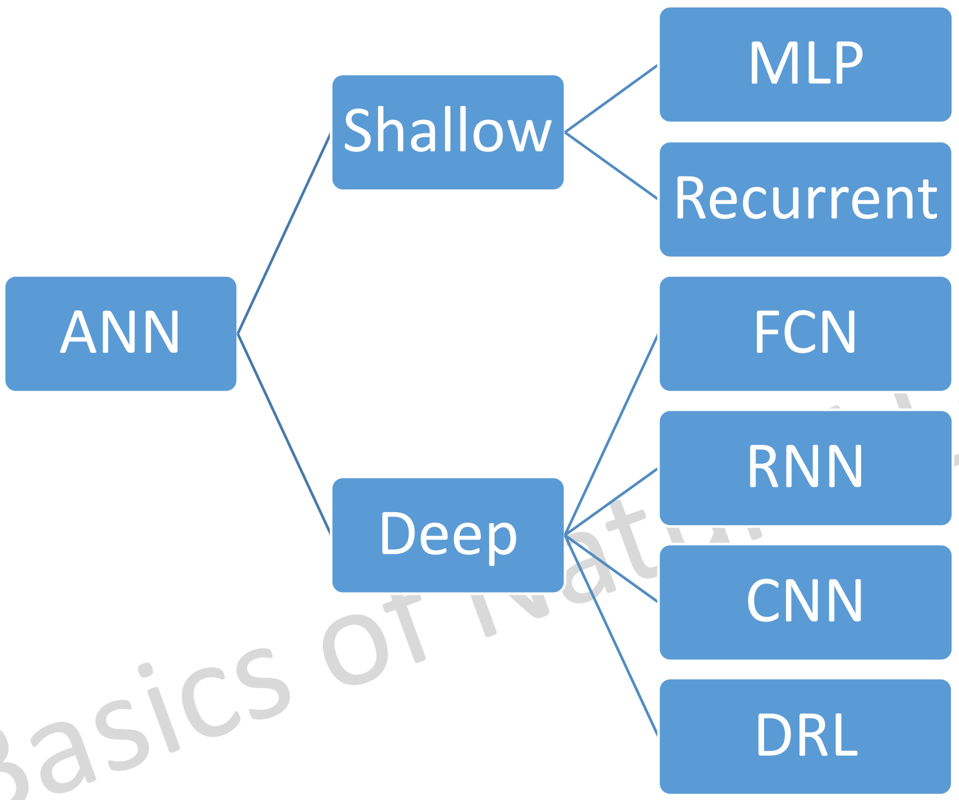


DRL for Summarization



[Narayan, S., Cohen, S. B., & Lapata, M. (2018). Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.]

ANN Architectures



+ Unsupervised ANNs
+ Generative Models
    + VAE
    + GAN

References of this session

Natural Language Processing (NLP) with Python — Tutorial at https://medium.com

Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

https://github.com/NirantK/awesome-project-ideas

System Requirements

python (>= 3.6)

Packages

- jupyter
- gensim
- numpy
- nltk

To install the packages, you may check The following page:
https://github.com/AzamRabiee/Lab-material-for-Intro-to-AI-ML