Natural Language Processing

Word Embeddings and Language Modeling

GAI Motivation (for NLP tasks)

Supervised learning
- Text classification
- QA system
- ...

Think about your learning strategies !

Issues
- Lack of training data
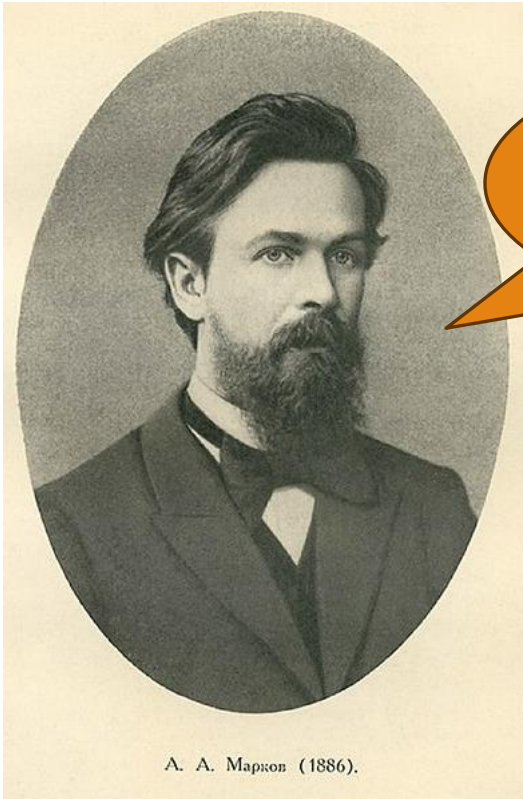- Limitation of domain knowledge

Natural Language Generation

The commonest way to generate sentences is by **writing the words down, one after another**.

e.g.   Please turn your homework …

⬆

The next could be ?

in ?

over ?

語言模型 (Language Model)

[1913] The chance of a letter appearing depends on the letter before it.

[1951] Prediction and Entropy of Printed English

A. A. Марков (1886).

Andrey Markov
1856 - 1922

Claude Shannon
1916 – 2001

Language Model

妳說這一句　很有夏天的感覺

消失的下雨天

我用幾行字形容妳是我
的誰

為妳翹課的那一天　花落的
那一天

怎麼這樣子　雨還沒停妳就撐
傘要走

童年的紙飛機 現在終於飛回我
手裡

天青色等煙雨 而我在
等妳

P (b | a)
P (c | ab)
.
.
.

P (說 | 妳) = 1/4
P (說 | 沒停妳) = 0

消失的下　天

翹課的那一天
花落

怎麼這樣子　雨還沒停妳　　就撐
傘要走

用幾行字形容

是　　為　的誰

說這一句　很有夏天的
感覺

童年的紙飛機 現在終於飛回我
手裡

天青色等煙雨　　而
在等
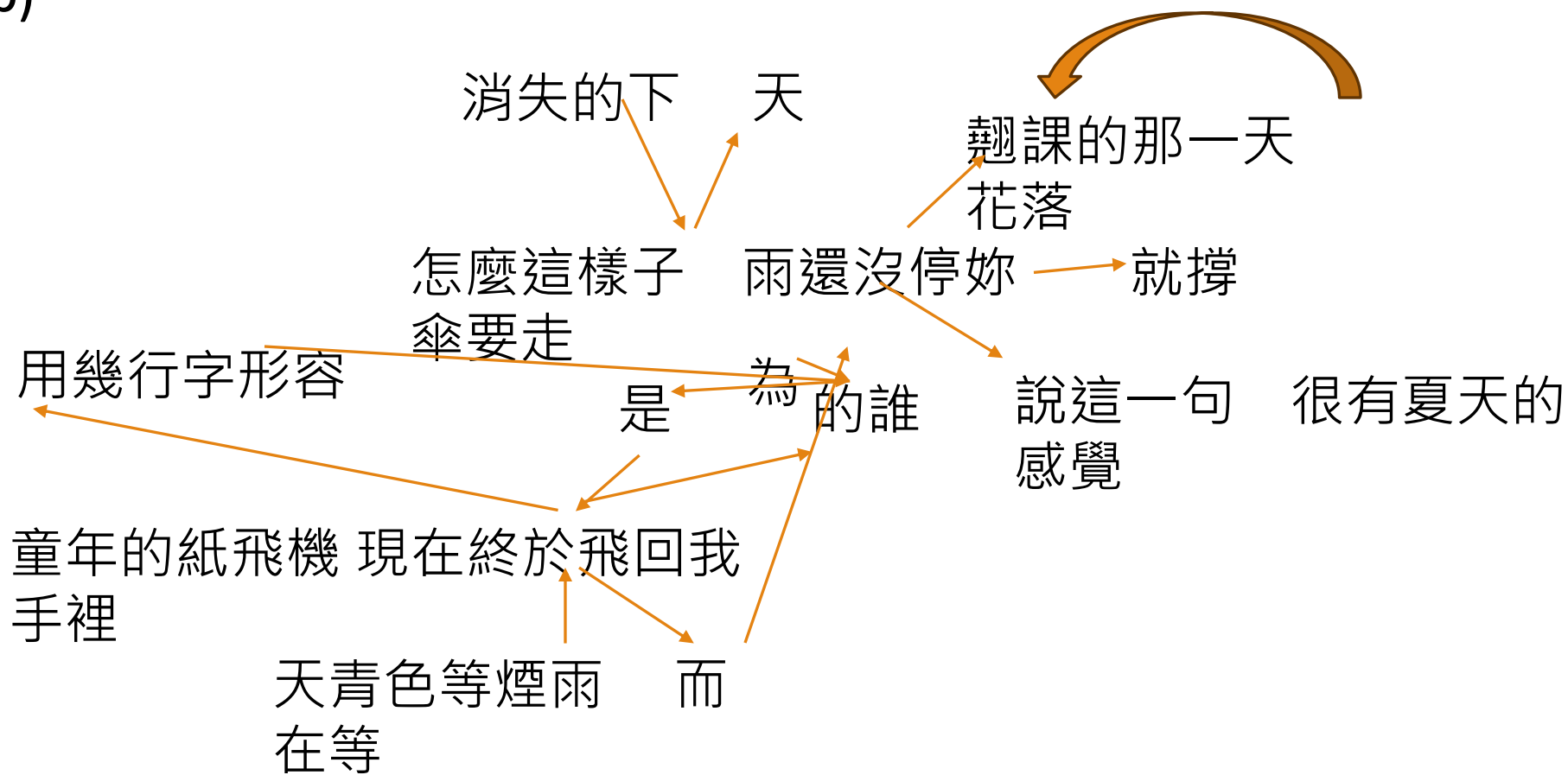
N-grams

An n-gram is a sequence of n words:

e.g.   Please turn your homework …

1-gram (unigram): "please", "turn", "your", or "homework"

2-gram (bigram): "please turn", "turn your", or "your homework"

3-gram (trigram): "please turn your", or "turn your homework"

…

linguistic signature

☐ JK Rowling vs. Robert Galbraith
  ☐ Harry Potter vs. The Cuckoo's Calling

Text Analysis Software: Patrick Juola's JGAAP

Stylometric Analysis:

Sequences of four-character strings (four-grams), which are strong indicators of authorship.

The frequency of the most common words.

The distribution of word lengths.

Pairs of words that frequently appeared together.

N-gram Language Models

We can use a naïve statistic method to model the language

I want to eat lunch.
I want to eat Chinese food.
I don't want to spend time cooking
…

In a bi-gram model we have to count the occurrences of each bi-gram.

e.g.
C(I want) = 2
C(want to) = 3
C(spend time) = 1
…

N-gram Language Models

An example of bi-gram LM.

Draw a table of bigram counts for eight of the words in all of the sentences

	I	want	to	eat	Chinese	food	launch	speed
I	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
Chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
launch	2	0	0	0	0	1	0	0
speed	1	0	1	0	0	0	0	0

N-gram Language Models

Apply add-k smoothing (k=1):

	I	want	to	eat	Chinese	food	launch	speed
I	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
Chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
launch	3	1	1	1	1	2	1	1
speed	2	1	2	1	1	1	1	1

Compute Probability (relative frequency):

	I	want	to	eat	Chinese	food	launch	speed
I	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.0075
want	0.0013	0.00042	0.26	0.0084	0.0029	0.0029	0.0025	0.0015
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
Chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
launch	0.0017	0.0056	0.00056	0.0056	0.00056	0.0011	0.0056	0.00056
speed	0.0012	0.0058	0.0012	0.0058	0.0058	0.0058	0.0058	0.0058

Probability of a Sequence

Let's begin with the task of computing the probability

$$P(w|h) = \frac{C(w,h)}{C(h)}$$

w: the word to be generated

h: some history

C: the times the pattern show up in the dataset

e.g., Compute the probability of the word "the" given the history "its water is so transparent that".

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$$

Probability of a Sequence

Compute probabilities of entire sequences like $w_1 \ldots w_n$ N-gram model(Chain Rule of Probabilities)

$$P(w_1 \ldots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \ldots P(w_n|w_{n-N+1})$$

$$= \prod_{k=1}^{n} P(w_k|w_{k-N+1})$$

e.g., Bi-gram model

$$P(w_1 \ldots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2) \ldots P(w_n|w_{n-1})$$

$$= \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Language Models

Language Models (LMs) :

Models that assign probabilities to sequences of words are called LMs.

Including:

N-Gram Language Models:

A purely statistical model of language.

Neural Language Models:

Use neural networks to predict the likelihood of sequences.

LM evaluation: Perplexity (困惑度)

Perplexity (PPL) is a quantitative criterion used to evaluate the capacities/effectiveness of language modeling models.

- Given the sequence of words $W = w_1 w_2 \ldots w_N$ and an N-gram model. The PPL of the model was computed by:

$$Perplexity(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\prod_{k=1}^{n} \frac{1}{P(w_k | w_{k-N+1:k})}}$$

$1 \sim \infty, 1 \sim |V|$

The **lower** the value of perplexity, the **better** the language modeling capability of the model.

Perplexity meaning

A Measure of Uncertainty: Perplexity quantifies the level of uncertainty or unpredictability that a model experiences when making predictions.

An Average Branching Factor: Perplexity can be viewed as an "average branching factor" of possible choices at each step in the sequence.

Quantification of Model Performance: In language modeling, perplexity reflects how well the model understands the rules and structure of the language.

◦ A lower perplexity indicates better language comprehension and prediction ability, signifying the model's efficiency in capturing language patterns.

Compression Efficiency Indicator: Perplexity can also be seen as a measure of how well the model compresses the test data.

◦ Lower perplexity means that the model predicts with higher probabilities, reducing the uncertainty and effectively compressing the information.

N-gram Language Models

Bigram model

Approximates the probability of a word given all the previous words $P(w_n|w_{1:n-1})$ by using only the conditional probability of the preceding word $P(w_n|w_{n-1})$

i.e., Instead of computing the probability $P(the|its\ water\ is\ so\ transparent\ that)$
Bigram model approximates it with the probability $P(the|that)$

The assumption that the probability of a word depends only on the previous word is called a Markov assumption.

Shortcomings of N-gram LMs

Limited context
- N-gram models are unable to capture longer-distance (>>N) language dependencies.

Data sparsity (High time/space complexity)
- As the N value increases, the number of parameters to store and compute grows exponentially.

Ignoring word order / context information
- N-gram models assume independence between words, neglecting the influence of word order on semantics.

Low flexibility
N-gram language models struggle with *synonyms* and have limited ability to adapt to varying conditions. (e.g. dialogue).

Rare/Unseen words?

New contexts or domains?

Sparse Vectors

Sparse vector embeddings represent words as high-dimensional vectors with mostly zero values.

Each dimension corresponds to a unique feature (word), measured by some well-designed methods.

- TF-IDF
- PPMI

TF-IDF (recap)

The mathematical representation of TF-IDF:

$$TF - IDF = TF \times IDF$$

where

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \qquad IDF_i = lg\frac{|D|}{|\{j : t_i \in d_j\}|}$$

◦ Where $n_{i,j}$ is the i-th word in j-th text in the dataset.

TF (Term Frequency)
◦ Represents the "frequency" of a term appearing in a text.

IDF (Inverse Document Frequency)
◦ Aims for terms to have higher specificity, meaning the fewer texts in the dataset contain the term, the better.

TF-IDF (recap)

After computing the scores of every terms, we get a sparse vector to represent the text.

An apple a day, keeps the doctors away.

$$[\ \cdots\ f_{an}\ \cdots\ f_{apple}\ \cdots\ f_{a}\ \cdots\ f_{day}\ \cdots\ f_{keeps}\ \cdots\ f_{the}\ \cdots\ f_{doctors}\ \cdots\ f_{away}\ \cdots\ ]$$
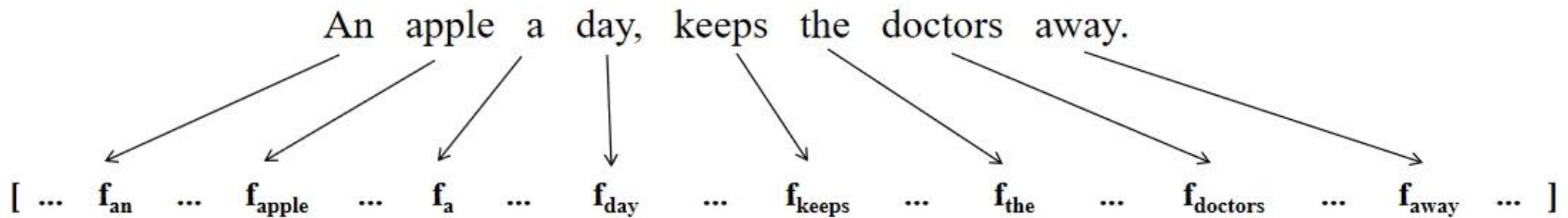
TF-IDF (recap)

Preprocessing text (optional)

Stemming
- By removing the suffixes from words (e.g."cats," "catlike," "catty" all have "cat" as their base), we can revert the words back to their root forms.

Feature Selection
- Filter and select which parts of speech to retain, such as verbs or nouns.
- Analyze the frequency of the terms using statistical methods or algorithms like TF-IDF.

Distributional Hypothesis

Words that occur in similar contexts tend to have similar meanings.

The NLP approaches utilize the context around the word to define its meaning.

e.g.
- ◦ I enjoy coding and I do it everyday!
- ◦ I like coding and I do it everyday!

"enjoy" and "like" are synonym and they are in the same context.

PPMI

Mutual Information (MI)
- ◦ It is a measure of how often two events x and y occur:

$$I(x, y) = log_2 \frac{P(x, y)}{P(x)P(y)}$$

PMI (Pointwise MI)
- ◦ The mutual information between a target word w and a context word c:

$$PMI(w, c) = I(w, c)$$

PPMI

PPMI (Positive PMI)
- ◦ PPMI replaces all negative PMI values with zero

$$PPMI(w, c) = max(PMI(w, c), 0)$$

e.g. Co-occurrence counts for 4 words in 5 contexts

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3972	378	5	13	7003
count(context)	4997	5673	473	512	61	11716

PPMI

Replacing the counts in with joint probabilities:

	P(w, context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

Compute the PPMI matrix

Sparse vectors are obtained
cherry=(0,0,0,4.38,3.30)

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

$=\log_2(0.0021/(0.0415*0.0052))$

Word embedding

Concept space

Term vector

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
便宜	1	3	2	3	0	-2	2	0
有名	3	1	4	2	0	2	0	1
讚	0	0	1	0	0	1	-1	0
嫩	1	3	0	1	0	0	0	0
難吃	0	0	0	0	1	0	2	0
太貴	0	0	0	0	3	1	0	1
差	-1	0	1	-1	0	2	1	0
老	0	-2	-1	0	1	3	2	1

Dense Vectors

Dense vector embeddings represent words in a continuous vector space

Semantically similar words are closer together.

Word2Vec

Contextualized Embeddings

Properties of Embeddings

Vectors for representing words are called **embeddings**

- Analogy/Relational Similarity
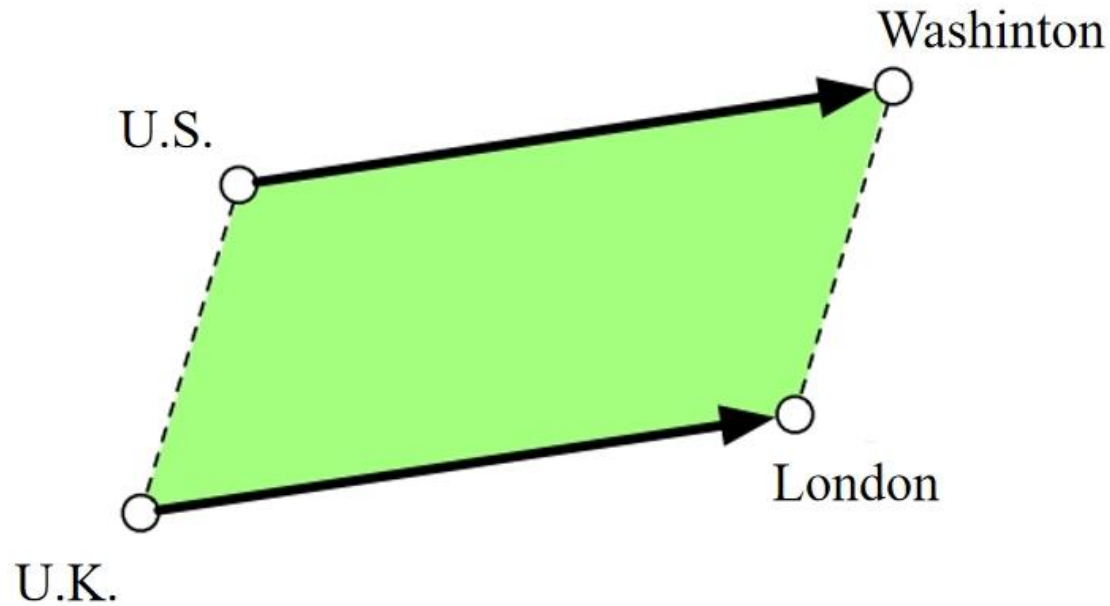
Washinton - U.S. = London - U.K.
Washinton - U.S. + U.K. = London

Properties of Embeddings

- Historical Semantics
  - ➢ Embeddings can also be a useful tool for studying how meaning changes over time
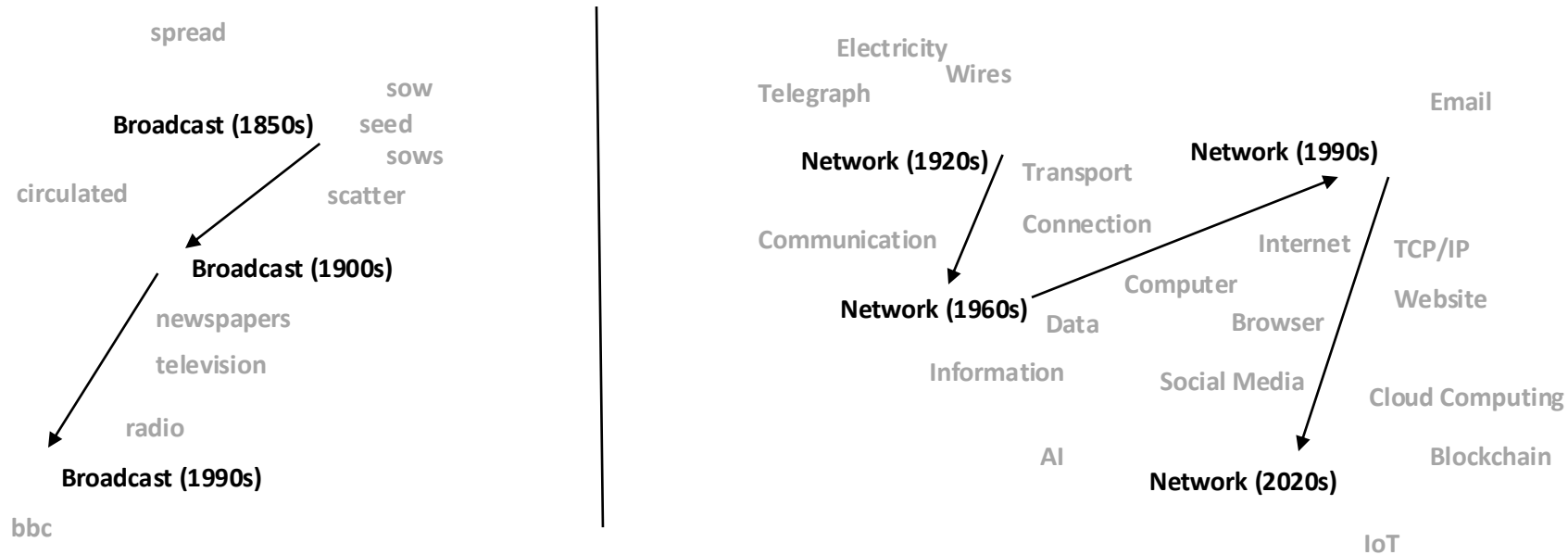
Word2Vec

➢ Skip-gram
  ▪ Use words to predict their contexts.
➢ CBOW
  ▪ Use the context to predict the target word.

CBOW

Skip-gram
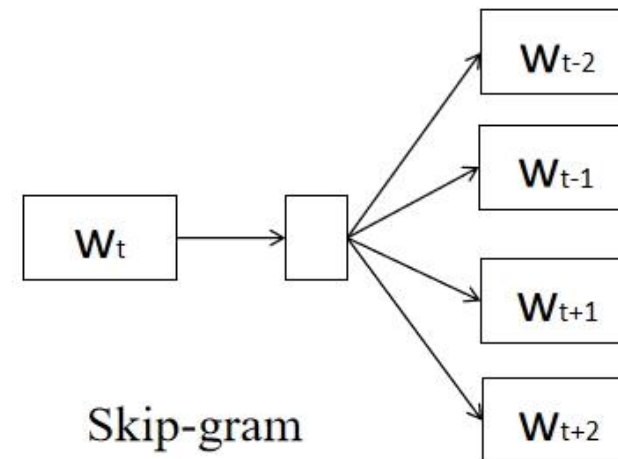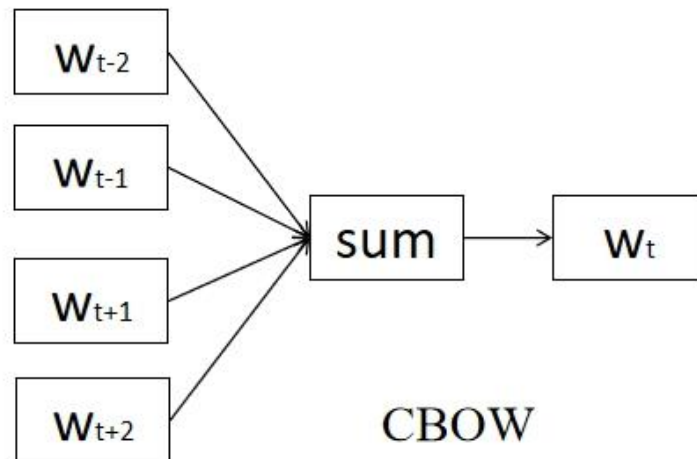
Word2Vec

1. Treat the target word and a neighboring context word as **positive examples**.

2. Randomly sample other words in the lexicon to get **negative samples**.

3. Use **logistic regression** to train a classifier to distinguish those two cases.

4. Use the learned weights as the embeddings.

Word2Vec

Calculate the co-occurrence matrix from the corpus, where each element represents how often a word appears in the context of another word within a certain window size.

Assume window size = 2

Word2Vec

Skip-gram:

Assume c is the word id in the context, w is the center word, u and v are vectors of c and w.

$$P(c|w) = \frac{exp(u_c^\top v_w)}{\sum_t exp(u_{c_t}^\top v_w)}$$

Model Parameters

word2vec uses a single hidden layer feedforward neural network
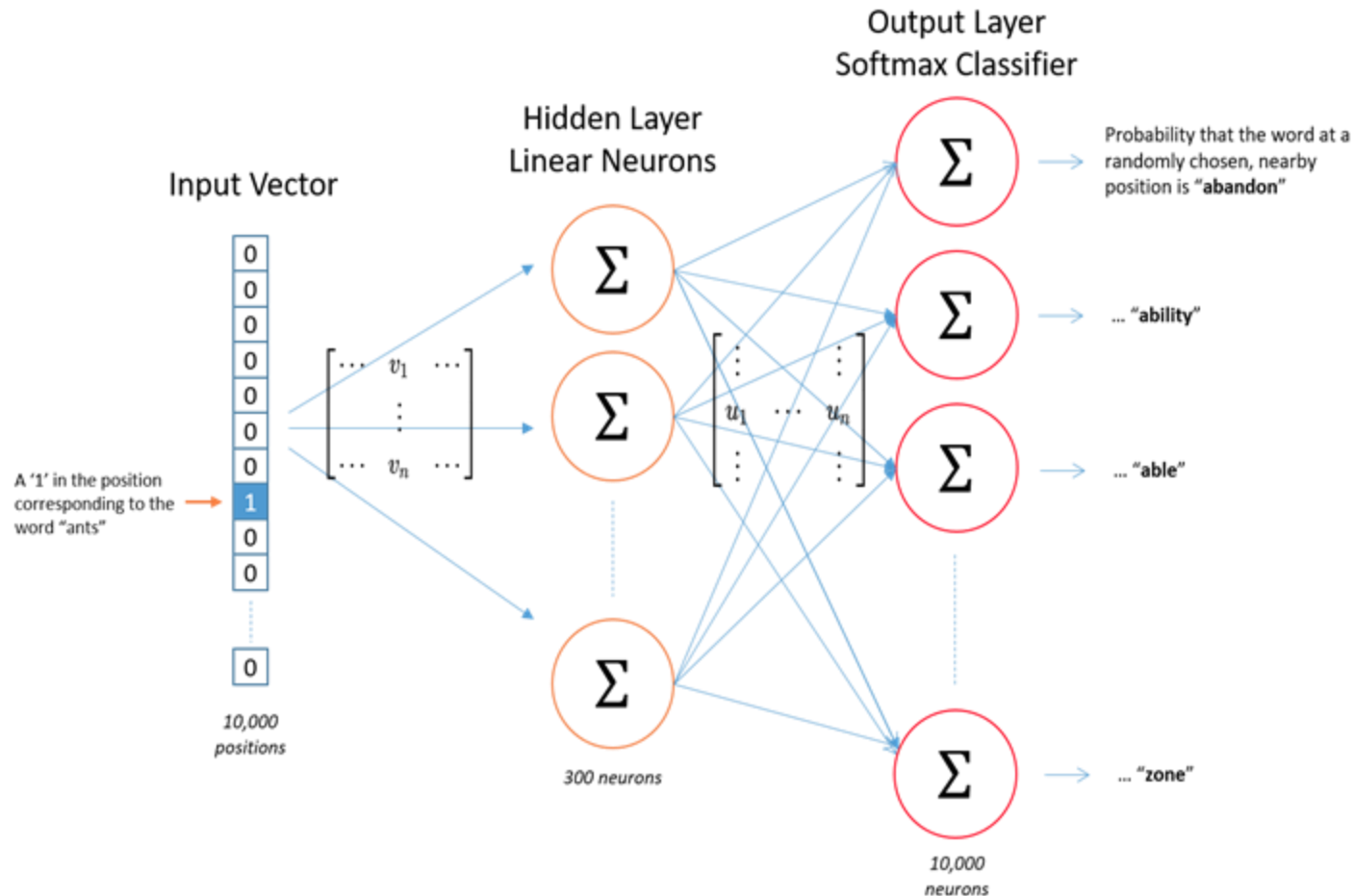
Input to hidden layer matrix has our target word embeddings $\mathbf{v}_i$

Hidden to output layer matrix has another set of word embeddings called output embeddings $\mathbf{u}_i$

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Contextualized Embeddings

➢ Contextualized embeddings
  o capture the meaning of a word in its context within a sentence or a larger body of text.

➢ Contextualized embeddings allows words to have different representations depending on their usage in different contexts.
  o Traditional word embeddings assign a fixed vector to each word.
  o Contextualized embeddings generate a vectors based on the surrounding words.

➢ Contextualized embeddings are widely used in various NLP tasks. (e.g., BERT, GPT)

Word context
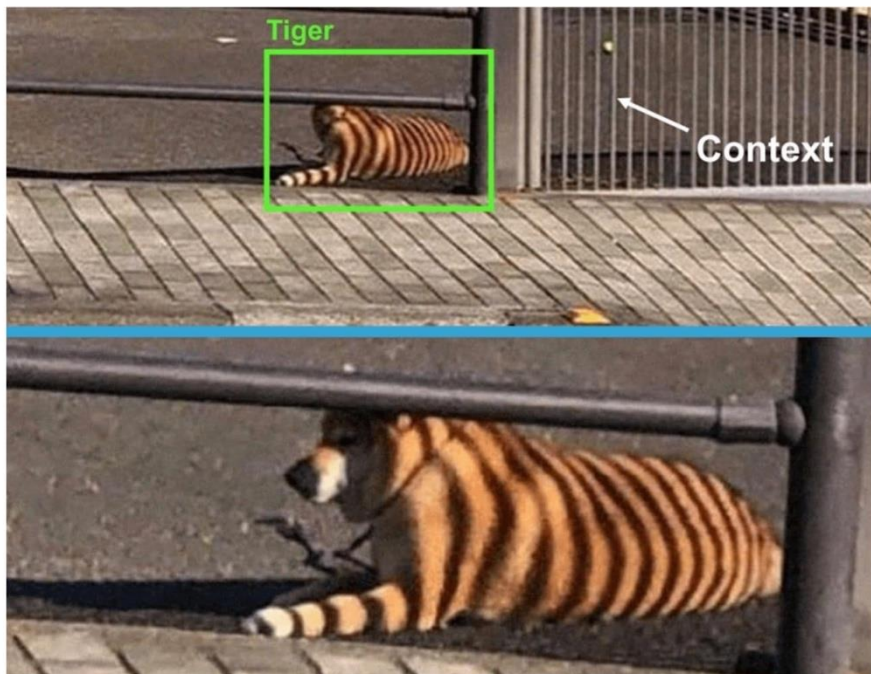
蘋果　公司

蘋果　派

蘋果改變了他的一生，　對**牛頓**來說

蘋果改變了他的一生，　對**賈伯斯**來說

Contextualized Word Embeddings

➤ Learn word vectors using long contexts instead of a context window
➤ Learn a deep Neural LM and use all its layers in prediction

The probability of a sequence becomes:

$$P(w_1 \ldots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \ldots P(w_n|w_{1:n-1})$$

$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

Contextualized Word Embeddings

➤ The embedding is designed to be a part of the model.

➤ When the downstream task provides a preferred context, it can be adjusted during the training process, incorporating the relevant information from the task context.

Inputs → Embedding → LM Layers

forzen? train together?

Neural Network Architecture
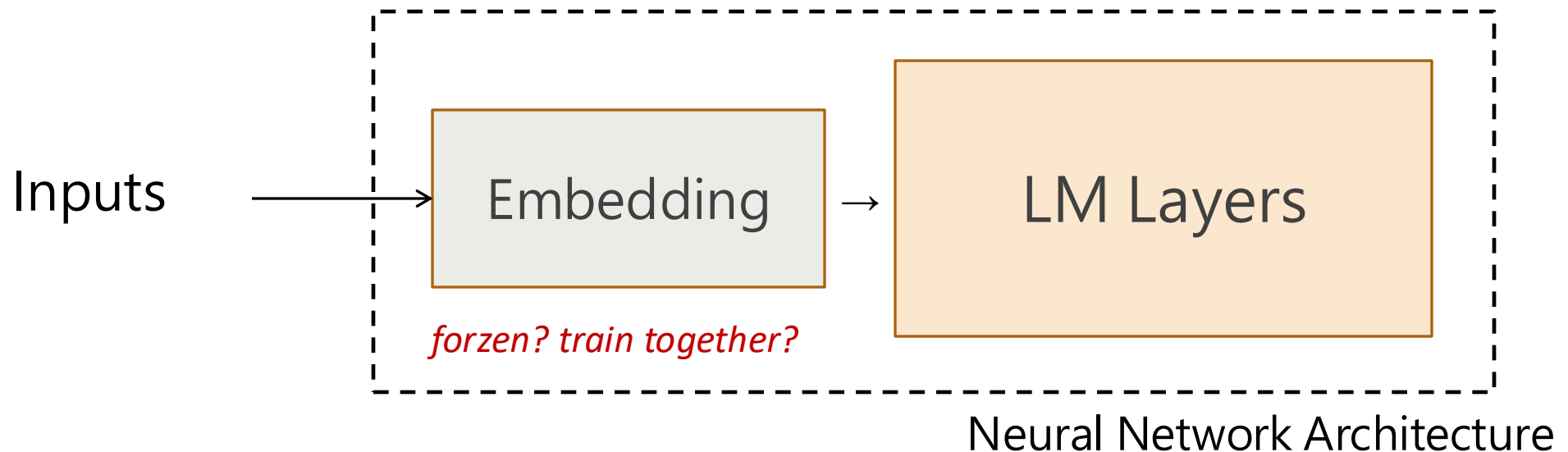
Neural Language Models

A model structure is needed to process the hidden feature in the embeddings.

Neural LMs leverage neural networks to learn and represent complex language patterns.

Basic Definitions
- ◦ FFN (Feedforward network)
- ◦ RNN (Recurrent Neural Network)

Basic Definitions

➢ In a deep learning project, there are some fundamental components.

1. Model

   ▪ A model refers to the architecture or structure used to **represent relationships between input data and output predictions**.

2. Optimizer

   ▪ An optimizer is an algorithm used to adjust the parameters of the model during training in order to **minimize the error** between predicted and actual output values.

3. Loss function

   ▪ A loss function (objective function) **measures the difference** between the predicted output of a model and the true target output.

Training Neural Networks

The training process typically involves the following steps:

1. **Data Preparation:** Prepare training and testing datasets.
2. **Model Construction:** Construct the model using a deep learning framework (TensorFlow, PyTorch, …)
3. **Loss Function Definition:** Select an appropriate loss function. (Cross-entropy, Logloss, …)
4. **Optimizer Selection:** Choose a suitable optimization algorithm. (Adam, SGD, …)
5. **Model Training:** Train the model using the training dataset.
6. **Model Evaluation:** Evaluate the trained model using the testing dataset. (F1, LCS, …)

Activation functions

➤ The core idea of using activation functions is to introduce **nonlinearity** into neural networks.

➤ Neural network models aim to **avoid the final processing stage being merely a linear transformation of the inputs**, **s**o that the model is able to have good performance on complex problems.

➤ Commonly used activation functions include: Sigmoid, ReLU, tanh, GeLU…

Activation functions

	Range	Applications
softmax	[0, 1] (sum up to 1)	It is commonly used in multi-class classification tasks where the model needs to predict the probability distribution over classes.
sigmoid	[0, 1]	It often used in binary classification tasks where a threshold is needed to predict probabilities of belonging to one of the two classes.
tanh	[-1, 1]	It is often preferred over sigmoid for hidden layers as it produces zero-centered output
ReLU	[0, +inf]	It introduce nonlinearity and avoid gradient vanishing

Training Neural Networks

In text viewpoint

FFN
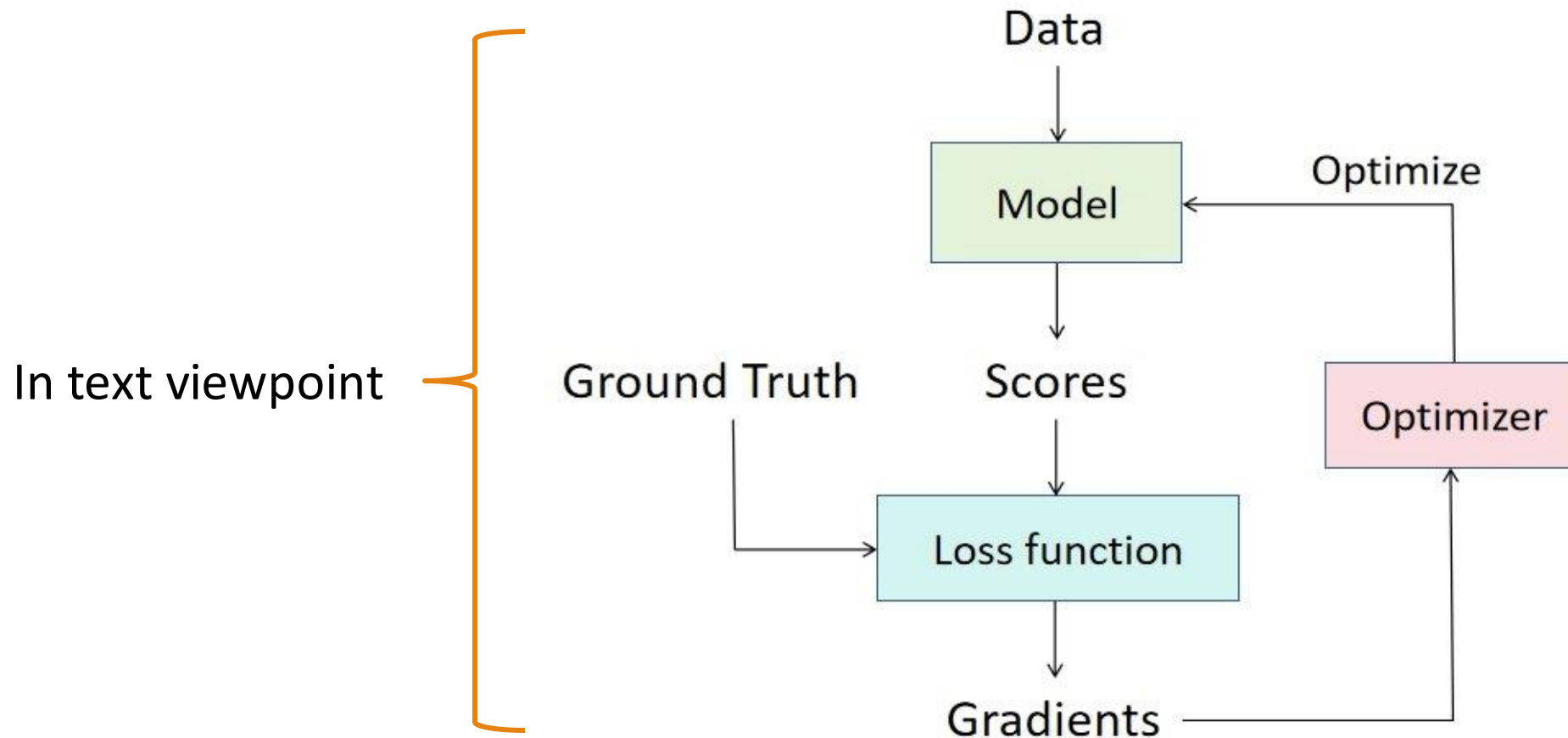
➢ Feedforward network (FFN) is a multilayer feedforward network in which the units are connected with no cycles.

$$h = \sigma(Wx + b)$$

$$z = Uh$$

$$y = softmax(z)$$

Input layer

Hidden layer

Output layer
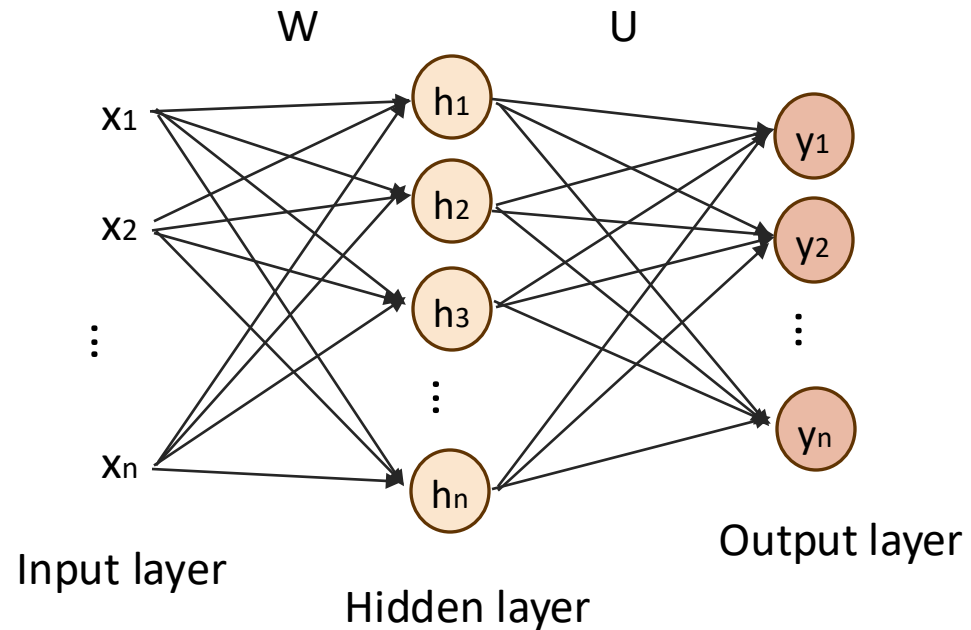
Short comings of FFN

➢ **Lack of Sequence Modeling:**
   o In NLP tasks, understanding the sequence of words and their dependencies is crucial for accurate predictions.

➢ **Fixed Input Size:**
   o FFNs require fixed-size inputs, which can be problematic for NLP tasks where input sequences vary in length.

➢ **Limited Contextual Information:**
   o Many NLP tasks benefit from capturing long-range dependencies and understanding the broader context of a text, which is better addressed by models capable of modeling sequential data effectively.

RNN

➢ Recurrent Neural Networks (RNNs) are a type of artificial neural network designed to handle sequential data by **capturing temporal dependencies**.

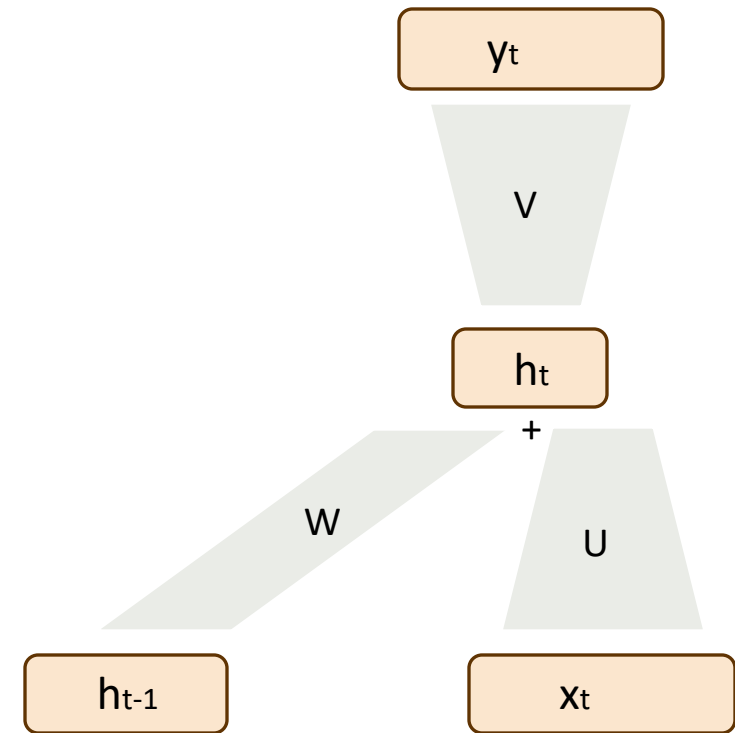 o The same set of weights and biases are used across all time steps
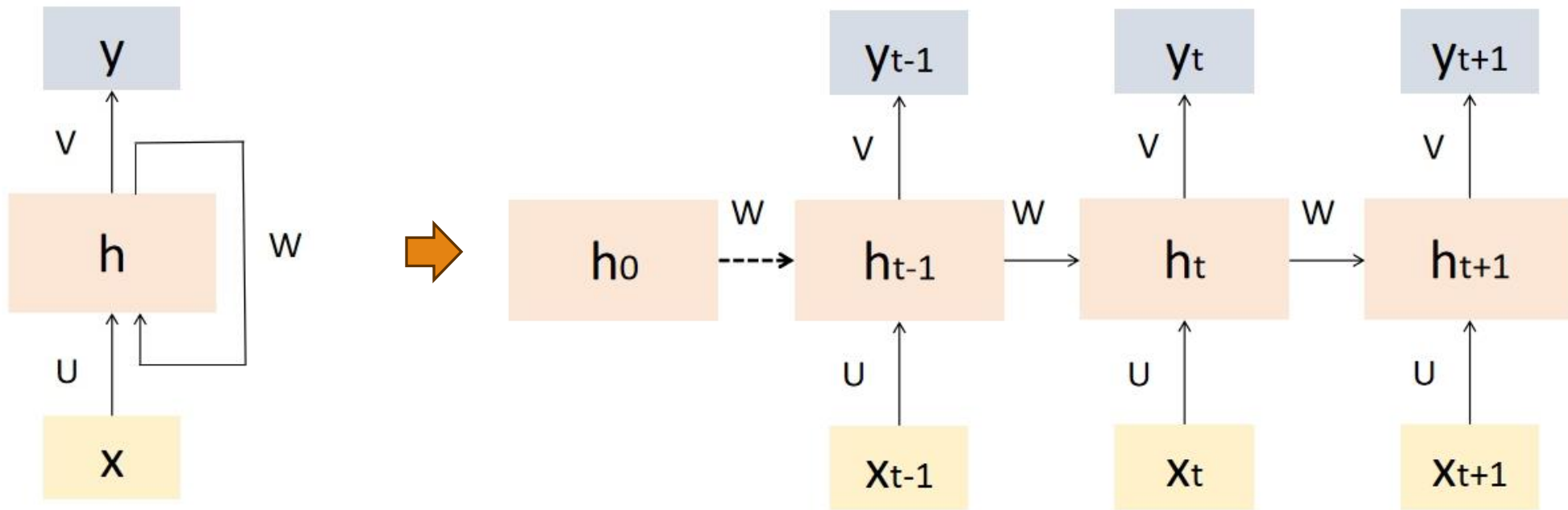
RNN

➢ The equation on step t is:

$$y_t = g(Vh_t)$$

$$h_t = f(Ux_t + Wh_{t-1})$$

where f and g are activation functions

Properties of RNNs

➢ **Sequential Processing:**
- RNNs handle sequences, allowing them to model temporal dependencies in data.

➢ **Recurrent Connections:**
- RNNs maintain internal memory, facilitating the capture of long-term dependencies.

➢ **Parameter Sharing:**
- RNNs share parameters across time steps, enhancing efficiency in learning sequential data.

➢ **Vanishing Gradient Problem:**
- Traditional RNNs may face vanishing gradient issues, hindering learning of long-term dependencies.

Example: Name Entity Recognition

➢ Name Entity Recognition (NER) is a fundamental task in NLP.

➢ The model needs to identify named entities within the sequence, such as countries, organizations, and individuals.
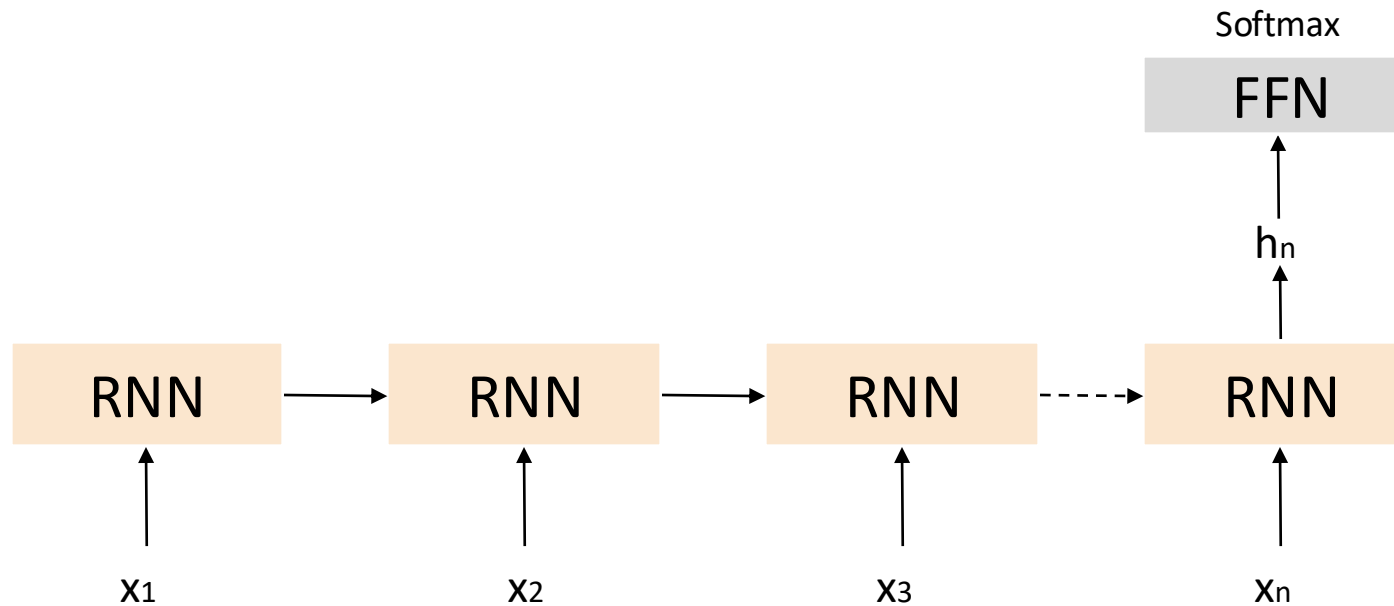
RNNs for NER

➤ In token classification task, every output should be mapped to a one-hot vector.

➤ A feed forward network is added to the RNN model.

$$[\ p_O,\ p_{B\text{-}GPE},\ p_{I\text{-}GPE},\ p_{B\text{-}GPE},\ p_{I\text{-}GPE}\ \dots\ ]$$
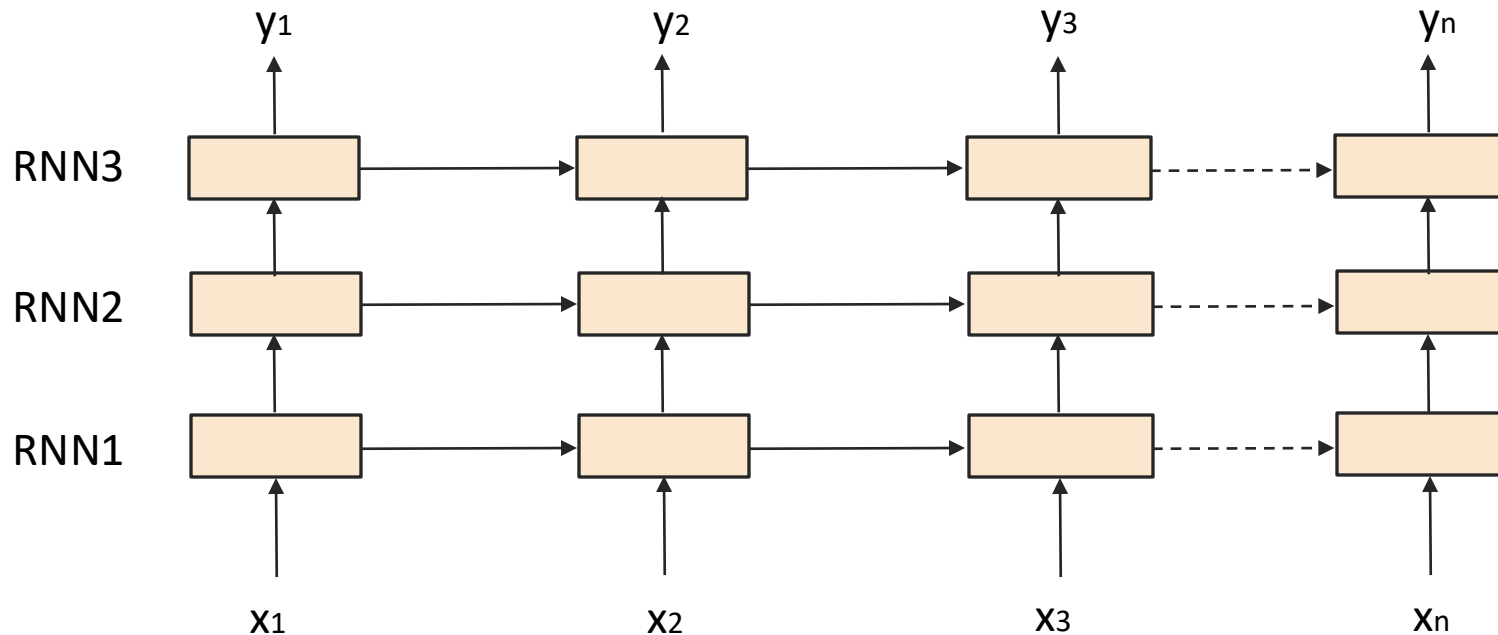
$$Pred_t = argmax(o_t)$$
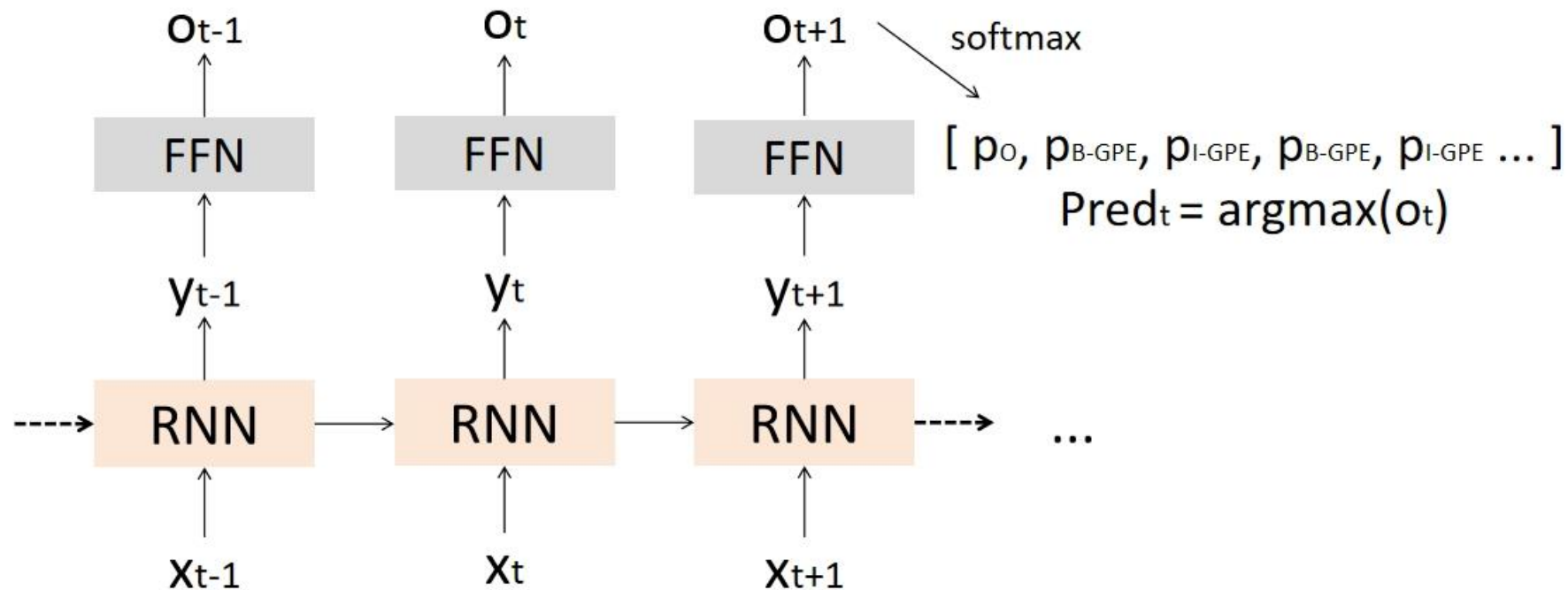
RNNs for Sequence Classification

➤ RNNs classify the entire sequences rather than the tokens within them.

➤ Take the hidden layer for the last token of the text.

Stacked RNNs

➢ Stacked RNNs consist of multiple networks where the output of one layer serves as the input to a subsequent layer


Figures are from https://reurl.cc/1389mD

Stacked RNNs

➢ Stacked RNNs generally outperform single-layer networks.
  o The network induces representations at differing levels of abstraction across layers
  o The initial layers of stacked networks induce representations that serve as useful abstractions for further layers

➢However, as the number of stacks is increased the training costs rise quickly.

Bidirectional RNNs

➤ In many applications, RNNs have to access the entire input sequence.

➤ Bidirectional RNN was introduced.
  o It combines two independent bidirectional RNNs, one where the input is processed from the start to the end, and the other from the end to the start
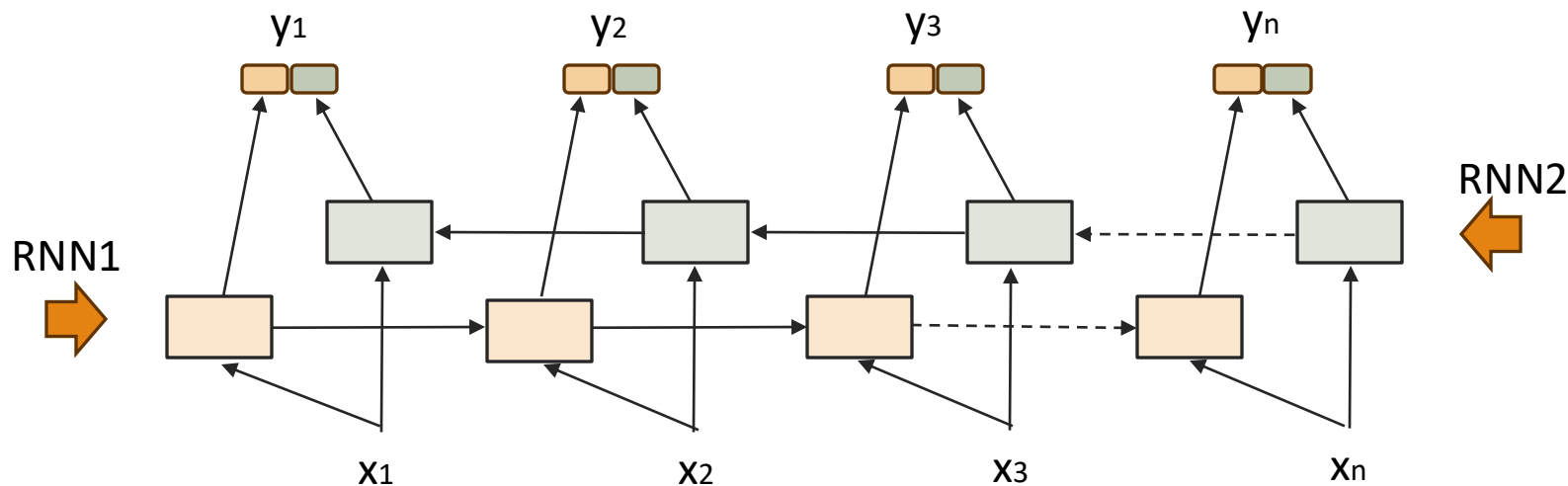
Bidirectional RNNs for Sequence Classification

➢ The final hidden units from the forward and backward passes are combined to represent the entire sequence.

➢ This combined representation serves as input to the subsequent classifier.
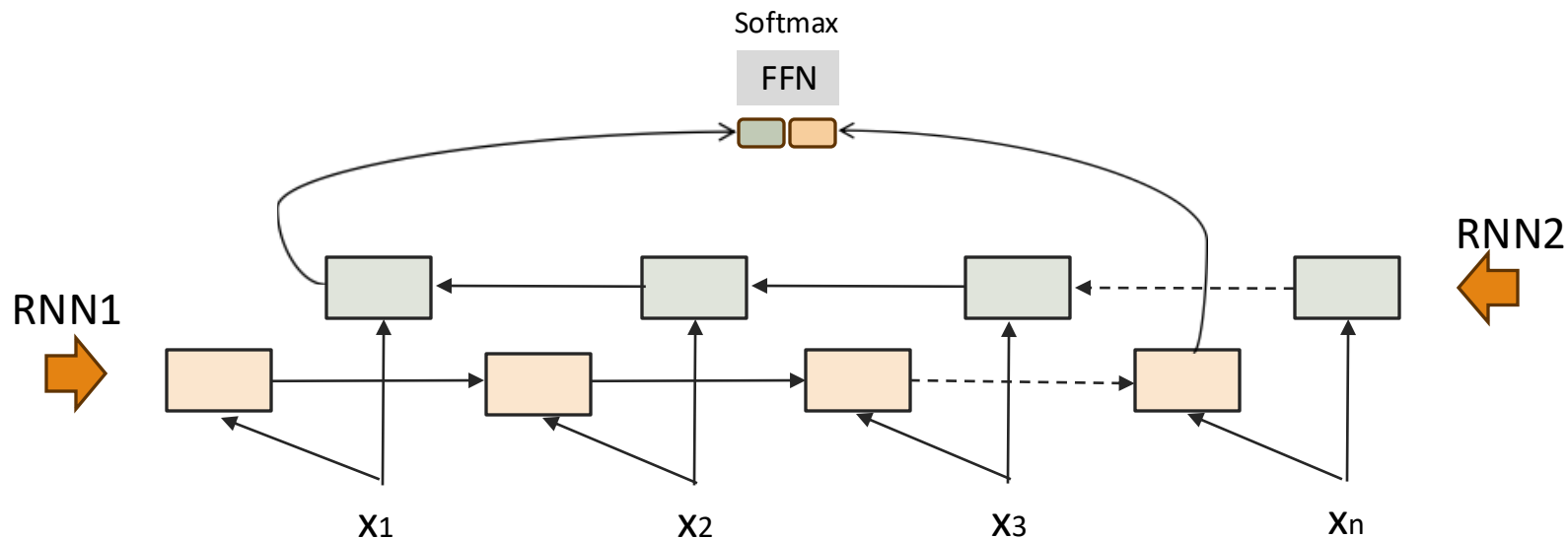
Summary

➢ **LM:** Models that assign probabilities to sequences of words

➢ **N-gram LM:** Traditional statistical model of language

Embedding approaches:

   ➢ **Sparse vectors:** Use contexts to encode the word embedding. (TF-IDF, PPMI)

   ➢ **Dense vectors:** Apply self-supervised training. (Word2Vec, Contextualized Embeddings)

Neural LMs:

   ➢ **FFN:** Fully-connected dense neural network.

   ➢ **RNN:** Recurrent, store temporal hidden-state.