自然語言處理與應用
Natural Language Processing and Applications

Attention and Sub-word Tokenization

Instructor: 林英嘉 (Ying-Jia Lin)
2025/03/17

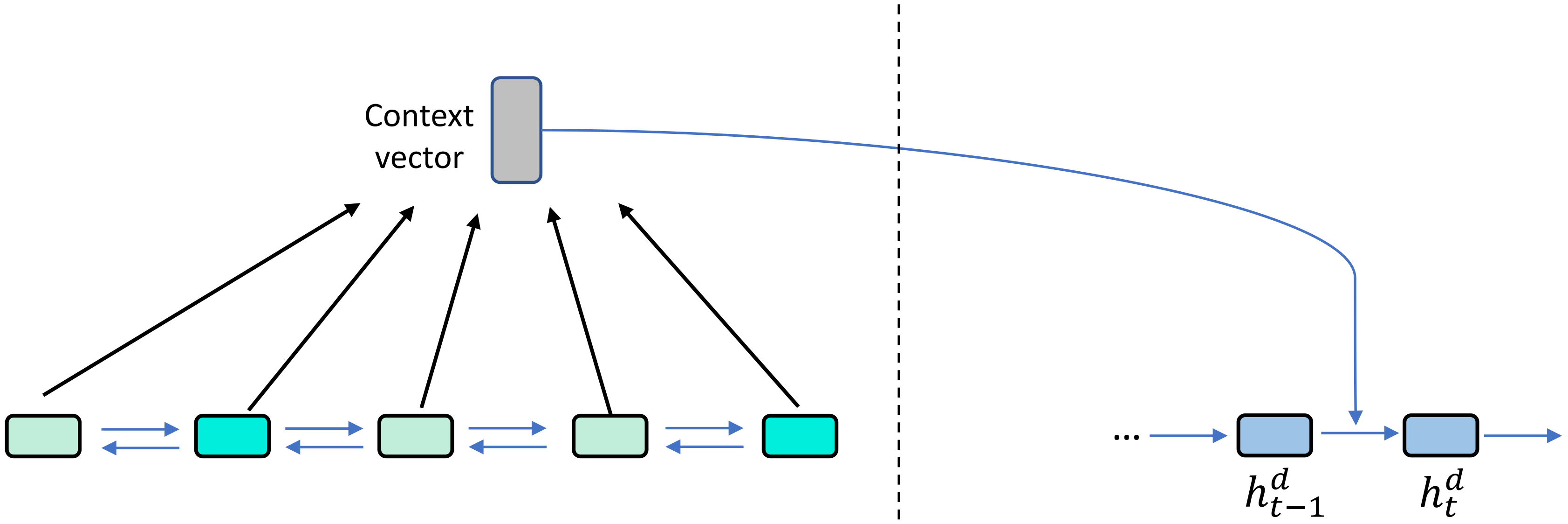
Course GitHub

Slido # NLP0317

自然語言處理基本功 (序列模型)

- Week 4 – Week 5
  - 遞迴神經網路 / PyTorch Tutorial
  - 注意力機制以及 Sub-word 分詞法

PyTorch



Encoder time steps

Decoder time steps

Syllabus

Version 0217

Tutorial

Exam

Report

Week	Topic	Note
1	自然語言處理介紹以及本課程綱要	
2	統計語言模型與基本詞向量方法	
3	詞嵌入模型	HW1
4	遞迴神經網路 / PyTorch Tutorial	
5	注意力機制以及 Sub-word 分詞法	HW2
6	自注意力機制以及文字生成策略	
7	BERT 與它的好夥伴	
8	期中考	HW3
9	HuggingFace 函式庫實作教學 (1)	
10	現代自然語言處理：GPT-3, InstructGPT, RLHF, DeepSeek	
11	HuggingFace 函式庫實作教學 (2)	HW4
12	如何訓練大模型？輕量化微調方法	
13	檢索式生成方法	HW5
14	檢索式生成方法實作教學	
15	小組實作成果報告 (1)	
16	小組實作成果報告 (2)	

Syllabus

Version 0317
(Please vote)

Tutorial

Exam

Report

Week	Topic	Note
1	自然語言處理介紹以及本課程綱要	
2	統計語言模型與基本詞向量方法	
3	詞嵌入模型	HW1
4	遞迴神經網路 / PyTorch Tutorial	
5	注意力機制以及 Sub-word 分詞法	
6	自注意力機制以及文字生成策略	HW2
7	BERT 與它的好夥伴	
8	期中考	
9	HuggingFace 函式庫實作教學 (1)	HW3
10	現代自然語言處理：GPT-3, InstructGPT, RLHF, DeepSeek	
11	HuggingFace 函式庫實作教學 (2)	
12	如何訓練大模型？輕量化微調方法	HW4
13	檢索式生成方法	
14	檢索式生成方法實作教學	
15	小組實作成果報告 (1)	
16	小組實作成果報告 (2)	

Scoring

Version 0317 (Please vote)

4 times

- Assignments ~~(5 times)~~: 40% 10% for each (HW1: 2% for free)
- Quizzes: 1% x 10 times
- Mid-term Exam: 20%
- Term Project: 30%

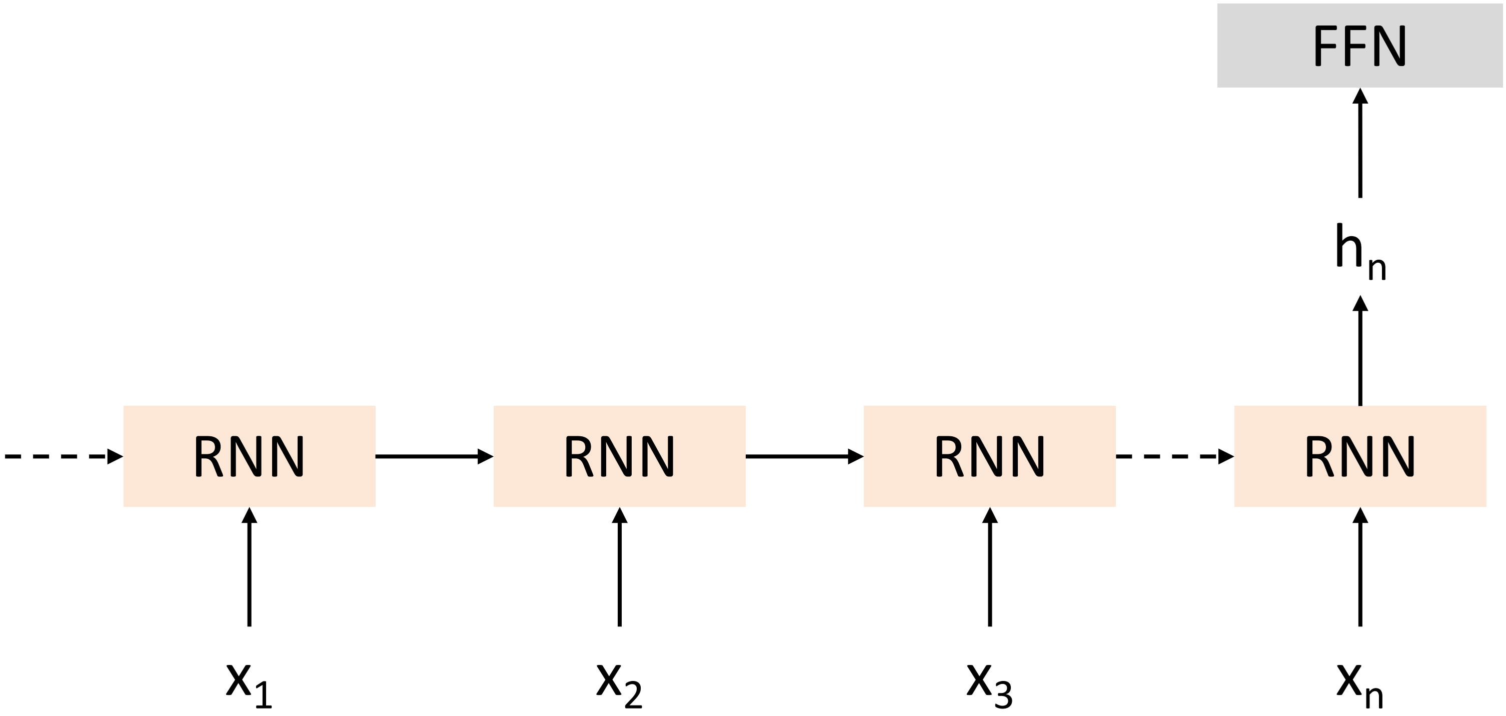
Outline

- Announcement, GAI money [10 min]
- Attention (Not self-attention a.k.a. Transformers) [30 min]
- Sub-word tokenization
  - Byte-pair Encoding (BPE) [30 min]
  - Unigram Language Model [10 min]
- PyTorch [60 min]
- Quiz [10 min]

Attention

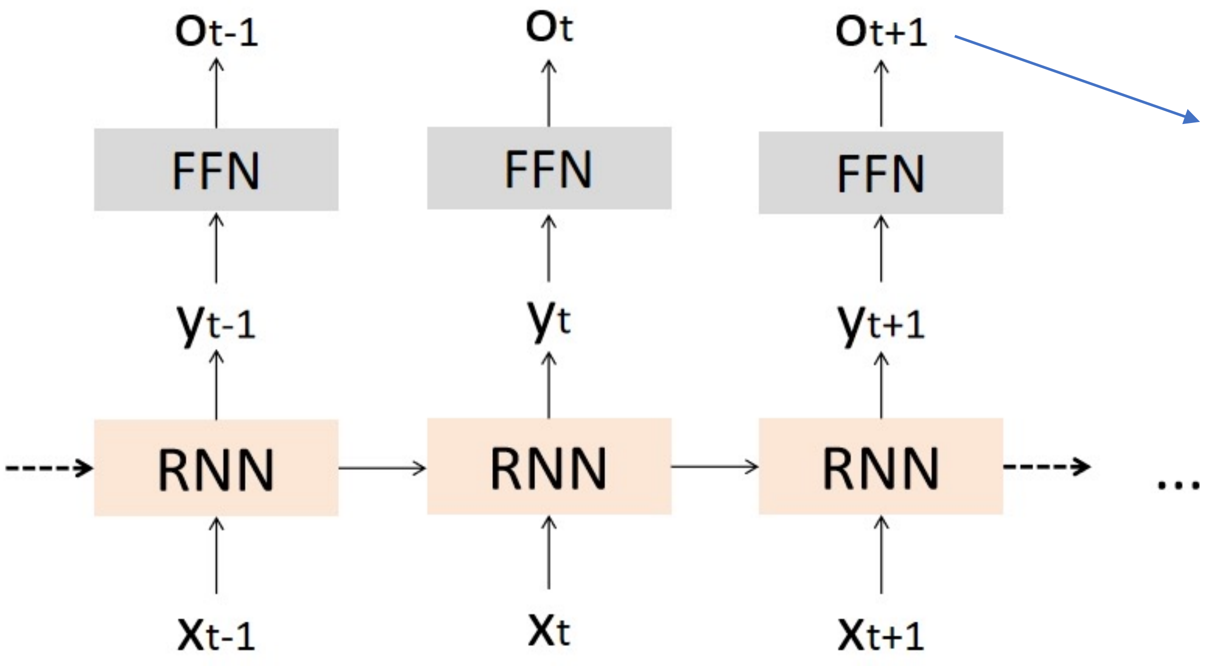
RNNs for Sequence Classification

- RNNs classify the entire sequences rather than the tokens within them.
- Take the hidden layer for the last token of the text.



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.



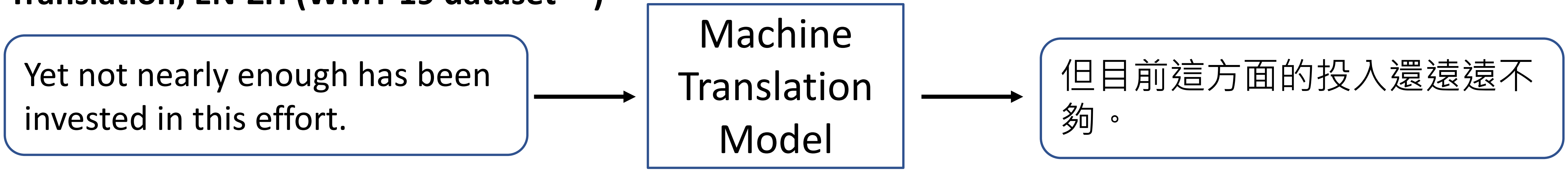
(Nx5)
matrix

	Meaning
B	Beginning
I	Inside
O	Outside
E	End
S	Single

(取最大值)

Machine Translation

Translation, EN-ZH (WMT-19 dataset [1])



Yet not nearly enough has been invested in this effort.

Machine Translation Model

但目前這方面的投入還遠遠不夠。

EMNLP 2024

NINTH CONFERENCE ON MACHINE TRANSLATION (WMT24)

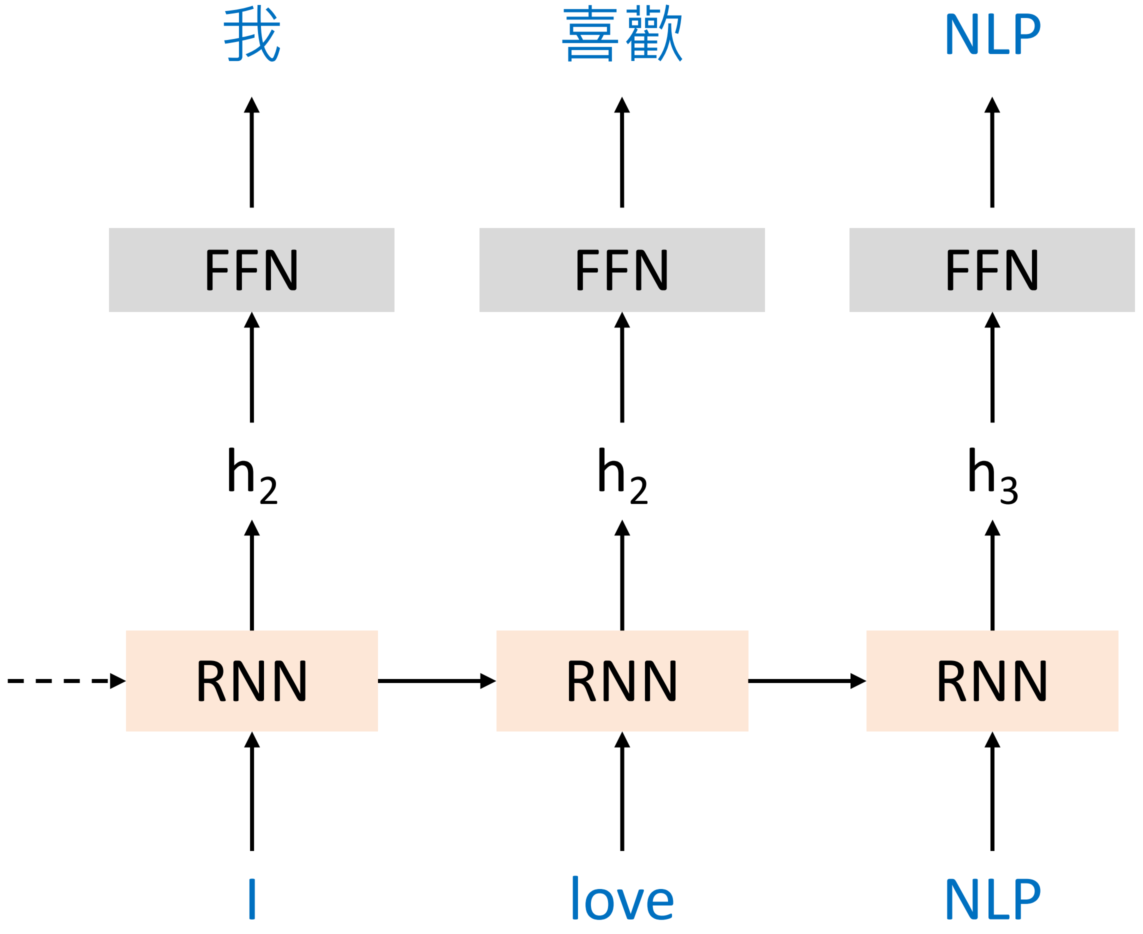
November 15-16, 2024
Miami, Florida, USA

[HOME] [PROGRAM] [PAPERS] [AUTHORS]
TRANSLATION TASKS: [GENERAL MT (NEWS)] [LOW-RESOURCE LANGUAGES OF SPAIN] [INDIC MT] [CHAT TASK] [BIOMEDICAL] [MULTIINDIC22MT TASK] [ENGLISH-TO-LOWRES MULTIMODAL MT TASK] [NON-REPETITIVE] [PATENT] [LITERARY]
EVALUATION TASKS: [METRICS TASK] [MT TEST SUITES] [QUALITY ESTIMATION]
OTHER TASKS: [OPEN LANGUAGE DATA INITIATIVE]

Figure source: https://www2.statmt.org/wmt24/mtdata/
[1] https://huggingface.co/datasets/wmt/wmt19/viewer/zh-en

RNNs for Machine Translation?

Method 1 (逐字對齊)

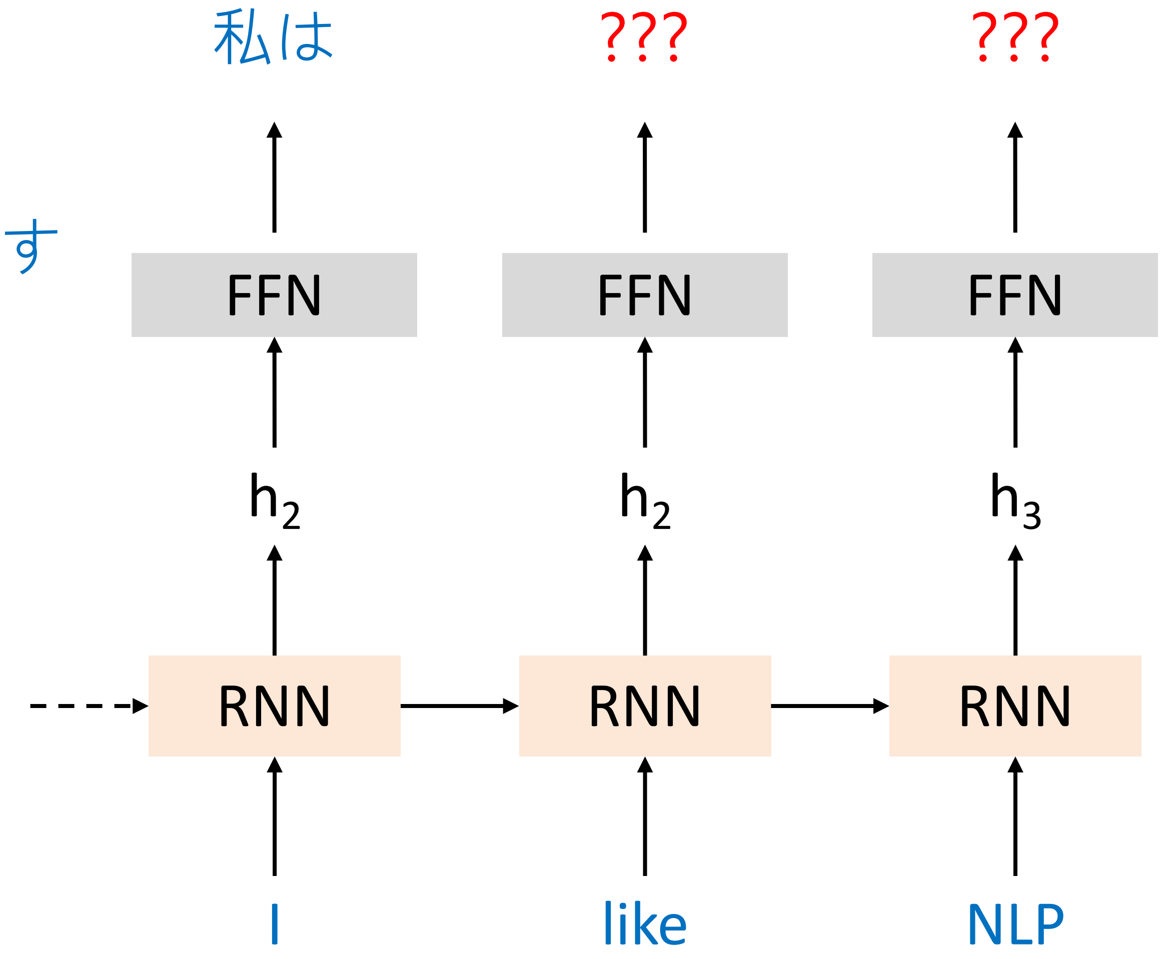


RNNs for Machine Translation?

Method 1 (逐字對齊)

Target sequence:

私は NLP が好きです

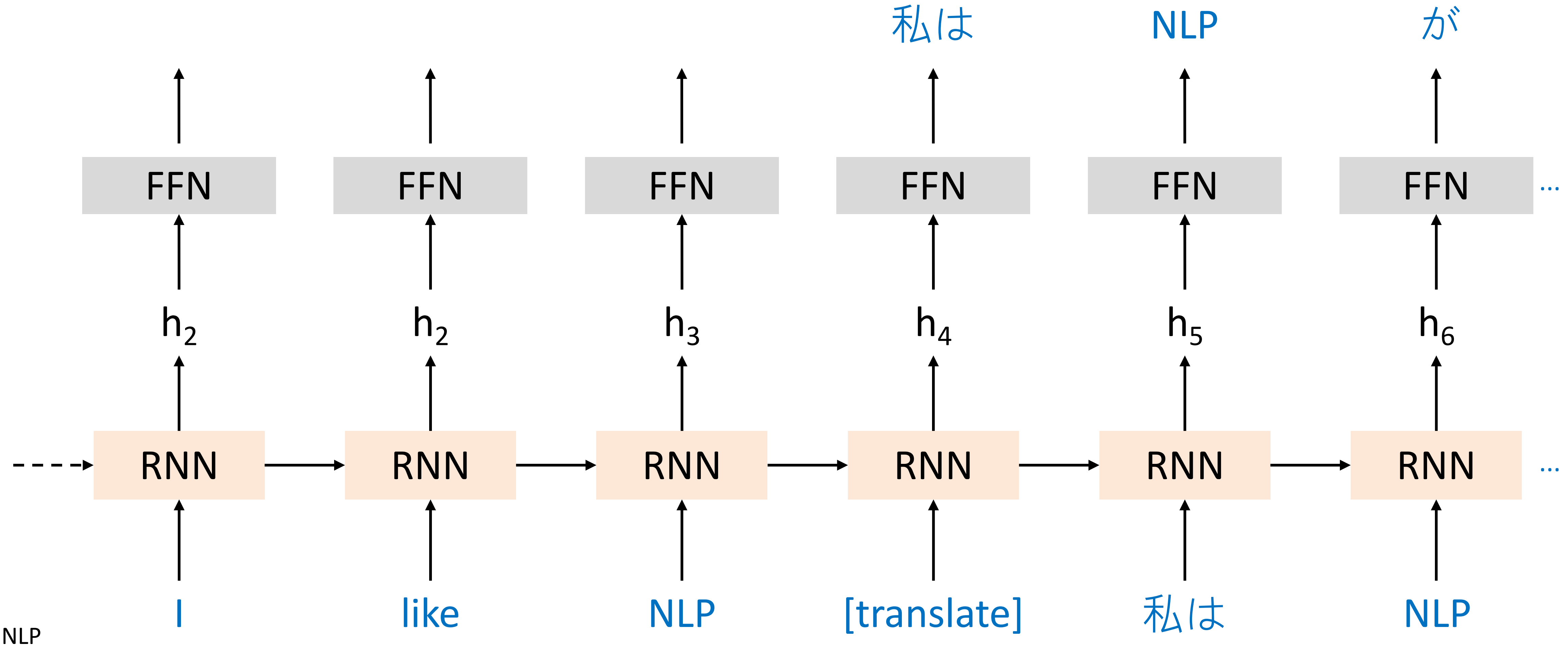


缺點：在不能逐字對齊的資料上效果不佳

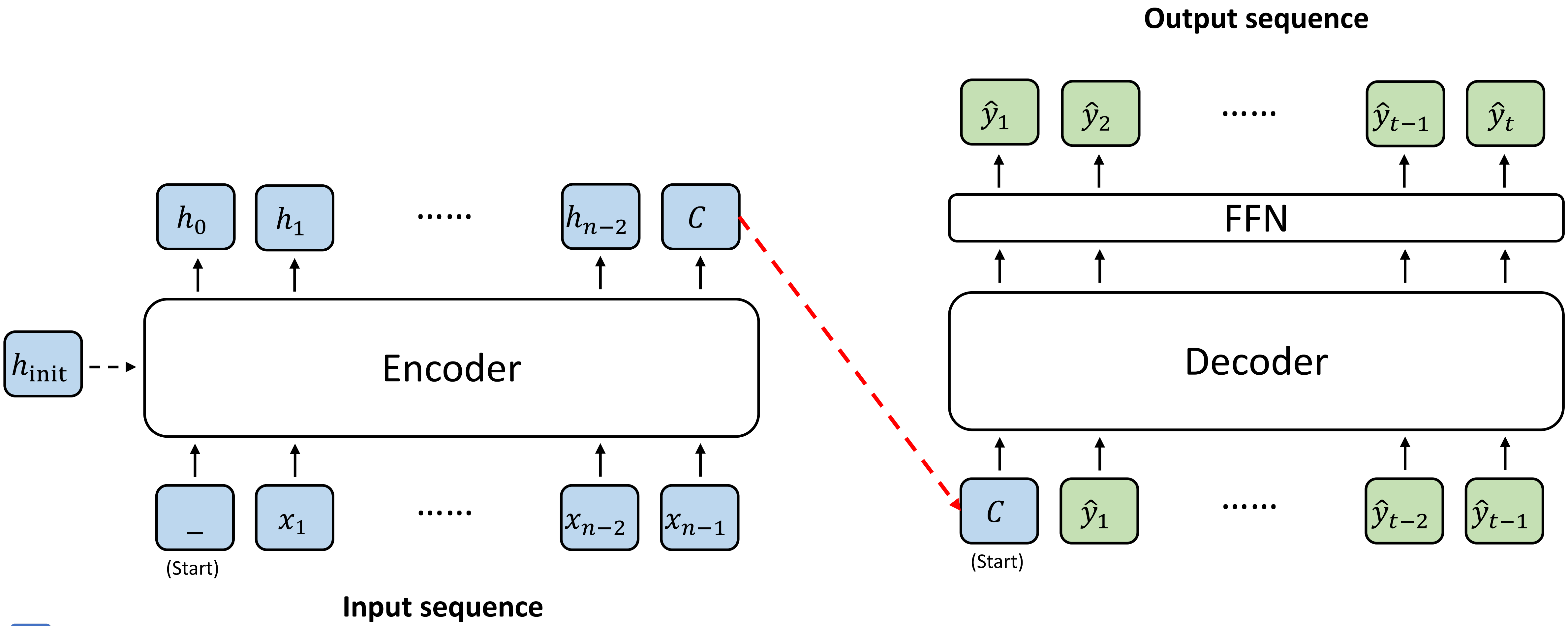
RNNs for Machine Translation?

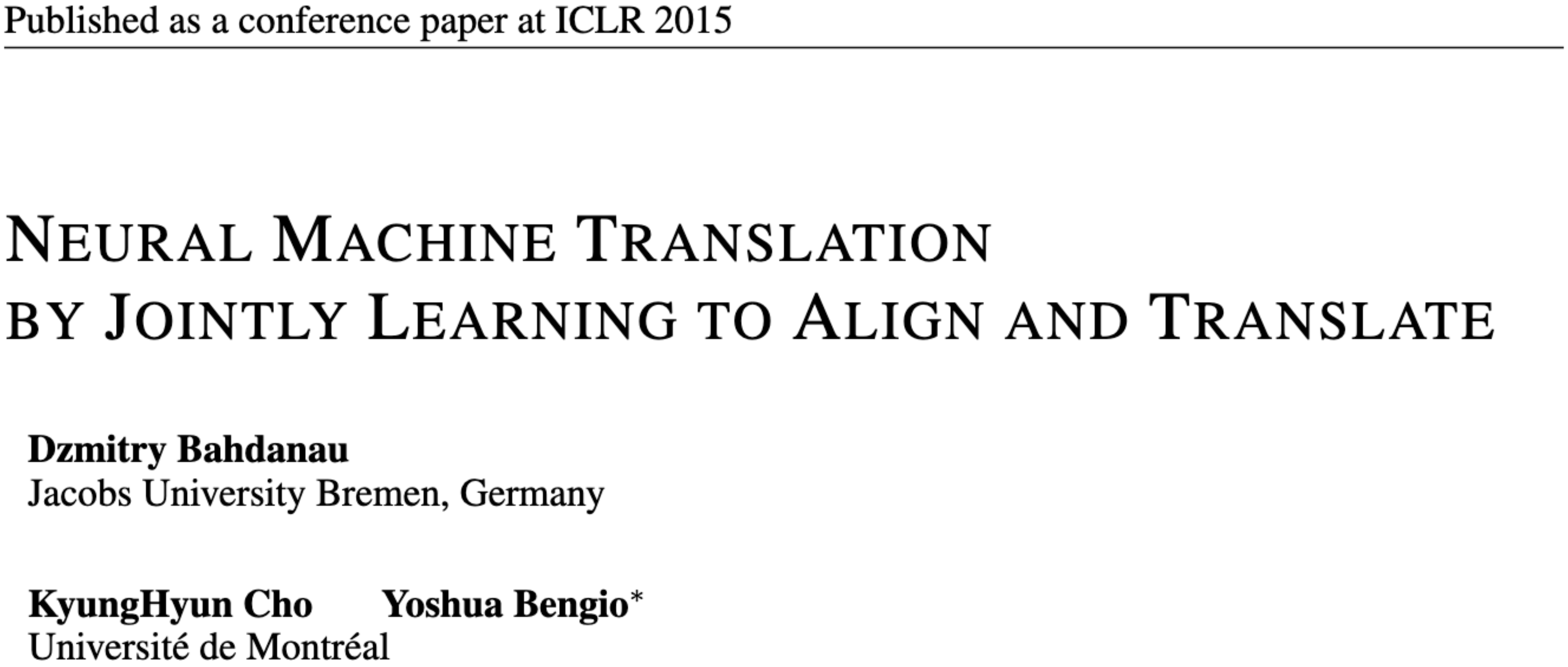
缺點：長距離的關係難以被記憶

Method 2 (利用特殊 token)



Encoder-Decoder Model





Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

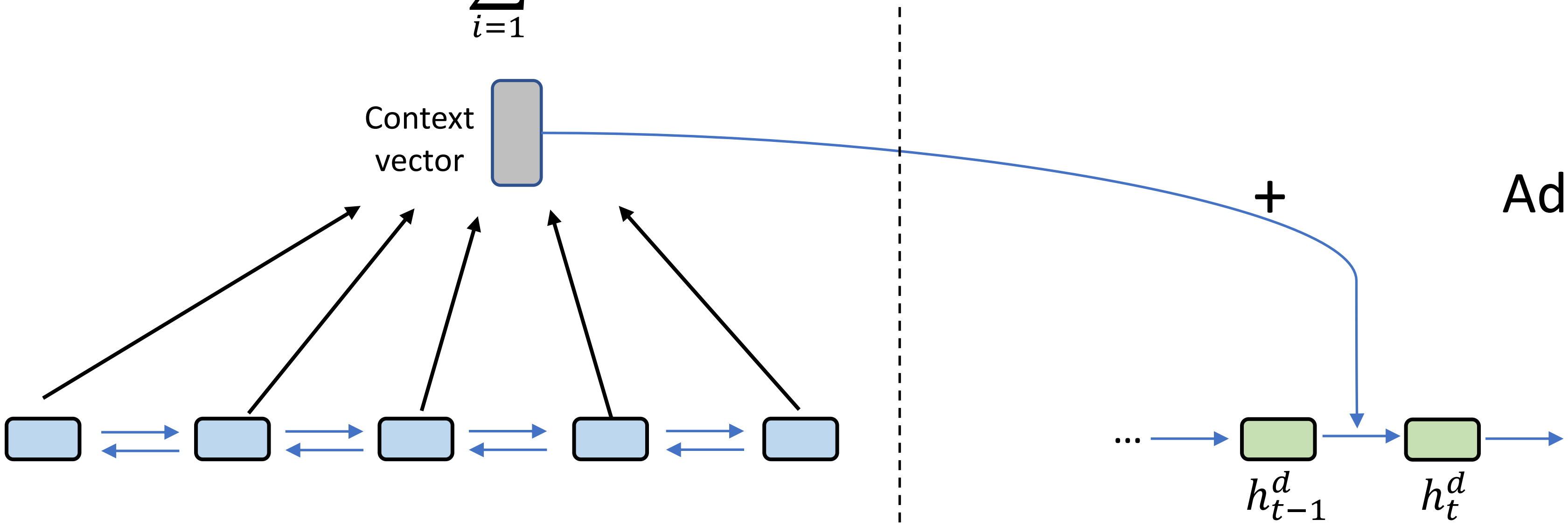
Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Attention mechanism (Bahdanau et al. 2014)

i: encoder timestep
t: decoder timestep

$$c_t^e = \sum_{i=1}^{n} \alpha_{ti}^e h_i^e$$



Additive attention

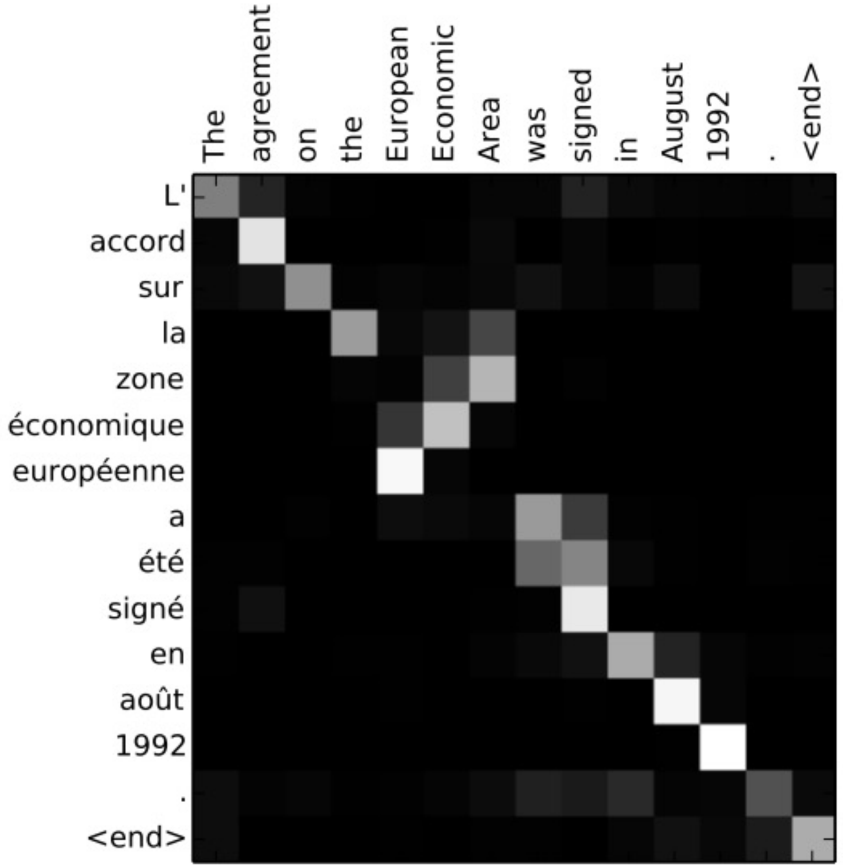
$$h_i^e = [h_i^{e\_fwd} \parallel h_i^{e\_bwd}]$$

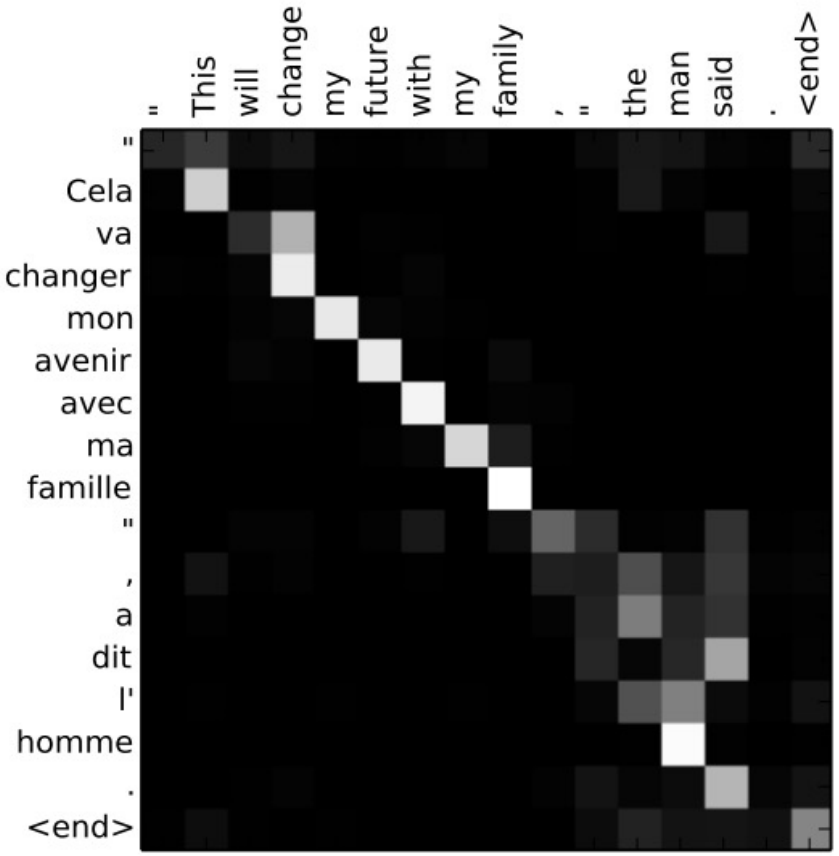
$$e_{ti} = \text{align}(h_{t-1}^d, h_i^e)$$

$$\alpha_{ti}^e = \frac{\exp(e_{ti})}{\sum_{j=1}^{n} \exp(e_{tj})}$$

Attention

An example of machine translation.





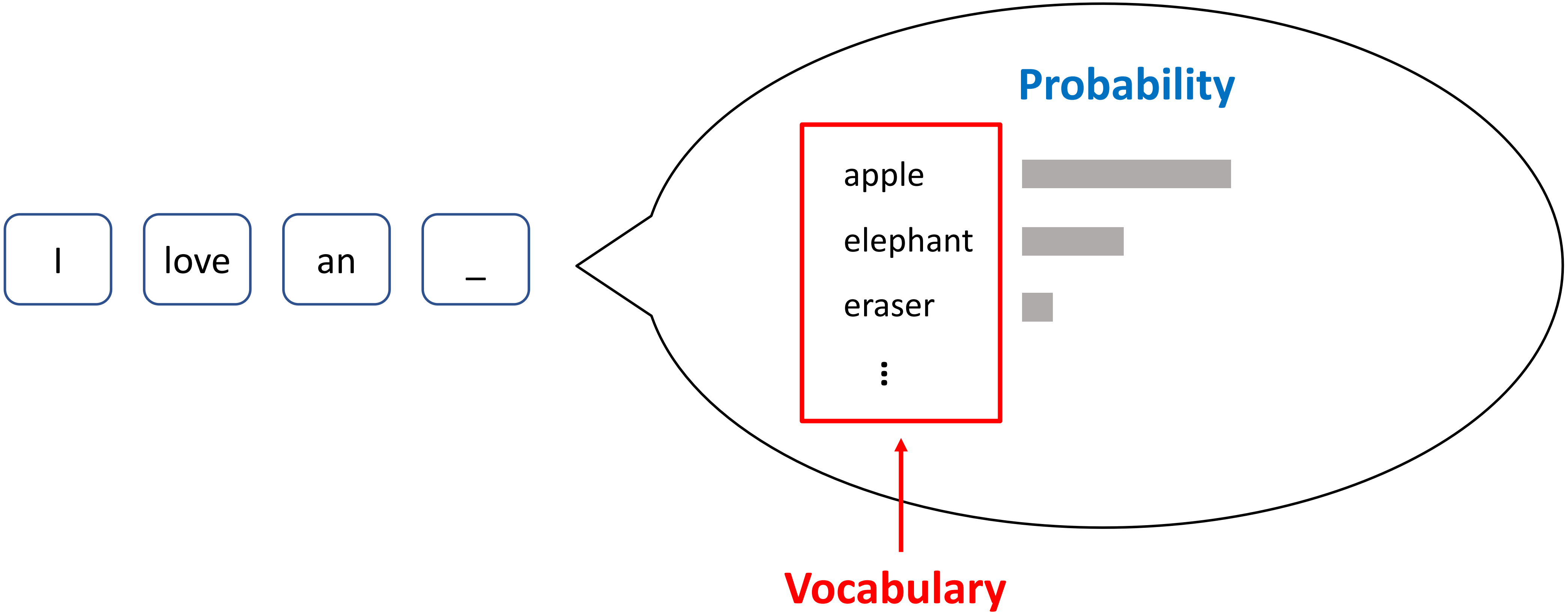
Other Attention Approaches

Dot-product Attention: https://arxiv.org/abs/1508.04025

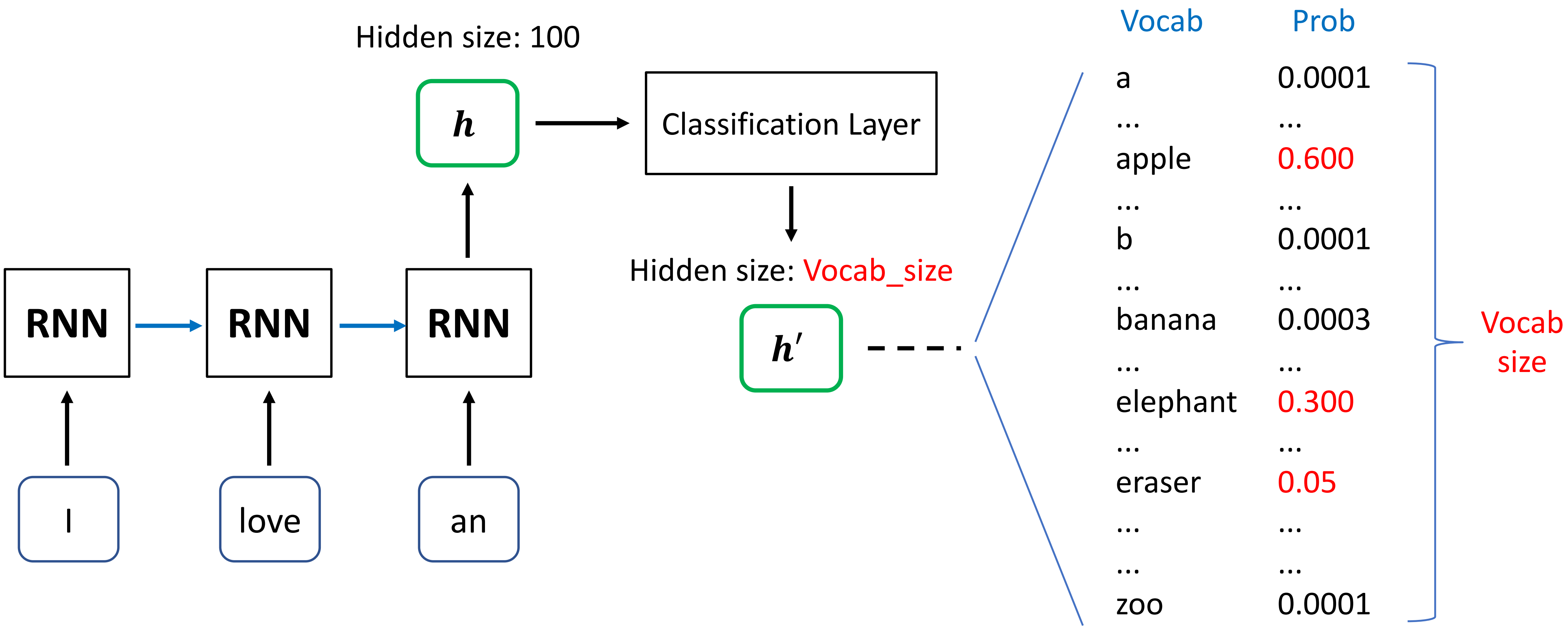
https://lilianweng.github.io/posts/2018-06-24-attention/

Sub-word Tokenization

Recap: Word Representations

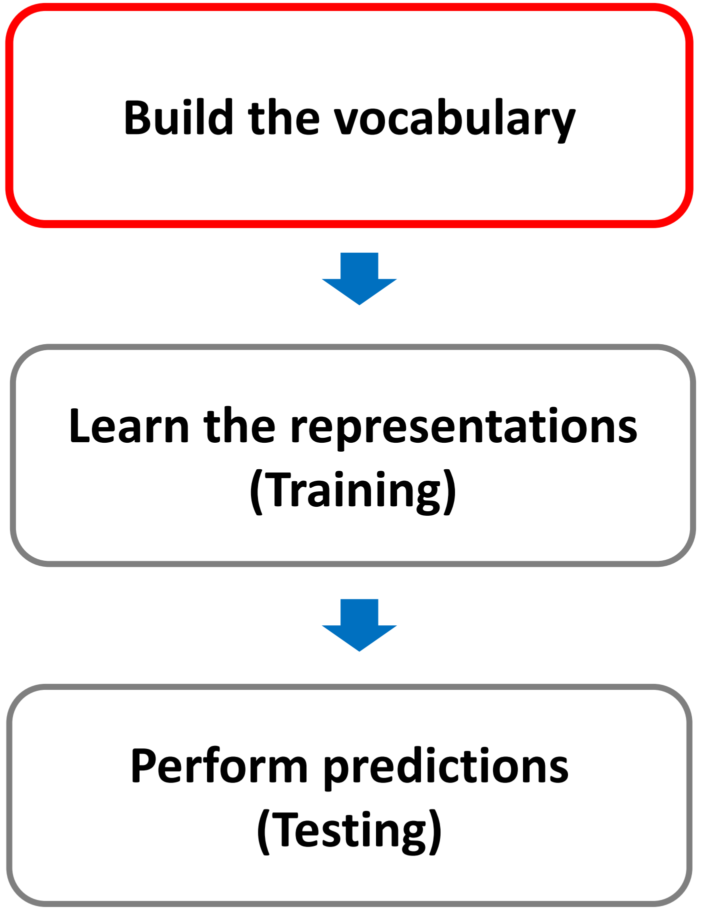


Recap: Word Representations (Details)



Vocab	Prob
a	0.0001
…	…
apple	0.600
…	…
b	0.0001
…	…
banana	0.0003
…	…
elephant	0.300
…	…
eraser	0.05
…	…
…	…
zoo	0.0001

Basic Pipeline of Natural Language Processing



Build the vocabulary

Learn the representations (Training)

Perform predictions (Testing)

How to build the vocabulary?

- Segment words based on delimiters (e.g., white spaces).
- Then we can collect the words from training corpora.

I love apples. I like apples and pineapples.

(You can also perform stemming after word segmentation!)

Vocabulary

Word	**ID**
and	0
apples	1
I	2
like	3
love	4
pineapples	5
.	6
<UNK>	7

Unknown words (not shown up in the training corpora)

Segmentation vs. Tokenization

- All tokenization is segmentation, but not all segmentation is tokenization.
- Segmentation can be:
  - Word segmentation from a sentence
  - Sentence segmentation from a document
- Tokenization can be:
  - Word segmentation from a sentence (then `words` become `tokens`)
  - Sub-word tokenization from a word or sentence

Issues of Delimiter-based Segmentation

- Only work for Western languages.
  - Cannot work for Chinese, Japanese, ... (但不是 Subword 能解決的問題)
- Cannot handle unseen words (not shown up in the training corpora)
  - A misspelled word contains morphological information but become an unknown word.

Issues of Delimiter-based Segmentation (Continued.)

- For machine translation, there is not always a 1-to-1 correspondence between source and target words since compound words may exist in target language.

- For example, sewage water treatment plant (English) -> Abwasserbehandlungsanlage (German)

sewage water treatment plant/facility

Sub-word units are favored.

Summary

- 為什麼我們需要 Sub-word Tokenization?
  - 為了解決 OOV 的問題
    - 有些語言特別容易出現長單詞，例如 Abwasserbehandlungsanlage 這種可拆式複合字
    - 沒看過的字 (Unseen words)
  - 擴增字典大小，如果原本字典在 30,000–50,000 個字，那如果用 Sub-word 的方式建立字典就可以在原本的字典大小範圍表達更多的字

Common Sub-word Tokenization Algorithms

- Byte Pair Encoding (Sennrich et al., 2016)[1]
- WordPiece (Schuster and Nakajima, 2012)[2]
- Unigram Language Model (Kudo, 2018)[3]

[1] Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units." Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), 2016.
[2] Schuster, Mike, and Kaisuke Nakajima. "Japanese and korean voice search." 2012 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2012.
[3] Kudo, Taku. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL). 2018.

Sub-word Tokenization (1)

- Given a training corpus, we first turn each word into a sequence of characters (separated by spaces)

Training corpus

low low low low low lower lower newest newest newest newest newest newest widest widest widest

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w e s t </w>	6
w i d e s t </w>	3

- </w> is a special end-of-word symbol, allowing us to restore the original tokenization.

Sub-word Tokenization (1)

- Initialize the vocabulary with the existing characters:

low low low low low lower lower newest newest newest newest newest newest widest widest widest

↓

Initial vocab: </w>, d, e, i, l, n, o, r, s, t, w

(逗號不算)

Sub-word Tokenization (2)

- Find the character **pair** with the highest frequency

Word	**Frequency**
l o w </w>	5
l o w e r </w>	2
n e w e s t </w>	6
w i d e s t </w>	3

Found “e s” with the highest frequency 6+3=9

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w

Sub-word Tokenization (3)

- Add the pair with the highest frequency to the vocab

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w e s t </w>	6
w i d e s t </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es

Sub-word Tokenization (4)

- Merge the characters by replacing all words in the corpus with the newly added pair.

Word	**Frequency**
l o w </w>	5
l o w e r </w>	2
n e w **es** t </w>	6
w i d **es** t </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es

Sub-word Tokenization (2)

Repeated (2)-(4) according to `num_merges`

- Find the character **pair** with the highest frequency

Word	**Frequency**
l o w </w>	5
l o w e r </w>	2
n e w es t </w>	6
w i d es t </w>	3

Found “es t” with the highest frequency 6+3=9

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es

Sub-word Tokenization (3)

Repeated (2)-(4)
according to `num_merges`

- Add the pair with the highest frequency to the vocab

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w es t </w>	6
w i d es t </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es est

Sub-word Tokenization (4)

Repeated (2)-(4)
according to `num_merges`

- Merge the characters by replacing all words in the corpus with the newly added pair.

Word	**Frequency**
l o w </w>	5
l o w e r </w>	2
n e w **est** </w>	6
w i d **est** </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est

Sub-word Tokenization (2)

Repeated (2)-(4) according to `num_merges`

- Find the character **pair** with the highest frequency

Word	**Frequency**
l o w </w>	5
l o w e r </w>	2
n e w est </w>	6
w i d est </w>	3

Found “est </w>” with the highest frequency 6+3=9

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est

Sub-word Tokenization (3)

Repeated (2)-(4)
according to `num_merges`

- Add the pair with the highest frequency to the vocab

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w est </w>	6
w i d est </w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>

Sub-word Tokenization (4)

Repeated (2)-(4) according to `num_merges`

- Merge the characters by replacing all words in the corpus with the newly added pair.

Word	**Frequency**
l o w </w>	5
l o w e r </w>	2
n e w **est</w>**	6
w i d **est</w>**	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>

Sub-word Tokenization (2)

Repeated (2)-(4) according to `num_merges`

- Find the character **pair** with the highest frequency

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w est</w>	6
w i d est</w>	3

Found “l o” with the highest frequency 5+2=7

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>

Sub-word Tokenization (3)

Repeated (2)-(4)
according to `num_merges`

- Add the pair with the highest frequency to the vocab

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w est</w>	6
w i d est</w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

Sub-word Tokenization (4)

Repeated (2)-(4) according to `num_merges`

- Merge the characters by replacing all words in the corpus with the newly added pair.

Word	Frequency
lo w </w>	5
lo w e r </w>	2
n e w est</w>	6
w i d est</w>	3

Current Vocabulary: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

Finish Sub-word Leaning

- Assume `num_merges`=4 (we just repeated four times.)
- `num_merges` is a hyperparameter that you need to set for BPE.
- The learned vocabulary is: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

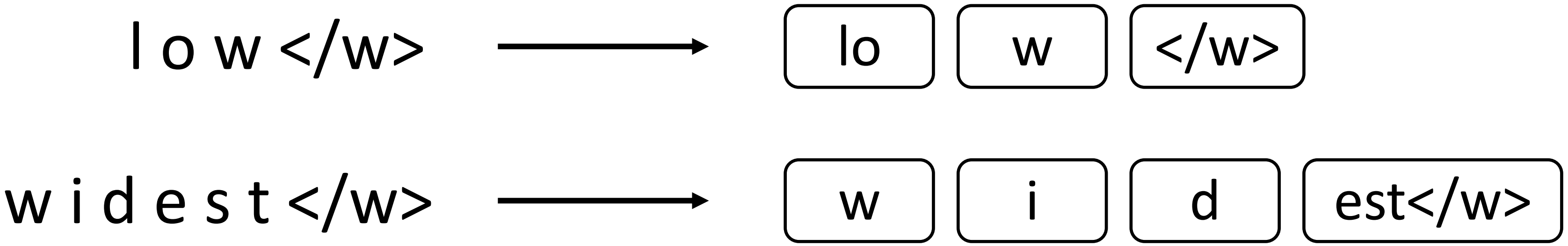
Tokenization with Learned BPE

- **The learned vocabulary** is: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

- We first turn each word into a sequence of characters with </w> placed at the end of a word, same as the first step during the learning phase.

- Then we can merge the characters according to **the learned vocabulary**.

- Examples:

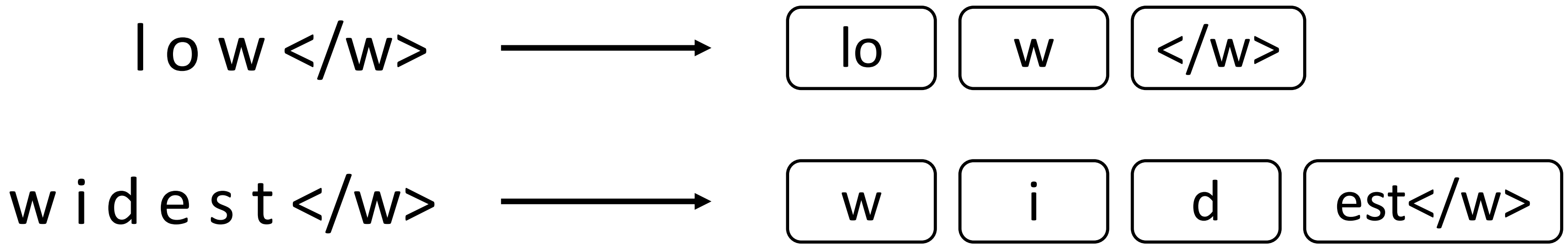


l o w </w> → lo w </w>

w i d e s t </w> → w i d est</w>

[注意事項] Tokenization with Learned BPE

- 合併的順序：left to right
- 每次合併的對象選擇：最長越好
  - 例如：假如字典中同時有 low 跟 lo，那 low 在該次合併時會被選擇，因為 low 比 lo 還長



l o w </w> → lo w </w>

w i d e s t </w> → w i d est</w>

Properties of BPE

- The final learned vocabulary size = **initial** size + `num_merges`

  Initial vocab: </w>, d, e, i, l, n, o, r, s, t, w

  Learned vocab: </w>, d, e, i, l, n, o, r, s, t, w, es, est, est</w>, lo

- This algorithm is based on statistics, so frequent sub-word units in provided corpora will be put to the learned vocabulary.

Why do we need Sub-word Tokenization?

- With sub-word tokenization algorithms, we can handle representations for unknown words (or mis-spelled words).
- In machine translation , the compound word issues between source and target languages can be alleviated.
- State-of-the-art pre-trained language models (e.g., GPT-3, BERT) adopt sub-word tokenization algorithms before pre-training.

Limitations of Sub-word Tokenization

- (Not many disadvantages for sub-word tokenization)
- The hyperparameter `num_merges` needs to be tuned.
- Once the learned vocabulary is created, it becomes fixed. The algorithm needs to be re-run after adding new data.

Unigram Language Model (Kudo 2018)

[KUDO, 2018] KUDO, TAKU. "SUBWORD REGULARIZATION: IMPROVING NEURAL NETWORK TRANSLATION MODELS WITH MULTIPLE SUBWORD CANDIDATES." ACL 2018.

The Problem of BPE

- BPE splits a sentence with larger sub-words in the vocabulary in default.

greedy, deterministic, and left-to-right

Original: Hello world

BPE: Hell / o /world ← **May be sub-optimal.**

Other choices: H / ello / world

He / llo / world

He / l / l / o / world

H / el / l / o / world

Tokenization with Probabilities?

H	0.03
He	0.001
Hell	0.007
el	0.002
ello	0.024
llo	0.062
l	0.003
o	0.055
world	0.011

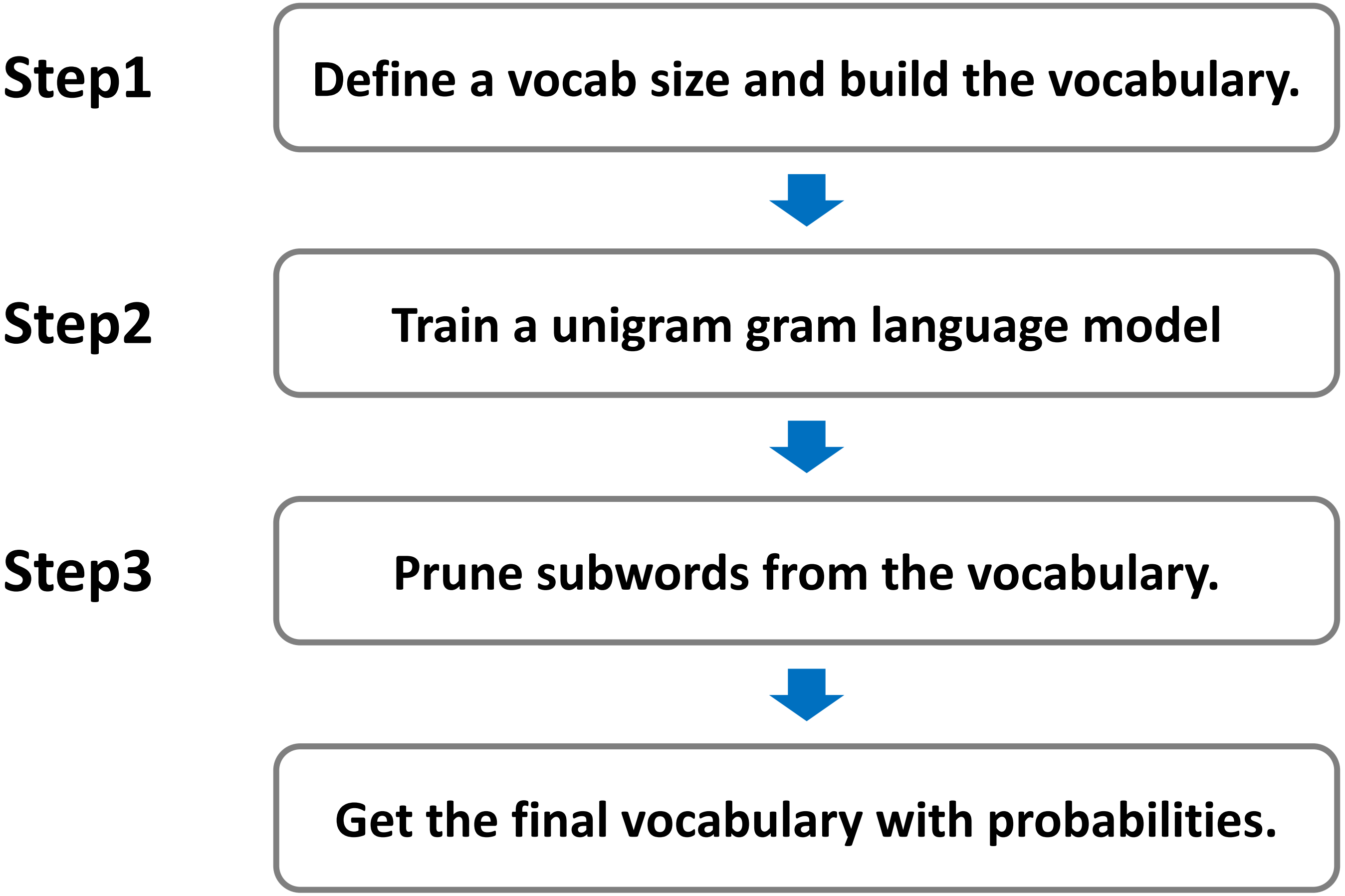
Sub-word Tokens	Product of occurrence probabilities
Hell / o /world	0.007 * 0.055 * 0.011 = 0.000004235
H / ello / world	0.03 * 0.024 * 0.011 = 0.00000792
He / llo / world	0.001 * 0.062 * 0.011 = 0.000000682
He / l / l / o / world	0.001 * 0.003 * 0.003 * 0.055 * 0.011 = 5.445E-12
H / el / l / o / world	0.03 * 0.002 * 0.003 * 0.055 * 0.011 = 1.089E-10

(BPE example)

Unigram Language Model Tokenization

- Similar to Byte Pair Encoding (BPE), Unigram Language Model Tokenization (ULM) provides an algorithm to tokenize a sentence into subwords.
- Unlike BPE, ULM performs tokenization based on joint probabilities for each sentence.
- In other words, every token in the vocabulary has each own probability trained from the corpus.

Steps of Unigram Tokenization



Step1 Define a vocab size and build the vocabulary.

Step2 Train a unigram gram language model

Step3 Prune subwords from the vocabulary.

Get the final vocabulary with probabilities.

Inference

- After a ULM is trained (字典被調整完之後), a word can be split into subwords according its joint probability.
- 例如 internationalization 的兩個選項：
  - P(international @@ ization) = 0.8 x 0.6 = 0.48 👍
  - P(internation @@ alization) = 0.77 x 0.5 = 0.385

Sub-word Tokenization Choices of PLMs

Sub-word Tokenization	Models
WordPiece (Schuster and Nakajima, 2012)	BERT, ALBERT, MT-DNN
BPE (Sennrich et al., 2016)	RoBERTa, XLM, GPT-1, GPT-2, GPT-3
Unigram (Kudo, 2018)	XLNet, T5, mT5

[Schuster and Nakajima, 2012] Schuster, Mike, and Kaisuke Nakajima. "Japanese and korean voice search." ICASSP 2012.
[Sennrich et al., 2016] Sennrich, Rico, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units. ACL 2016.
[Kudo, 2018] Kudo, Taku. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." ACL 2018.

Comparison between BPE and ULM

Aspect	ULM	BPE
Algorithm Type	Expectation-Maximization (EM) algorithm	Greedy algorithm without probabilistic modeling
Training Complexity	Slower	Faster
Segmentation Consistency	Can produce different segmentations for the same word	Produces consistent segmentations for the same word every time.
Speed at Inference Time	Slower	Faster
Downstream tasks	Better at machine translation (maybe)	Suitable for most tasks

Thank you!

Instructor: 林英嘉

✉ yjlin@cgu.edu.tw

TA: 吳宣毅

✉ m1161007@cgu.edu.tw