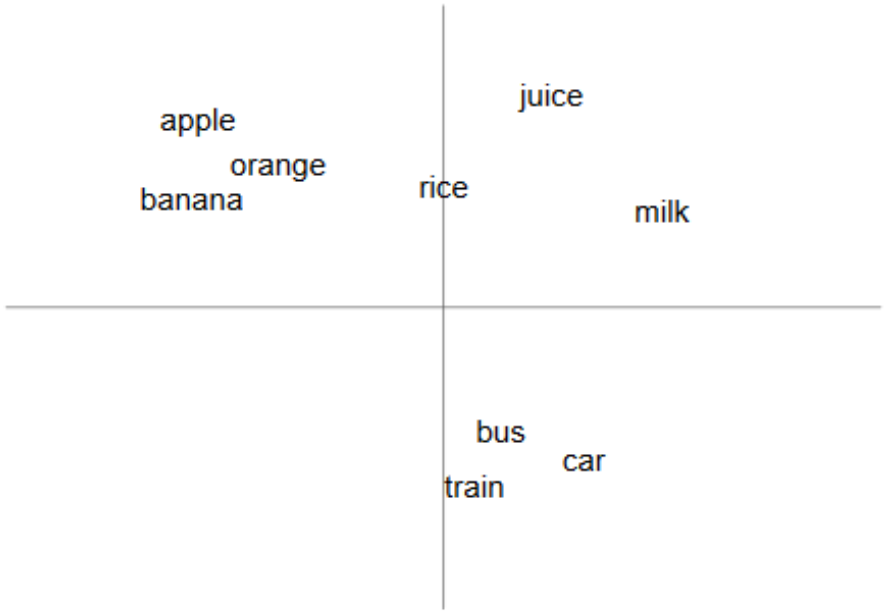
Lecture 2:
Natural Language Processing (I): Word Representation

Data Science, Fall 2020

Hong-Han Shuai

Thanks to Girish K from Texas A&M University and Prof. Hung-yi Lee from NTU for the slides.

“A word is known by the company it keeps”



Reference Materials

- Deep Learning for NLP by Richard Socher (http://cs224d.stanford.edu/)
- Tutorial and Visualization tool by Xin Rong (http://www-personal.umich.edu/~ronxin/pdf/w2vexp.pdf)
- Word2vec in Gensim by Radim Řehůřek (http://rare-technologies.com/deep-learning-with-word2vec-and-gensim/)

Word Representations

Traditional Method - Bag of Words Model	Word Embeddings
• Uses one hot encoding • Each word in the vocabulary is represented by one bit position in a HUGE vector. • For example, if we have a vocabulary of 10000 words, and "Hello" is the 4th word in the dictionary, it would be represented by: 0 0 0 1 0 0 . . . . . 0 0 0 0 • Context information is not utilized	• Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300) • Unsupervised, built just by reading huge corpus • For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02] • Dimensions are basically projections along different axes, more of a mathematical concept.

High dimension example

- [1 0 0 ……]
- [0 1 0 ……]
- [1 0 0 ……]

Curse (Con't)

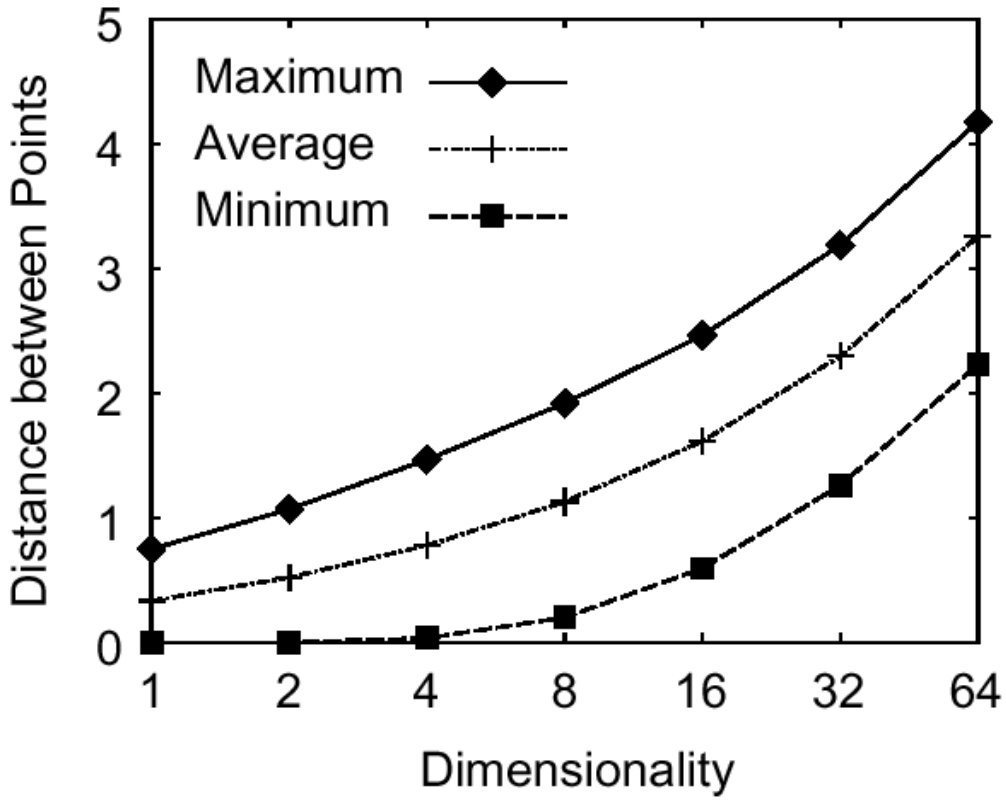


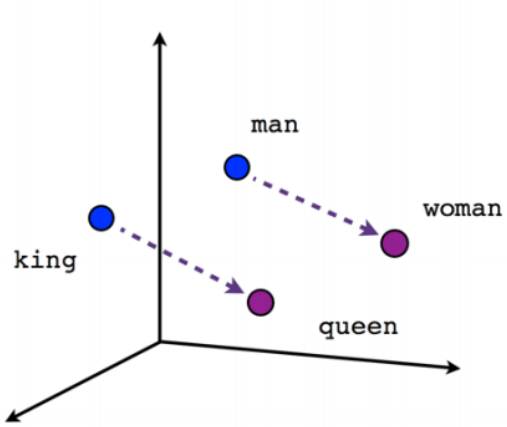
Figure 1. Distances among 100k points generated at random in a unit hypercube

Source: N. Katayama, S. Satoh. Distinctiveness Sensitive Nearest Neighbor Search for Efficient Similarity Retrieval of Multimedia Information. ICDE Conference, 2001.

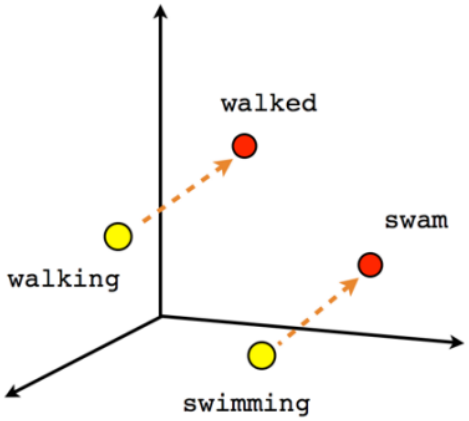
The Power of Word Vectors

- They provide a fresh perspective to ***ALL*** problems in NLP, and not just solve one problem.
- Technological Improvement
  - Rise of deep learning since 2006 (Big Data + GPUs + Work done by Andrew Ng, Yoshua Bengio, Yann Lecun and Geoff Hinton)
  - Application of Deep Learning to NLP – led by Yoshua Bengio, Christopher Manning, Richard Socher, Tomas Mikalov
- The need for unsupervised learning . (Supervised learning tends to be excessively dependant on hand-labelled data and often does not scale)

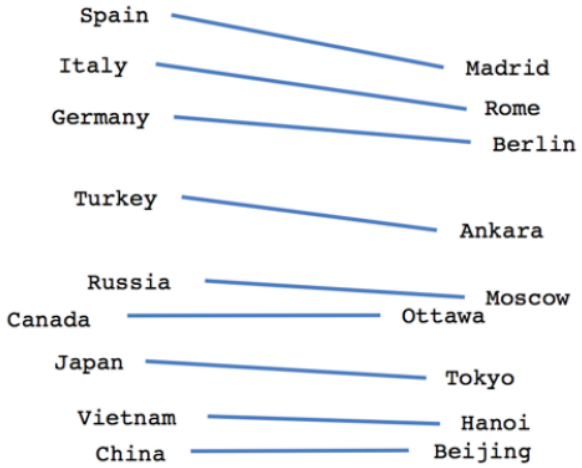
Examples



Male-Female



Verb tense



Country-Capital

vector[Queen] = vector[King] - vector[Man] + vector[Woman]

So, how exactly does Word Embedding ‘solve all problems in NLP’?

Applications of Word Vectors

1. Word Similarity

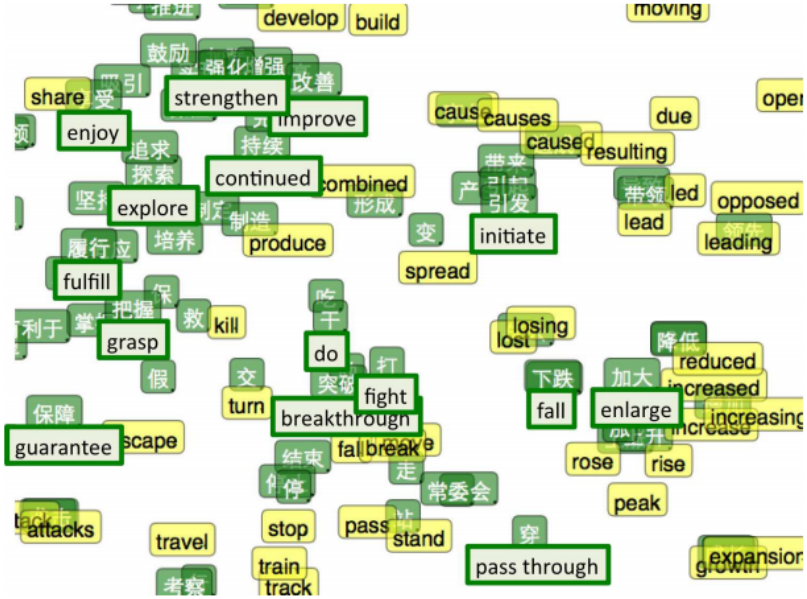
Classic Methods : Edit Distance, WordNet, Porter's Stemmer, Lemmatization using dictionaries

- Easily identifies similar words and synonyms since they occur in similar contexts
- Stemming (thought -> think)
- Inflections, Tense forms
- *eg. Think, thought, ponder, pondering,*
- *eg. Plane, Aircraft, Flight*

Applications of Word Vectors

2. Machine Translation

Classic Methods : Rule-based machine translation, morphological transformation



Applications of Word Vectors

3. Part-of-Speech and Named Entity Recognition

Classic Methods : Sequential Models (MEMM , Conditional Random Fields), Logistic Regression

	POS WSJ (acc.)	NER CoNLL (F1)
State-of-the-art*	97.24	89.31
Supervised NN	**96.37**	**81.47**
Unsupervised pre-training followed by supervised NN**	**97.20**	**88.87**
+ hand-crafted features***	97.29	89.59

Applications of Word Vectors

4. Relation Extraction

Classic Methods : OpenIE, Linear programing models, Bootstrapping

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Applications of Word Vectors

5. Sentiment Analysis

Classic Methods : Naive Bayes, Random Forests/SVM

- Classifying sentences as positive and negative
- Building sentiment lexicons using seed sentiment sets
- No need for classifiers, we can just use cosine distances to compare unseen reviews to known reviews.

Applications of Word Vectors

6. Clustering

- Words in the same class naturally occur in similar contexts, and this feature vector can directly be used with any conventional clustering algorithms (K-Means, agglomerative, etc). Human doesn't have to waste time hand-picking useful word features to cluster on.

7. Semantic Analysis of Documents

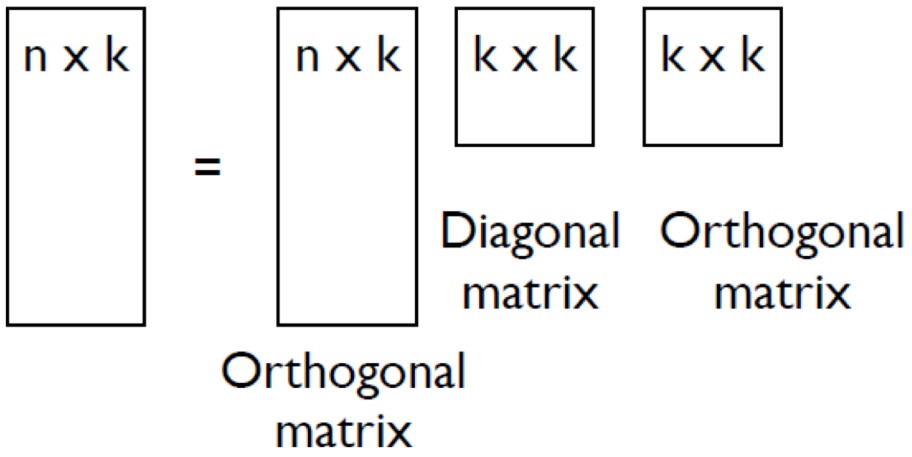
- Build word distributions for various topics, etc.

Building these magical vectors . . .

- How do we actually build these super-intelligent vectors, that seem to have such magical powers?
- How to find a word's friends?
- We will discuss the most famous methods to build such lower-dimension vector representations for words based on their context
  1. Co-occurrence Matrix with SVD
  2. word2vec (*Google)*
  3. Global Vector Representations (GloVe) (*Stanford)*

Co-occurrence Matrix with Singular Value Decomposition

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$$



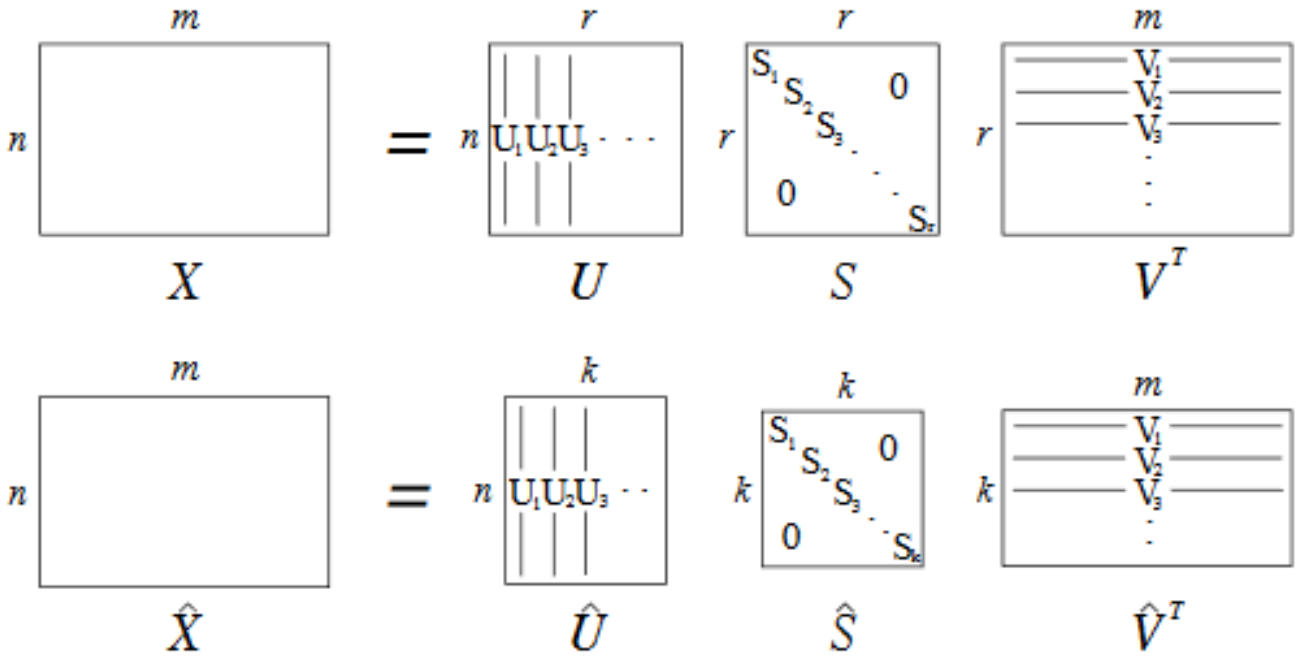
Building a co-occurrence matrix

Corpus = {“I like deep learning”
“I like NLP”
“I enjoy flying”}

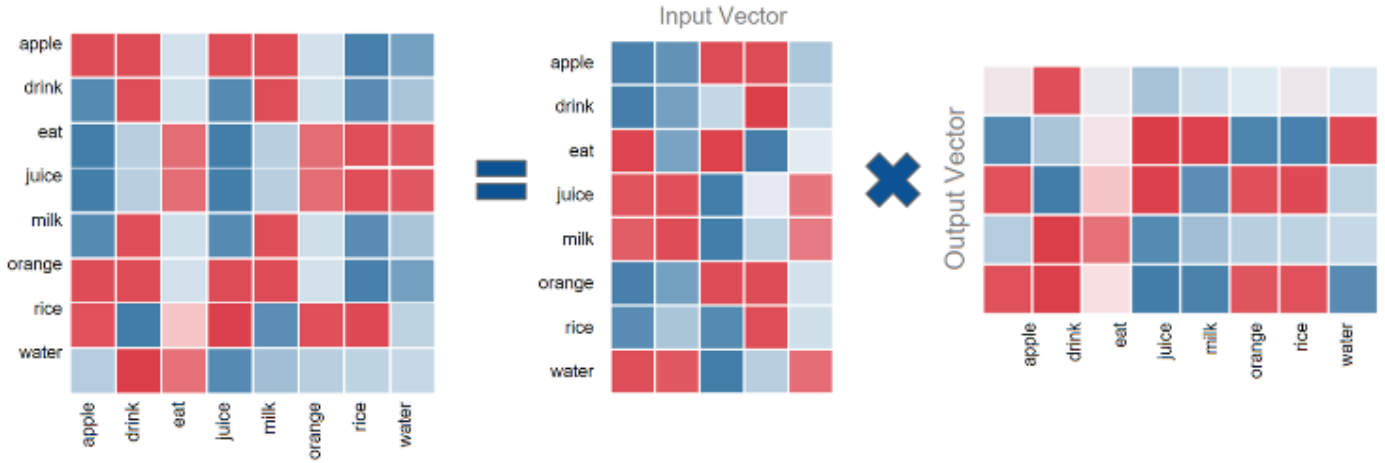
Context = previous word and next word

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

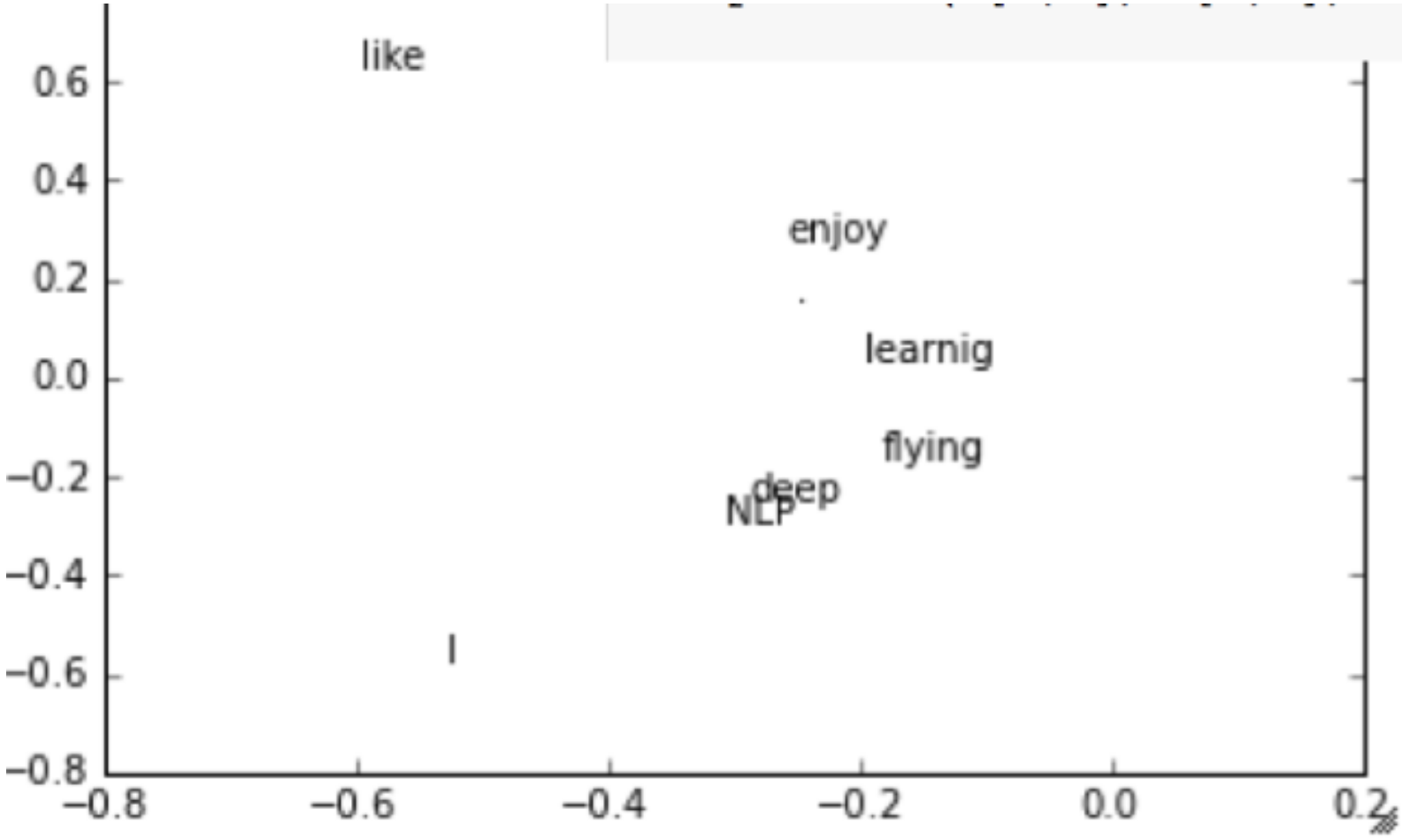
Dimension Reduction using Singular Value Decomposition



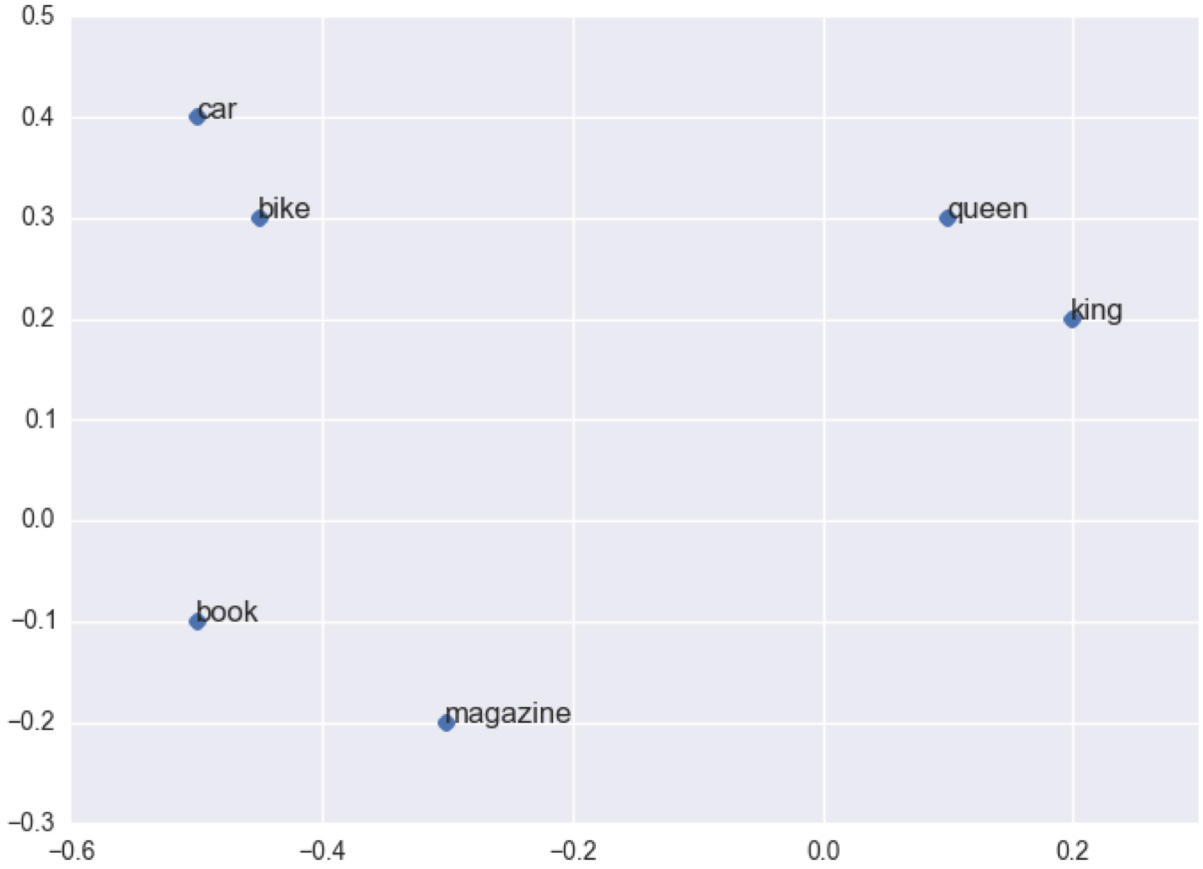
Singular Value Decomposition



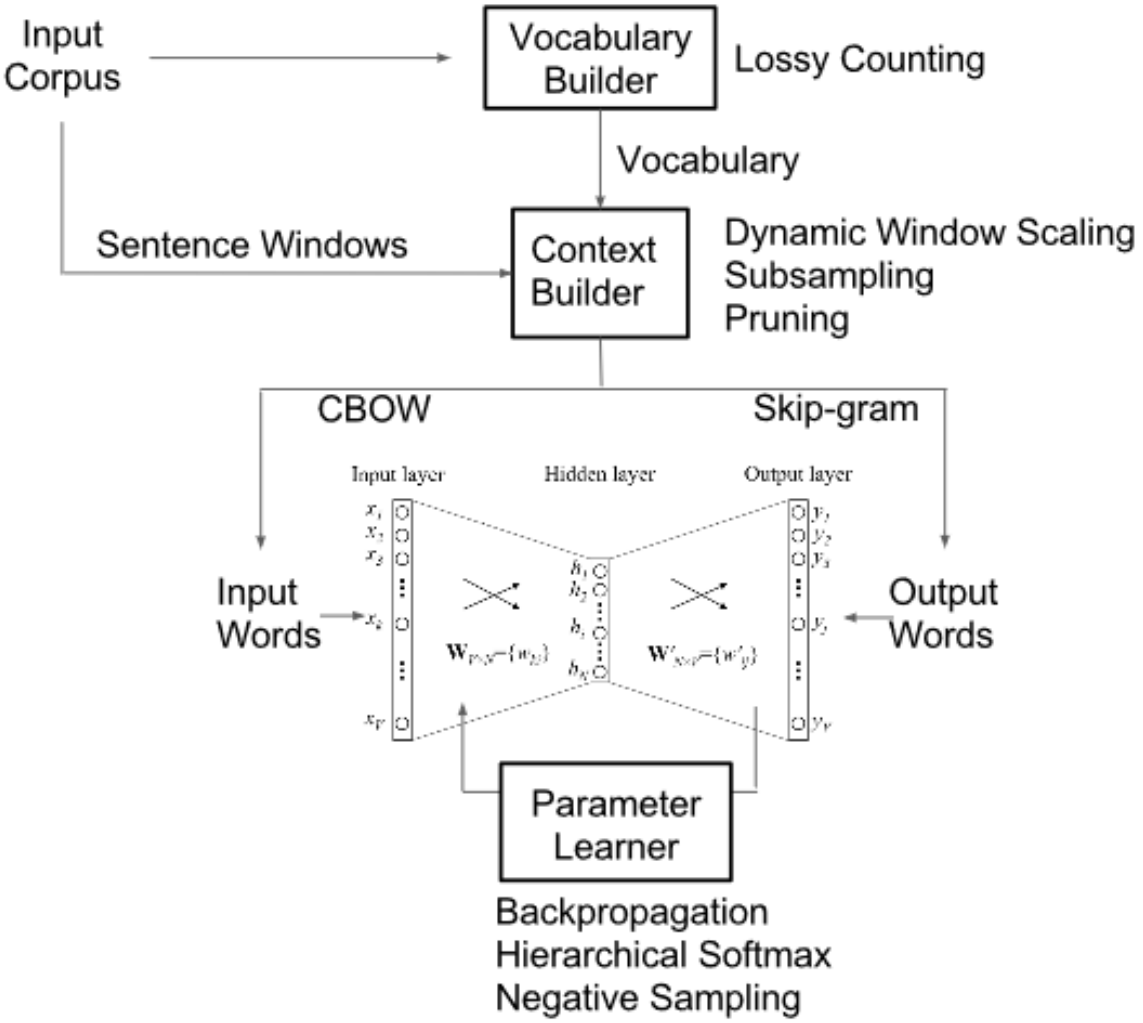
The problem with this method, is that we may end up with matrices having billions of rows and columns, which makes SVD computationally restrictive.


like
enjoy
.
learnig
flying
deep
NLP
I
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2

word2vec


0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.6
−0.4
−0.2
0.0
0.2
car
bike
queen
king
book
magazine

Architecture



Context windows

- Context can be anything – a surrounding n-gram, a randomly sampled set of words from a fixed size window around the word

For example, assume context is defined as the word following a word.

i.e. $\boldsymbol{context(w_i) = w_{i+1}}$

Corpus : I ate the cat

Training Set : I|ate, ate|the , the|cat, cat|.

Training Data

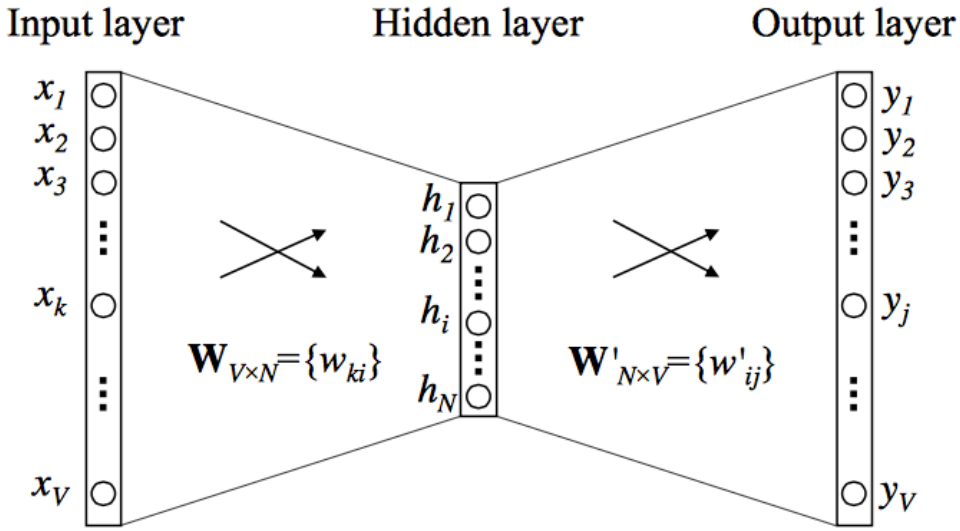
1. eat|apple
2. eat|orange
3. eat|rice
4. drink|juice
5. drink|milk
6. drink|water
7. orange|juice
8. apple|juice
9. rice|milk
10. milk|drink
11. water|drink
12. juice|drink

Concept :

1. Milk and Juice are drinks
2. Apples, Oranges and Rice can be eaten
3. Apples and Orange are also juices
4. Rice milk is a actually a type of milk!

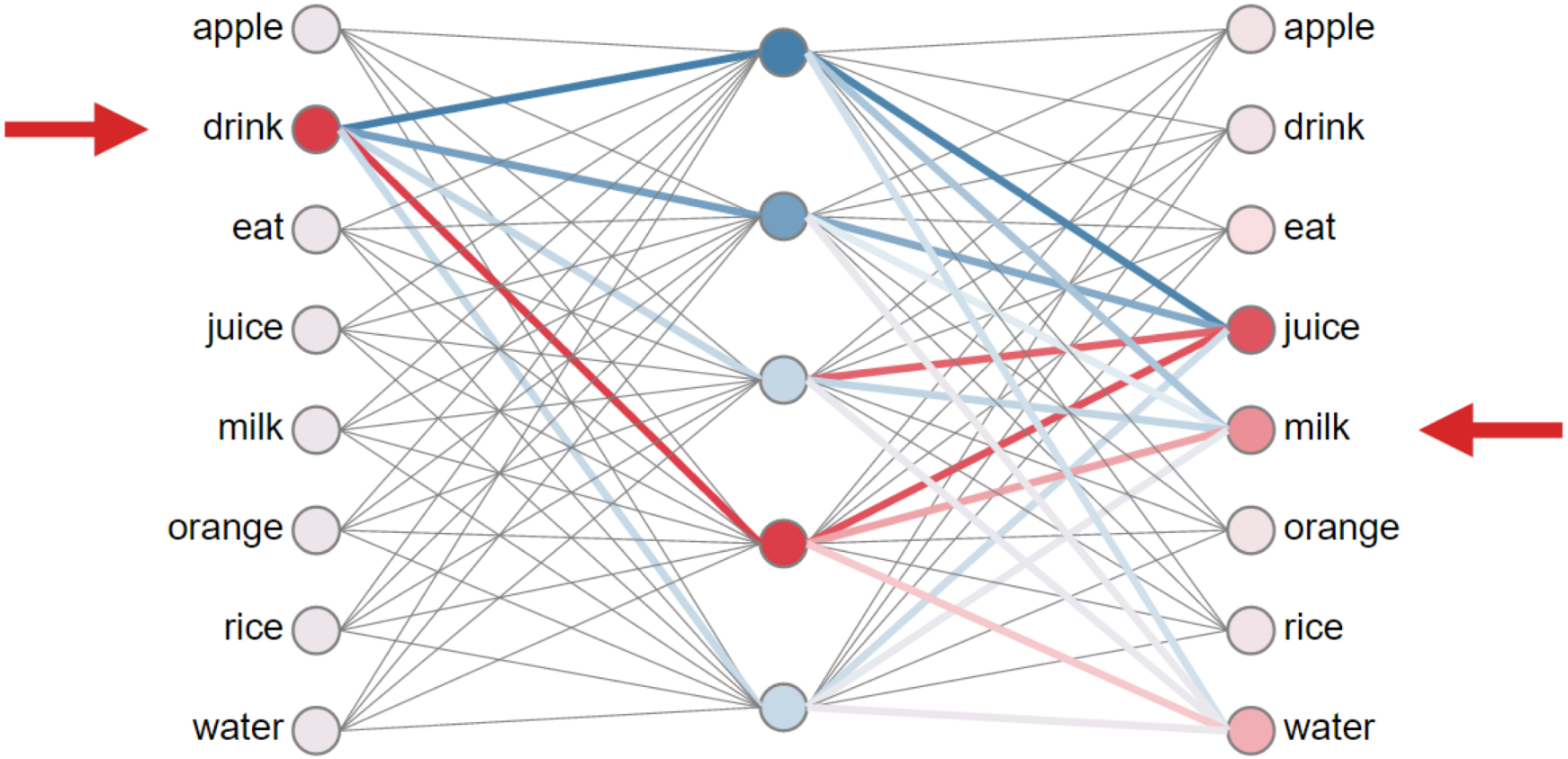
Intuitive Idea

- Some things are better explained using a blackboard and chalk! (*The content in this slide is just a formality!)*



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^{\top}v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top}v_{w_I}\right)}$$


apple
drink
eat
juice
milk
orange
rice
water
apple
drink
eat
juice
milk
orange
rice
water

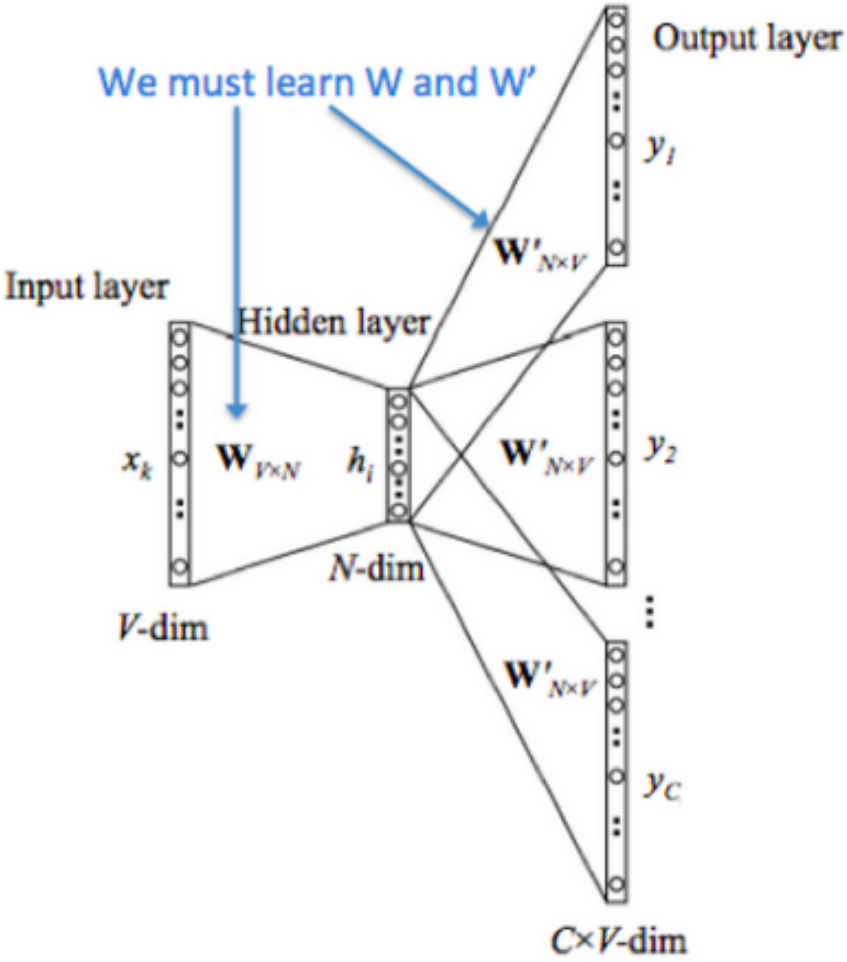
Some other buzzwords and trivia

- Known as neural embedding
- Often optimized using two methods
  1. Hierarchical Softmax
  2. Negative Sampling

- CBOW(continuous bag-of-words) and Skip-gram based training
- Down Sampling of Frequent words
- Phrasal and paragraph vectors

Take Skip-gram model as an example

$$\arg\max_{\theta} \prod_{w \in Text} \left[\prod_{c \in C(w)} p(c|w;\theta) \right]$$



Take Skip-gram model as an example

- Corpus:*美國 總統 川普 參加 大甲 媽祖 遶境*
- *中心詞 w = [美國, 總統, 川普 , 參加, 大甲, 媽祖, 遶境]*

window size 為 1

C(美國) = [__, 總統] ,
C(總統) = [美國, 川普],
C(川普) = [總統, 參加],
C(參加) = [川普, 大甲],
C(大甲) = [參加, 媽祖],
C(媽祖) = [大甲, 遶境],
C(遶境) = [媽祖, __]

Update θ

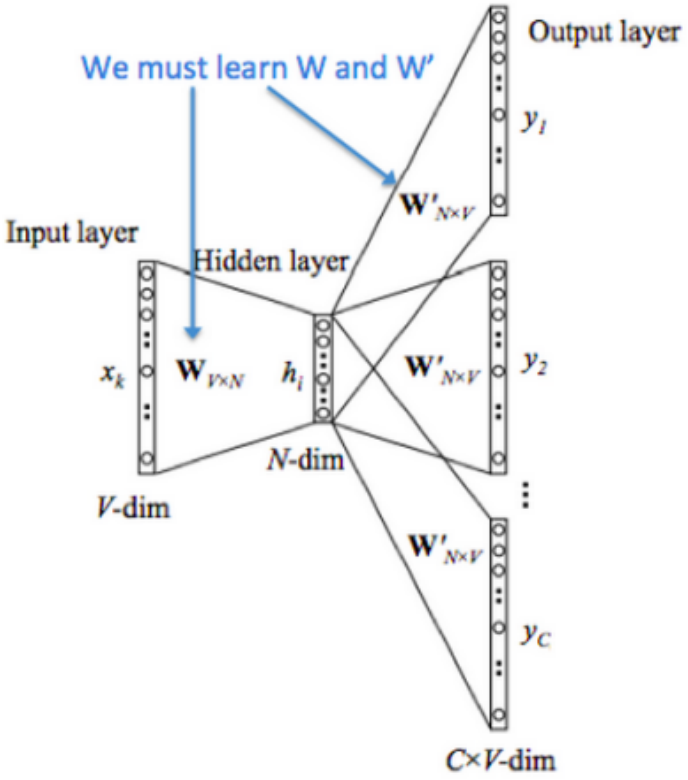
- argmax (p(總統|美國; θ)p(美國 |總統; θ) p(川普 |總統; θ)p(總統|川普; θ) p(參加|川普; θ) p(川普|參加; θ) p(大甲|參加; θ) p(參加|大甲; θ) p(媽祖|大甲; θ) p(大甲|媽祖; θ)p(遶境|媽祖; θ) p(媽祖|遶境; θ))

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta)$$

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}$$

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}$$

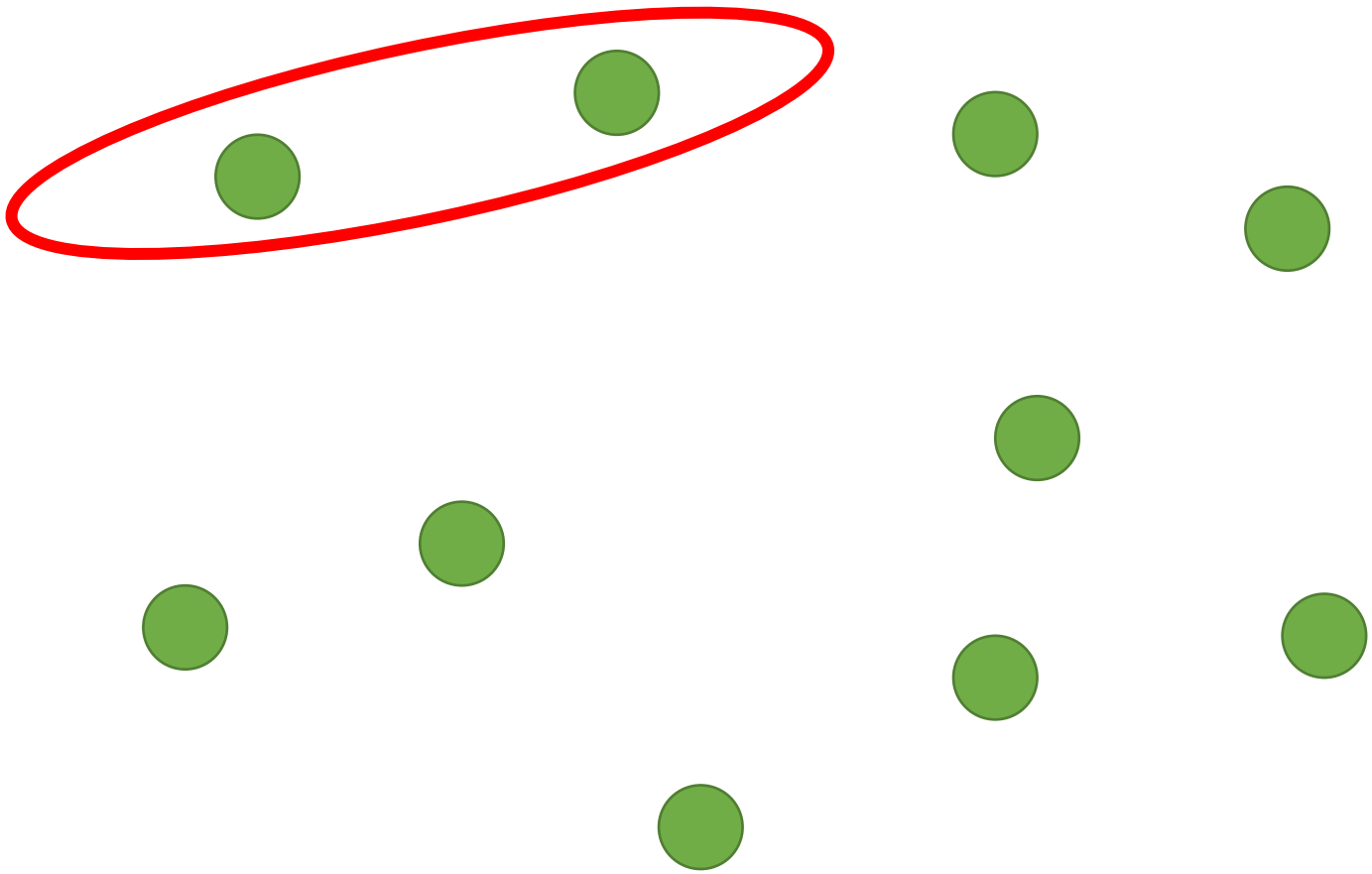
$$\arg\max_{\theta} \sum_{(w,c) \in D} \log p(c|w) = \sum_{(w,c) \in D} (\log e^{v_c \cdot v_w} - \log \sum_{c'} e^{v_{c'} \cdot v_w})$$



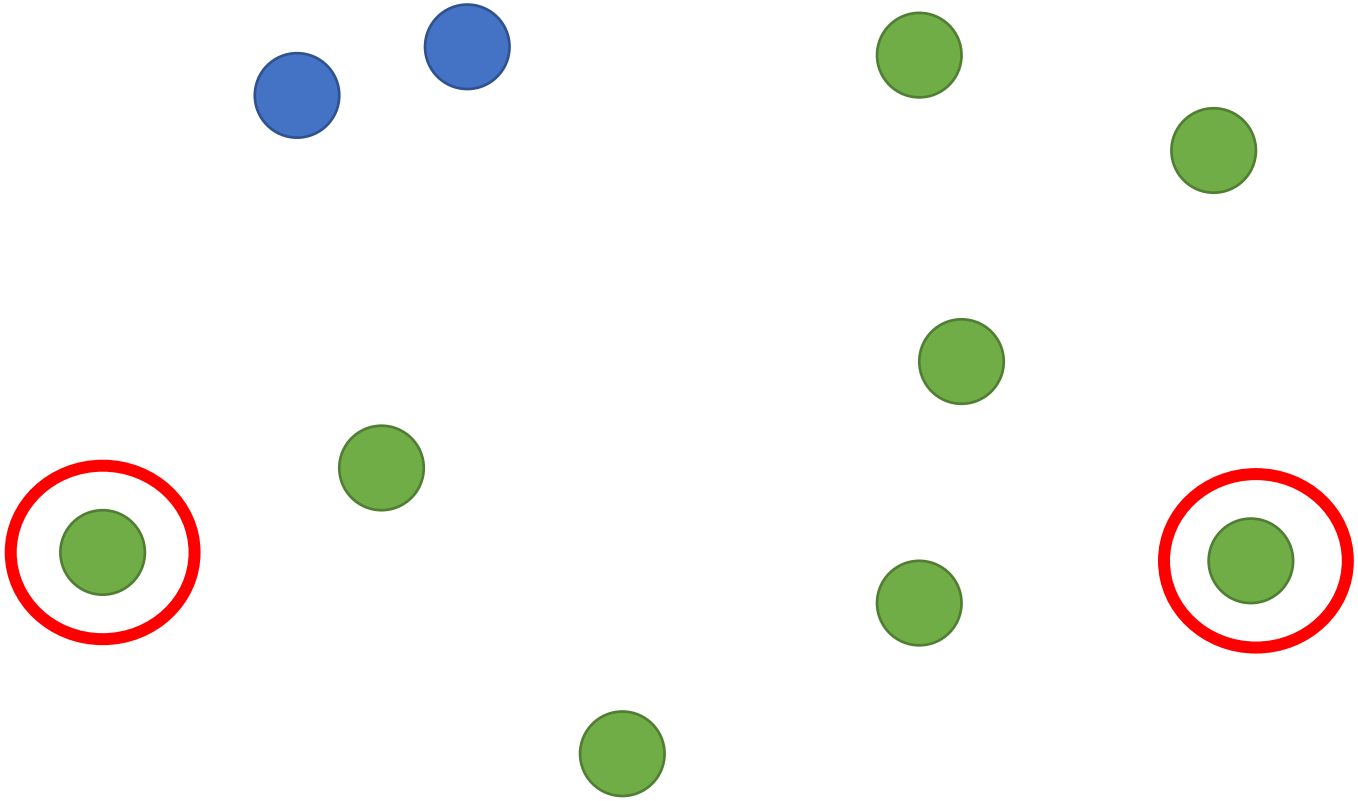
Why we use W and W'

- *vec(Trump) =[-3, 2, 1, -2, 4]*
- *vec(Trump) · vec(Trump) = 9+4+1+4+16 = 34*

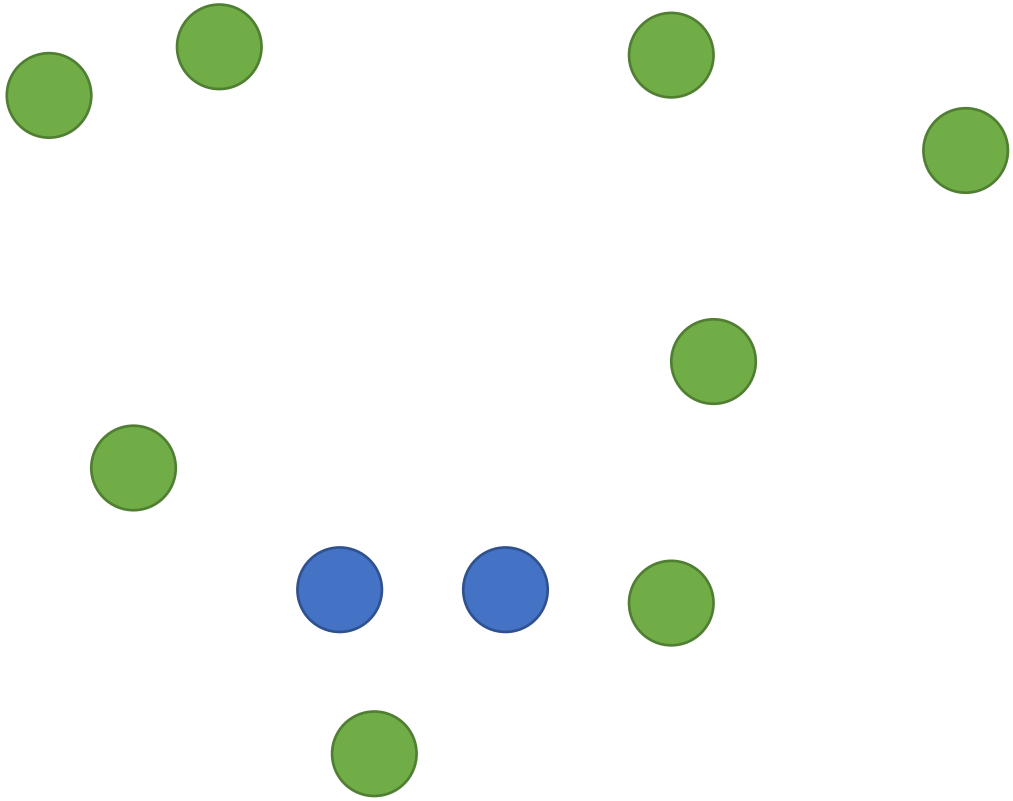
Problems?



Problems?

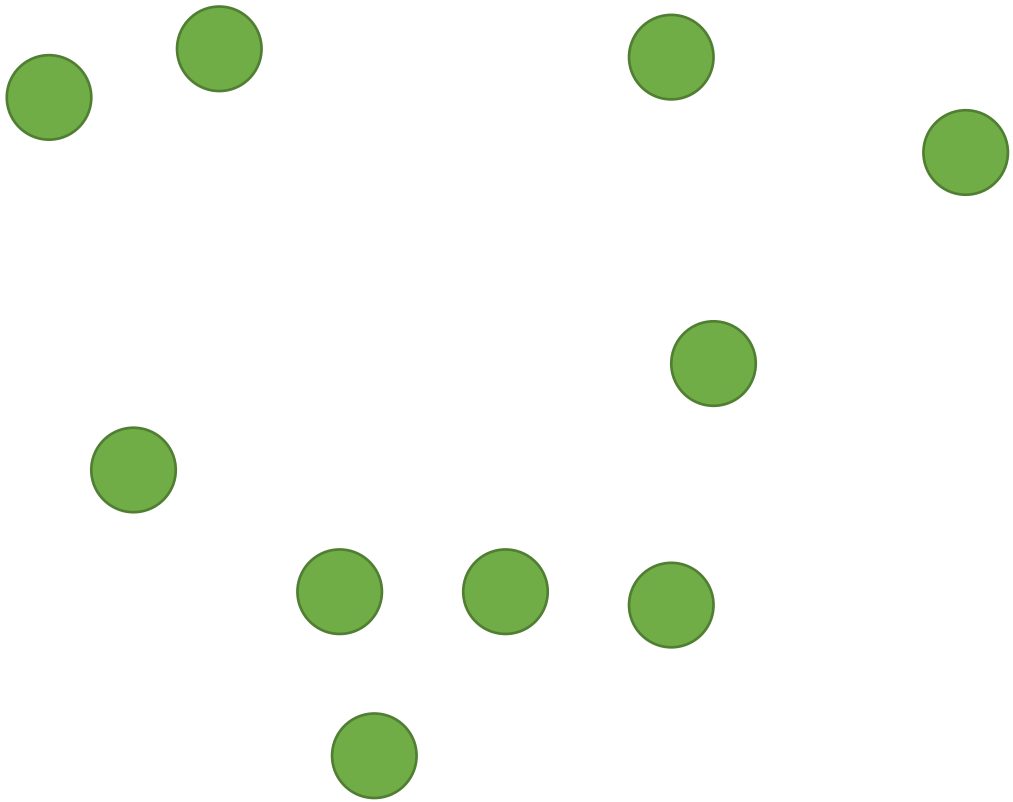


Problems?




嗯。
這個遊戲…
有必勝的方法。

計畫通？



Negative Sampling

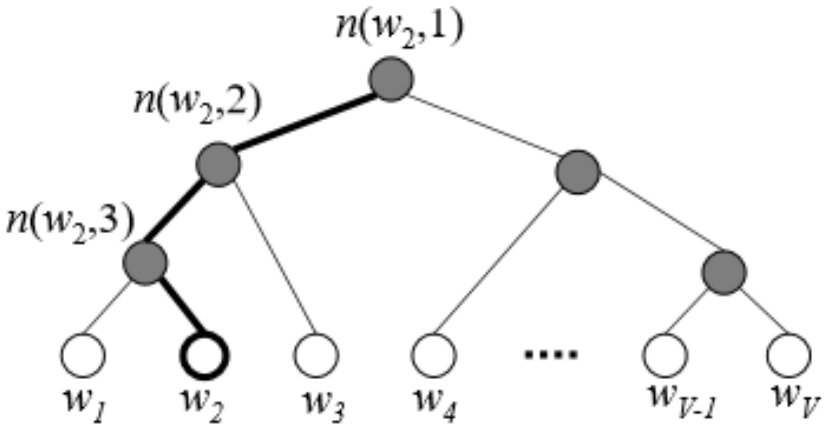
- Suppose we know that some words won't be put together.
  - [Trump, NBA]
  - [Trump, Vitamin C]
  - [Trump, Washing Machine]

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log \frac{1}{1+e^{-v_c \cdot v_w}} + \sum_{(w,c)\in D'} \log\left(\frac{1}{1+e^{v_c \cdot v_w}}\right)$$

$$= \arg\max_{\theta} \sum_{(w,c)\in D} \log \sigma(v_c \cdot v_w) + \sum_{(w,c)\in D'} \log \sigma(-v_c \cdot v_w)$$

Hierarchical Softmax (CBOW)

- Key Idea: Modify the output layer so we can accelerate the training process.
  - Huffman coding
  - O(n) -> O(logn)



CBOW

- Corpus:*美國 總統 川普 參加 大甲 媽祖 遶境*
- *Window size: 2*
- *Input: 美國, 總統, 參加, 大甲*
- *Output:川普*
- *Suppose 川普 is coded with 110*
- *Left:* $\sigma(X_\omega^T\theta) = \frac{1}{1+e^{-X_\omega^T\theta}}$ *Right:* $1-\sigma(X_\omega^T\theta)$

$$p(\omega|Context(\omega)) = \prod_{j=2}^{l^\omega} p(d_j^\omega|X_\omega, \theta_{j-1}^\omega)$$

$$p(d_j^\omega|X_\omega, \theta_{j-1}^\omega) = \begin{cases} \sigma(X_\omega^T\theta_{j-1}^\omega), d_j^\omega = 0; \\ 1-\sigma(X_\omega^T\theta_{j-1}^\omega), d_j^\omega = 1; \end{cases}$$

Using word2vec in your research . . .

- Easiest way to use it is via the Gensim libarary for Python (tends to be slowish, even though it tries to use C optimizations like Cython, NumPy)

https://radimrehurek.com/gensim/models/word2vec.html

- Original word2vec C code by Google

https://code.google.com/archive/p/word2vec/

Word Embedding Visualization

http://ronxin.github.io/wevi/

Global Vectors for Word Representation (GloVe)

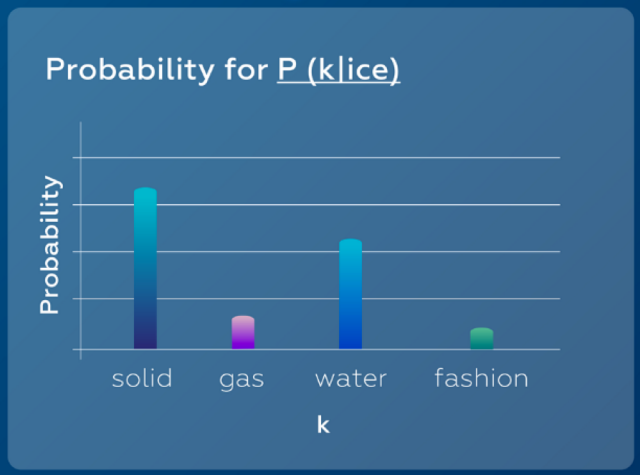
Main Idea

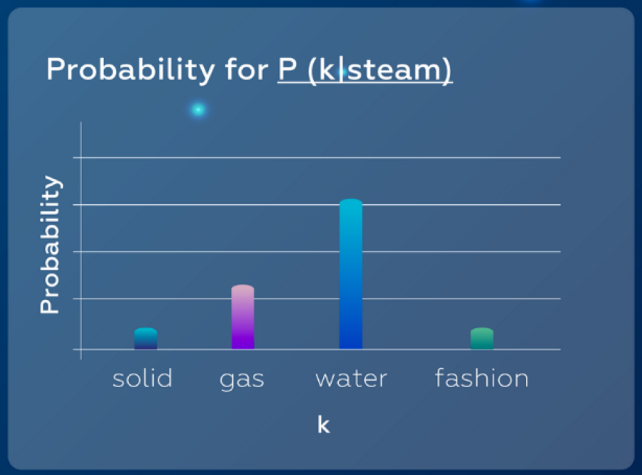
- Uses ratios of co-occurrence probabilities, rather than the co-occurrence probabilities themselves

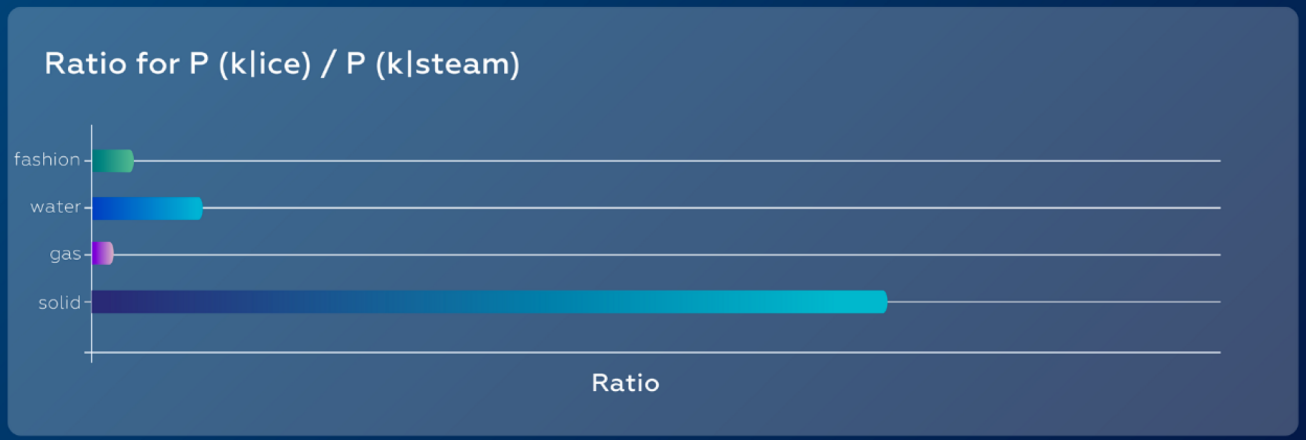
Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

A simple example based on the words

ICE AND STEAM







Least Squares Problem

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 ,$$

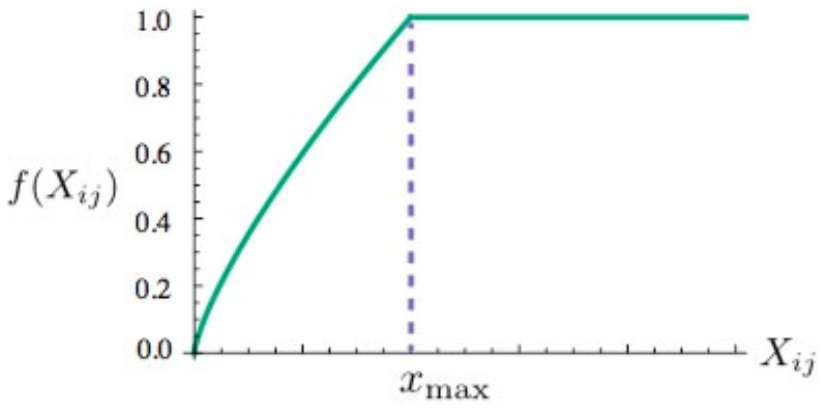


Figure 1: Weighting function f with $\alpha = 3/4$.

Weakness of Word Embedding

- Very vulnerable, and not a robust concept
- Can take a long time to train
- Non-uniform results
- Hard to understand and visualize

Embeddings/language models

- Two categories
  - Fixed: meaning that they generate a single representation for each word in the vocabulary
  - Contextualized: meaning that a representation is generated based on both the word and its surrounding context, so that a single word can have multiple representations, each one depending on how it is used.

New things from ACL 2020

Top Conferences of NLP/NLU

- ACL
- EMNLP
- NAACL
- ----------
- SIGIR
- NeurIPS
- ICML
- ICLR
- AAAI
- IJCAI

ACL'20 Long Papers about Word Embeddings

- A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages
- Adaptive Compression of Word Embeddings
- BERTRAM: Improved Word Embeddings Have Big Impact on Contextualized Model Performance
- Double-Hard Debias: Tailoring Word Embeddings for Gender Bias Mitigation
- Revisiting the Context Window for Cross-lingual Word Embeddings
- When do Word Embeddings Accurately Reflect Surveys on our Beliefs About People?

Adaptive Compression of Word Embeddings

ACL'20

Motivations

- Deep neural networks have greatly improved the performance in various tasks, such as image classification, text classification, and machine translation.
- The large memory footprint of word embeddings makes it challenging to deploy NLP models to memory-constrained devices (e.g., self-driving cars, mobile devices).
- To alleviate this computation/memory limitation, several works have proposed methods that compress the neural models while minimizing loss of accuracy as much as possible.

However,

- Deploying models for natural language processing (NLP) tasks is challenging.
  - Unlike other domains, NLP models have an embedding layer which maps words and phrases to real-valued vectors. The problem is that these embeddings usually take more parameters than the remaining networks.
  - For example, in OpenNMT, the word embedding parameters account for **80%** of the total parameters.

Solutions

- To compress word embeddings, several works proposed code-book based approaches, which represent each word as few discrete and shared codes.
- For example, the word dog and dogs could be represented as (3, 5, 2, 1) and (3, 5, 2, 7), respectively.
- However, these methods assign the same length of codes to each word without considering the significance of downstream tasks.

Solutions (Cont.)

- The goal is to compress word embeddings by adaptively assigning different lengths of codes to each word in an end-to-end manner.
- First, each word in pre-trained word embeddings learns to select its code length.
- After selecting its code length, each word learns discrete codes through a binary-constraint encoder and decoder network by Gumbel softmax tricks.

Strategy

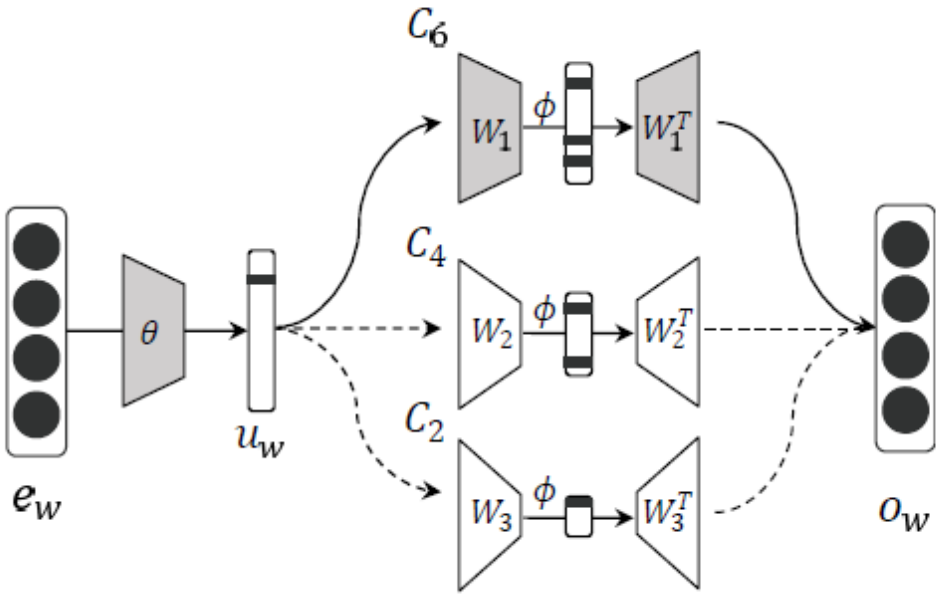


Figure 1: Main strategy of our compression model (AdaComp). Solid line indicates the selected codebook.

$$L_{pre} = \sum_{w \in V} \|o_w - e_w\|_2^2$$

Model	SST-5		CoNLL-2000		SNLI		PTB	
	accuracy	ratio	F1	ratio	accuracy	ratio	ppl	ratio
GloVe	42.1	x1	93.1	x1	79	x1	100.3	x1
QWE (4-bit) (Ling et al., 2016)	41.8	x8	93.1	x8	77.9	x8	113.8	x8
QWE (8-bit) (Ling et al., 2016)	41.9	x4	93.3	x4	78.6	x4	109.1	x4
Pruning 90% (Han et al., 2015)	35.4	x10	90.4	x10	78	x10	113.2	x10
Pruning 80% (Han et al., 2015)	41.6	x5	91.7	x5	78.2	x5	124.7	x5
NC (16×16) (Shu and Nakayama, 2018)	37.2	x46	91.8	x50	77.8	x71	119.2	x30
NC (32×16) (Shu and Nakayama, 2018)	40.9	x23	92.4	x25	78.5	x35	112.4	x15
Bin (64) (Tissier et al., 2019)	36.8	x95	91.5	x100	77.3	x116	116	x74
Bin(128) (Tissier et al., 2019)	39.1	x48	92.7	x49	77.6	x59	110.1	x37
AdaComp (32)	42.0	x171	92.1	x173	77.6	x232	110.8	x105
AdaComp (64)	43.2	x84	93	x89	78.4	x119	106	x52
AdaComp (128)	42.9	x45	93.1	x44	78.7	x60	108.9	x26

Double-Hard Debias: Tailoring Word Embeddings for Gender Bias Mitigation

ACL'20

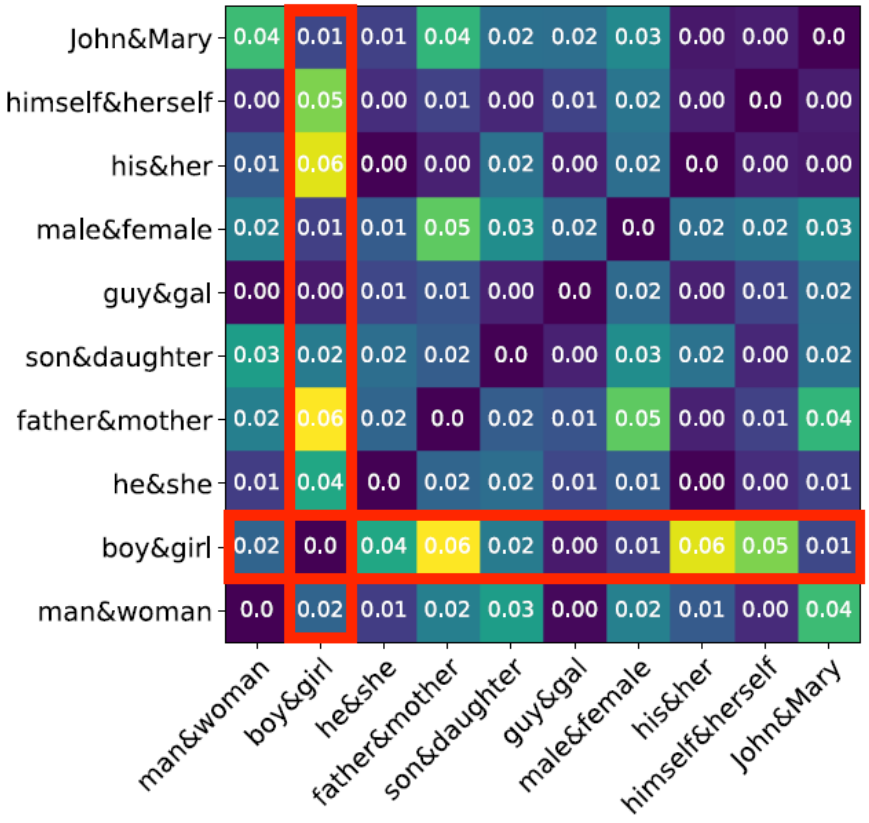
Motivation

- Word embeddings derived from human-generated corpora inherit strong gender bias which can be further amplified by downstream models.
  - In word2vec embeddings trained on the Google News dataset, “programmer” is more closely associated with “man” and “homemaker” is more closely associated with “woman”.
- Some works apply post-processing procedures that project pre-trained word embeddings into a subspace orthogonal to an inferred gender subspace.

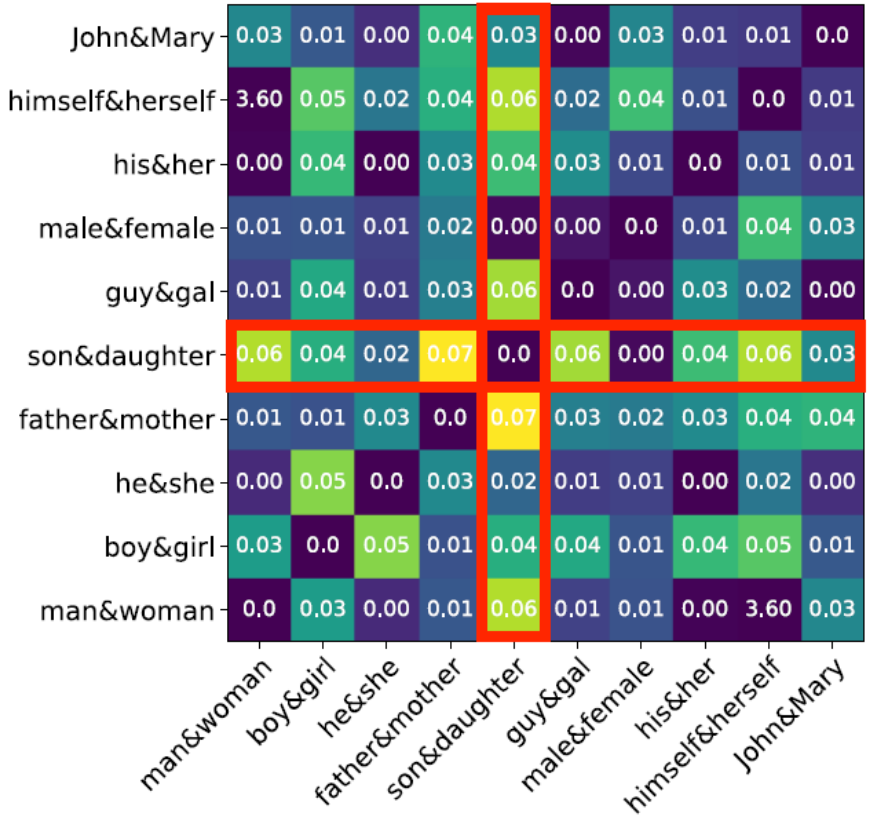
Goals

- The goal is to prevent discrimination in NLP systems!
- To mitigate gender bias, prior work have proposed to remove the gender component from pre-trained word embeddings through postprocessing.
- This work hypothesizes that it is difficult to isolate the gender component of word embeddings in the manner employed by existing post-processing methods.
- Current post-hoc debiasing methods attempt to reduce gender bias in word embeddings by subtracting the component associated with gender from them.

$$\vec{v}_{boy,girl} = \vec{w}_{boy} - \vec{w}_{girl}$$



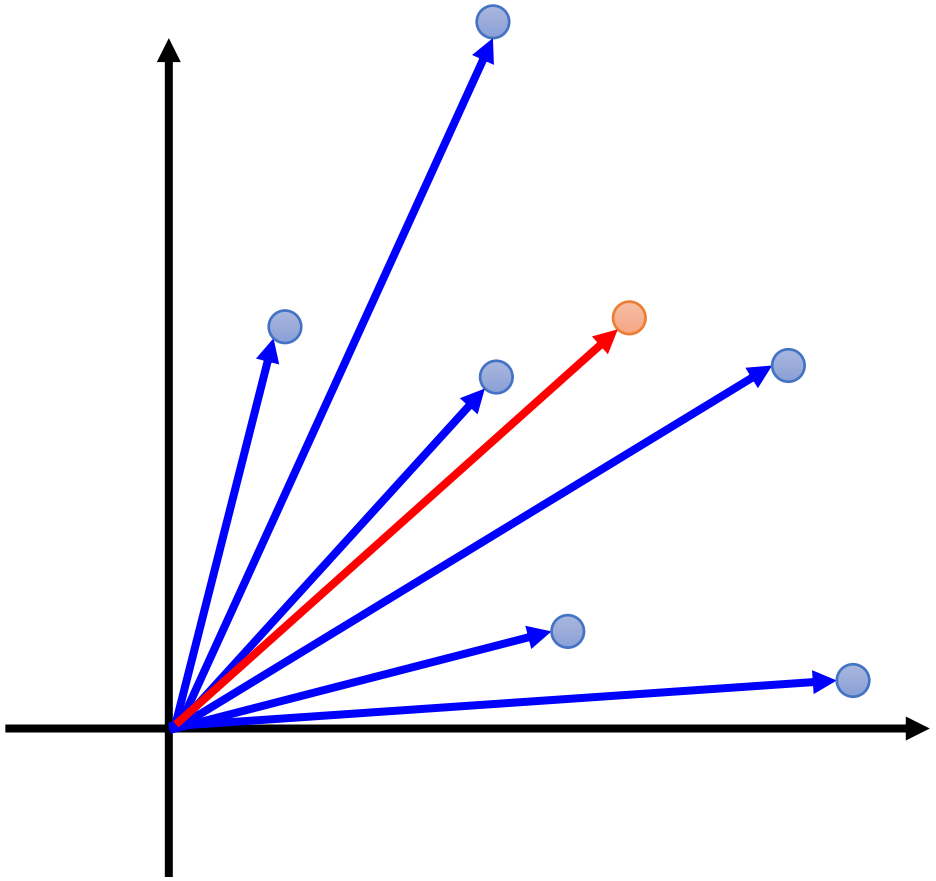
(a) Change the frequency of "boy".



(b) Change the frequency of "daughter".

Application: Information Retrieval (IR)

Information Retrieval (IR)



Vector Space Model

The documents are vectors in the space.

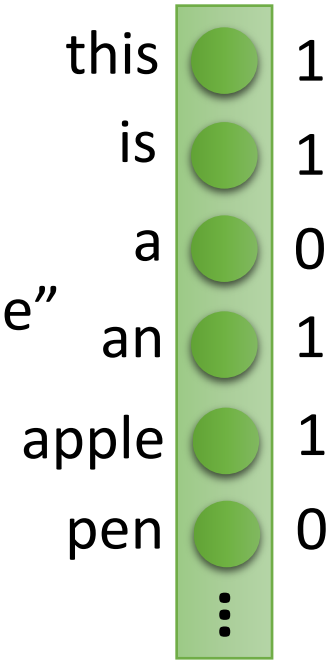
The query is also a vector.

How to use a vector to represent word sequences

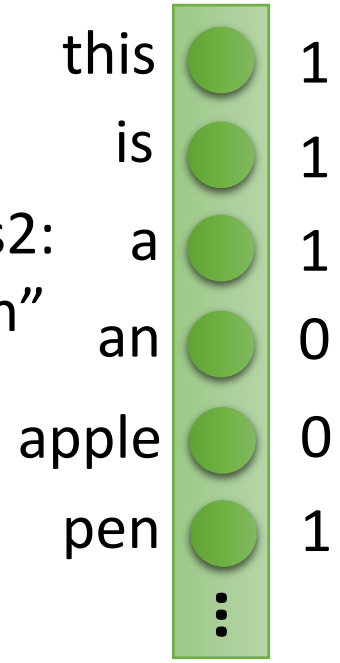
Information Retrieval (IR)

Bag-of-word

word string s1:
"This is an apple"



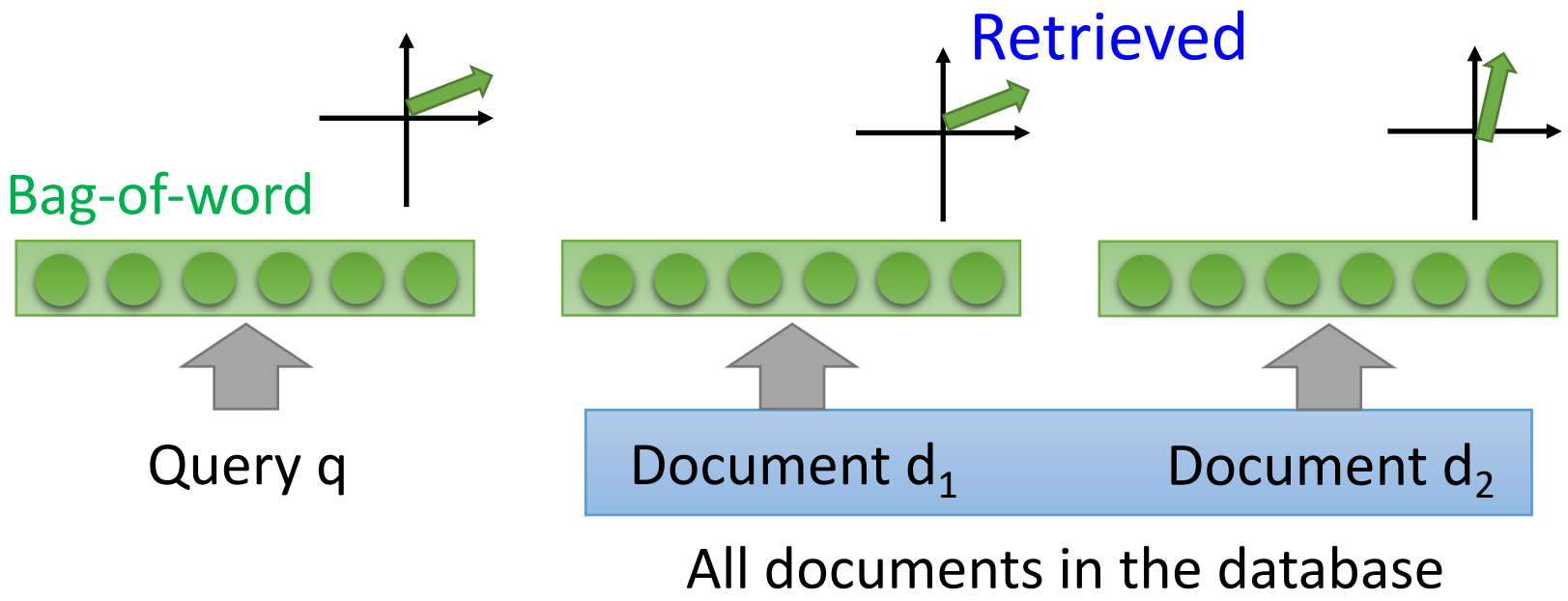
word string s2:
"This is a pen"



Weighted by IDF

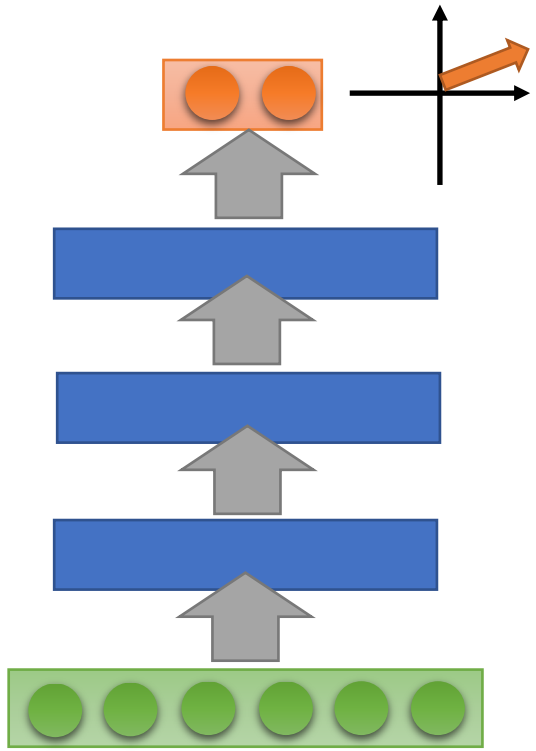
Information Retrieval (IR)

Vector Space Model + Bag-of-word



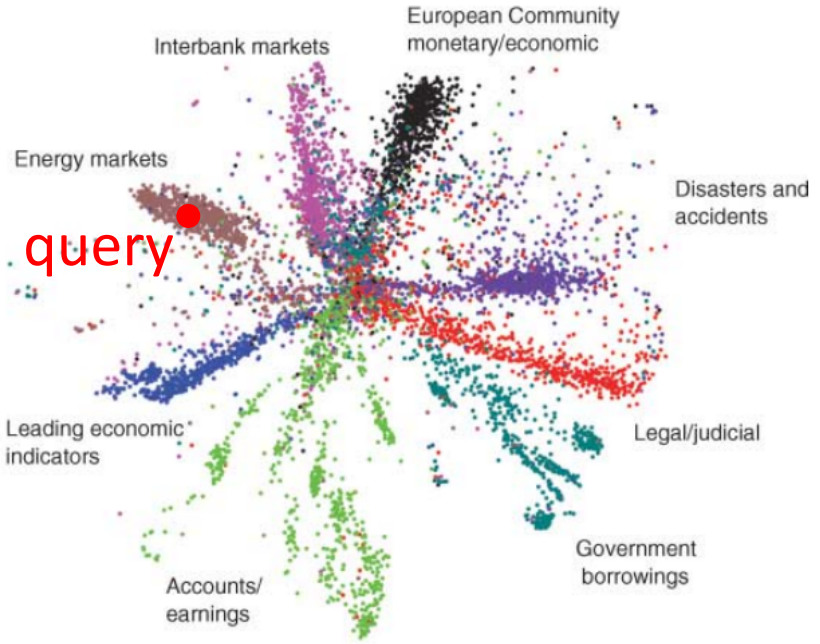
- All the words are treated as discrete tokens.
- Never considered: Different words can have the same meaning, and the same word can have different meanings.

IR - Semantic Embedding



Bag-of-word

word string
(document or query)



Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

How to achieve that? (No target ……)

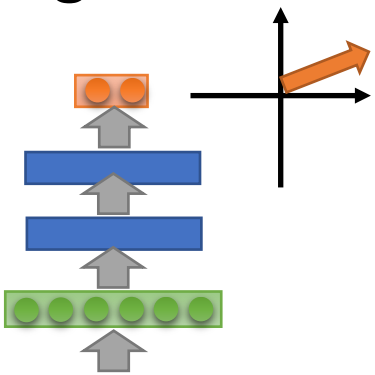
DSSM

Click-through data: $q_1 \rightarrow d_1 : + \quad d_2 : -$

$q_2 \rightarrow d_3 : - \quad d_4 : +$

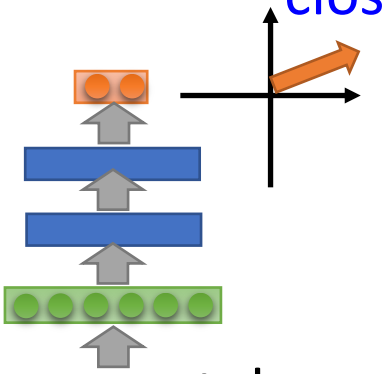
……

Training:



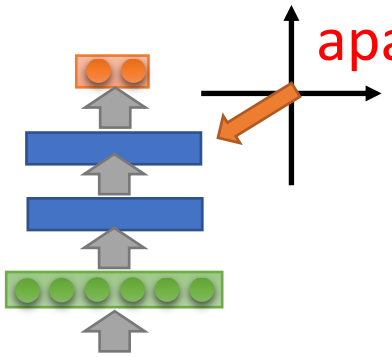
query q_1

close

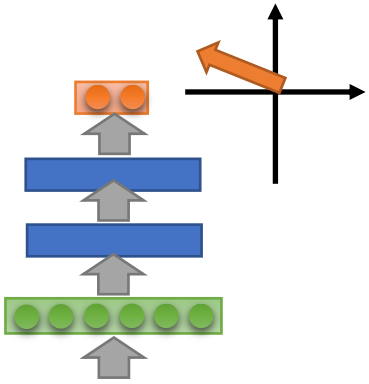


document d_1 +

Far apart

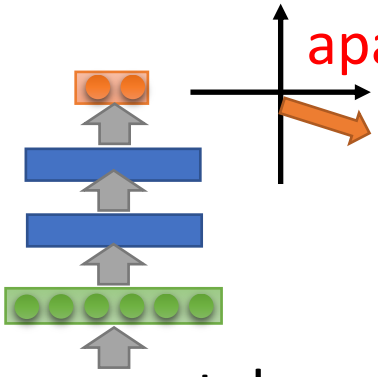


document d_2 -



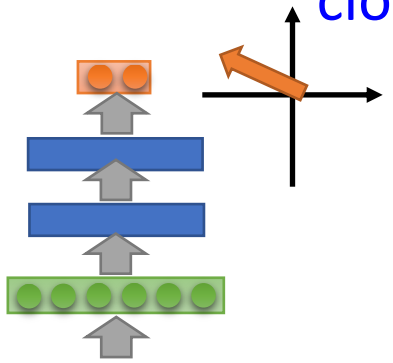
query q_2

Far apart



document d_3 -

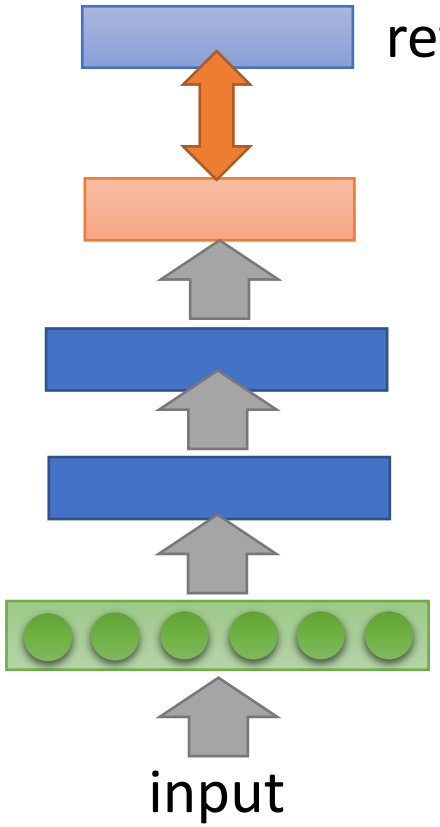
close



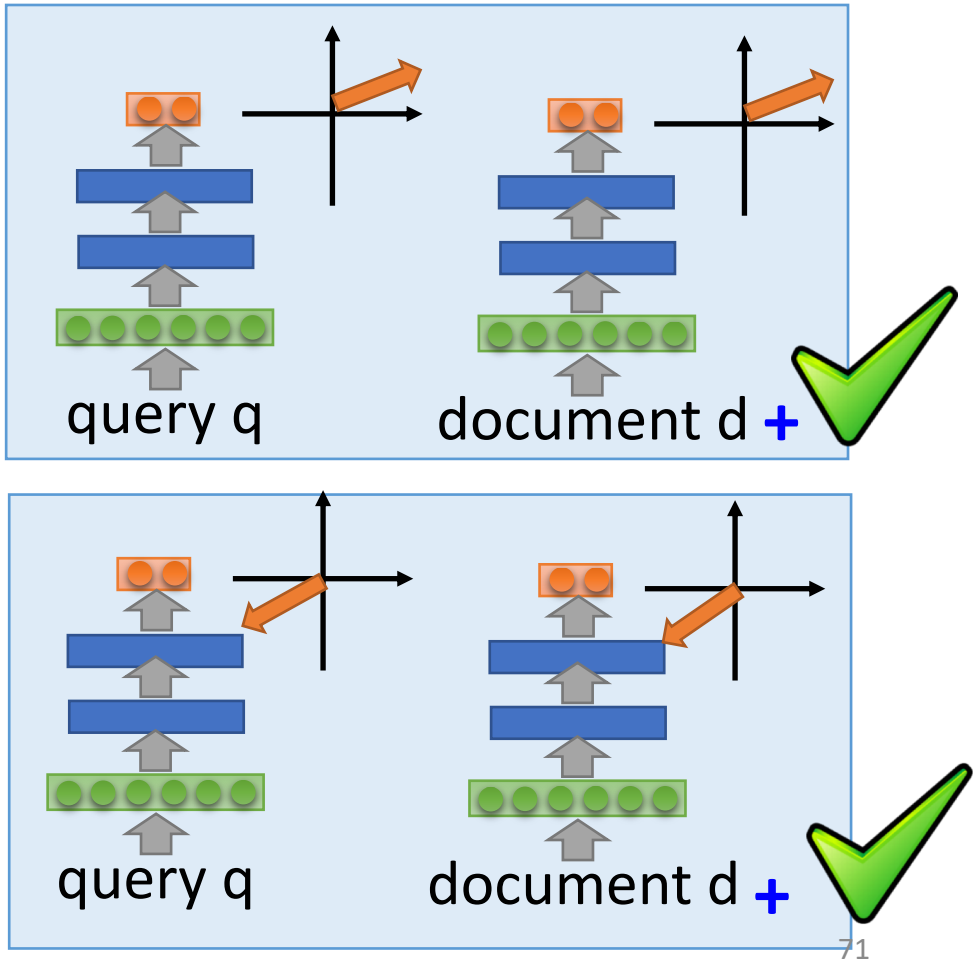
document d_4 +

DSSM vs. Typical DNN

Typical DNN



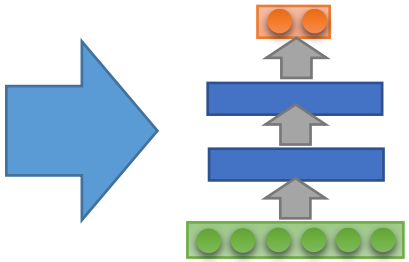
DSSM



Click-through data: $q_1 \rightarrow d_1 : +$ $d_2 : -$

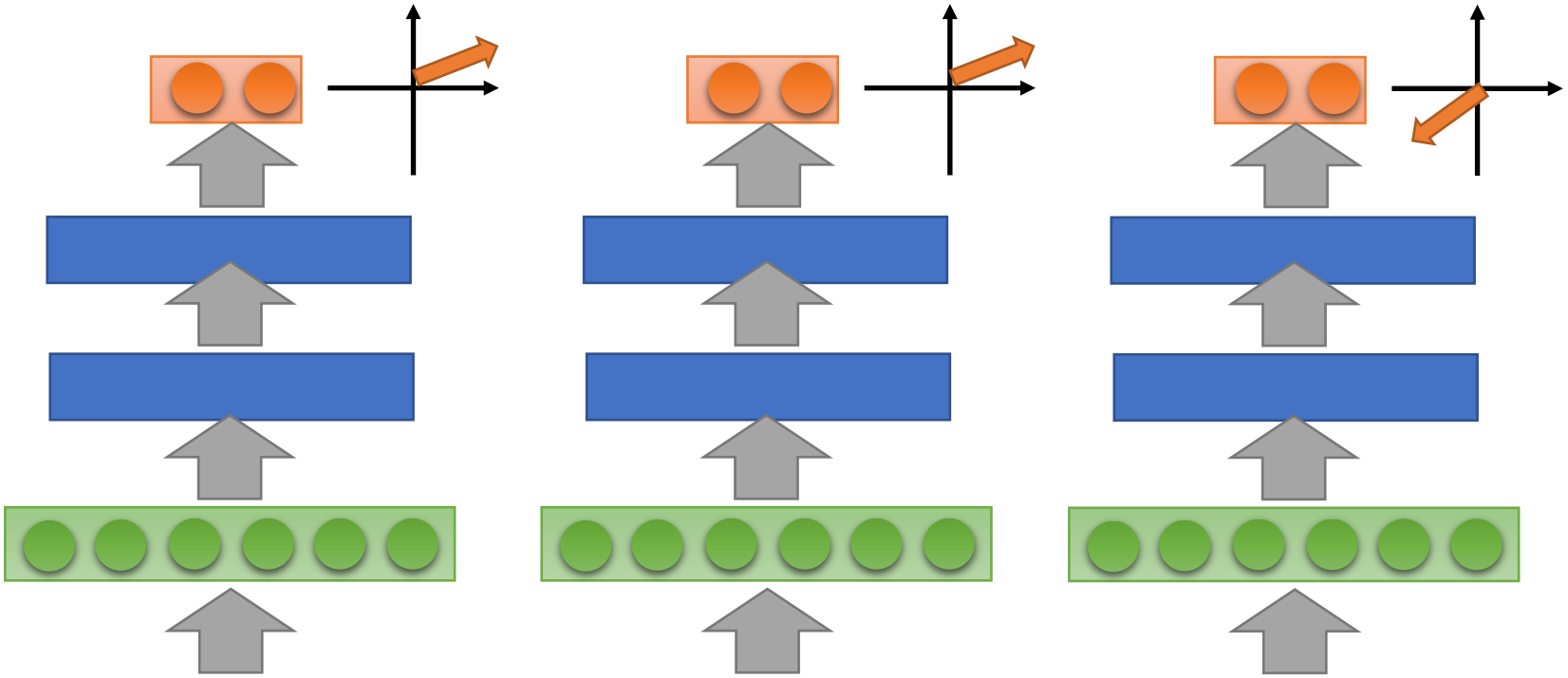
$q_2 \rightarrow d_3 : -$ $d_4 : +$

……



- How to do retrieval?

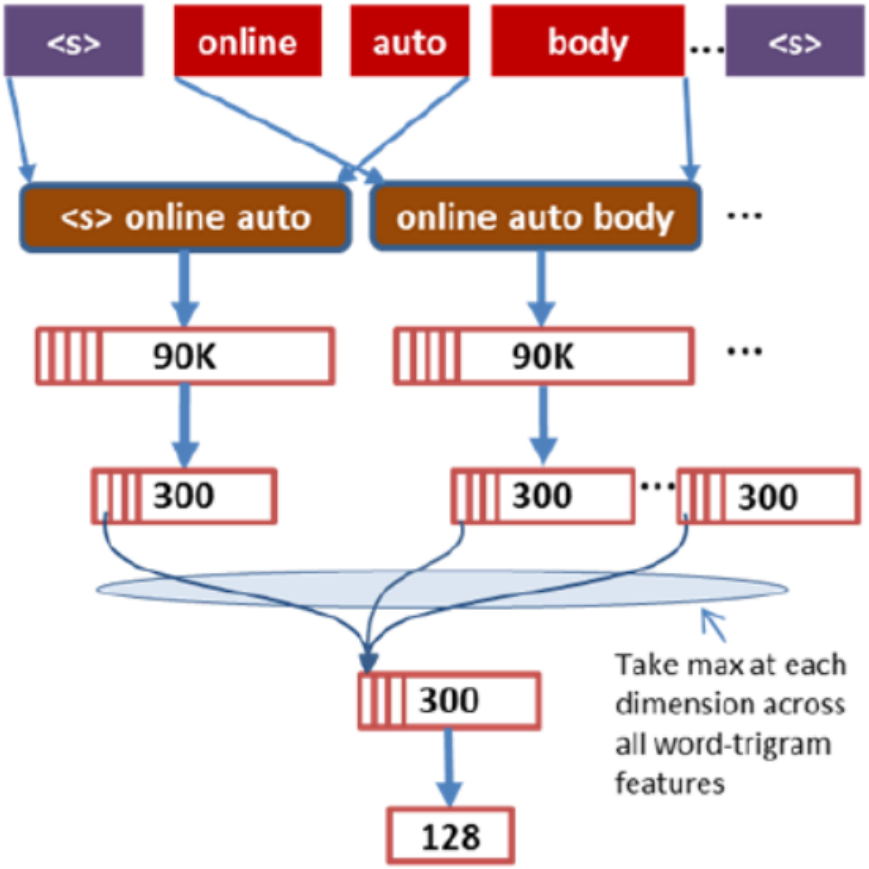
Retrieved



New Query q’ Document d_1 Document d_2

Reference

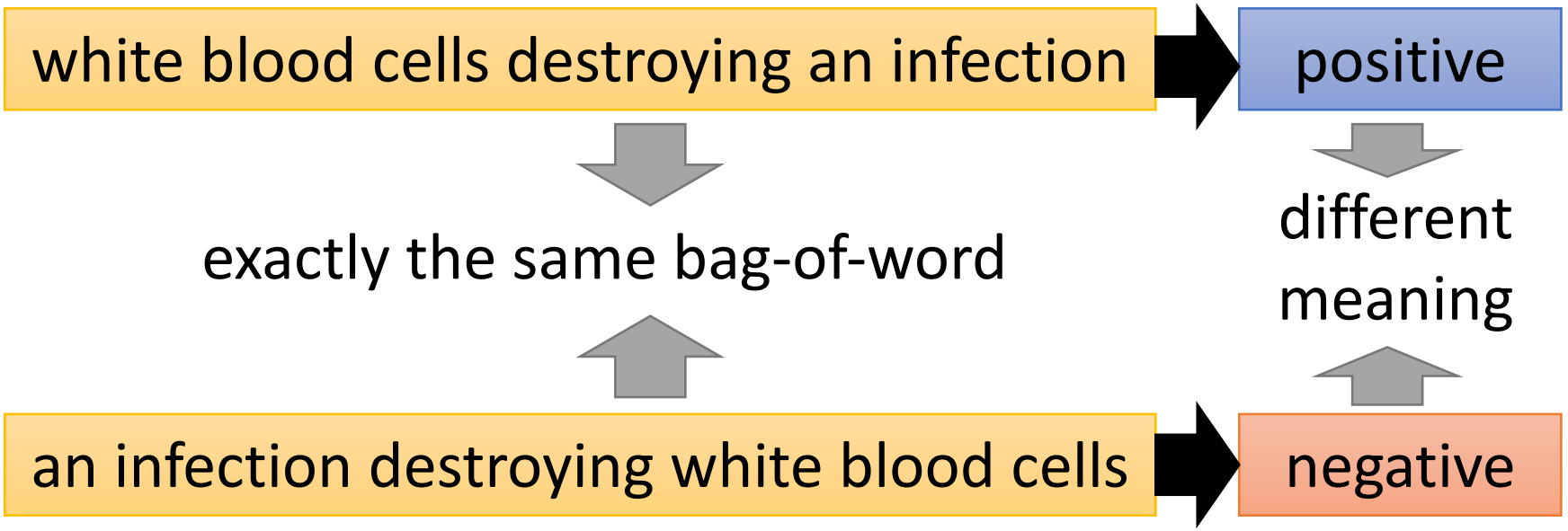
- Huang, Po-Sen, et al. "Learning deep structured semantic models for web search using clickthrough data." ACM, 2013.
- Shen, Yelong, et al. "A latent semantic model with convolutional-pooling structure for information retrieval." ACM, 2014.



Application: Sentiment Analysis, Sentence Relatedness

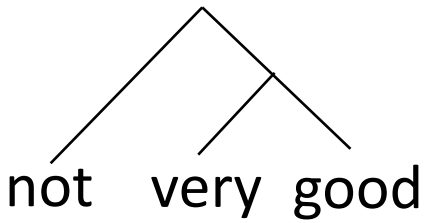
Recursive Deep Model

- To understand the meaning of a word sequence, the order of the words can not be ignored.



Recursive Deep Model

syntactic structure

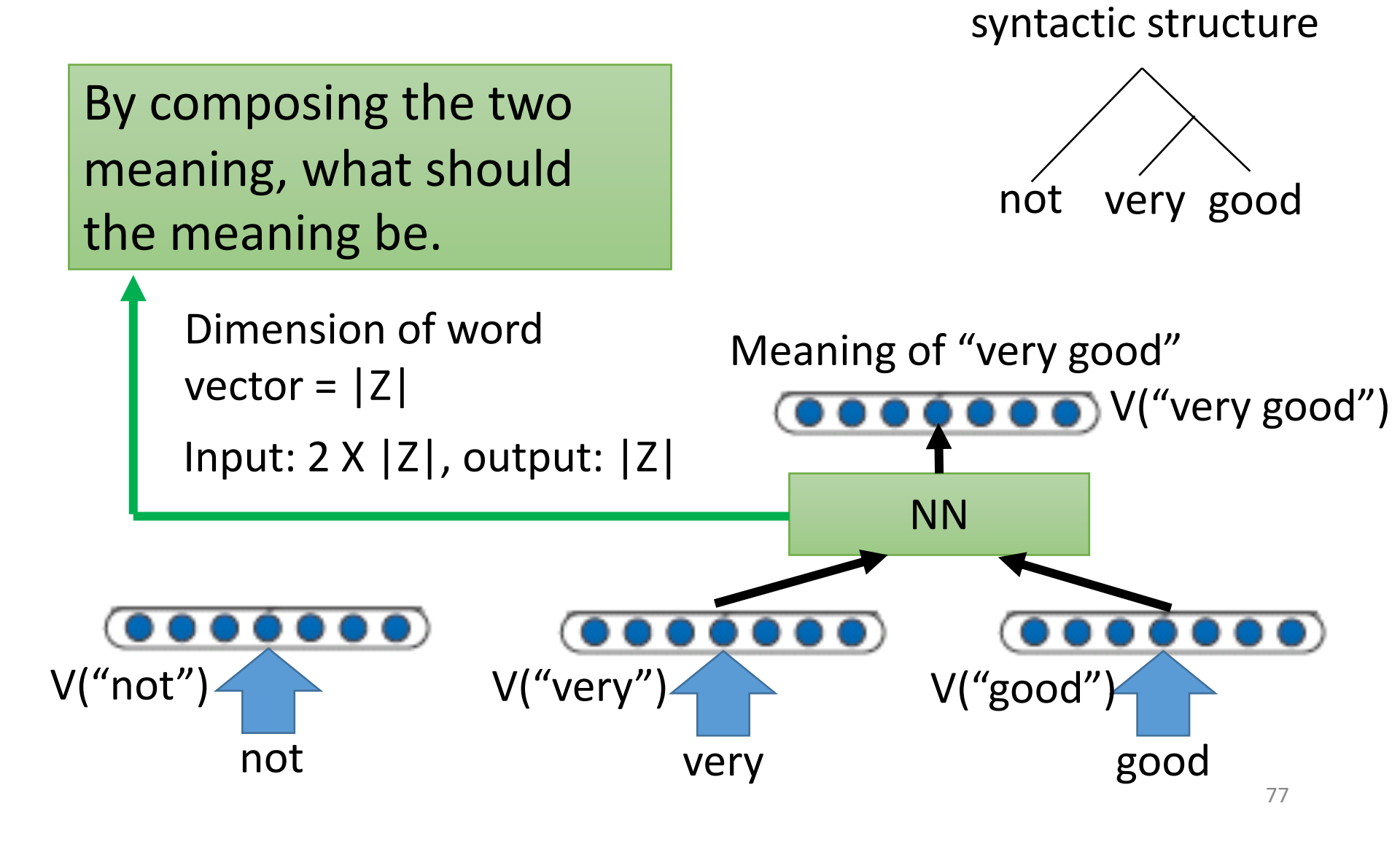


How to do it is out of the scope

word sequence:

not very good

Recursive Deep Model



Recursive Deep Model

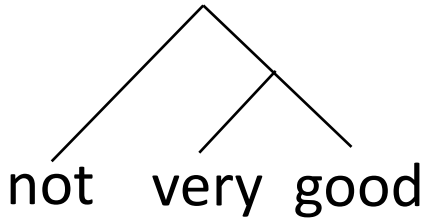
$V(w_A\, w_B) \neq V(w_A) + V(w_B)$

"not": neutral

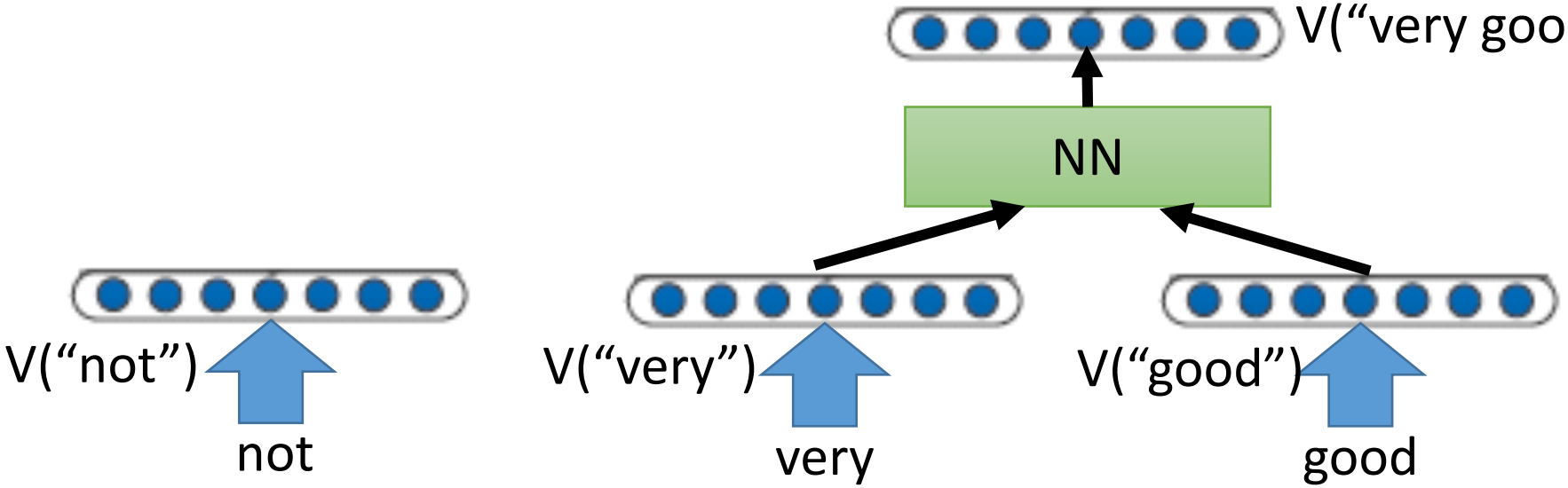
"good": positive

"not good": negative

syntactic structure



Meaning of "very good"



Recursive Deep Model

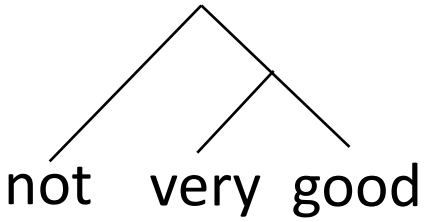
$V(w_A\, w_B) \neq V(w_A) + V(w_B)$

“棒”: positive

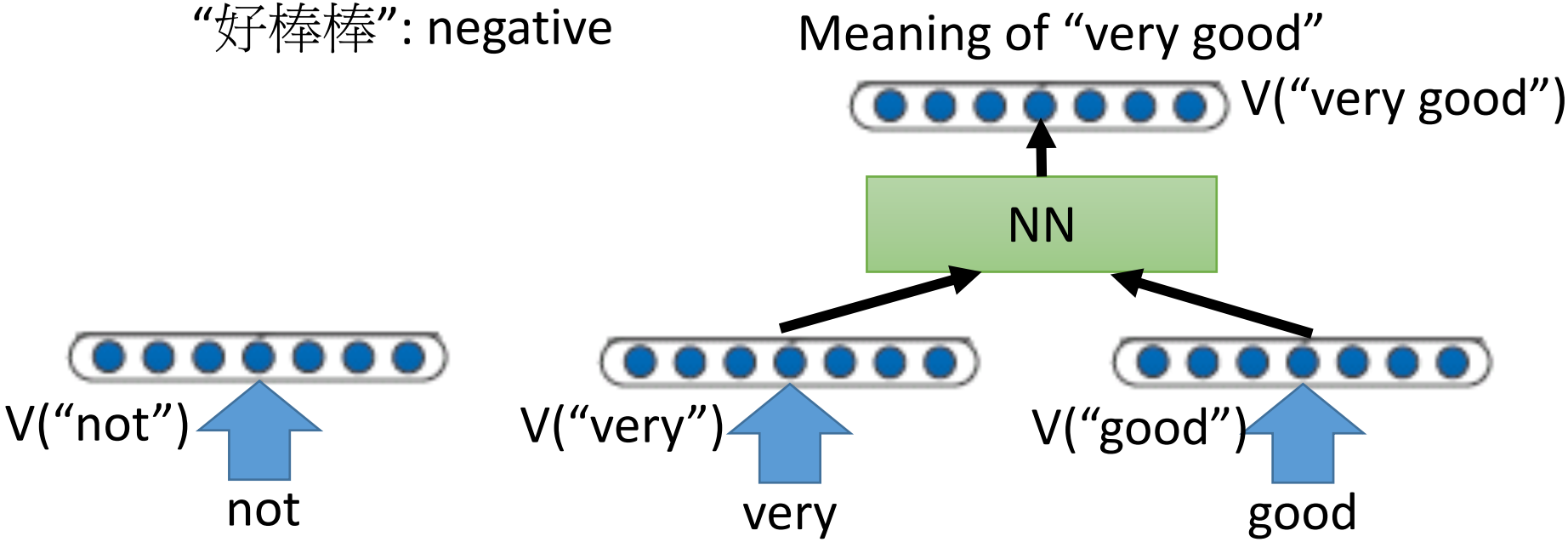
“好棒”: positive

“好棒棒”: negative

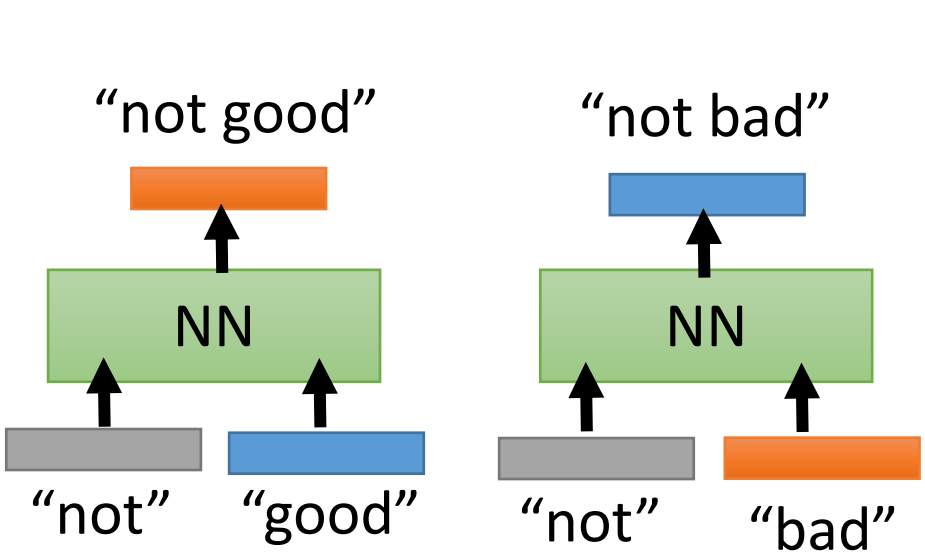
syntactic structure

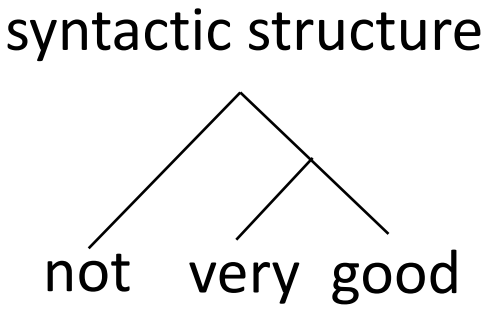


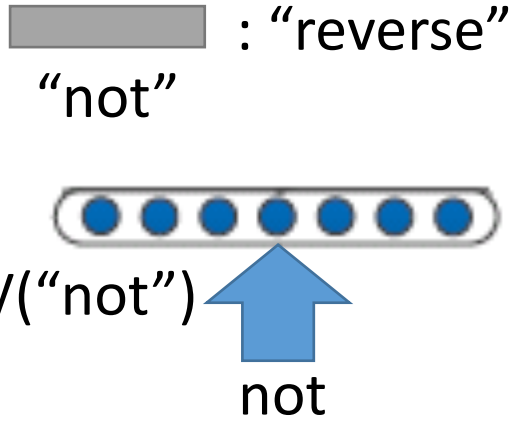
Meaning of “very good”

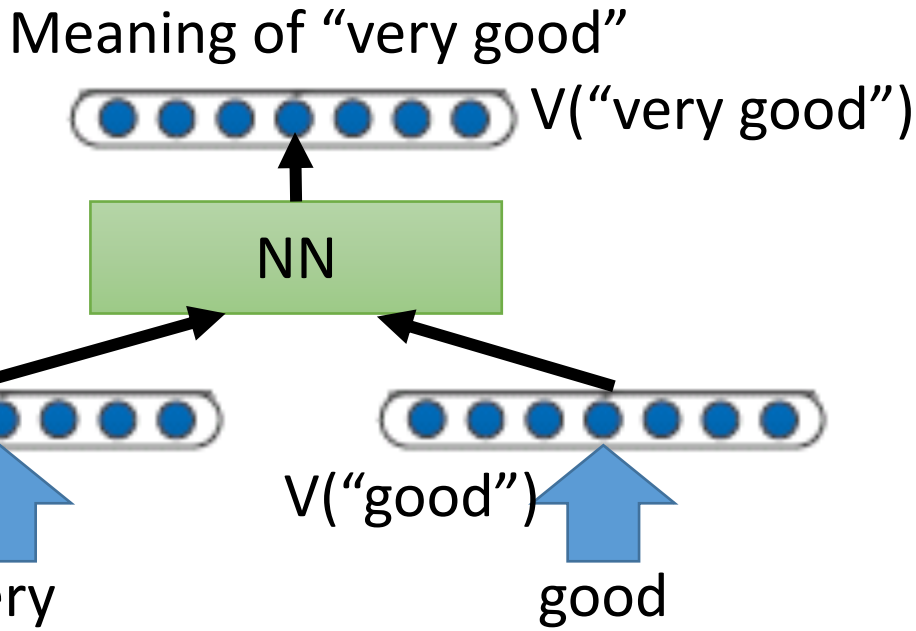


Recursive Deep Model

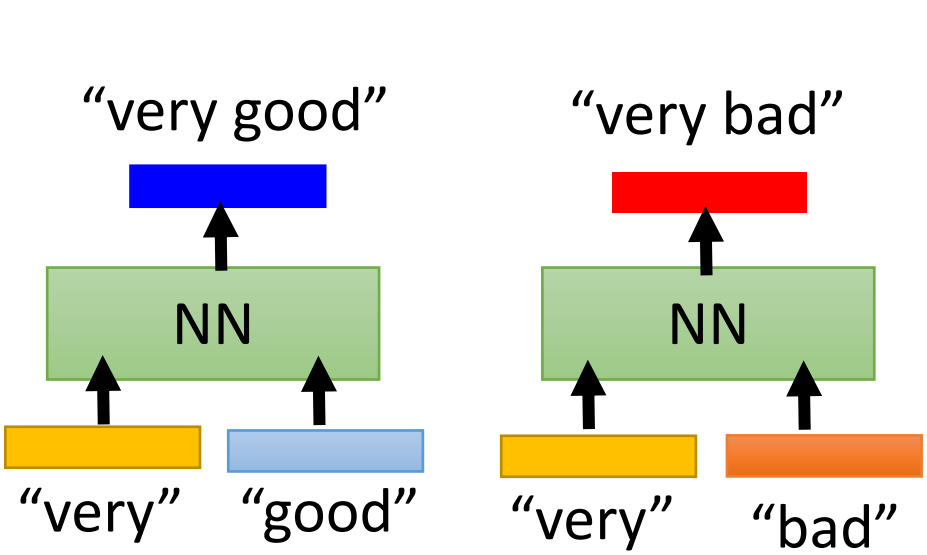


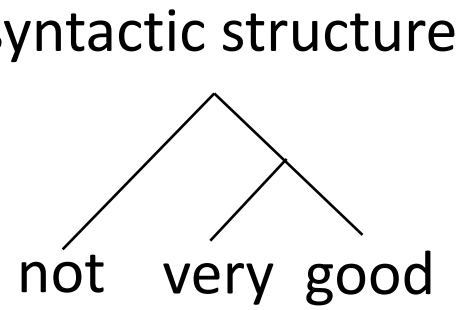


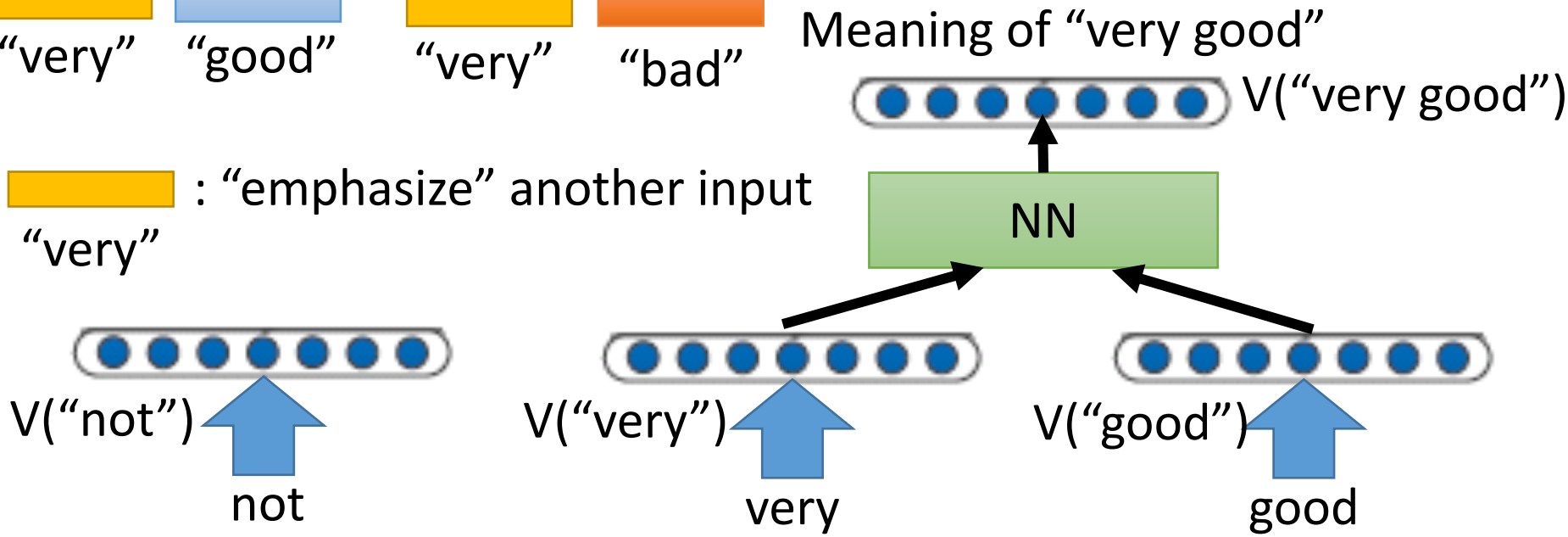




Recursive Deep Model

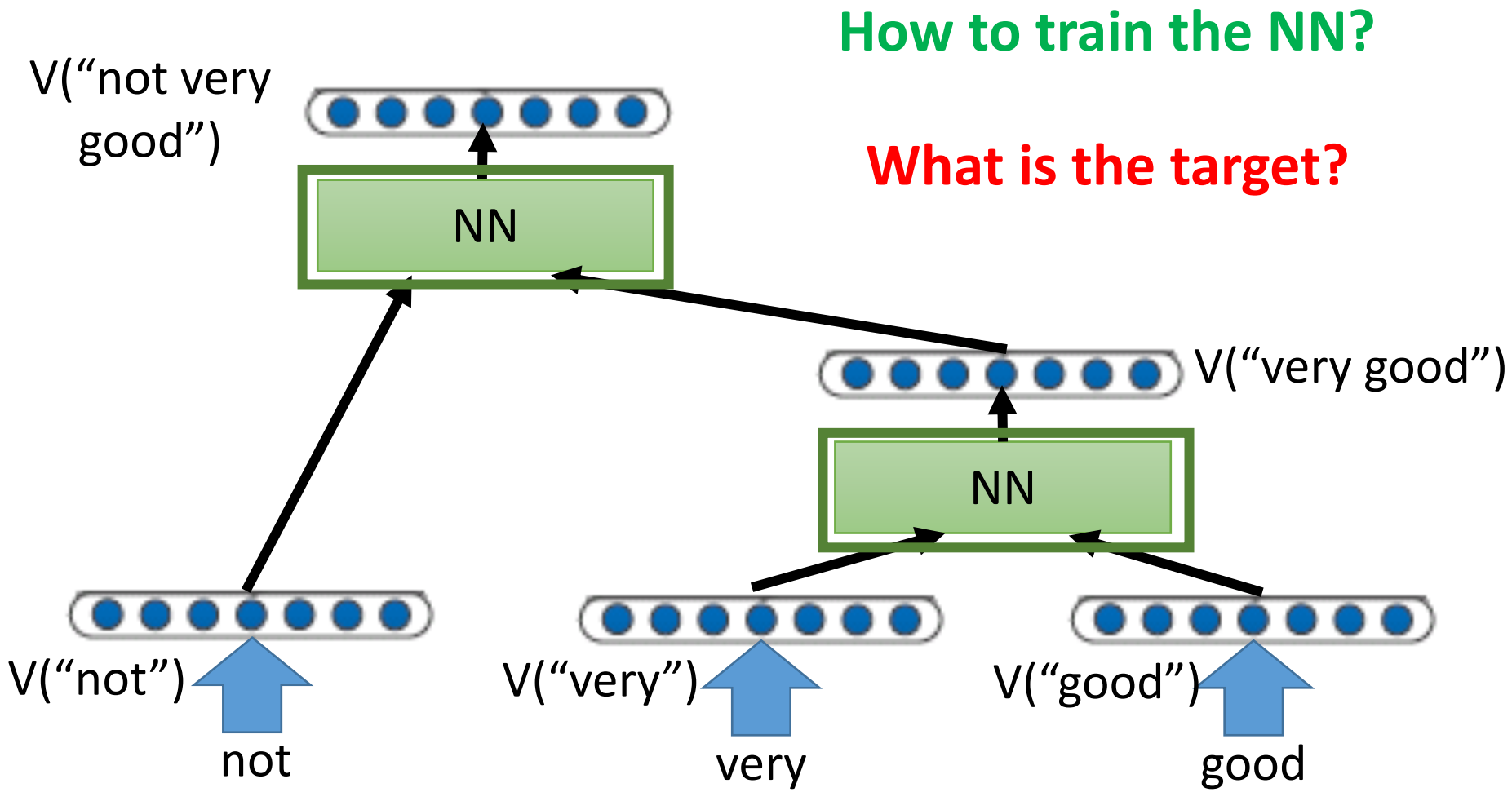






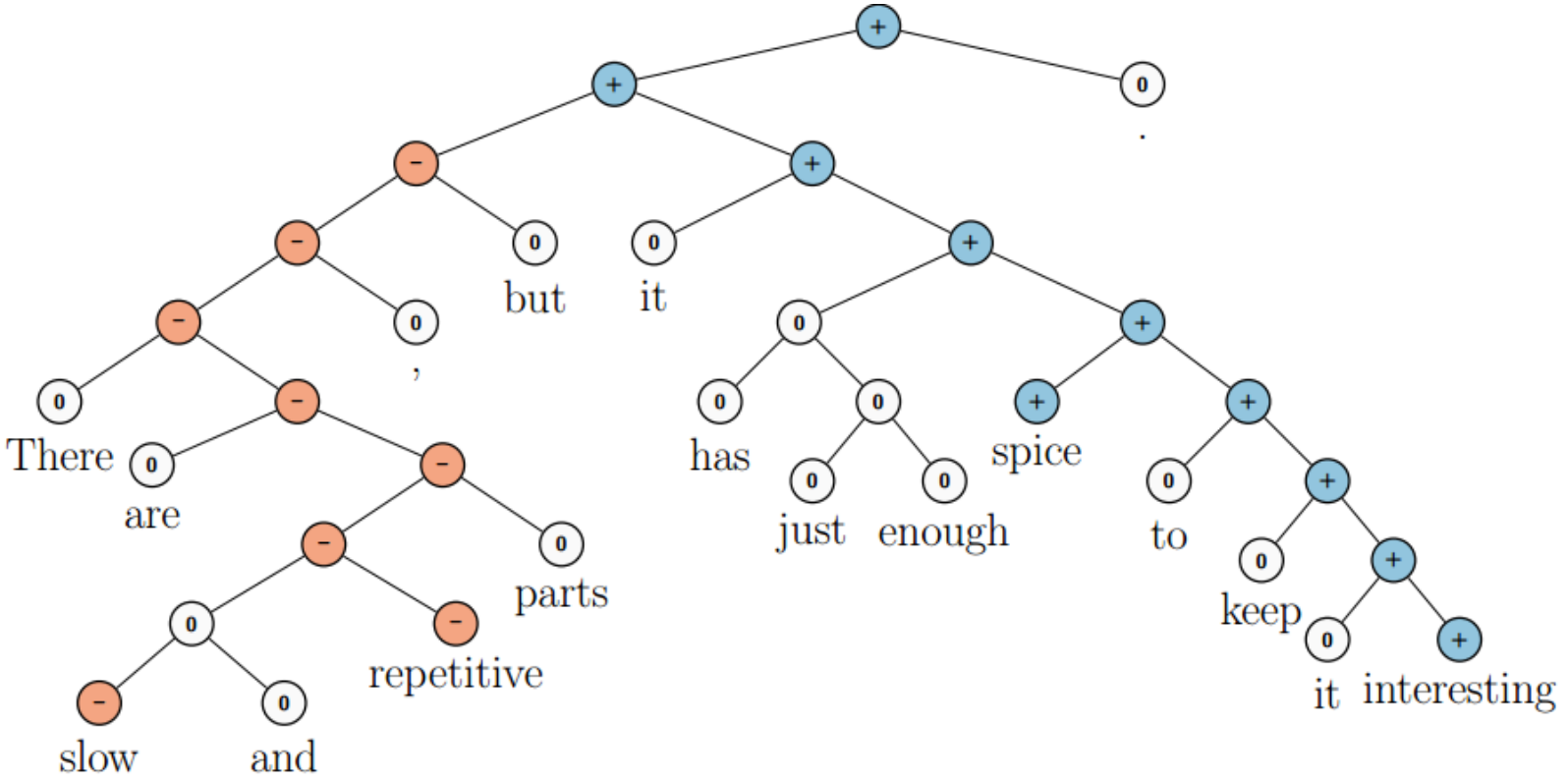
The word order is considered.

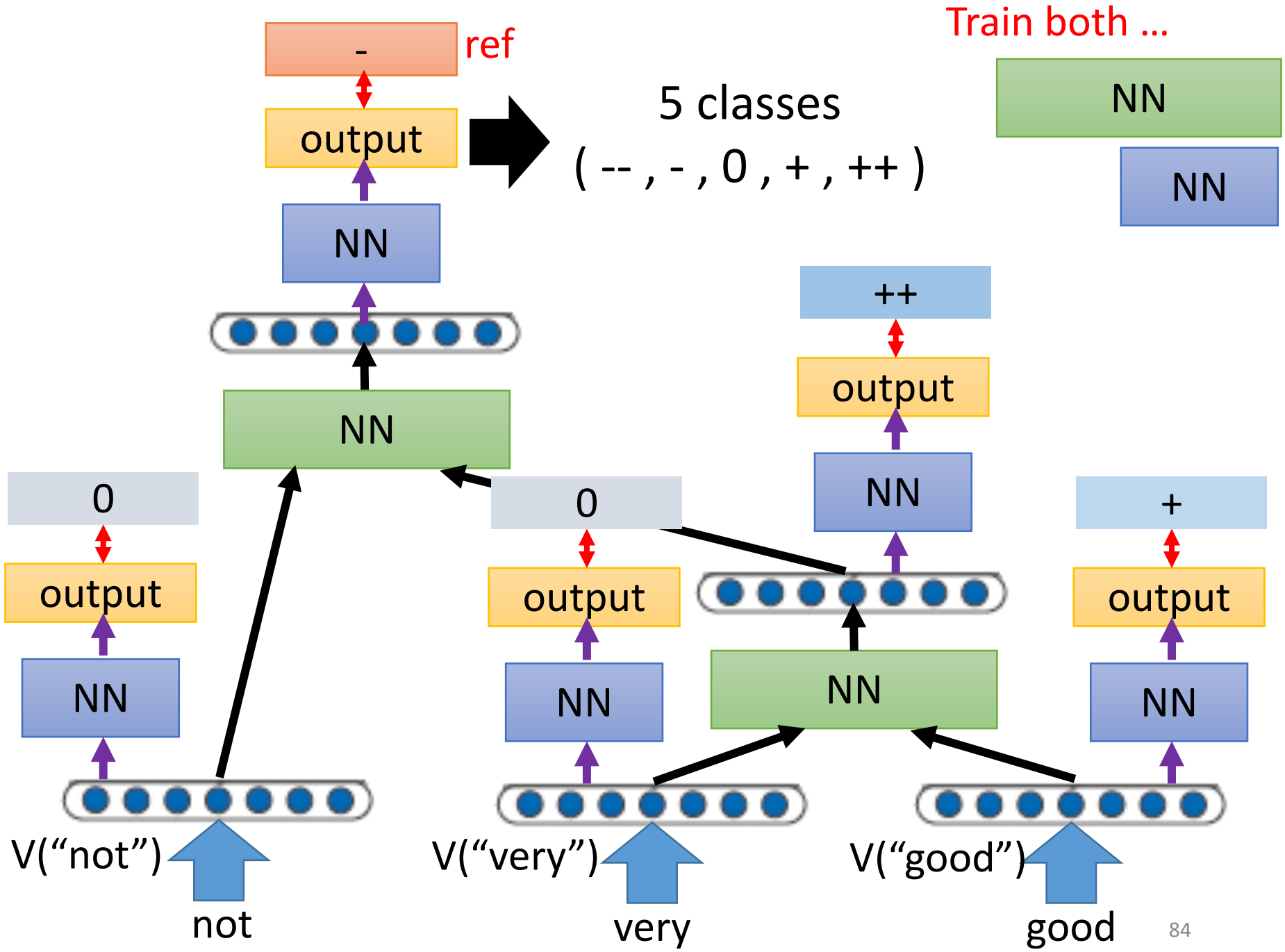
The representation of the sequence will change if the order of the words are changed



Need a Training Target ……

5-class sentiment classification (-- , - , 0 , + , ++)




Train both …
NN
NN
-
ref
output
5 classes
(-- , - , 0 , + , ++)
NN
NN
++
output
NN
0
output
NN
0
output
NN
NN
+
output
NN
V(“not”)
not
V(“very”)
very
V(“good”)
good

Need a Training Target ……

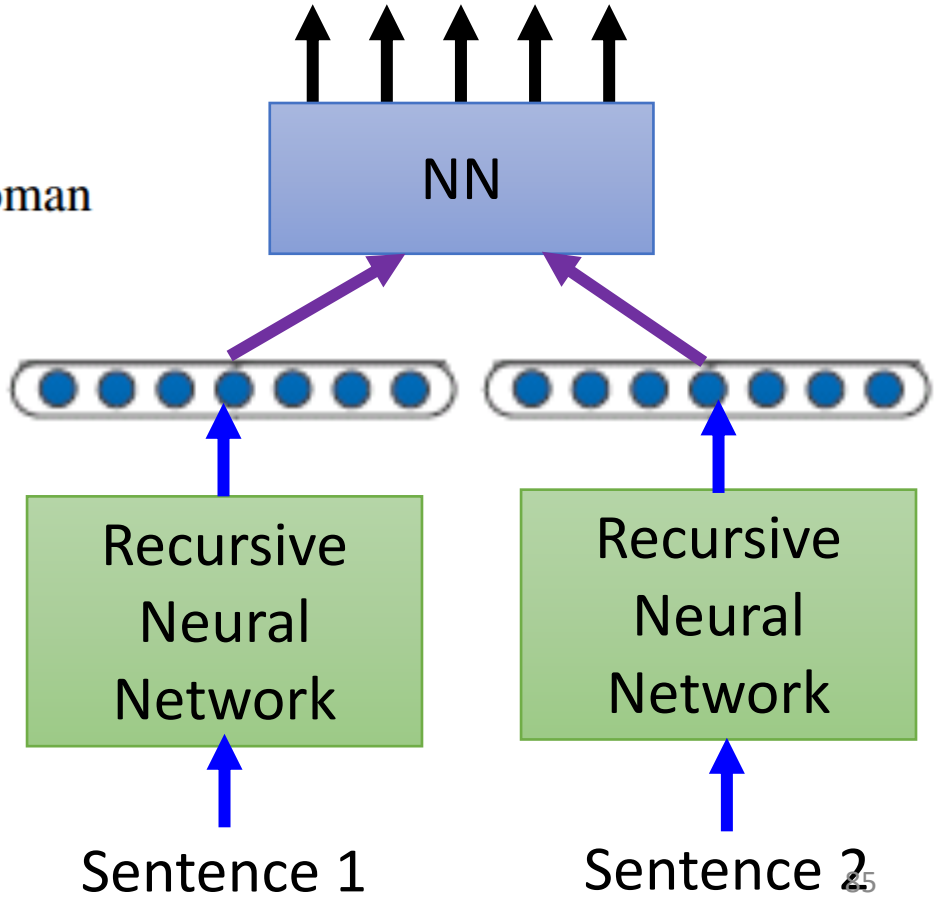
- Sentence relatedness

a woman is slicing potatoes

4.82 a woman is cutting potatoes

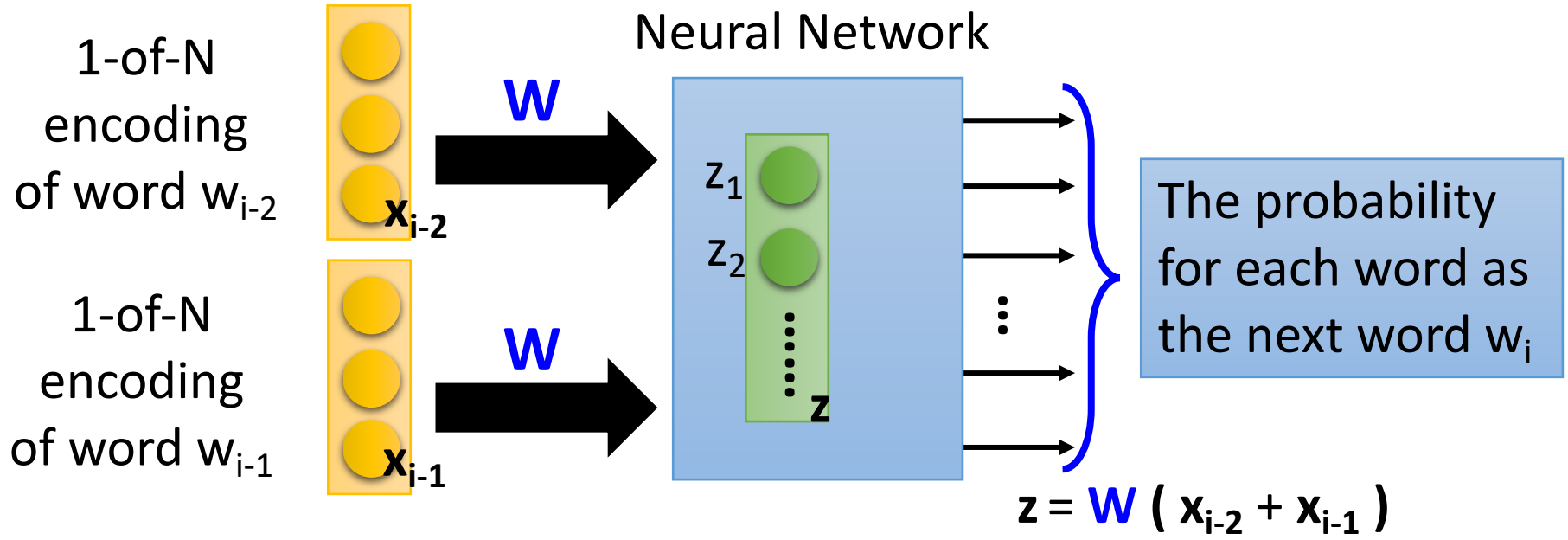
4.70 potatoes are being sliced by a woman

4.39 tofu is being sliced by a woman



Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. "Improved semantic representations from tree-structured long short-term memory networks." *arXiv preprint arXiv:1503.00075* (2015).

Paragraph Vector
Sequence-to-sequence auto-encoder



Paragraph d_1: (The paragraph is from "The lord of the ring")

…… 魔君 名叫 索倫 (Sauron) ……

w_{i-2} w_{i-1} w_i

$z = W (x_{i-2} + x_{i-1})$

Paragraph d_2: (The paragraph is from "仙五")

…… 魔君 名叫 姜世離 ……

w_{i-2} w_{i-1} w_i

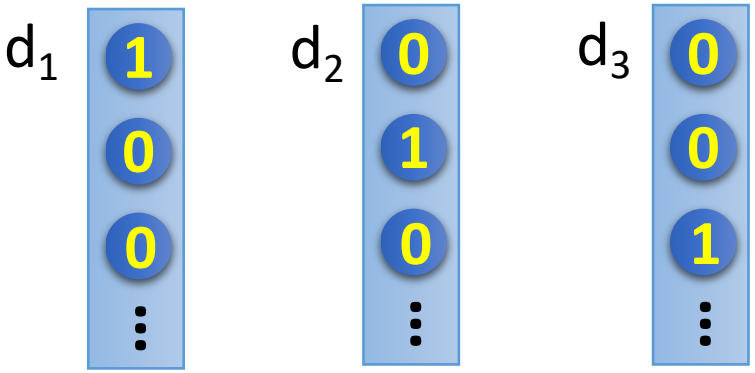
$z = W (x_{i-2} + x_{i-1})$

the same → Same output

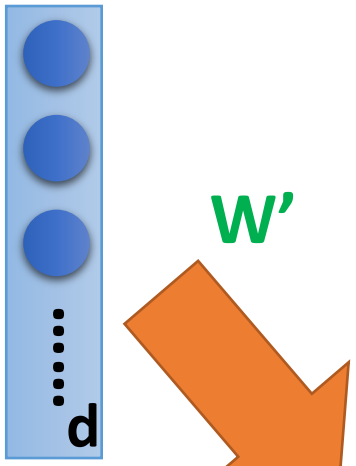
Paragraph Vector

Le, Quoc, and Tomas Mikolov. "Distributed Representations of Sentences and Documents." ICML, 2014

1-of-N encoding of paragraph d



1-of-N encoding of paragraph d



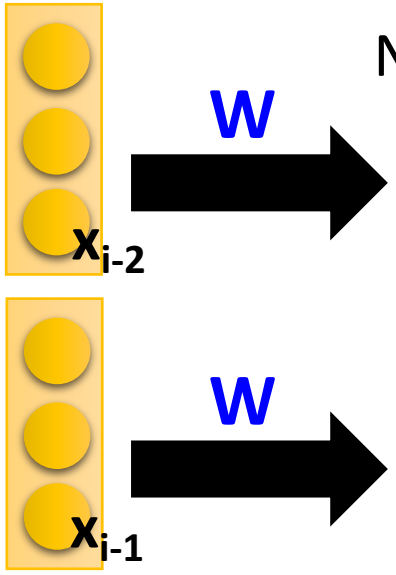
Original word vector: $z = W (x_{i-2} + x_{i-1})$

Paragraph vector:

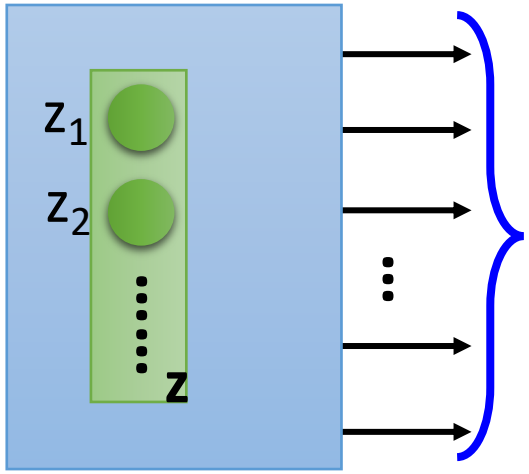
$z = W (x_{i-2} + x_{i-1}) + W' d$

1-of-N encoding of word w_{i-2}

1-of-N encoding of word w_{i-1}



Neural Network



The probability for each word as the next word w_i

Paragraph Vector

Le, Quoc, and Tomas Mikolov. "Distributed Representations of Sentences and Documents." ICML, 2014

Original word vector:

$$z = W (x_{i-2} + x_{i-1})$$

Paragraph vector:

$$z = W (x_{i-2} + x_{i-1}) + W' d$$

Then error of the prediction can be explained by the meaning of the paragraphs.

Paragraph d_1: (The paragraph is related to "The lord of the ring")

…… 魔君 名叫 索倫 (Sauron) ……

w_{i-2} w_{i-1} w_i

$z = W (x_{i-2} + x_{i-1}) + W' d_1$

different

Paragraph d_2: (The document is related to "仙五")

…… 魔君 名叫 姜世離 ……

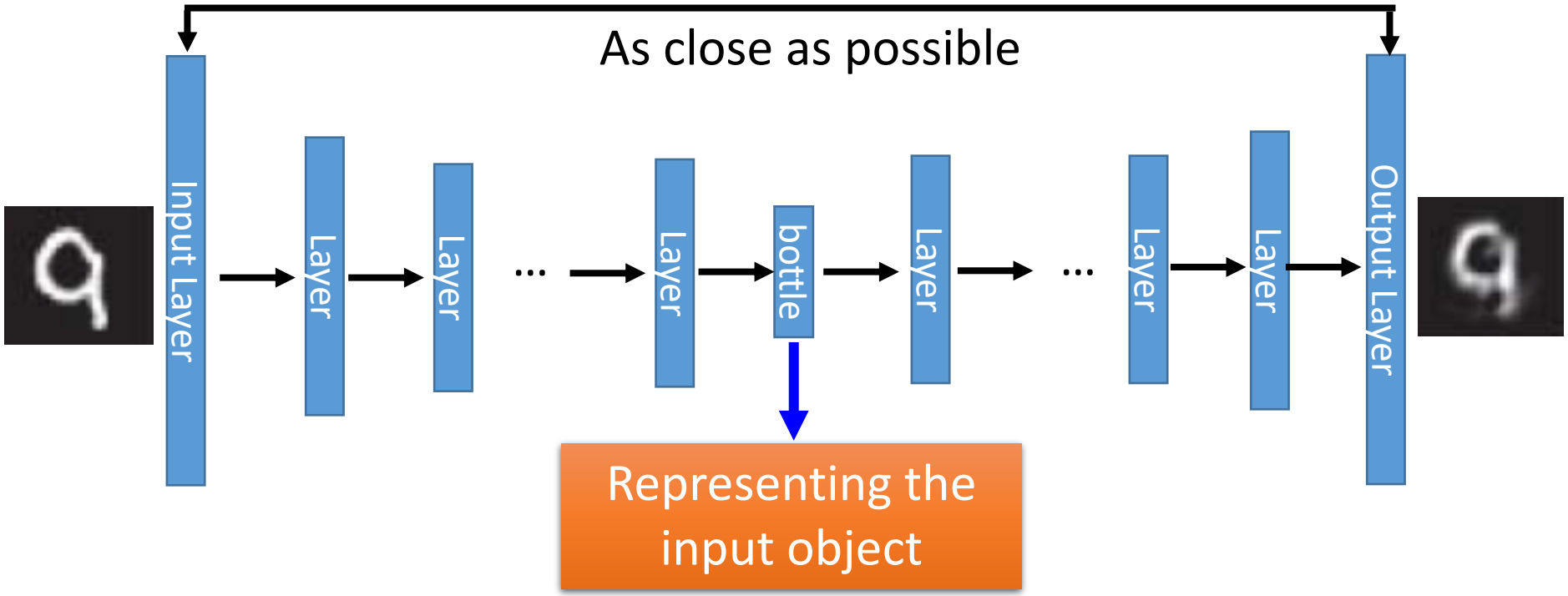
w_{i-2} w_{i-1} w_i

$z = W (x_{i-2} + x_{i-1}) + W' d_2$

Paragraph vector of d:V(d) = $W' d$ ➡ Meaning of the paragraph

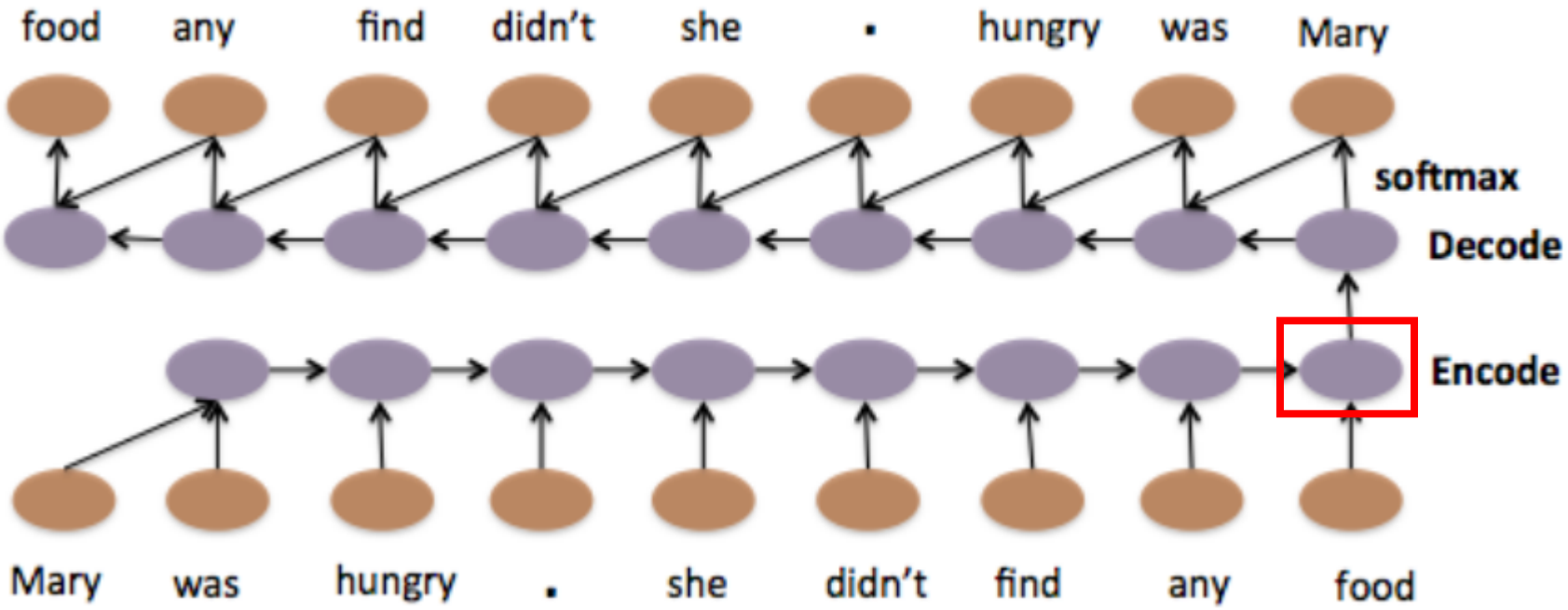
Sequence-to-sequence Auto-encoder

- Original Auto-encoder



Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

Sequence-to-sequence Auto-encoder



Li, Jiwei, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents." *arXiv preprint arXiv:1506.01057*(2015).

Tools

- 中文斷詞
  - Jieba
- 英文stopwords
  - NLTK stopwords
- Easiest way to use word2vec is via the Gensim libarary for Python (tends to be slowish, even though it tries to use C optimizations like Cython, NumPy)

https://radimrehurek.com/gensim/models/word2vec.html

FastText

- Library for fast text representation and classification
  - Courtesy of Facebook Research
- Recent state-of-the-art English word vectors.
- Word vectors for 157 languages trained on Wikipedia and Crawl.
- Models for language identification and various supervised tasks.

- https://github.com/facebookresearch/fastText?utm_source=mybridge&utm_medium=blog&utm_campaign=read_more

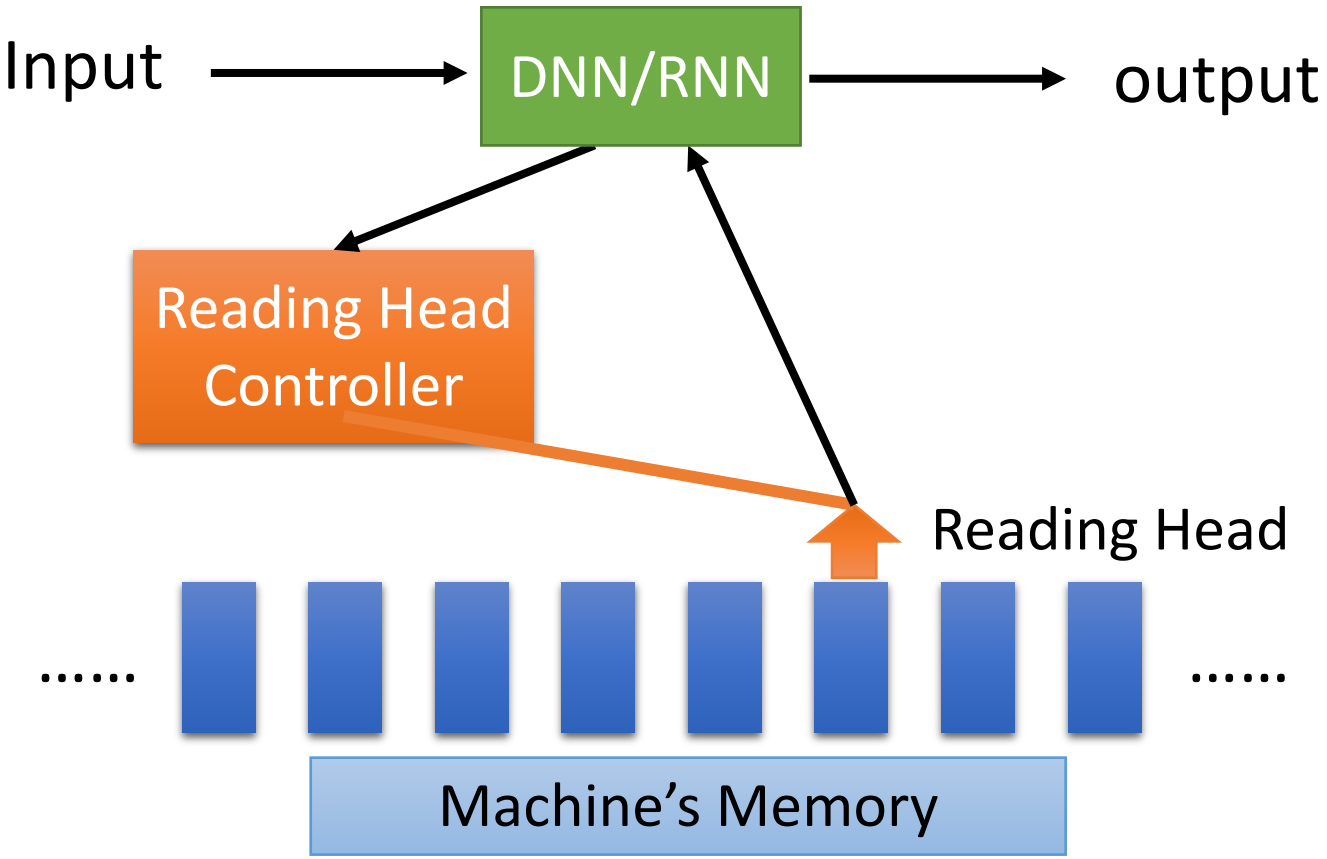
Fairseq

- sequence-to-sequence learning toolkit for Torch from Facebook AI Research
- English to French, English to German and English to Romanian translation

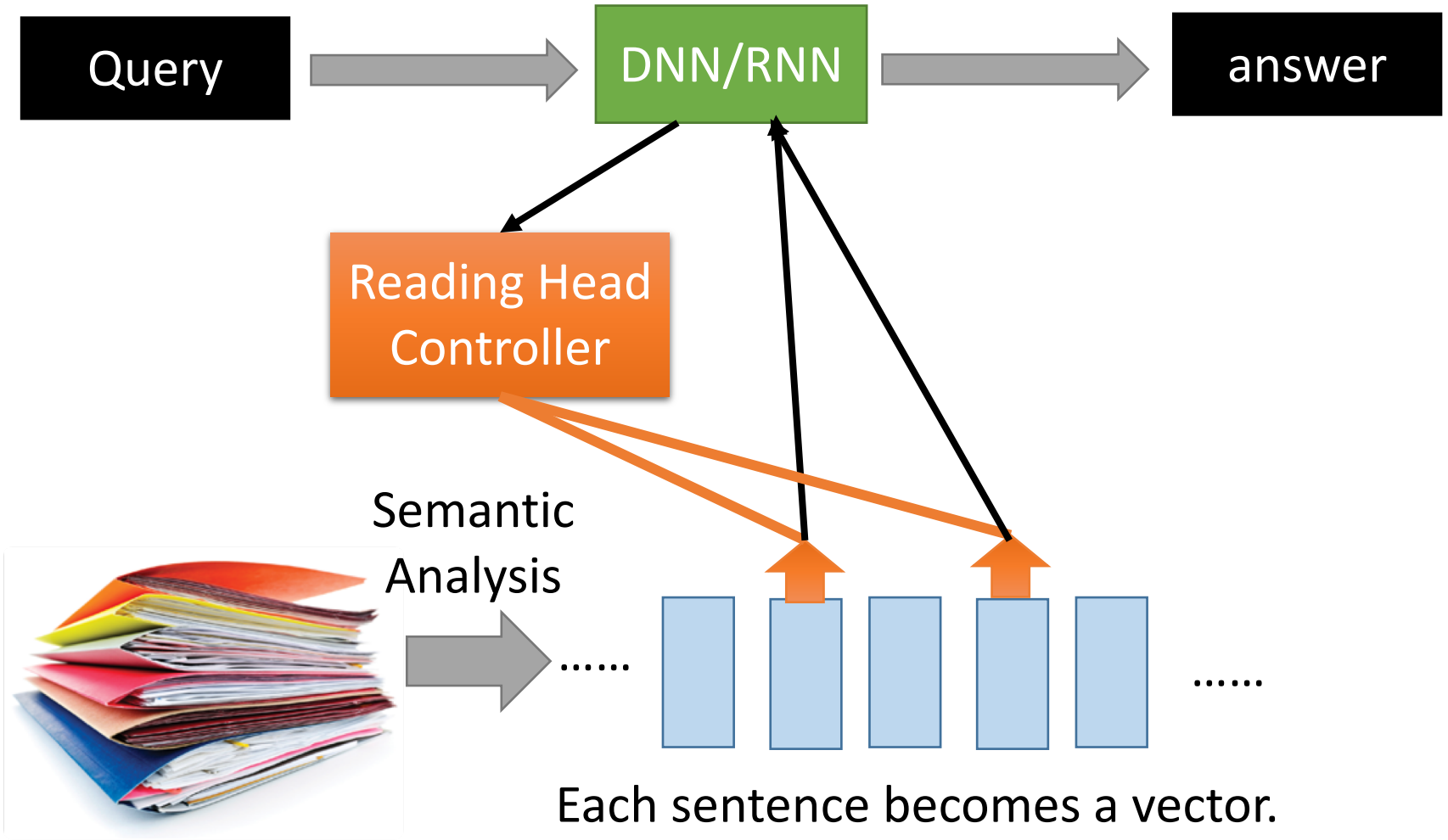
https://github.com/facebookresearch/fairseq?utm_source=mybridge&utm_medium=blog

Attention-based Model

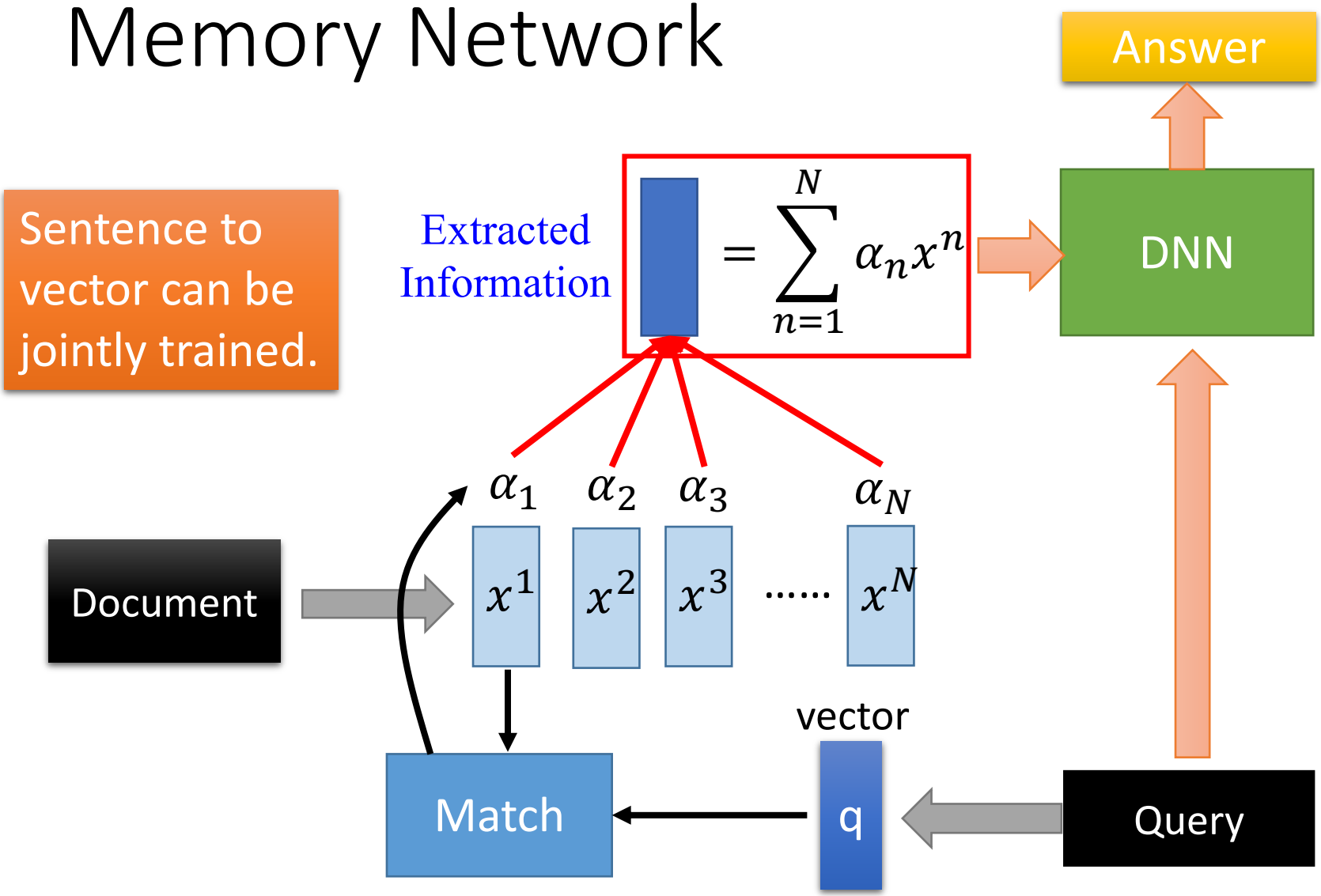
External Memory



Reading Comprehension

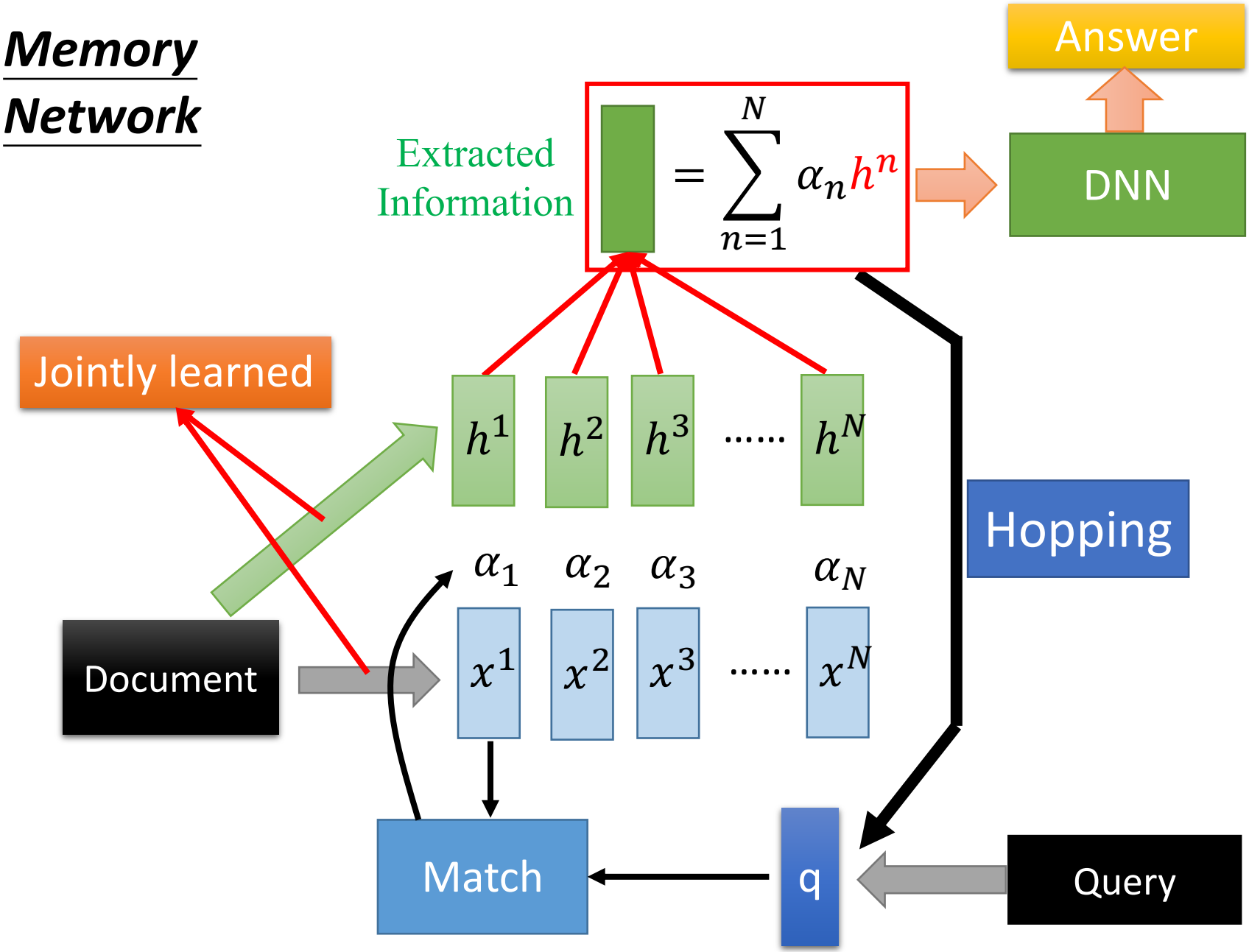


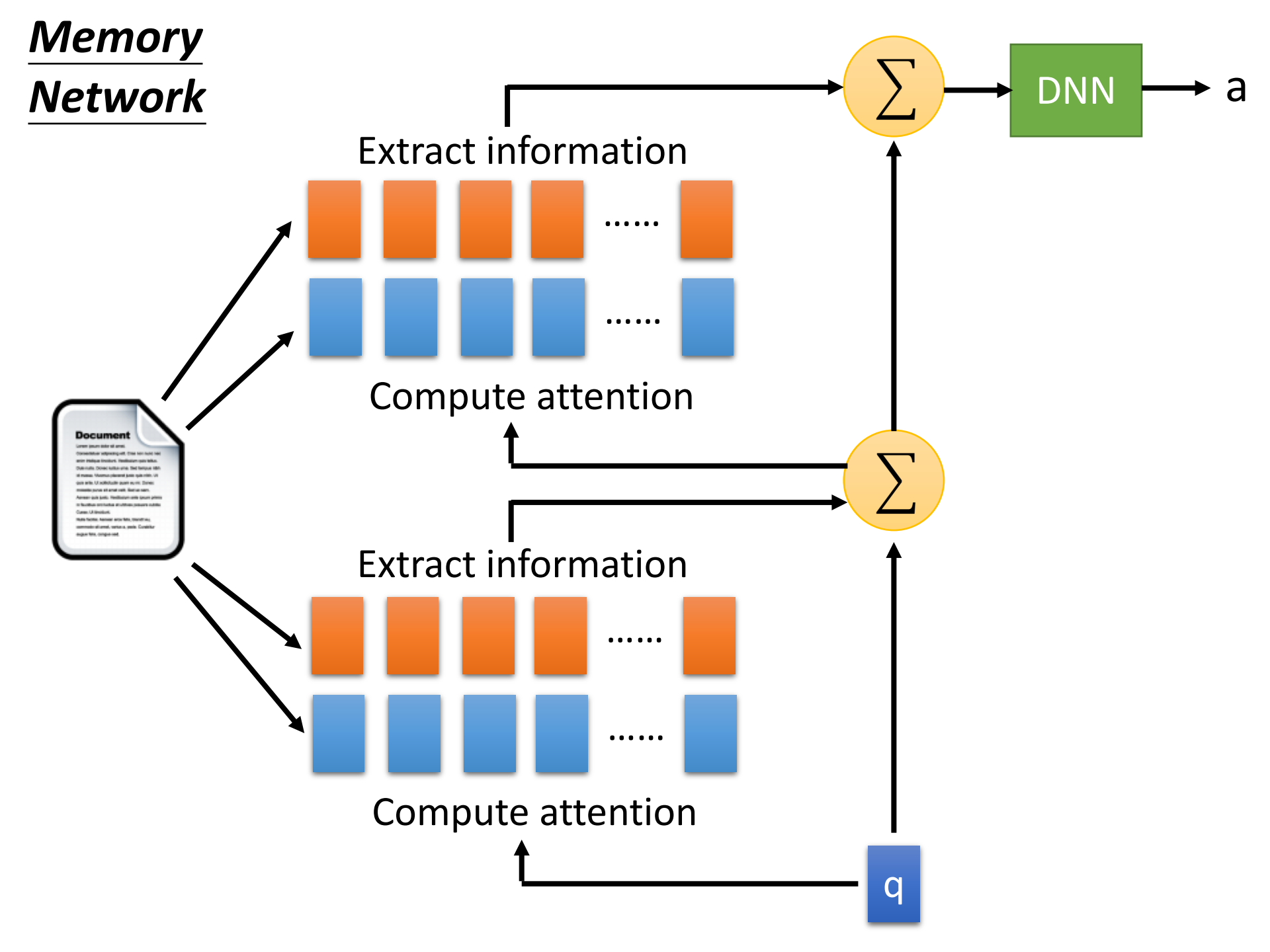
Memory Network



Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, "End-To-End Memory Networks", NIPS, 2015

Memory Network




Memory Network
Extract information
Compute attention
Extract information
Compute attention
Document
DNN
a
q

Multiple-hop

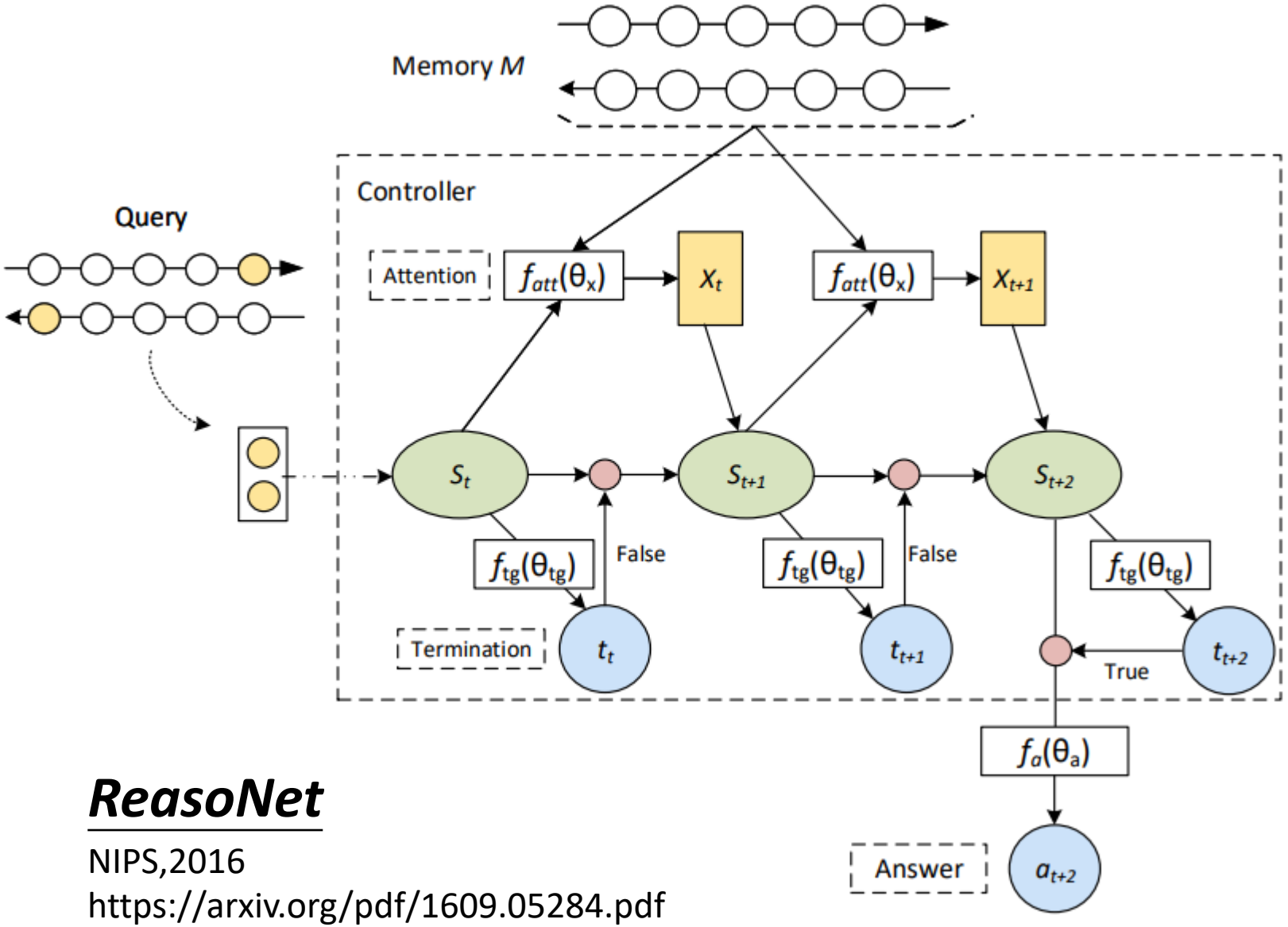
- End-To-End Memory Networks. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. NIPS, 2015.

The position of reading head:

Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow	**Prediction: yellow**			

Keras has example:
https://github.com/fchollet/keras/blob/master/examples/babi_memnn.py

Multiple-hop

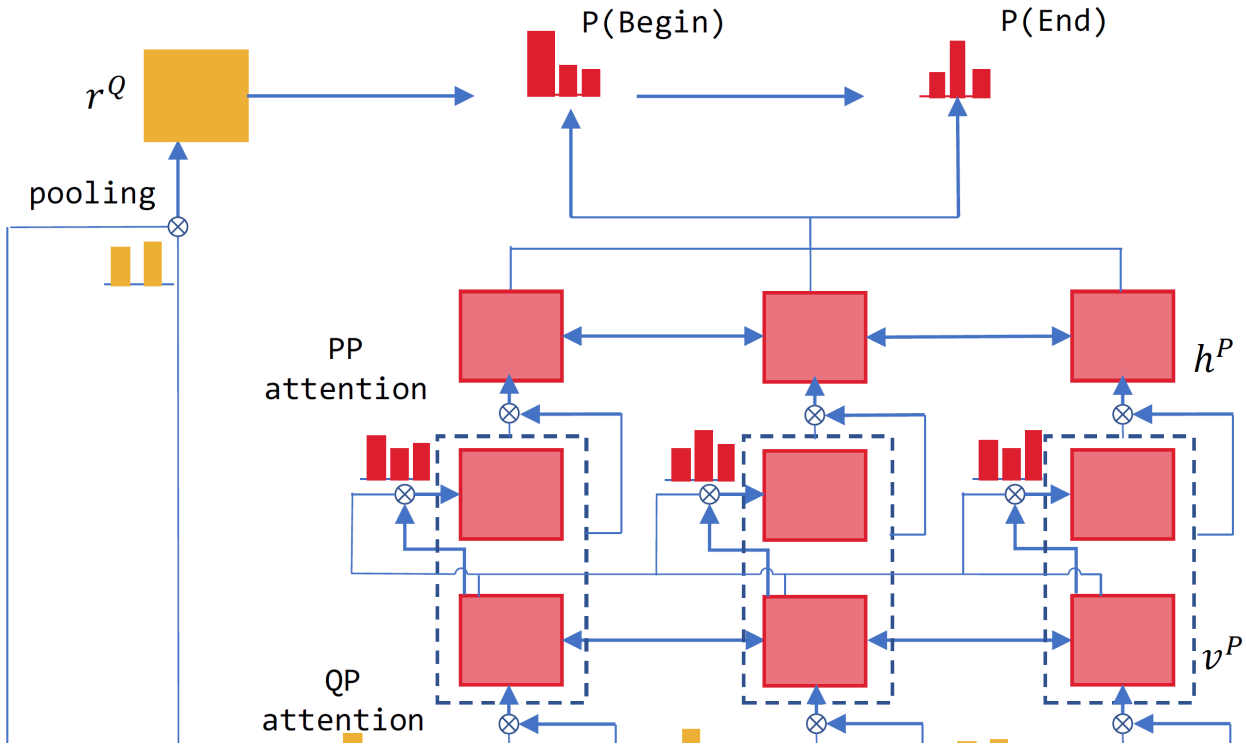


ReasoNet

NIPS,2016

https://arxiv.org/pdf/1609.05284.pdf

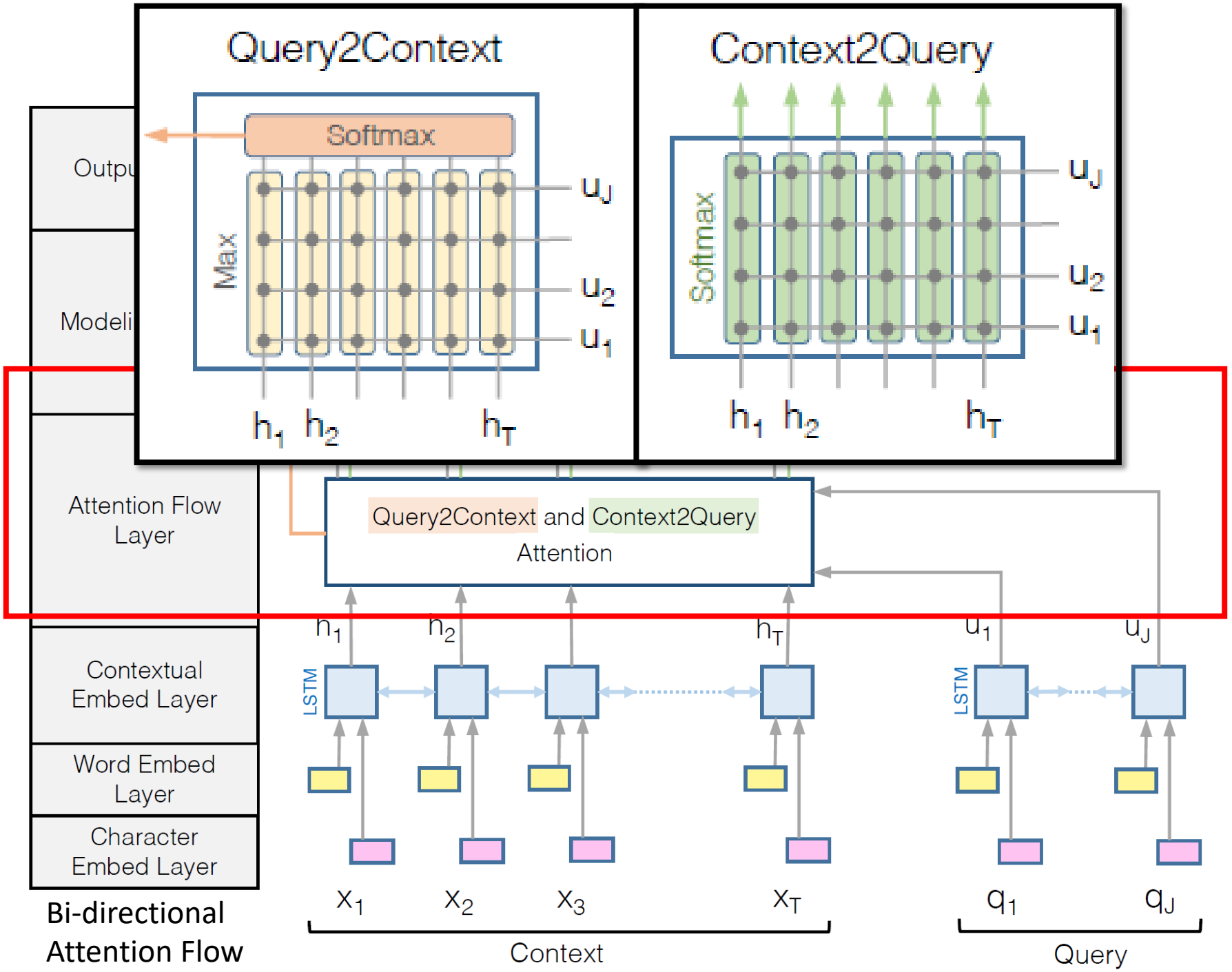
R-Net

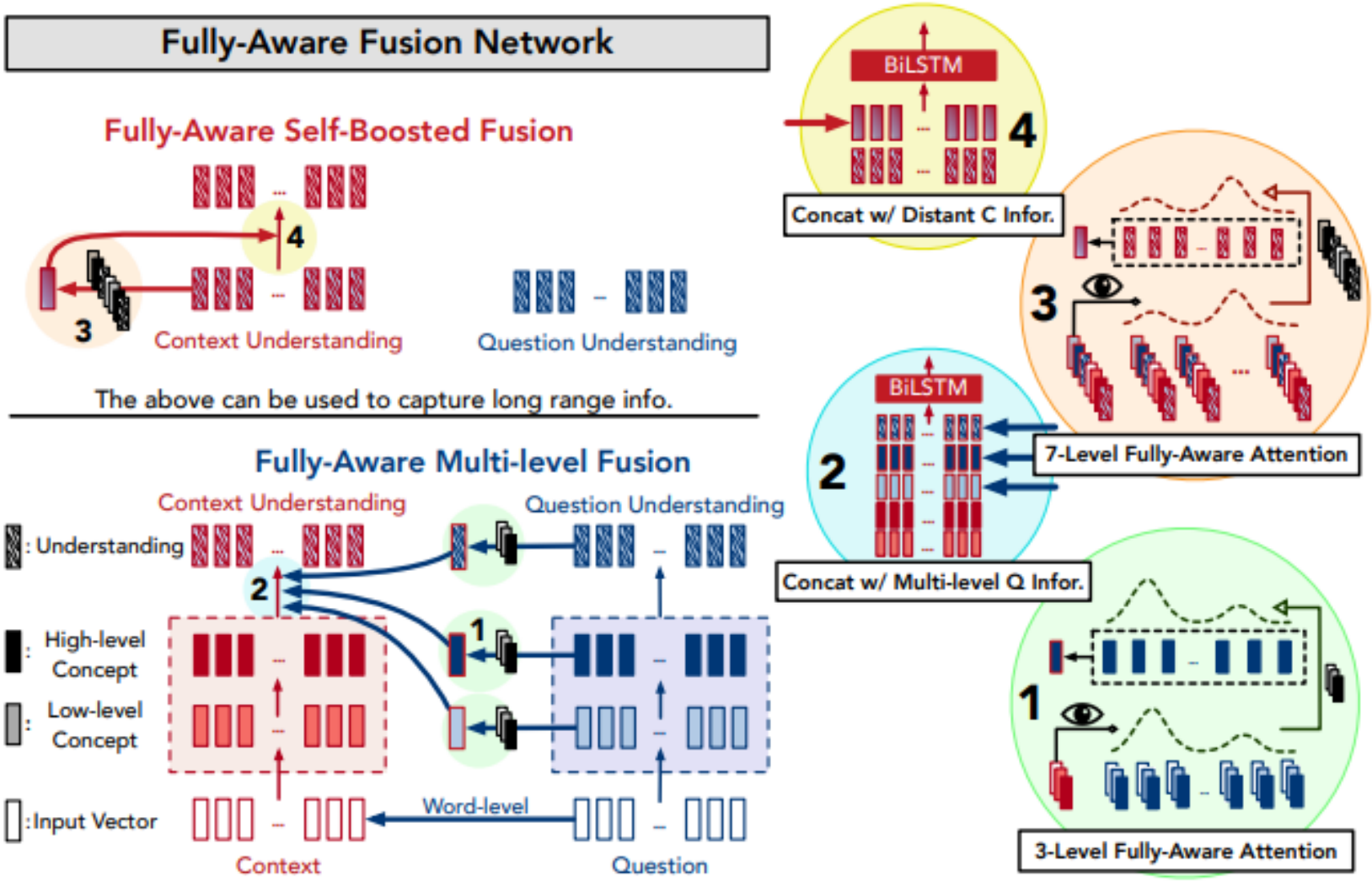


In 1870, Tesla moved to Karlovac, **to attend school at the Higher Real Gymnasium**, where he was profoundly influenced by a math teacher **Martin Sekulić**. The classes were held in **German**, as it was a school within the Austro-Hungarian Military Frontier. Tesla was able to perform integral calculus in his head, which prompted his teachers to believe that he was cheating. He finished a four-year term in three years, graduating in 1873.

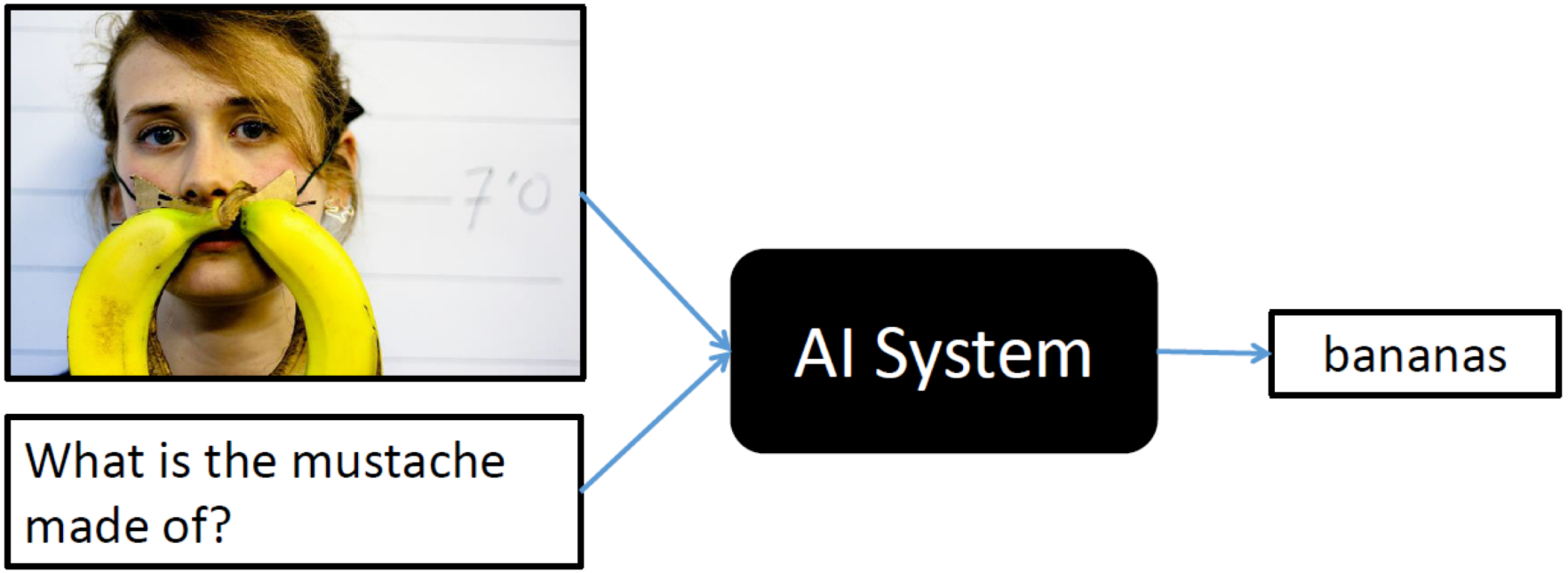
1. In what language were the classes given?	German
2. Who was Tesla's main influence in Karlovac?	Martin Sekulić
3. Why did Tesla go to Karlovac?	attend school at the Higher Real Gymnasium

Question Passage


Query2Context
Softmax
Max
u_J
u_2
u_1
h_1 h_2 h_T
Context2Query
Softmax
u_J
u_2
u_1
h_1 h_2 h_T
Output
Modeling
Attention Flow Layer
Query2Context and Context2Query Attention
Contextual Embed Layer
LSTM
h_1 h_2 h_T
u_1 u_J
Word Embed Layer
Character Embed Layer
x_1 x_2 x_3 x_T
q_1 q_J
Context
Query
Bi-directional Attention Flow

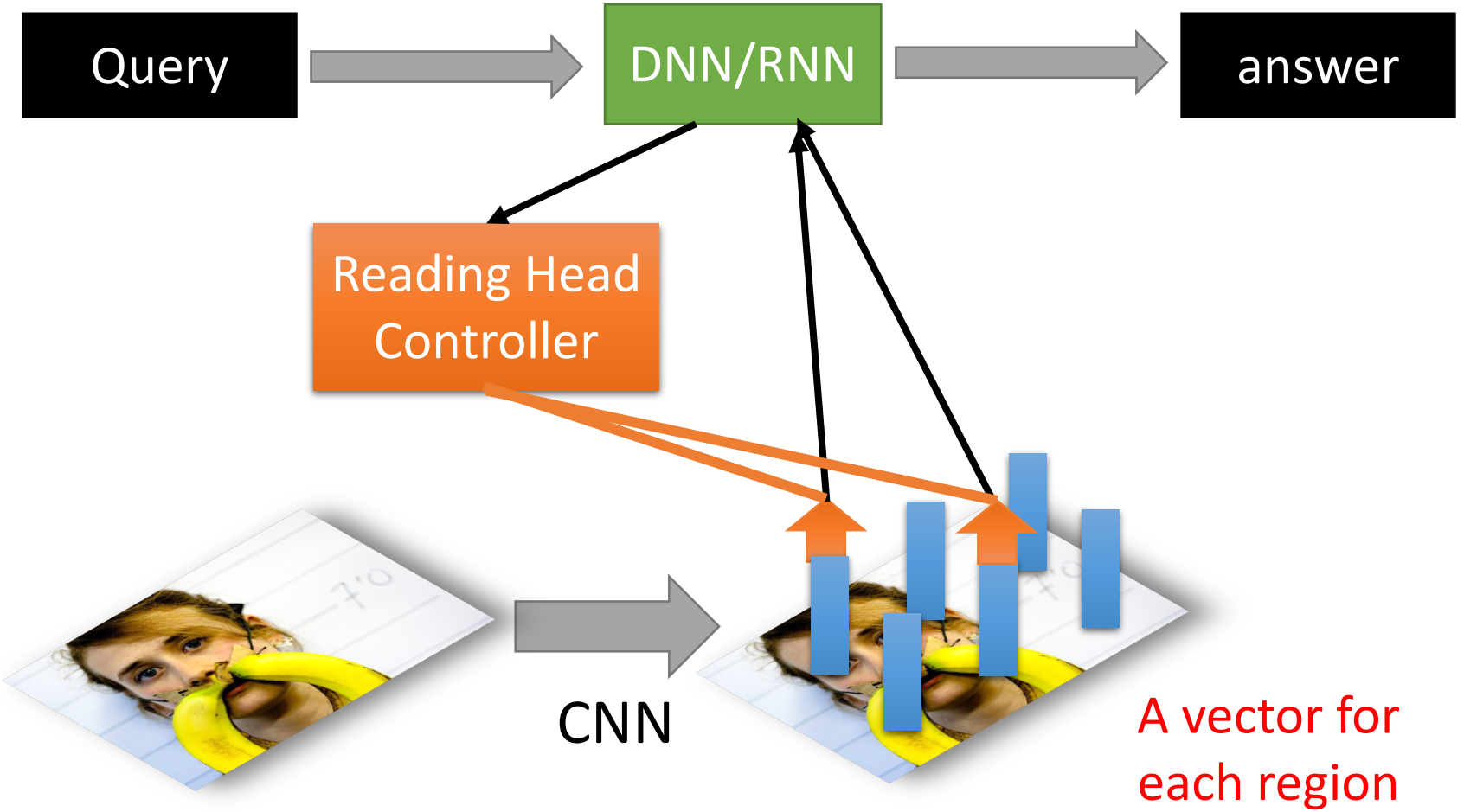

Fully-Aware Fusion Network
Fully-Aware Self-Boosted Fusion
4
3
Context Understanding
Question Understanding
The above can be used to capture long range info.
Fully-Aware Multi-level Fusion
Context Understanding
Question Understanding
: Understanding
2
1
: High-level Concept
: Low-level Concept
: Input Vector
Word-level
Context
Question
BiLSTM
4
Concat w/ Distant C Infor.
3
7-Level Fully-Aware Attention
BiLSTM
2
Concat w/ Multi-level Q Infor.
1
3-Level Fully-Aware Attention

Visual Question Answering



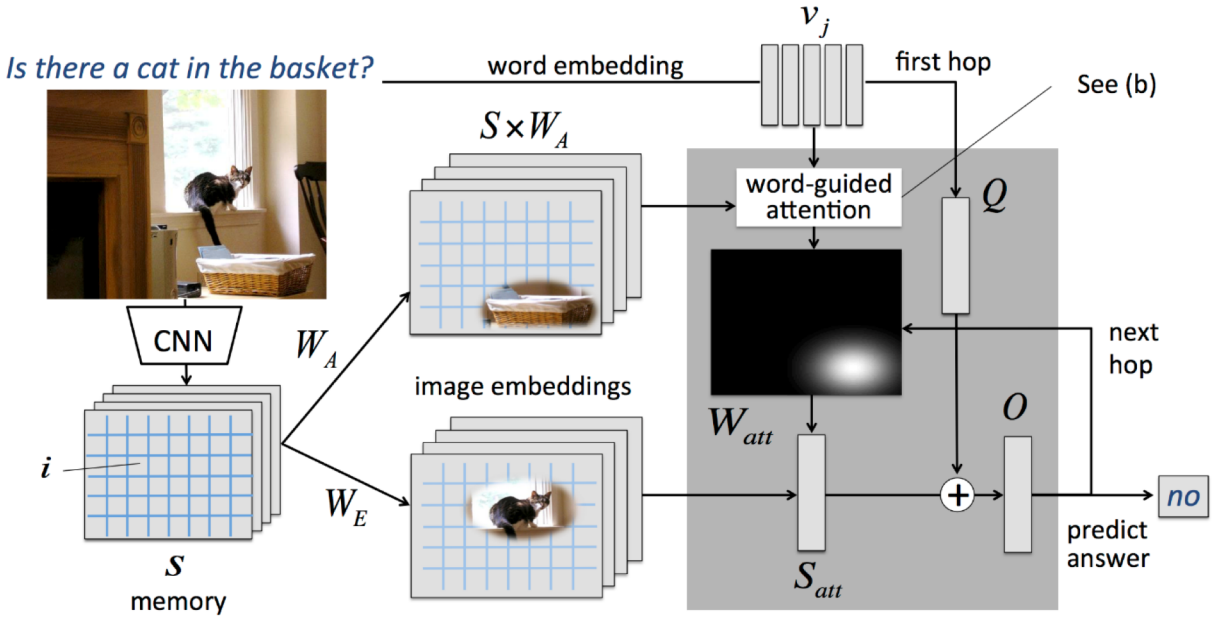
source: http://visualqa.org/

Visual Question Answering

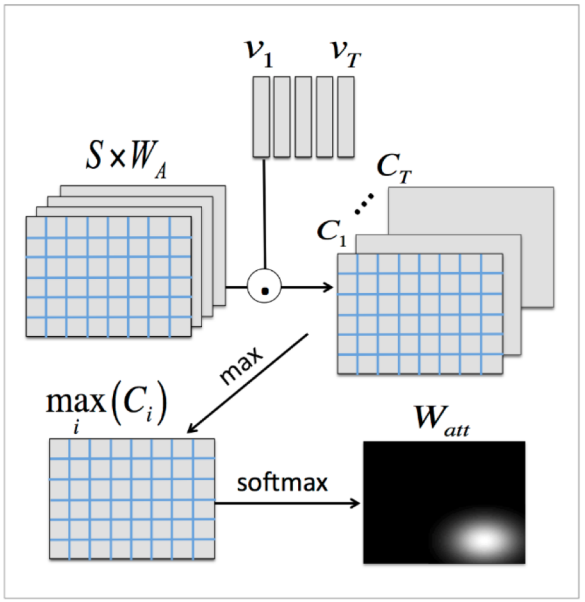


Visual Question Answering

- Huijuan Xu, Kate Saenko. Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. arXiv Pre-Print, 2015



(a) Overview



(b) Word-guided attention

Visual Question Answering

- Huijuan Xu, Kate Saenko. Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. ECCV, 2016.

Class Announcement and Quick Review

2020/10/13

Class Annoucement

- HW#1 due
  - Check the details…
  - The devil is hidden in the details.

- HW#2 starts
  - Act together we go far
  - Check points for three weeks (key point detection, descriptor, matching, blending, cylinder projection…)

Case study on Crawling and Analysis

- Headlines: 5 個步驟、55 次約會，美國工程師用 Python 成功脫魯！
  - 要如何在交友網站向女生搭訕，並且交到女朋友？如果你是工程師，會用 Python 的話，拜託不要再打「安安你好」了，因為這簡直糟蹋了你的專業技能。
  - 看美國超狂工程師 McKinlay 如何使用 Python 做數據分析，透過 5 個步驟，55 次約會，找到自己的靈魂伴侶。

https://buzzorange.com/techorange/2019/09/26/find-girlfriend-by-python/?fbclid=IwAR0JT4ZF9o8Dnv7Un5h3eTOVOTTAW59GBW6lH8oz2TtdjUmTdqrJ0bdQY4w

- Dating website: Match.com, J-Date, OkCupid
- 在 OkCupid 上爬取每一條相關資訊，利用關鍵的 K-Modes 的改良貝爾實驗室演算法找出數據規律，縮小範圍，然後一擊命中。

- 第一步：分析失敗原因，找到女人真正在意的問題
  - OkCupid:上千道問題的題庫裡選出 350 道問題：「以下哪種情形最有可能讓你去看電影？」或是「宗教/上帝在你的生活中有多重要？」
  - OkCupid 的演算法只會使用雙方都願意回答的問題去計算，而 McKinlay 選擇的問題有些隨機，並不主流。
  - 透過統計取樣，確定哪些問題是他喜歡類型的女人願意回答的，那麼他便可以誠實地回答這些問題而忽略其他問題，以此來建立一個全新的用戶。

- 狗 vs. 貓？
- 夜店 vs. 書店?
- 夏威夷披薩?
- 菸?
- 酒?
- …

- 第二步：借助 Python，求助好友，收集 2 萬個女性網友數據
  - 利用 Python 腳本瀏覽了上百道 OkCupid 的問卷題目，然後把女性用戶分為七類，每一類都會貼上獨特的標籤，例如「多才多藝的」和「細心體貼的」等等。
  - 建了 12 個假的 OkCupid 帳號並寫好了 Python 腳本去管理這些帳號。這個腳本會搜尋他的目標群體（25 到 45 歲的異性戀或雙性戀的女人），訪問她們的主頁，然後爬取她們帳號上每一條有用的資訊：種族、身高、是否吸煙、星座…

三週後，他已經收集了來自全國各地 20,000 名女性的 600 萬個問題和答案。

- 第三步：將 20,000 名女性分為 7 類，找出最適合自己的類型
- 第四步：真誠填寫女性關注的問題，找出靈魂伴侶
  - 文本挖掘，以瞭解她們感興趣的內容：教學是一個熱門話題，因此他撰寫了一篇強調了他作為數學教授工作的文章。
  - 兩個類群中最受歡迎的 500 個問題，並決定誠實地填寫答案。

- 第五步：走出房間，來一場真正的約會
  - 6 月 30 日， McKinlay 開著他的 Nissan，穿越城鎮來到加州大學洛杉磯分校的健身房，開始他的實際約會。Sheila 是一位來自「A 類群」的年輕的網頁設計師。他們在 Echo Park 的一家咖啡廳吃了午餐。「太不可思議了，這簡直像一場學術活動一樣。
  - 雖然他被拒絕了，但每天仍然收到 20 條消息。實際約會顯然與電腦配置文件約會是完全不同的。他開始忽略一些主頁沒有內容的人的消息，只回應那些有幽默感或在首頁展示一些有趣的東西的人。
  - 大多數不成功的約會者都面臨著自尊問題。對於 McKinlay 來說其實更糟糕。他不得不檢查他的計算。

SPAM

專任師資 | IEEE Fellow | 兼任師資 | 退離職教師 | 其他老師 | 主管及行政人員

教授兼系主任

姓名：陳科宏

職稱：教授兼系主任

電子郵件：khchen@cn.nctu.edu.tw

聯絡電話：03-5712121 ext54390

副教授兼副系主任

姓名：冀泰石

職稱：副教授兼副系主任

電子郵件：tschi@mail.nctu.edu.tw

Example

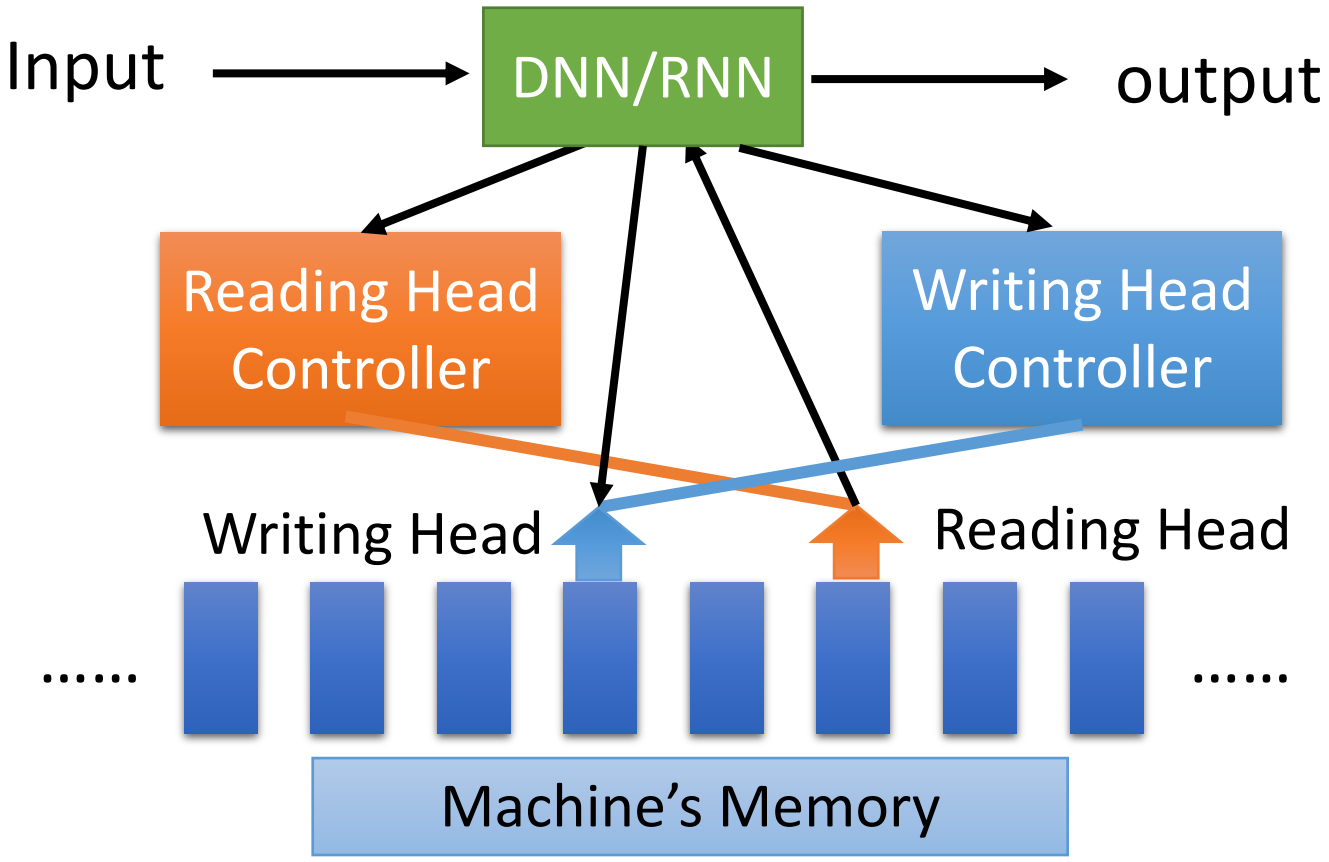
I need a favour from you, please email me back as soon as possible.

Ke-Horng Chen
Chairman

Kind regards

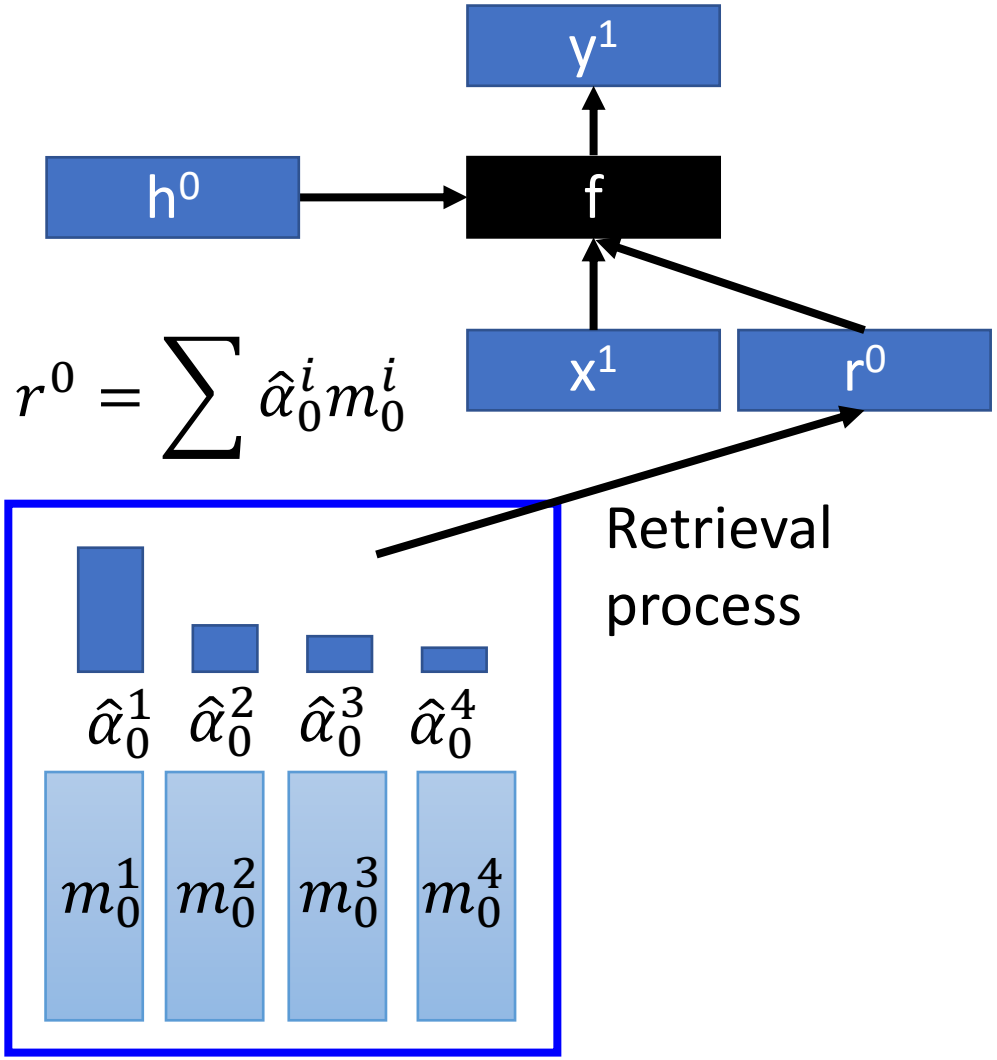
Sent from my iPad

External Memory v2

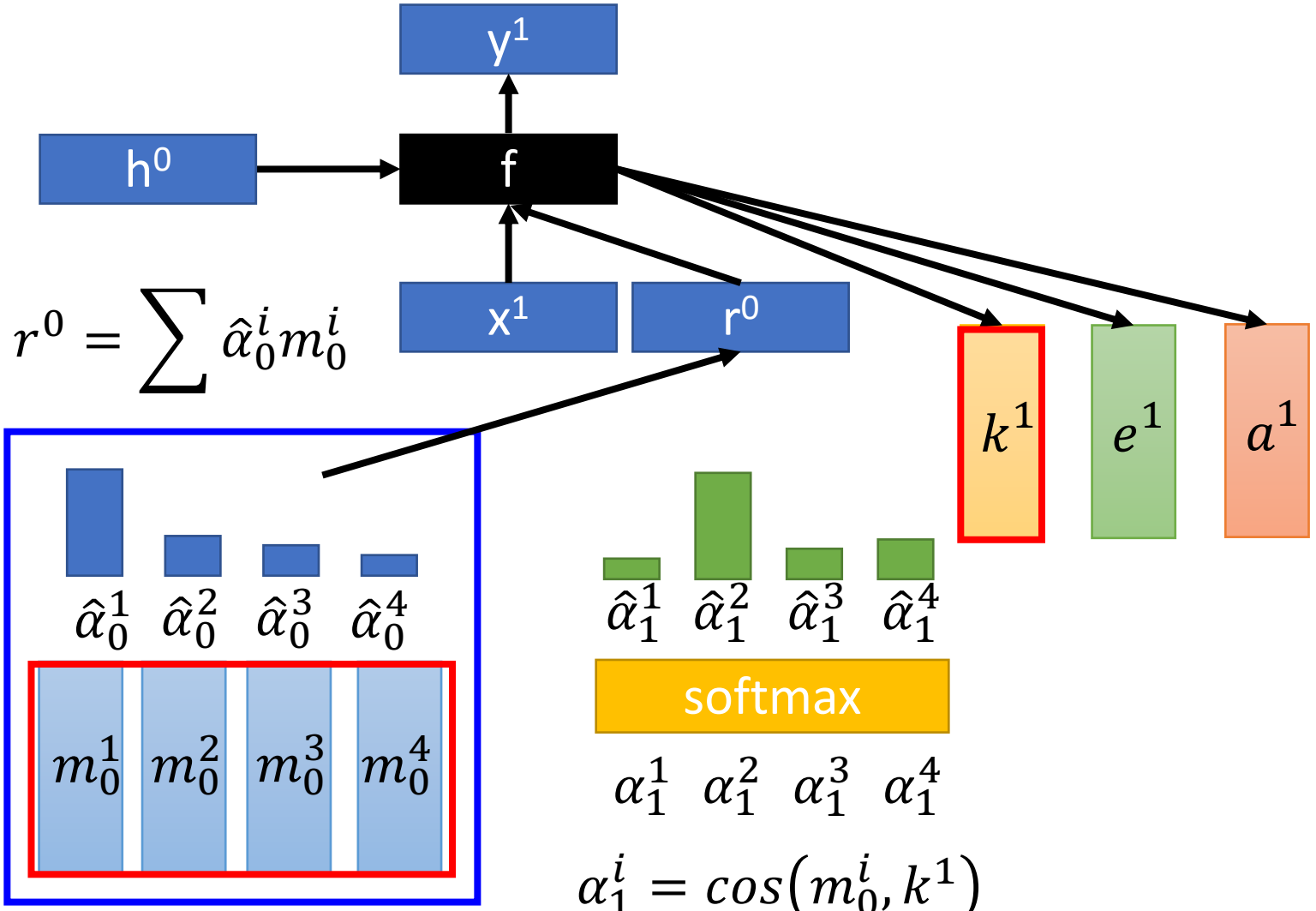


Neural Turing Machine

Neural Turing Machine



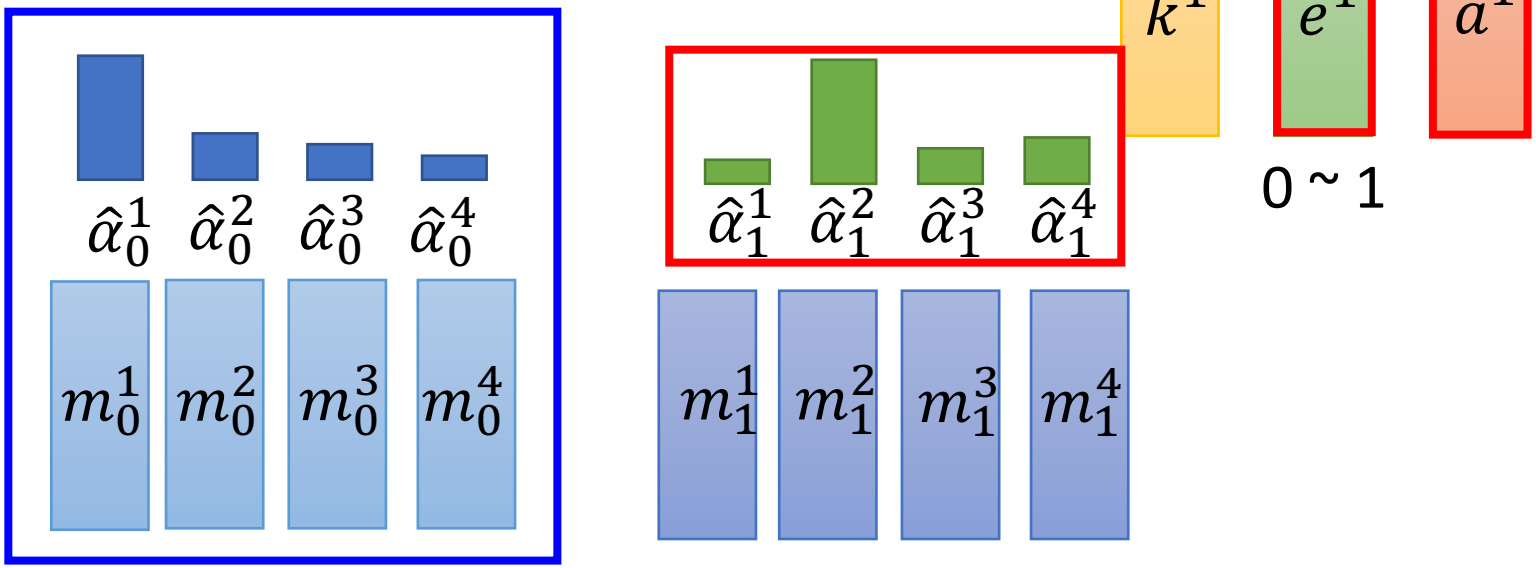
Neural Turing Machine



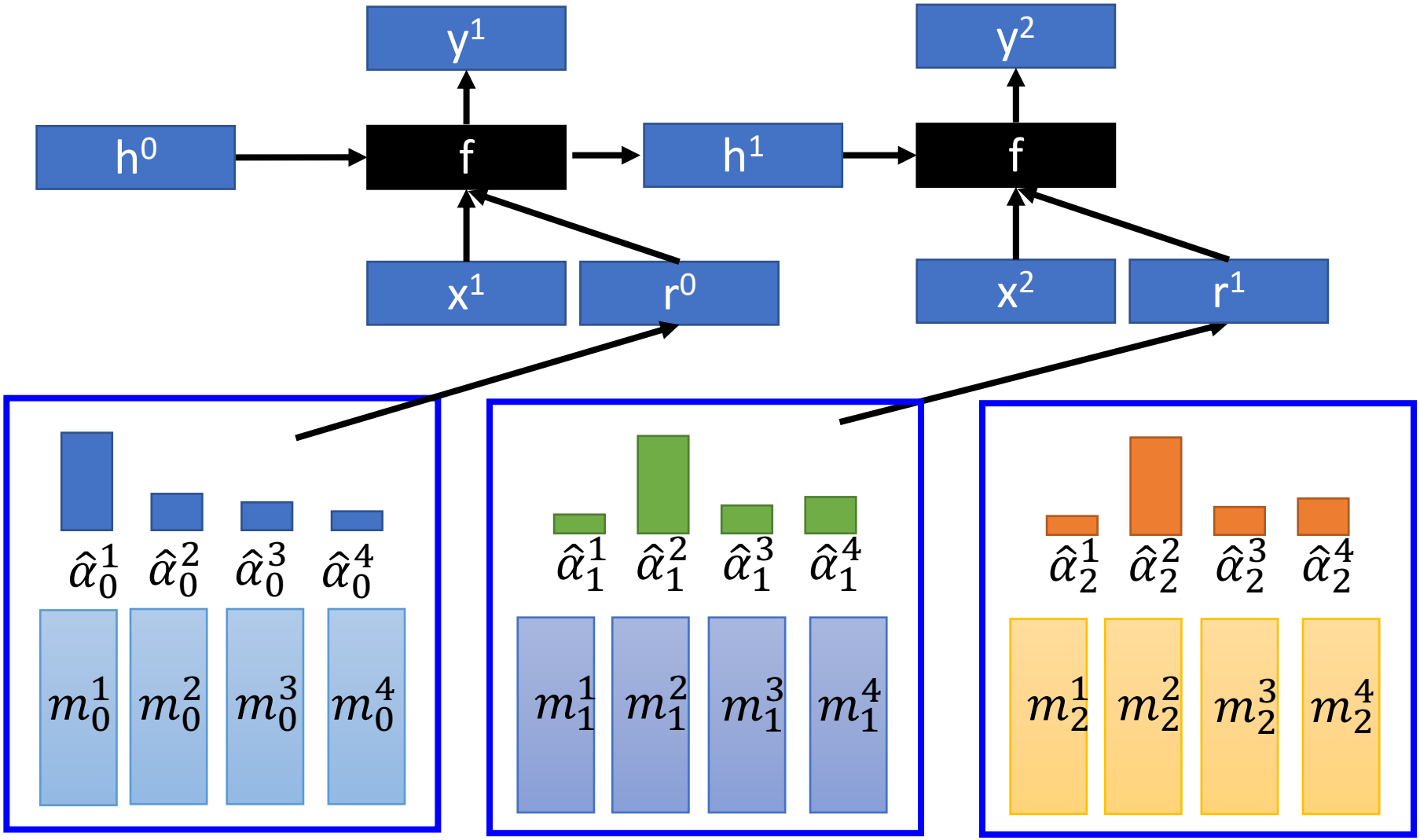
Neural Turing Machine

$$m_1^i = m_0^i - \hat{\alpha}_1^i \, e^1 \odot m_0^i + \hat{\alpha}_1^i \, a^1$$

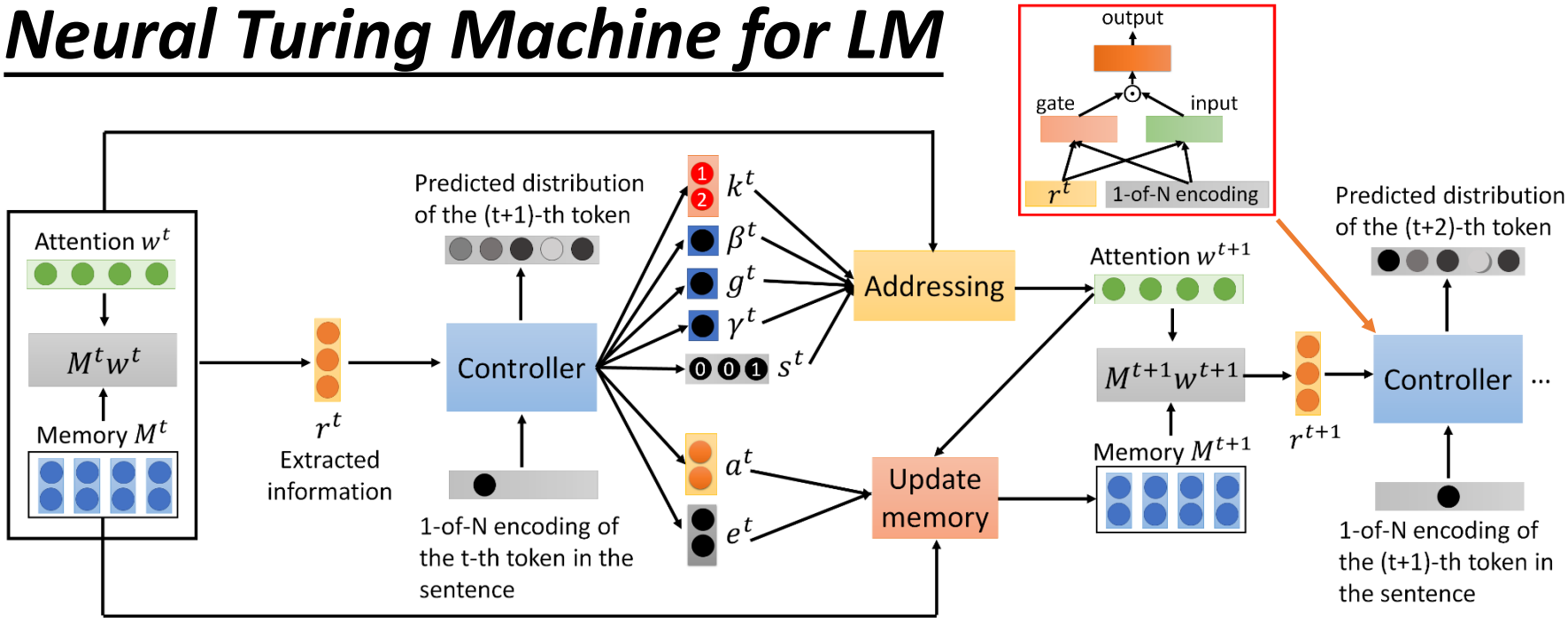
(element-wise)

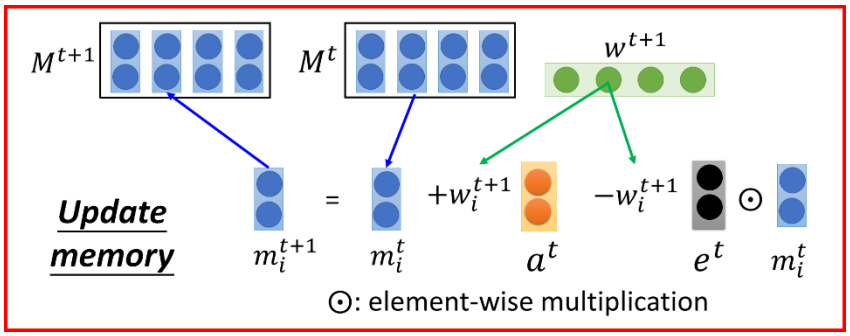


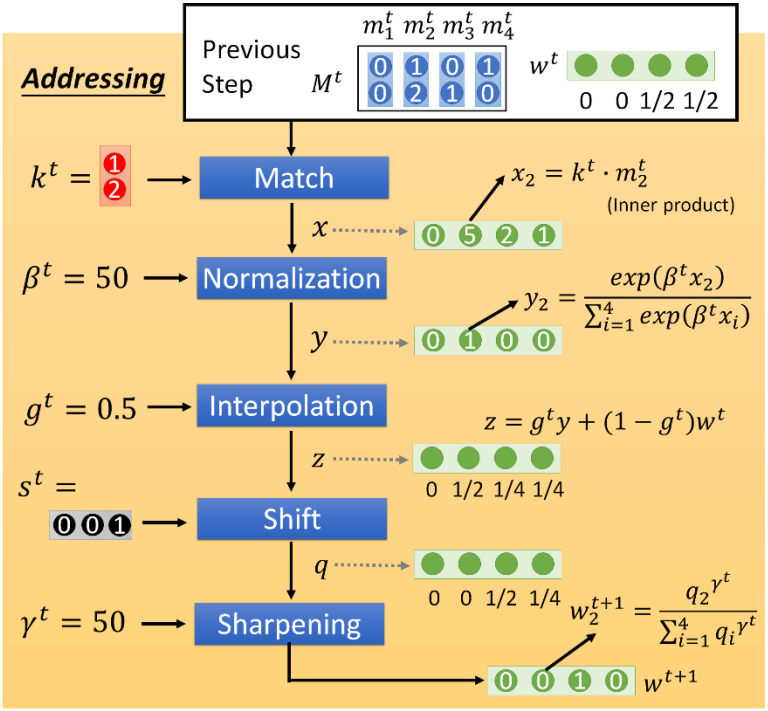
Neural Turing Machine



Neural Turing Machine for LM



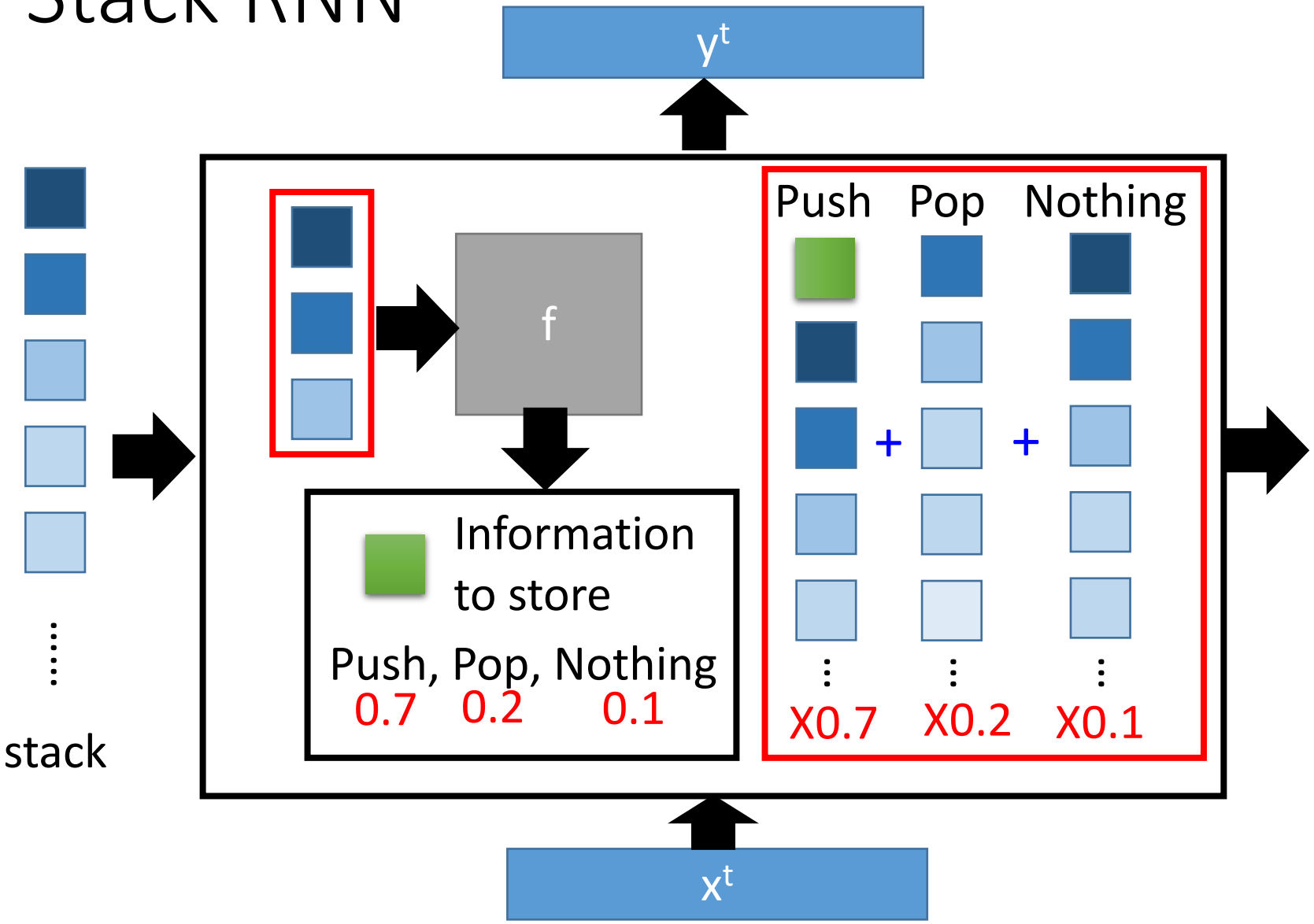




Wei-Jen Ko, Bo-Hsiang Tseng, Hung-yi Lee, "Recurrent Neural Network based Language Modeling with Controllable External Memory", ICASSP, 2017

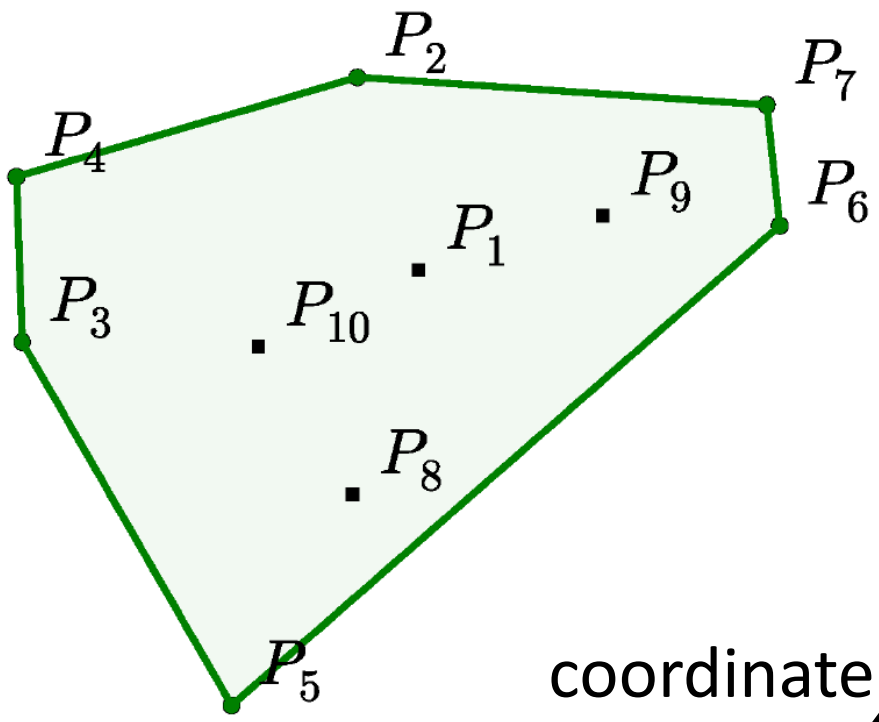
Armand Joulin, Tomas Mikolov, Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets, NIPS, 2015

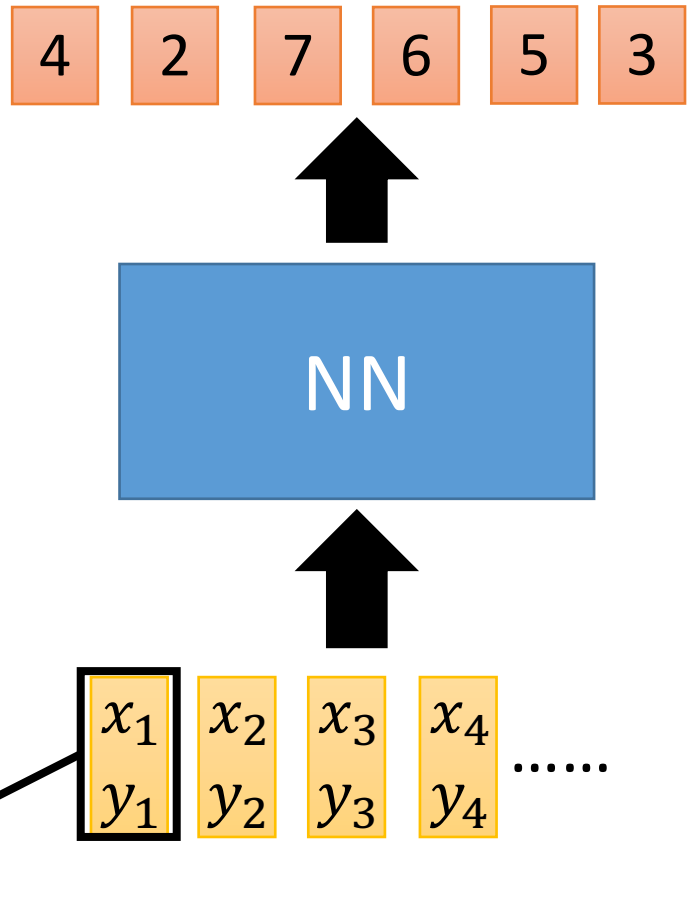
Stack RNN



Pointer Network

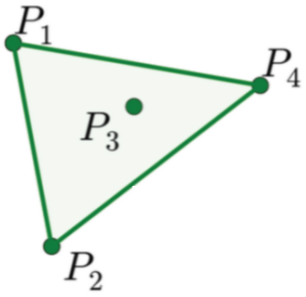
Pointer Network

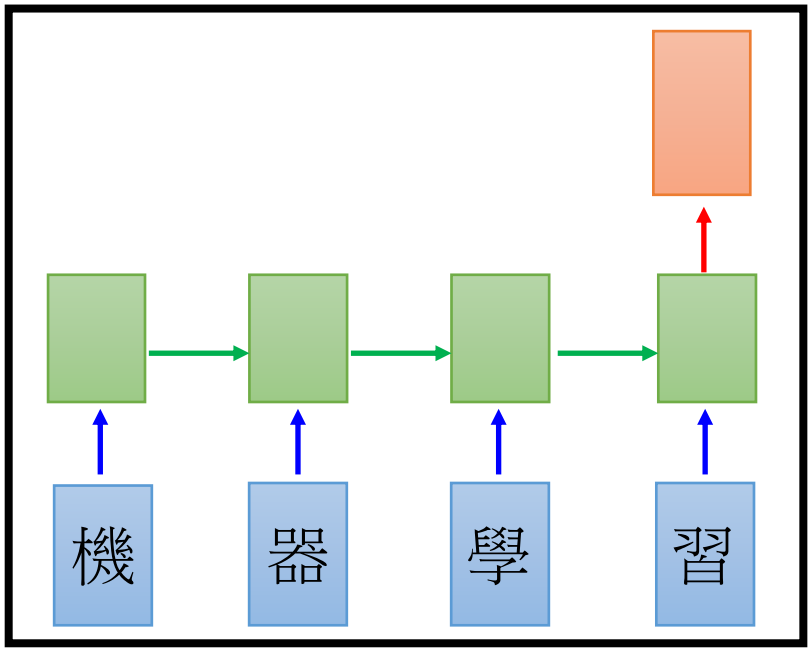




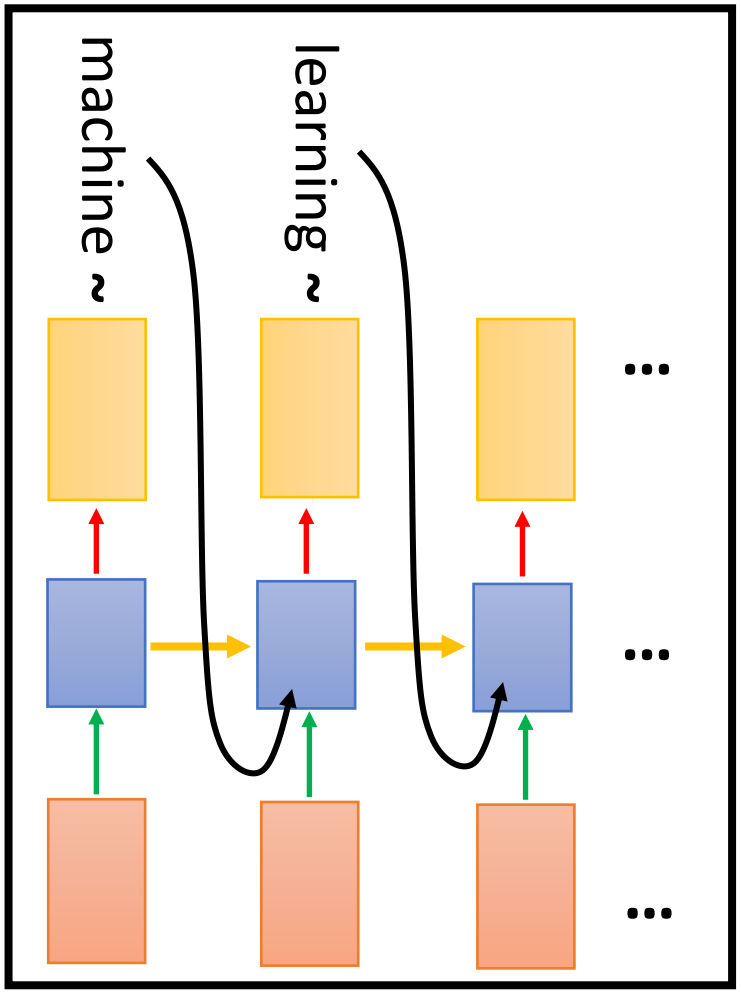
planar convex hulls, computing Delaunay triangulations, and the planar Travelling Salesman Problem

Sequence-to-sequence?





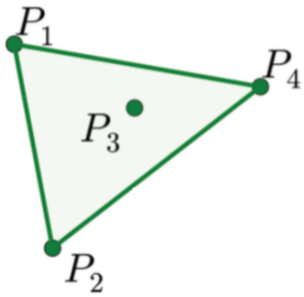
Encoder



Decoder

Pointer Network

x_0 y_0 : END



Attention Weight

key

0.5

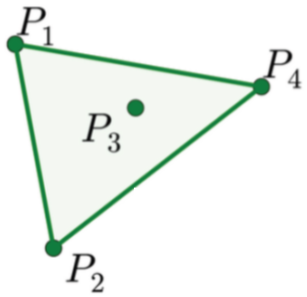
z^0

h^0 h^1 h^2 h^3 h^4

x_0 y_0 | x_1 y_1 | x_2 y_2 | x_3 y_3 | x_4 y_4

Pointer Network

x_0 y_0 : END

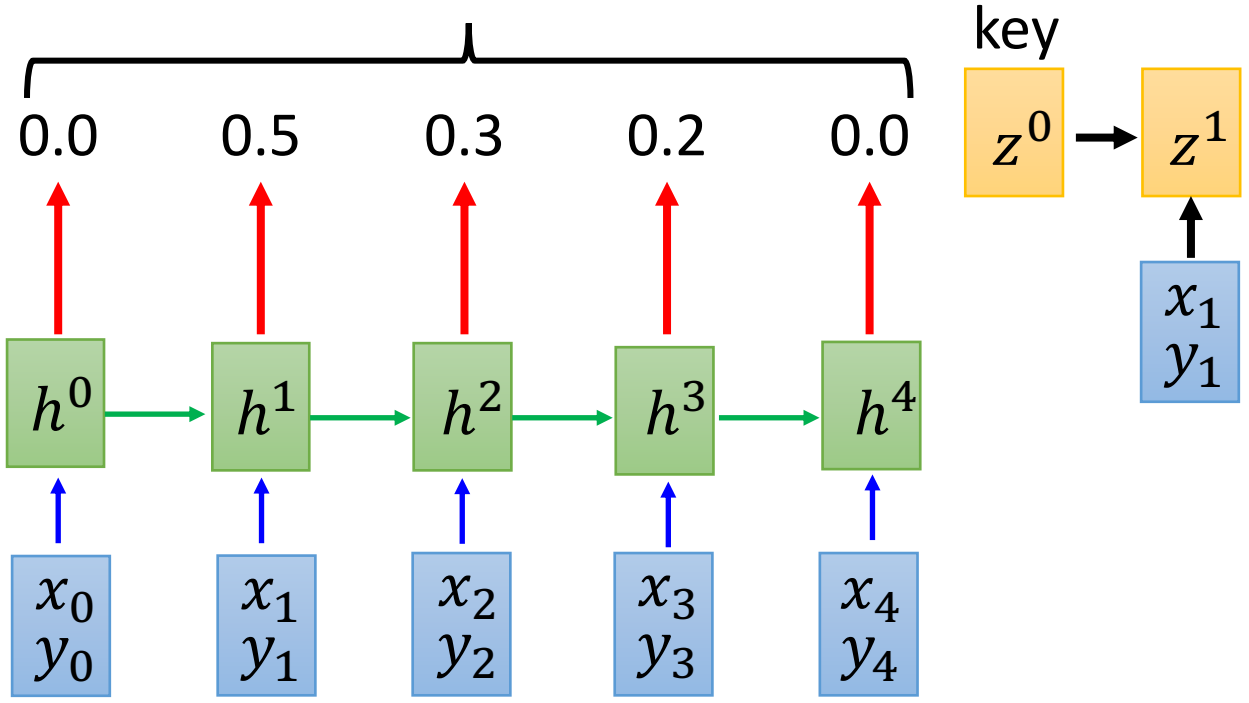


Output: 1

≈

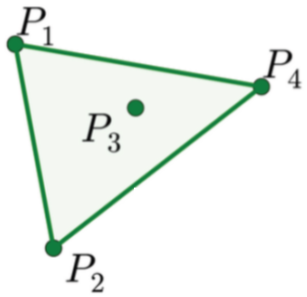
argmax from this distribution

What decoder can output depends on the input.



Pointer Network

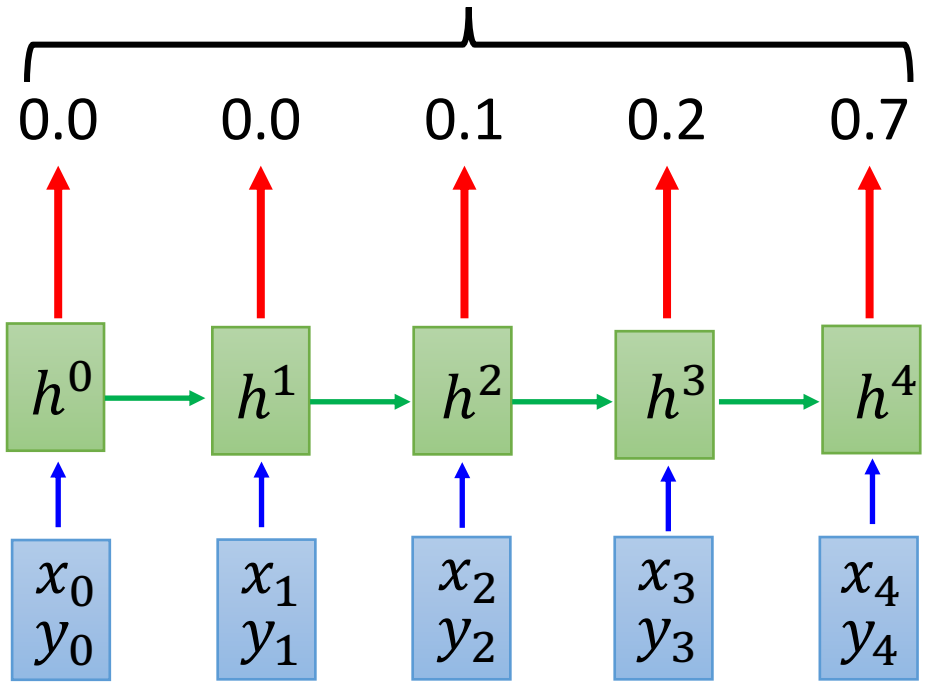
x_0 y_0 : END



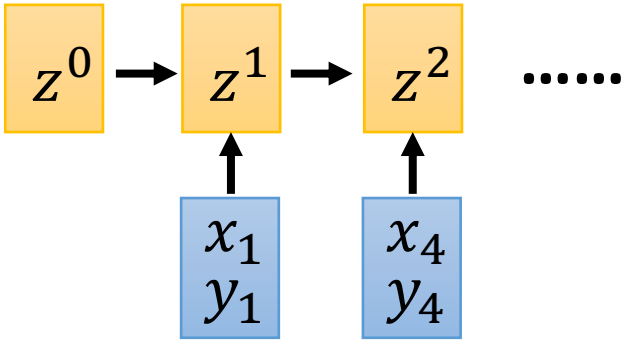
Output: 4

What decoder can output depends on the input.

argmax from this distribution

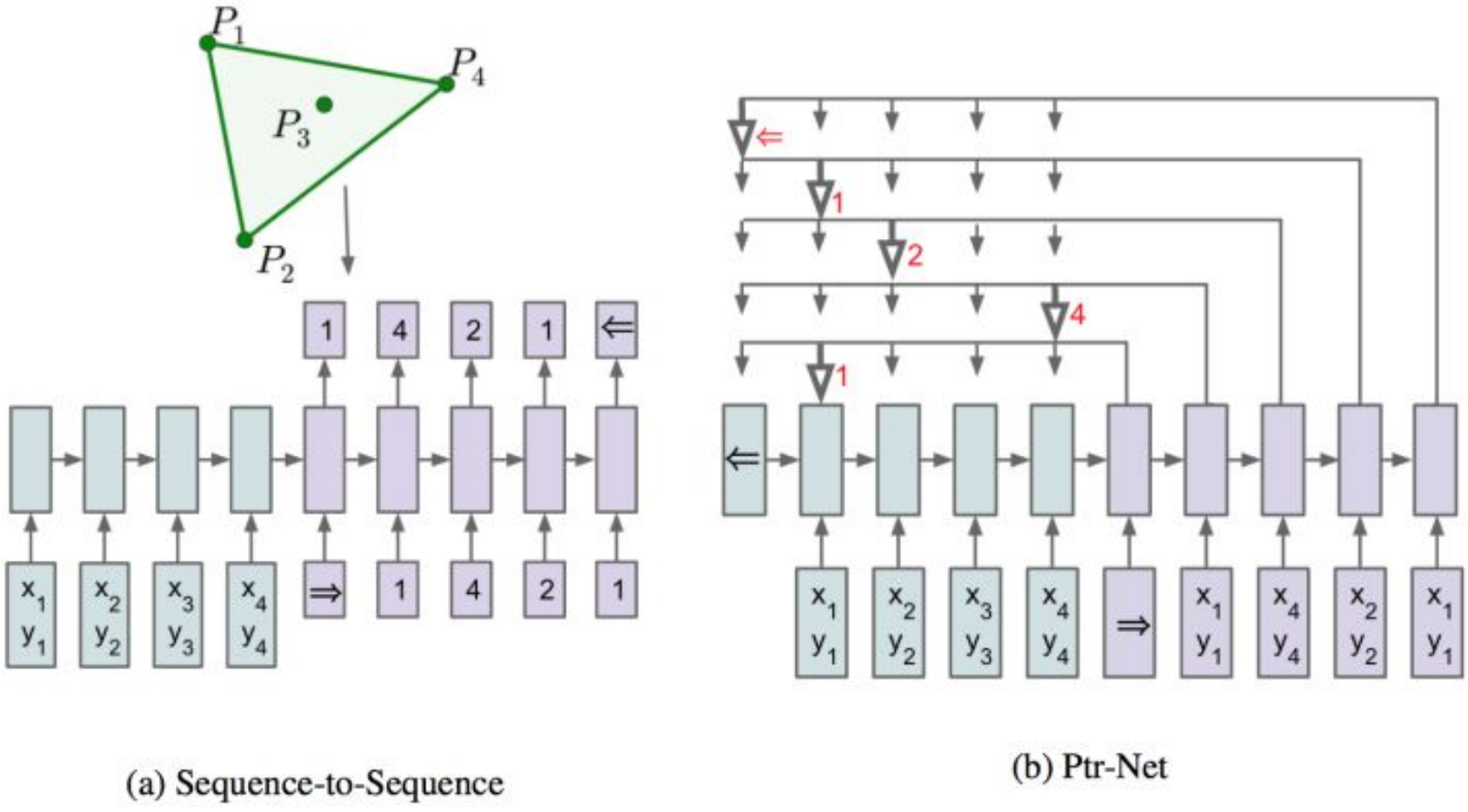


key



The process stops when "END" has the largest attention weights.

Pointer Network

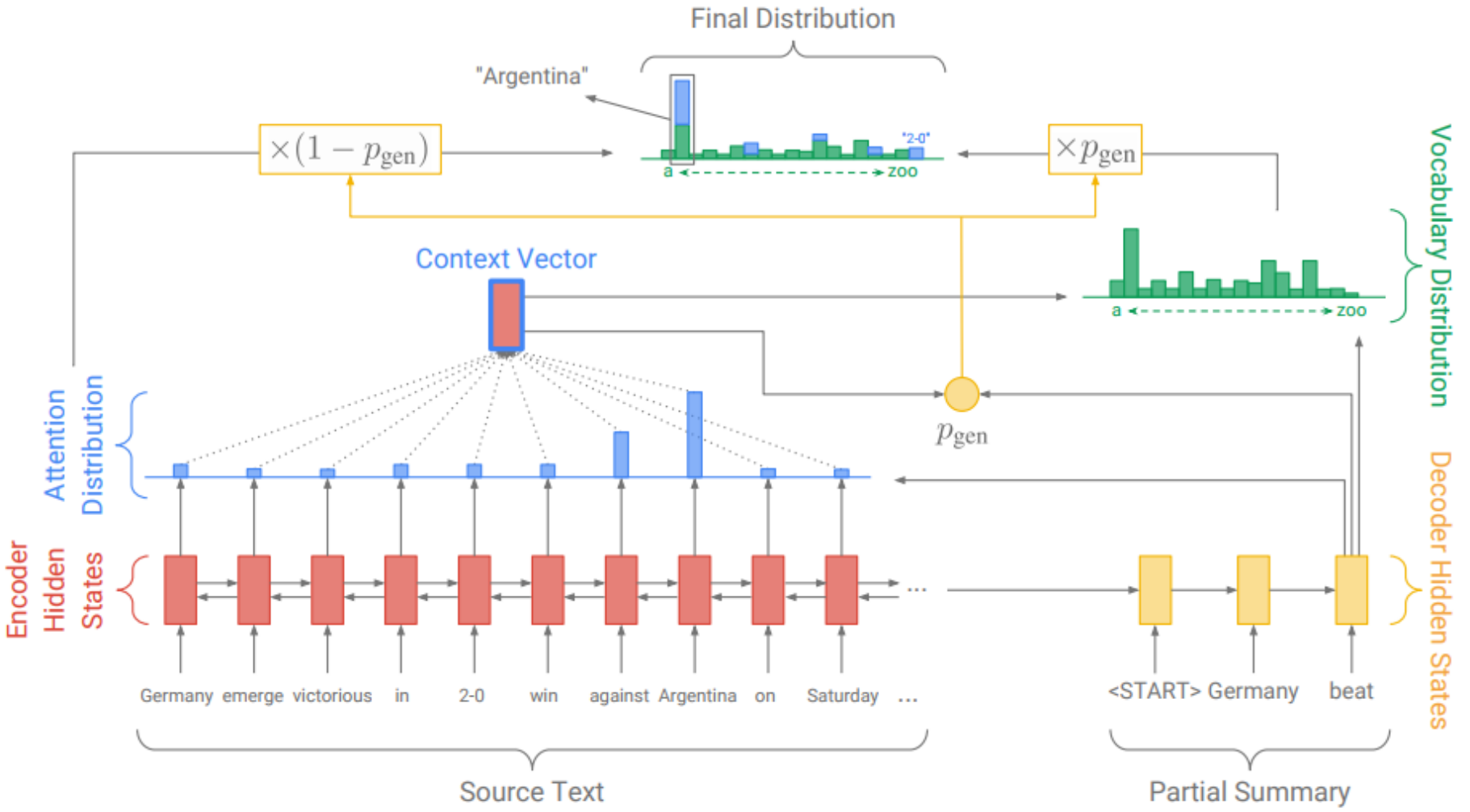


(a) Sequence-to-Sequence

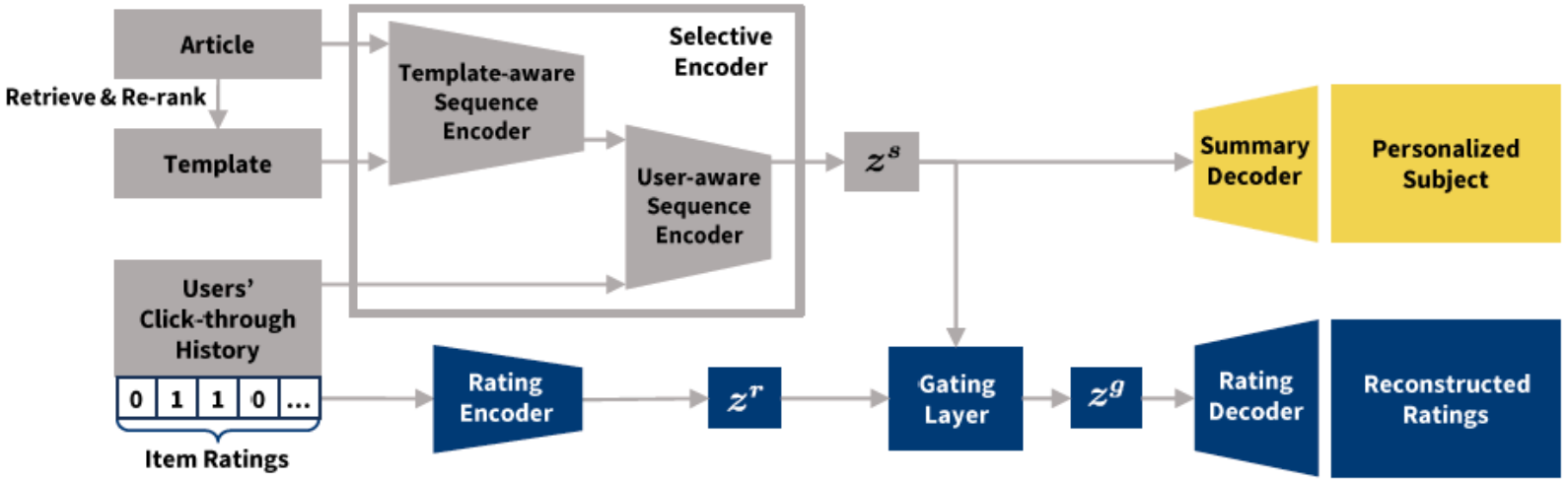
(b) Ptr-Net

https://arxiv.org/abs/1704.04368

Applications - Summarization

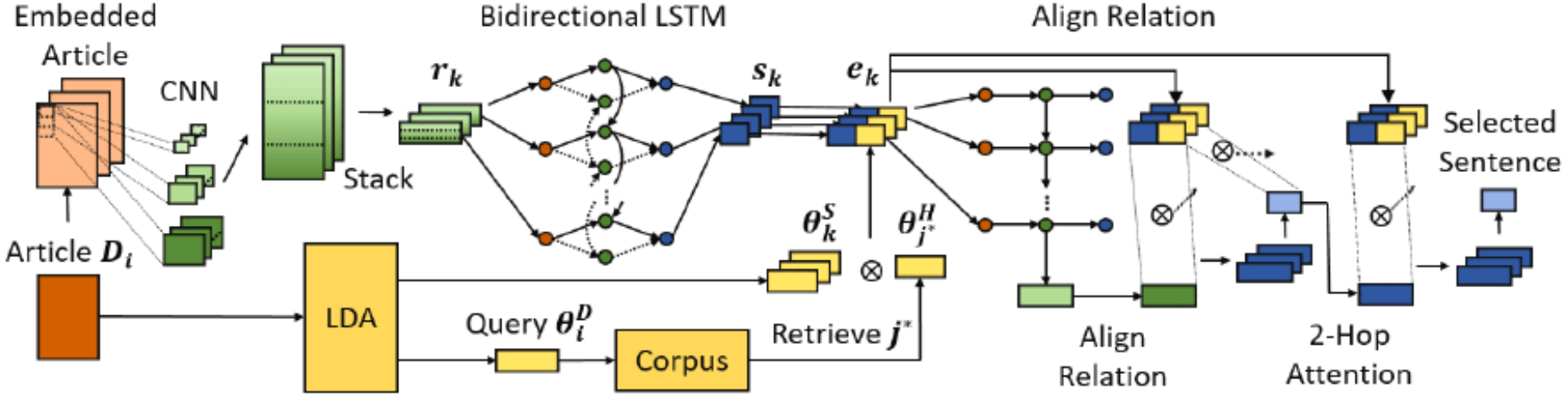


TemPEST: Soft Template-based Personalized EDM Subject Generation Through Collaborative Summarization



Yu-Hsiu Chen, Pin-Yu Chen, Hong-Han Shuai, Wen-Chih Peng, AAAI, 2020

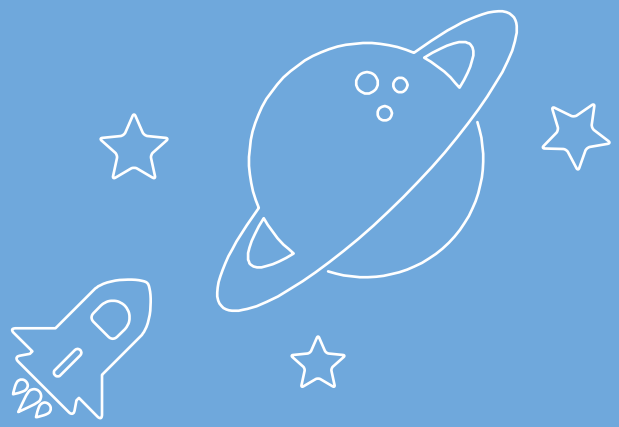
Attractive or Faithful? Popularity-Reinforced Learning for Inspired Headline Generation



Yun-Zhu Song, Hong-Han Shuai, Sung-Lin Yeh, Yi-LunWu, Lun-Wei Ku, Wen-Chih Peng, AAAI 2O2O.

Transformer

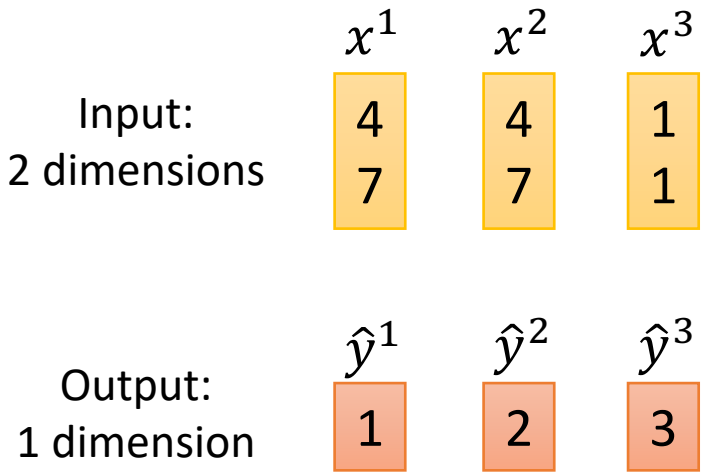
Foundation of BERT

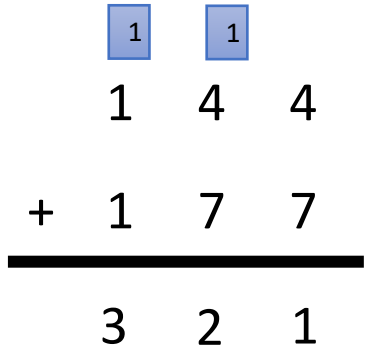


Preliminaries (RNN)

Neural Network with Memory

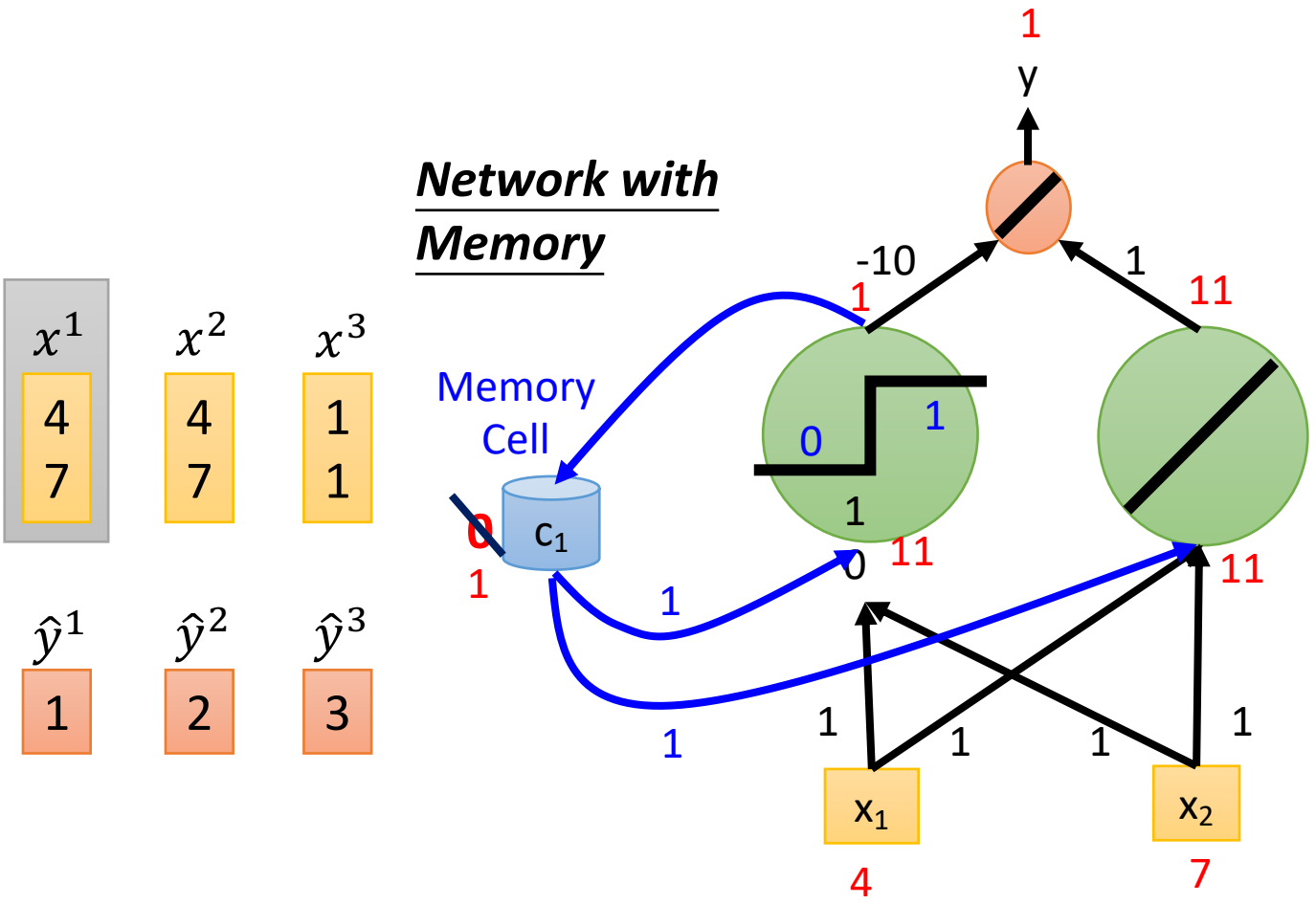
Memory is important



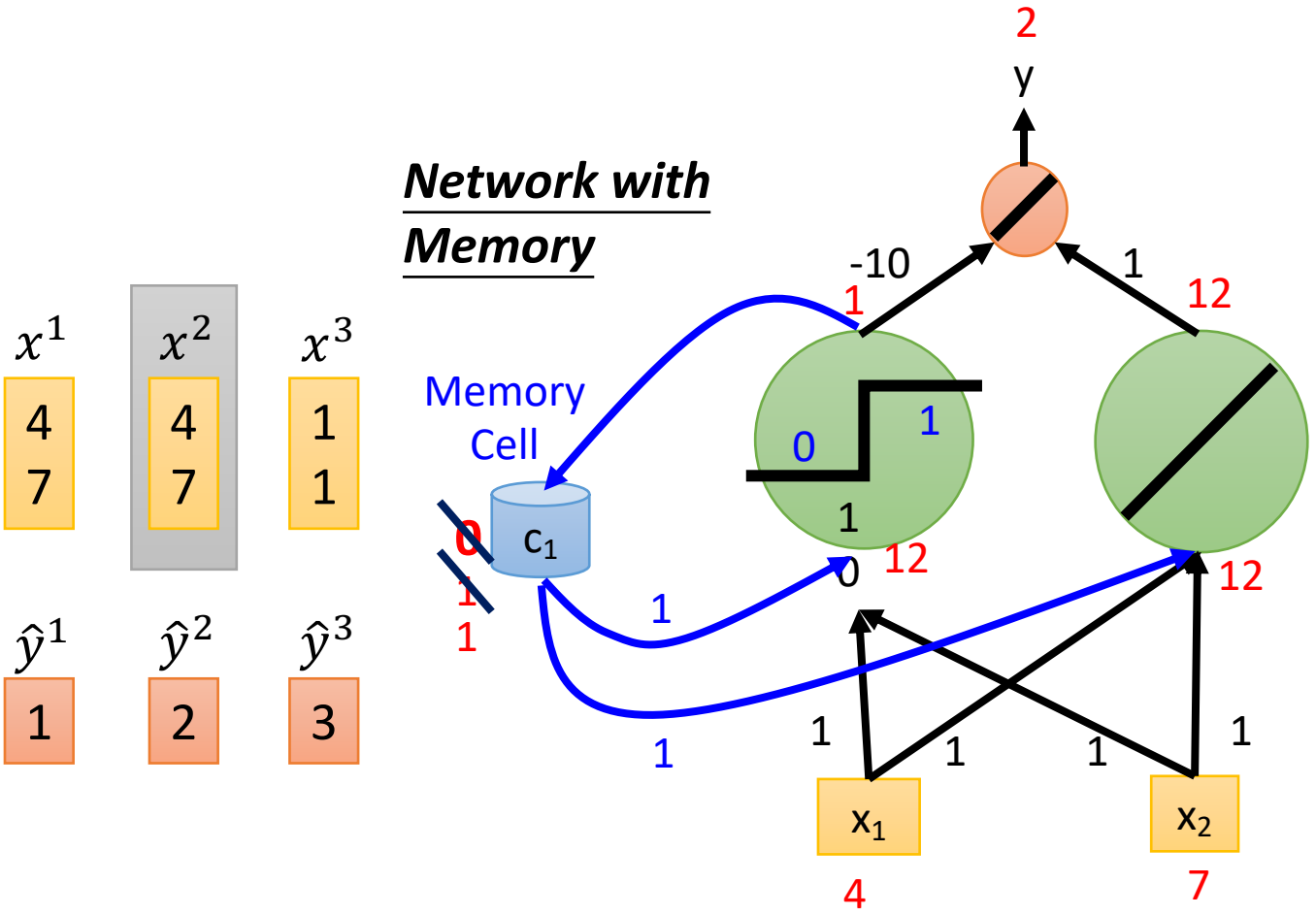


Network needs memory to achieve this

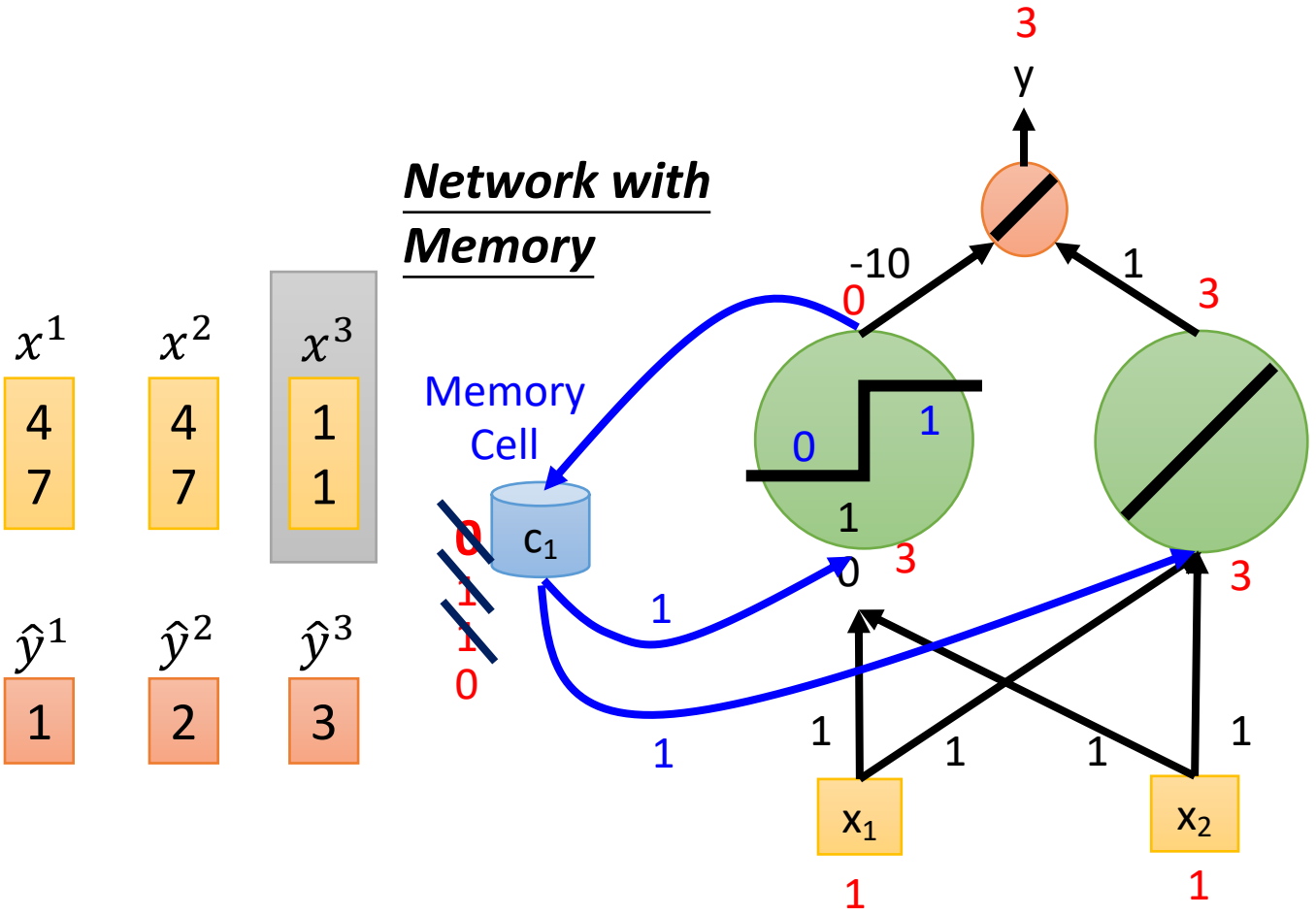
Memory is important



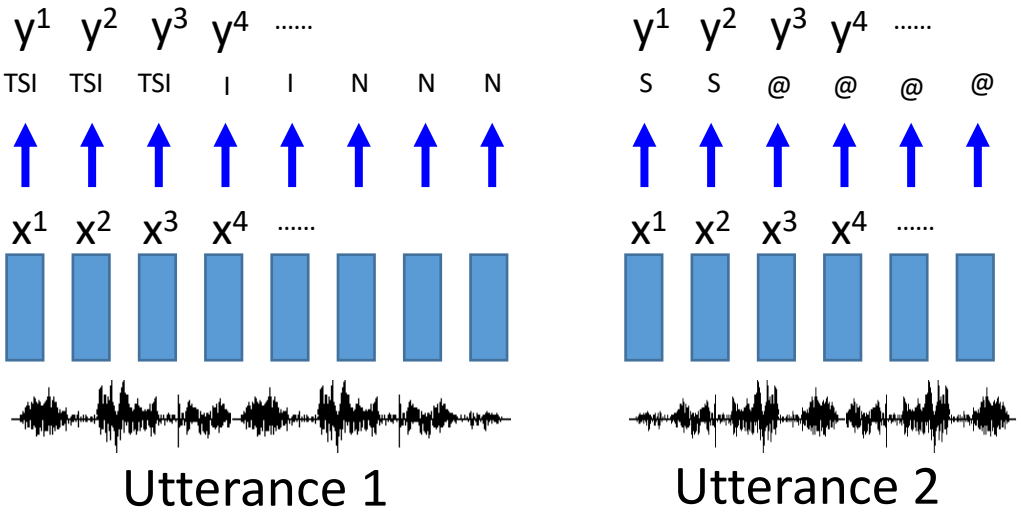
Memory is important



Memory is important

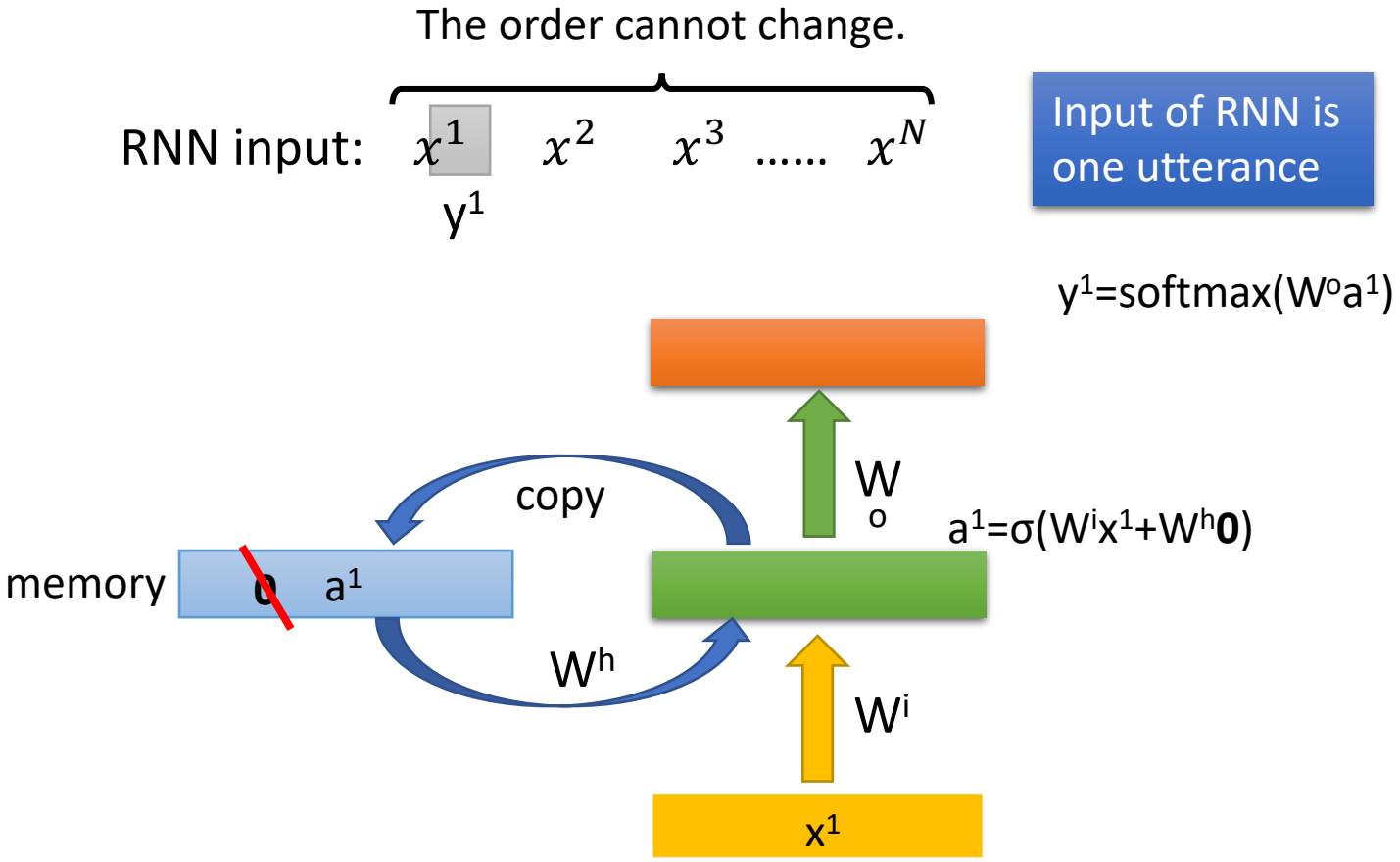


- (Simplified) Speech Recognition

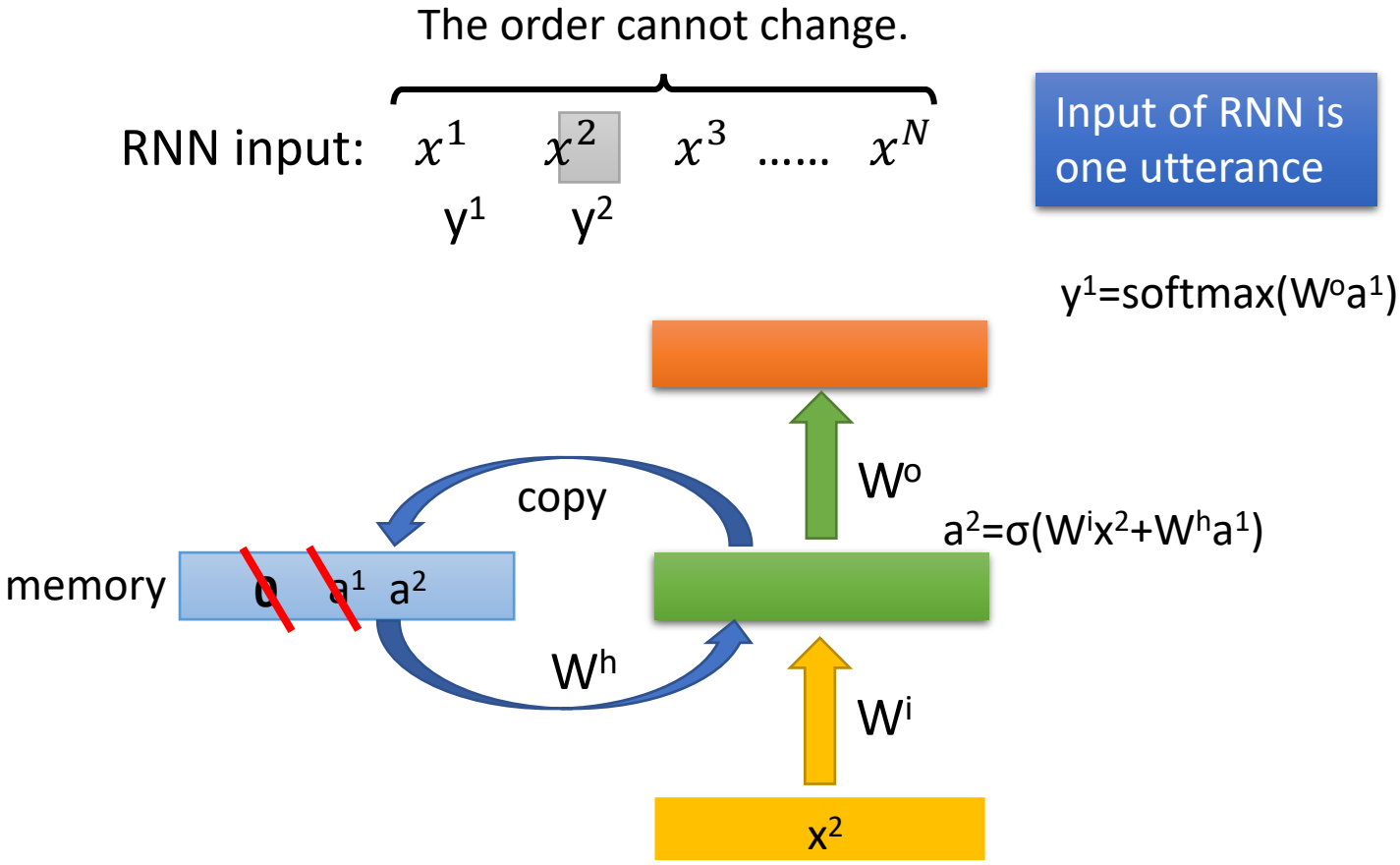


We use DNN. All the frames are considered independently.

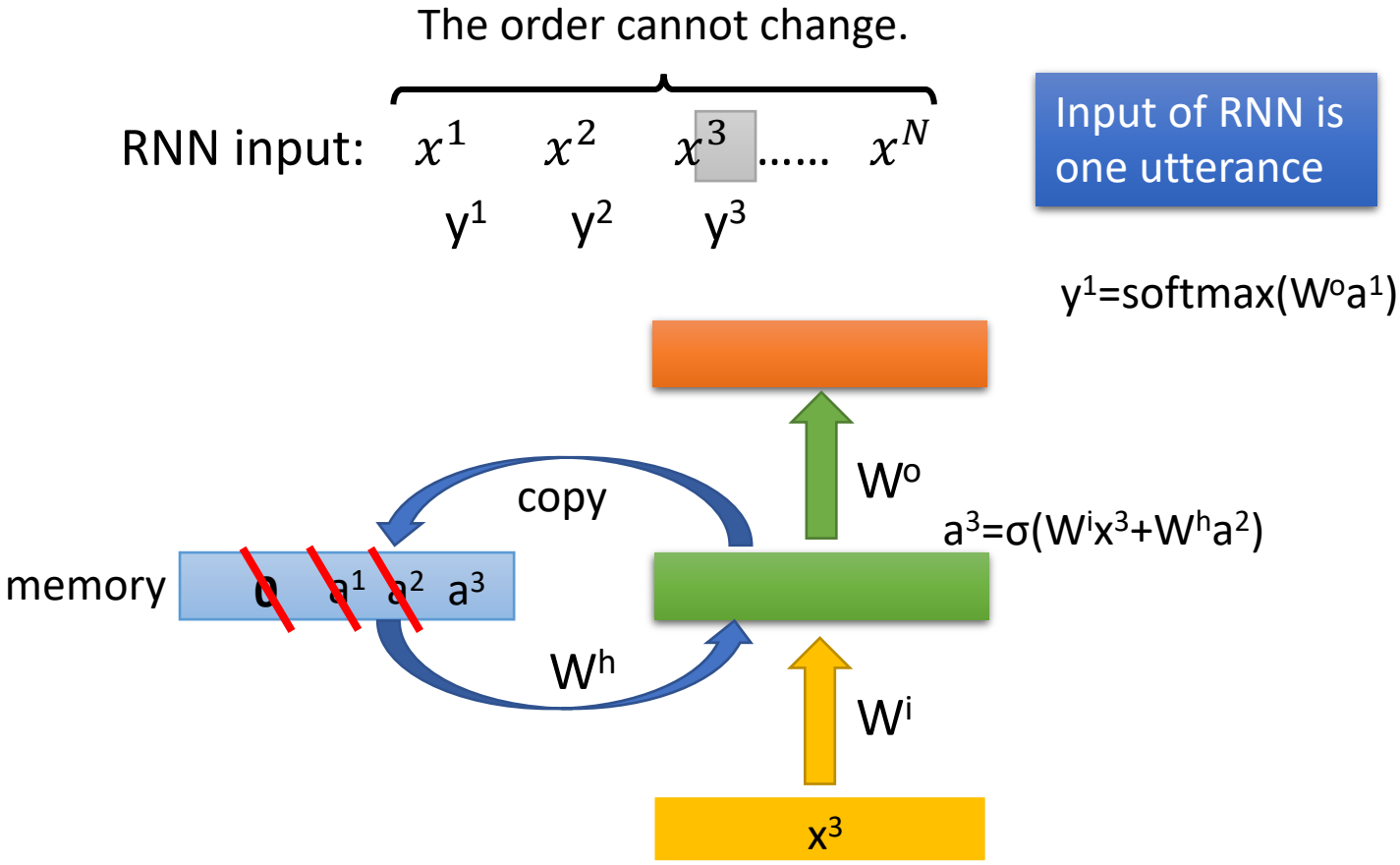
RNN


The order cannot change.
RNN input:
x^1 x^2 x^3 ..... x^N
y1
Input of RNN is one utterance
y1=softmax(Woa1)
copy
W
o
a1=σ(Wix1+Wh0)
memory
0
a1
Wh
Wi
x1

RNN


The order cannot change.
RNN input:
x^1 x^2 x^3 ..... x^N
y1 y2
Input of RNN is one utterance
y1=softmax(Woa1)
Wo
a2=σ(Wix2+Wha1)
copy
memory
0 a1 a2
Wh
Wi
x2

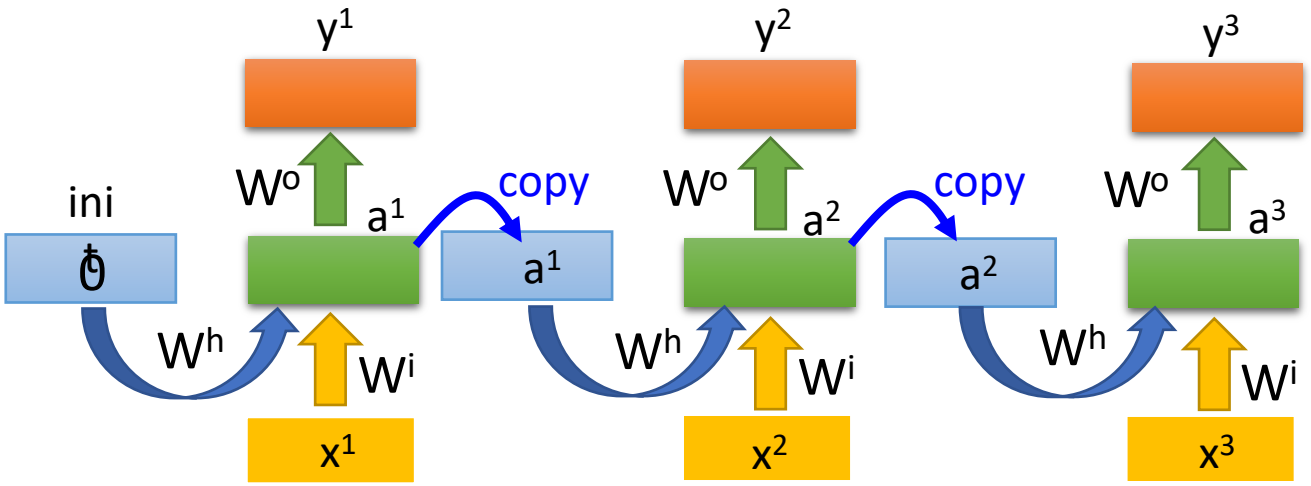
RNN


The order cannot change.
RNN input:
x^1 x^2 x^3 ..... x^N
y^1 y^2 y^3
Input of RNN is one utterance
$y^1=\text{softmax}(W^o a^1)$
W^o
$a^3=\sigma(W^i x^3+W^h a^2)$
copy
memory
0 a^1 a^2 a^3
W^h
W^i
x^3

RNN

Input of RNN is one utterance

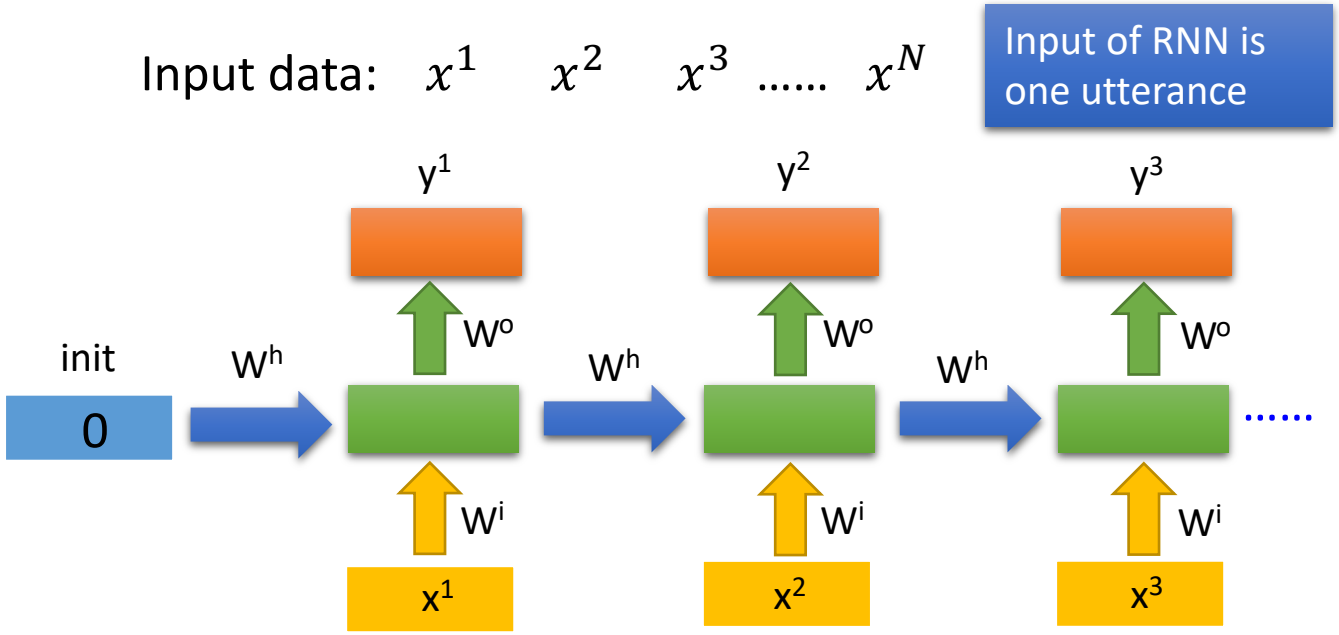
Input data: x^1 x^2 x^3 …… x^N



The same network is used again and again.

Output y^i depends on x^1, x^2, …… x^i

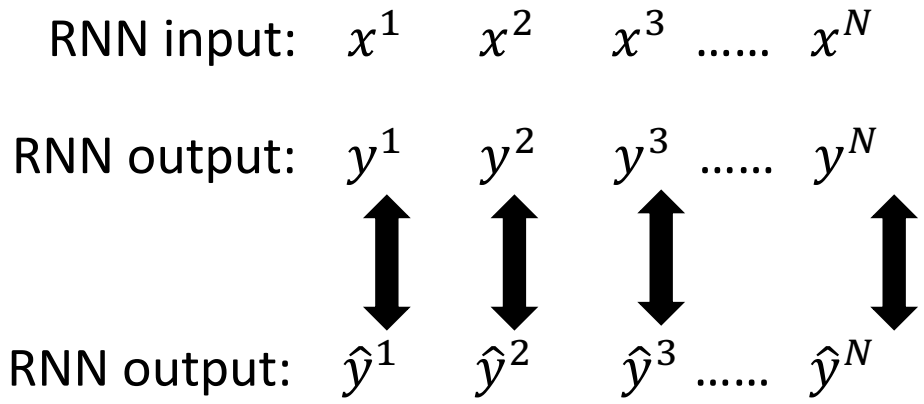
RNN



The same network is used again and again.

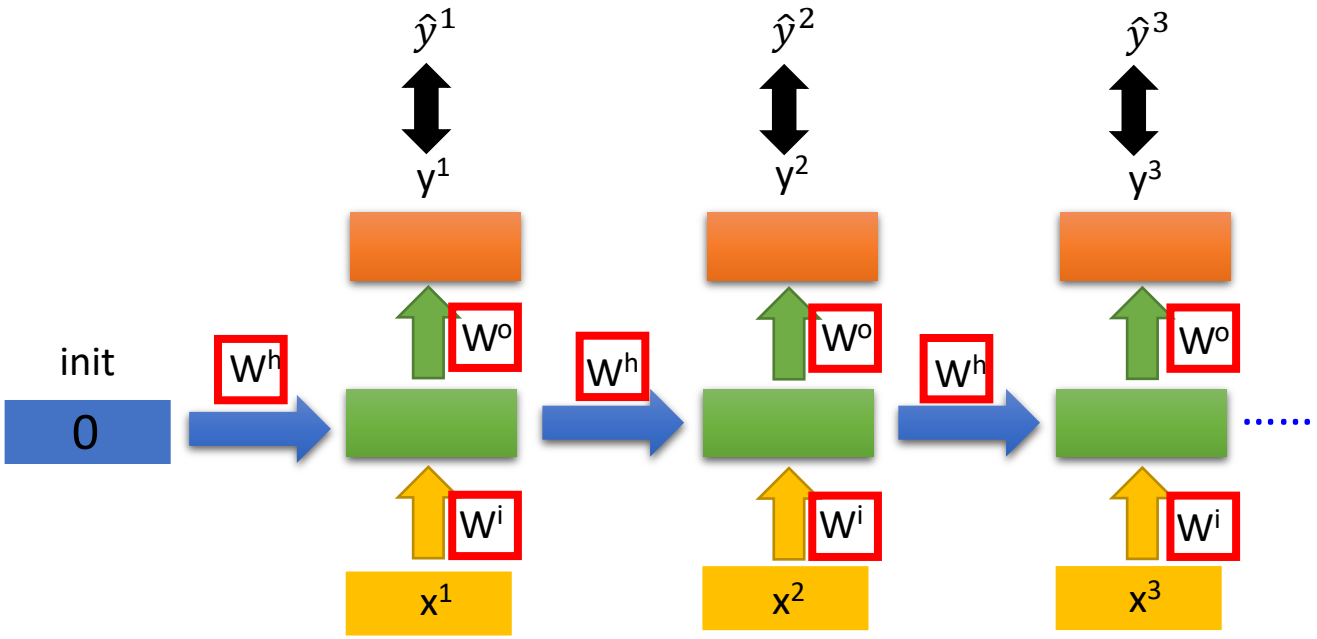
Output y^i depends on x^1, x^2, ... x^i

Cost



$$C = \frac{1}{2}\sum_{n=1}^{N} \|y^n - \hat{y}^n\|^2 \qquad C = \frac{1}{2}\sum_{n=1}^{N} -log y_{r^n}^n$$

Training

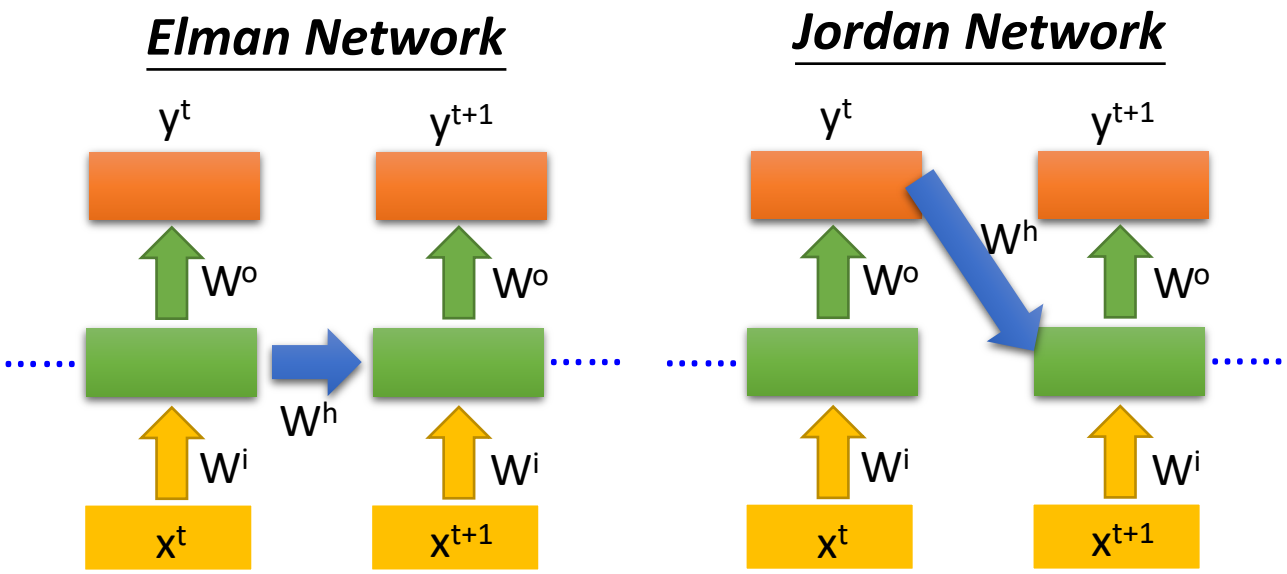


w is an element in W^h, W^i or W^o ➡ $w \leftarrow w - \eta \partial C / \partial w$

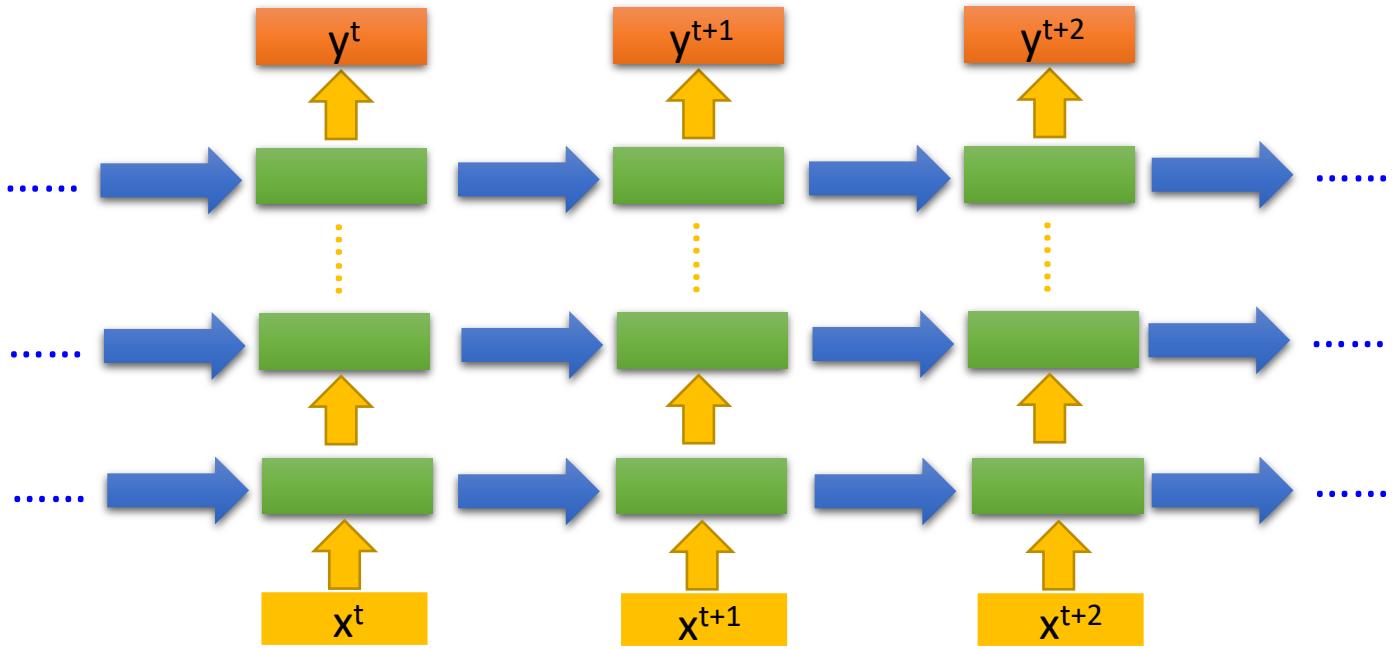
➡ Backpropagation through time (BPTT)

RNN Training is very difficult in practice.

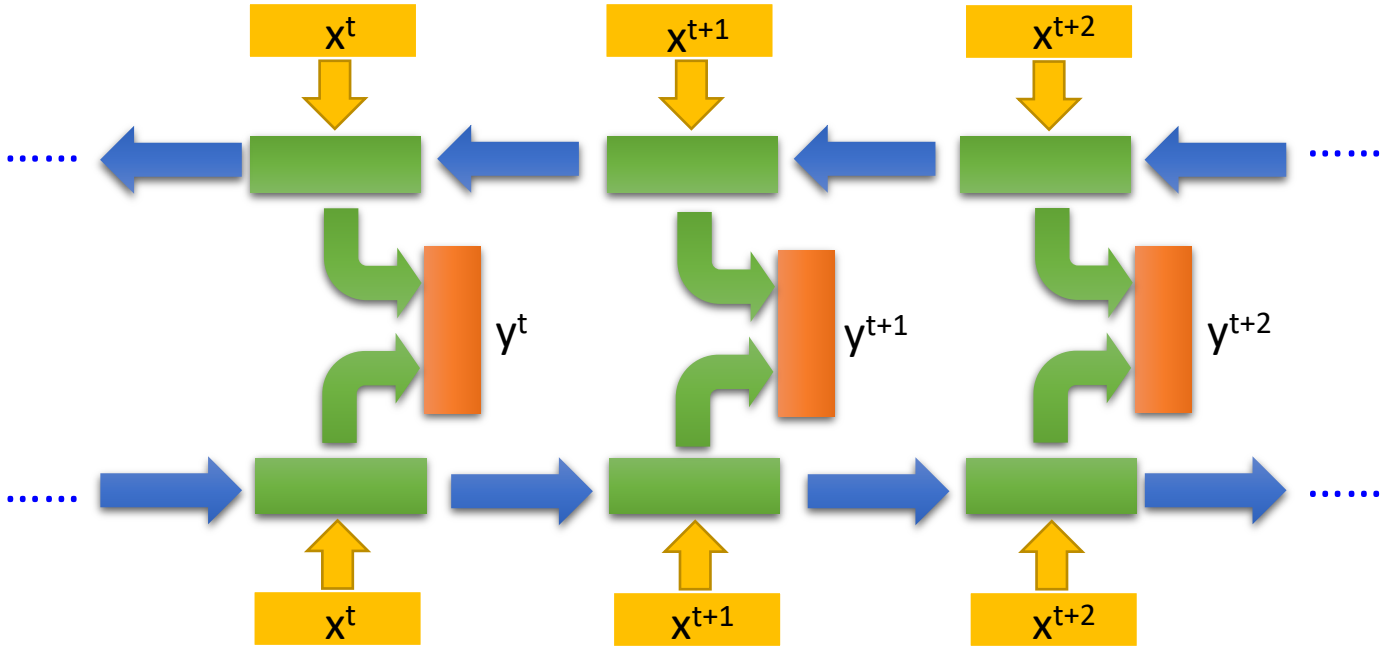
Elman Network & Jordan Network



Deep RNN



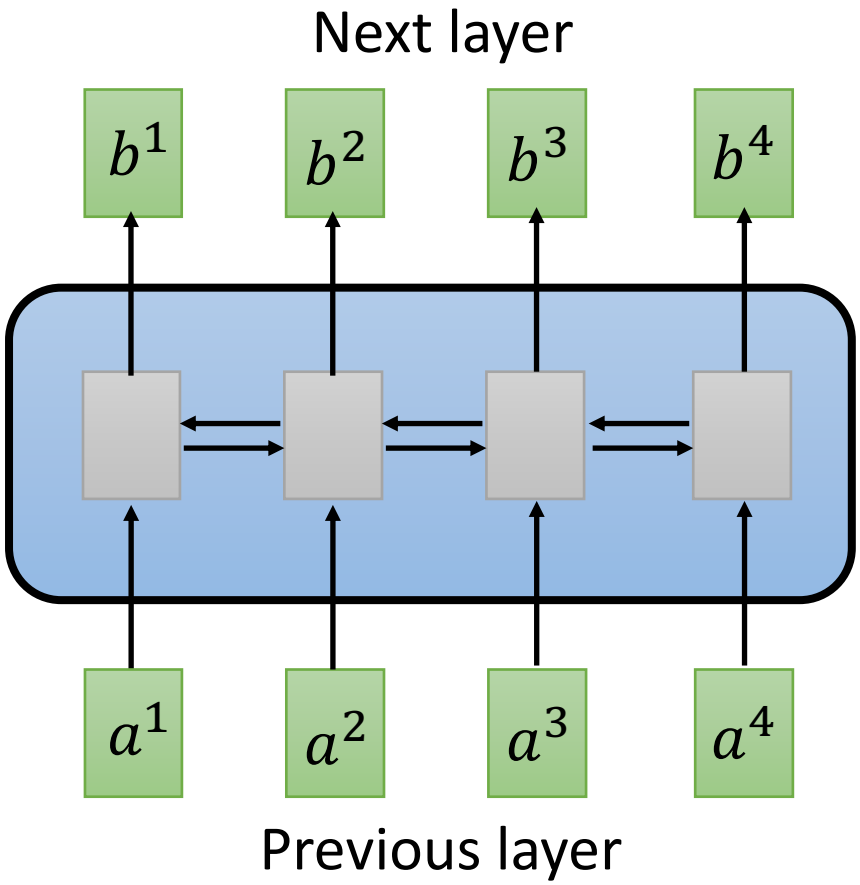
Bidirectional RNN



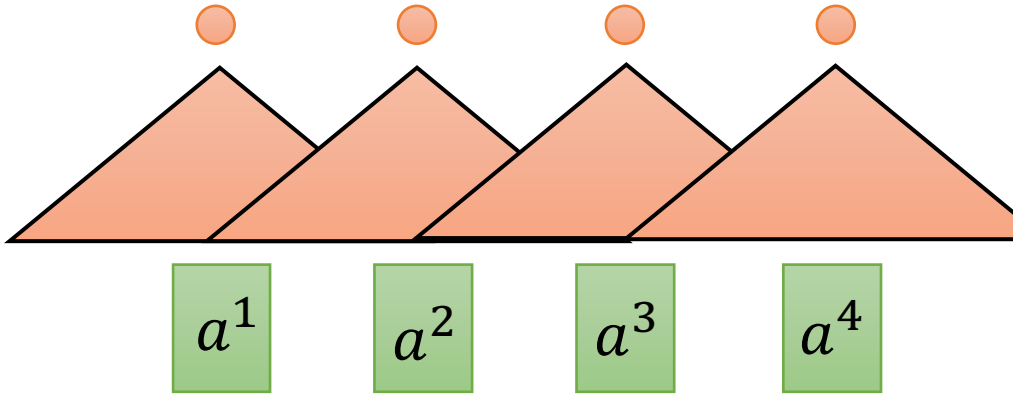
下列文句何者不是倒裝句型？

- (A)惟兄嫂是依
- (B)白雪紛紛何所似
- (C)撒鹽空中差可擬
- (D)不患人之不己知

Sequence

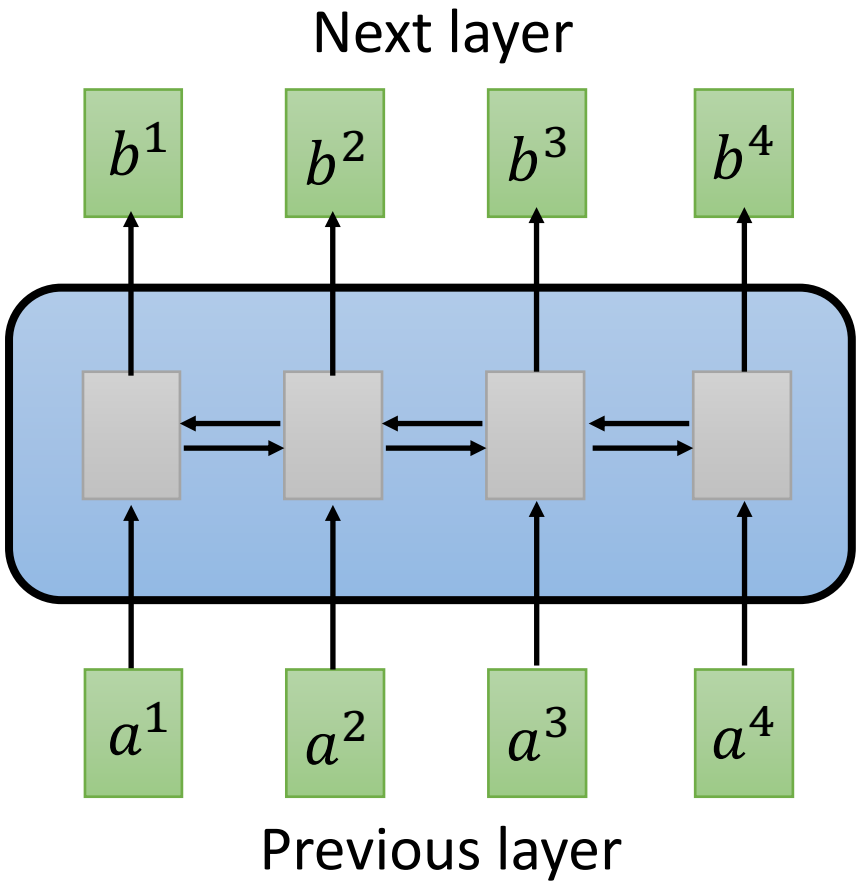


Hard to parallel !



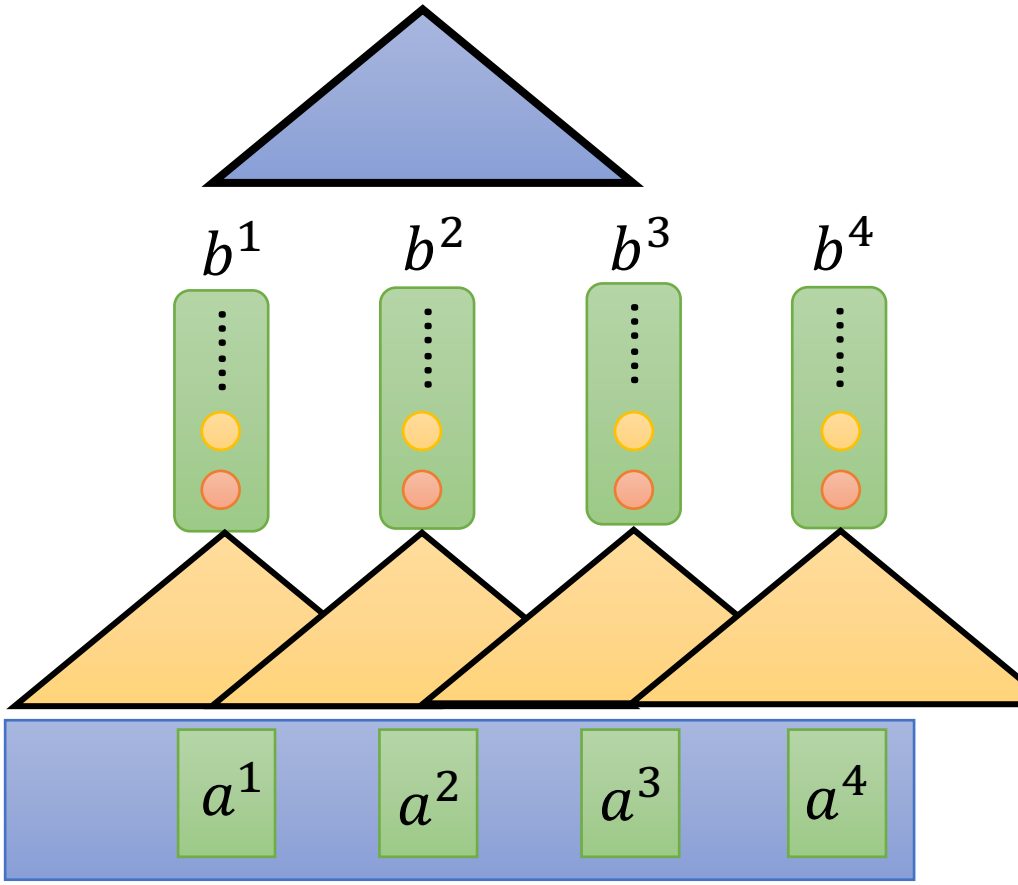
Using CNN to replace RNN

Sequence



Hard to parallel

Filters in higher layer can consider longer sequence

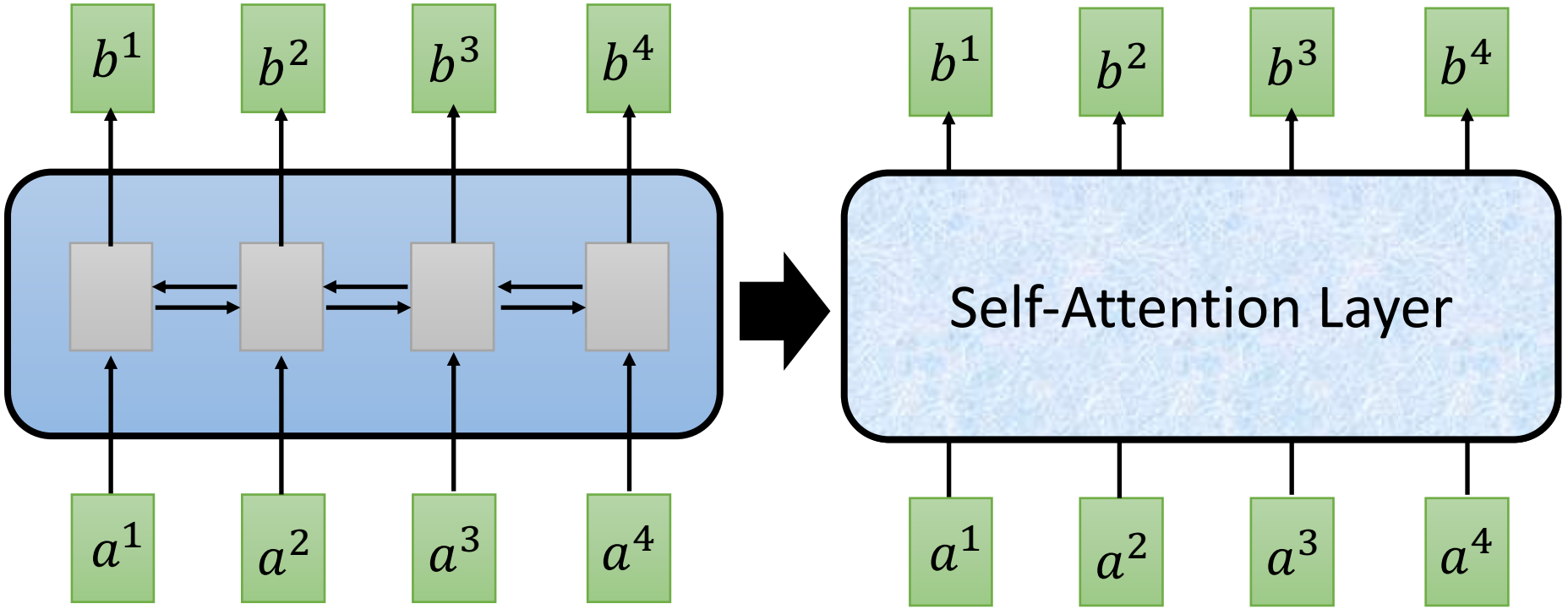


Using CNN to replace RNN (CNN can parallel)

Self-Attention

b^i is obtained based on the whole input sequence.

b^1, b^2, b^3, b^4 can be parallelly computed.



You can try to replace any thing that has been done by RNN with self-attention.

Self-attention

https://arxiv.org/abs/1706.03762

q: query (to match others)

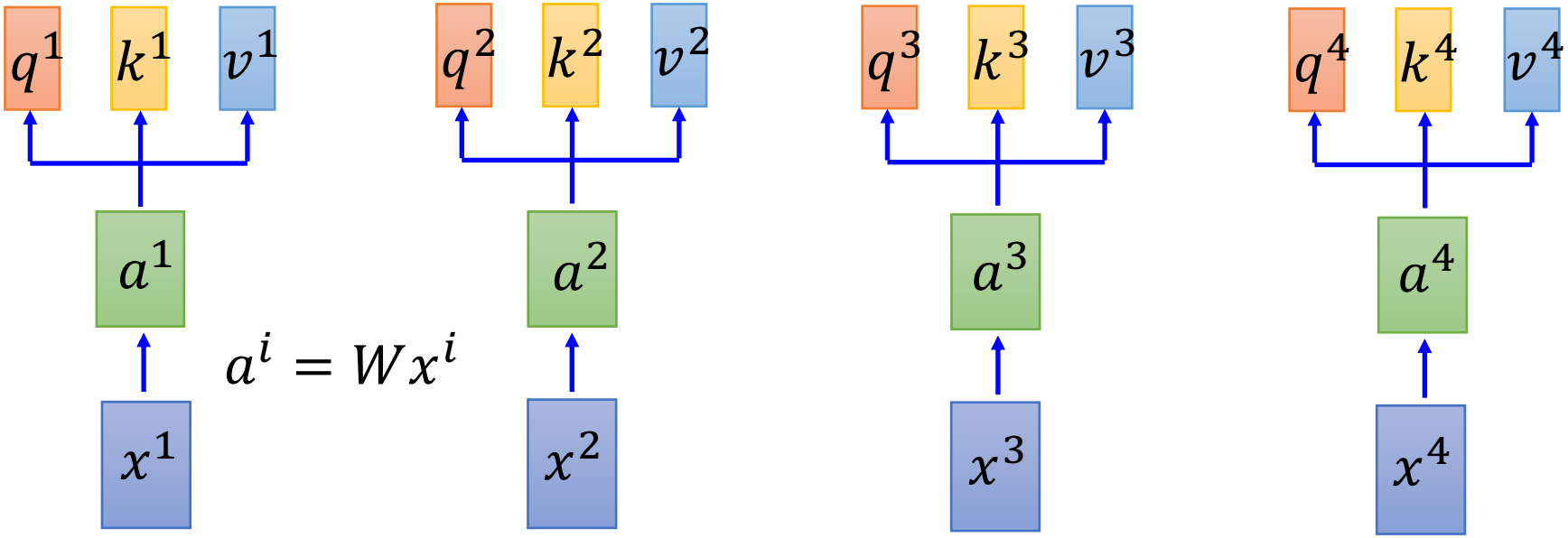
$$q^i = W^q a^i$$

k: key (to be matched)

$$k^i = W^k a^i$$

v: information to be extracted

$$v^i = W^v a^i$$

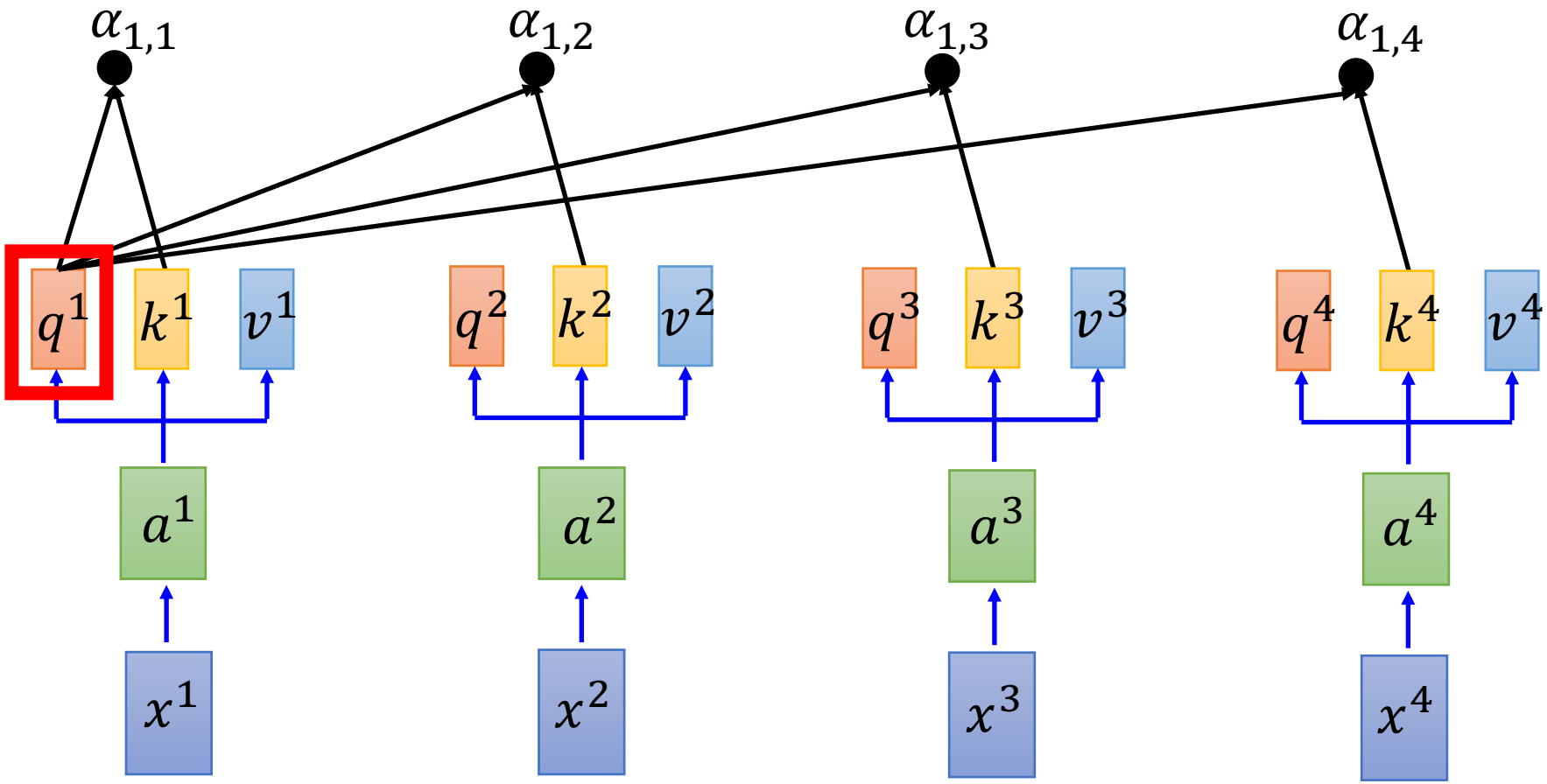


Self-attention

拿每個 query q 去對每個 key k 做 attention

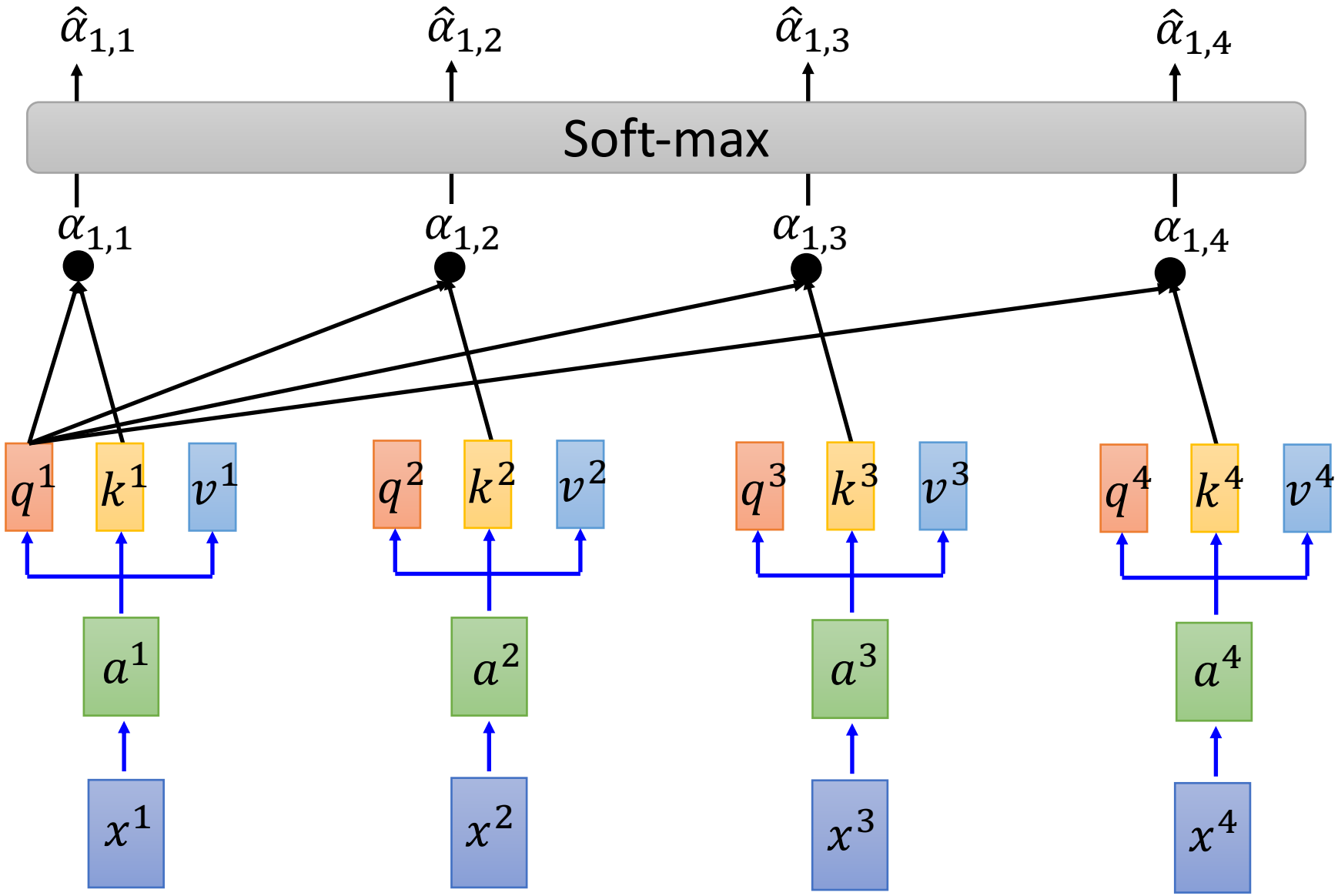
d is the dim of q and k

Scaled Dot-Product Attention: $\alpha_{1,i} = \underbrace{q^1 \cdot k^i}_{\text{dot product}} / \sqrt{d}$



Self-attention

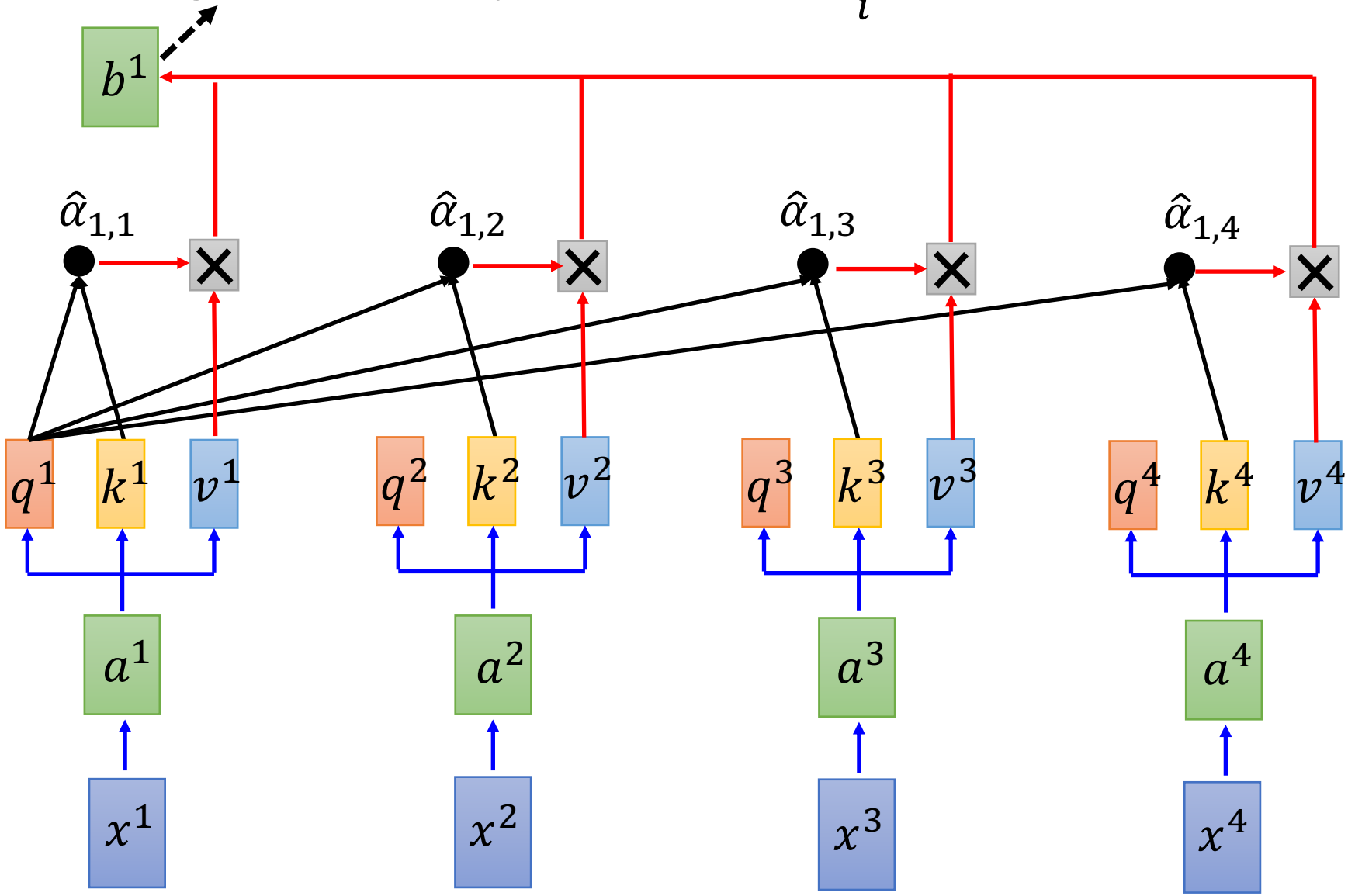
$$\hat{\alpha}_{1,i} = exp(\alpha_{1,i}) / \sum_j exp(\alpha_{1,j})$$



Self-attention

Considering the whole sequence

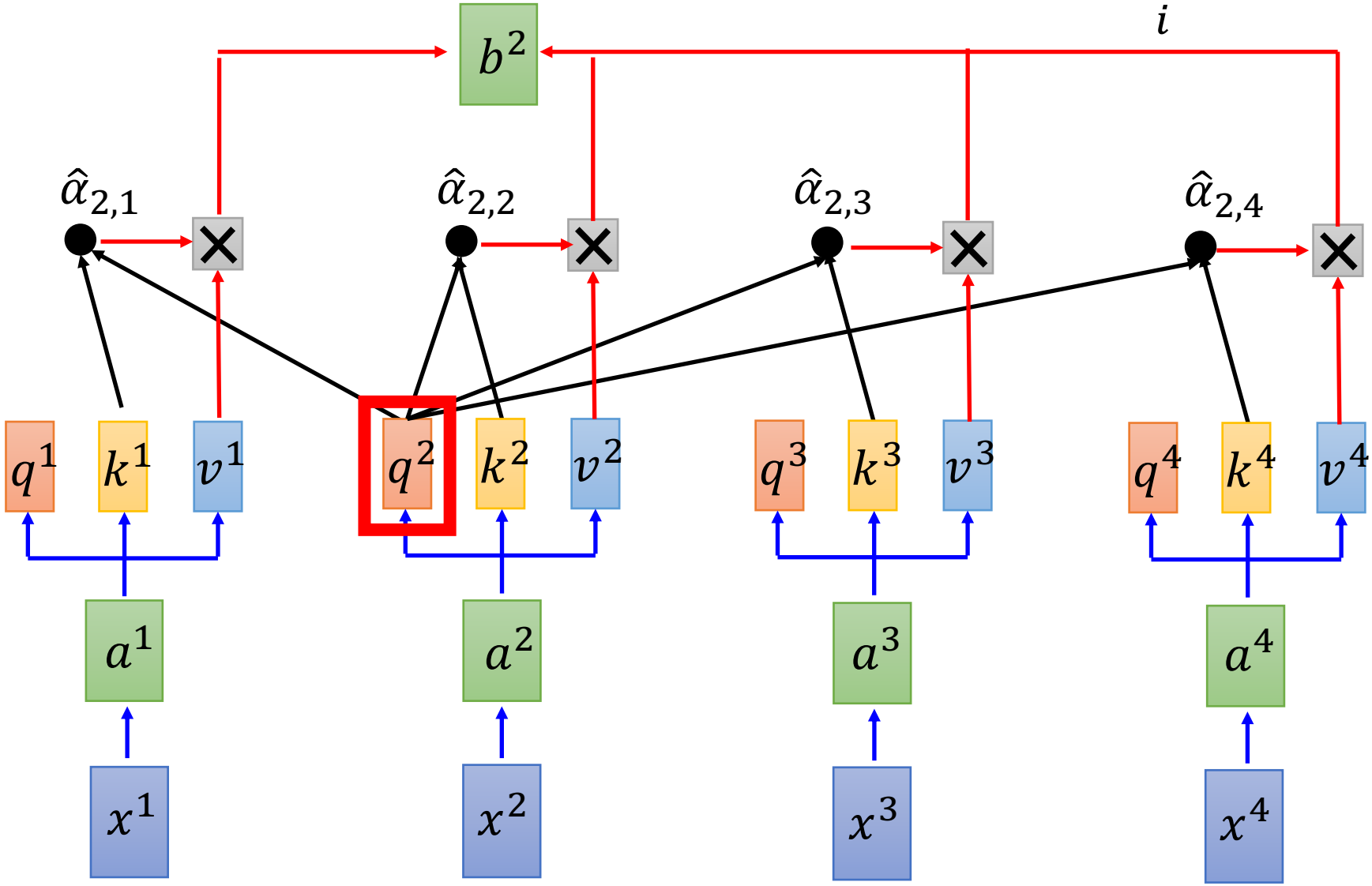
$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$



Self-attention

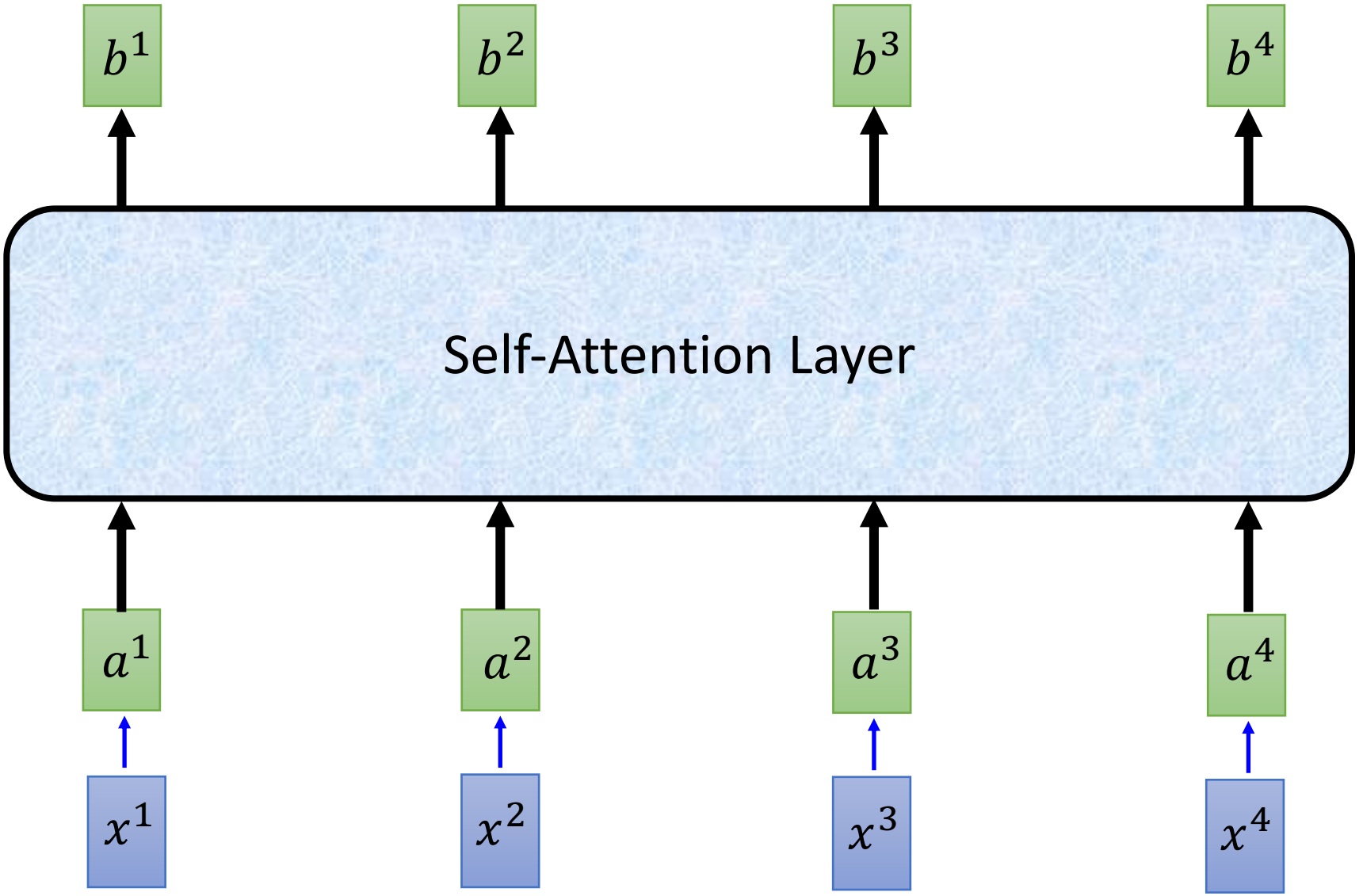
拿每個 query q 去對每個 key k 做 attention

$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



Self-attention

b^1, b^2, b^3, b^4 can be parallelly computed.



Self-attention

$$q^i = W^q a^i$$

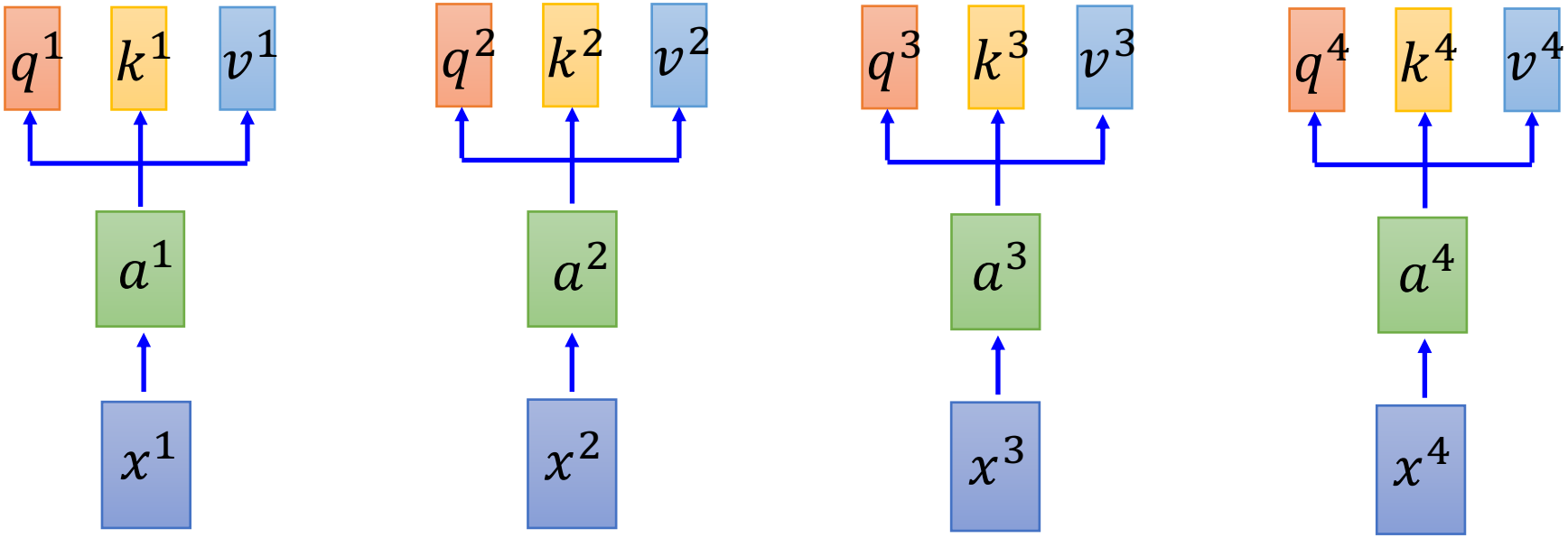
$$k^i = W^k a^i$$

$$v^i = W^v a^i$$

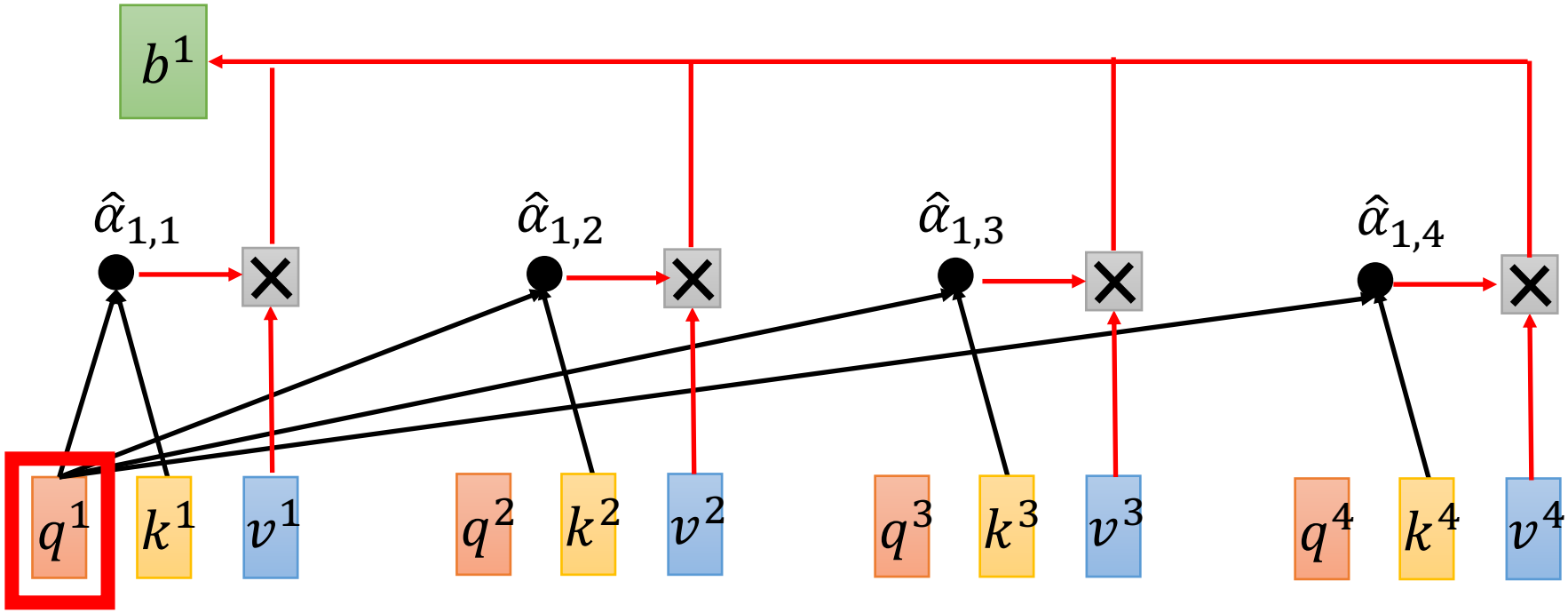
$$\underbrace{q^1\ q^2\ q^3\ q^4}_{Q} = W^q \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$

$$\underbrace{k^1\ k^2\ k^3\ k^4}_{K} = W^k \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$

$$\underbrace{v^1\ v^2\ v^3\ v^4}_{V} = W^v \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$



Self-attention



$\alpha_{1,1} = k^1 \; q^1$ $\alpha_{1,2} = k^2 \; q^1$

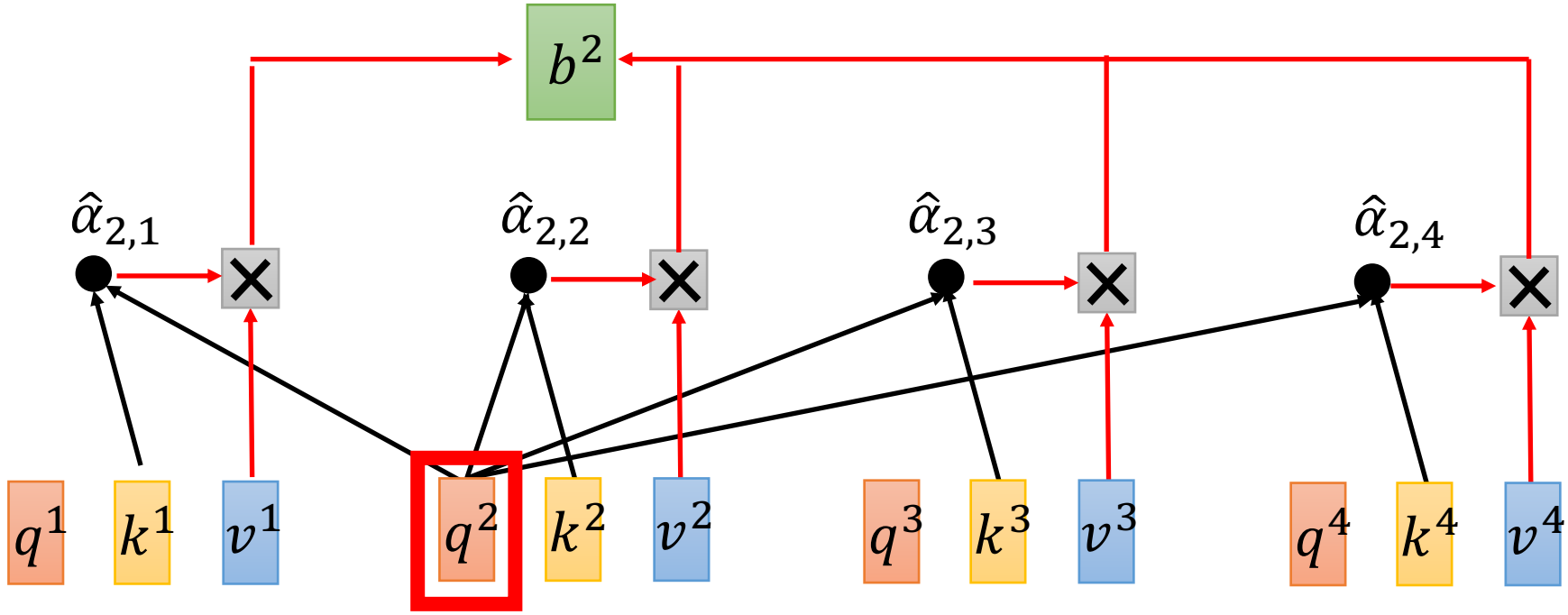
$\alpha_{1,3} = k^3 \; q^1$ $\alpha_{1,4} = k^4 \; q^1$

$$\begin{bmatrix} \alpha_{1,1} \\ \alpha_{1,2} \\ \alpha_{1,3} \\ \alpha_{1,4} \end{bmatrix} = \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} q^1$$

(ignore $\sqrt{d}$ for simplicity)

Self-attention

$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$

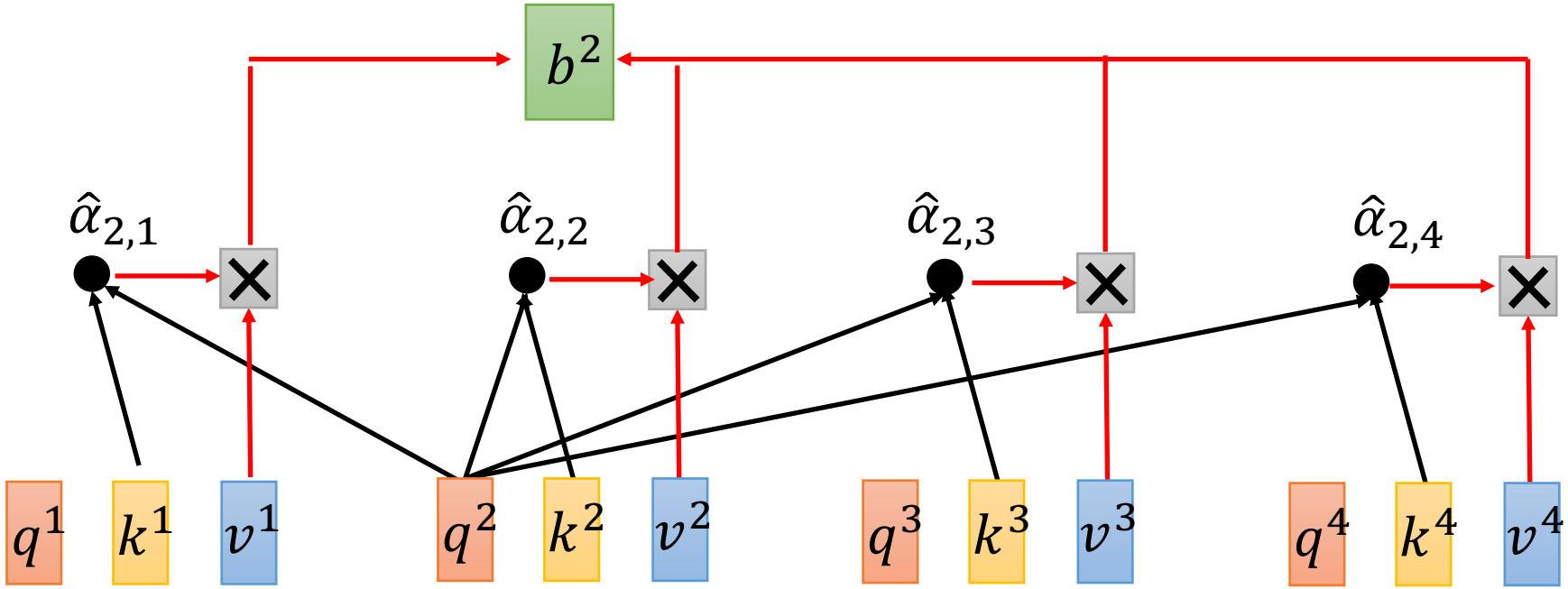


$$\begin{bmatrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{bmatrix} \leftarrow \begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{bmatrix} = \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} \begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix}$$

$\hat{A}$ A K^T Q

Self-attention

$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



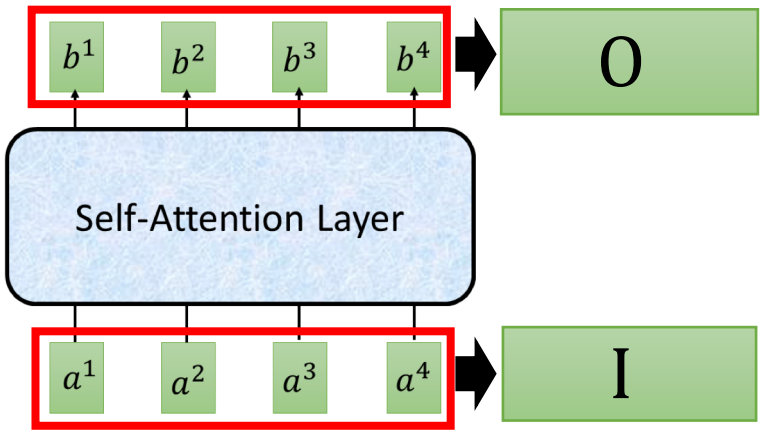
$b^1\ b^2\ b^3\ b^4$ = $v^1\ v^2\ v^3\ v^4$

O V

$$\begin{bmatrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{bmatrix}$$

$\hat{A}$

Self-attention



O

I

Q = W^q I

K = W^k I

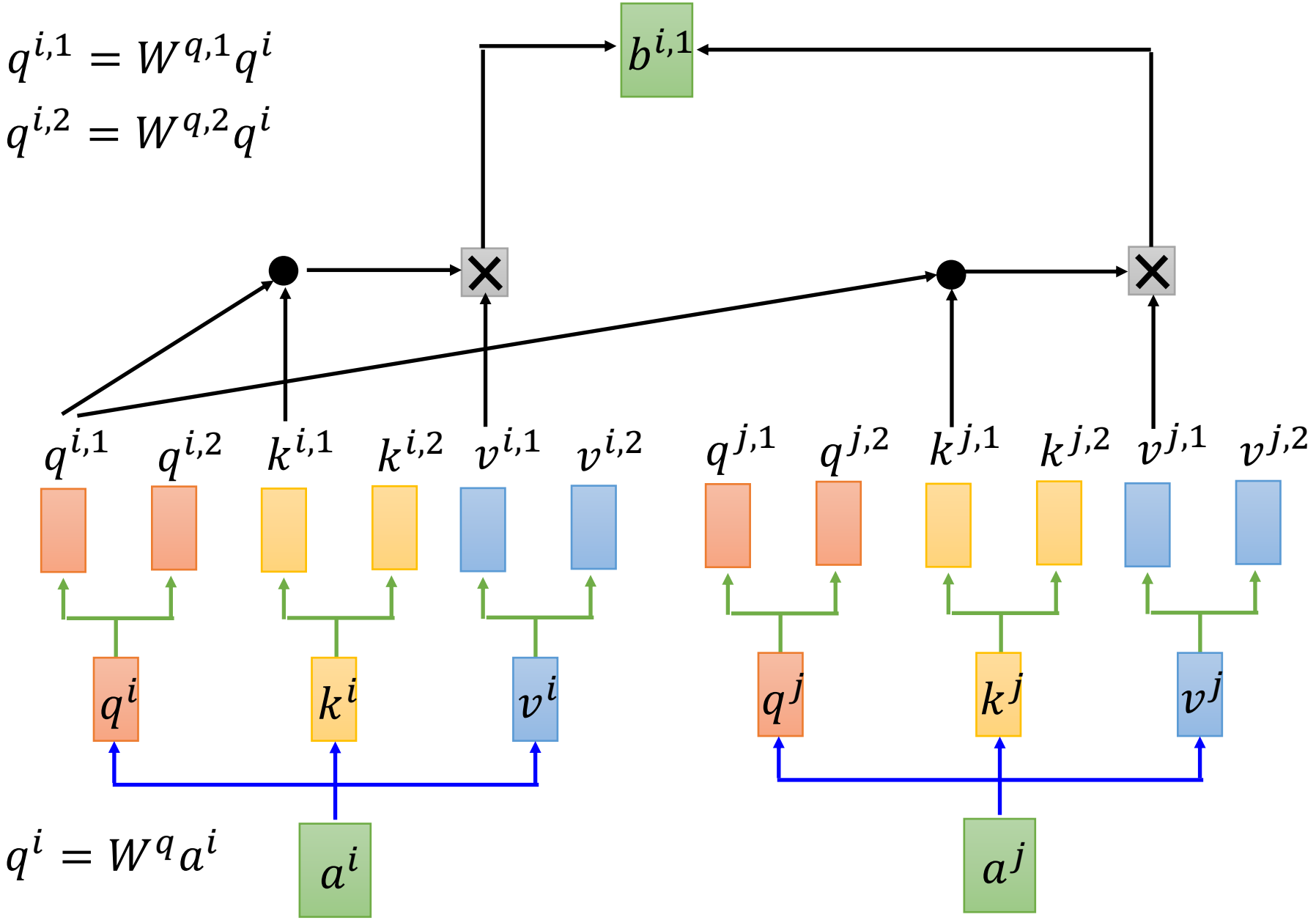
V = W^v I

$\hat{A}$ ← A = K^T Q

O = V $\hat{A}$

反正就是一堆矩陣乘法，用 GPU 可以加速

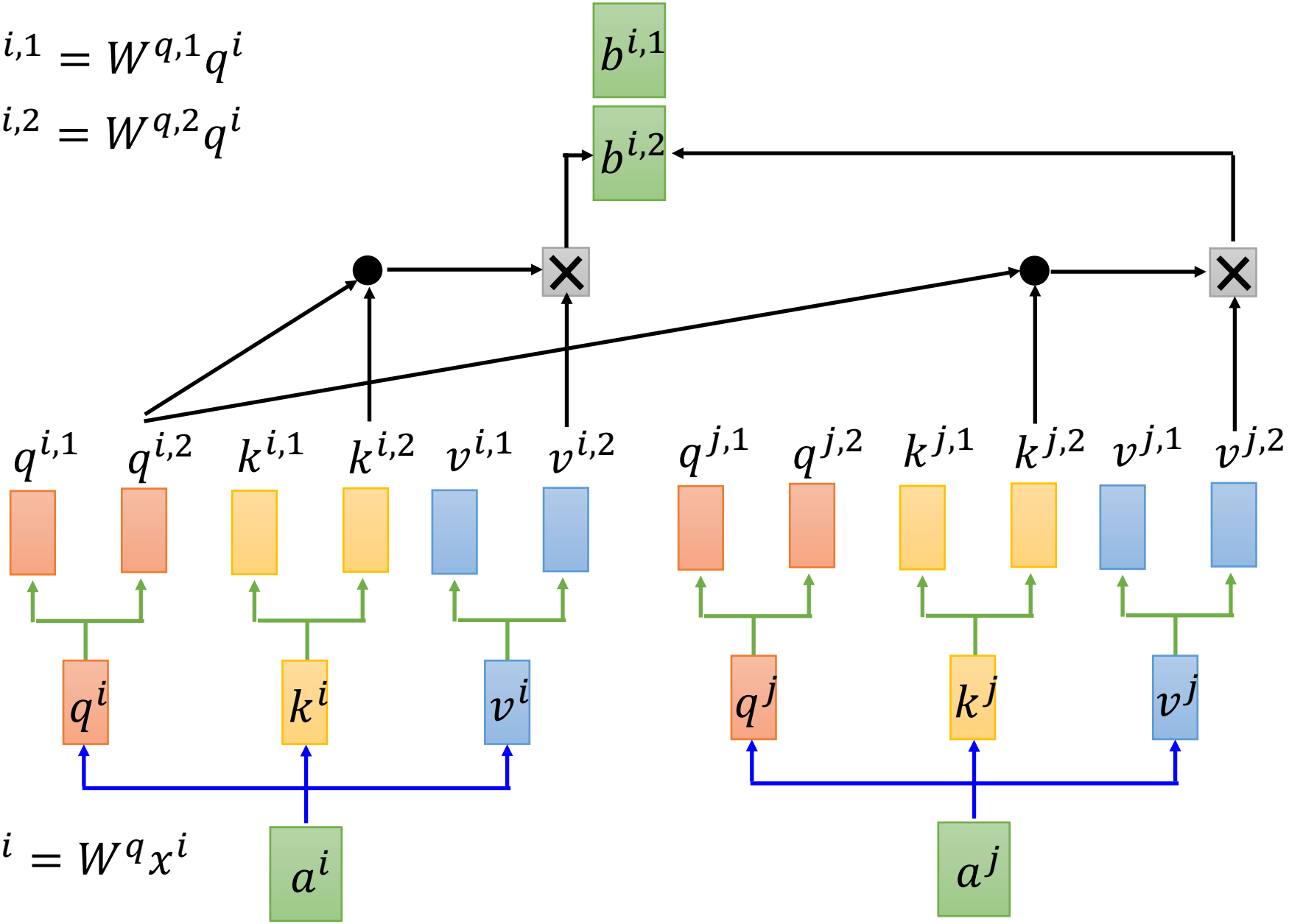
Multi-head Self-attention (2 heads as example)



Multi-head Self-attention (2 heads as example)

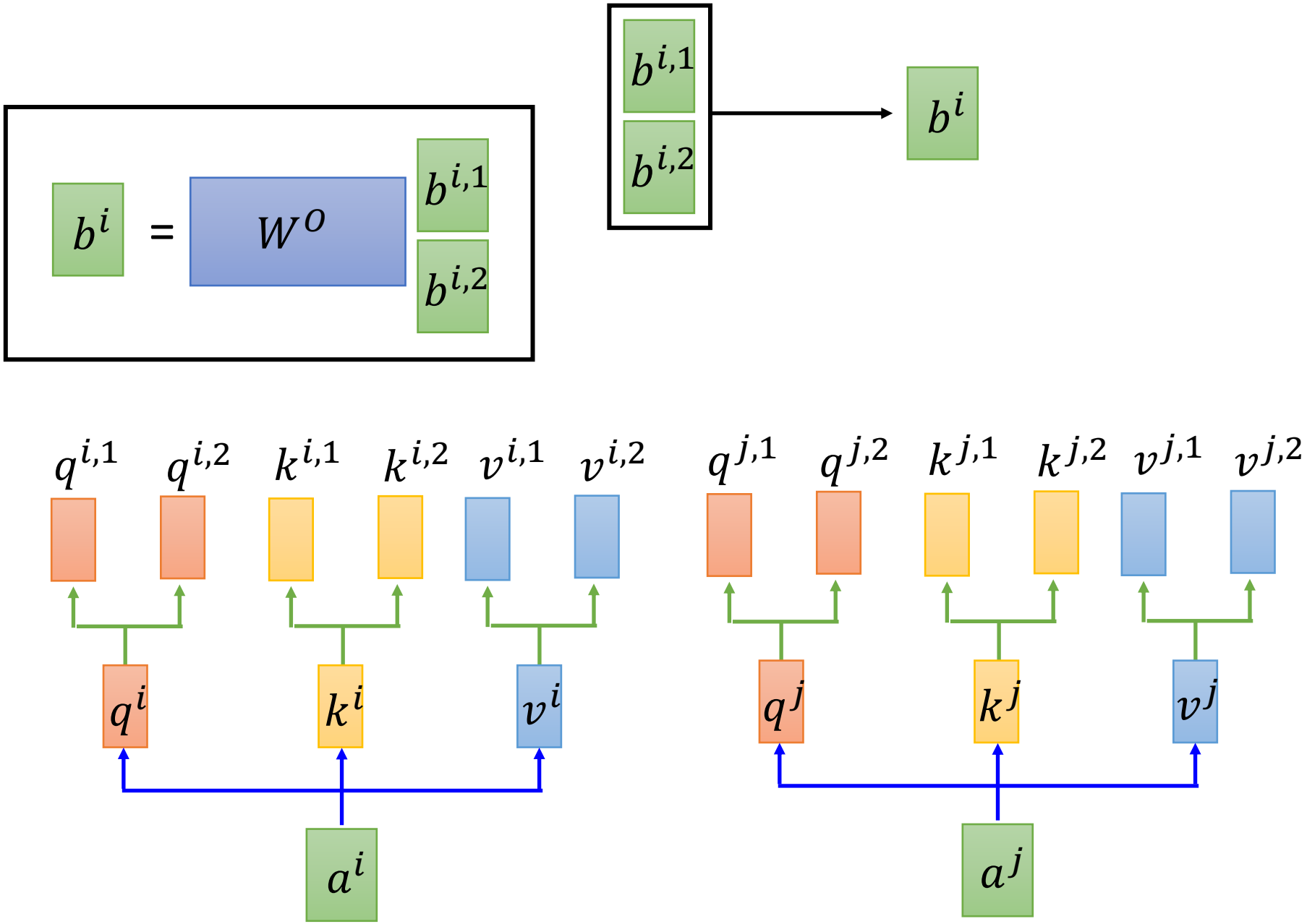
$q^{i,1} = W^{q,1} q^i$

$q^{i,2} = W^{q,2} q^i$

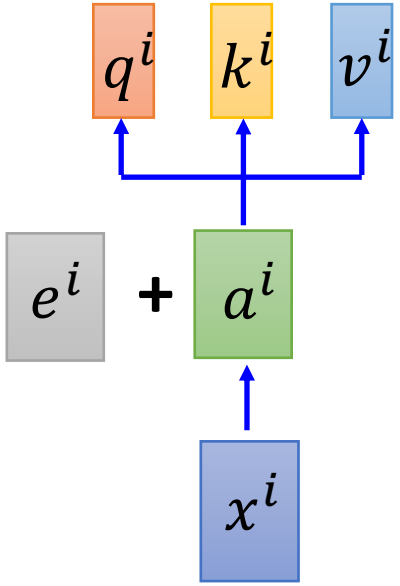


$q^i = W^q x^i$

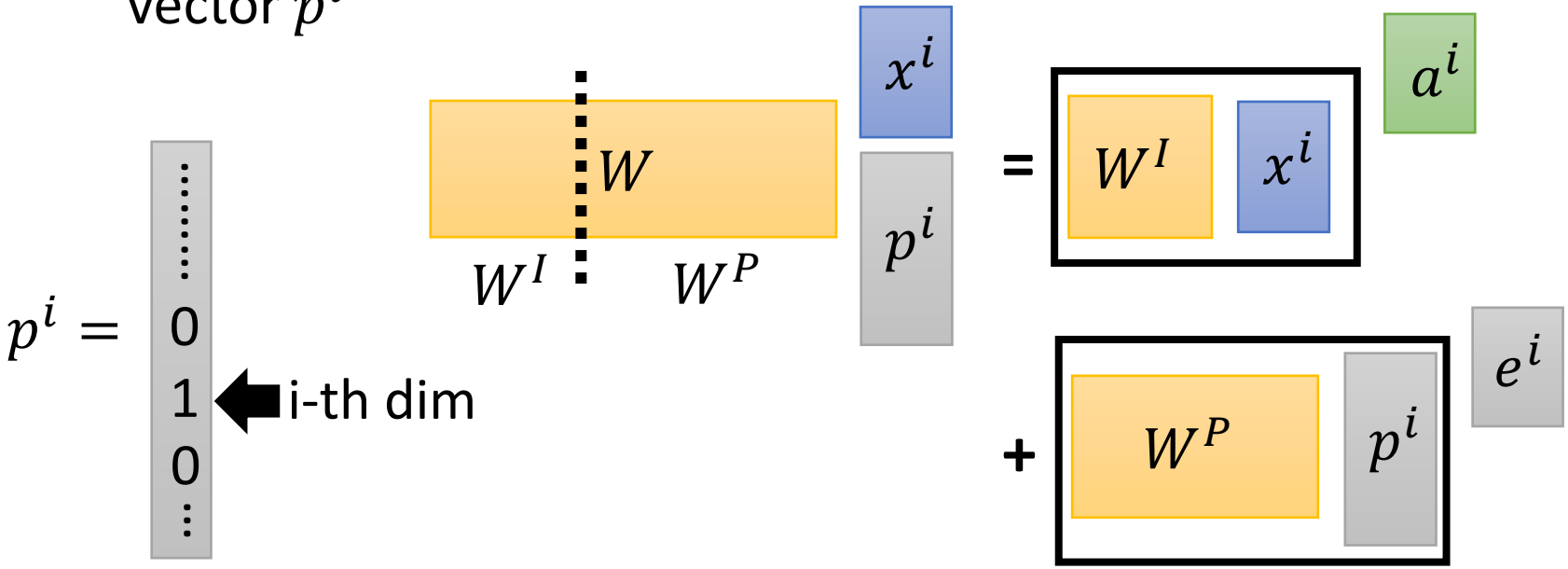
Multi-head Self-attention (2 heads as example)

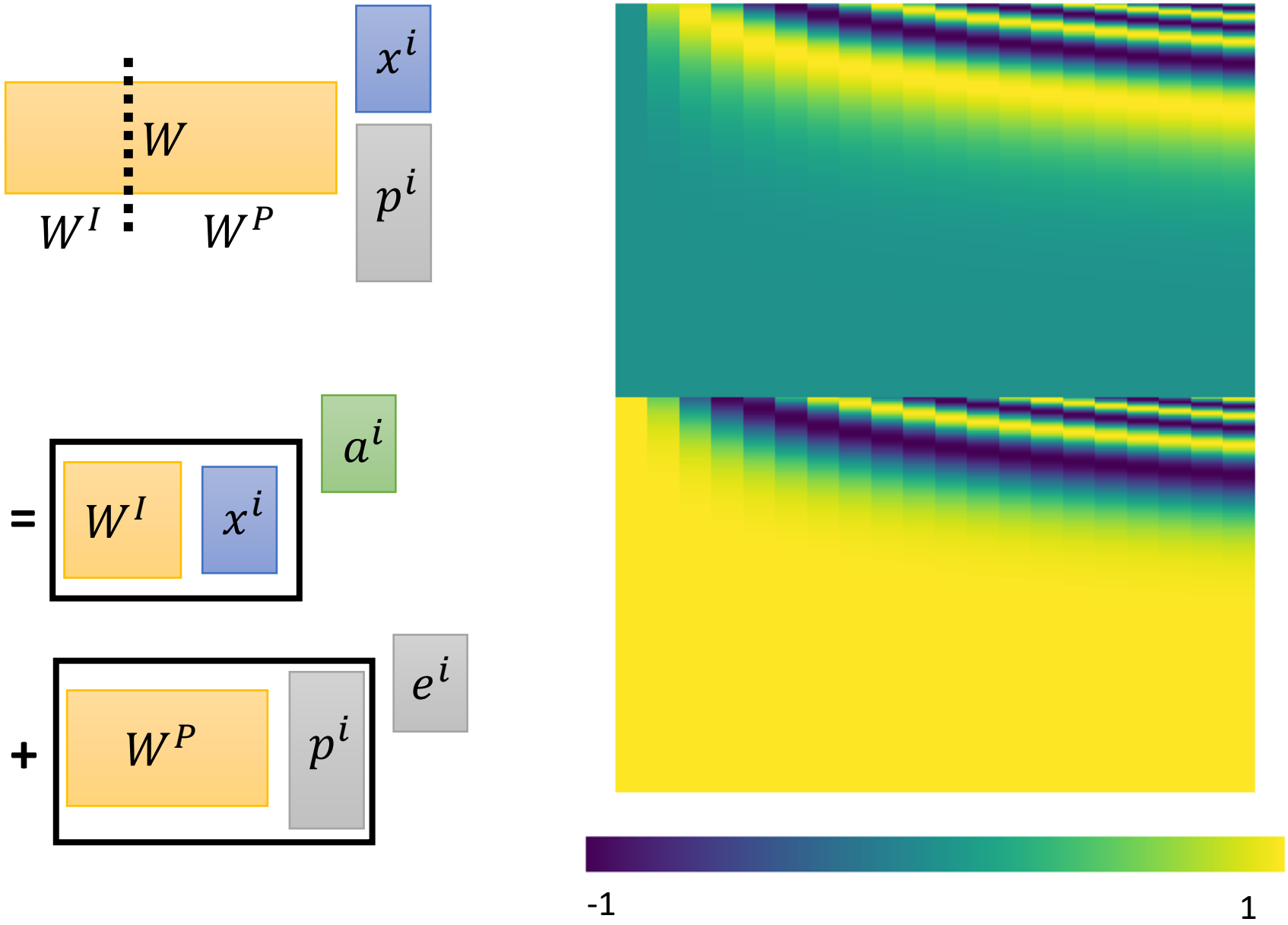


Positional Encoding



- No position information in self-attention.
- Original paper: each position has a unique positional vector e^i (not learned from data)
- In other words: each x^i appends a one-hot vector p^i

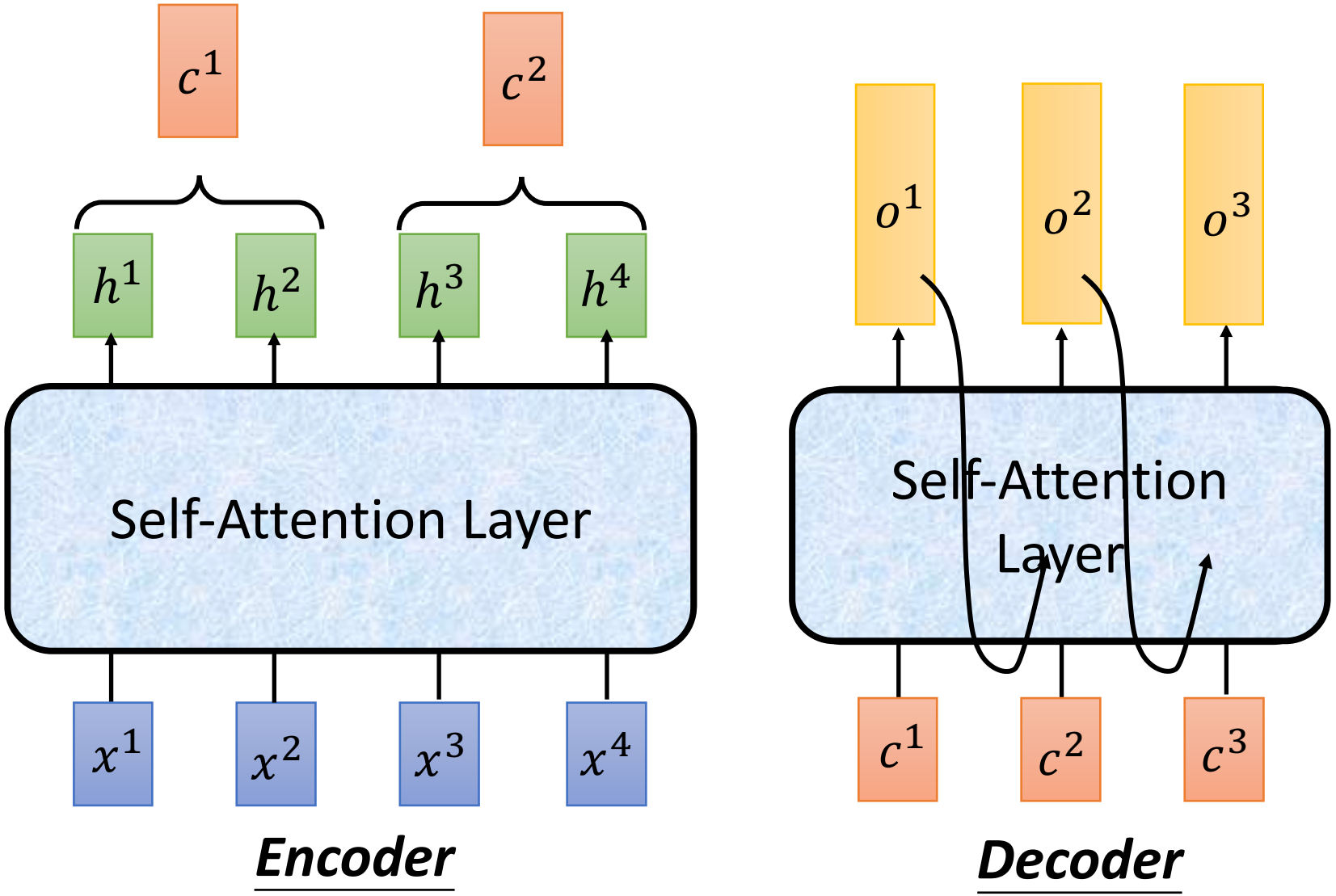




source of image: http://jalammar.github.io/illustrated-transformer/

Review: https://www.youtube.com/watch?v=ZjfjPzXw6og&feature=youtu.be

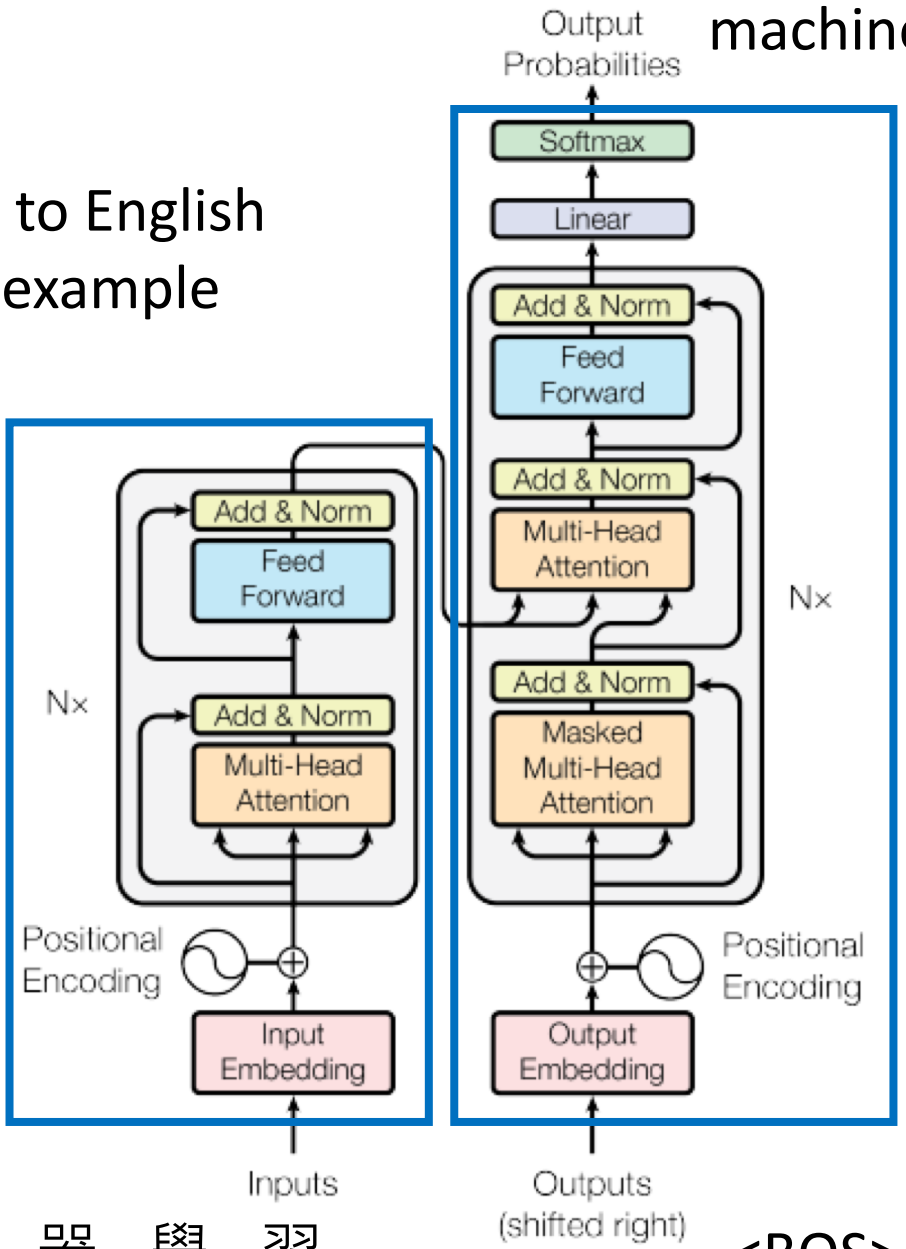
Seq2seq with Attention



https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Transformer

Using Chinese to English translation as example



Encoder

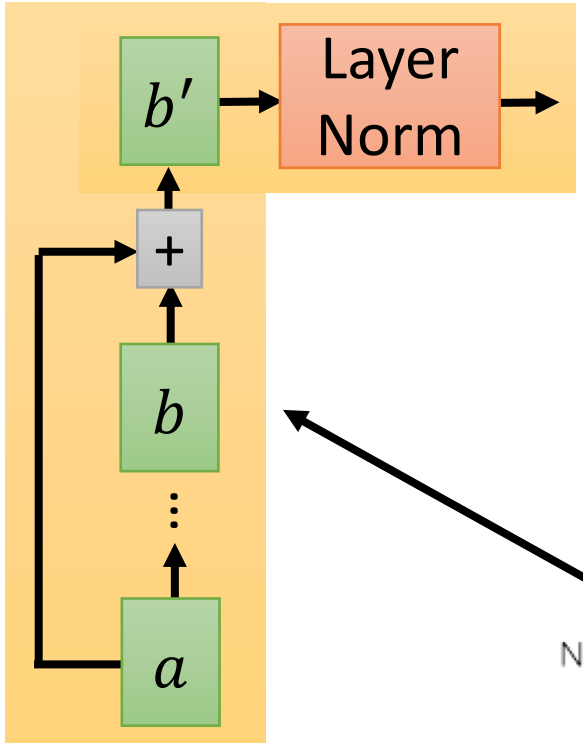
Decoder

機 器 學 習

<BOS> machine

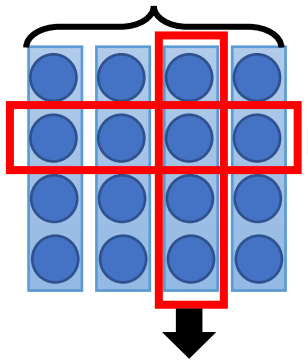
machine learning

Transformer



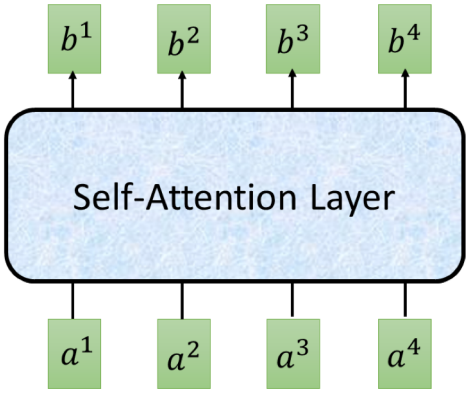
Layer Norm:
https://arxiv.org/abs/1607.06450

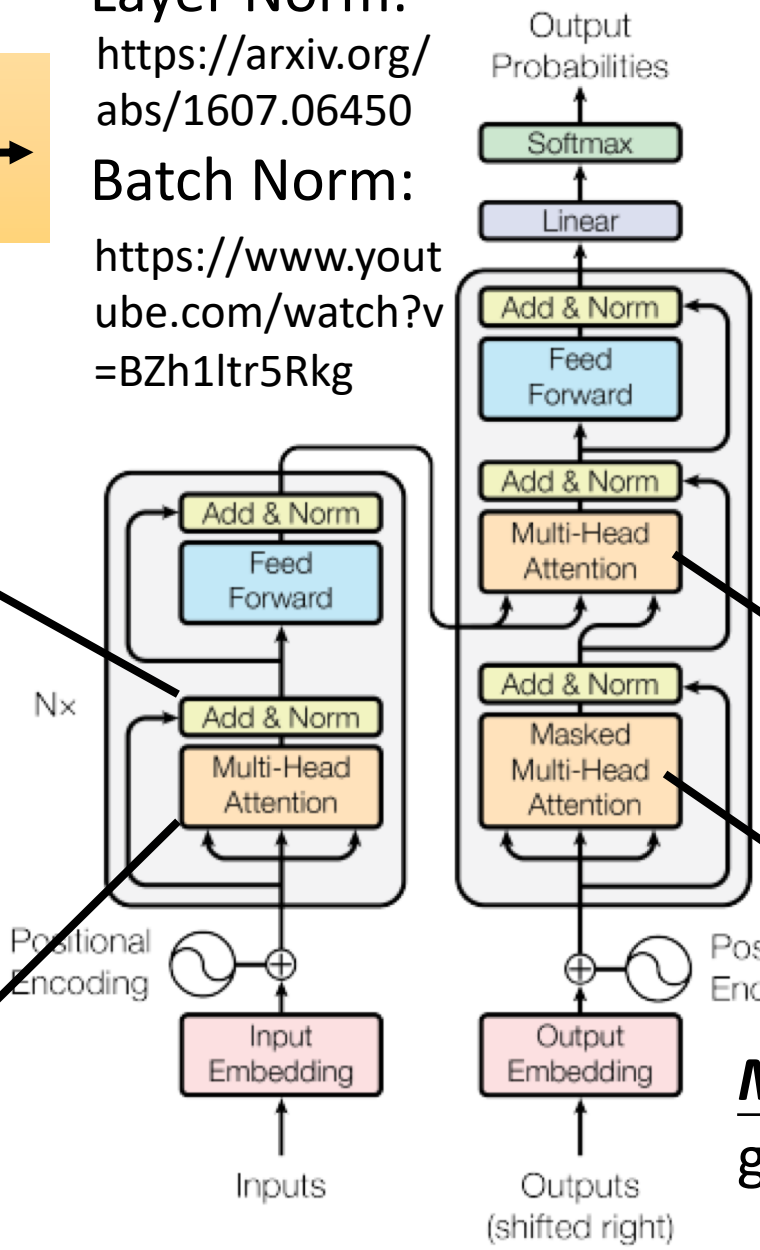
Batch Norm:
https://www.youtube.com/watch?v=BZh1ltr5Rkg



$\mu = 0, \sigma = 1$
Batch

$\mu = 0, \sigma = 1$
Layer

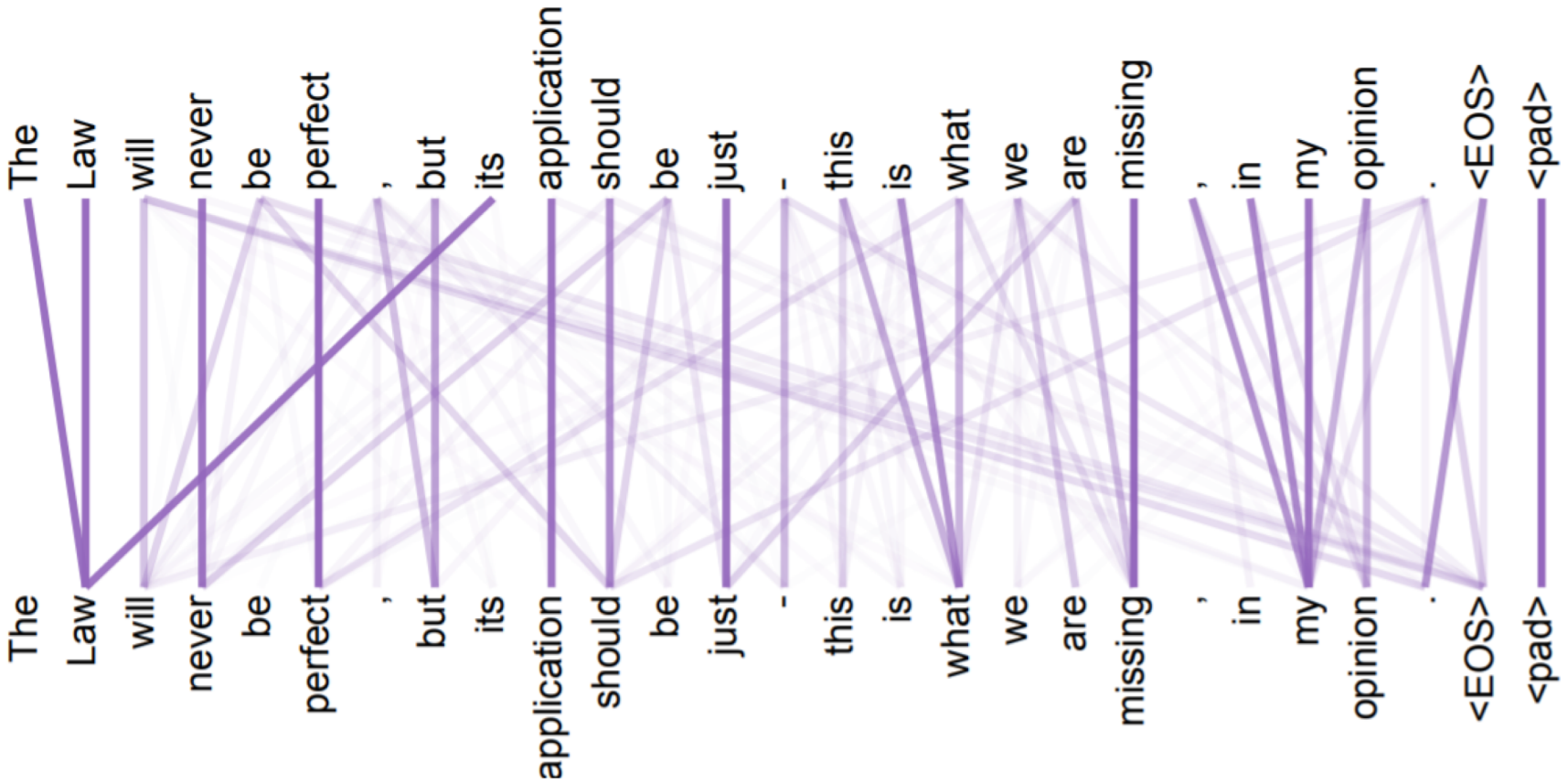




attend on the input sequence

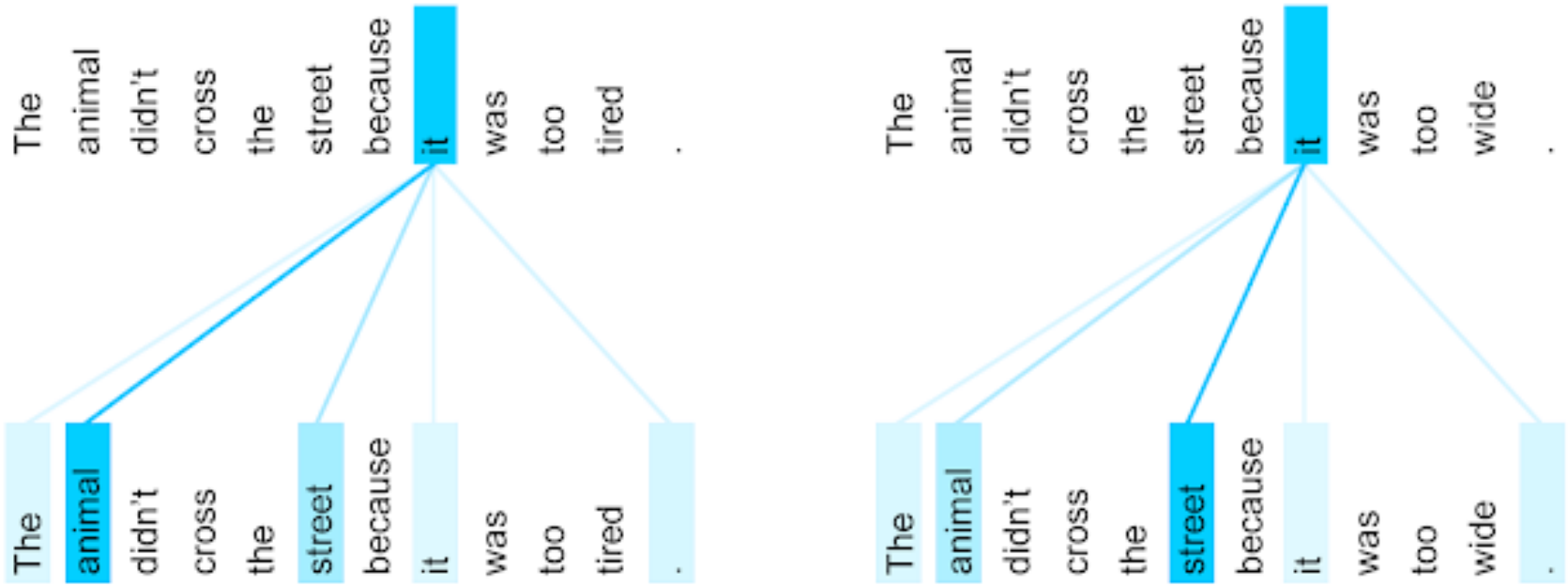
Masked: attend on the generated sequence

Attention Visualization



https://arxiv.org/abs/1706.03762

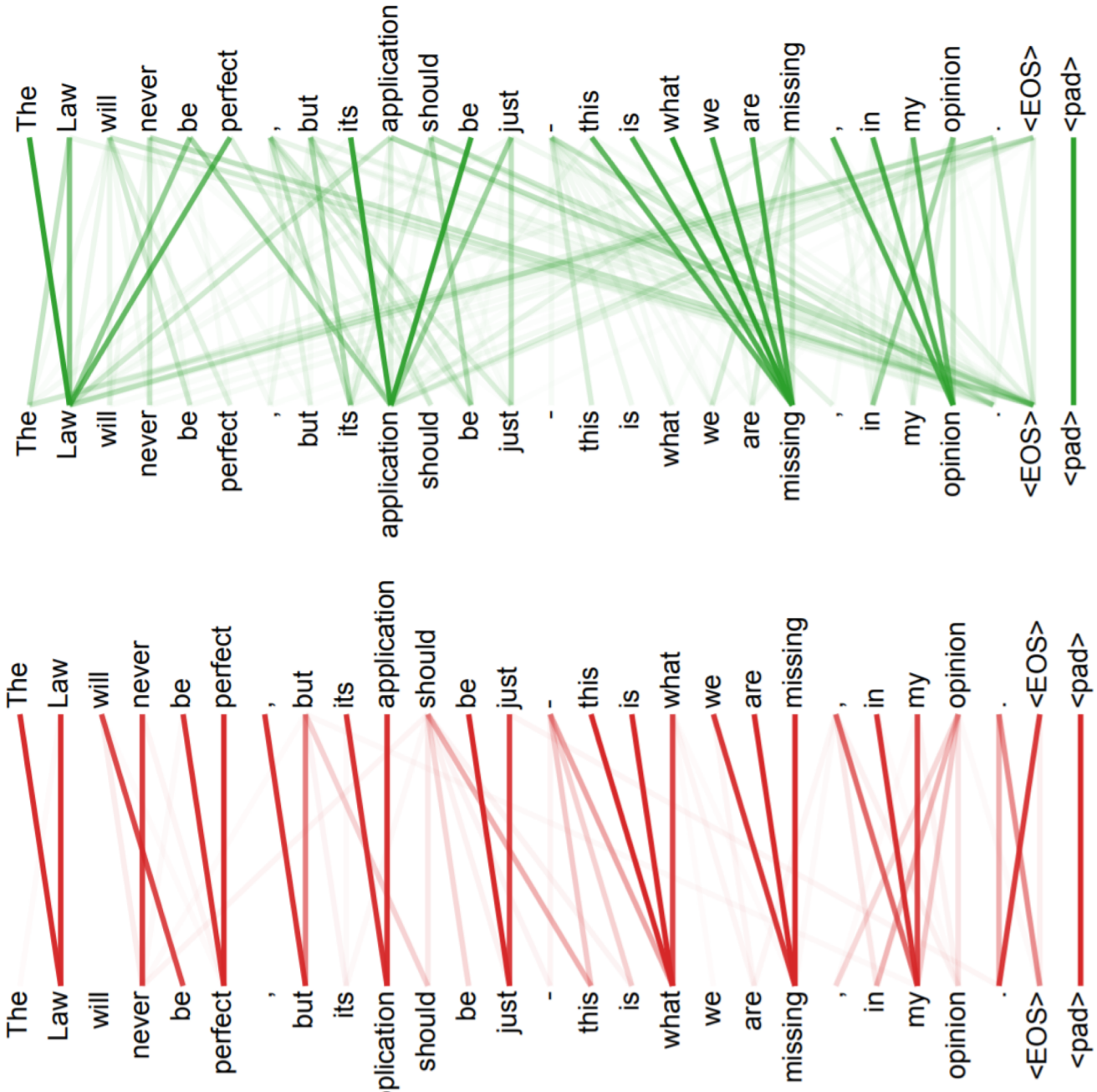
Attention Visualization



The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

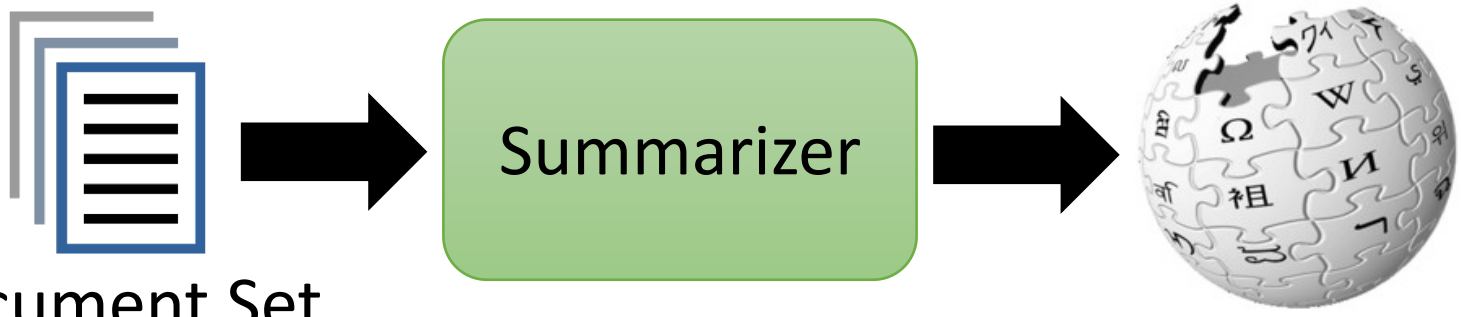
https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Multi-head Attention



Example Application

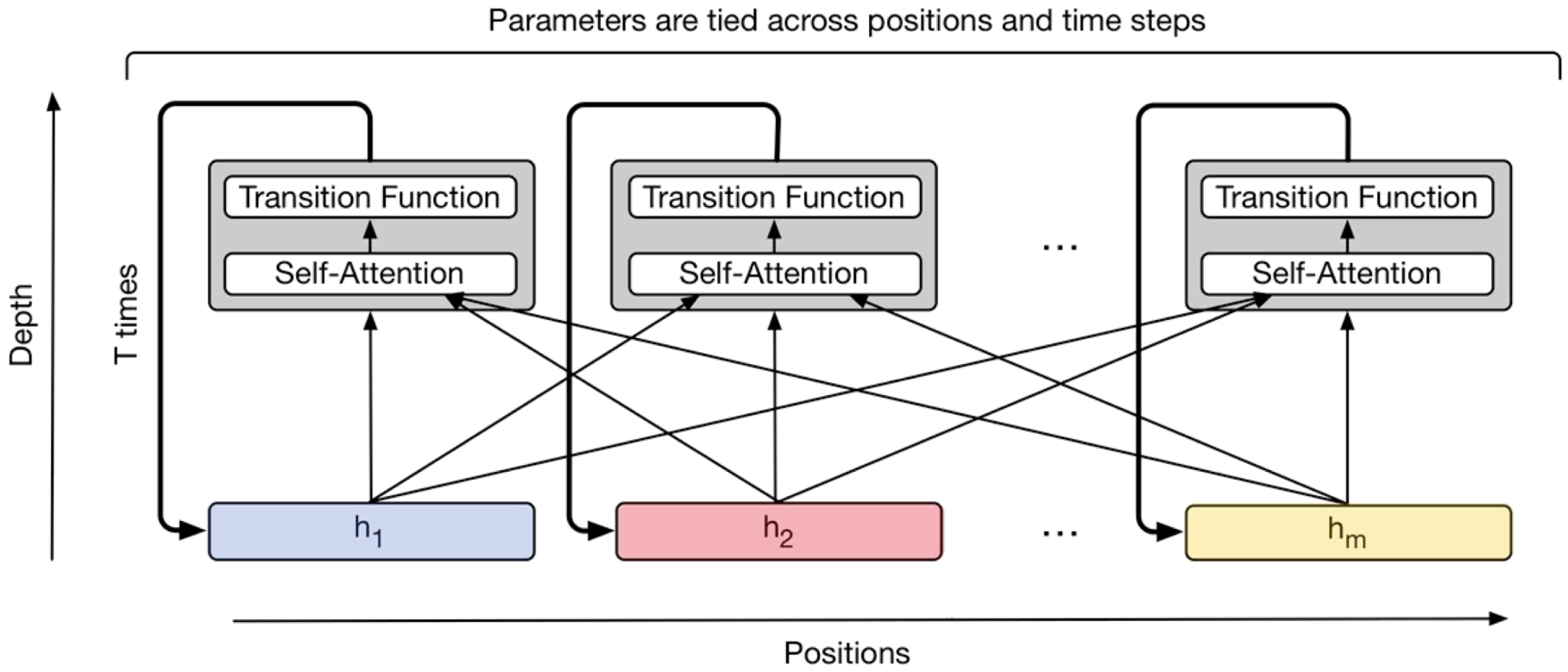
- If you can use seq2seq, you can use transformer.



Dataset	Input	Output	# examples
Gigaword (Graff & Cieri, 2003)	10^1	10^1	10^6
CNN/DailyMail (Nallapati et al., 2016)	10^2–10^3	10^1	10^5
WikiSum (ours)	10^2–10^6	10^1–10^3	10^6

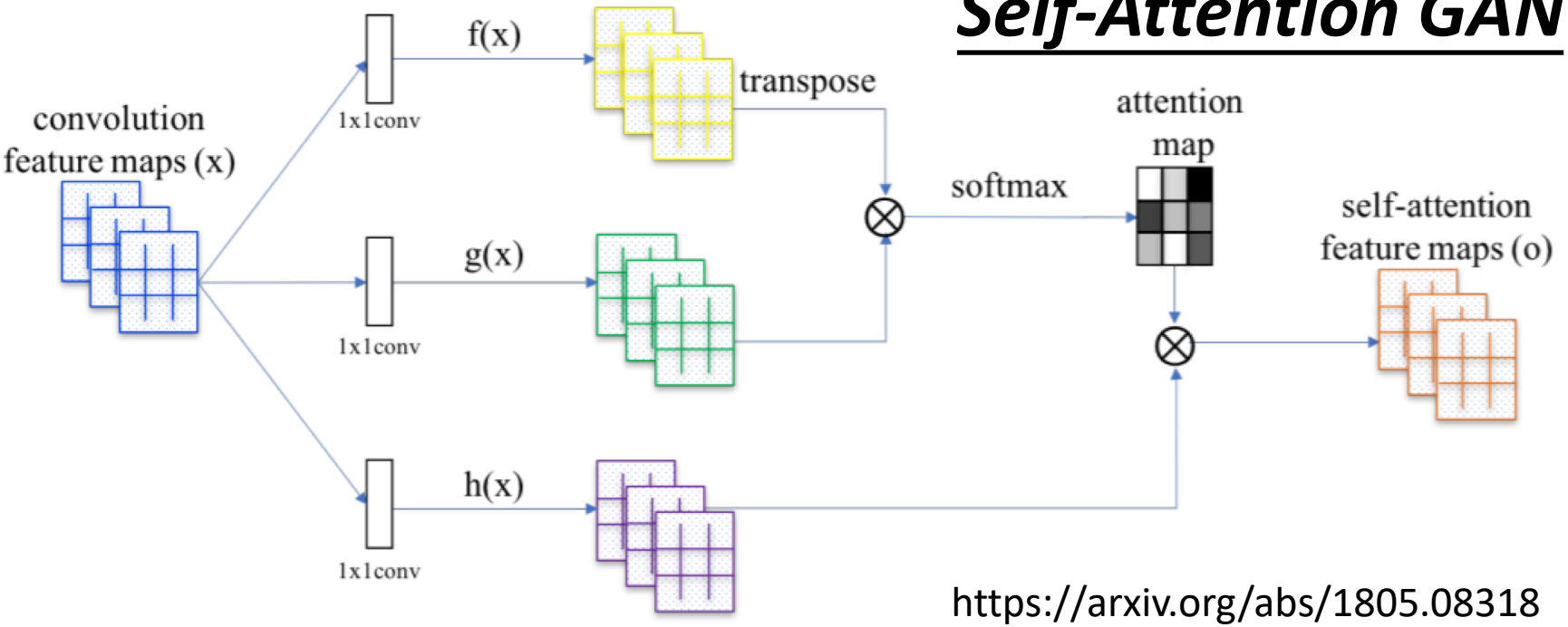
https://arxiv.org/abs/1801.10198

Universal Transformer

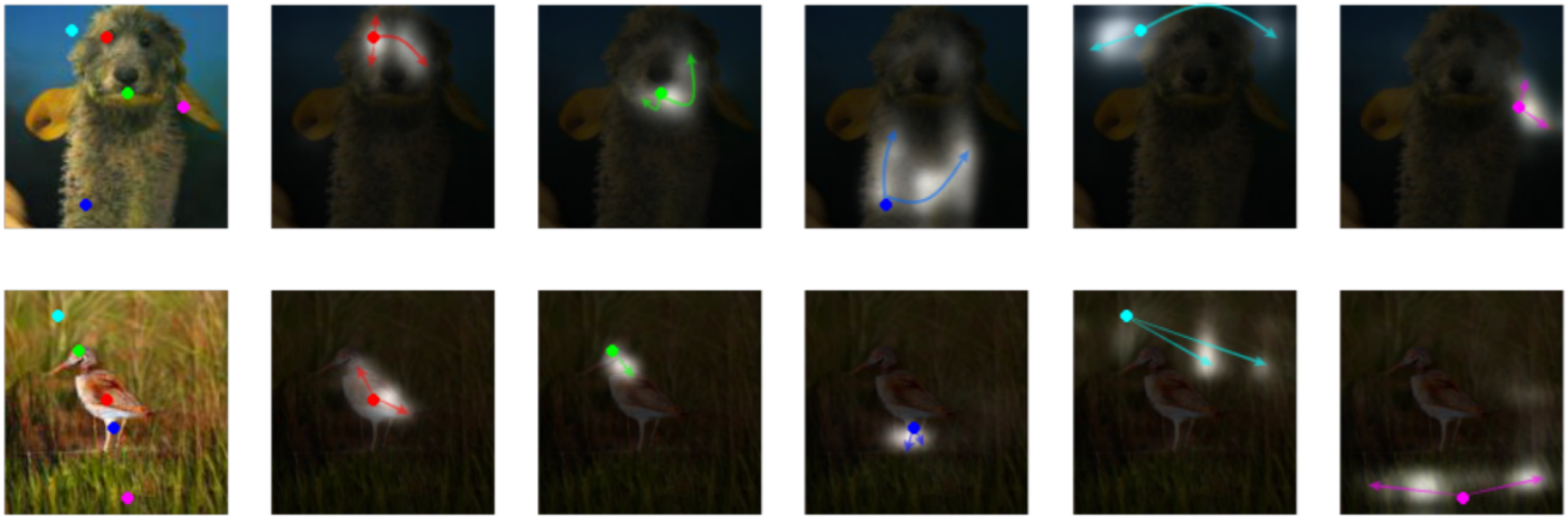


https://ai.googleblog.com/2018/08/moving-beyond-translation-with.html

Self-Attention GAN



https://arxiv.org/abs/1805.08318



ELMO, BERT and GPT

Syllabus

Week	Contents
1	Introduction to Data Science
2	Data Crawling [HW1: Crawling Releases]
3	Natural Language Processing (I): Word Representation
4	Natural Language Processing (II): Recurrent Neural Network
5	Natural Language Processing (III): Sentiment Analysis and Recommendation System (HW2: Attractiveness Prediction)
6	Natural Language Processing (IV): Text Generation
7	Natural Language Processing (V): Question Answering
8	Midterm

1-of-N Encoding

apple = [1 0 0 0 0]

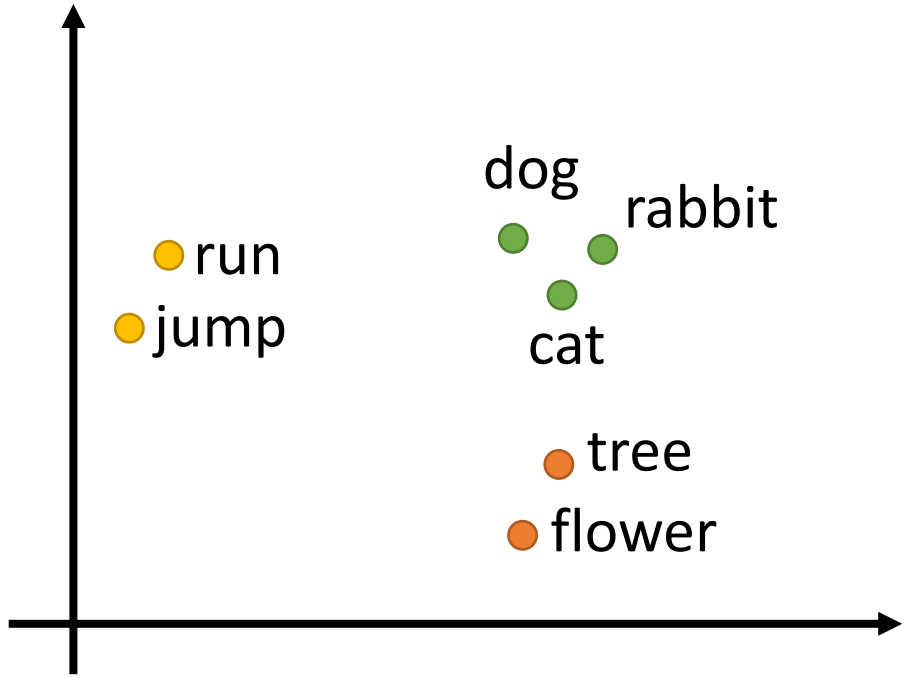
bag = [0 1 0 0 0]

cat = [0 0 1 0 0]

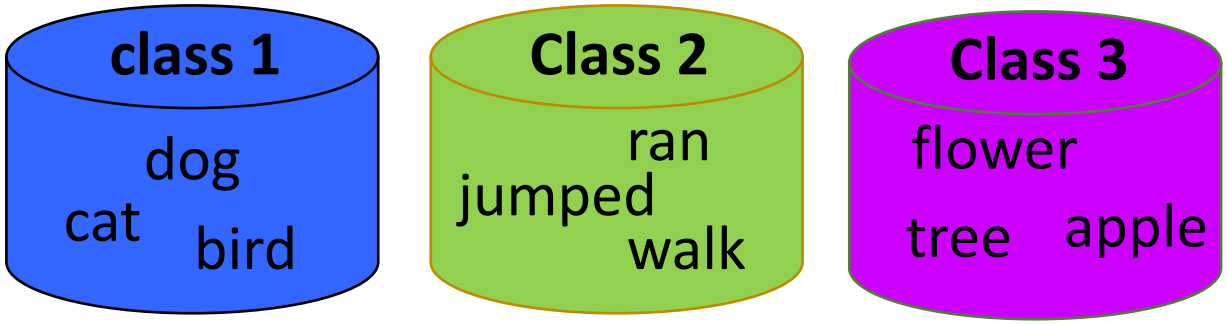
dog = [0 0 0 1 0]

elephant = [0 0 0 0 1]

Word Embedding



Word Class



https://arxiv.org/abs/1902.06006

A word can have multiple senses.

Have you paid that money to the bank yet ?
It is safest to deposit your money in the bank .

The victim was found lying dead on the river bank .
They stood on the river bank to fish.

The hospital has its own blood bank.

The third sense or not?

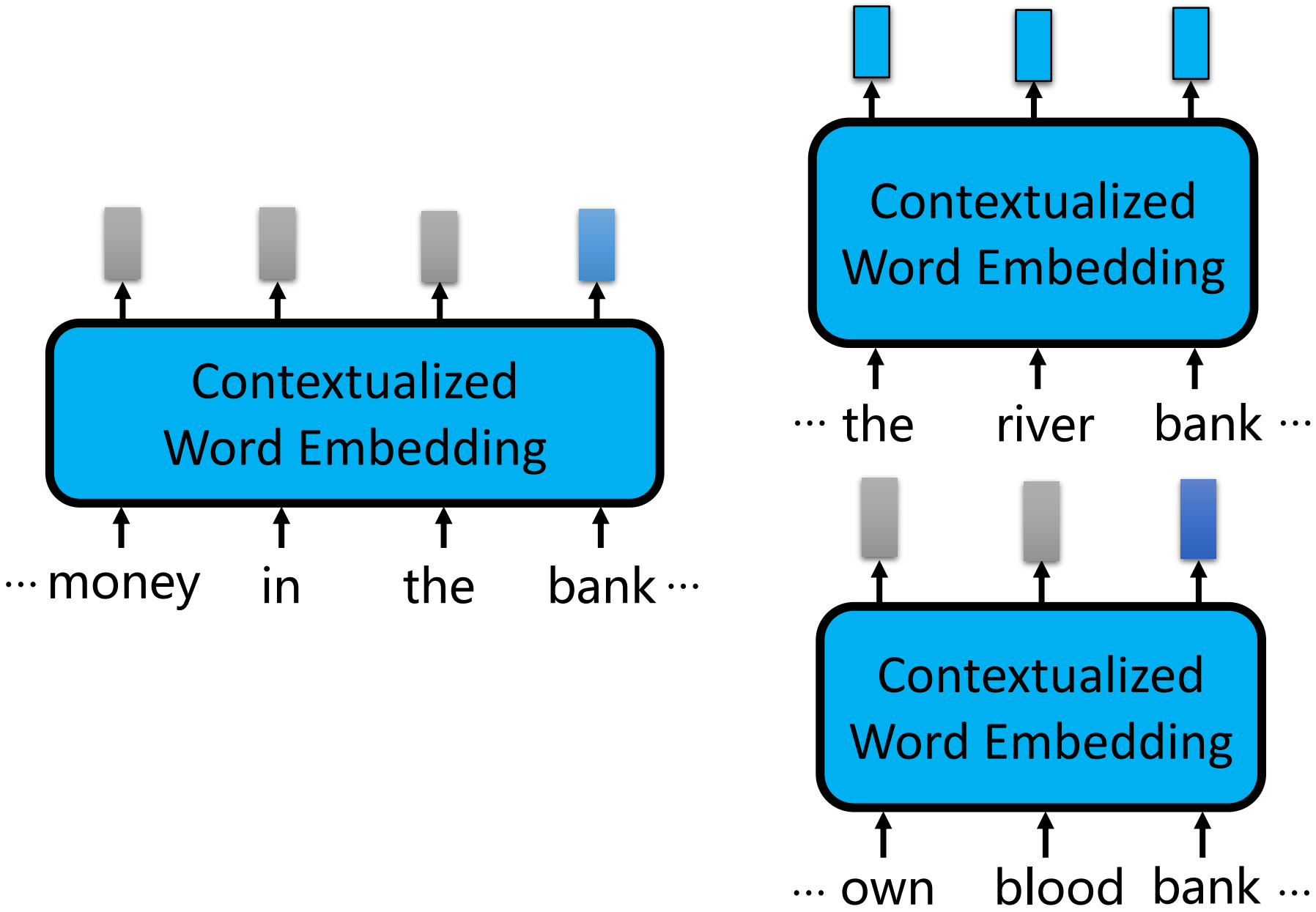
More Examples

小明走了一天了

More Examples

阿嬤，你要哪片音樂光碟？我燒給你。

Contextualized Word Embedding

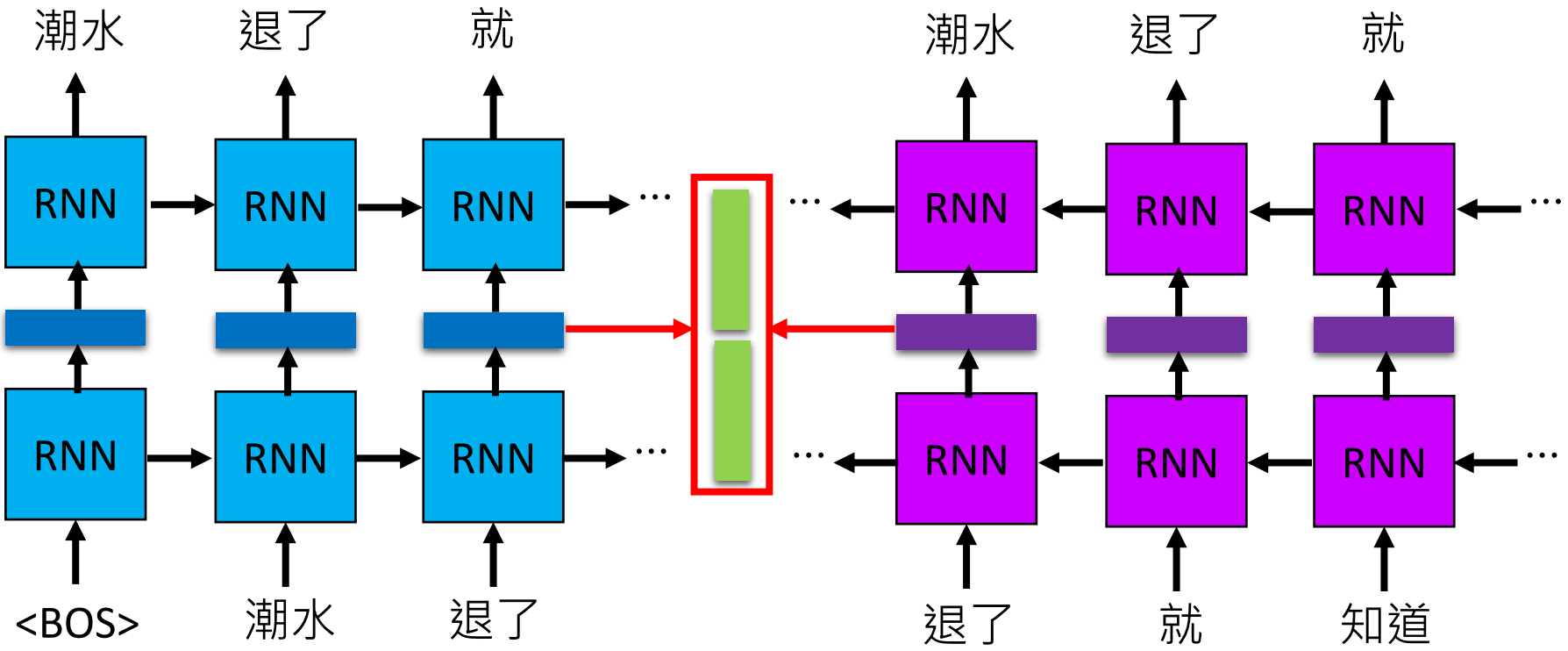


Embeddings from Language Model (ELMO)

https://arxiv.org/abs/1802.05365

- RNN-based language models (trained from lots of sentences)

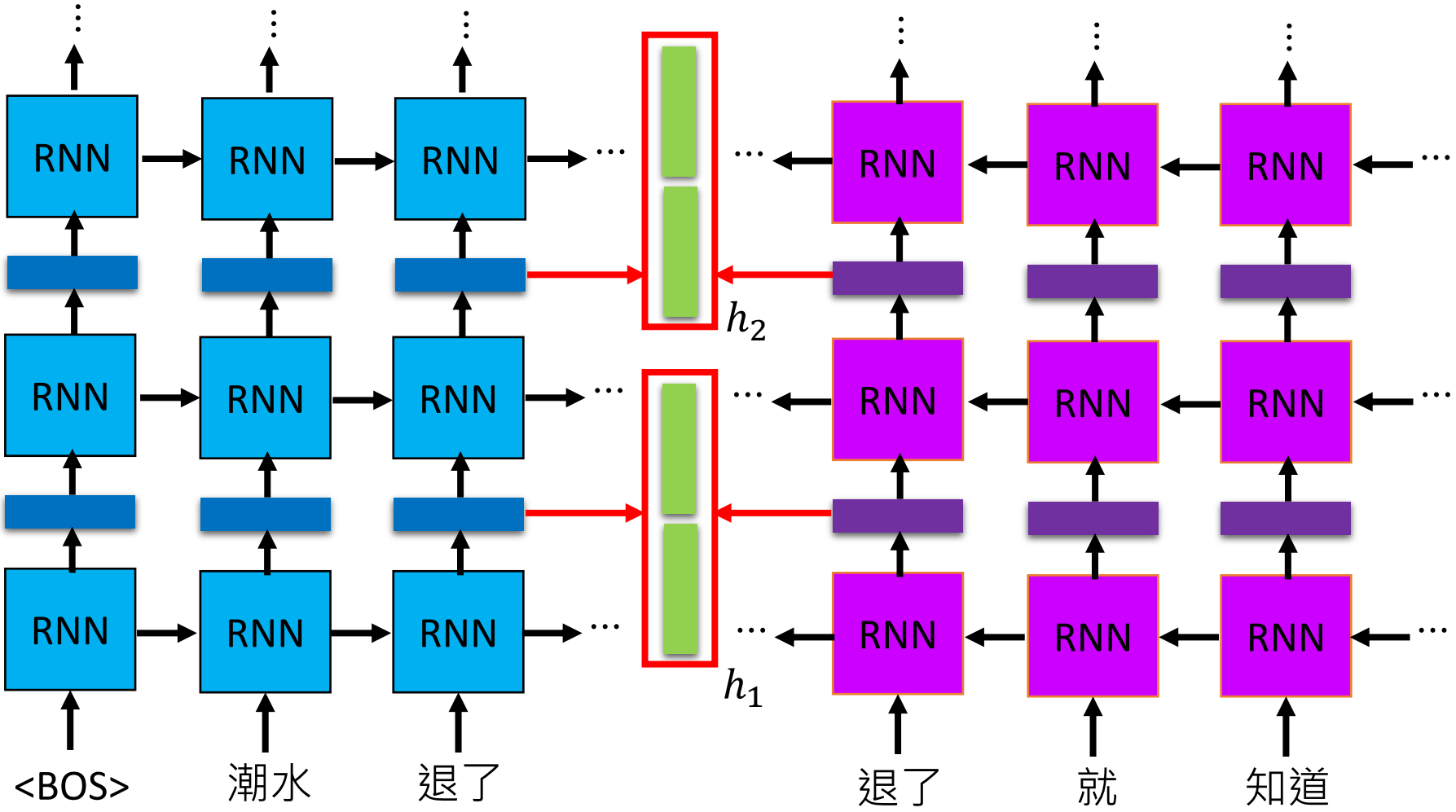
e.g. given “潮水 退了 就 知道 誰 沒穿 褲子”



ELMO

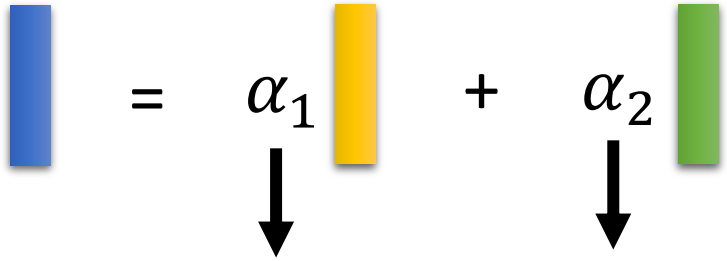
Each layer in deep LSTM can generate a latent representation.

Which one should we use???

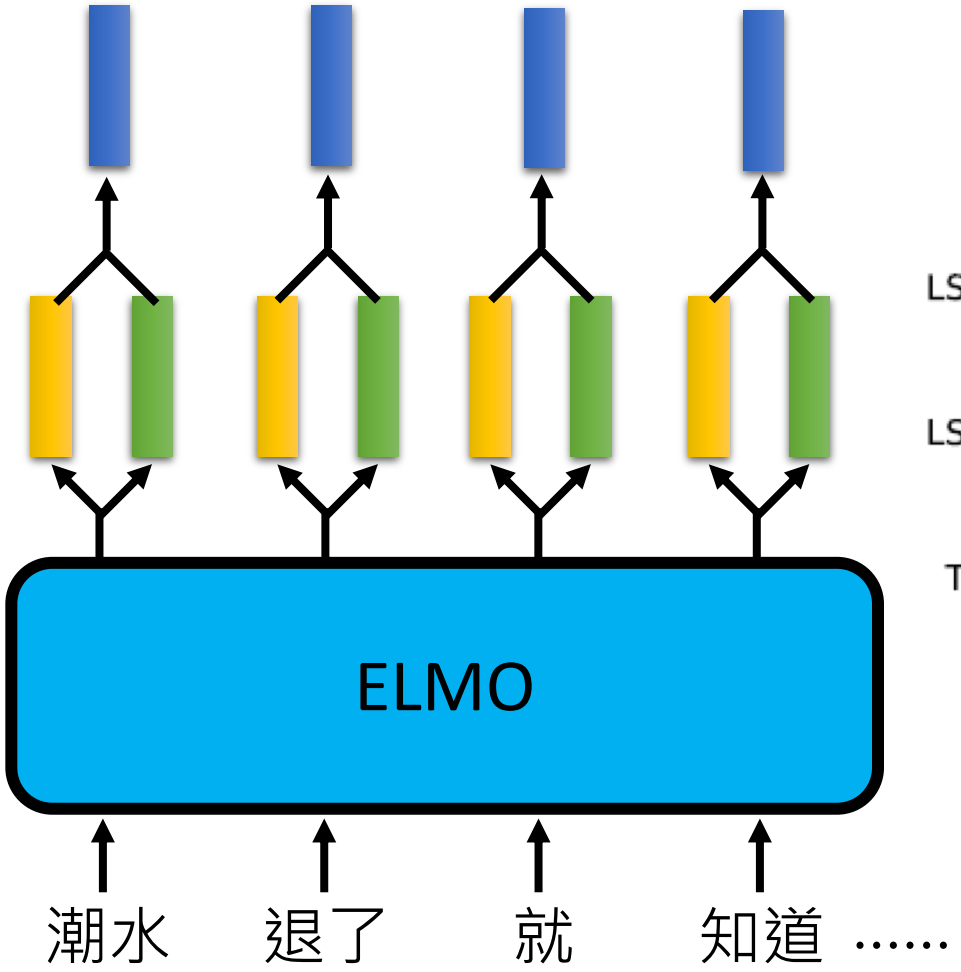


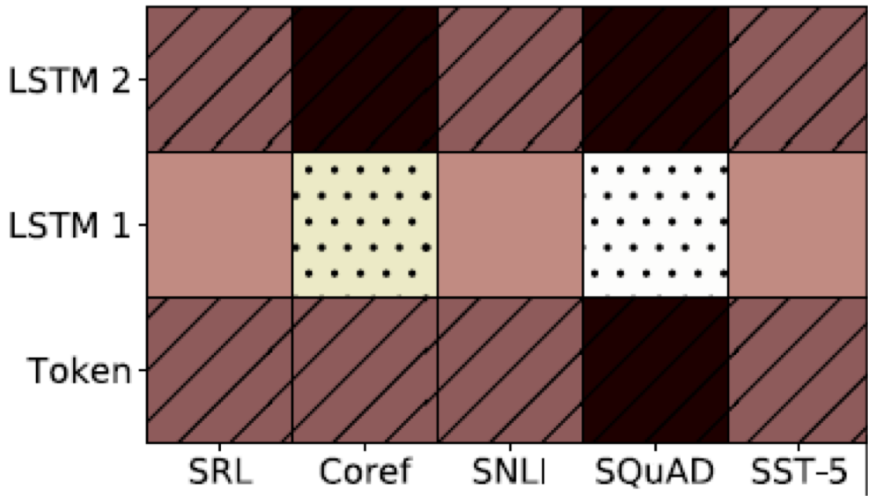

我全都要

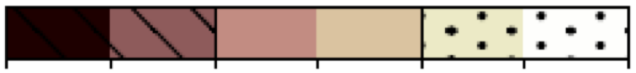
ELMO



Learned with the down stream tasks





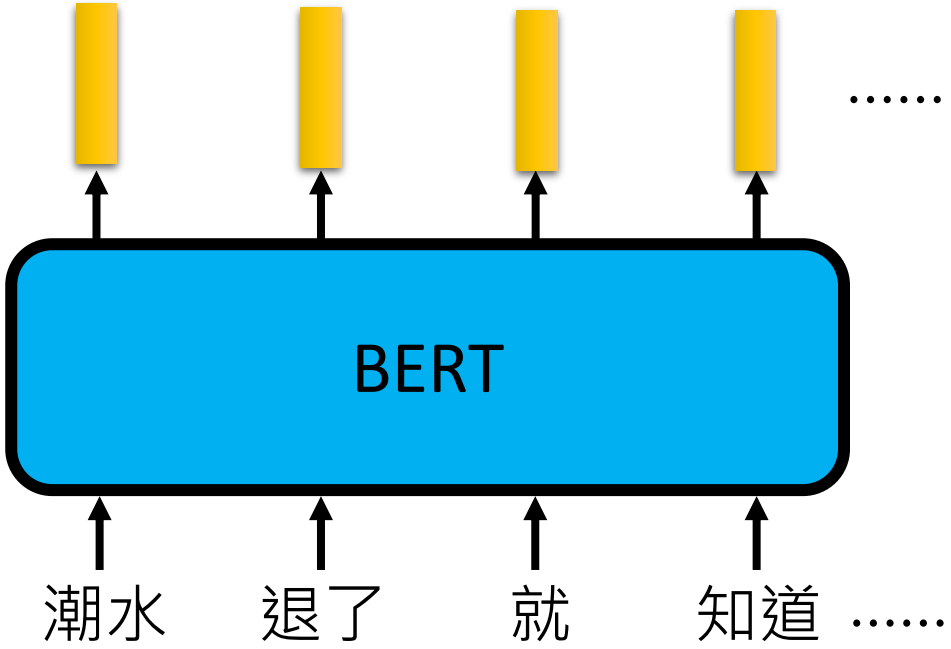


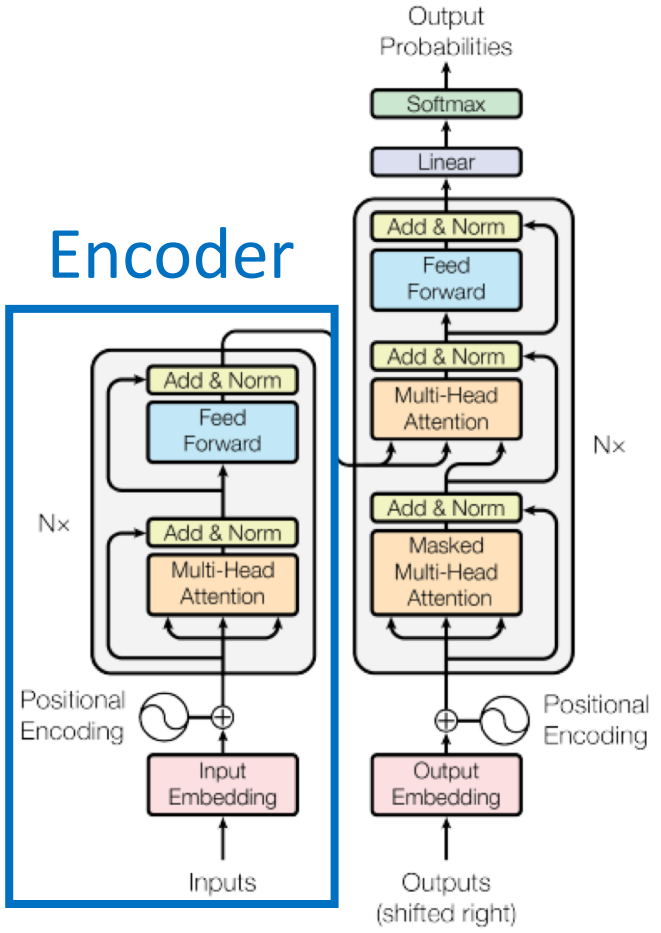
small large

Bidirectional Encoder Representations from Transformers (BERT)

- BERT = Encoder of Transformer

Learned from a large amount of text without annotation

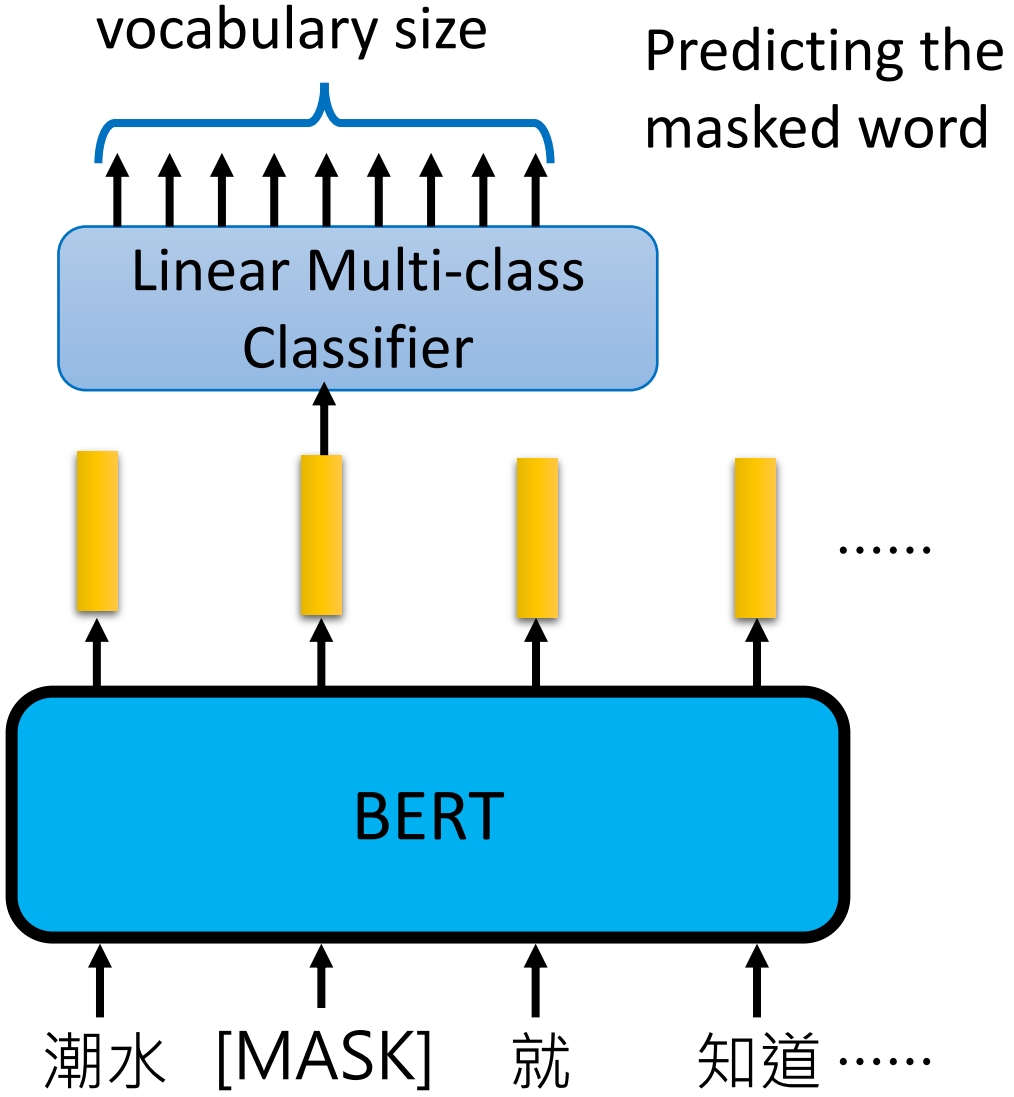


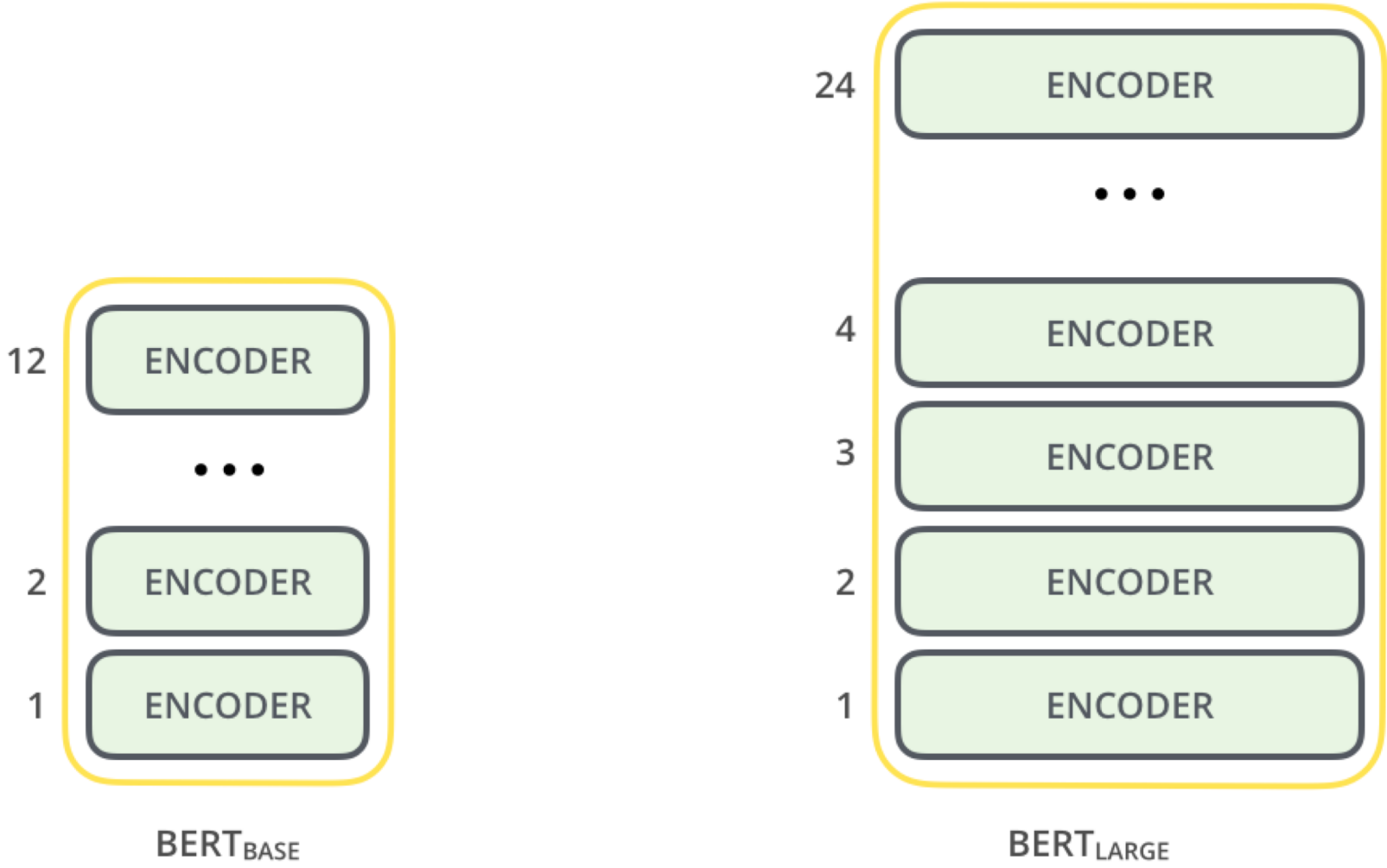


Training of BERT

- Approach 1:

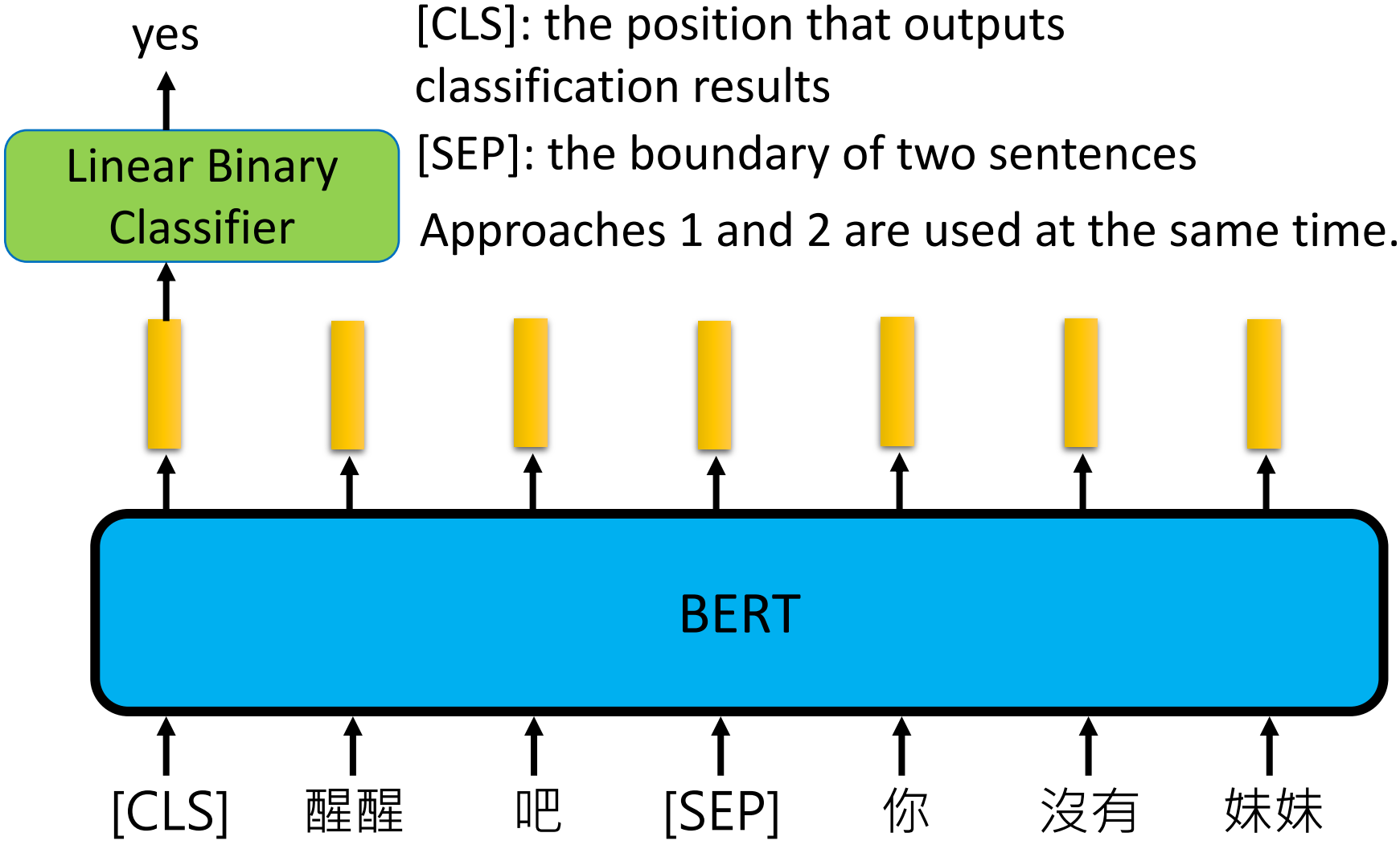
  Masked LM




12
ENCODER
2
ENCODER
1
ENCODER
BERT BASE
24
ENCODER
4
ENCODER
3
ENCODER
2
ENCODER
1
ENCODER
BERT LARGE

Training of BERT

Approach 2: Next Sentence Prediction



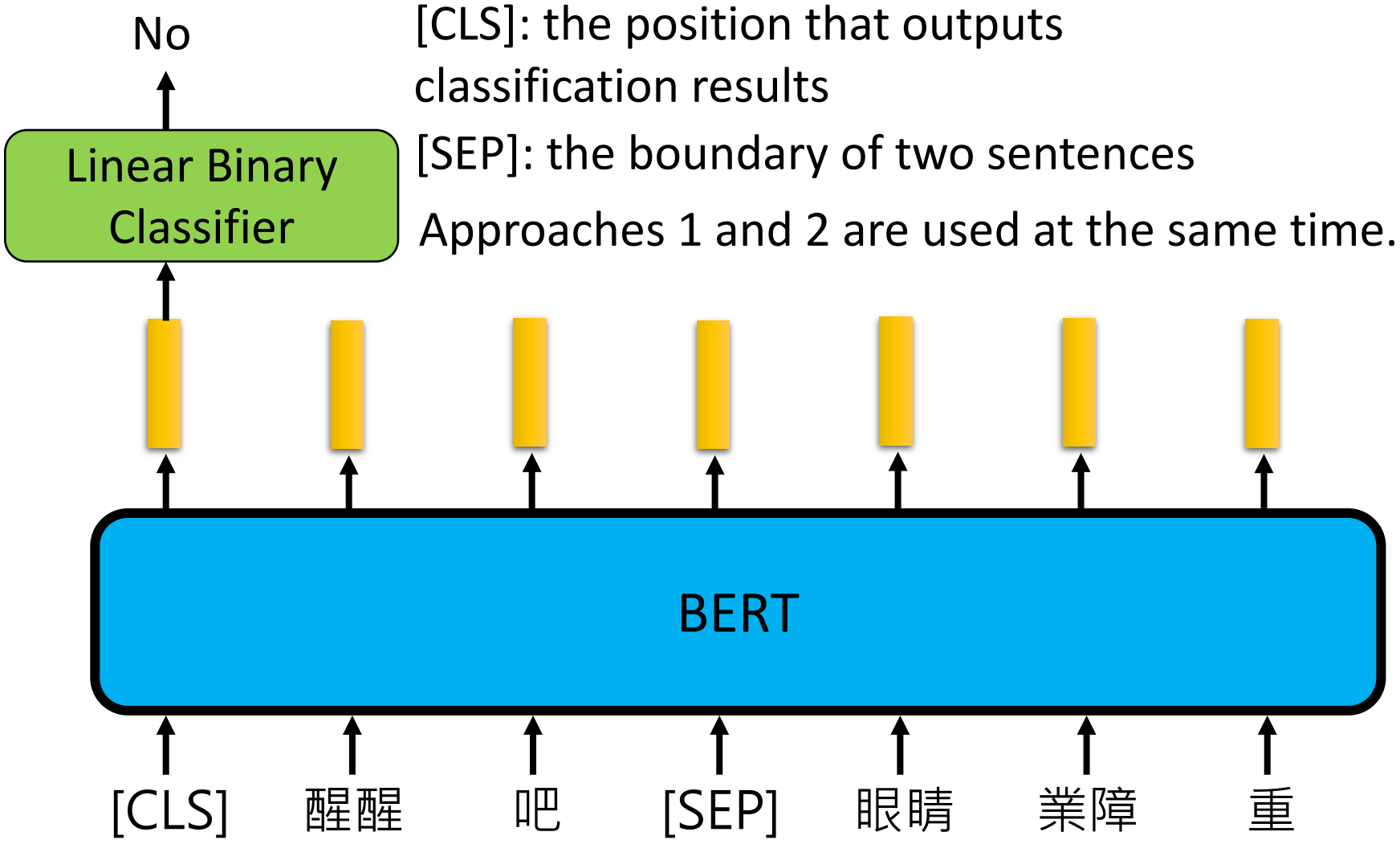
[CLS]: the position that outputs classification results

[SEP]: the boundary of two sentences

Approaches 1 and 2 are used at the same time.

Training of BERT

Approach 2: Next Sentence Prediction

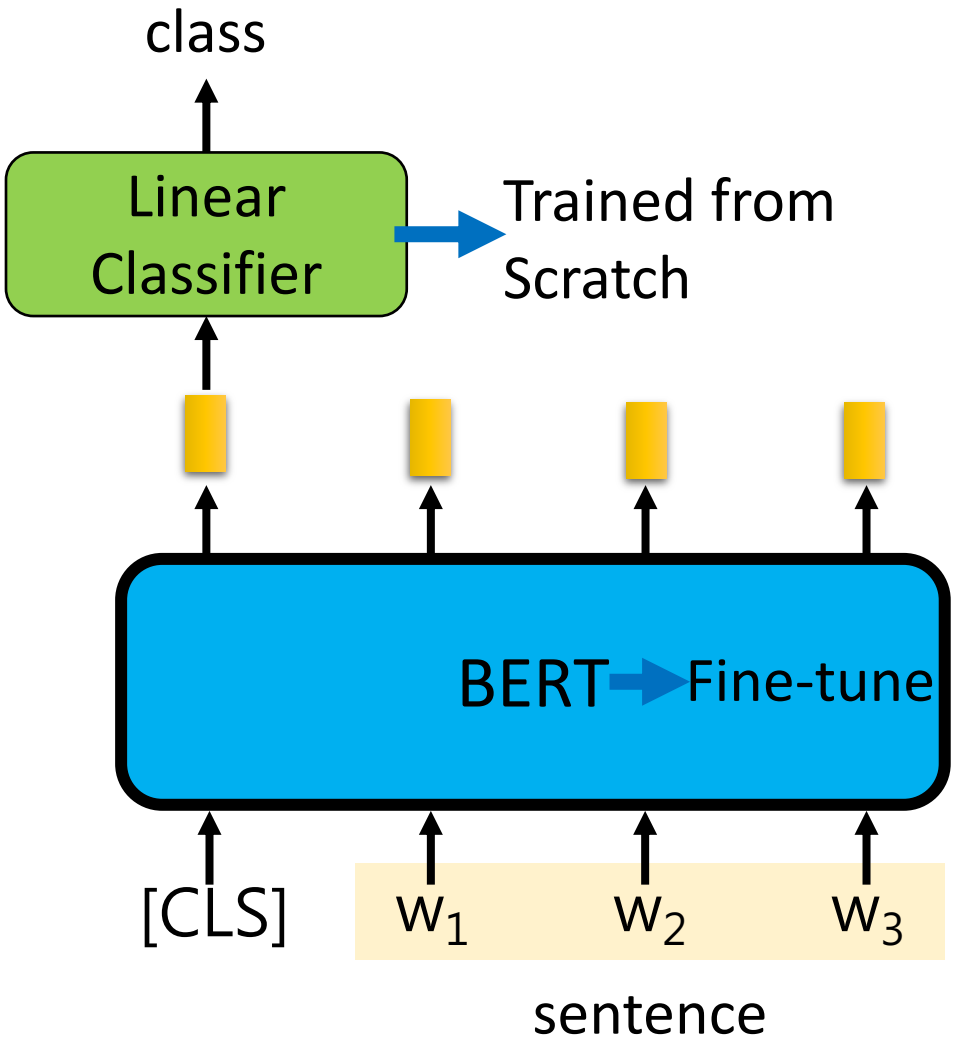


[CLS]: the position that outputs classification results

[SEP]: the boundary of two sentences

Approaches 1 and 2 are used at the same time.

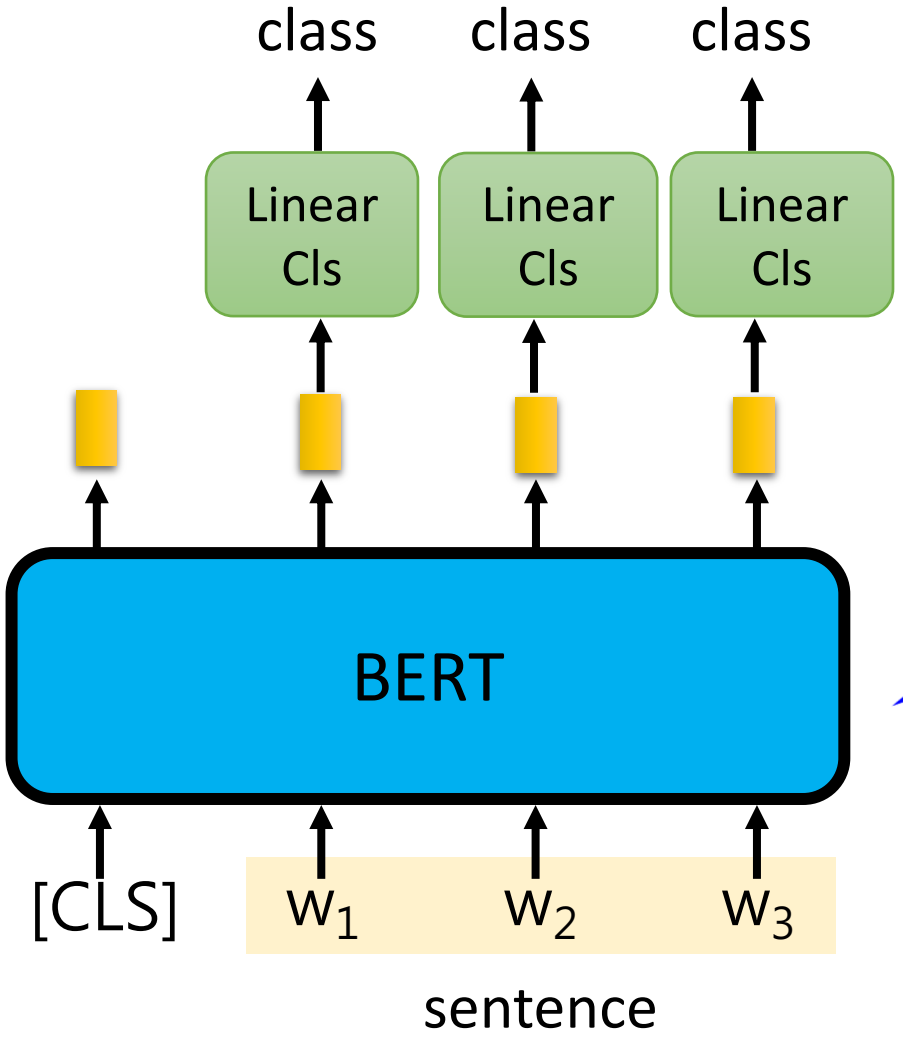
How to use BERT – Case 1



Input: single sentence, output: class

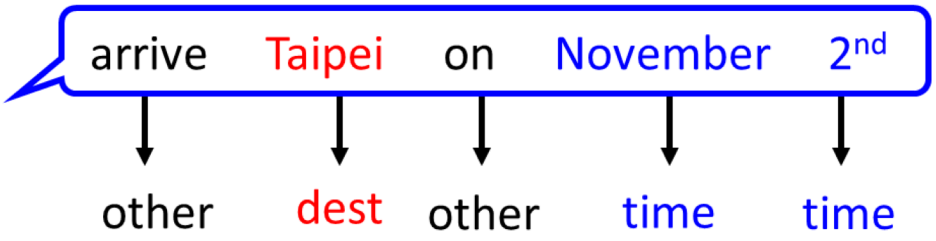
Example:
Sentiment analysis (our HW),
Document Classification

How to use BERT – Case 2

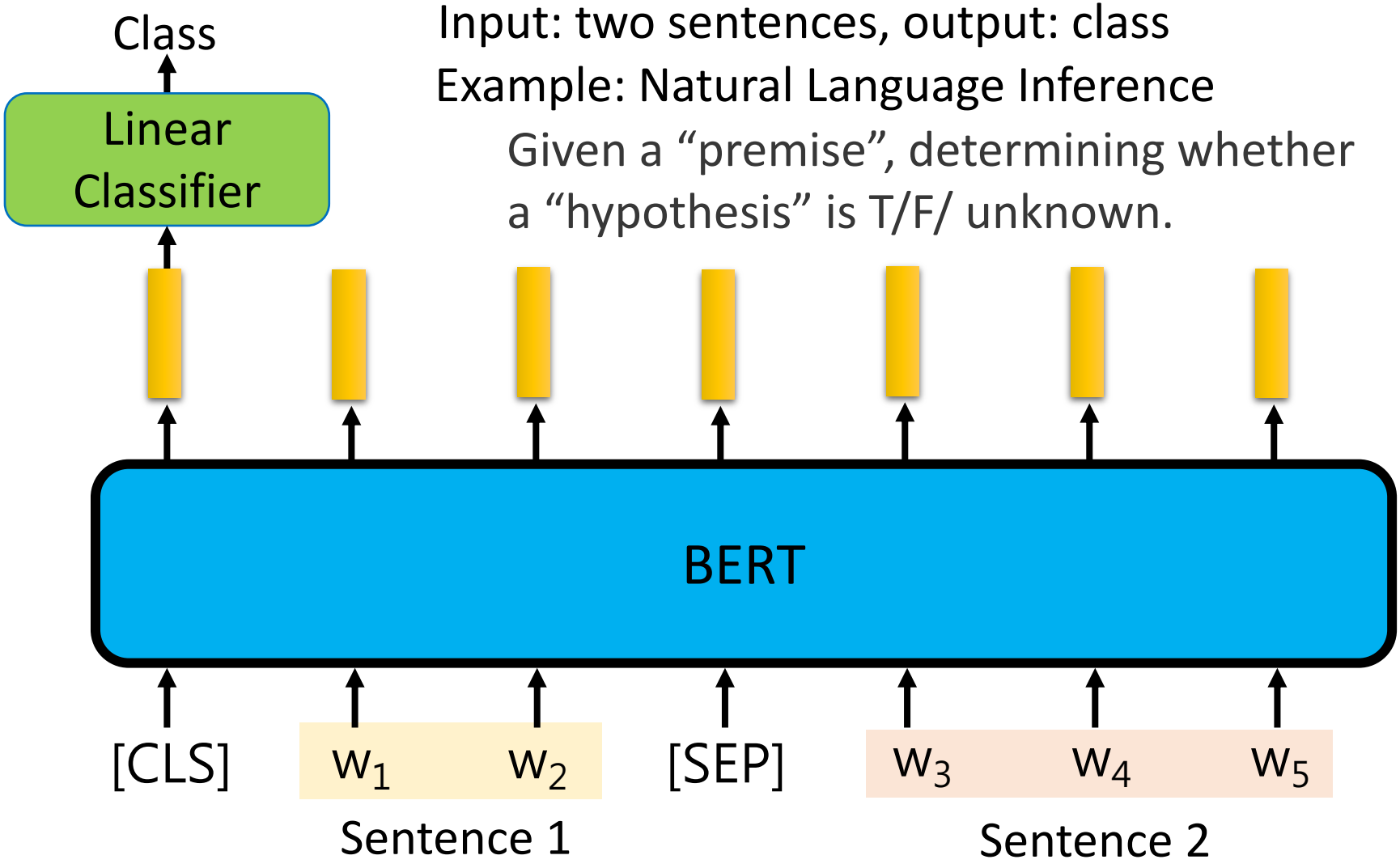


Input: single sentence, output: class of each word

Example: Slot filling



How to use BERT – Case 3



How to use BERT – Case 4

- Extraction-based Question Answering (QA) (E.g. SQuAD)

Document: $D = \{d_1, d_2, \cdots, d_N\}$

Query: $Q = \{q_1, q_2, \cdots, q_N\}$

output: two integers (s, e)

Answer: $A = \{q_s, \cdots, q_e\}$

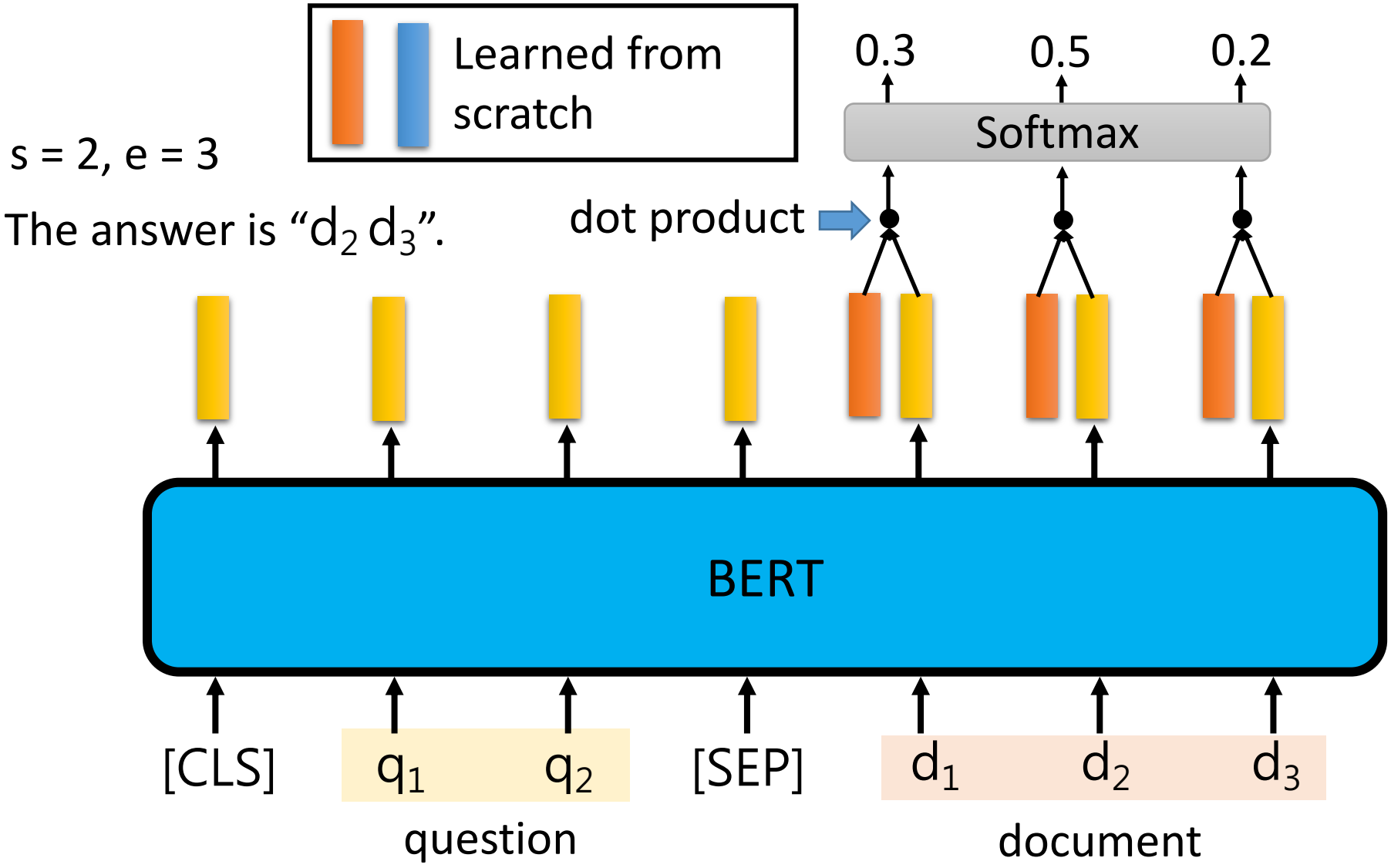
In meteorology, precipitation is any product of the condensation of 17 spheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain 77 atte 79 cations are called "showers".

What causes precipitation to fall?
gravity
$s = 17, e = 17$

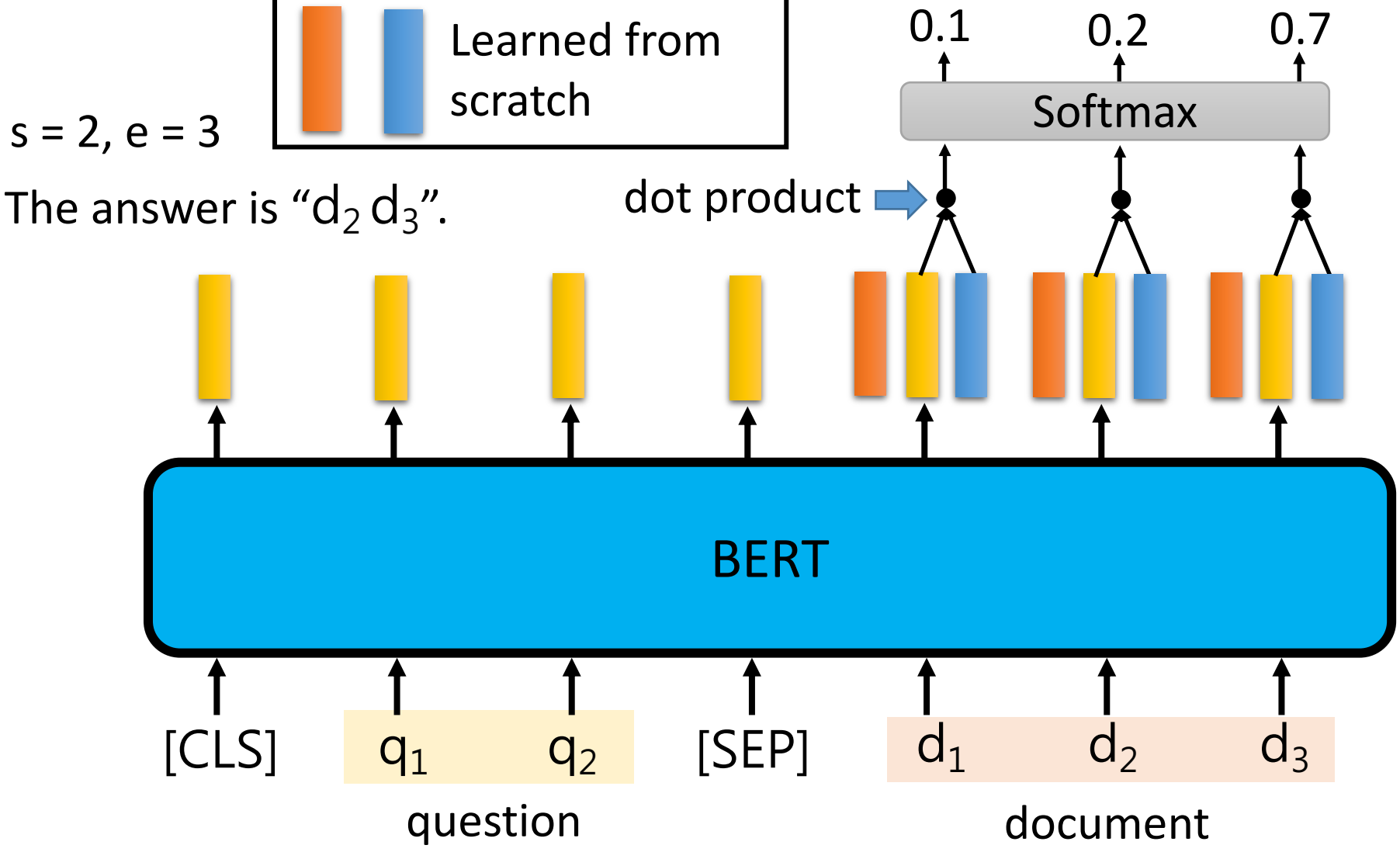
What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud
$s = 77, e = 79$

How to use BERT – Case 4



How to use BERT – Case 4



Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Mar 20, 2019	BERT + DAE + AoA (ensemble) *Joint Laboratory of HIT and iFLYTEK Research*	**87.147**	**89.474**
2 Mar 15, 2019	BERT + ConvLSTM + MTL + Verifier (ensemble) *Layer 6 AI*	86.730	89.286
3 Mar 05, 2019	BERT + N-Gram Masking + Synthetic Self-Training (ensemble) *Google AI Language* https://github.com/google-research/bert	86.673	89.147
4 May 21, 2019	XLNet (single model) *XLNet Team*	86.346	89.133
5 Apr 13, 2019	SemBERT(ensemble) *Shanghai Jiao Tong University*	86.166	88.886

SQuAD 2.0

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Apr 06, 2020	SA-Net on Albert (ensemble) *QIANXIN*	**90.724**	**93.011**
2 May 05, 2020	SA-Net-V2 (ensemble) *QIANXIN*	90.679	92.948
2 Apr 05, 2020	Retro-Reader (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694	90.578	92.978
3 Jul 31, 2020	ATRLP+PV (ensemble) *Hithink RoyalFlush*	90.442	92.877
3 May 04, 2020	ELECTRA+ALBERT+EntitySpanFocus (ensemble) *SRCB_DML*	90.442	92.839

BERT Implementation

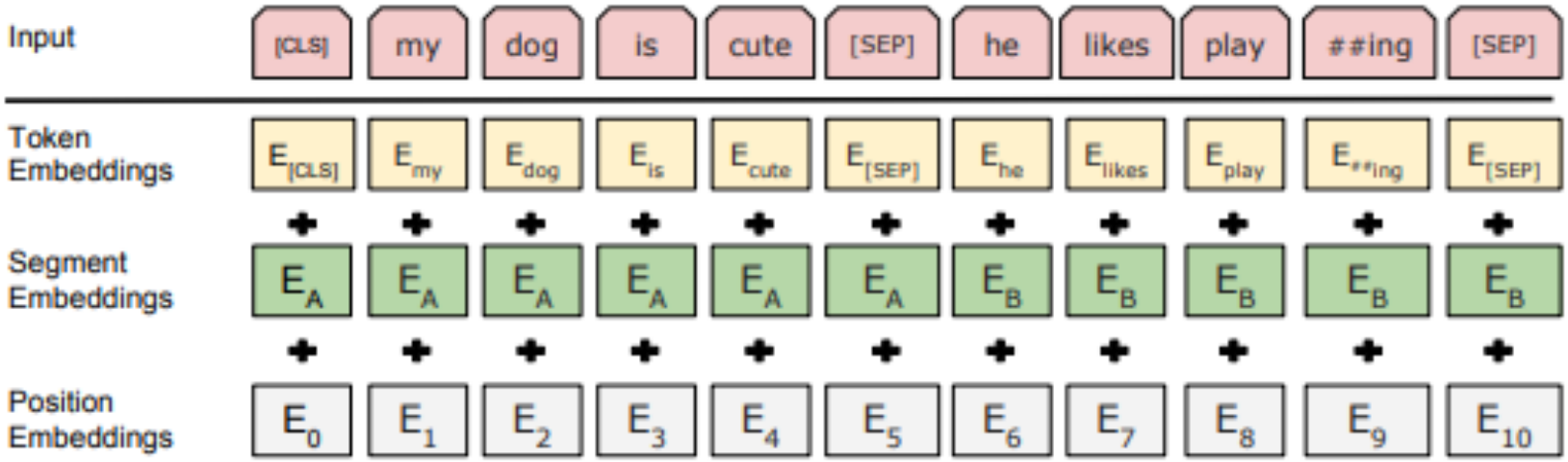


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

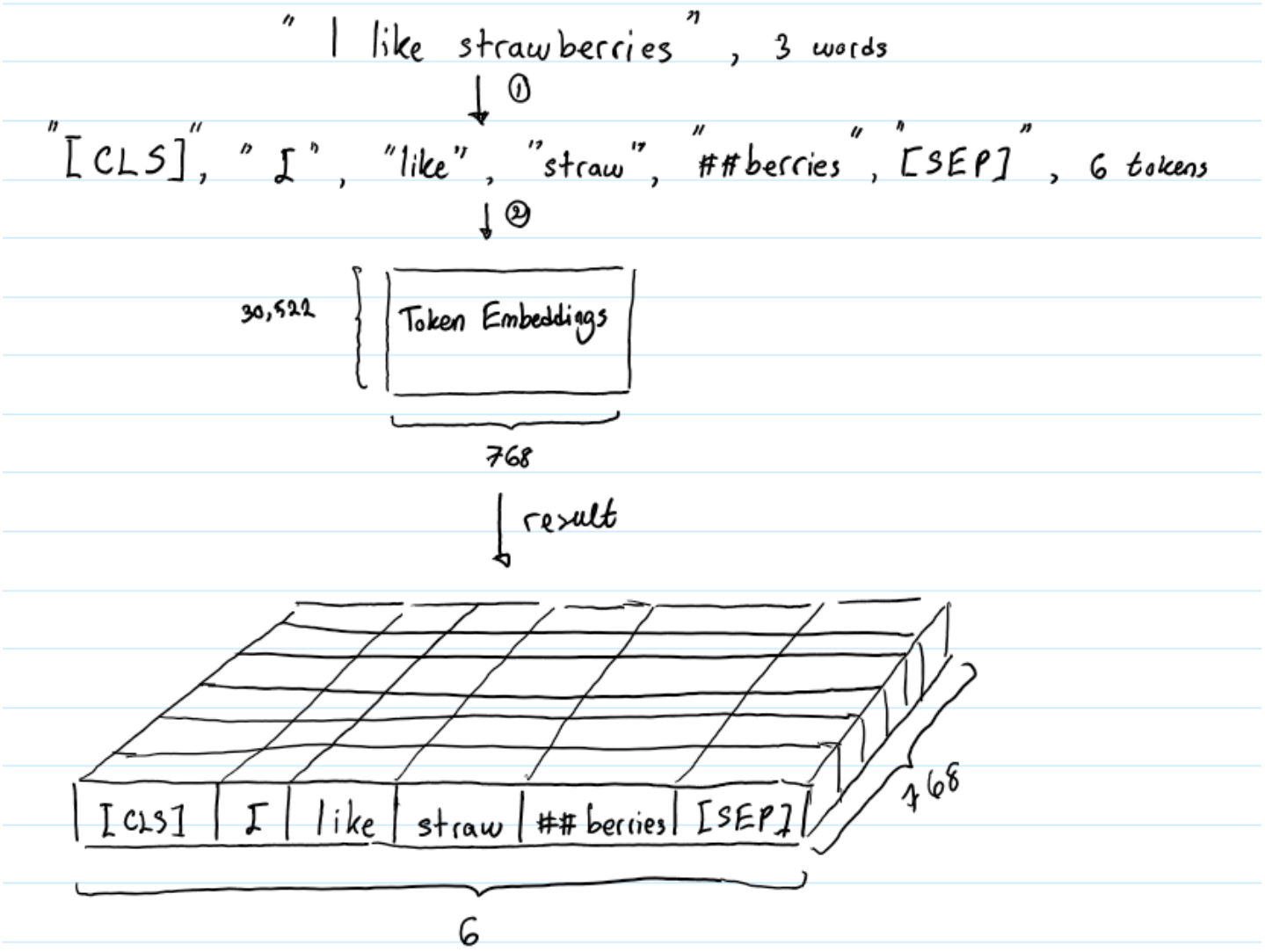
https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

Token Embeddings

- **Purpose**
  - The role of the Token Embeddings layer is to transform words into vector representations of fixed dimension.
- Method
  - WordPiece tokenization: 30,522 words
    - I like strawberries > [CLS], I, like, straw, ##berries, [SEP]
    - This is a data-driven tokenization method that aims to achieve a balance between **vocabulary size** and **out-of-vocab words**. This is way "strawberries" has been split into "straw" and "berries".

https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

Illustrative Example

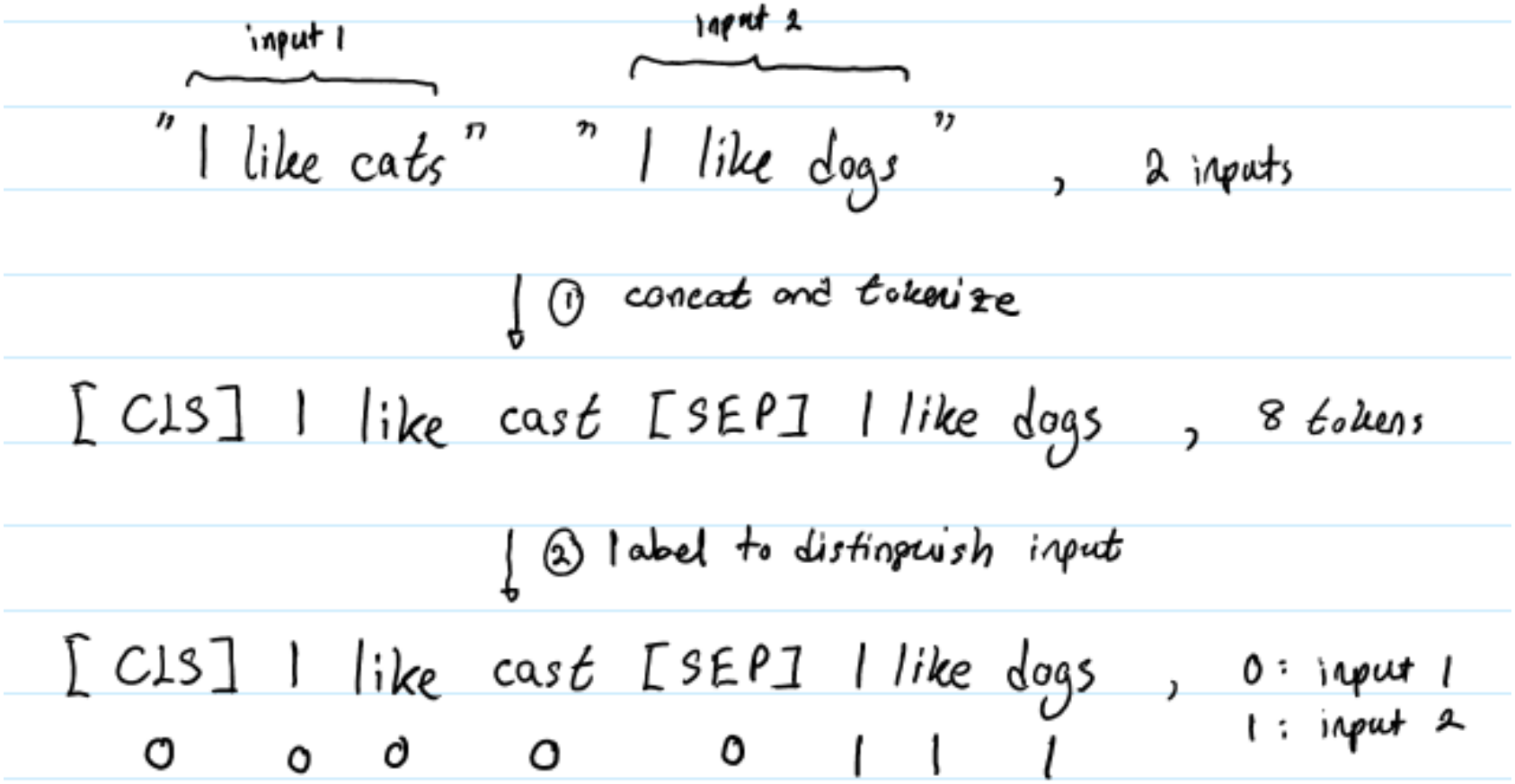


Segment Embeddings

- **Purpose**
  - BERT is able to solve NLP tasks that involve text classification given a pair of input texts.
  - How to distinguish the inputs in a given pair? Segment!
- Method
  - The Segment Embeddings layer only has 2 vector representations.
  - The first vector (index 0) is assigned to all tokens that belong to input 1 while the last vector (index 1) is assigned to all tokens that belong to input 2.

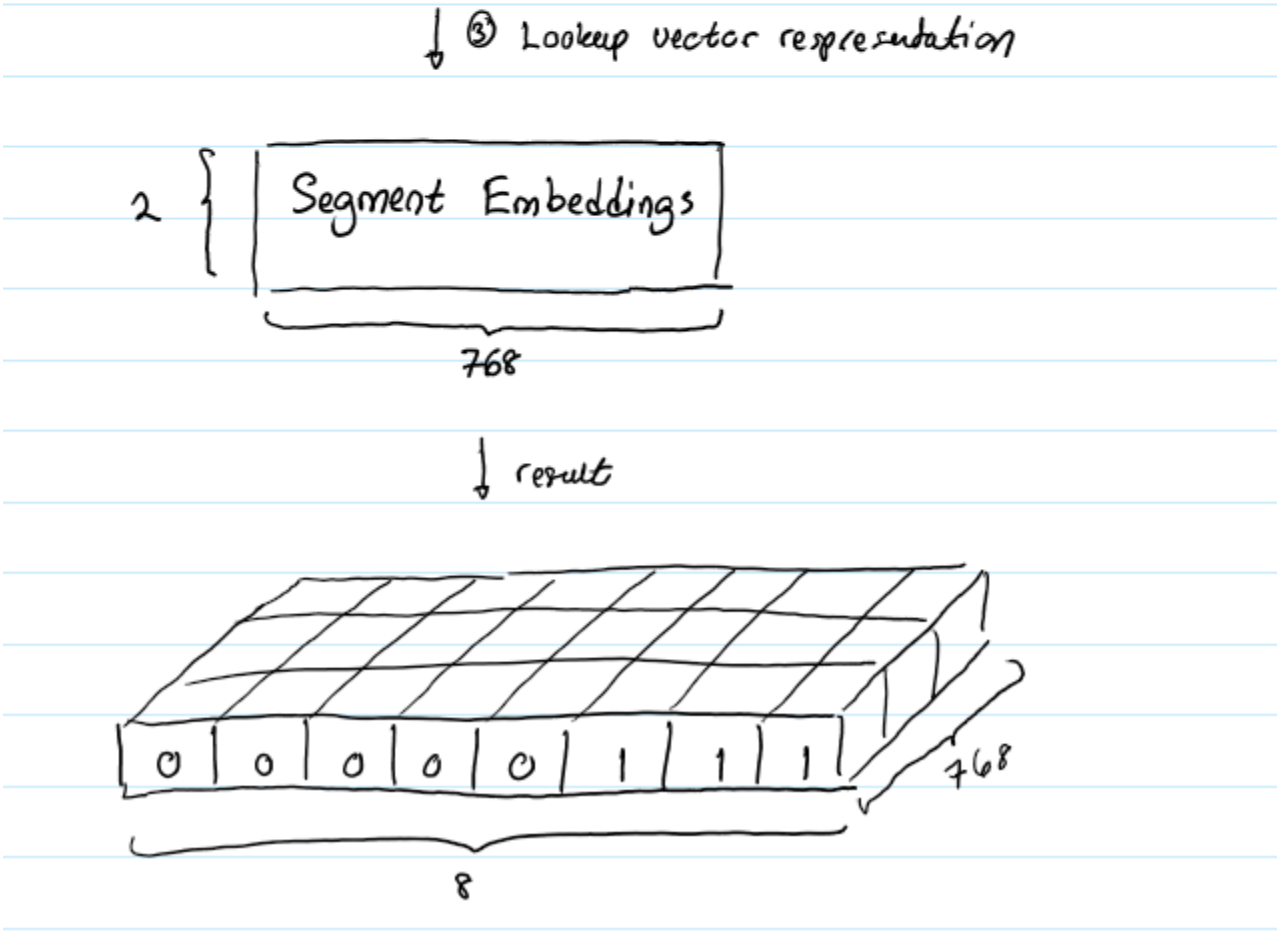
https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

Illustrative Example



https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

Illustrative Example



https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

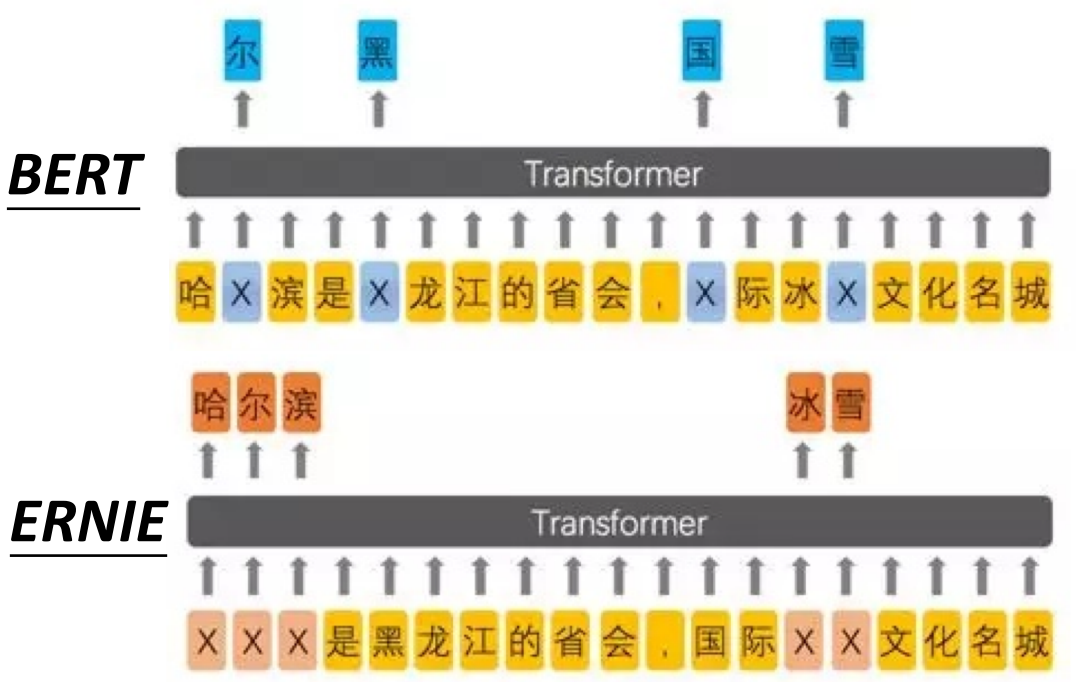
Position Embeddings

- **Purpose**
  - Transformers do not encode the sequential nature of their inputs.
- **Method**
  - BERT was designed to process input sequences of up to length 512. The authors incorporated the sequential nature of the input sequences by having BERT learn a vector representation for each position.
  - Position Embeddings layer is a lookup table of size (512, 768) where the first row is the vector representation of any word in the first position…

https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

Enhanced Representation through Knowledge Integration (ERNIE)

- Designed for Chinese



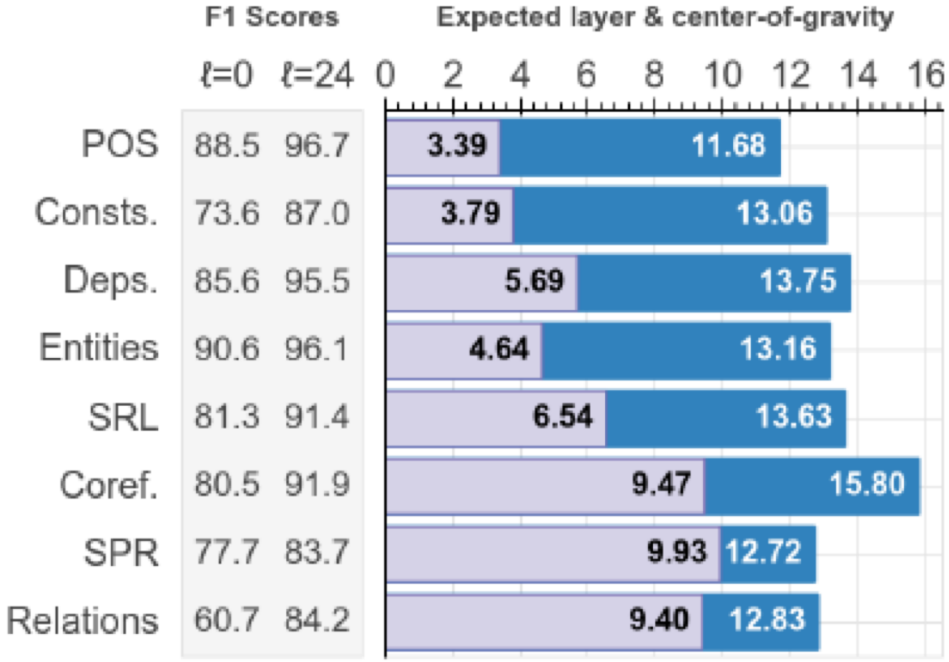
Source of image:
https://zhuanlan.zhihu.com/p/59436589

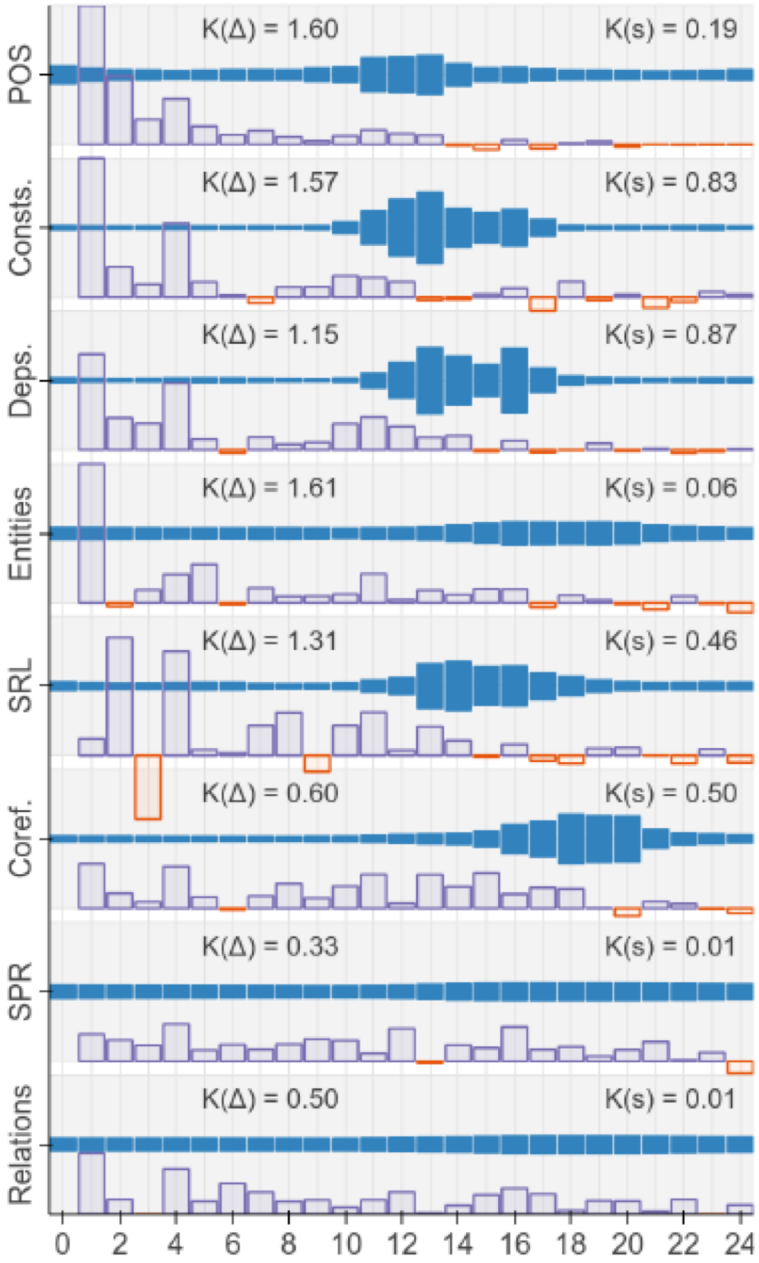
https://arxiv.org/abs/1904.09223

What does BERT learn?

BERT Rediscovers the Classical NLP Pipeline

Ian Tenney[1] **Dipanjan Das**[1] **Ellie Pavlick**[1,2]
[1]Google Research [2]Brown University
{iftenney,dipanjand,epavlick}@google.com





What does BERT learn?

- Scalar Mixing Weights

$$\mathbf{h}_{i,\tau} = \gamma_\tau \sum_{\ell=0}^{L} s_\tau^{(\ell)} \mathbf{h}_i^{(\ell)}$$

- Center-of-Gravity

$$\bar{E}_s[\ell] = \sum_{\ell=0}^{L} \ell \cdot s_\tau^{(\ell)}$$

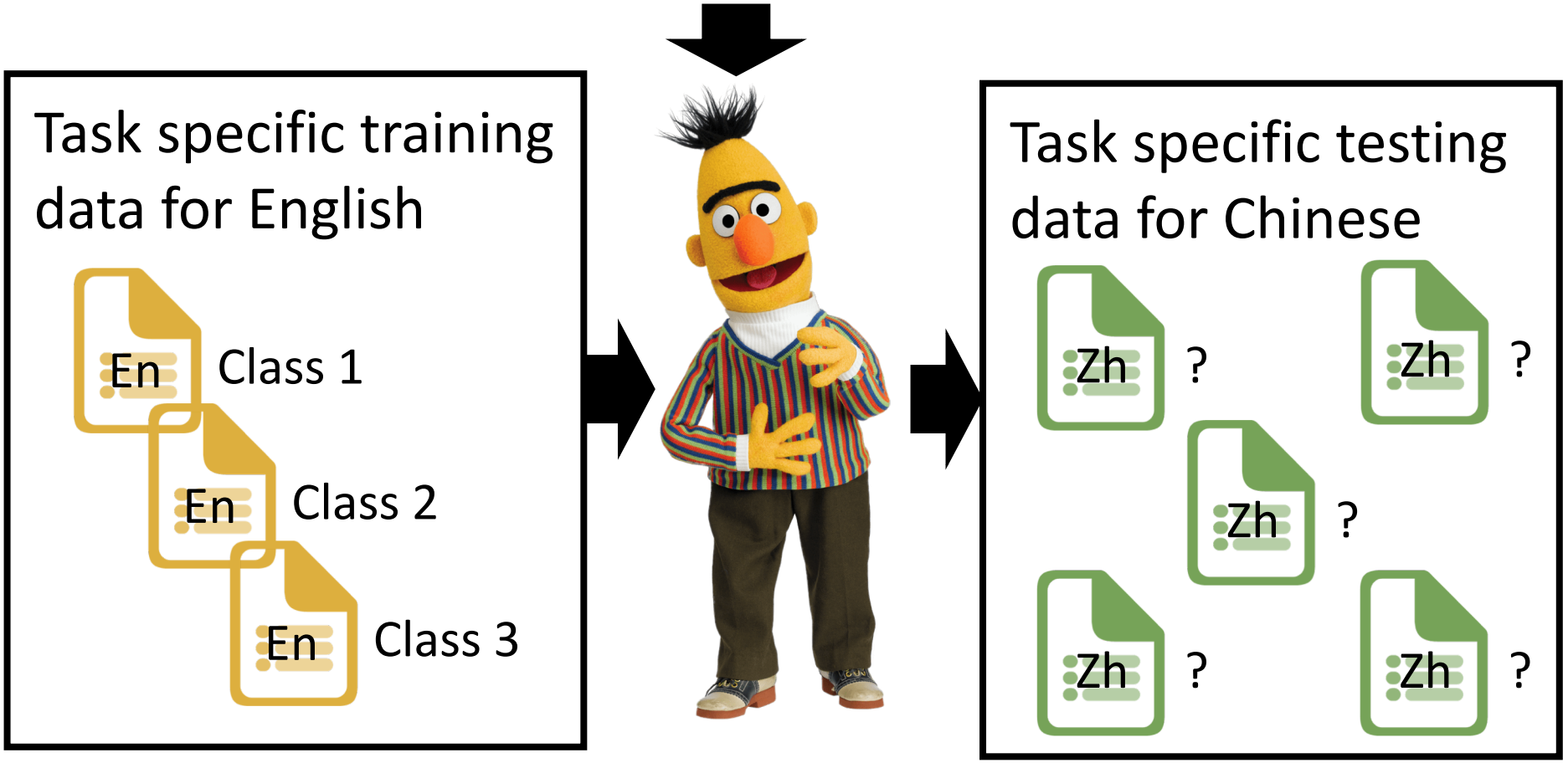
- Cumulative Scoring $\Delta_\tau^{(\ell)} = \text{Score}(P_\tau^{(\ell)}) - \text{Score}(P_\tau^{(\ell-1)})$
- Expected Layer

$$\bar{E}_\Delta[\ell] = \frac{\sum_{\ell=1}^{L} \ell \cdot \Delta_\tau^{(\ell)}}{\sum_{\ell=1}^{L} \Delta_\tau^{(\ell)}}$$

https://arxiv.org/abs/1904.09077

Multilingual BERT

Trained on 104 languages



- BERT is a kind of **language model**.

- Generally, language model aims to model following distribution:

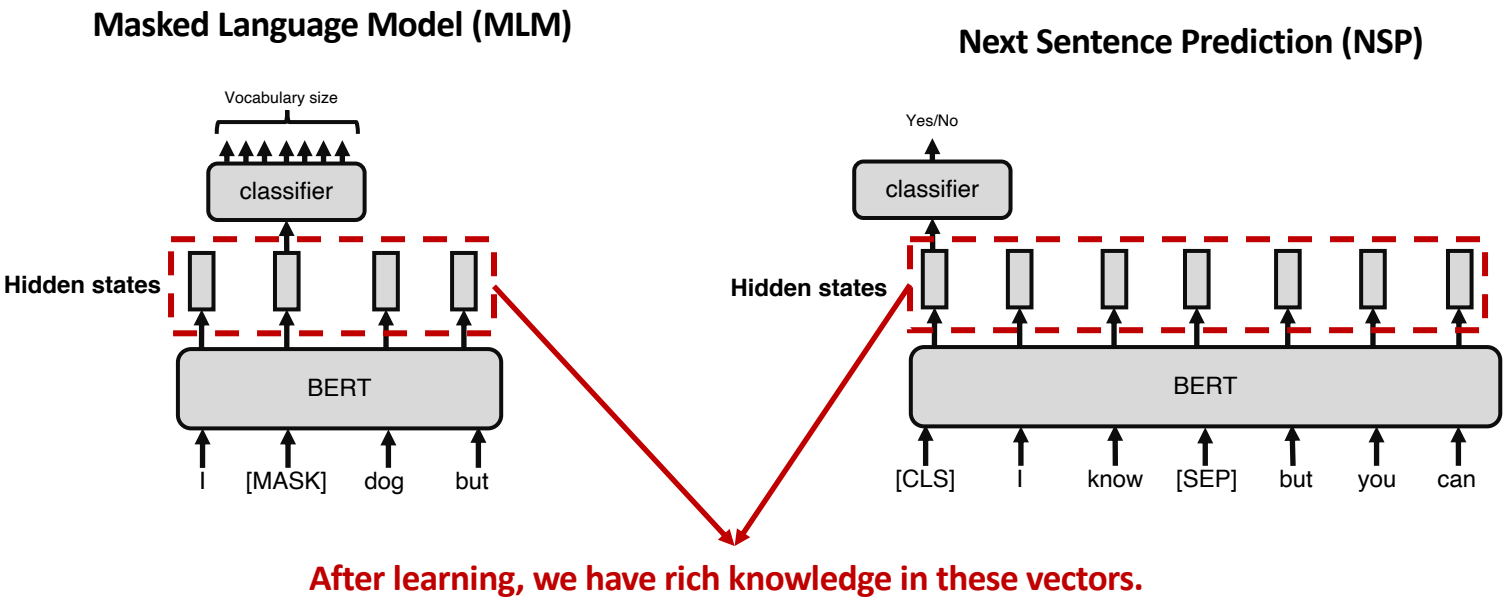
$$p(w_1, w_2, \dots, w_n)$$

-> How likely the sentence $[w_1, w_2, \dots, w_n]$ happens in real world?

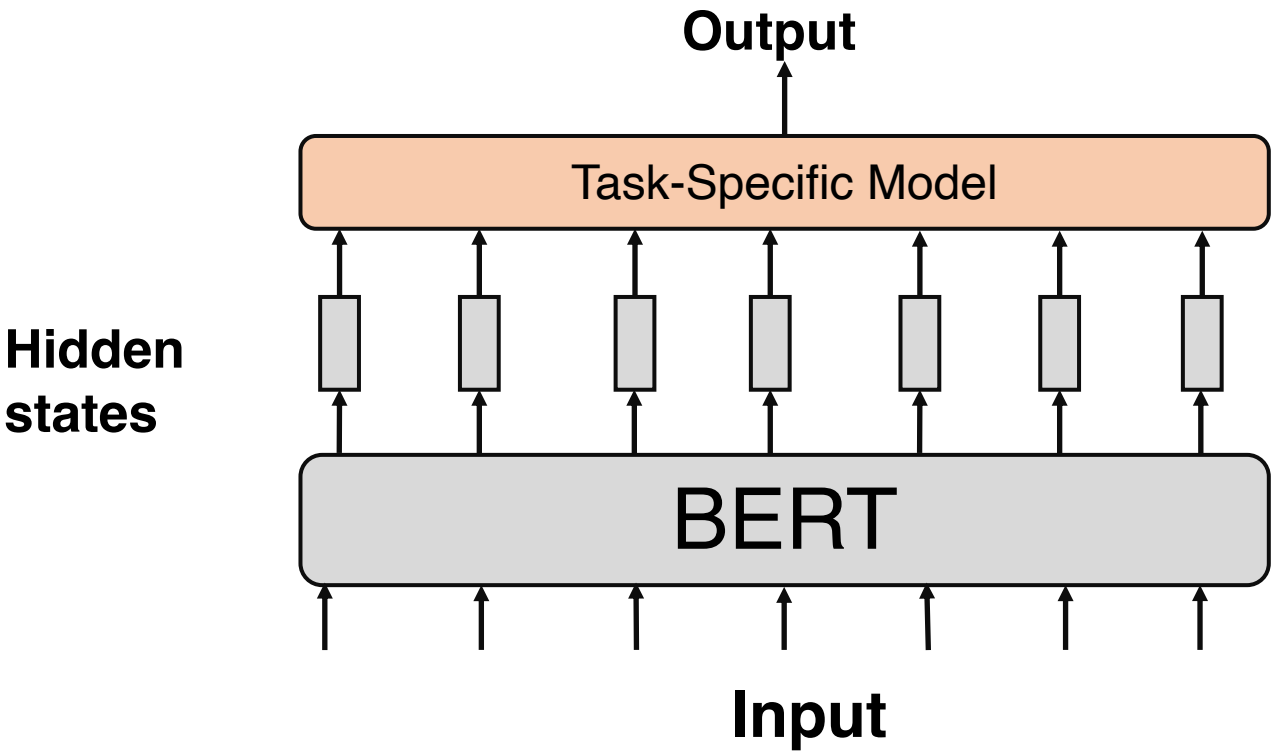
- Example:

$$p(\text{"I"}, \text{"eat"}, \text{"an"}, \text{"apple"}, \text{"."}) > p(\text{"I"}, \text{"apple"}, \text{"an"}, \text{"eat"}, \text{"!"})$$

- There are many ways to learn a language model, and BERT is one of them.
- BERT learns the knowledge with following two tasks:



- Next, we can use BERT to handle down-stream tasks with prior knowledge



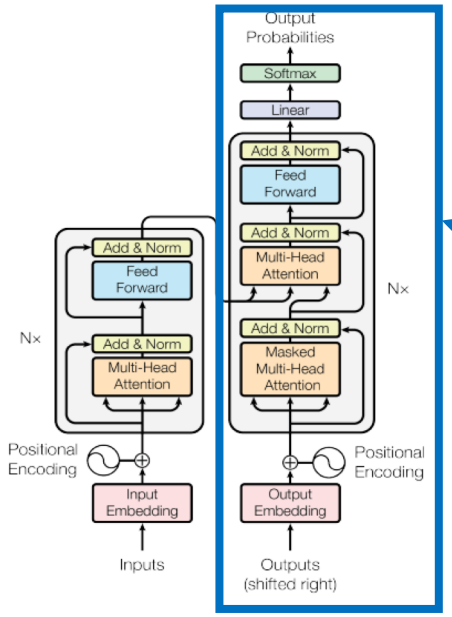
- Why BERT is important?

1. It does **not need target** for training, which means data can be derived everywhere.
2. The model can learn **useful general information**, e.g., syntactic structure, context semantic, etc.
3. The reusable property can **reduce develop cost.** It takes 64 TPUs with 4 days to train the BERT. (7000$)

NOTE: BERT is a general model. We still need as many task-specific labeled data as possible to get good results.

https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Generative Pre-Training (GPT)



Transformer Decoder

BERT (340M)

ELMO (94M)

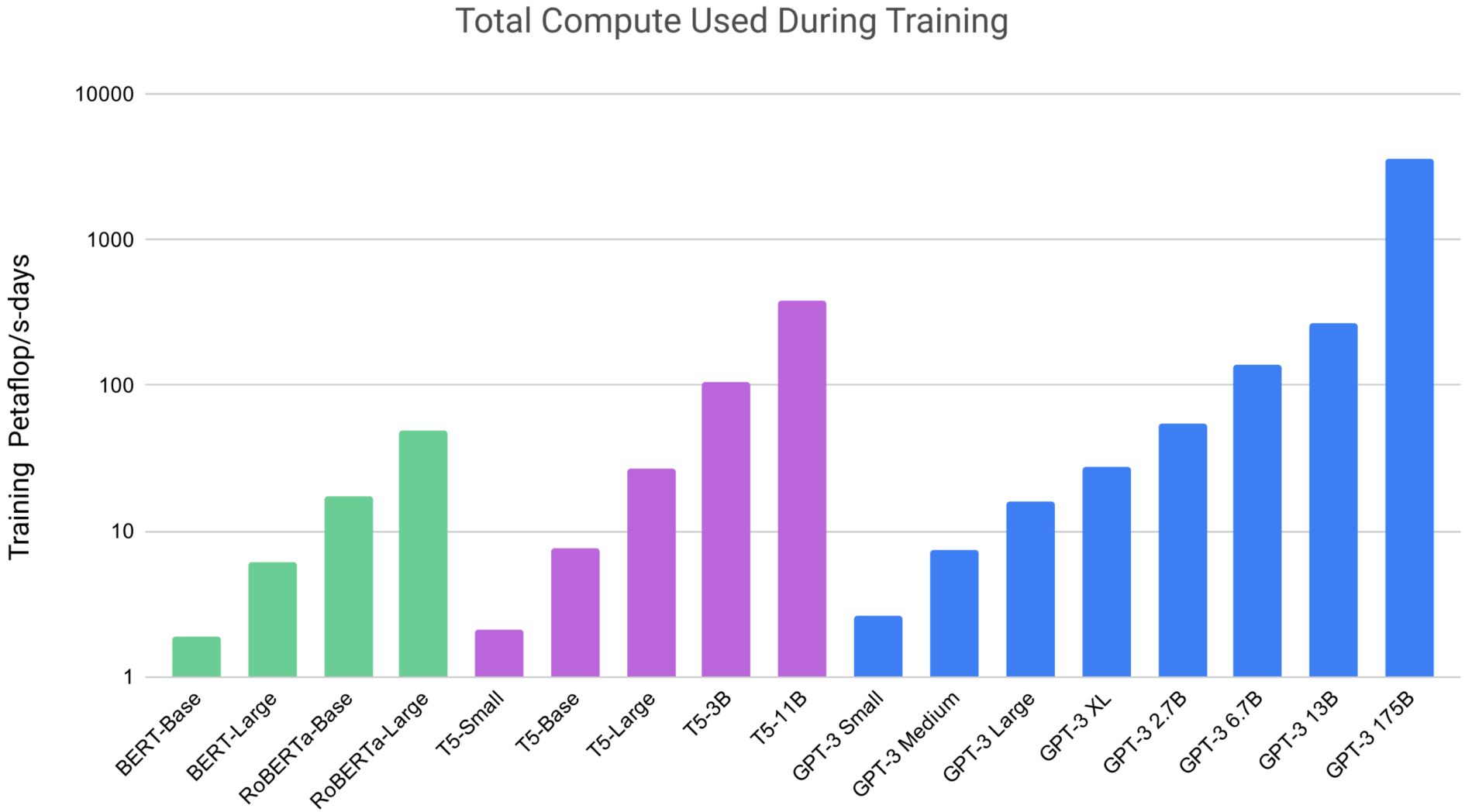
GPT-2 (1542M)

Source of image: https://huaban.com/pins/1714071707/

one thousand trillion floating-point operations per second per day for 3,640 days

GPT-3

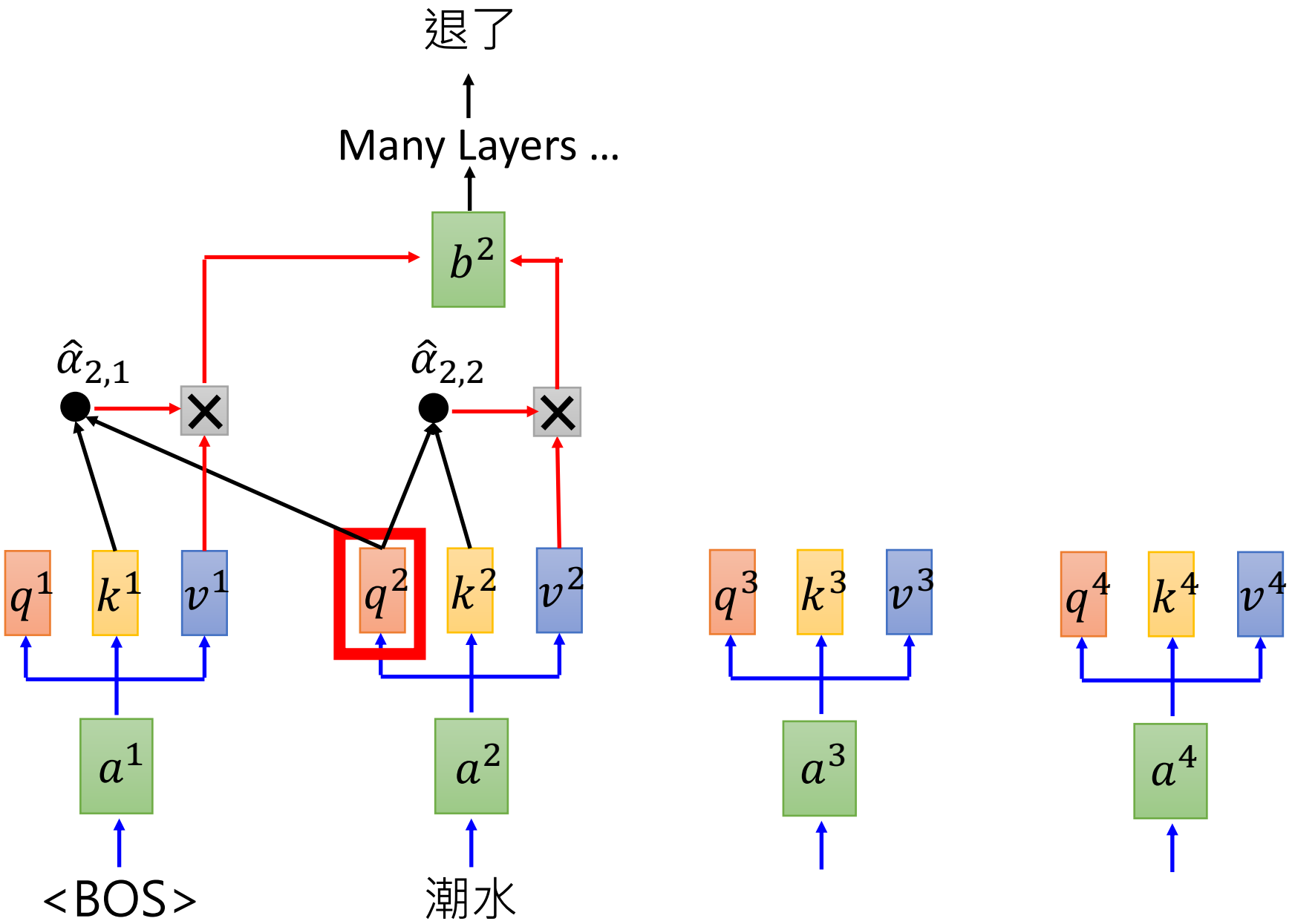
175B



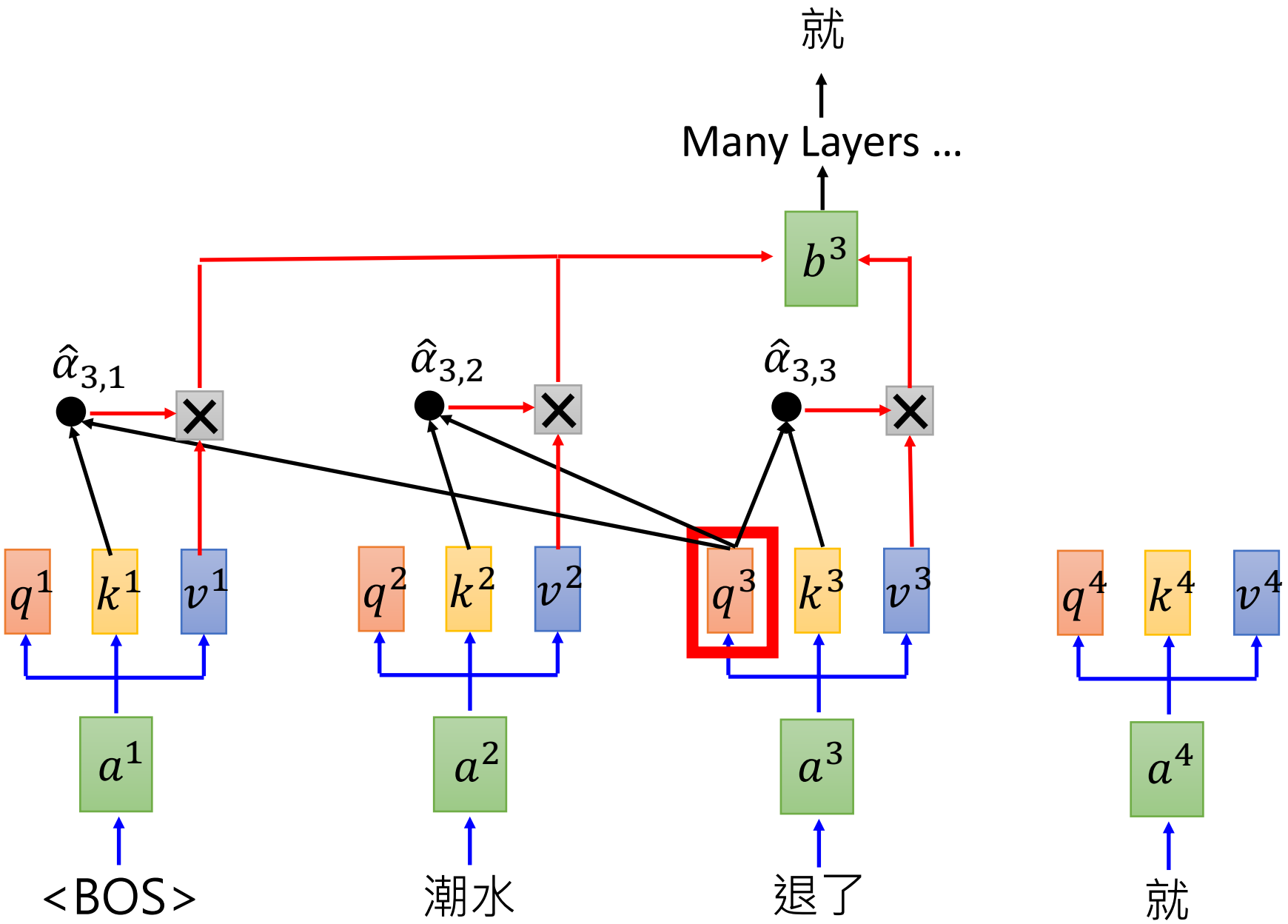
Demo

- https://maraoz.com/2020/07/18/openai-gpt3/
- https://twitter.com/sharifshameem/status/1282676454690451457?s=20
- https://forms.office.com/Pages/ResponsePage.aspx?id=VsqMpNrmTkioFJyEllK8s0v5E5gdyQhOuZCXNuMR8i1UQjFWVTVUVEpGNkg3U1FNRDVVRFg3U0w4Vi4u

Generative Pre-Training (GPT)



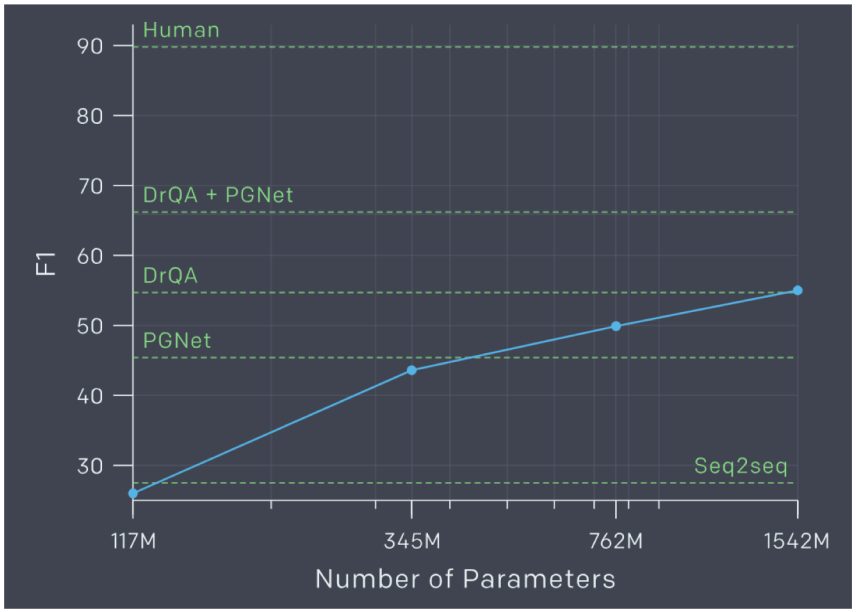
Generative Pre-Training (GPT)



Zero-shot Learning?

- ***Reading Comprehension***

$d_1, d_2, \cdots, d_N,$
"Q:", $q_1, q_2, \cdots, q_N,$
"A:"



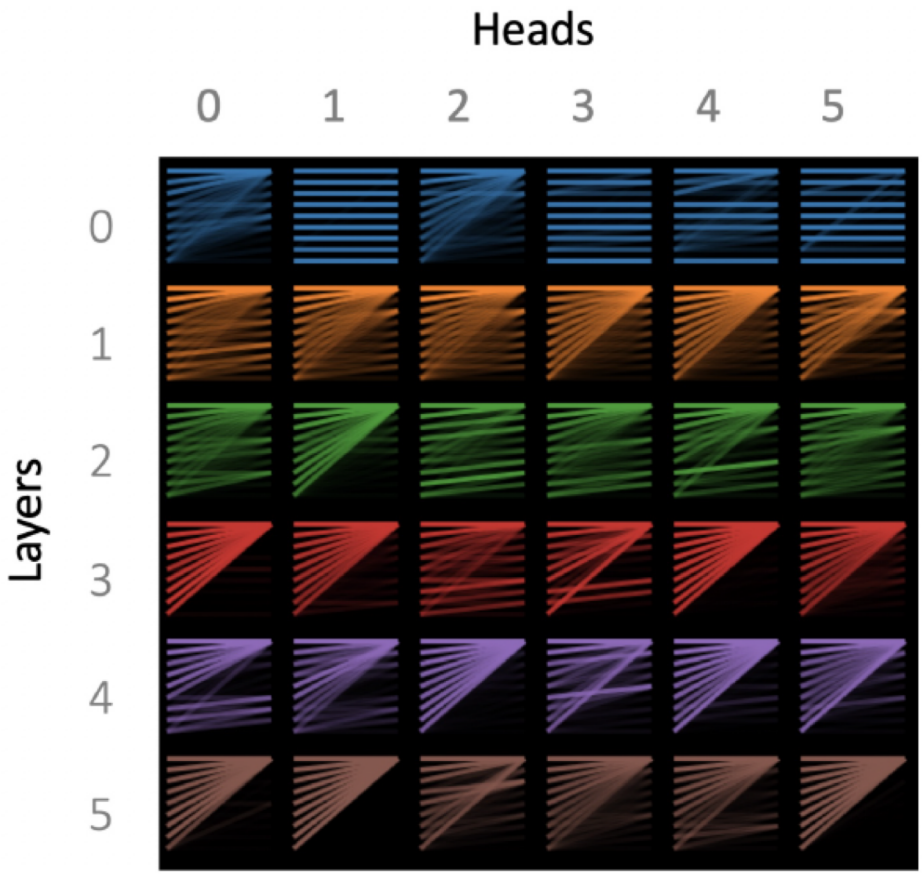
- ***Summarization*** $d_1, d_2, \cdots, d_N$,"TL;DR:"

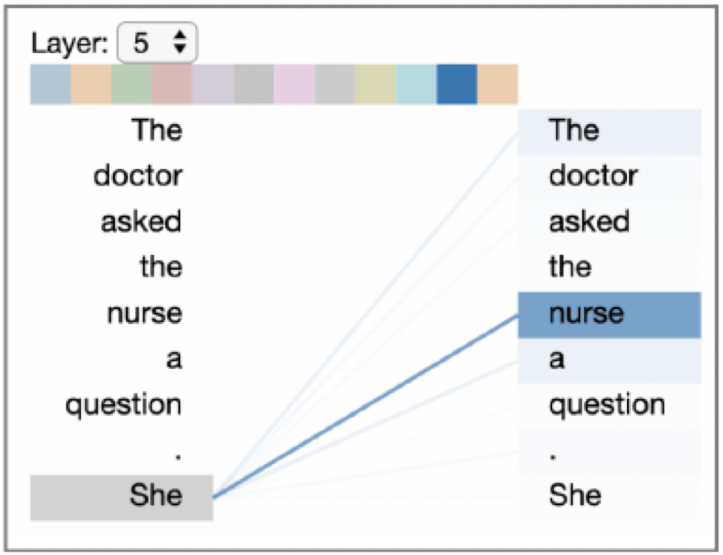
- ***Translation***

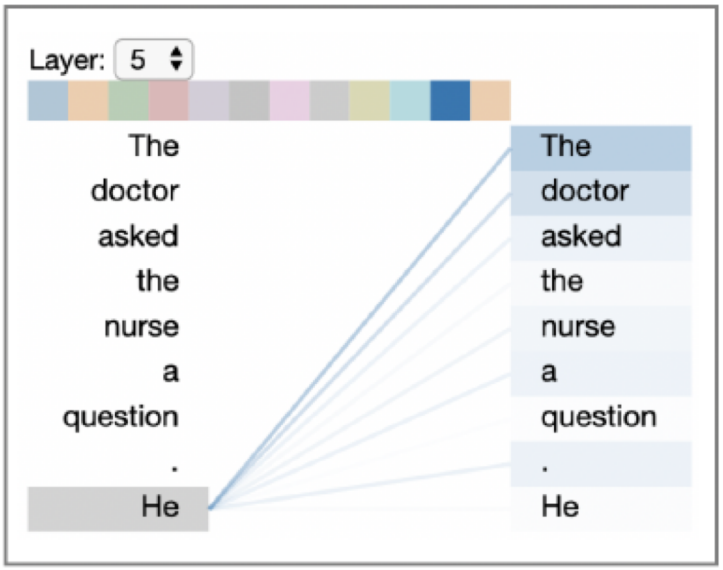
English sentence 1	=	French sentence 1
English sentence 2	=	French sentence 2
English sentence 3	=	

Visualization

https://arxiv.org/abs/1904.02679
(The results below are from GPT-2)







EM PROMPT
-WRITTEN)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL
MPLETION
(MACHINE-
10 TRIES)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

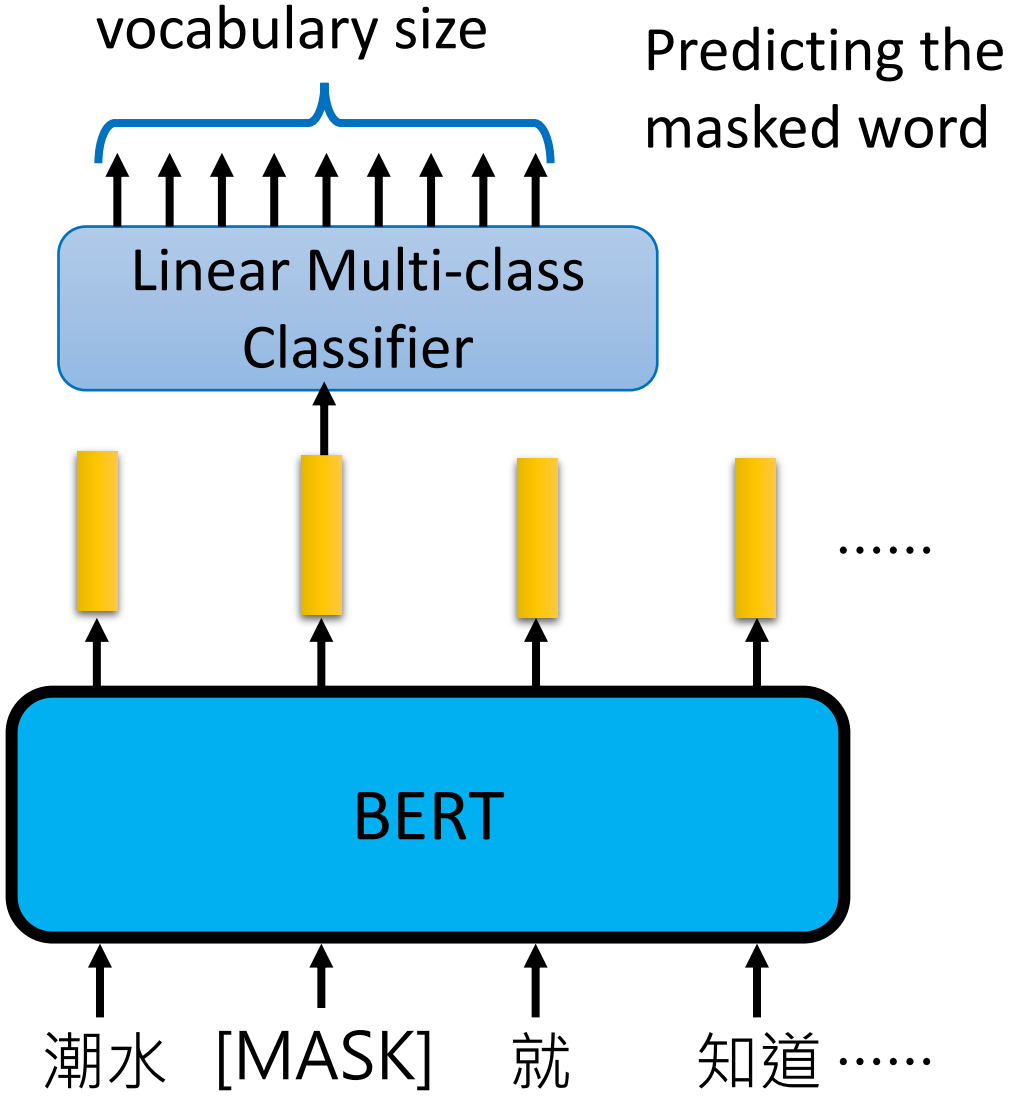
https://talktotransformer.com/

Credit: Greg Durrett

Training of BERT

- Approach 1:

  Masked LM



XLNET

XLNet: Generalized Autoregressive Pretraining for Language Understanding

Zhilin Yang[*1], **Zihang Dai**[*12], **Yiming Yang**[1], **Jaime Carbonell**[1],
Ruslan Salakhutdinov[1], **Quoc V. Le**[2]
[1]Carnegie Mellon University, [2]Google Brain
{zhiliny,dzihang,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

https://github.com/zihangdai/xlne

Abstract

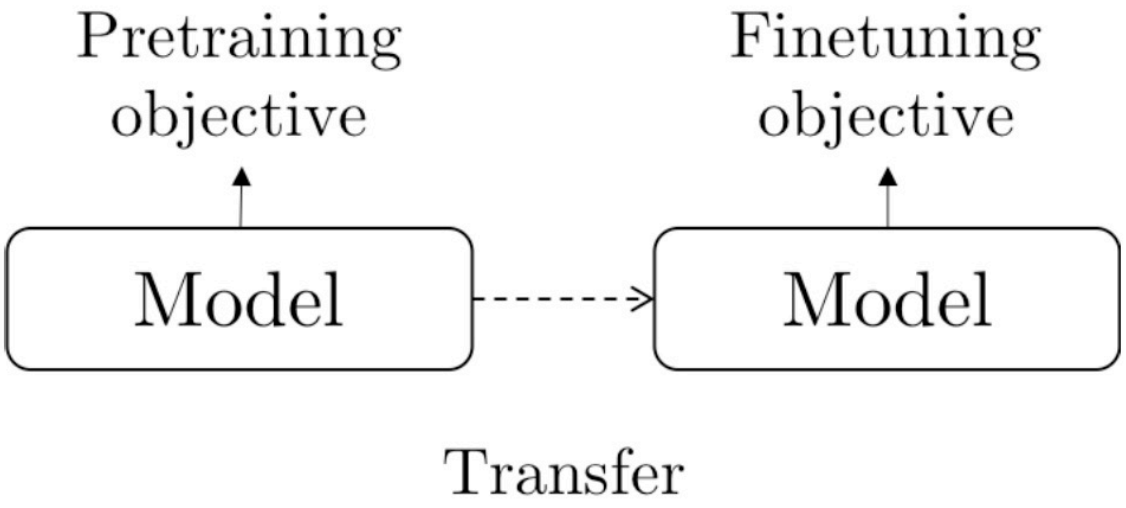
- Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context
- Based on autoregressive language modeling
- Enables learning bidirectional contexts
- Overcomes the limitations of BERT (longer text sequence)
- XLNet outperforms BERT on 20 tasks, often by a large margin

Learning From Unlabeled Data

Unlabeled data
Abundant (1000x more), accessible

Labeled data
Scarce, expensive

Unsupervised Pretraining

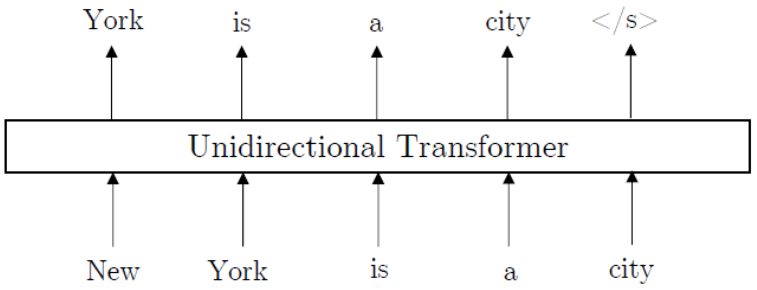


Related Work

- RBMs (Salakhutdinov et al 2007), Autoencoders (Vincent et al 2008), Jigsaw (Noroozi and Favaro 2016), GANs (Donahue and Simonyan 2019)...
- word2vec (Mikolov et al 2013), GloVe (Pennington et al 2014)
- Semi-supervised sequence learning (Dai and Le 2015), ELMo (Peters et al 2017), CoVe (McCann et al 2017), GPT (Radford et al 2018), BERT (Devlin et al 2018)

Two Objectives for Pretraining

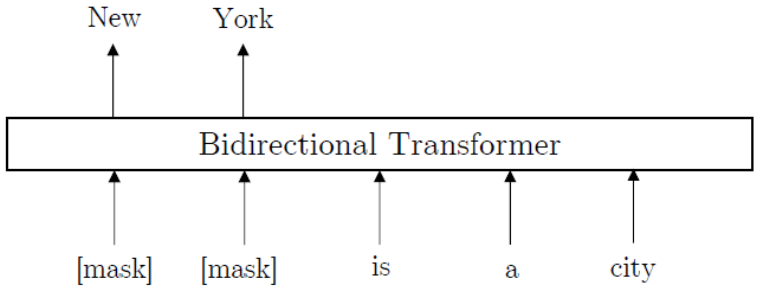
Auto-regressive (AR) language modeling



$$\log p(\mathbf{x}) = \sum_{t=1}^{T} \log p(x_t | \mathbf{x}_{<t})$$

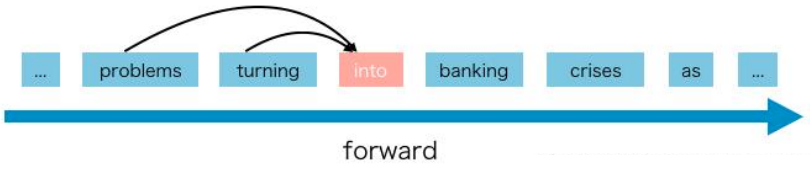
Not able to model bidirectional context. ☹

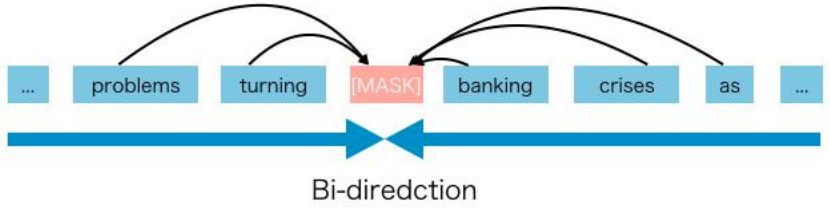
(Denoising) Auto-encoding (AE)



$$\log p(\bar{\mathbf{x}} | \hat{\mathbf{x}}) = \sum_{t=1}^{T} \text{mask}_t \log p(x_t | \hat{\mathbf{x}})$$

Predicted tokens are independent of each other. ☹
[mask] is not used during finetuning. ☹





New Objective: Permutation Language Modeling

- Sample a factorization order
- Determine the attention masks based on the order
- Optimize a standard language modeling objective

$$\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T}\log p(x_{z_t}|\mathbf{x}_{\mathbf{z}<t})\right]$$

- Benefits:
  - Autoregressive, avoiding disadvantages of AE
  - Able to model bidirectional context

Examples

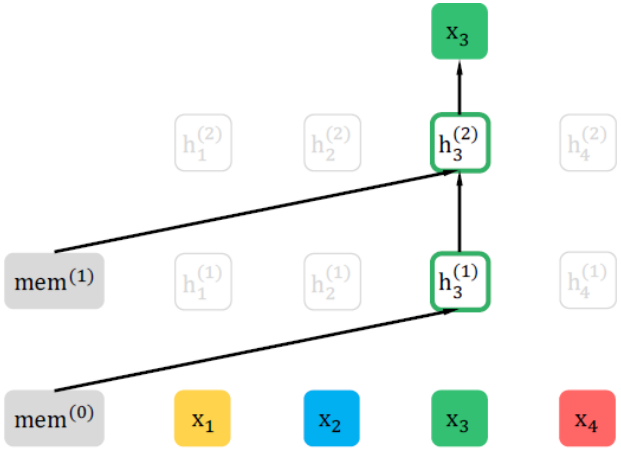
Factorization order: New York is a city

$$
\begin{aligned}
&\text{P(New York is a city)} \\
=\ &\text{P(New)} * \text{P(York} \mid \text{New)} * \text{P(is} \mid \text{New York)} * \text{P(a} \mid \text{New York is)} * \text{P(city} \mid \text{New York is a)}
\end{aligned}
$$

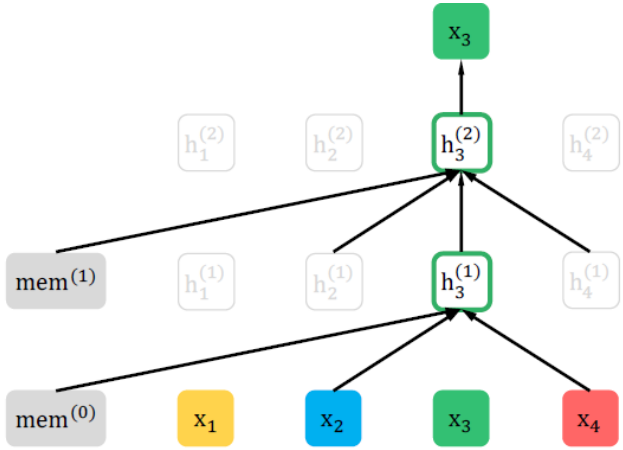
Factorization order: city a is New York

$$
\begin{aligned}
&\text{P(New York is a city)} \\
=\ &\text{P(city)} * \text{P(a} \mid \text{city)} * \text{P(is} \mid \text{city a)} * \text{P(New} \mid \text{city a is)} * \text{P(York} \mid \text{city a is New)}
\end{aligned}
$$

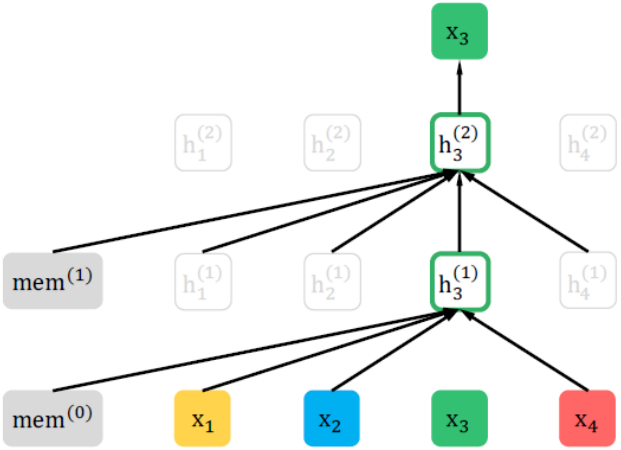
Sequence order is not shuffled.
Attention masks are changed to reflect factorization order.



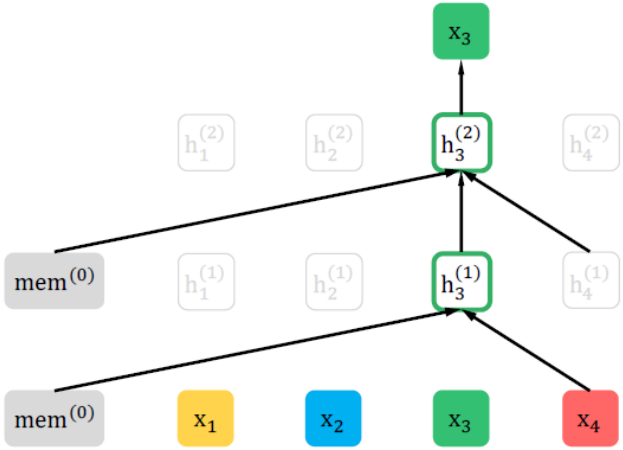
Factorization order: 3 → 2 → 4 → 1



Factorization order: 2 → 4 → 3 → 1



Factorization order: 1 → 4 → 2 → 3



Factorization order: 4 → 3 → 1 → 2

Comparing XLNet and BERT Objectives

BERT objective (auto-encoding)

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city})$$

New and York are independent. ☹

XLNet objective (auto-regressive)

$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{New}, \text{is a city})$$

$$\text{or } \mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{York}, \text{is a city}) + \log p(\text{York} \mid \text{is a city})$$

Able to model dependency between New and York. ☺
Able to model bidirectional context. ☺
Factorize the joint probability using a product rule that holds universally.

Reparameterization

Standard Parameterization

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{\mathbf{z}_{<t}}) = \frac{e(x)^\top h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})}{\sum_{x'} e(x')^\top h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})}$$

h does not contain the position of the target.

$$p(X_3 = \text{is} \mid \text{New York}) = p(X_4 = \text{is} \mid \text{New York}) = p(X_5 = \text{is} \mid \text{New York})$$

Reduced to predicting a bag of words.

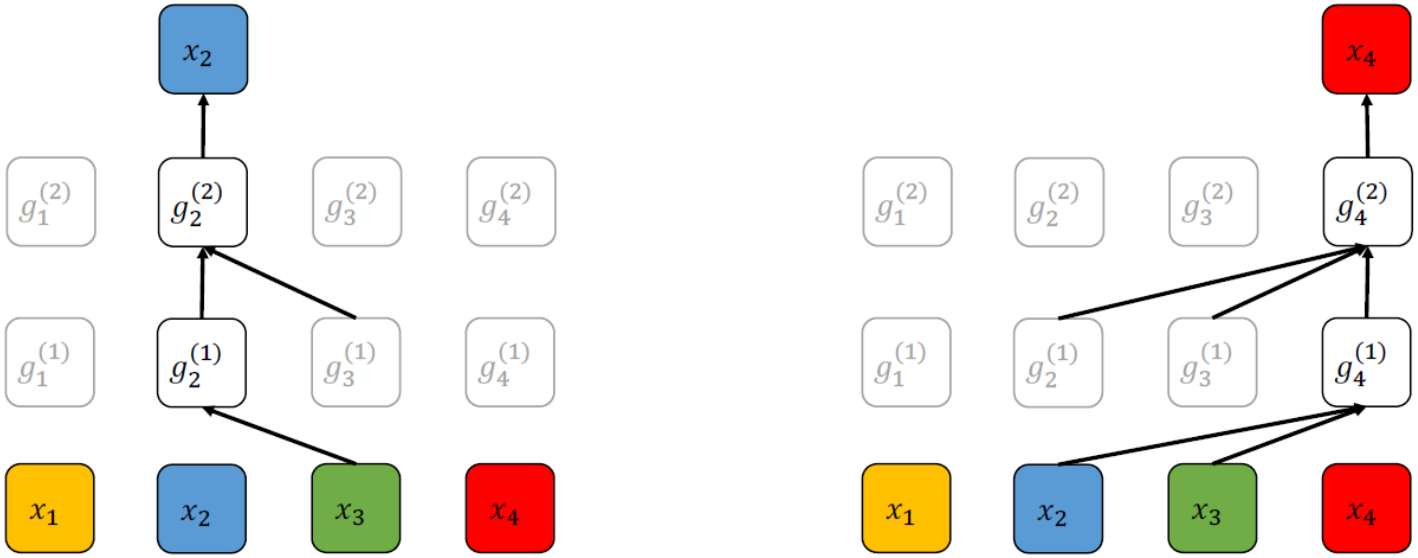
Solution: condition the distribution on the position.

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{\mathbf{z}_{<t}}) = \frac{e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, \boxed{z_t})}{\sum_{x'} e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, \boxed{z_t})}$$

"Stand at" z_t and predict self

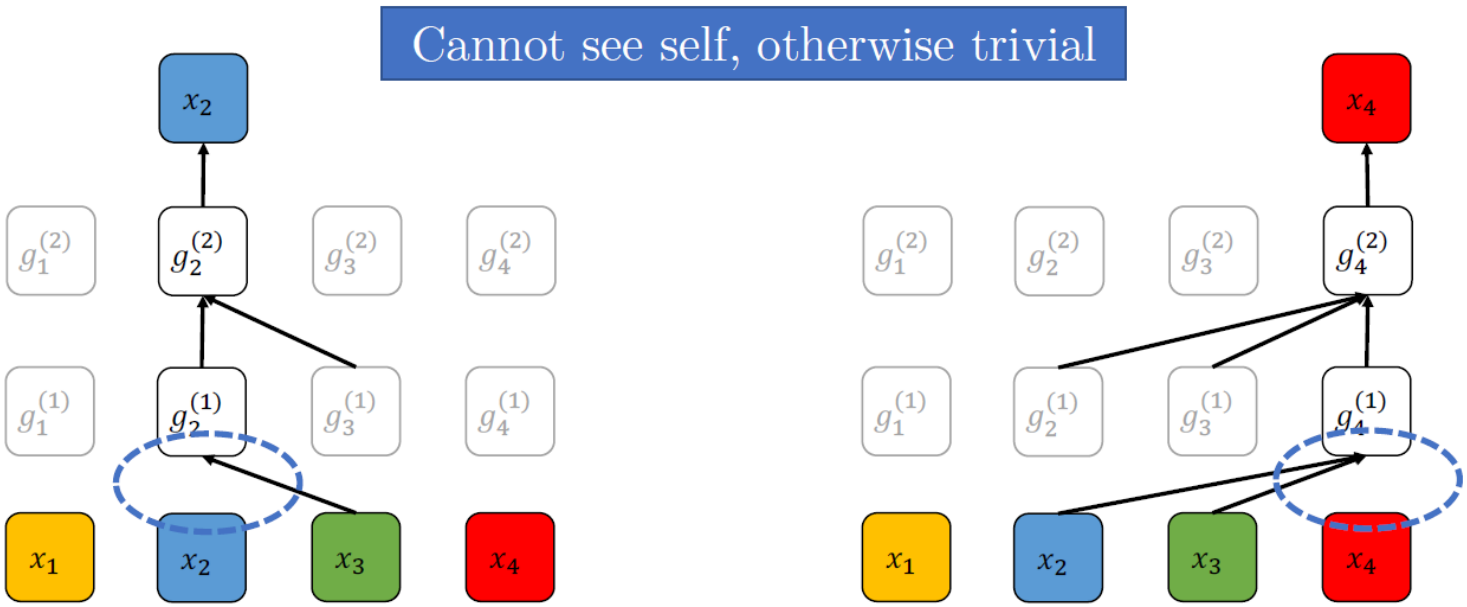
How to formulate features g

Let $g_i^{(l)}$ denote the feature of the i-th token on layer l
Suppose the factorization order is 3 2 4 1



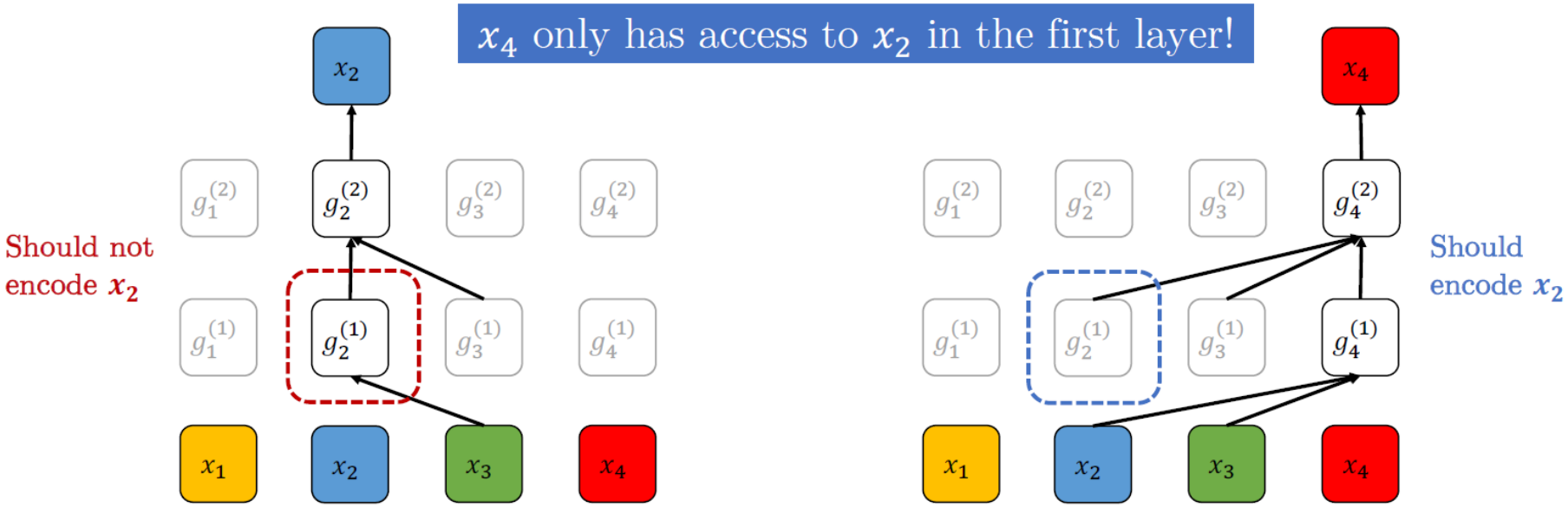
How to formulate features g

Let $g_i^{(l)}$ denote the feature of the i-th token on layer l
Suppose the factorization order is 3 2 4 1



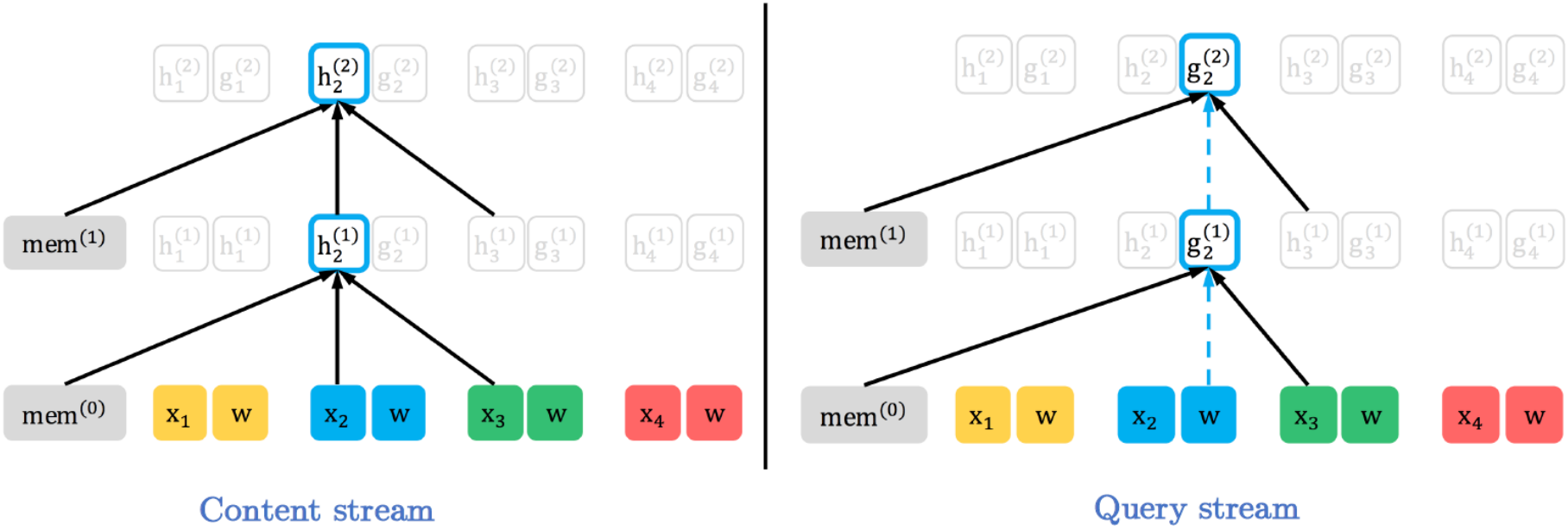
How to formulate features g

Let $g_i^{(l)}$ denote the feature of the i-th token on layer l
Suppose the factorization order is 3 2 4 1

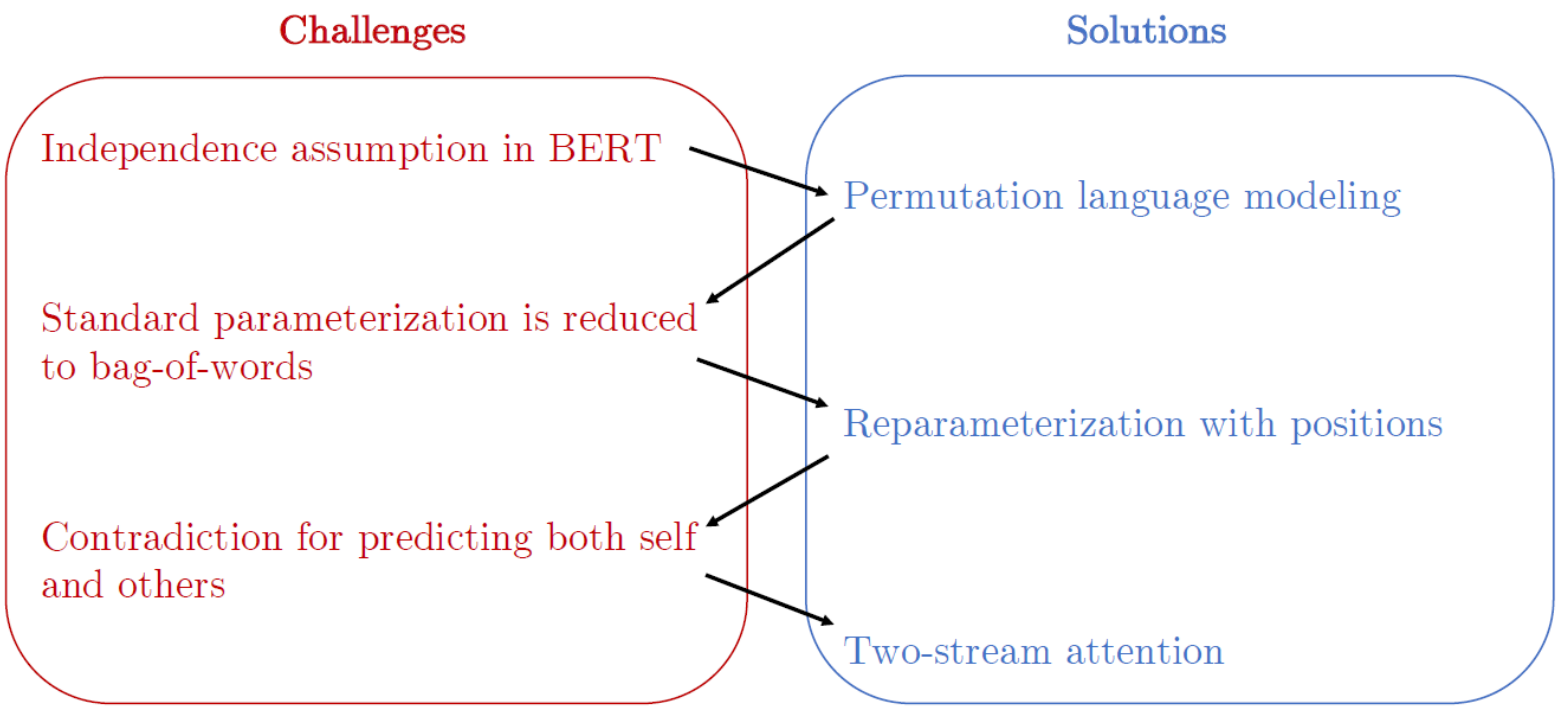


Two-Stream Attention

- Factorization order: 3, 2, 4, 1



Summarizing XLNet



Bridging the Gap between LM and Pretraining

- XLNet connects LM and pretraining
- Benefits
  - Taking the advantages of LM, while being able to model bidirectional contexts
  - Progress in LM directly connected to target tasks
  - LM research relies on less computational resources
- Incorporating Transformer-XL into pretraining

Unifying Encoders and Decoders

- Content stream: similar to encoder
- Query stream: similar to decoder
- Applications:
  - Seq2seq
  - Text generation

ELECTRA

Pretraining Text Encoders as Discriminators rather than Generators (ICLR 2020)

Currently on NLP

ELMo : Deep contextualized word representations
BERT : Bidirectional Encoder Representations from Transformers
ERNIE : 百度 and 清華 same name but different approach

Currently on NLP

Autoregressive : GPT-2 (2018) , ELMo (2018)

Pros : easy, can be used for text generation

Cons: Only learn context in one direction

Masked Language Modeling : BERT, SpanBERT, ERNIE (all 2019)

Masked + Autoregressive : MASS (2019), UniLM (2019)

Pros : Learn bidirectional context

Cons: **pretrain-finetune discrepancy** (No [MASK] in finetuning)

Sequence Permutation : **XLNet** (2019)

Pros : No pretrain-finetune discrepancy

Cons : Complex mask input for sequence factorization

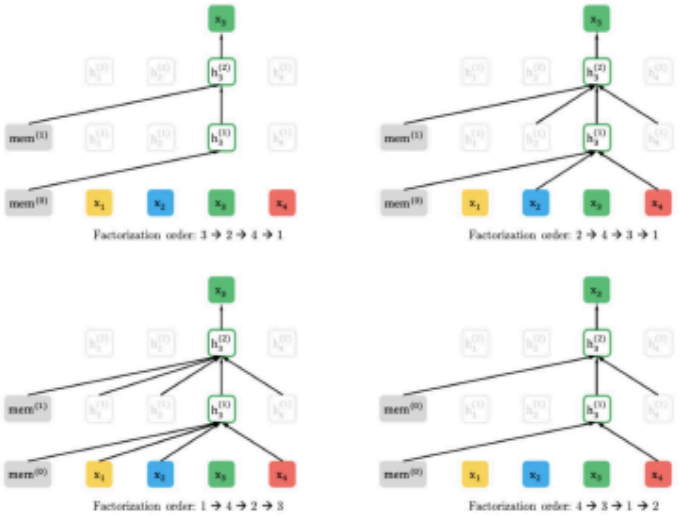


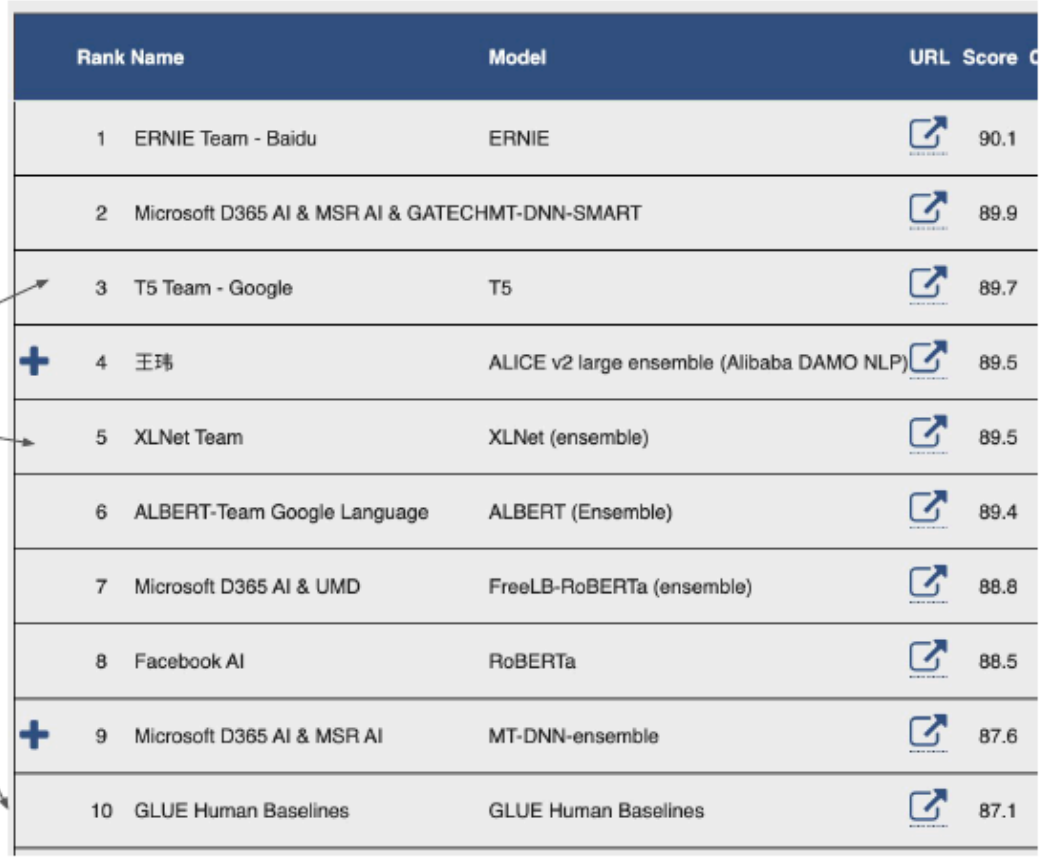
Figure 1: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence x but with different factorization orders.

Glue Benchmark : language understanding

- CoLA : determine a sentence is grammtical or not
- SST : Sentiment prediction (1 or 0)
- MRPC : predict if two sentence is semantically same or not
- STS : sentence similarity on scale of 1 to 5
- QQP : two question is the same or not
- MNLI : given a precondition sentence and hypothesis sentence and predict whether the premise entails
- the hypothesis, contradict or neither
- QNLI : predict which sentence is the answer to the question
- RTE : similar to MNLI except do not have the “neither” scenario

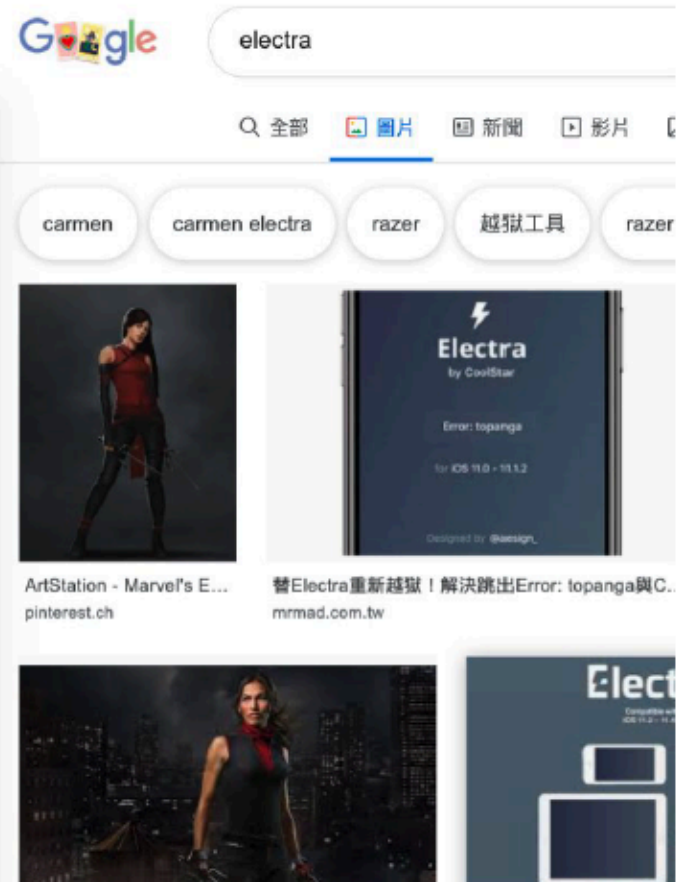
Glue Benchmark : language understanding

isn't based on BERT



Rank	Name	Model	URL	Score
1	ERNIE Team - Baidu	ERNIE		90.1
2	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9
3	T5 Team - Google	T5		89.7
4	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.5
5	XLNet Team	XLNet (ensemble)		89.5
6	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4
7	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8
8	Facebook AI	RoBERTa		88.5
9	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6
10	GLUE Human Baselines	GLUE Human Baselines		87.1

How to name NLP model



1. try to piece an acronym from your paper
2. google [your model acronym]
3. result has the following criterias

- Not a character from my childhood entertainment : Sesame Street
- Character from Marvel Daredevil comic
- an iOS jailbreak tool

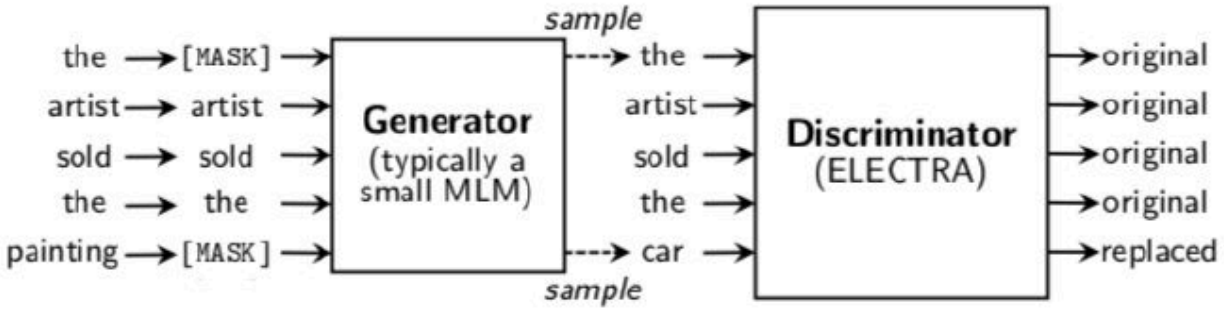
cool

ELECTRA :
Efficiently Learning an Encoder that Classifies Token Re-placements Accurately

✓Predicting Y/N is easier than reconstruction

✓Every output position is used

ELECTRA : Introduction



Generator $p_G(x_t|x) = \frac{exp(e(x_t)^\top h_G(x_t))}{\sum_i exp(e(x)_i^\top h_G(x_i)))}$

$x_{masked} = REPLACE(x, m, [mask])$

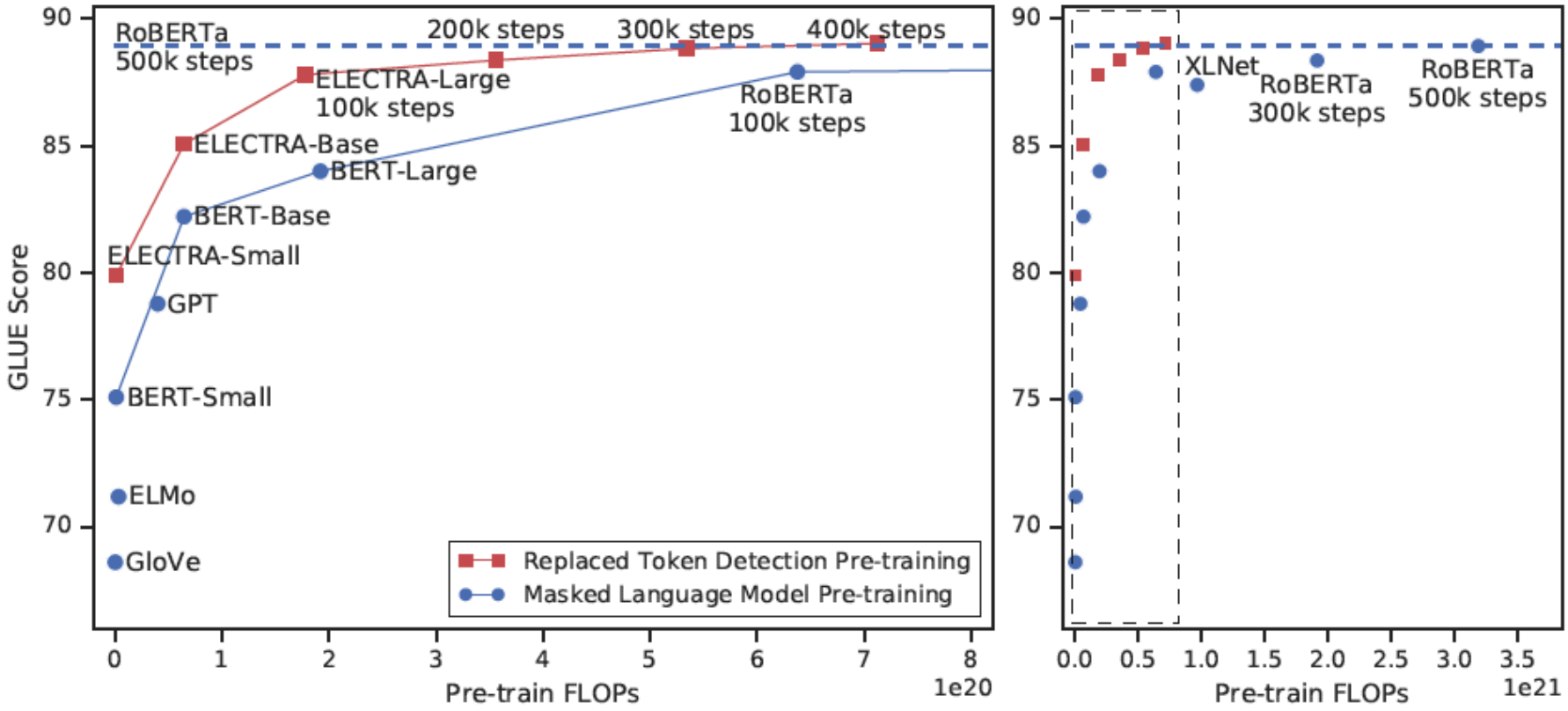
Discriminator $p_D(x,t) = sigmoid(w^T h_D(x_t))$

$x_{corrupt} = REPLACE(x, m, \bar{x})$

$$L_{MLM}(x, \theta_G) = \sum -log p_G(x_i|x^{masked}$$

$$L_{Disc}(x, \theta_D) = \sum (x_t^{corrupt} = x_t) log D(x^{corrupt}, x_t) + (x_t^{corrupt} \neq x_t) log(1 - D(x^{corrupt}, x_t)$$

$$min_{\theta_G, \theta_D} \sum L_{MLM}(\theta_G) + \lambda L_{Disc}(\theta_D)$$


90
85
80
75
70
GLUE Score
RoBERTa
500k steps
200k steps
300k steps
400k steps
ELECTRA-Large
100k steps
RoBERTa
100k steps
ELECTRA-Base
BERT-Large
BERT-Base
ELECTRA-Small
GPT
BERT-Small
ELMo
GloVe
Replaced Token Detection Pre-training
Masked Language Model Pre-training
0
1
2
3
4
5
6
7
8
1e20
Pre-train FLOPs
XLNet
RoBERTa
300k steps
RoBERTa
500k steps
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e21
Pre-train FLOPs

ELECTRA : Introduction

$$min_{\theta_G,\theta_D} \sum L_{MLM}(\theta_G) + \lambda L_{Disc}(\theta_D)$$

Discriminator and Generator, looks like GAN

So can we make generator loss backprop from discriminator? Yes

ELECTRA : adversarially (RL) version

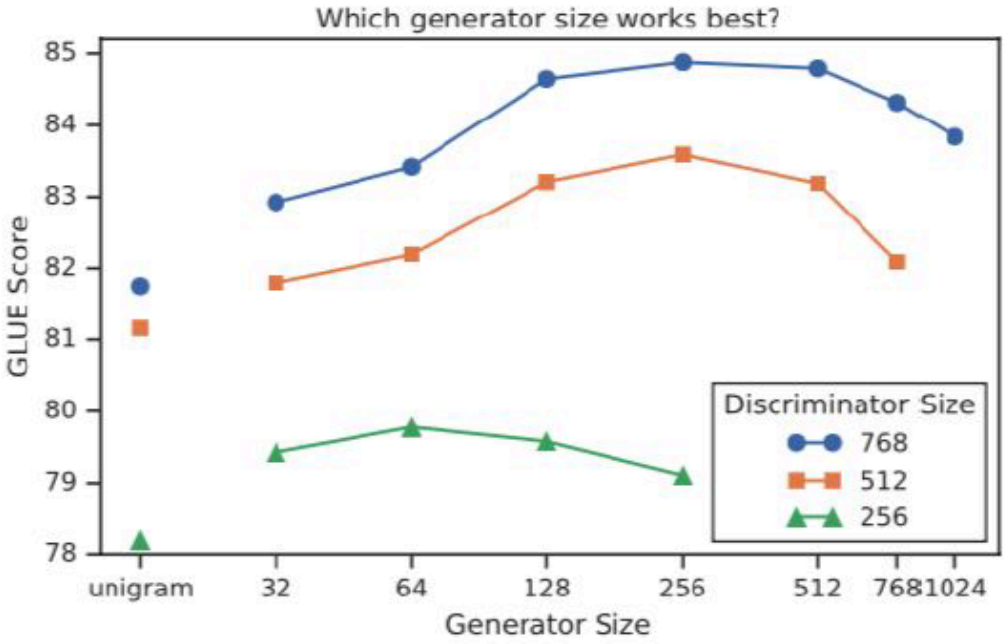
Use REINFORCE gradient to make generator maximize a reward function :

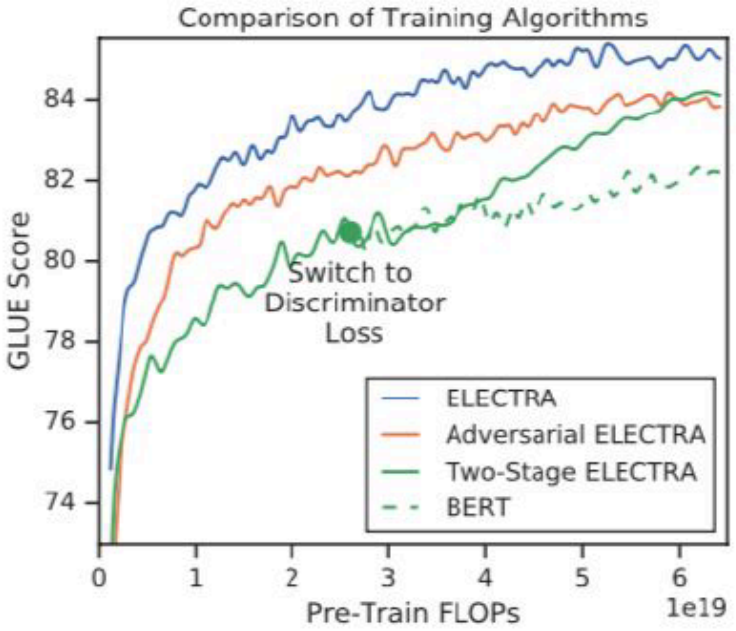
recover masked token and minimize the discriminator (b) prediction

$$\nabla_{\theta_G} \mathcal{L}_{\text{Disc}} \approx \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{m}} \sum_{t \in \boldsymbol{m}} \mathop{\mathbb{E}}_{\hat{x}_t \sim p_G} \nabla_{\theta_g} \log p_G(\hat{x}_t | \boldsymbol{x}^{\text{masked}}) [R(\hat{x}_t, \boldsymbol{x}) - b(\boldsymbol{x}^{\text{masked}}, t)]$$

But RL version achieve 58% accuracy at masked language model while MLE get 65% and no improvement over MLE version on downstream tasks

Comparison between different strategy





ELECTRA Result

GLUE dev dataset

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
XLNet	9.6e20 (1.3x)	360M	63.6	95.6	89.2	91.8	91.8	89.8	93.9	83.8	87.4
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa	3.2e21 (4.5x)	356M	68.0	**96.4**	90.9	**92.4**	92.2	90.2	**94.7**	86.6	88.9
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	**90.5**	94.5	**86.8**	**89.0**
Test set results for models with standard single-task finetuning (no ensembling, task-specific tricks, etc.)											
BERT	1.9e20 (0.27x)	335M	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	83.3
SpanBERT	7.1e20 (1x)	335M	64.3	94.8	87.9	89.9	89.5	87.7	94.3	79.0	85.9
ELECTRA	7.1e20 (1x)	335M	**68.2**	**96.9**	**89.6**	**91.0**	**90.1**	**90.1**	**95.4**	**83.6**	**88.1**

ELECTRA Result

GLUE test dataset

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
TinyBERT	6.4e19+ (45x+)	14.5M	43.3	92.6	81.2	79.9	89.2	82.5	87.7	62.9	77.4
GPT	4.0e19 (29x)	117M	45.4	91.3	75.7	80.0	88.5	82.1	88.1	56.0	75.9
BERT-Base	6.4e19 (45x)	110M	52.1	93.5	84.8	85.8	89.2	84.6	90.5	66.4	80.9
BERT-Large	1.9e20 (135x)	335M	**60.5**	**94.9**	85.4	86.5	**89.3**	**86.7**	**92.7**	70.1	83.3
ELECTRA-Small	1.4e18 (1x)	14M	54.6	89.1	83.7	80.3	88.0	79.7	87.7	60.8	78.0
ELECTRA-Base	6.4e19 (45x)	110M	59.7	93.4	**86.7**	**87.7**	89.1	85.8	**92.7**	**73.1**	**83.5**

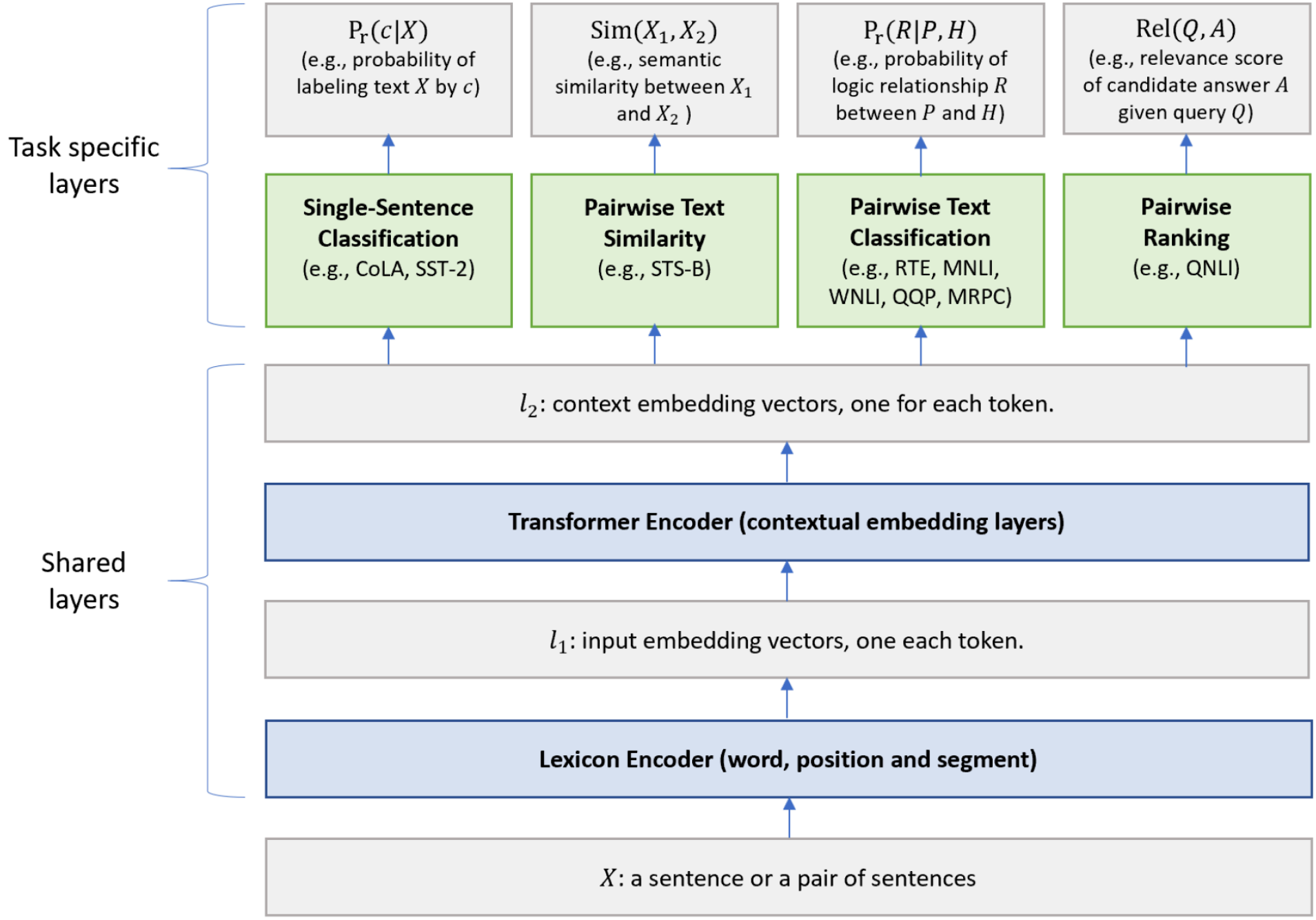
BERT：我訓練再久一點、數據量再大一點，就能重返SOTA

2019-07-19

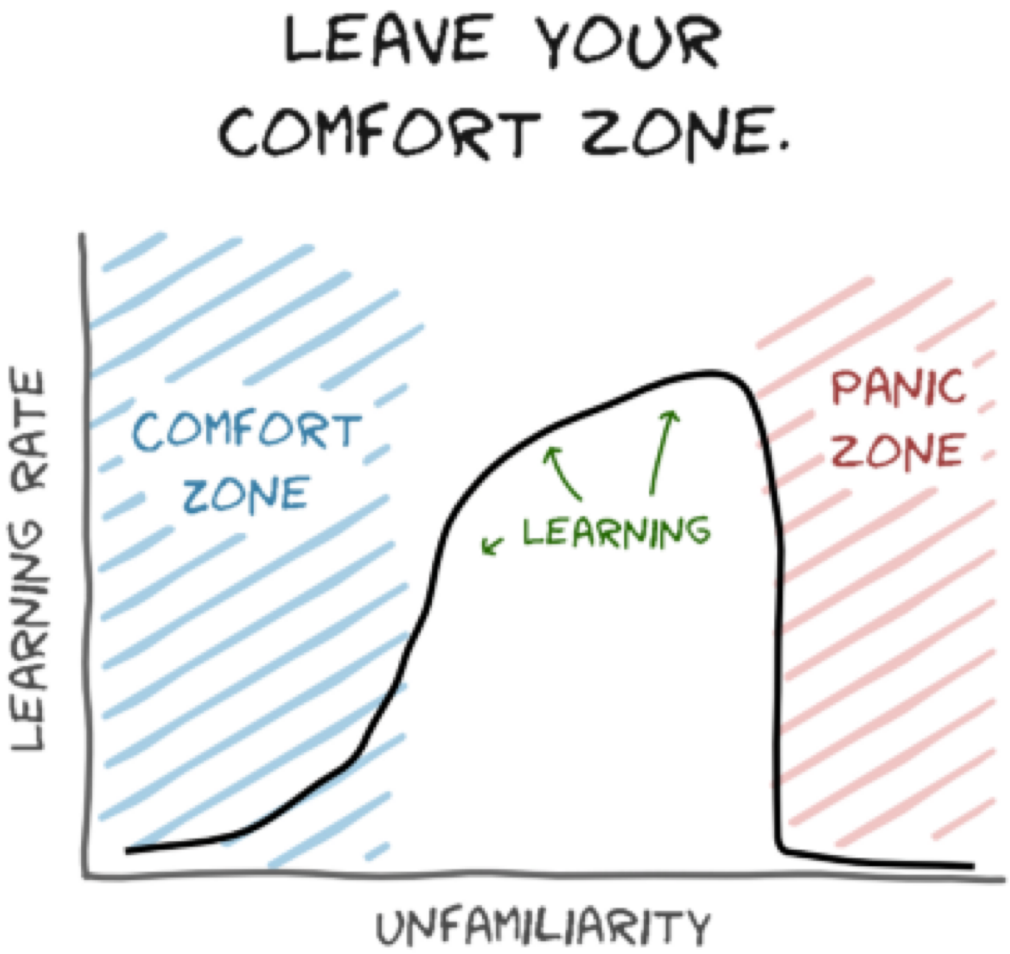
> XLNet 冠軍寶座還沒坐熱，劇情又一次發生反轉。

Robustly Optimized BERT pretraining Approach

	Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
−	1	Microsoft D365 AI & MSR AI	MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	50.2
			MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9	87.4	96.0	86.3	89.0	42.8
			MT-DNN		85.1	65.4	95.6	91.2/88.1	89.6/89.0	73.7/89.9	87.9	87.4	96.0	85.7	75.3	42.8
	2	T5 Team - Google	T5		89.7	70.8	97.1	91.9/89.2	92.5/92.1	74.6/90.4	92.0	91.7	96.7	92.5	93.2	53.1
	3	ERNIE Team - Baidu	ERNIE		89.6	72.2	97.5	93.0/90.7	92.9/92.5	75.0/90.6	91.2	90.6	98.0	90.3	90.4	49.4
+	4	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.5	71.3	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.7	90.4	99.2	87.4	91.8	48.4
	5	XLNet Team	XLNet (ensemble)		89.5	70.2	97.1	92.9/90.5	93.0/92.6	74.7/90.4	90.9	90.9	99.0	88.5	92.5	48.4
	6	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5	91.3	91.0	99.2	89.2	91.8	50.2
	7	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3	91.1	90.7	98.8	88.7	89.0	50.1
	8	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	98.9	88.2	89.0	48.7


Task specific layers
$\mathrm{P_r}(c|X)$ (e.g., probability of labeling text X by c)
$\mathrm{Sim}(X_1, X_2)$ (e.g., semantic similarity between X_1 and X_2)
$\mathrm{P_r}(R|P, H)$ (e.g., probability of logic relationship R between P and H)
$\mathrm{Rel}(Q, A)$ (e.g., relevance score of candidate answer A given query Q)
Single-Sentence Classification (e.g., CoLA, SST-2)
Pairwise Text Similarity (e.g., STS-B)
Pairwise Text Classification (e.g., RTE, MNLI, WNLI, QQP, MRPC)
Pairwise Ranking (e.g., QNLI)
Shared layers
l_2: context embedding vectors, one for each token.
Transformer Encoder (contextual embedding layers)
l_1: input embedding vectors, one each token.
Lexicon Encoder (word, position and segment)
X: a sentence or a pair of sentences

Learning curve



Is it Time to Swish?
Comparing Deep Learning Activation Functions Across NLP tasks

Steffen Eger, Paul Youssef, Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science
Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

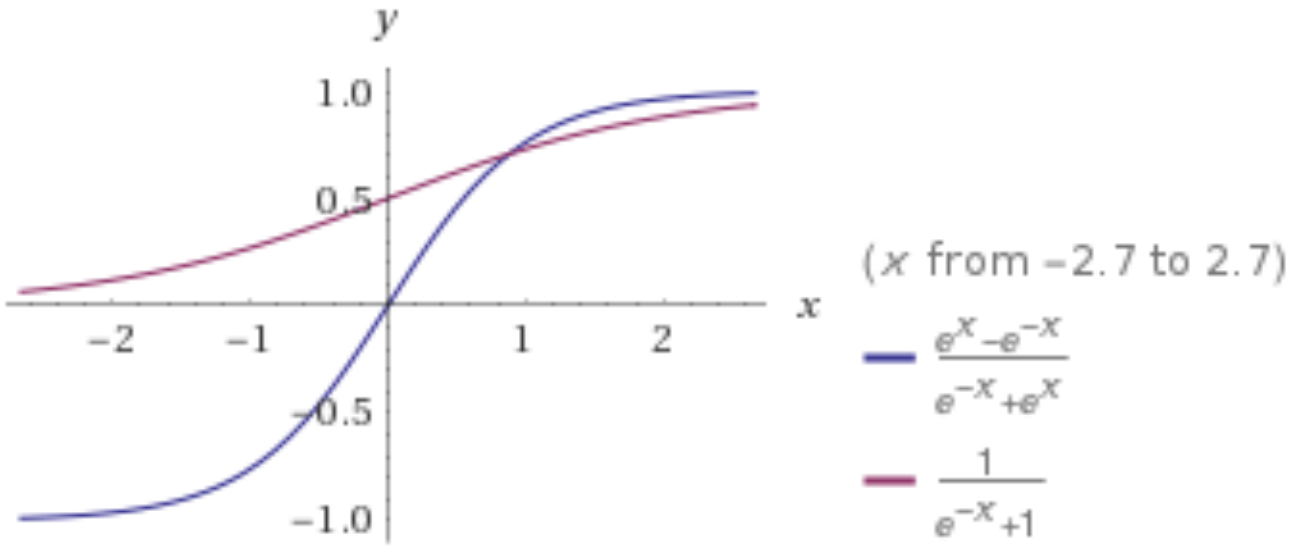
- (1) 比較了21 種不同的激活函數，包括Ramachandran et al. (2017) 中通過自動搜索找到的6 種表現最佳的激活函數。
- (2) 採用了3 種常見的NLP 任務類型（句子分類、文檔分類、序列標註），包含8 項單個任務
- (3)我們使用了3 種常用的NLP 架構，即MLP、CNN 和RNN
- (4)我們在兩個不同維度上比較了所有這些函數，即最佳表現和平均表現。

Ramachandran et al. (2017)

sigmoid	$f(x) = \sigma(x) = 1/(1 + \exp(-x))$
swish	$f(x) = x \cdot \sigma(x)$
maxsig	$f(x) = \max\{x, \sigma(x)\}$
cosid	$f(x) = \cos(x) - x$
minsin	$f(x) = \min\{x, \sin(x)\}$
arctid	$f(x) = \arctan(x)^2 - x$
maxtanh	$f(x) = \max\{x, \tanh(x)\}$
lrelu-0.01	$f(x) = \max\{x, 0.01x\}$
lrelu-0.30	$f(x) = \max\{x, 0.3x\}$
penalized tanh	$f(x) = \begin{cases} \tanh(x) & x > 0, \\ 0.25 \tanh(x) & x \leq 0 \end{cases}$

Property	Description	Problems	Examples
derivative	f'	> 1 exploding gradient (e) < 1 vanishing (v)	sigmoid (v), tanh (v), cube (e)
zero-centered	range centered around zero?	if not, slower learning	tanh (+), relu (−)
saturating	finite limits	vanishing gradient in the limit	tanh, penalized tanh, sigmoid
monotonicity	$x > y \implies f(x) \geq f(y)$	unclear	exceptions: sin, swish, minsin

Table 2: Frequently cited properties of activation functions



Results

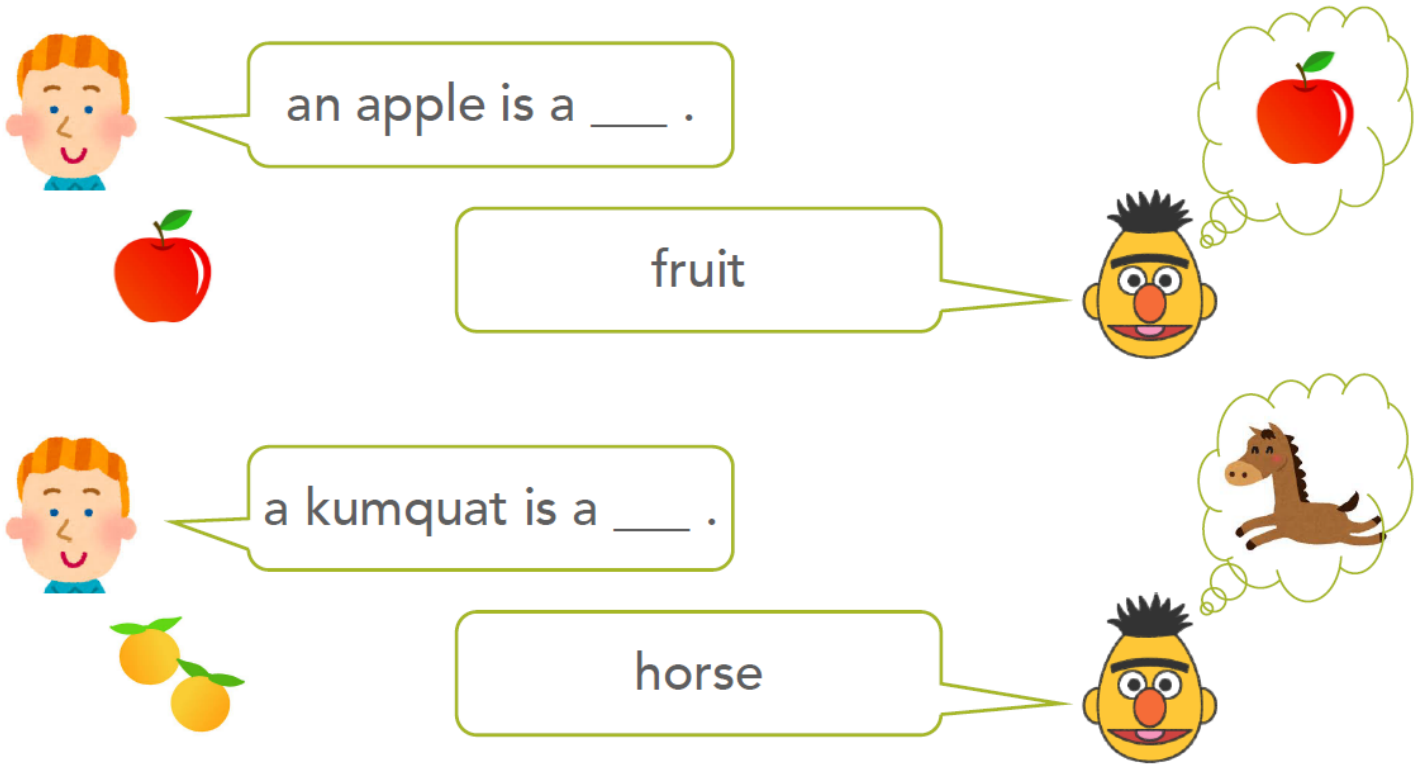
best	penalized tanh (6), swish (6), elu (4), relu (4), lrelu-0.01 (4)
mean	penalized tanh (16), tanh (13) sin (10)

BERTRAM: Improved Word Embeddings Have Big Impact on Contextualized Representations

ACL’2020

BERT and Rare Words

- Motivation: BERT fails with rare words. (e.g., "a kumquat is a ___", "an arghul is an ___").



Questions