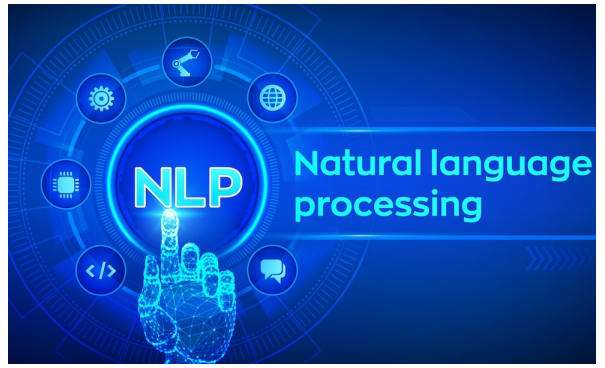
AI3603

NLP and Transformer

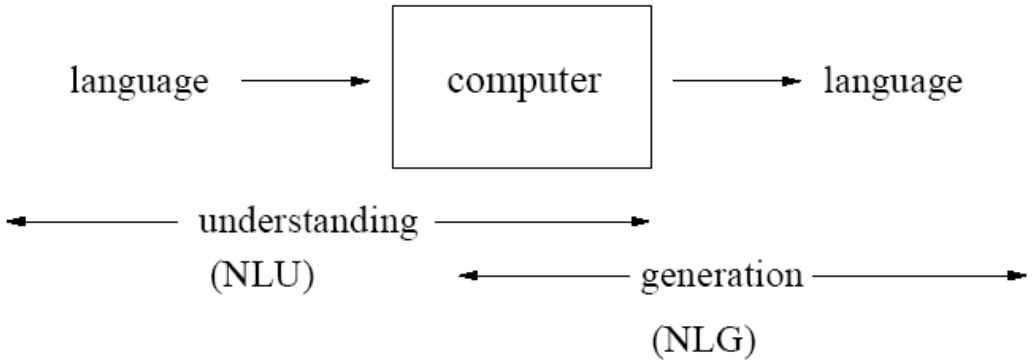
Yue Gao

Shanghai Jiao Tong University



Natural Language Processing

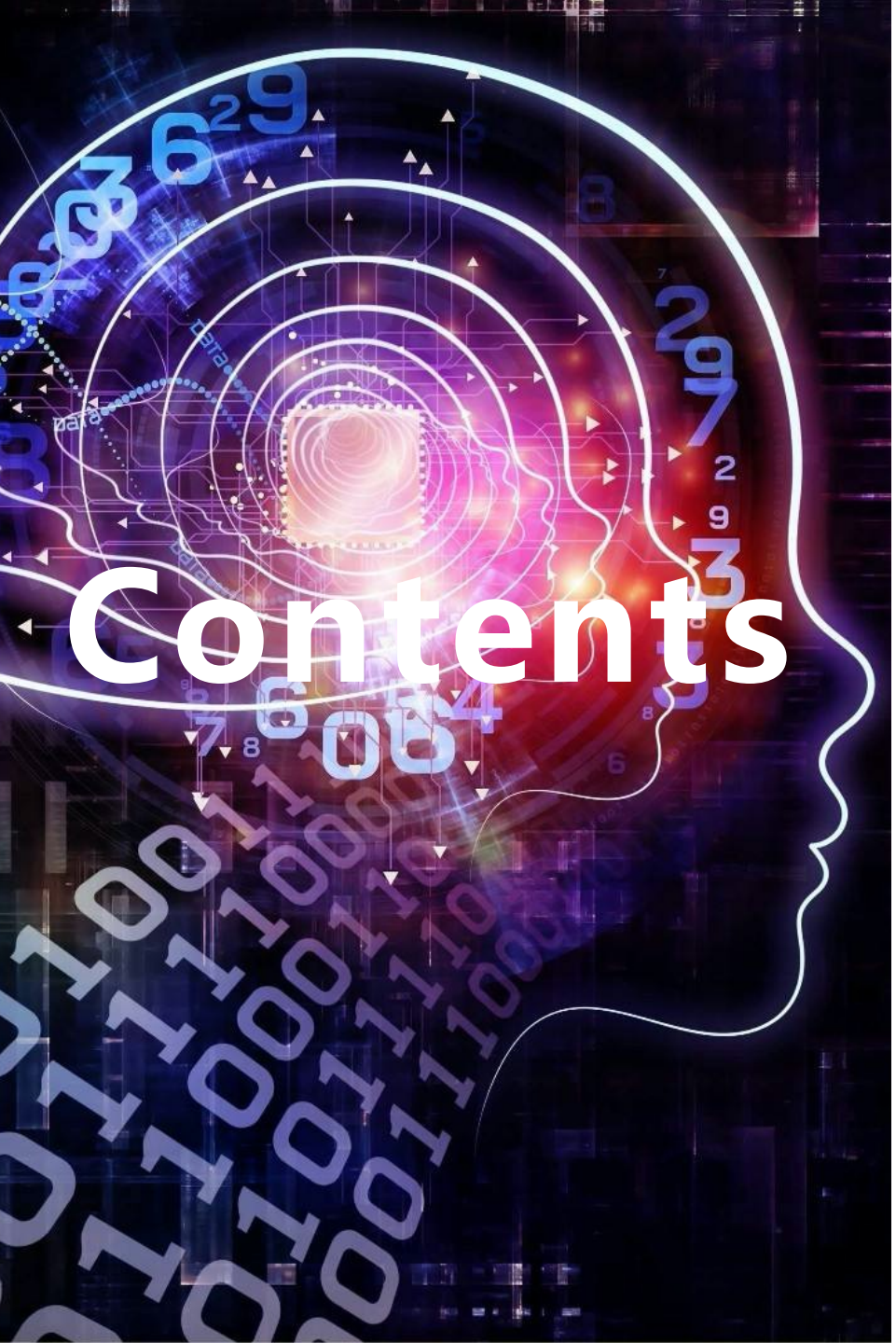
- Natural Language Processing (NLP) is **a field in Artificial Intelligence (AI)** devoted to creating computers that use natural language as **input or output**.



NLU	NLP	NLG
It is a narrow concept.	It is a wider concept.	It is a narrow concept.
If we only talk about an understanding text, then it is enough.	But if we want more than understanding, such as decision making, then it comes into play.	It generates a human-like manner text based on the structured data.
It is a subset of NLP.	It is a combination of it and NLG for conversational Artificial Intelligence problems.	It is a subset of NLP.
It is not necessarily that what is written or said is meant to be the same. There can be flaws and mistakes. It makes sure that it will infer correct intent and meaning even data is spoken and written with some errors. It is the ability to understand the text.	But, if we talk about NLP, it is about how the machine processes the given data. Such as make decisions, take actions, and respond to the system. It contains the whole End to End process. Every time it doesn't need to contain it.	It generates structured data, but it is not necessarily that the generated text is easy to understand for humans. Thus NLG makes sure that it will be human-understandable.
It reads data and converts it to structured data.	It converts unstructured data to structured data.	NLG writes structured data.

Natural Language Processing

- To access large amount of information and knowledge stored in the form of human languages quickly.
  - emails, news articles, web pages, IM, scientific articles, insurance claims, technical documents, government documents……
- To interact with computing devices using human (natural) languages.
  - Building intelligent robots.
  - Enabling voice-controlled operation.



1. What is NLP — Turing Test

- Alan Turing (1950) proposed a test of a machine's capability to perform human-like conversation.

- A human judge engages in a natural language conversation with two other parties, one a human and the other a machine; if the judge **cannot reliably tell which is which**, then the machine is said to pass the test.

1. What is NLP — Early NLP Systems

- **ELIZA** (by Joseph Weizenbaum), 1966
  - A psychotherapist
  - No real understanding; simple pattern-matching: input -> canned responses
  - First chatterbots; first programs capable of attempting the Turing test.

```
User:   I am unhappy.
ELIZA:  DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User:   I need some help, that much seems certain.
ELIZA:  WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User:   Perhaps I could learn to get along with my mother.
ELIZA:  TELL ME MORE ABOUT YOUR FAMILY
User:   My mother takes care of me.
ELIZA:  WHO ELSE IN YOU FAMILY TAKES CARE OF YOU
User:   My father.
ELIZA:  YOUR FATHER
User:   You are like my father in some ways.
```

Weizenbaum (1966)

1. What is NLP — Brief History of NLP

- 1940s –1950s: **Foundational Insights**
  - Two foundational paradigms
    - Automaton
    - Probabilistic / Information-Theoretic Models

人类DNA中已经有一些grammar

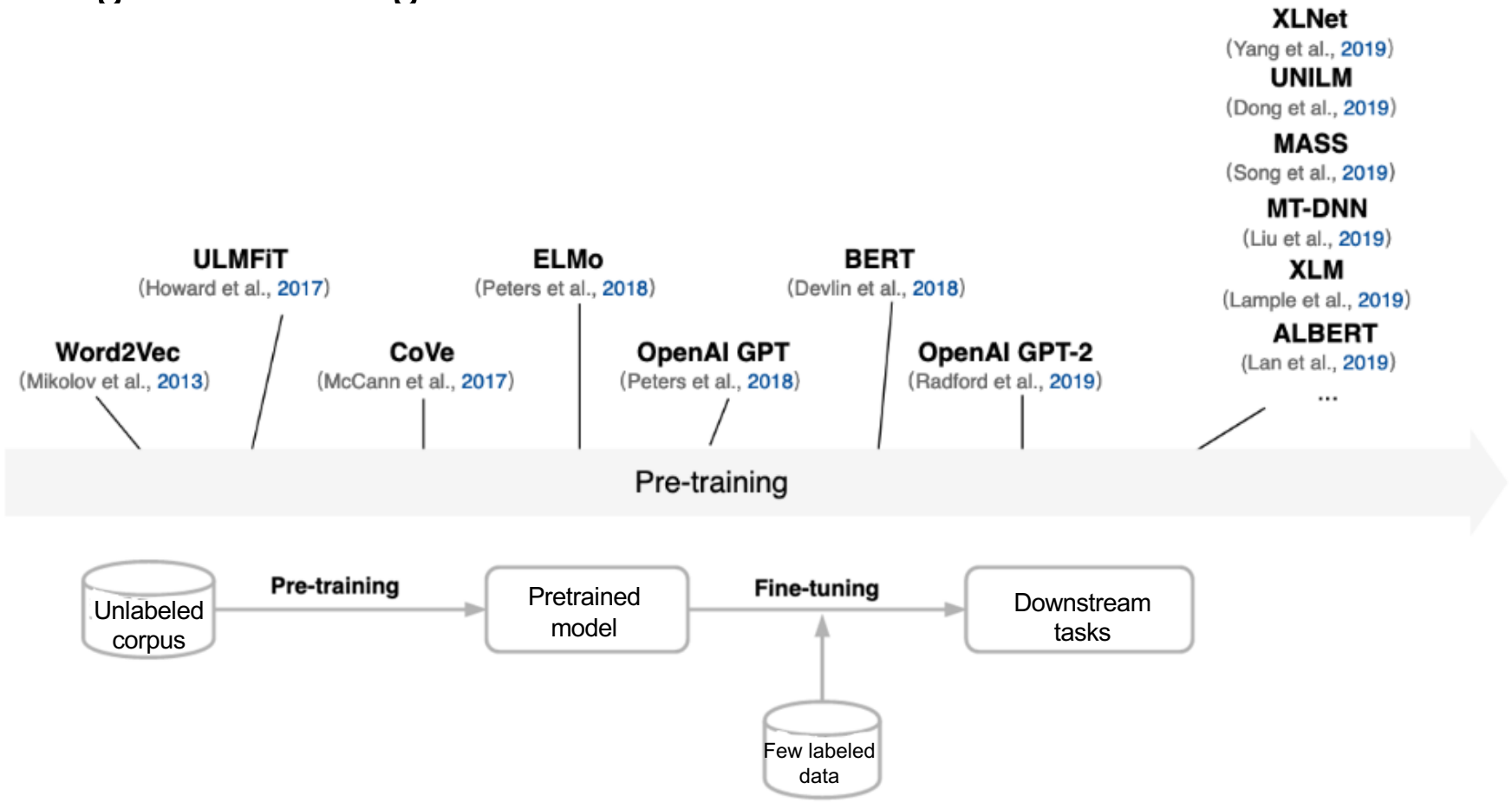
- 1957-1970: **The two camps**
  - **Symbolic paradigm**: Chomsky and others on formal language theory and generative syntax
  - **Stochastic paradigm**

1. What is NLP — Brief History of NLP

- 1970 – 1983: Different paradigms
  - Stochastic
  - Logic-based
  - Natural language understanding
- 1983-1993: Empiricism and Finite State Models
- 1994-1999: The fields come together. **Probabilistic and data-driven models**
- Rise of ML:
  - Lots of data and compute

1. What is NLP — Brief History of NLP

- 2018 –
  - Pretraining + fine-tuning



1. What is NLP — Challenge of NLP

The diversity of natural language

Many-to-many mapping between
***symbolic* language** and ***semantic* meaning**

Ambiguity

Example: I saw a man on a hill with a telescope.

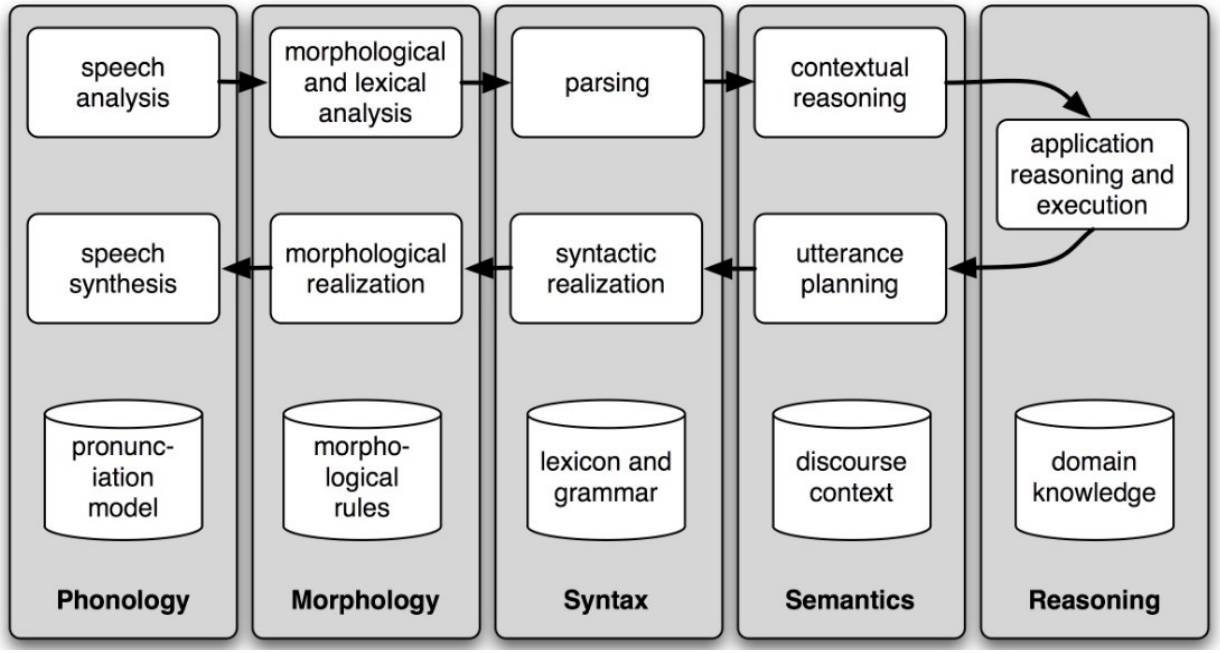
- *There's a man on a hill, and I'm watching him with my telescope.*
- *There's a man on a hill, who I'm seeing, and he has a telescope.*
- *I'm on a hill, and I saw a man using a telescope.*

Paraphrase

Example: How long is the X river?

- *The Mississippi River is 3,734 km (2,320 mi) long.*
- *...is a short river, some 4.5 miles (7.2 km) in length*
- *The total length of the river is 2,145 kilometers.*
- *… at the estimated length of 5,464 km (3,395 mi)…*
- *… has a meander length of 444 miles (715 km)…*
- *… Bali's longest river, measuring approximately 75 kilometers from source to mouth.*
- *The … mainstem is 2.75 miles (4.43 km) long although total distance from headwater source tributaries to the sea is 14 miles (23 km).*

Traditional NLP



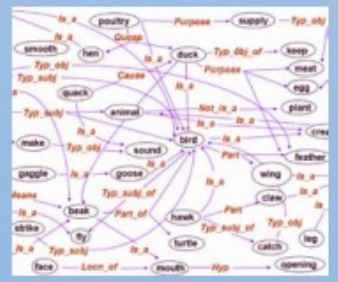
- **Study knowledge of language at different levels**
  - **Phonology** – the study of linguistic sounds
  - **Morphology** – the study of the meaning of components of words
  - **Syntax** – the study of the structural relationships between words
  - **Semantics** – the study of meaning
  - **Discourse** – they study of linguistic units larger than a single utterance

1. What is NLP — Mordern NLP

Traditional NLP to Neural Methods

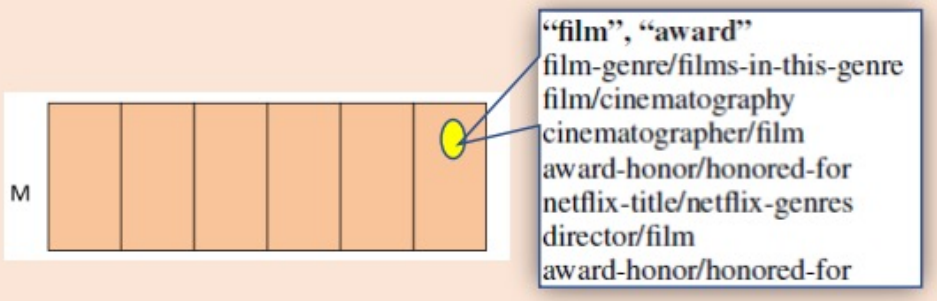
Traditional symbolic approaches

- Discrete, symbolic space
- Human comprehensible
  - easy to debug
- Computationally inefficient
  - Sensitive to ambiguity/paraphrase
  - Cascaded models prone to error propagation and require careful feature engineering



Deep Learning (DL) approaches

- Continuous, neural space
- Human incomprehensible
  - hard to debug
- Computationally efficient
  - Robust to ambiguity/paraphrase
  - E2E learning leads to better performance and simplified systems



1. What is NLP — NLP Applications

- **Sentiment analysis:** to understand the negative or positive polarity of a sentence, paragraph, or text which can represent the general idea of the consumers on our products, services and etc.
- **Text classification:** analyzing the natural language data to classify them in various groups. An example would be spam detection where we can process the content of an email to classify it as spam/non-spam.
- **Question answering:** to give a proper reply to the questions. An example would be chatbots that reply to your inquiries.
- **Text summarization:** to shorten a long text. An application would be detecting duplicate texts in a job-board where the job poster cannot spam by uploading the same opening in different wordings.

1. What is NLP — NLP Applications

Three prominent application areas:

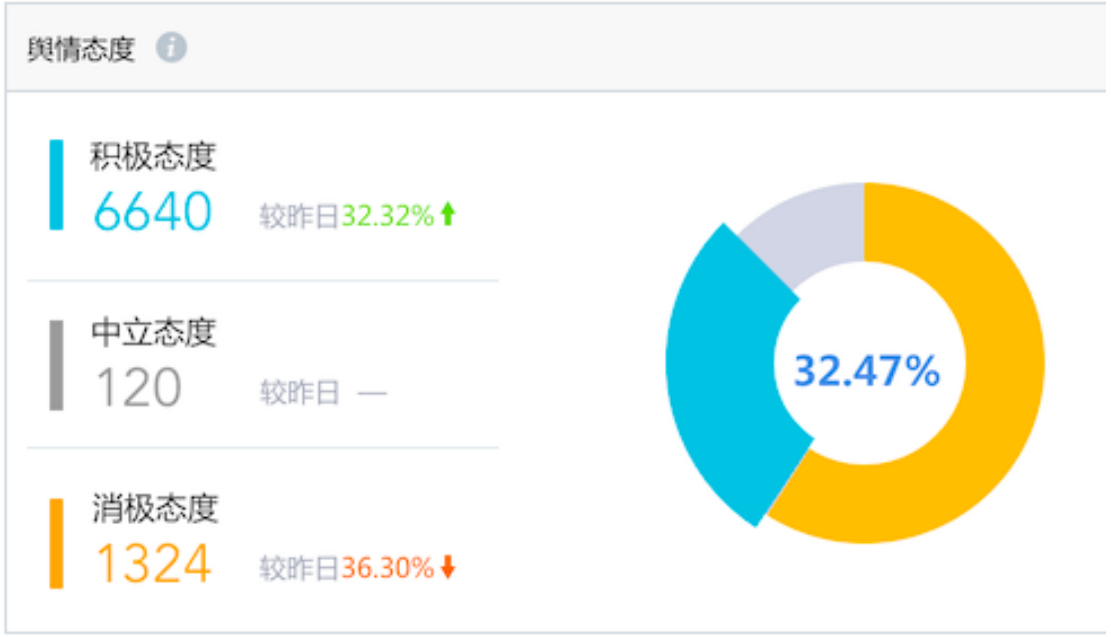
- Text analytics/mining (from "unstructured data")
  - Sentiment analysis
  - Topic identification
- Conversational agents
  - Siri, Cortana, Amazon Alexa, Google Assistant
  - Chatbots
- Machine translation

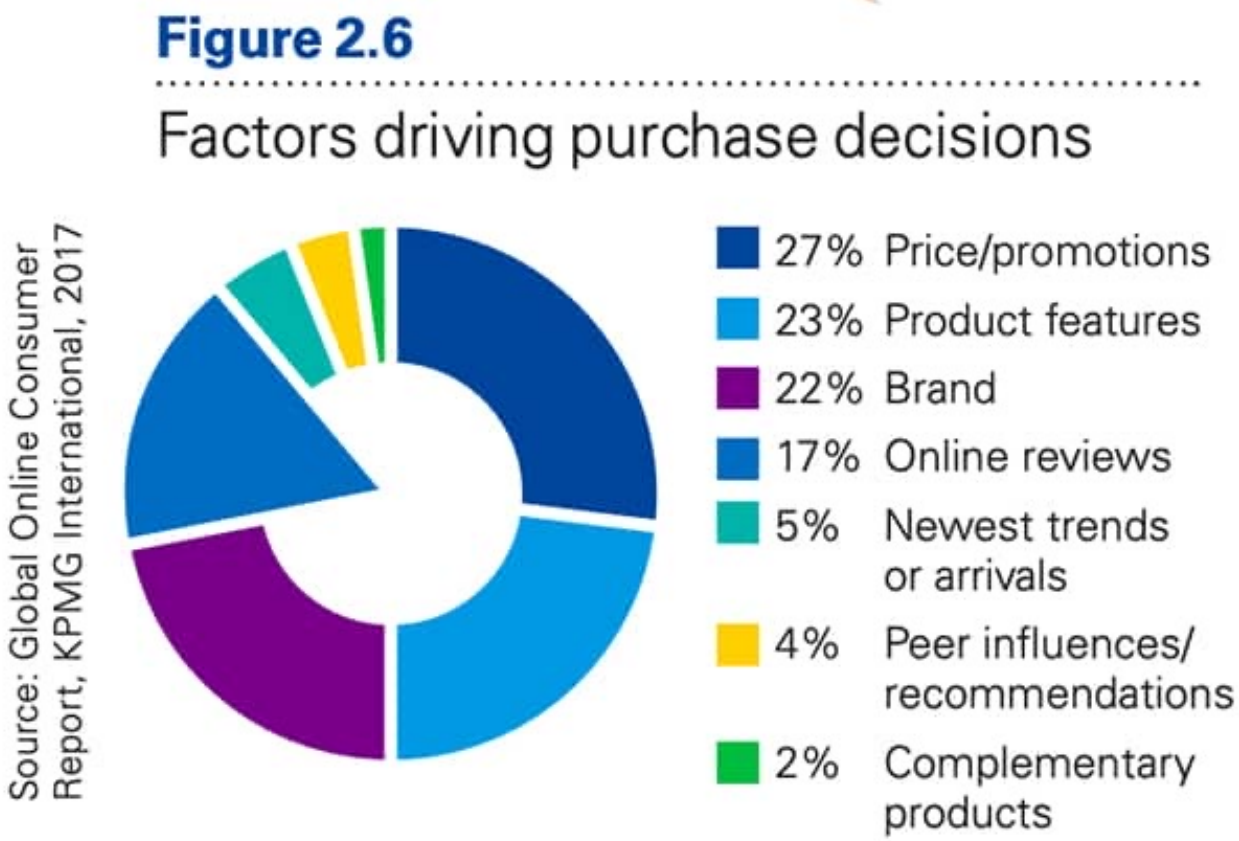
1. What is NLP — NLP Applications

- Dialogue systems
  - Siri
  - Cortana
  - Microsoft AI-Xiaobing
- Search engines
  - Google
  - Baidu
- Speech recognition
  - iFLYTEK
- Maching translation
  - Google translation
  - Baidu translation

1. What is NLP — NLP Applications

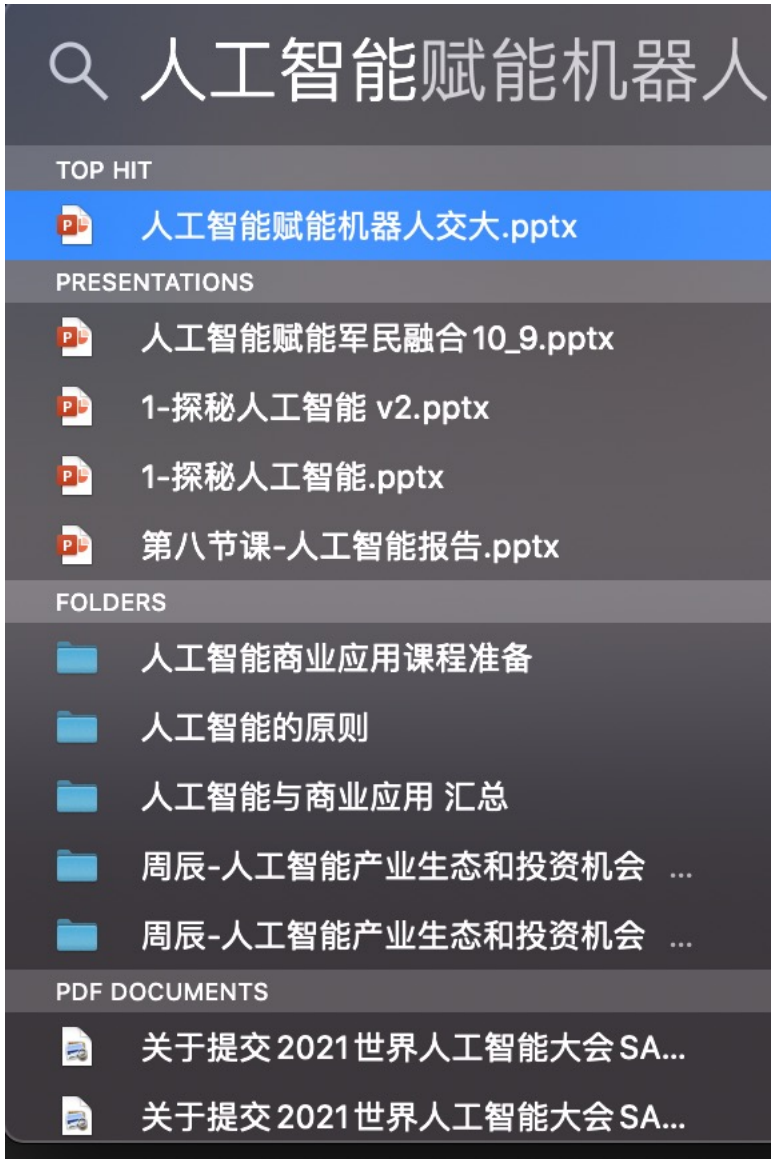
Emotion Analysis



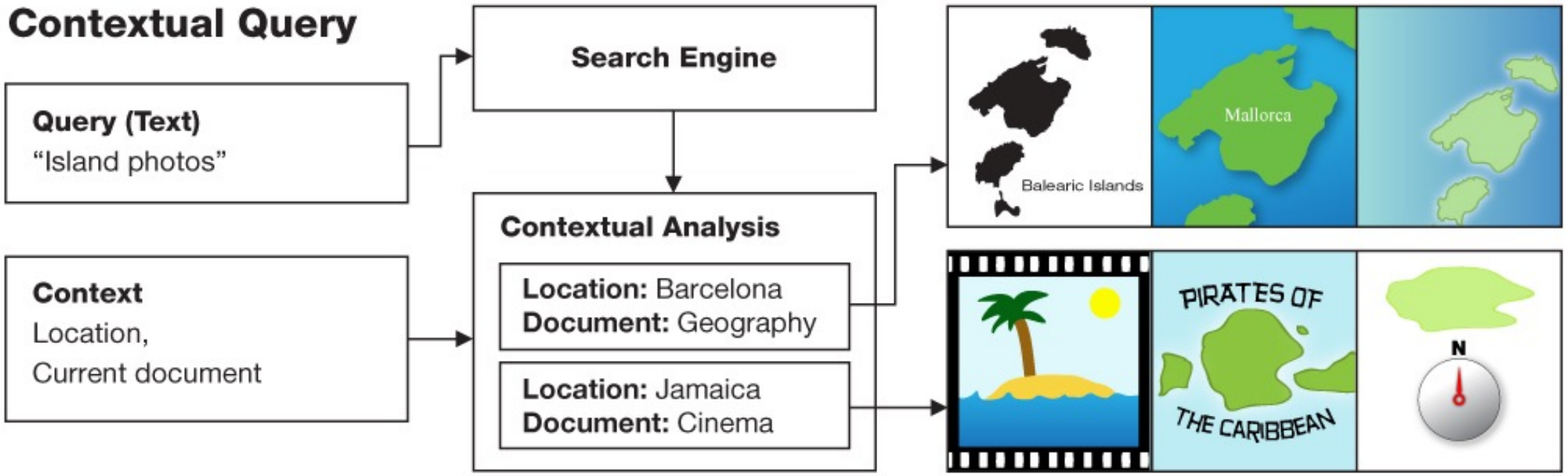


1. What is NLP — NLP Applications

Search (public, private)

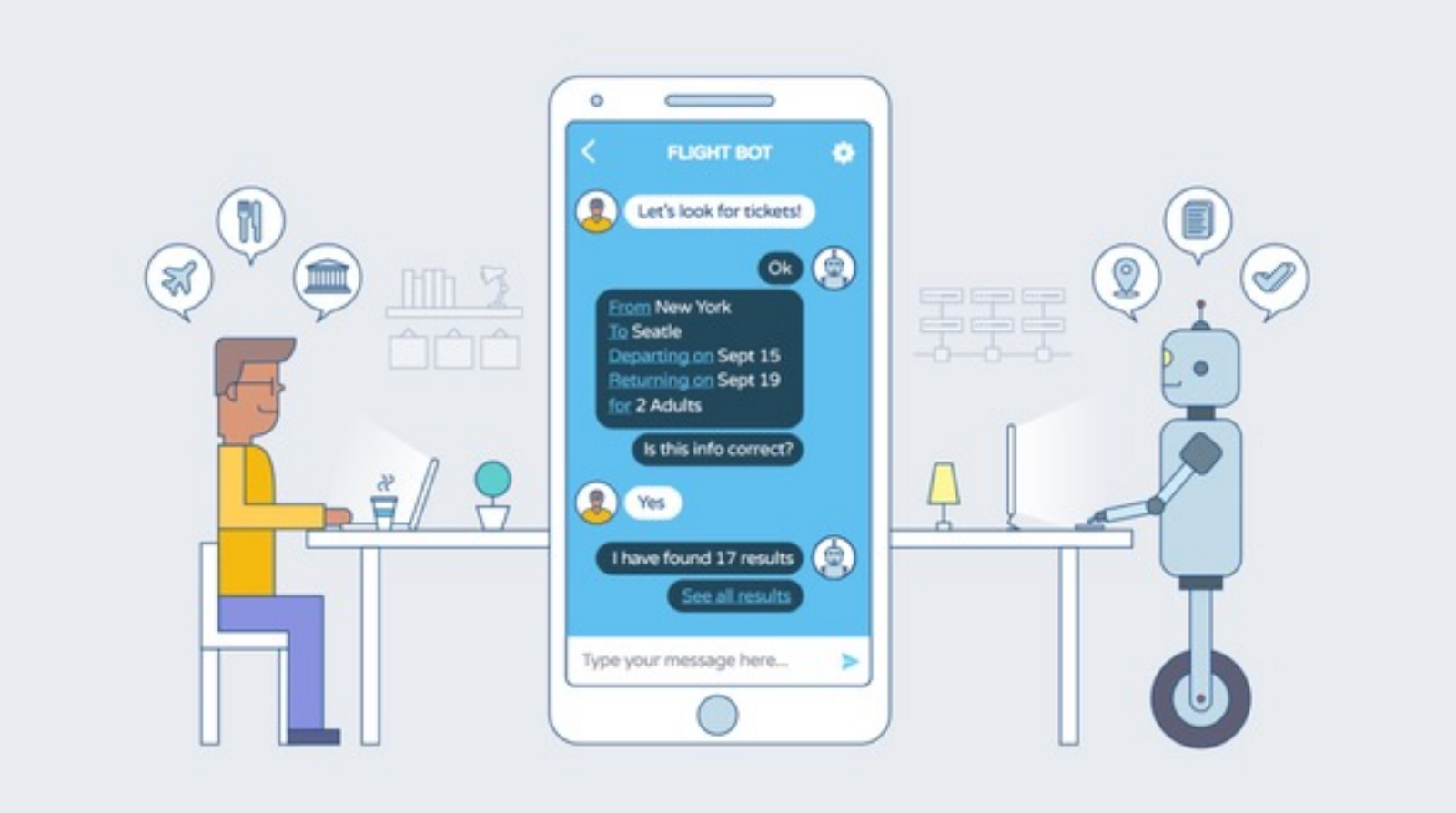


Multimodal Search (public, private)



1. What is NLP — NLP Applications

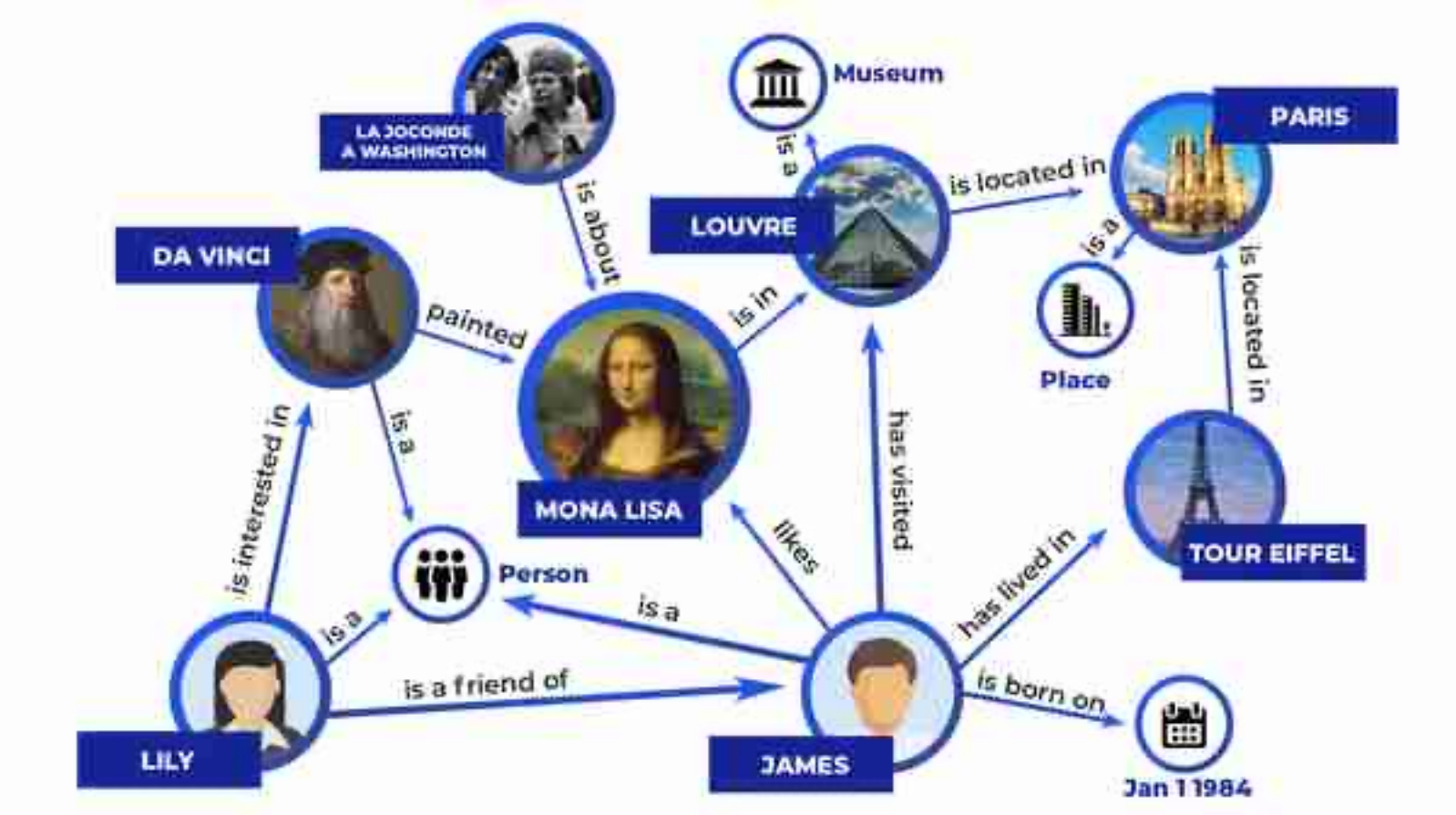
QA



撩妹机器人

1. What is NLP — NLP Applications

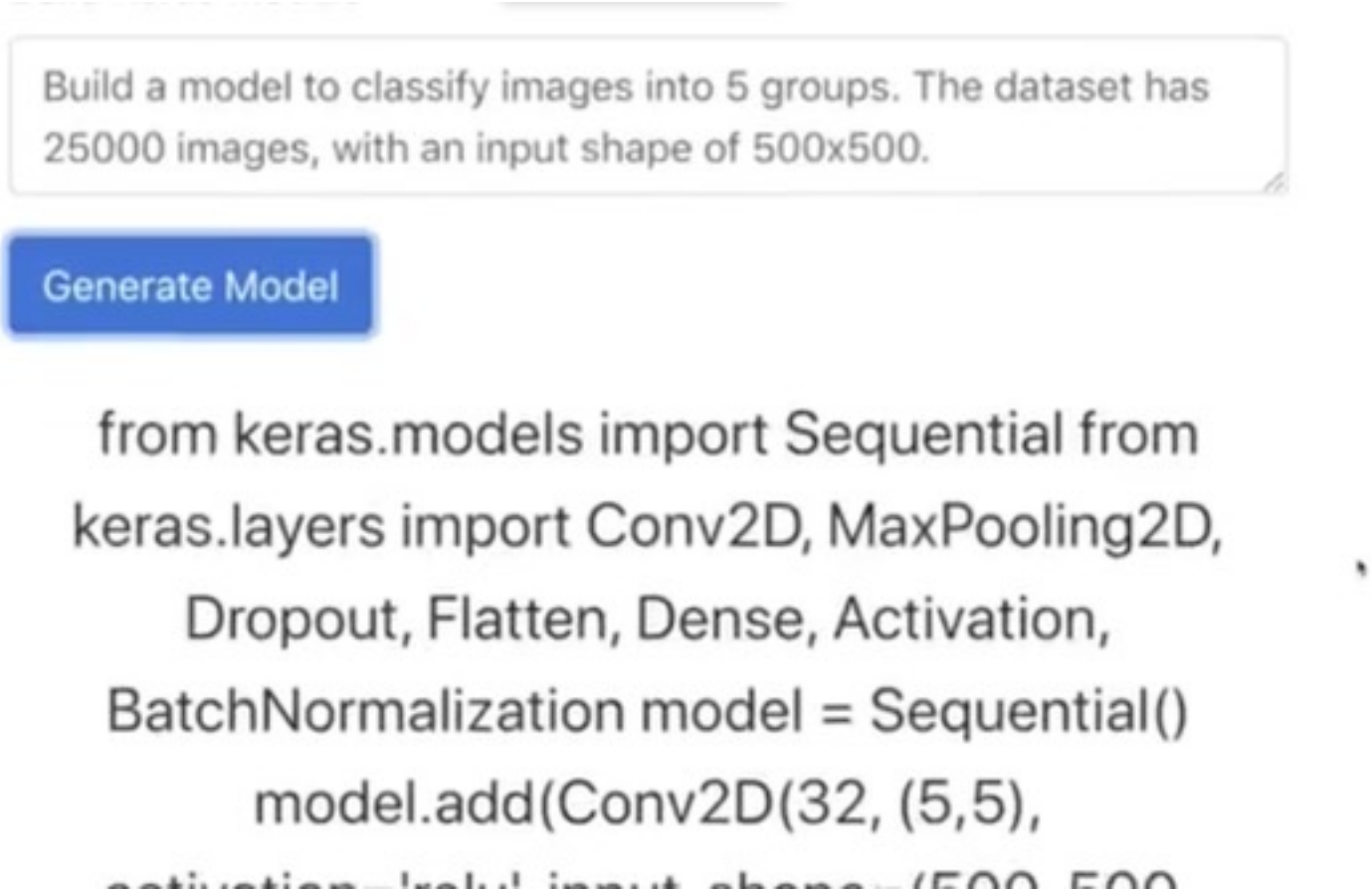
Knowledge Graph



1. What is NLP — NLP Applications

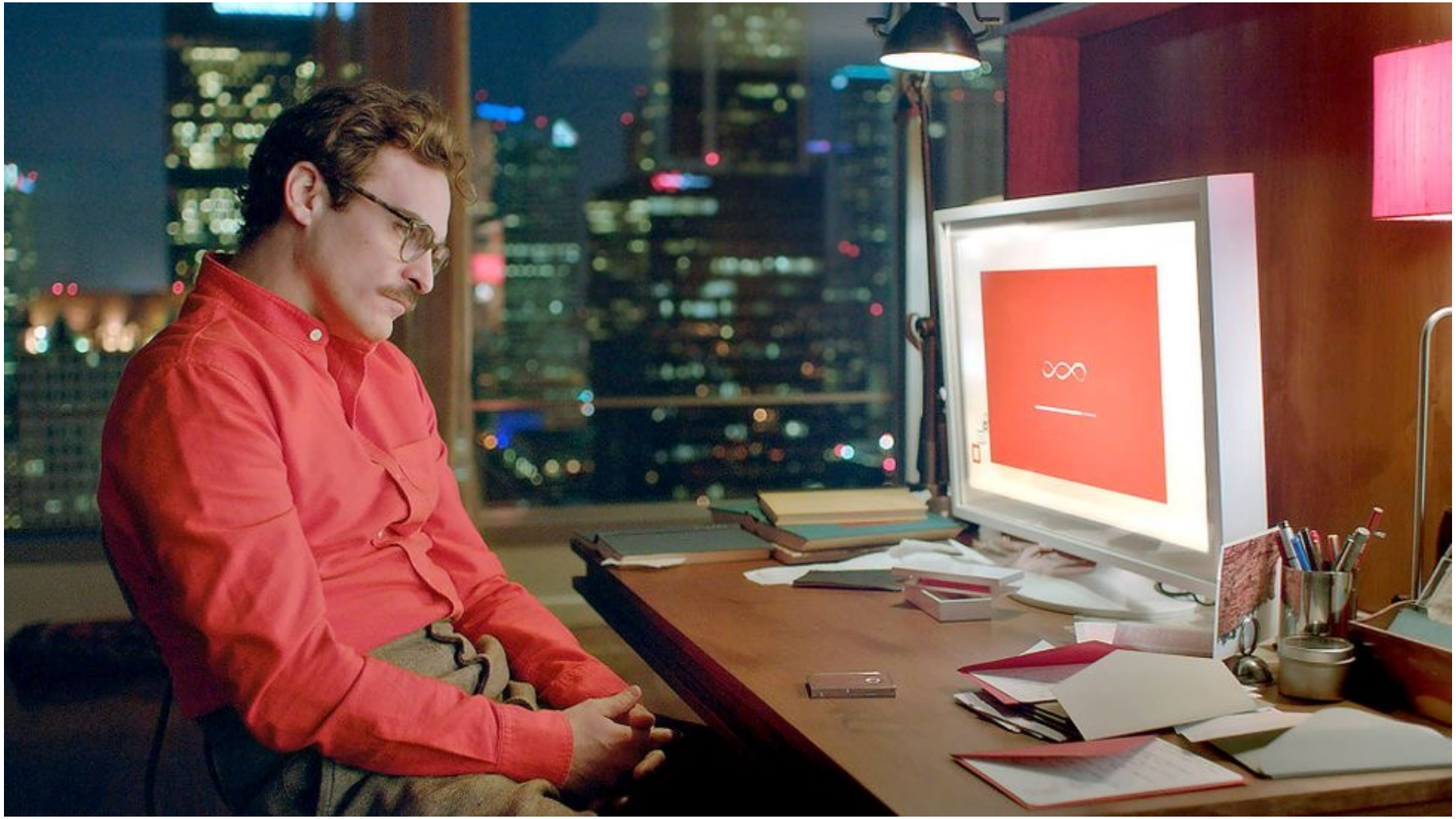
Generate Text

Generate Text



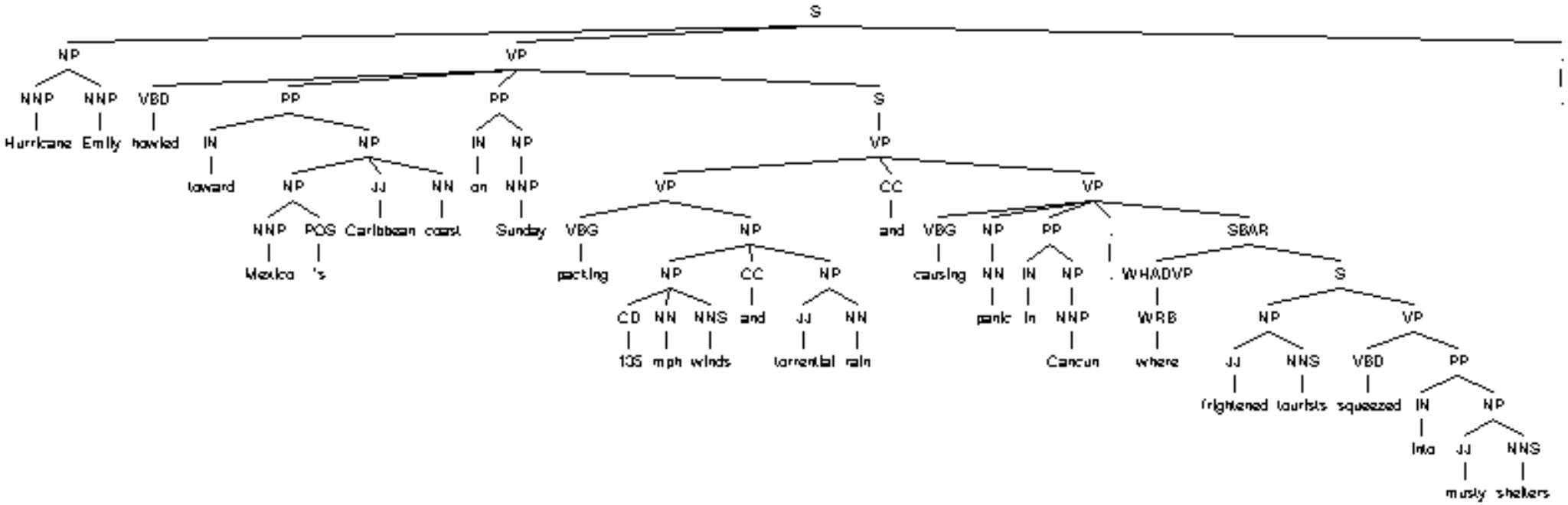
1. What is NLP — NLP Applications

Future?



1. What is NLP — Three views of NLP

- **Classical View: Layered Processing**

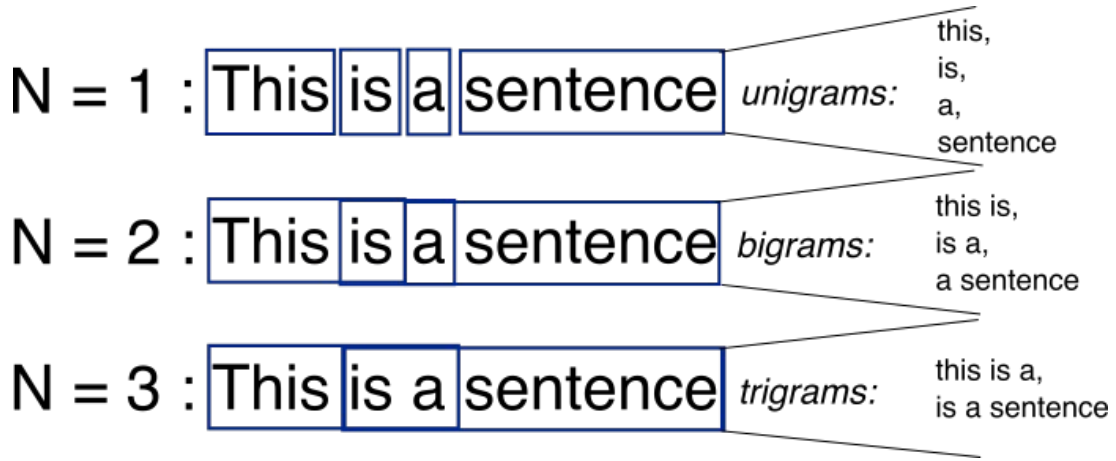


- Statistical/Machine Learning View
- Deep Learning View

1. What is NLP — Three views of NLP

- Classical View: Layered Processing
- **Statistical/Machine Learning View**

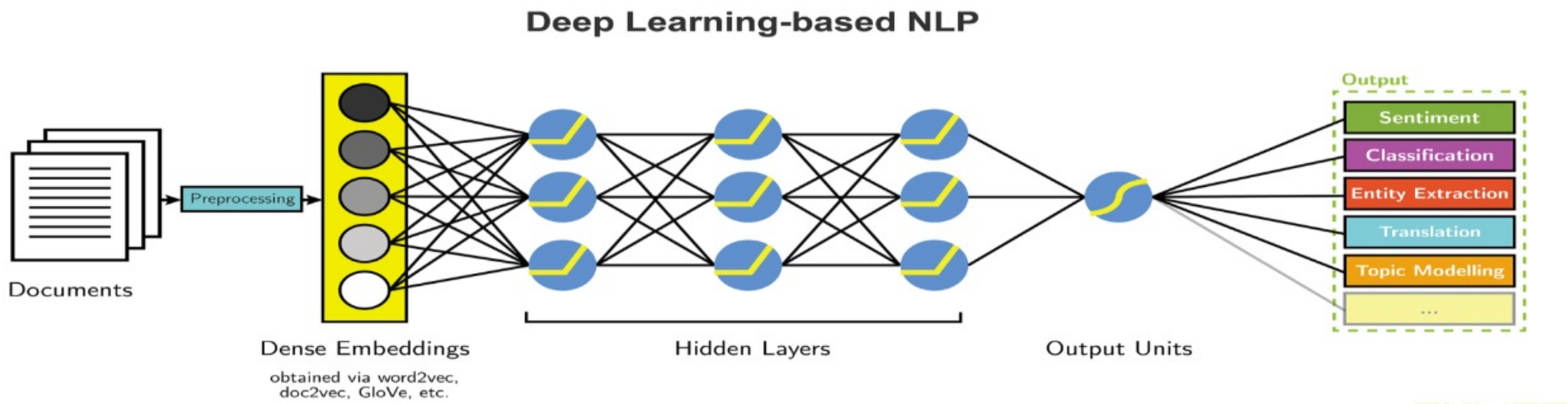
n-gram:



- Deep Learning View

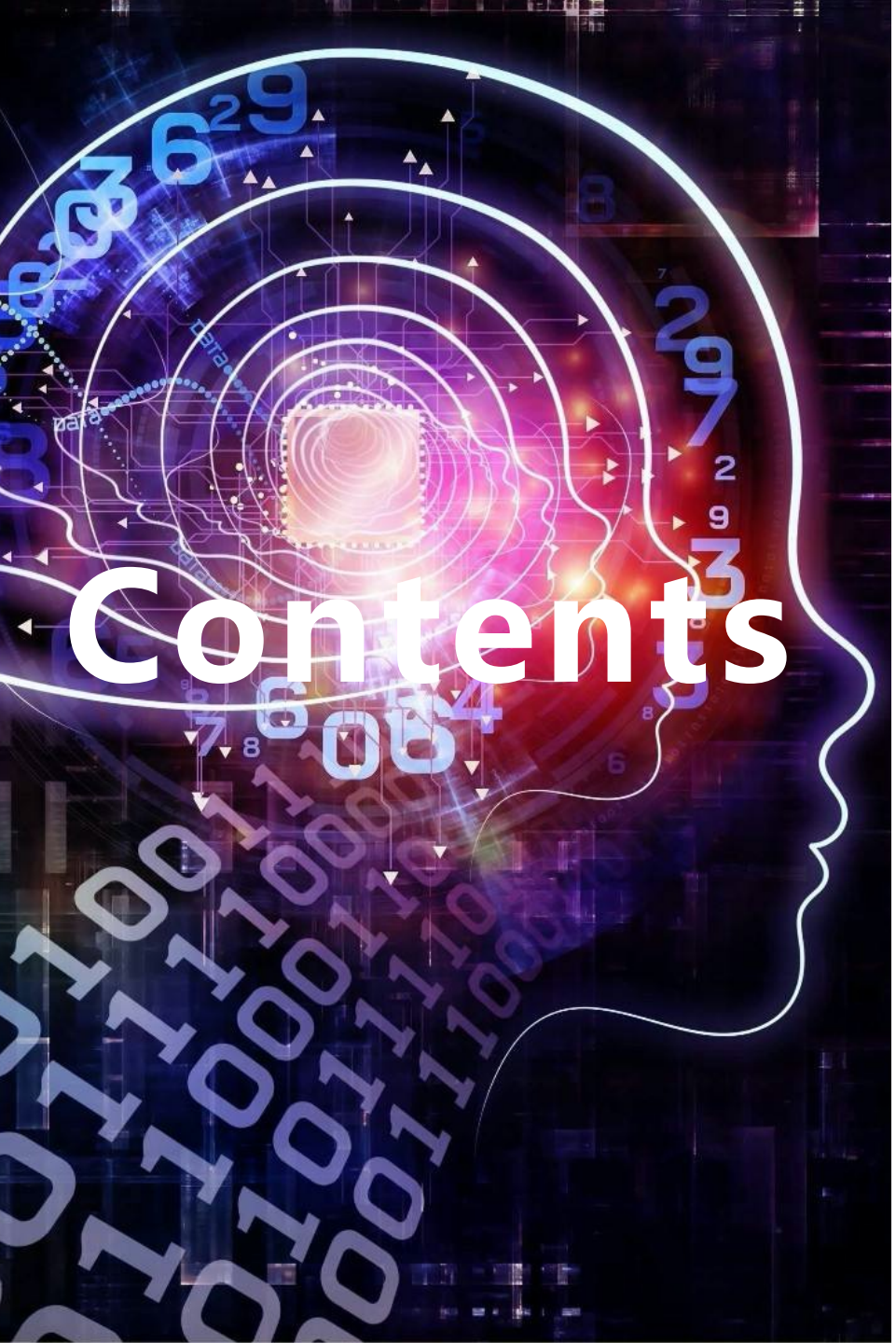
1. What is NLP — Three views of NLP

- Classical View: Layered Processing
- Statistical/Machine Learning View(n-gram)
- **Deep Learning View**



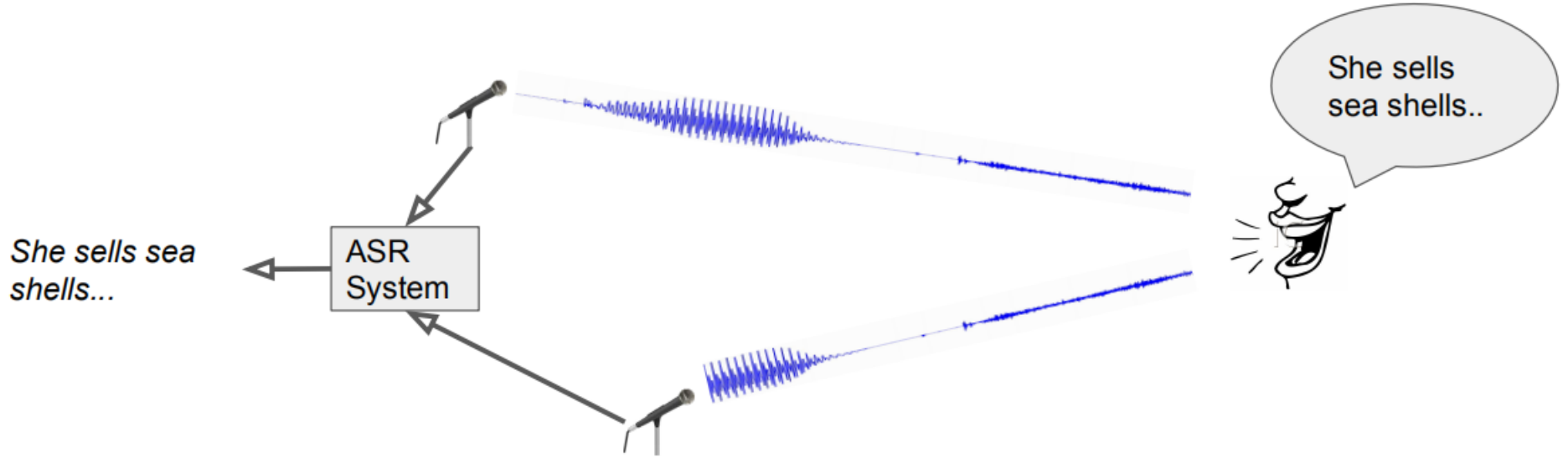
1. What is NLP — Three views of NLP

- Classical View: Layered Processing; Various Ambiguities
- **Statistical/Machine Learning View(n-gram)**
- **Deep Learning View**



ASR

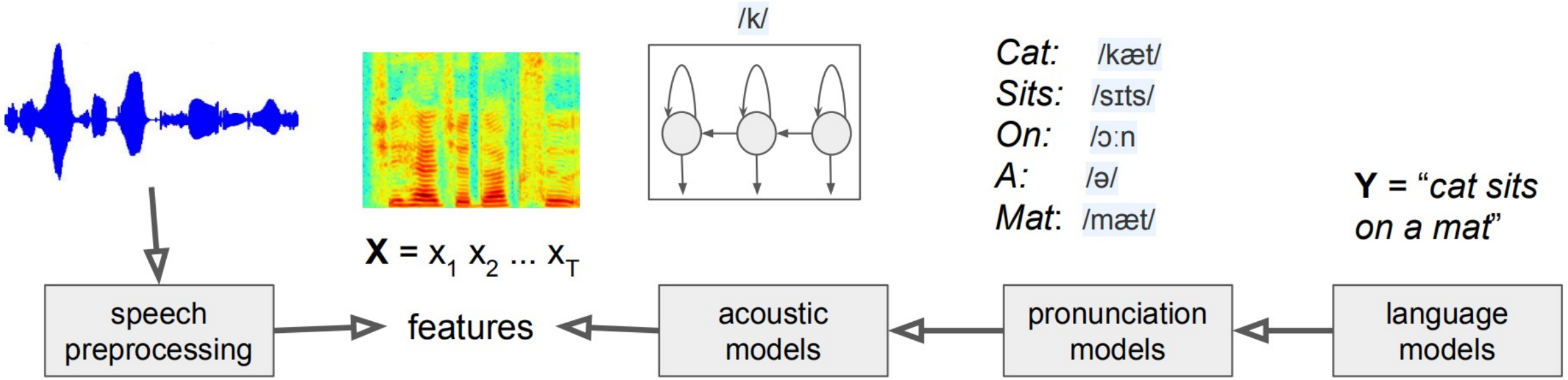
- **Automatic speech recognition (ASR)** is a process by which an acoustic speech signal is converted into a set of words [Rabiner et al., 1993]



2. Speech Recognition — The Classical Ways

ASR —— the classical ways

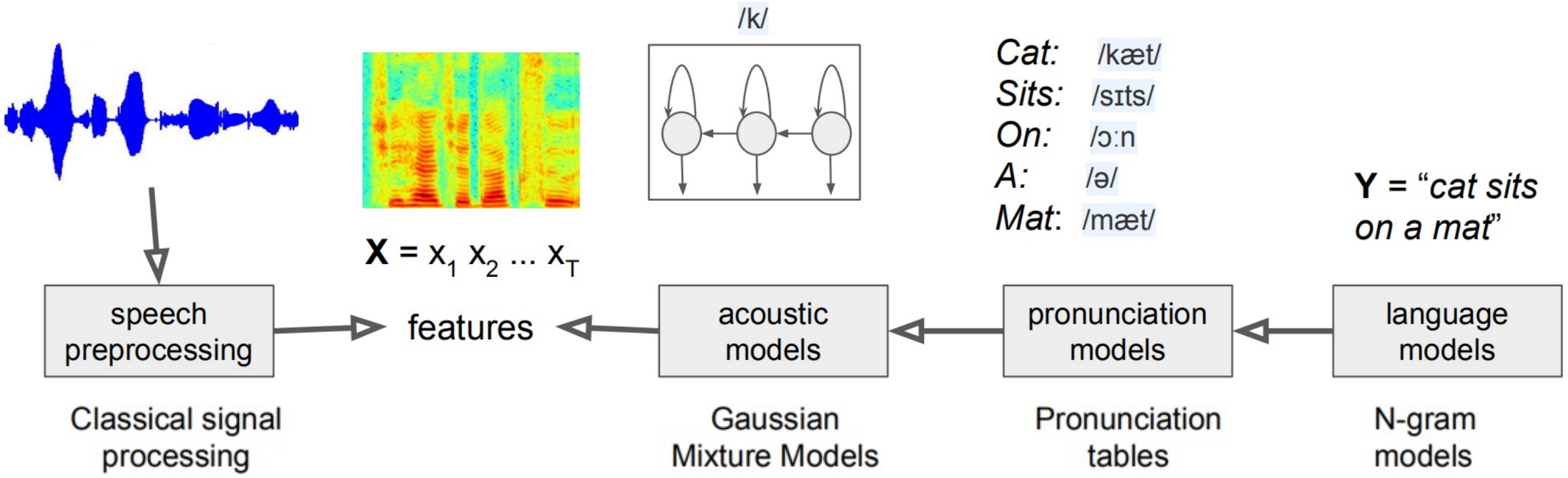
- **Building a statistical model of speech starting from text sequences $Y = y_1 y_2 \dots y_L$ to audio features $X = x_1 x_2 \dots x_T$**



2. Speech Recognition — The Classical Ways

ASR —— the classical ways

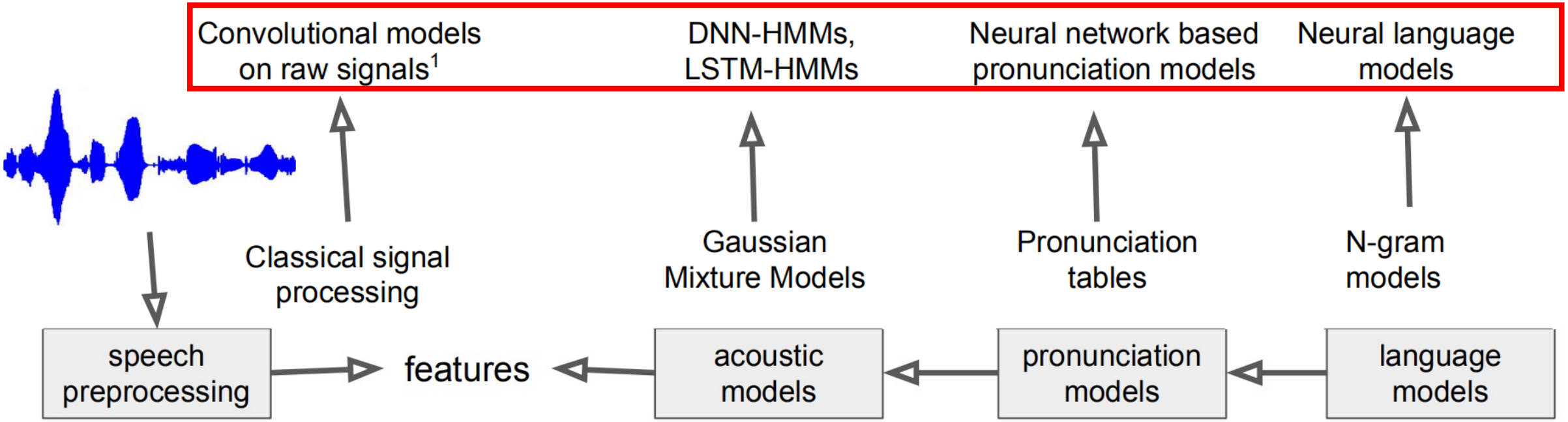
- Different statistical models used in different components



2. Speech Recognition — Neural Network

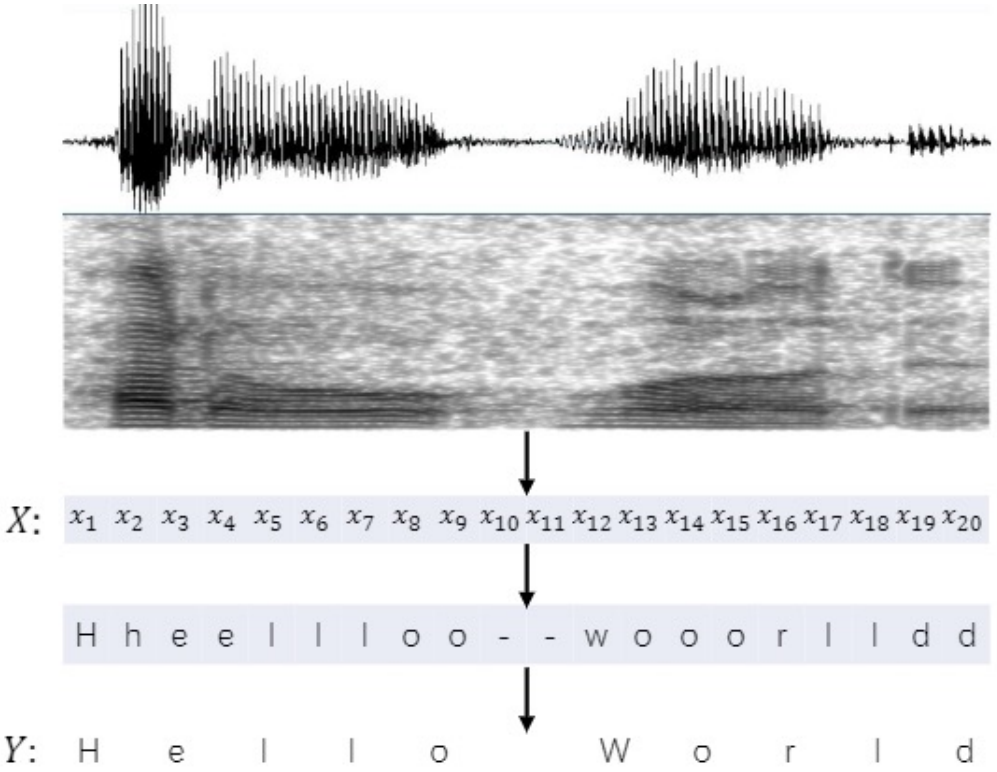
ASR —— the neural network invasion

- **Each of the components seems to be better off with a neural network**



CTC(Connectionist Temporal Classification)

- A probabilitistic model $\mathrm{p}(Y|X)$, where
  - $X = x_1 x_2 \ldots x_T$,
  - $Y = y_1 y_2 \ldots y_L$
  - $T \geq L$



2. Speech Recognition — Language Models

2. Speech Recognition — Language Models

- **Language models :** assign a probability to a sentence

  A = `all of a sudden I notice three guys standing on the sidewalk`

  B = `on guys all I of notice sidewalk three a sudden standing the`

  P(A) > P(B)

- **Application :**
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - Machine Translation:
    - P(**got out of** the car) > P(**got off** the car)
  - Spell Correction
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)

- **Recall**
  - Chain rule : $$\begin{aligned} P(a) &= P(w_1 w_2 \dots w_m) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\dots P(w_m|w_1 w_2 \dots w_{m-1}) \\ &= \prod_{k=1}^{m} P(w_k|w_1 \dots w_{k-1}) \end{aligned}$$

e.g.

$$P(\text{John read a book})$$

$$= P(\text{John}) \times P(\text{read} \mid \text{John}) \times P(\text{a} \mid \text{John read}) \times P(\text{book} \mid \text{John read a})$$

2. Speech Recognition — Language Models

- **Existing models**
  - **N-gram**
  - N-pos
  - NNLM
  - SENNA
  - HLBL
  - RNNLM
  - …

2. Speech Recognition — N-Gram

$$P(\text{John read a book})$$

$$= P(\text{John}) \times P(\text{read} \mid \text{John}) \times P(\text{a} \mid \text{John read}) \times P(\text{book} \mid \text{John read a})$$

- **"John read a _____"**
- Predict the next word, given (n-1) previous words
- **Markov Assumption**: only the n-1 previous words affect the next word. (n-1)th Markov Model or n-gram.
- Determine probability of different sequences by examining training corpus

$$P(a) = \prod_{k=1}^{m} P(w_k \mid w_{k-n+1} \ldots w_{k-1})$$

size of windows is n-1→n-gram

- n=1 unigram
- n=2 bigram
- n=3 trigram

2. Speech Recognition — N-Gram

- n=2 bigram

$$P(John\ read\ a\ book)$$
$$= \boldsymbol{P(John|<START>)} \times P(read|John) \times P(a|read) \times P(book|a) \times \boldsymbol{P(<END>|book)}$$

2. Speech Recognition — N-Gram

- n=2 bigram
- S1: John read Moby Dick
- S2: Mary read a different book
- S3: She read a book by Cher
- P(John read Moby) = 1/3* 1*1/3*0.0001
  - P (John|<start>) = 1/3
  - P (read | John) = 1
  - P(Moby|read) = 1/3
  - P(<end>|Moby) = 0.0001

2. Speech Recognition — Selecting an N

How to select an n?

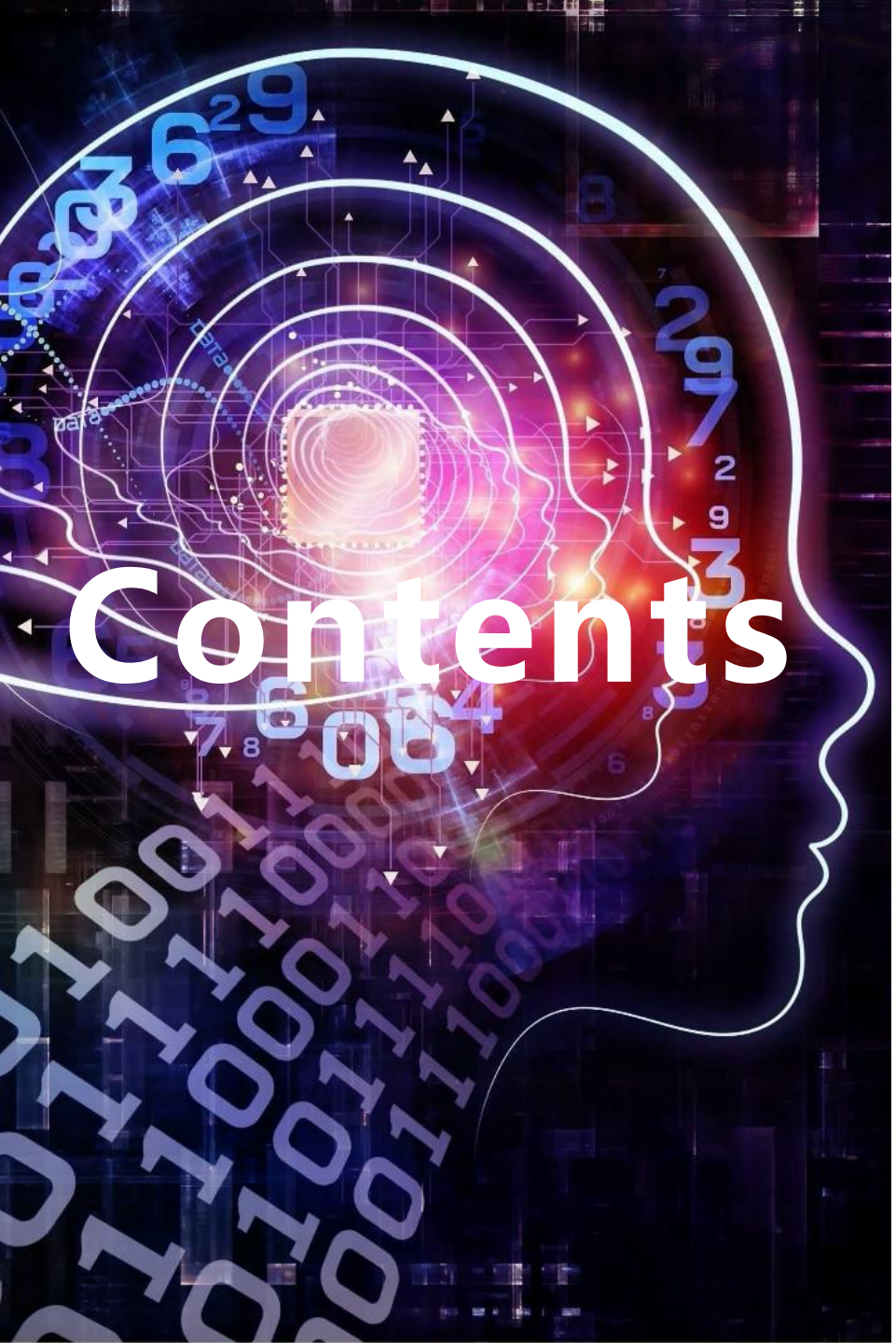
"large green _____"

tree? mountain? frog? car?

"swallowed the large green _____"

pill? broccoli?

- **Reliability vs. Discrimination** 权衡
  - **larger n**: more information about the context of the specific instance (greater discrimination)
  - **smaller n**: more instances in training data, better statistical estimates (more reliability)



How do we represent the meaning of a word?

- Definition: **meaning** (Webster dictionary)
  - the idea that is represented by a word, phrase, etc.
  - the idea that a person wants to express by using words, signs, etc.
  - the idea that is expressed in a work of writing, art, etc.
- Commonest linguistic way of thinking of meaning:

signifier (symbol) ⟺ signified (idea or thing)

= denotational semantics cf. connotational semantics – implied not literal

How do we have usable meaning in a computer ?

- Common **solution**: Use e.g. WordNet, a thesaurus containing lists of synonym sets and hypernyms (“is a” relationships)

e.g. synonym sets containing “good”:

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g. hypernyms of “panda”:

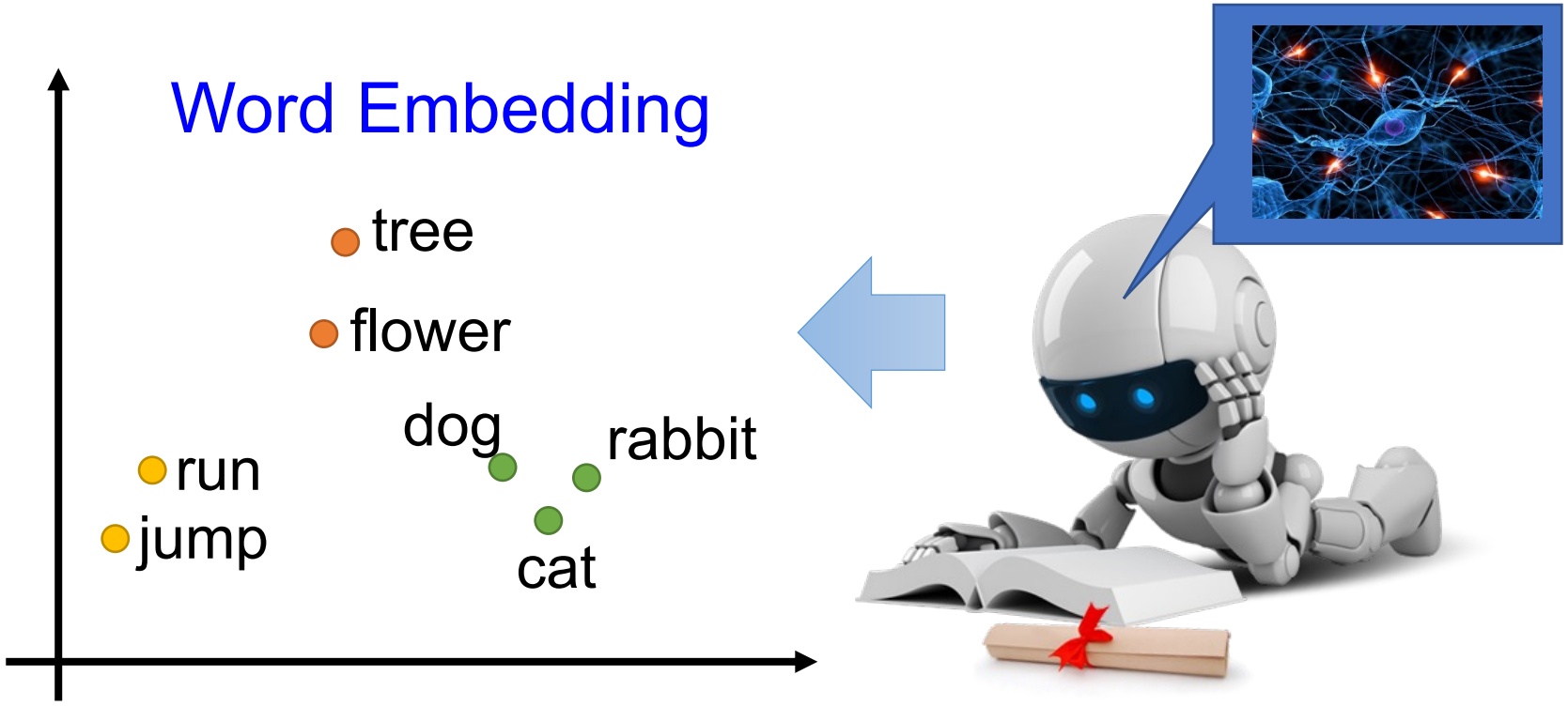
```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

Problems with resources like WordNet

- Great as a resource but missing nuance.
  - e.g. "proficient" is listed as a synonym for "good". This is only correct in some contexts.
- Missing new meanings of words
  - e.g. wicked, badass, nifty, wizard, genius, ninja, bombest
  - Impossible to keep up-to-date!
- Subjective
- Requires human labor to create and adapt
- Can't compute accurate word similarity

- Machine learns the meaning of words from reading a lot of documents without supervision



Representing words as discrete symbols

- In traditional NLP, we regard words as discrete symbols.

Means one 1, the rest 0s

- Words can be represented by one-hot vectors:

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

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

- Vector dimension = number of words in vocabulary e.g. 500,000

3. Text Processing — Discrete Symbols

Problems with discrete symbols

Example: In web search, if user searches for "Seattle **motel**", we would like to match documents containing "Seattle **hotel**".

But:

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

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

orthogonal

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

Solution :

- Could try to rely on WordNet's list of synonyms to get similarity ?
  - But it is well-known to fail badly: incompleteness, etc.
- Instead: **learn to encode similarity in the vectors themselves**

3. Text Processing — Word2Vec

One hot

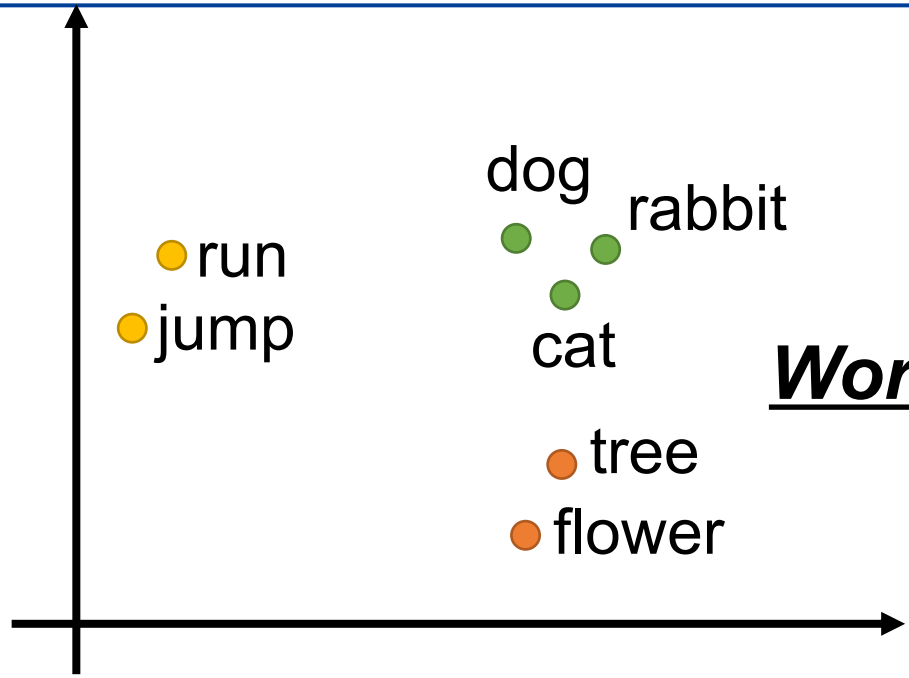
apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

cat = [0 0 1 0 0]

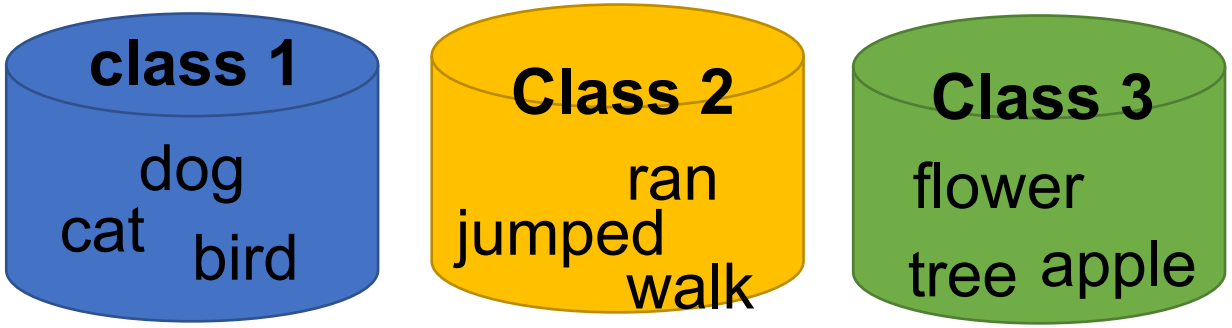
dog = [0 0 0 1 0]

elephant = [0 0 0 0 1]



Word2Vec

Word Class



3. Text Processing — Distributional Semantics

- **Distributional semantics**: A word's meaning is given **by the words that frequently appear close-by.**
  - "**You shall know a word by the company it keeps**" (J. R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical NLP!

- When a word w appears in a text, its context is the set of words that appear nearby within a fixed-size window.
- Use the many contexts of w to build up a representation of w

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

3. Text Processing — Word Vectors

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

motel = [0.792, −0.177, −0.107, 0.109, 0.542, ...]
hotel = [0.883, -0.154, −0.125, 0.1589, 0.613, ...]

similar

Note: word vectors are sometimes called **word embeddings**. They are a distributed representation.

- **How to get word vectors?**
  - Language models
  - **Word2Vec**

3. Text Processing — Word2Vec

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

Basic Ideas:

- We have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position in a text, which has a **center word c** and **context words o**
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability

3. Text Processing — CBOW&Skip-Gram

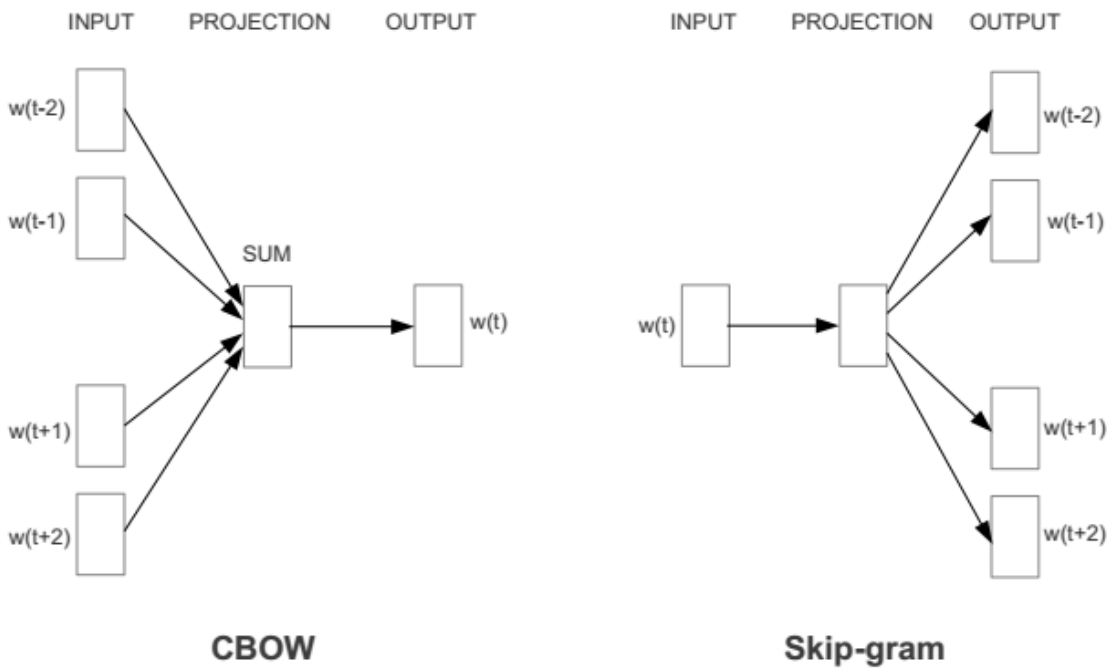
Two model variants:

- Continuous Bag of Words (CBOW)

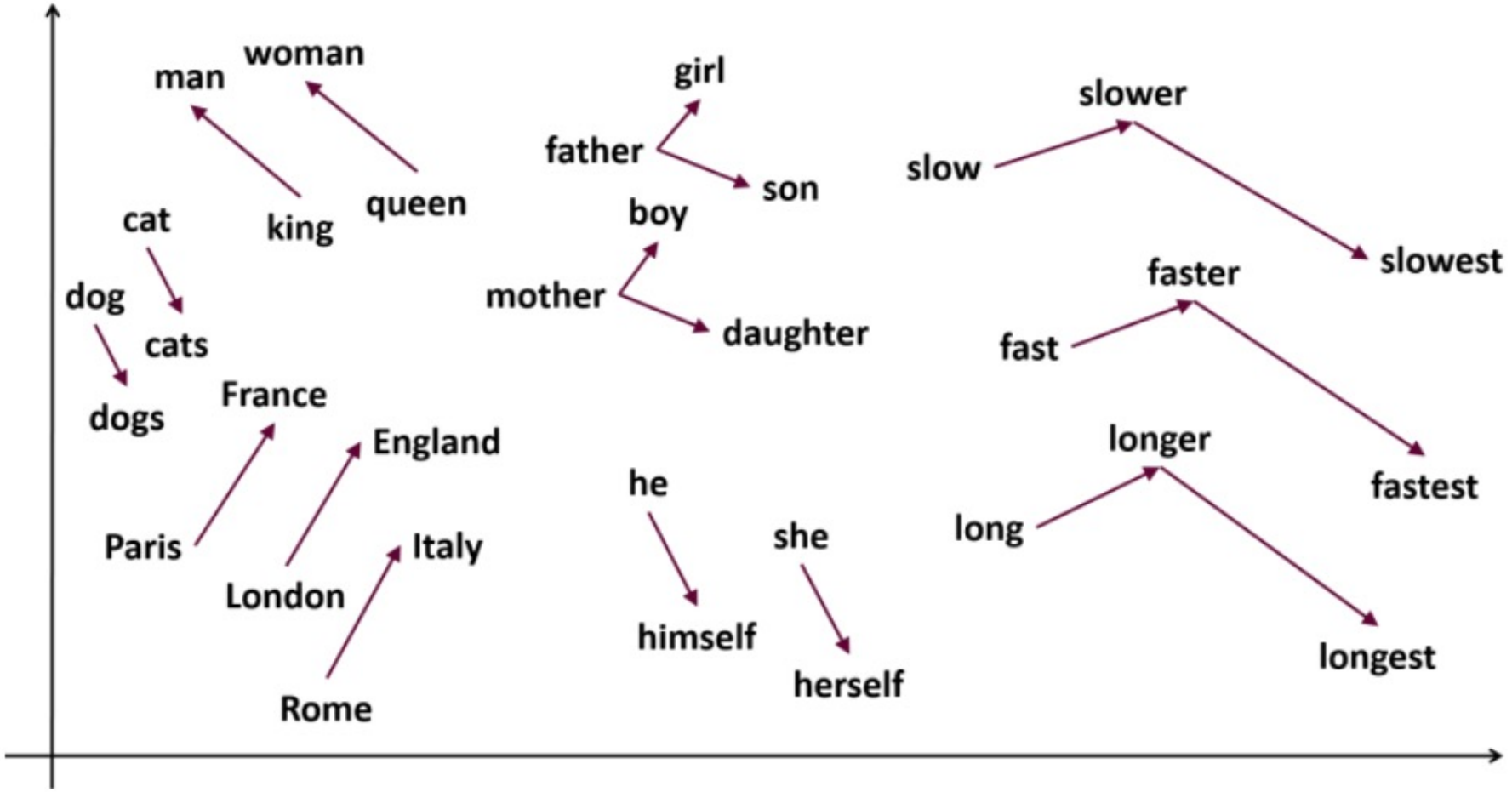
  Predict **center word** from **context words**

- Skip-gram (SG)

  Predict **context words** given **center word**



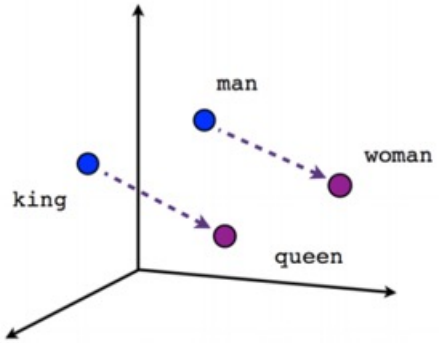
3. Text Processing — Result

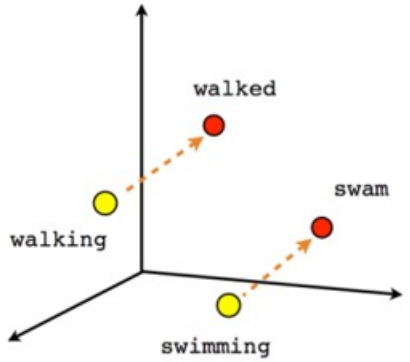


Low-dimensional vector spaces

3. Text Processing — Application(1)

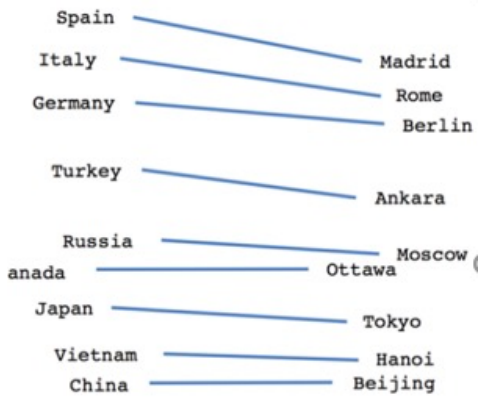
- **Capture the relations between words**





Male-Female

Verb tense



Country-Capital

woman – man ≈ **queen** - king

walked – walking ≈ **swam** – swimming

China – Beijing ≈ **Japan** – Tokyo

```
1 model_english.most_similar(positive=['woman','king'],negative=['man'],topn=1)
```

executed in 56ms, finished 10:46:55 2021-04-26

```
[('queen', 0.7515912055969238)]
```

```
1 model_english.most_similar(positive=['walked','swimming'],negative=['walking'],topn=1)
```

executed in 55ms, finished 10:47:36 2021-04-26

```
[('swam', 0.7947896718978882)]
```

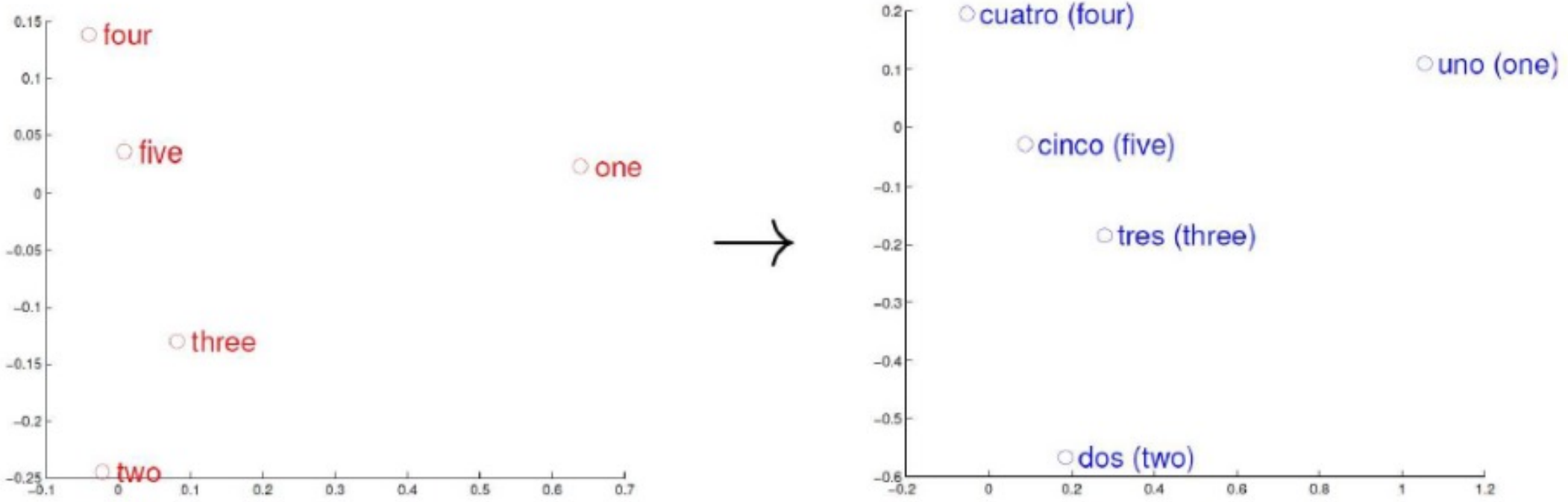
```
1 model_english.most_similar(positive=['China','Tokyo'],negative=['Beijing'],topn=1)
```

executed in 55ms, finished 10:48:12 2021-04-26

```
[('Japan', 0.8739495873451233)]
```

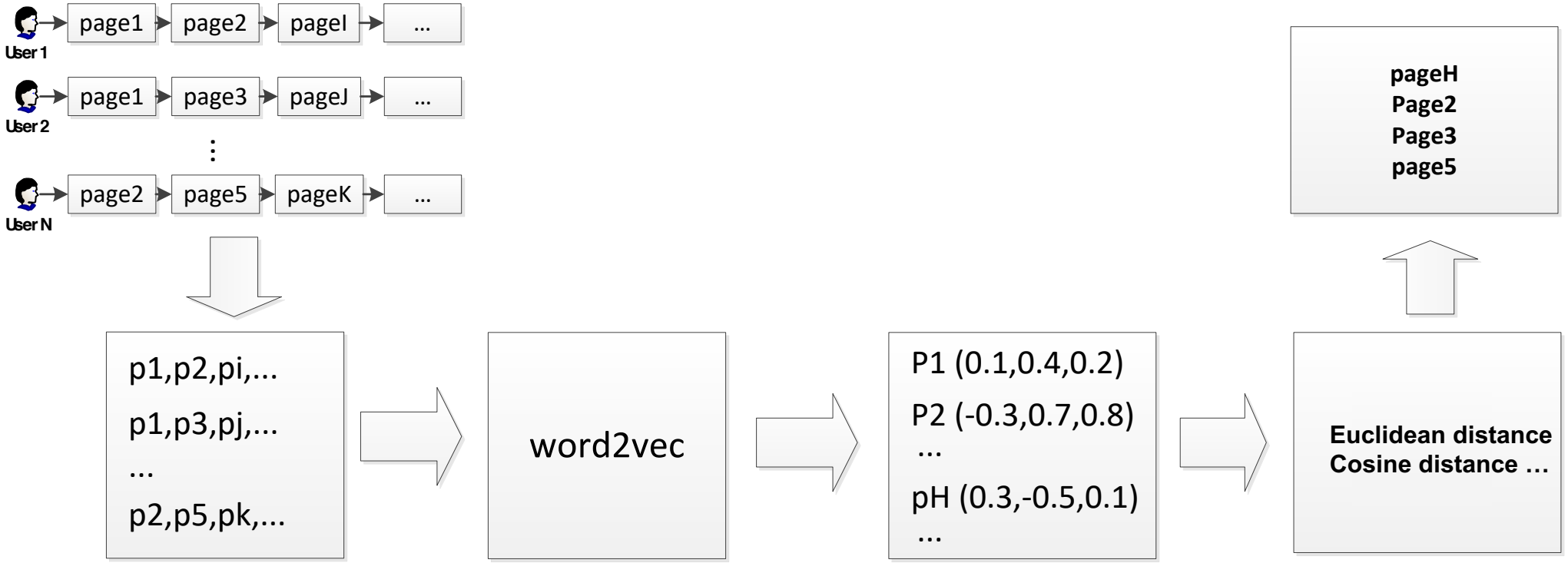
3. Text Processing — Application(2)

- **Machine translation**
  - One vector space(English) → another vector space(Spanish)
  - Accuracy : 90%



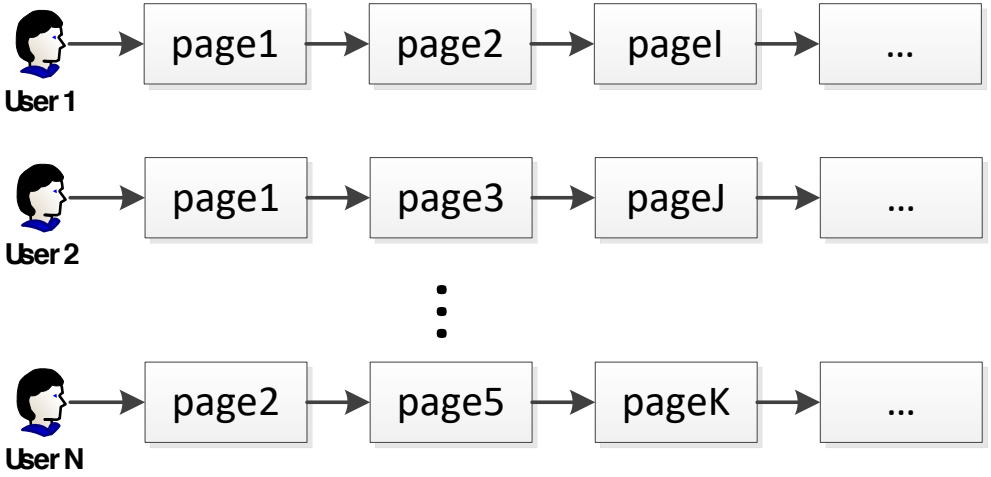
3. Text Processing — Application(3)

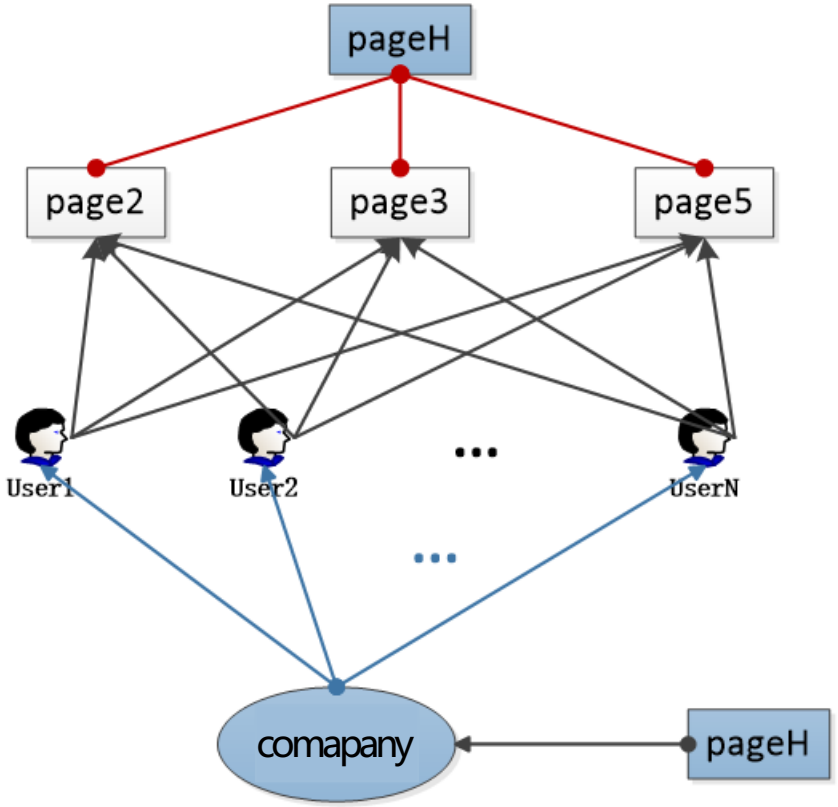
- **Recommendation system**
  - Every page is considered as a 'word'



3. Text Processing — Application(3)

- **Recommendation system**
  - If PageH is close to Page2,3,4 in the vector space
  - Recommend PageH to those users who often visit Page2,3,4





3. Text Processing — Word2vec's Problem

Word2vec

- Pros：
  - fast, efficient to train
  - easily available online with code and pretrained embeddings
- Cons：
  - static embeddings，can't model **Polysemy**

1.I ***read*** the book yesterday. past tense
2.Can you ***read*** the letter now? present tense

The uses of words vary across <u>linguistic contexts!</u>

Embeddings from Language Models(ELMo)

- NAACL' 18: Deep contextualized word representations
- Context Matters!

Embeddings from Language Models(ELMo)

- Key idea:
  - Train an LSTM-based language model on large corpus
  - Use the hidden states of the LSTM for each token to compute a vector representation of each word

3. Text Processing — ELMo

Pretrain two LM

- Forward LM
- Backward LM

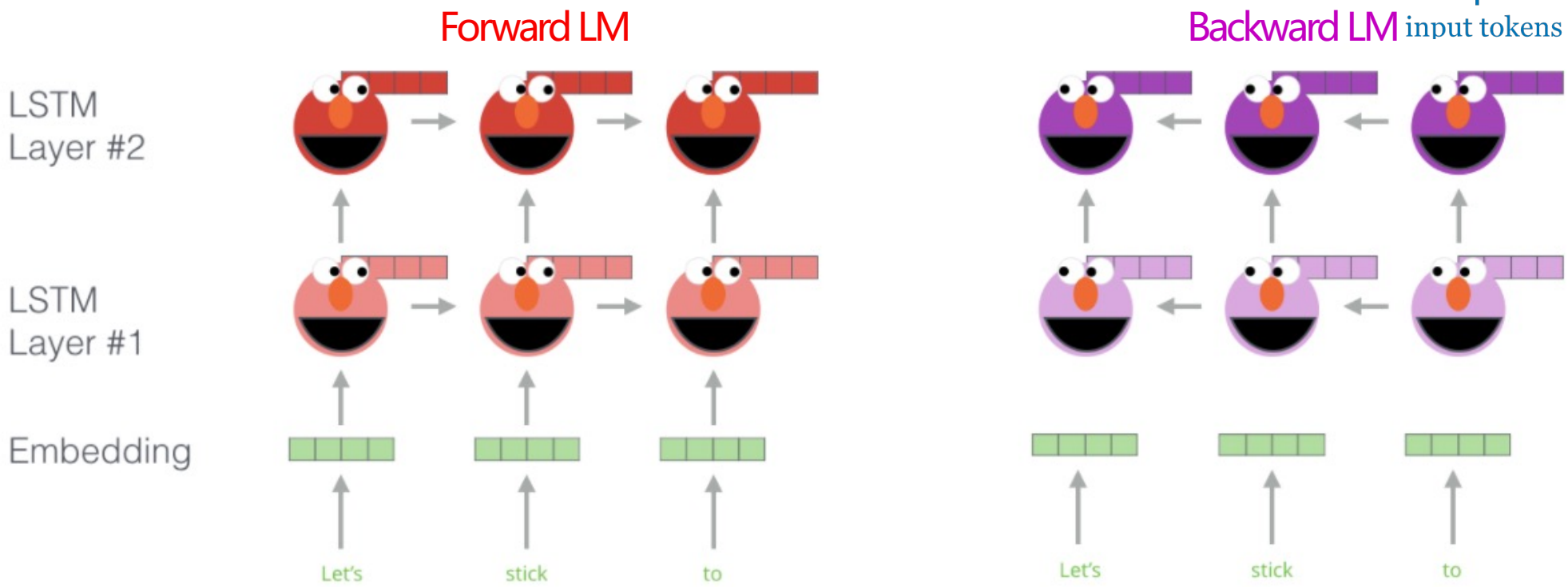
$$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right)$$

tokens in the sentence

LSTM parameters

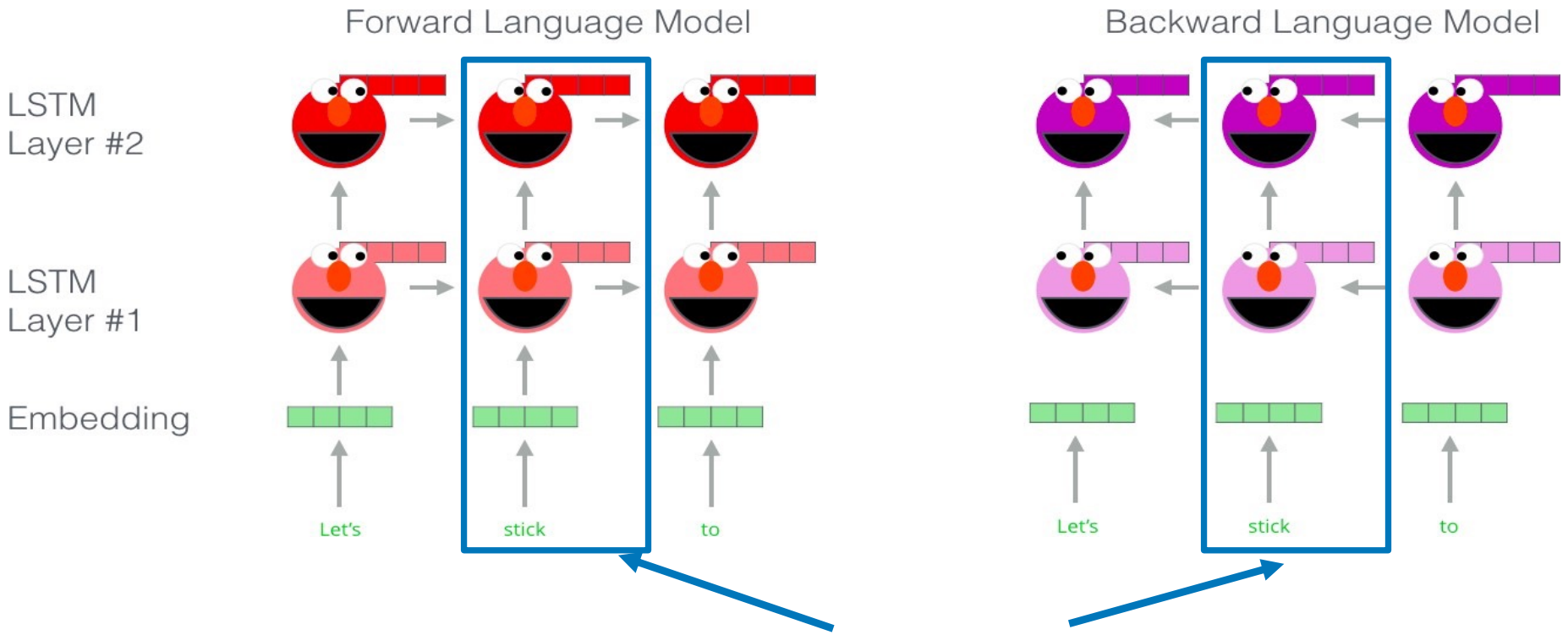
input tokens

softmax



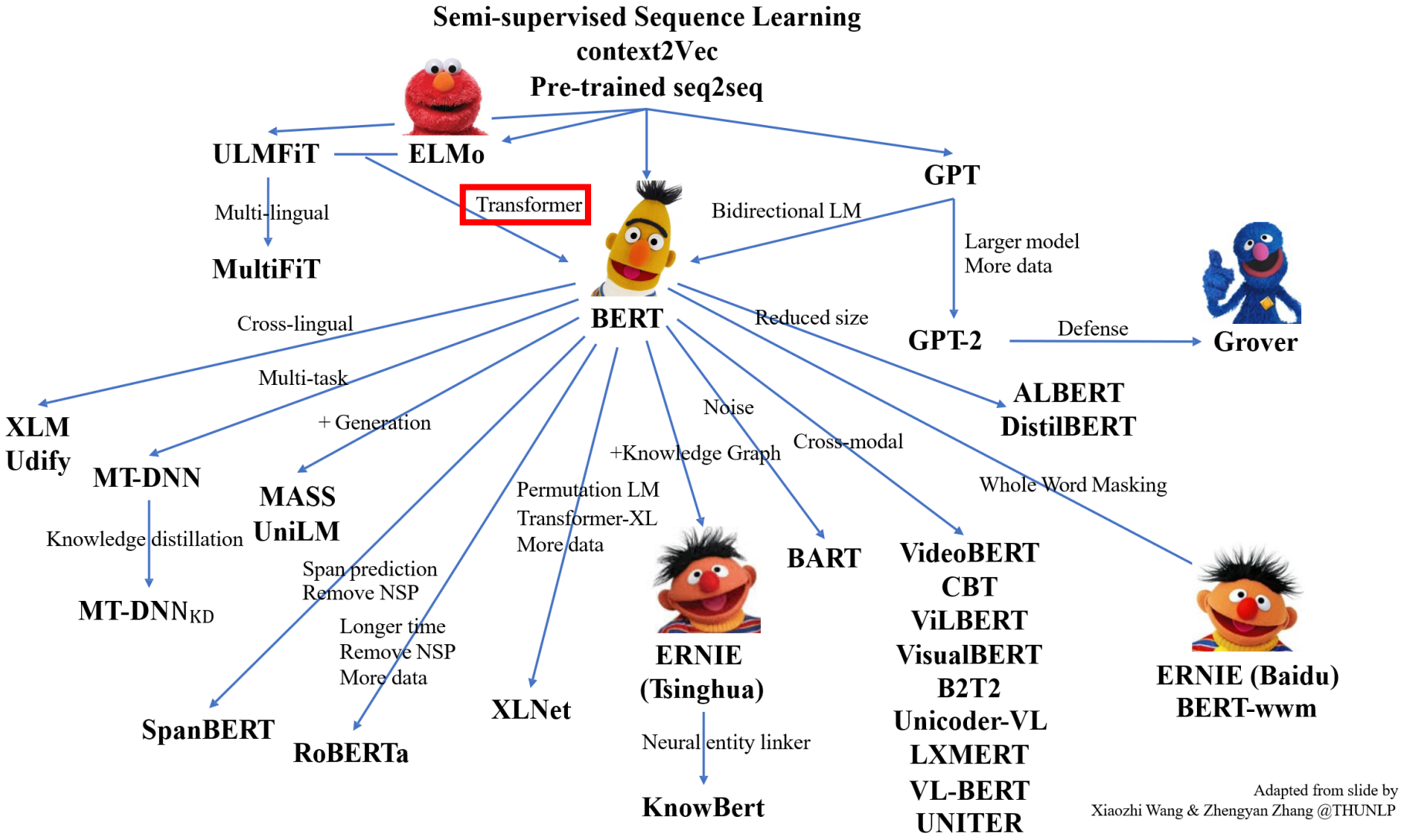
3. Text Processing — ELMo

After training LM



To get the ELMO embedding of a word ("stick"):
Concatenate forward and backward embeddings and take weighted sum of layers

3. Text Processing — context2Vec



3. Text Processing — Transformers

All of these models are Transformer models

ELMo
Oct 2017
Training:
800M words
42 GPU days

GPT
June 2018
Training
800M words
240 GPU days

BERT
Oct 2018
Training
3.3B words
256 TPU days
~320–560
GPU days

GPT-2
Feb 2019
Training
40B words
~2048 TPU v3
days according to
a reddit thread

XL-Net,
ERNIE,
Grover
RoBERTa, T5
July 2019—

(slide credit: Stanford CS224N, Chris Manning)

3. Text Processing — BERT

Bidirectional Encoder Representations from Transformers(BERT)

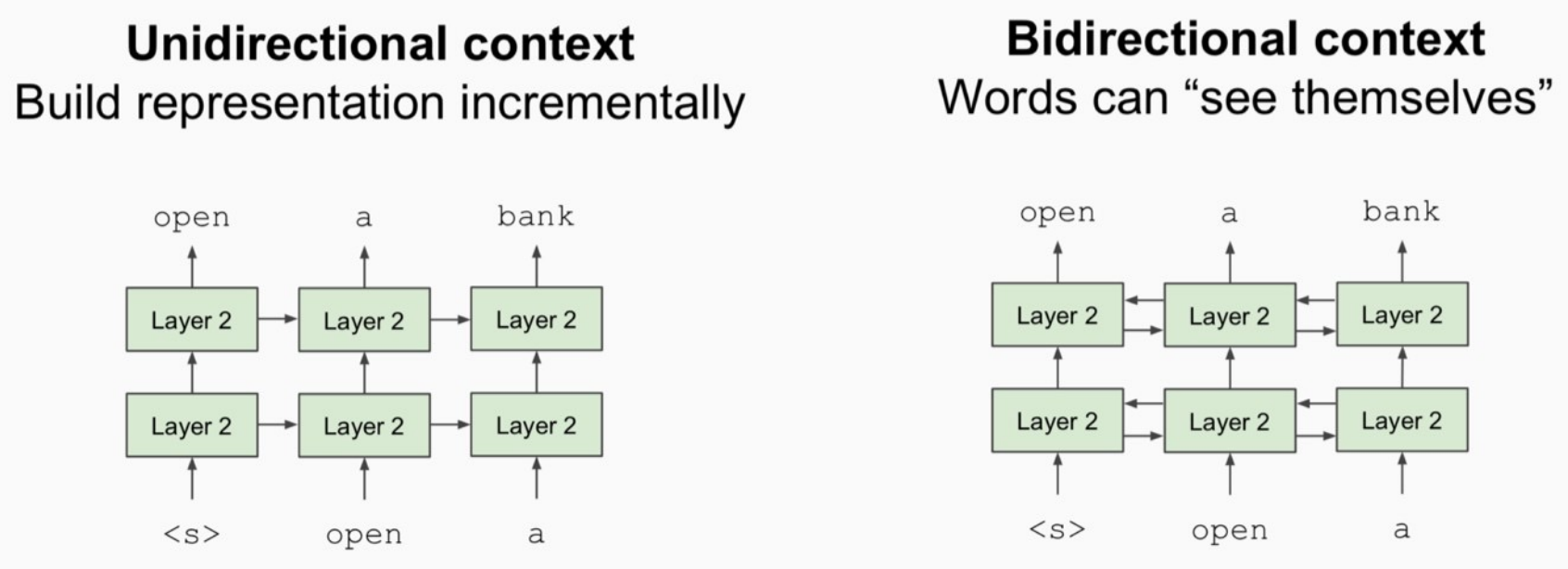
- First released in Oct 2018.
- NAACL' 19: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

How is BERT different from ELMo?

1. Two unidirectional context vs bidirectional context
2. LSTMs vs Transformers (will explain more later)
3. The weights are not frozen, fine-tuning

Bidirectional encoders

- Language models only use left context or right context although ELMo used two independent LMs from each direction
- Language understanding is bidirectional



3. Text Processing — BERT

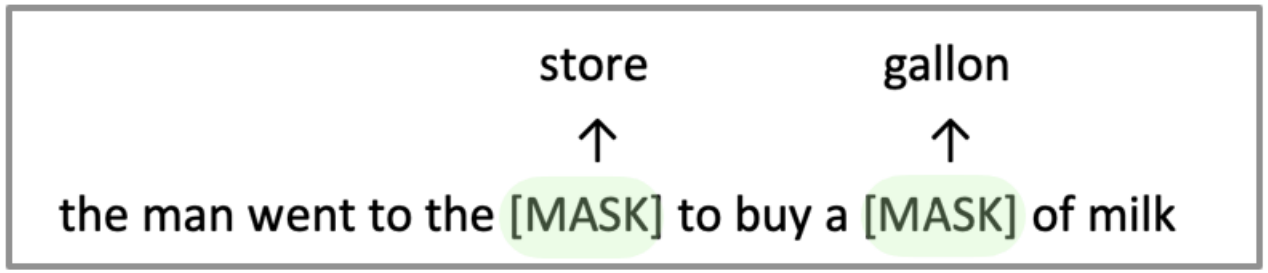
How to pre-train BERT?

- **Task 1:** Masked language models (MLMs)
- **Task 2:** Next sentence prediction (NSP)

3. Text Processing — BERT Pre-training Task 1

Masked language models (MLMs)

- Bidirectional conditioning would allow each word to indirectly **see itself** in a multi-layered context
- Solution: Mask out 15% of the input words, and then predict the masked words



- Too little masking: too expensive to train
- Too much masking: not enough context

3. Text Processing — BERT Pre-training Task 2

Next sentence prediction (NSP)

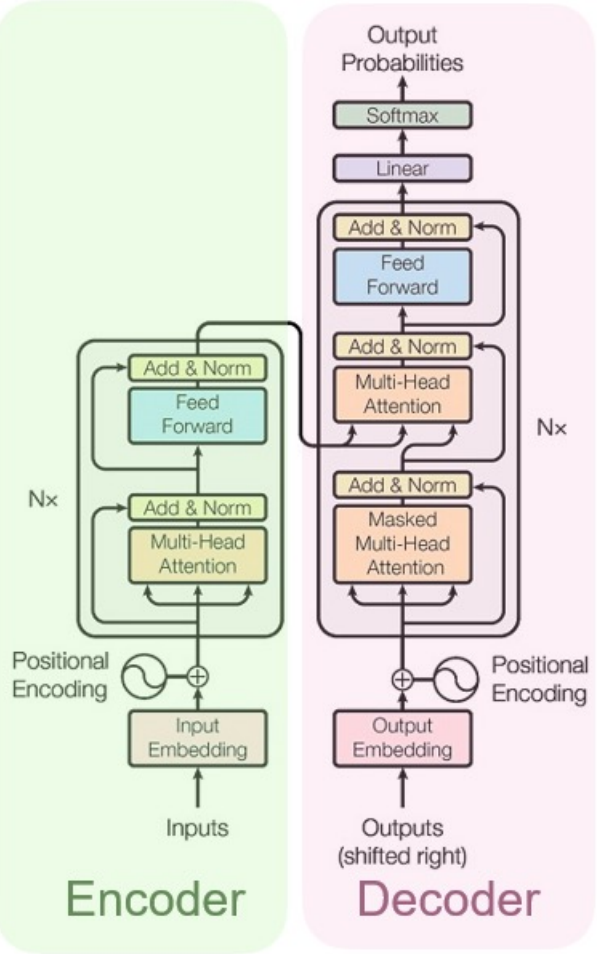
always sample two sentences predict whether the second sentence is followed after the first one

$$\text{Input} = \texttt{[CLS] the man went to [MASK] store [SEP]}$$
$$\texttt{he bought a gallon [MASK] milk [SEP]}$$
$$\text{Label} = \texttt{IsNext}$$

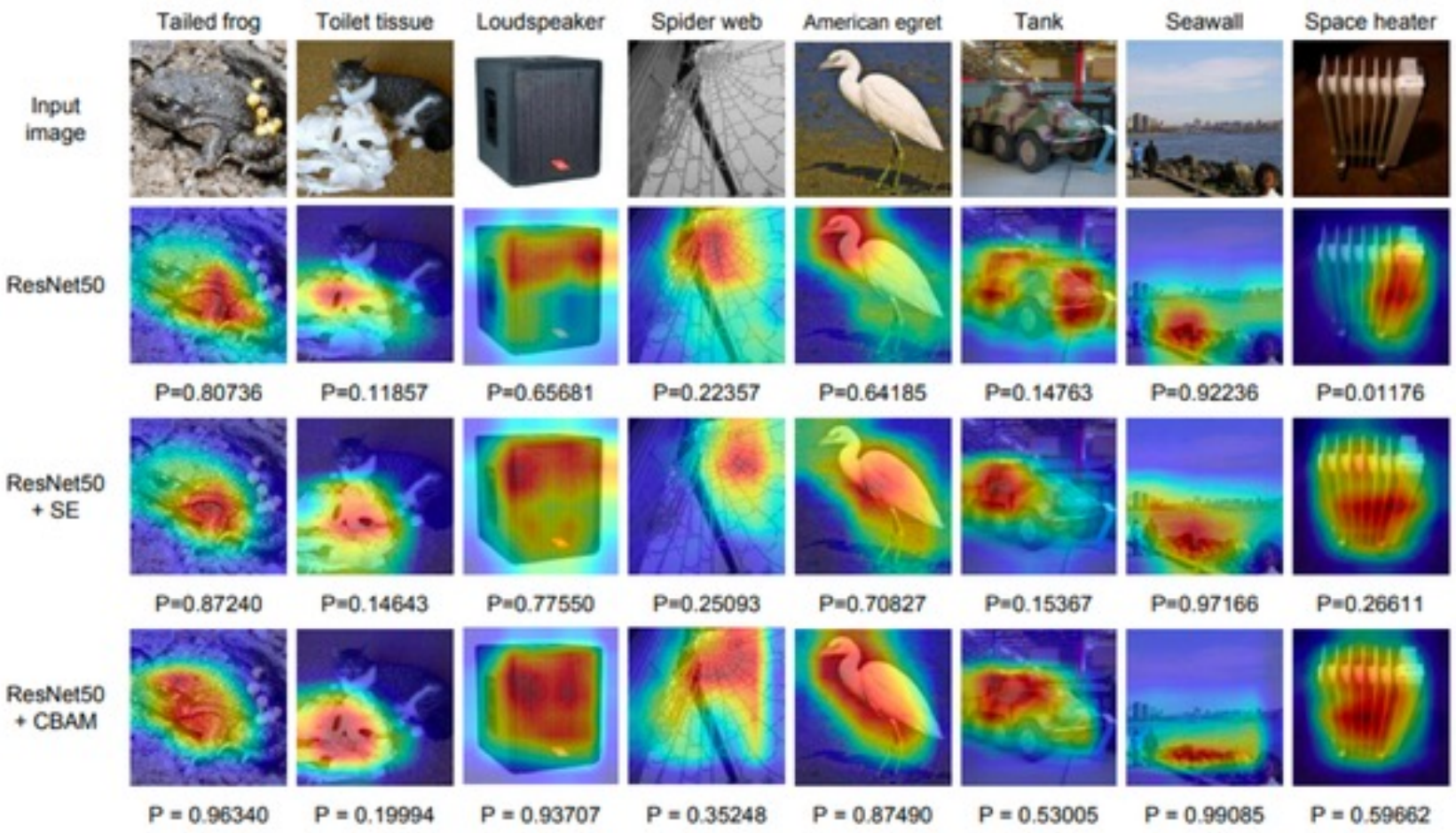
$$\text{Input} = \texttt{[CLS] the man [MASK] to the store [SEP]}$$
$$\texttt{penguin [MASK] are flight ##less birds [SEP]}$$
$$\text{Label} = \texttt{NotNext}$$

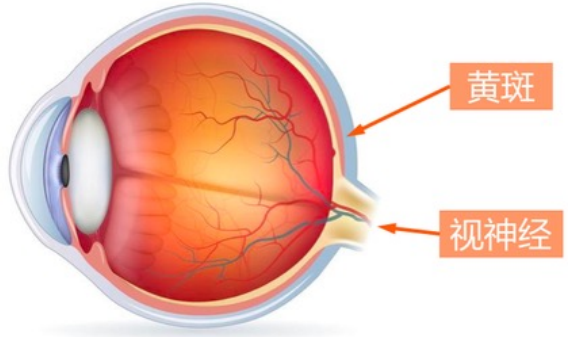
3. Text Processing — Transformers

- NIPS' 17: **Attention is All You Need**
- encoder - decoder framework
- Used as the base model of BERT (encoder only)
- Key idea: Multi-head self-attention
- No recurrence structure any more so it trains much faster



3. Text Processing — Attention





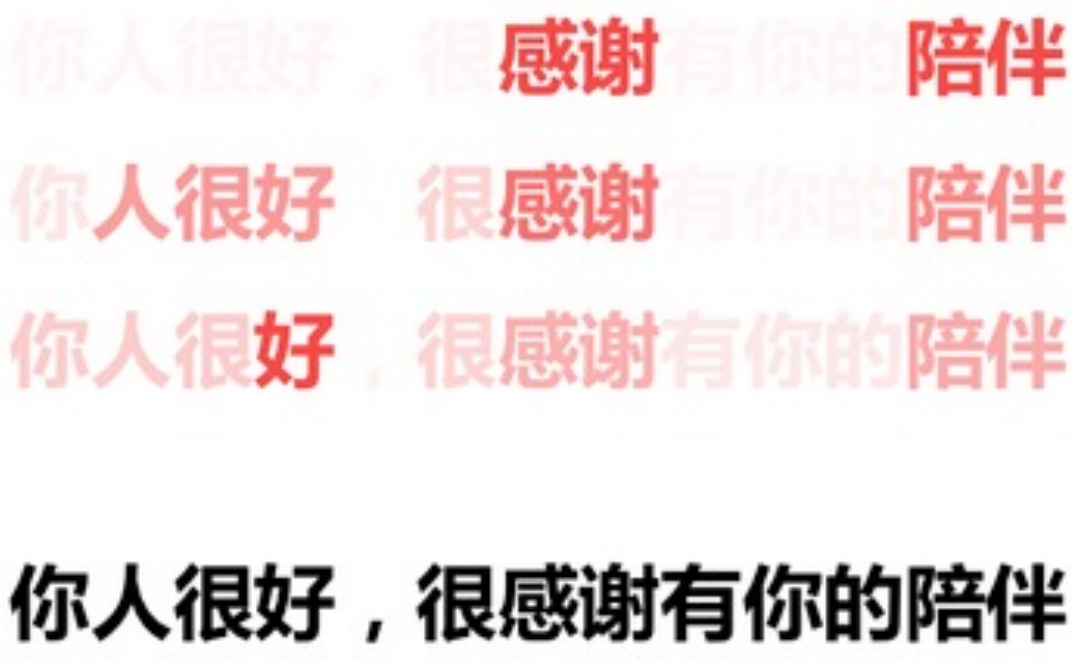
3. Text Processing — Attention

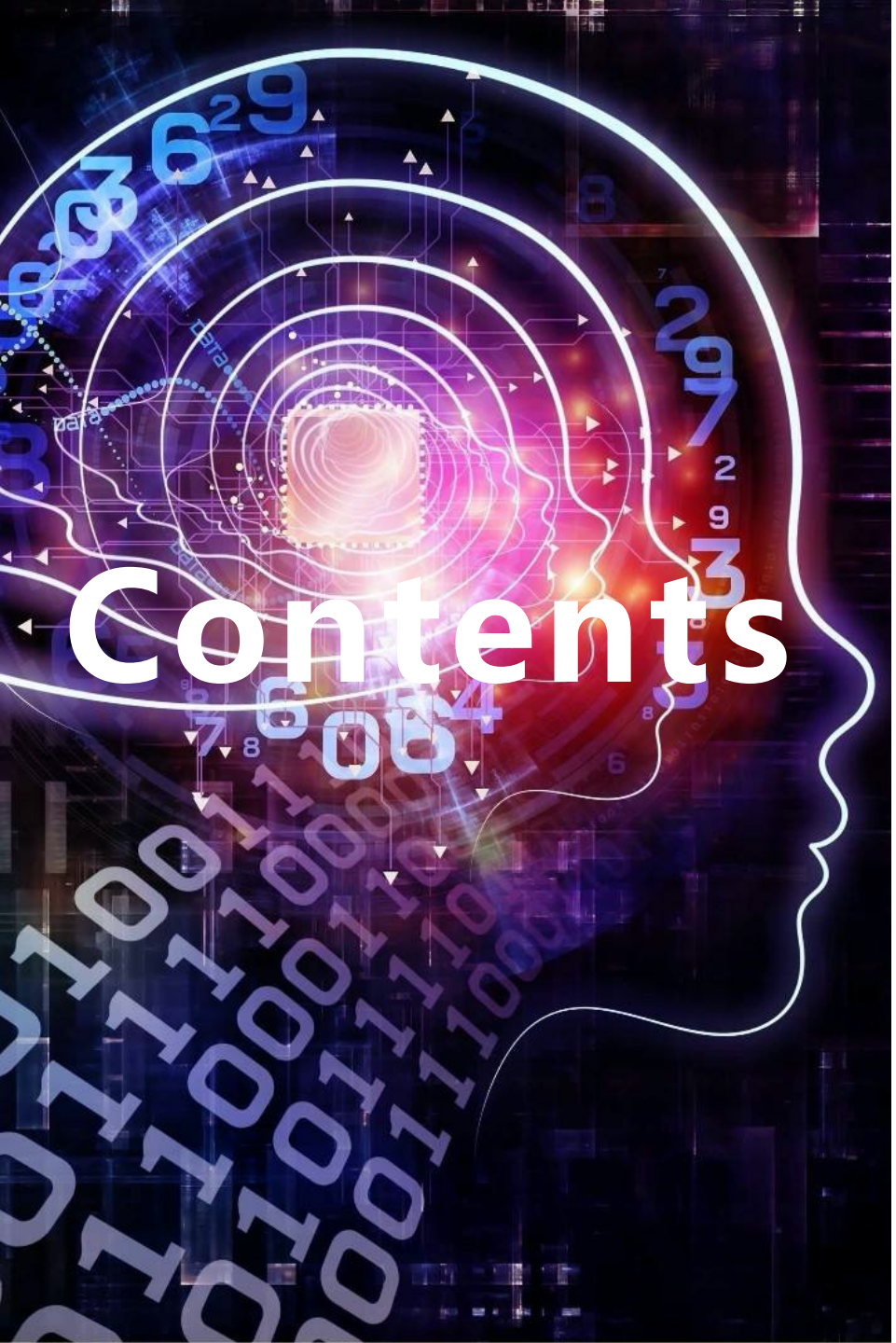
3. Text Processing — Attention

这台手机是一款性价比较高的千元入门机，拥有性能强劲的处理器、不过不支持红外NFC，随着这款机型的降价，现在又变得香起来，对于你这种全面屏爱好者是个不错的选择，机身背面采取的是当下最流行的亮樱桃红色，建议购买4GB+64GB版本。

这台手机是一款性价比较高的千元入门机，拥有性能强劲的处理器、不过不支持红外NFC，随着这款机型的降价，现在又变得香起来，对于你这种全面屏爱好者是个不错的选择，机身背面采取的是当下最流行的亮樱桃红色，建议购买4GB+64GB版本。

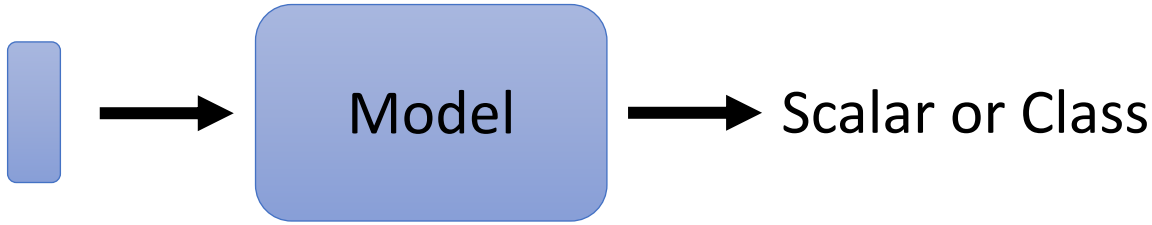
3. Text Processing — Attention



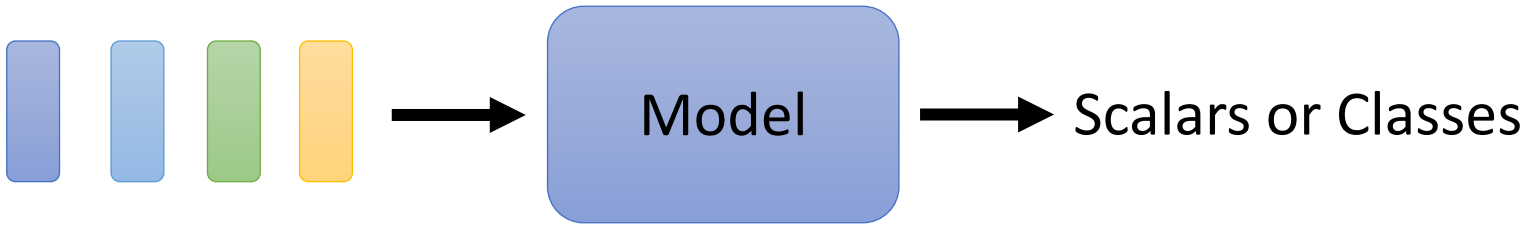


Sophisticated Input

- Input is **a vector**



- Input is **a set of vectors**

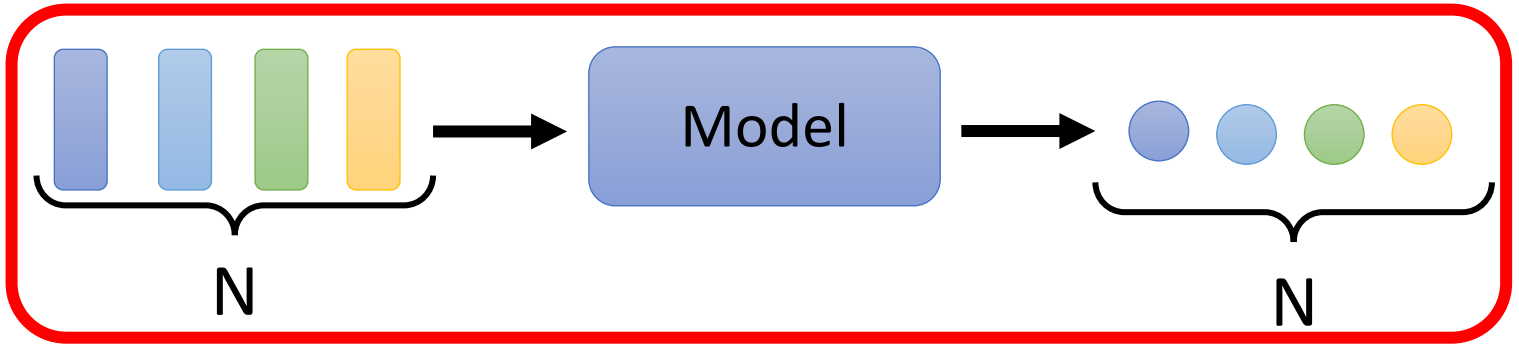


(may change length)

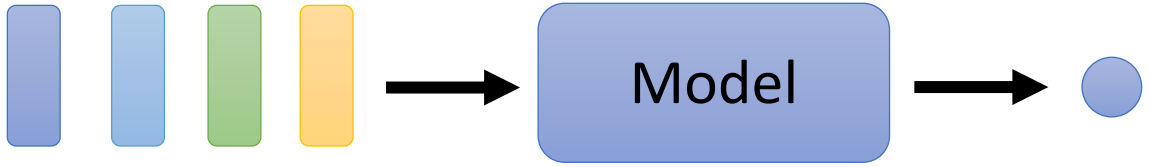
HUNG-YI LEE ML（P77-130）

What is the output?

- Each vector has a label.

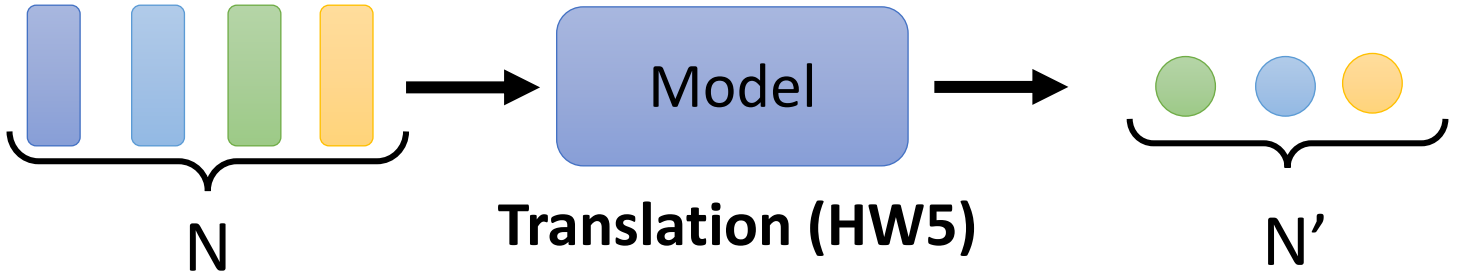


- The whole sequence has a label.



- Model decides the number of labels itself.

seq2seq



Sequence Labeling

Is it possible to consider the context?

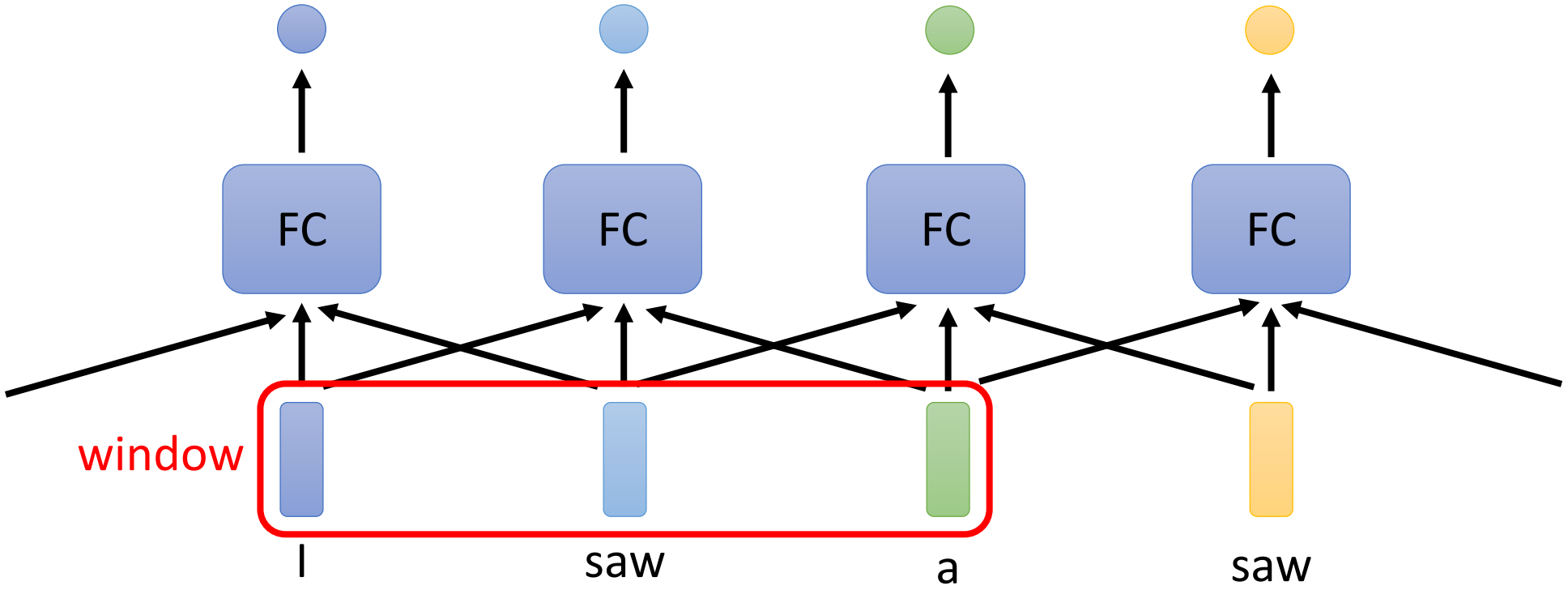
FC can consider the neighbor

How to consider the whole sequence?

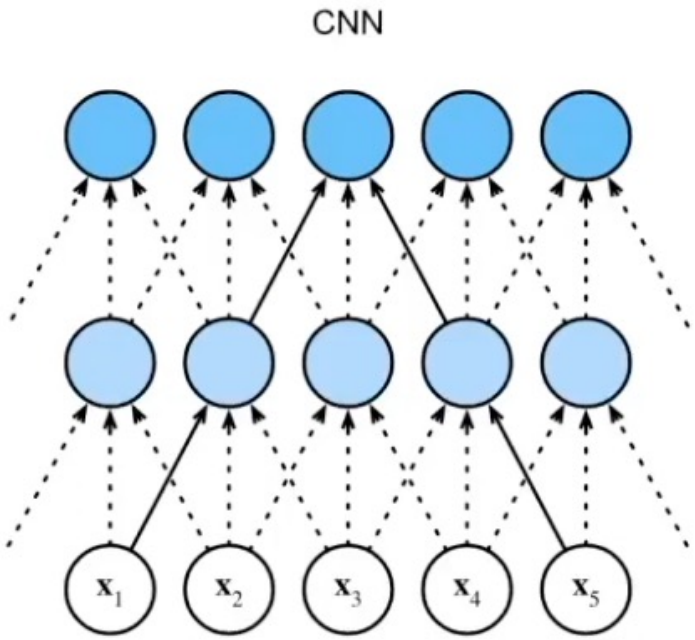
a window covers the whole sequence?

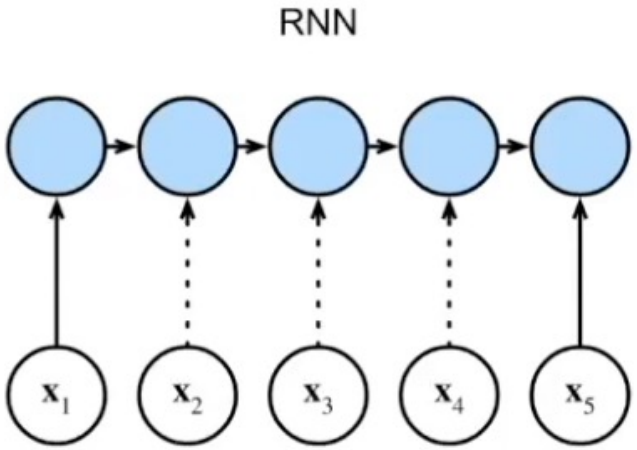
FC

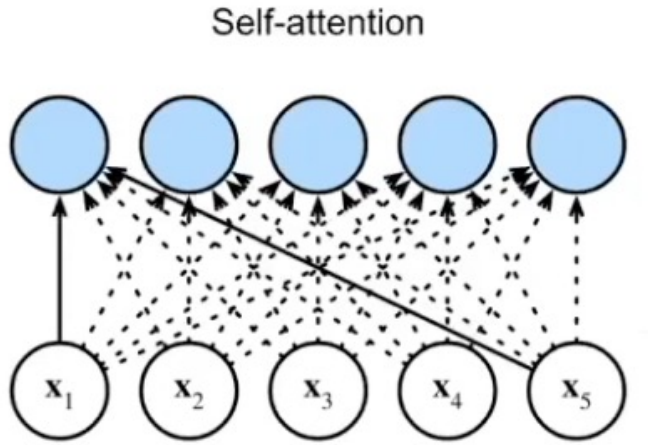
Fully-connected



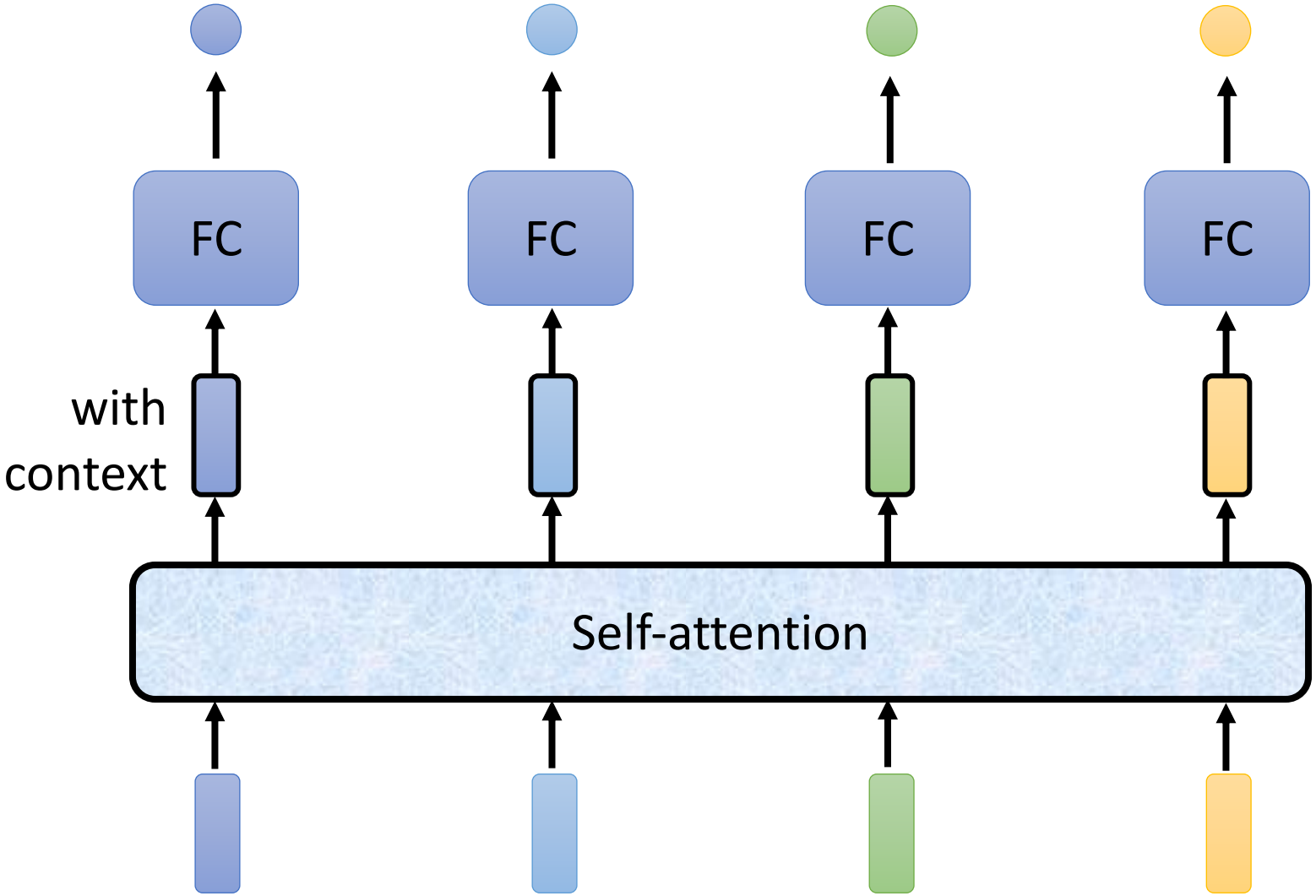
Sequence Labeling

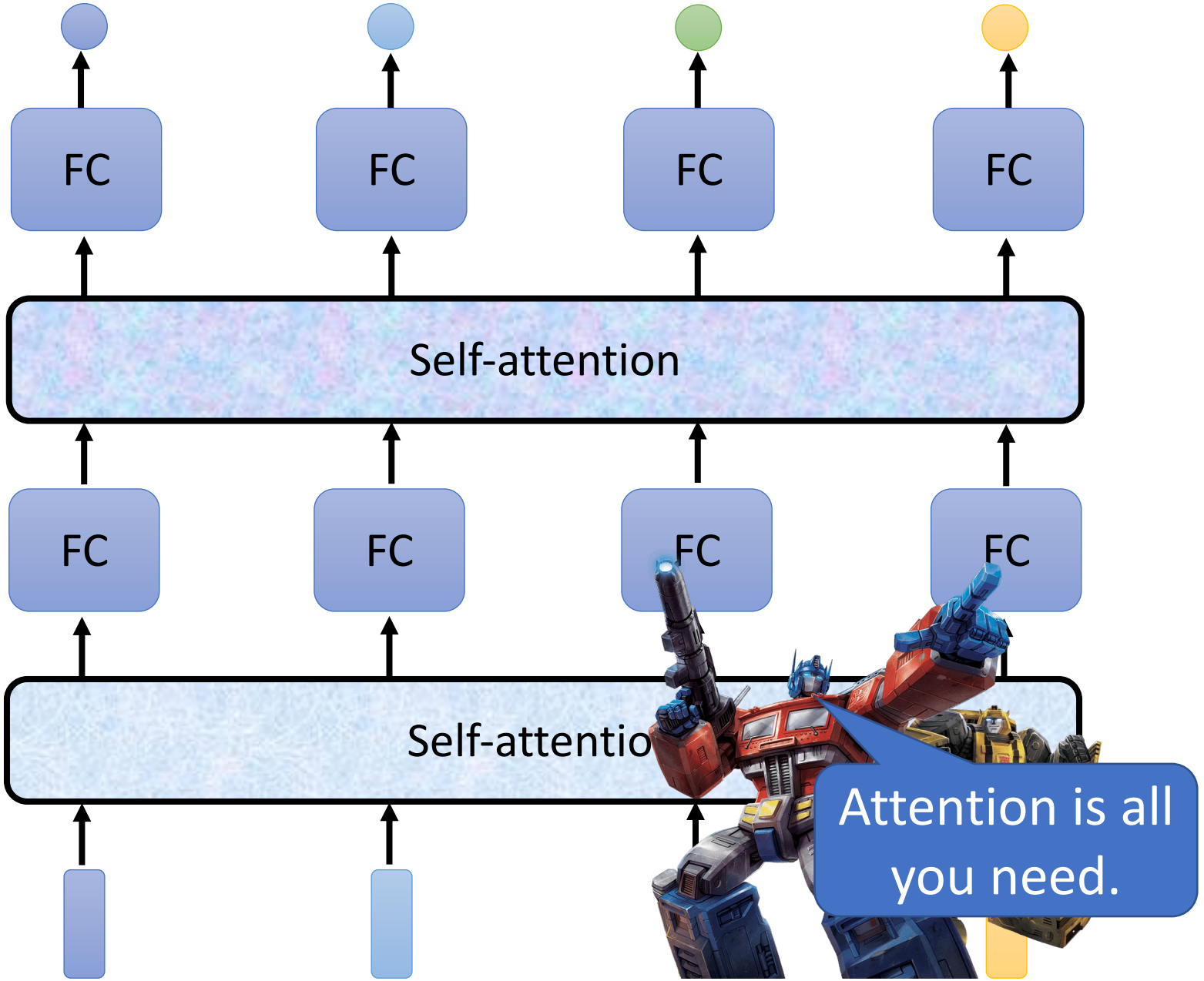






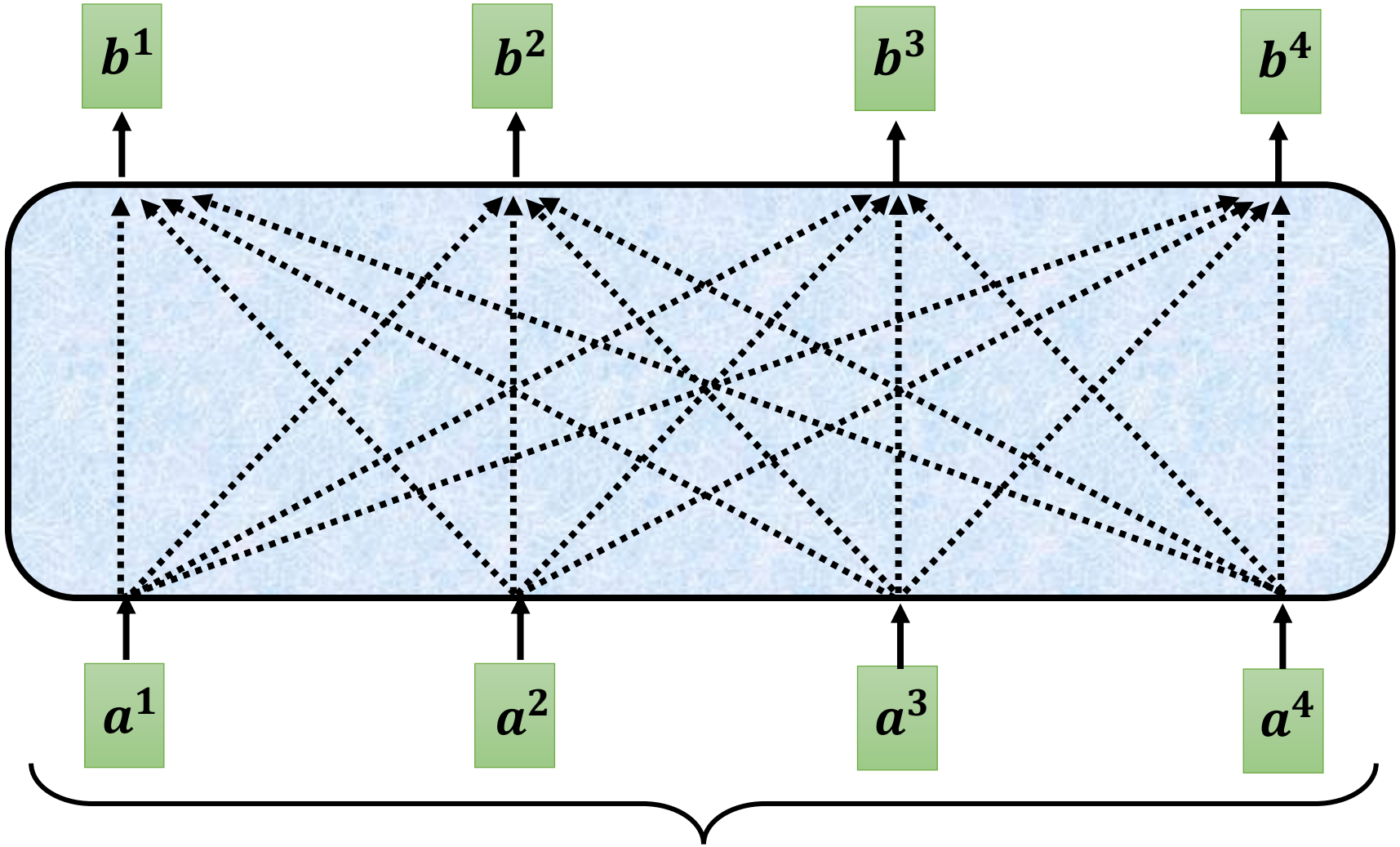
Self-attention





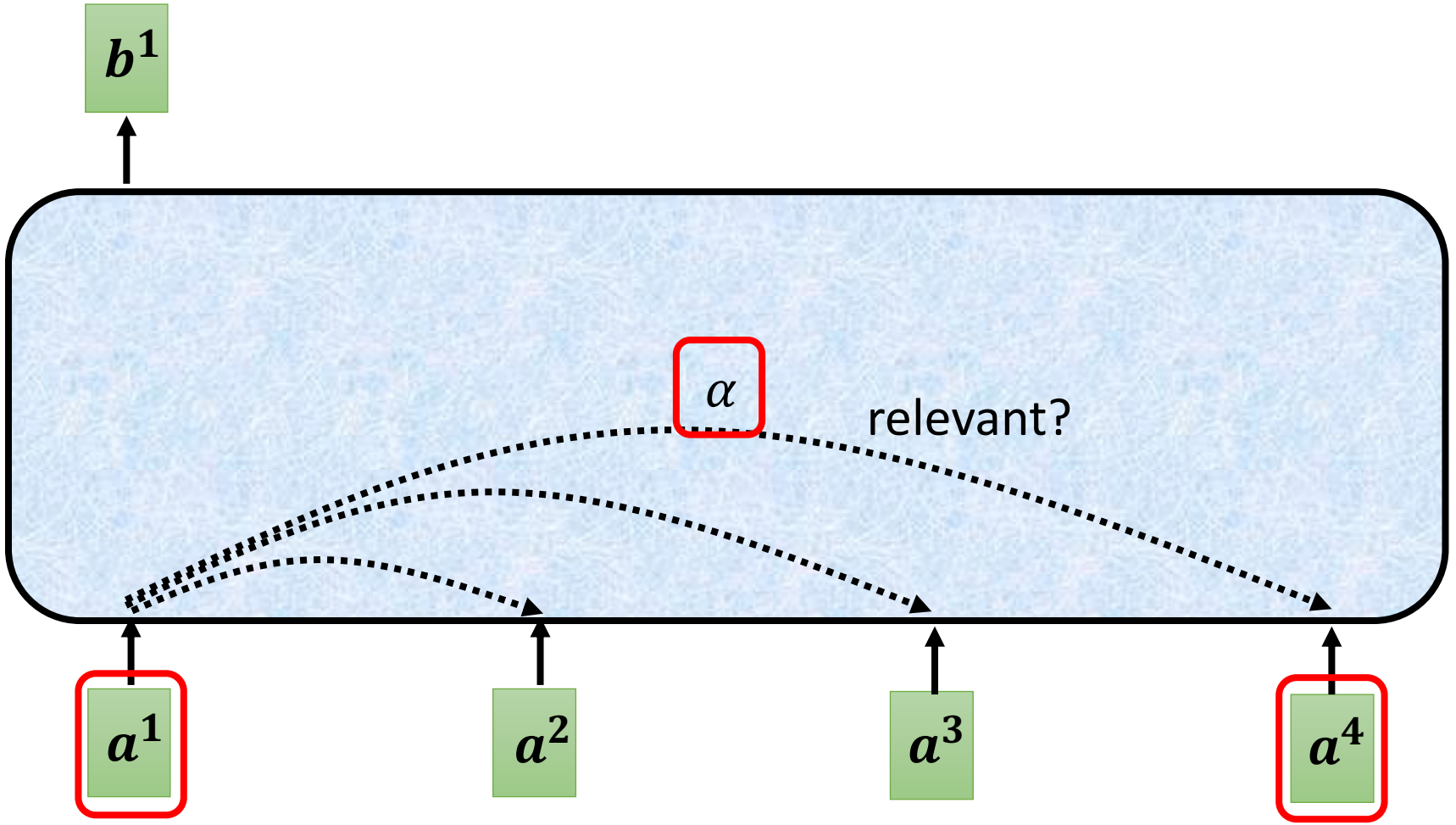
https://arxiv.org/abs/1706.03762

Self-attention



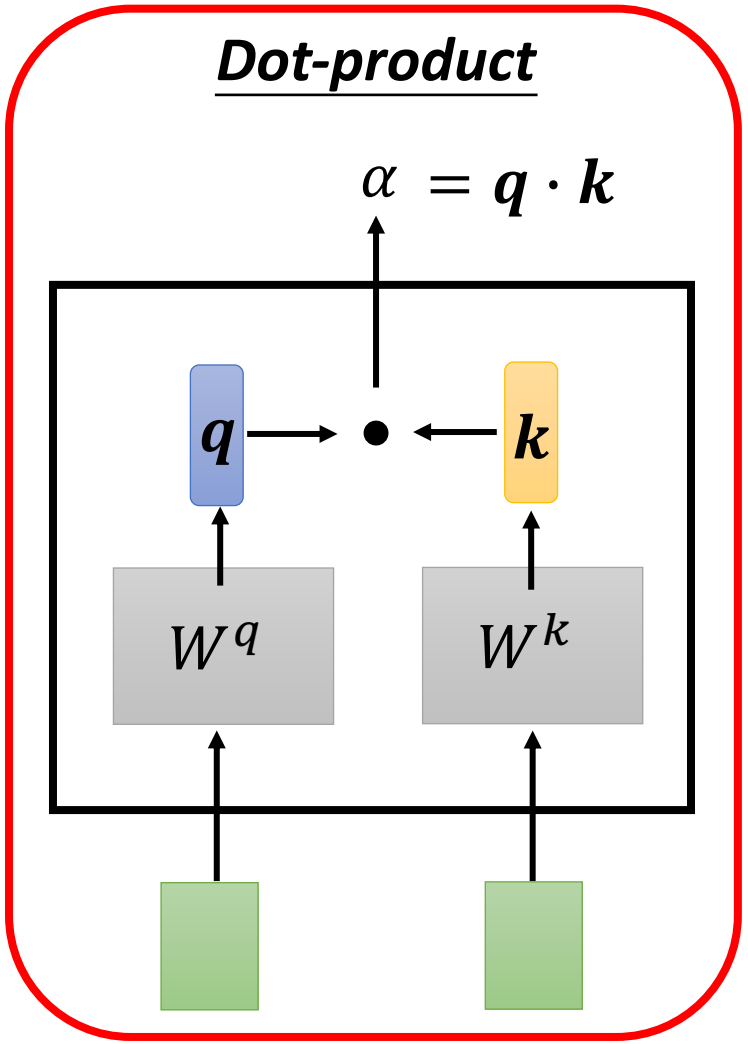
Can be either **input** or **a hidden layer**

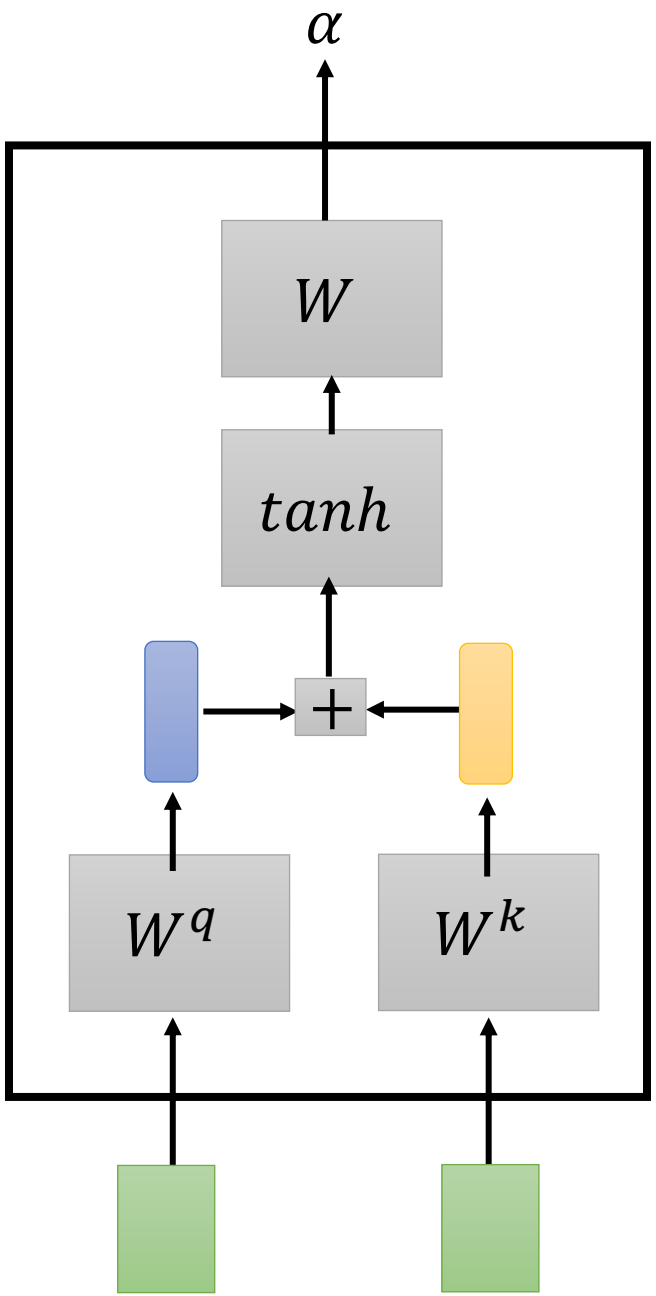
Self-attention



Find the relevant vectors in a sequence

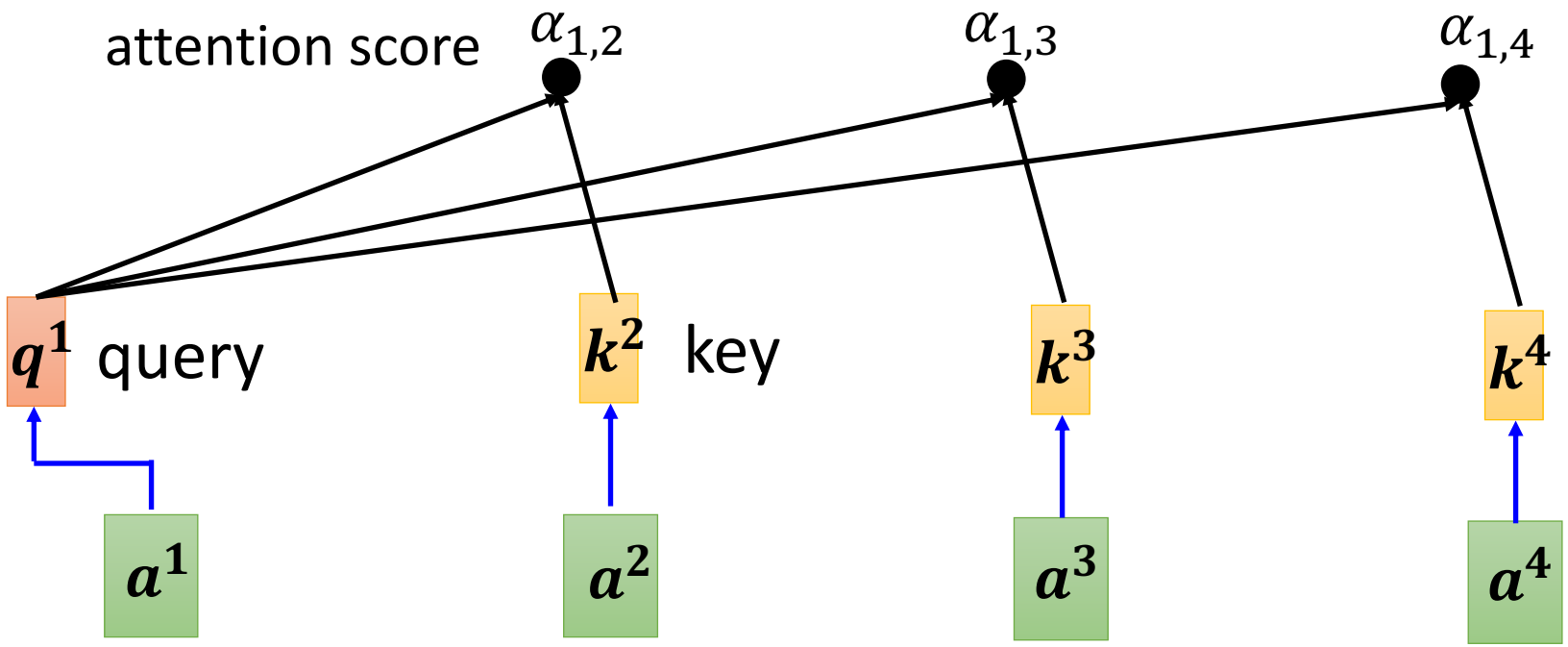
Self-attention





Self-attention

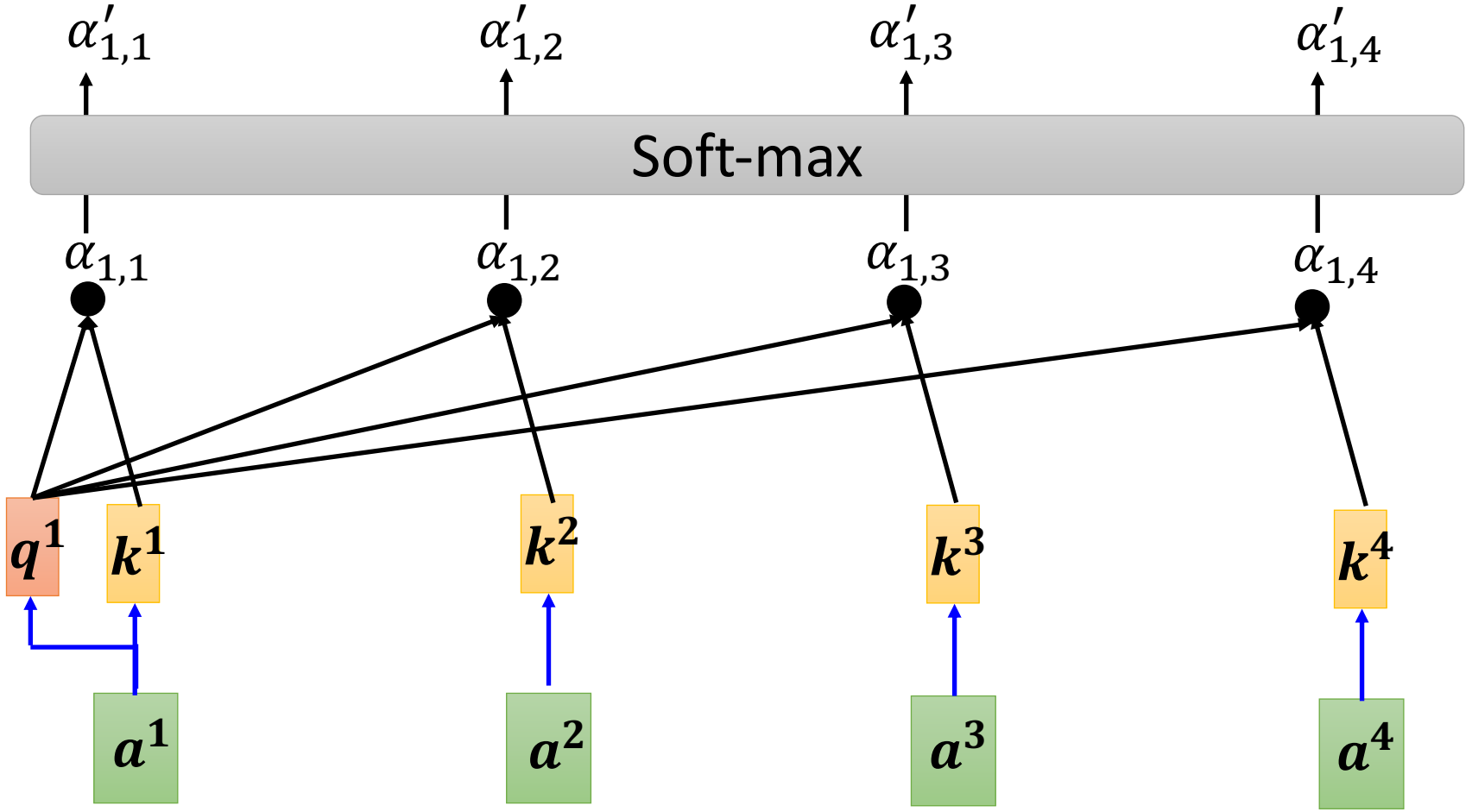
$\alpha_{1,2} = \boldsymbol{q^1} \cdot \boldsymbol{k^2}$ $\alpha_{1,3} = \boldsymbol{q^1} \cdot \boldsymbol{k^3}$ $\alpha_{1,4} = \boldsymbol{q^1} \cdot \boldsymbol{k^4}$



$\boldsymbol{q^1} = W^q \boldsymbol{a^1}$ $\boldsymbol{k^2} = W^k \boldsymbol{a^2}$ $\boldsymbol{k^3} = W^k \boldsymbol{a^3}$ $\boldsymbol{k^4} = W^k \boldsymbol{a^4}$

Self-attention

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



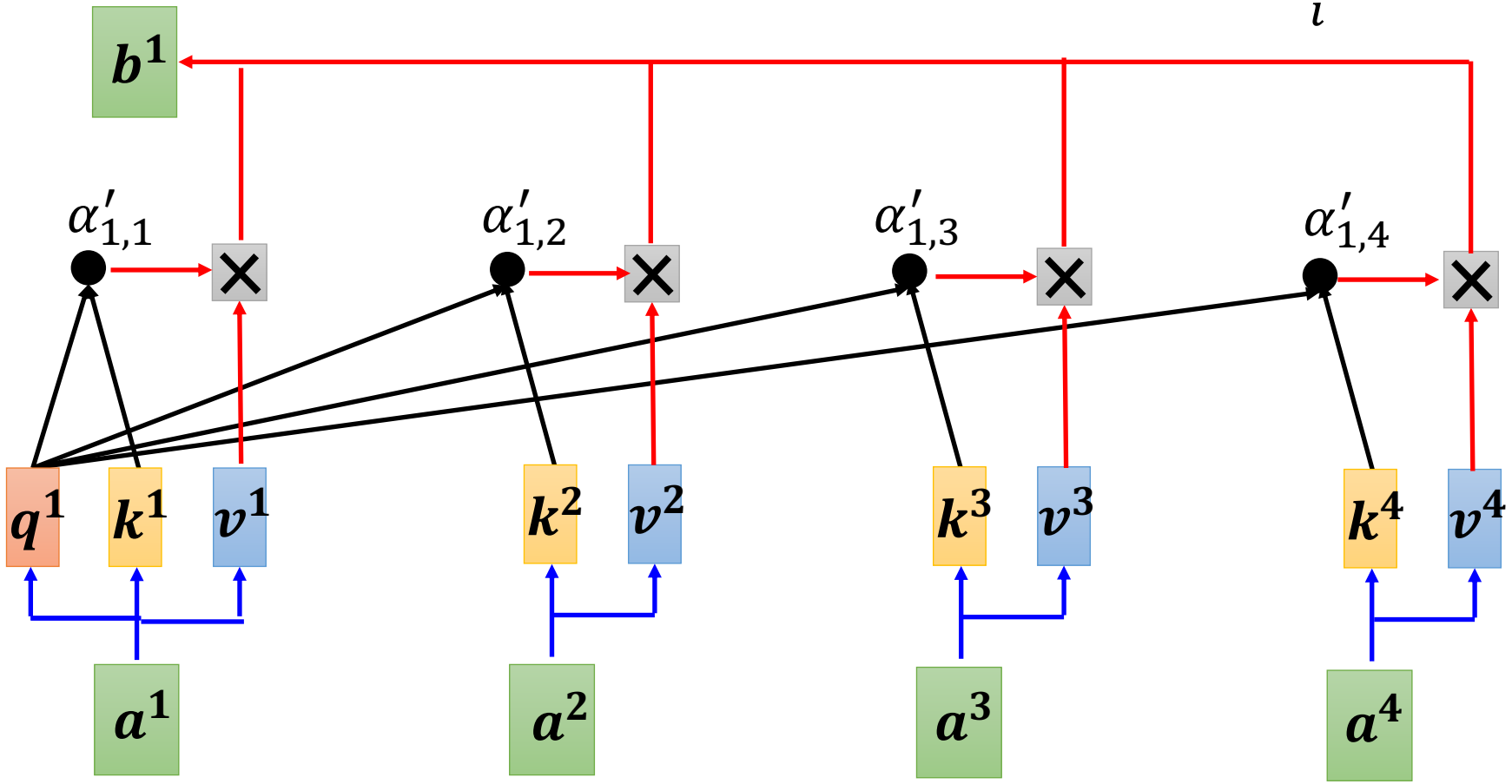
$q^1 = W^q a^1$ $k^2 = W^k a^2$ $k^3 = W^k a^3$ $k^4 = W^k a^4$

$k^1 = W^k a^1$

Self-attention

Extract information based on attention scores

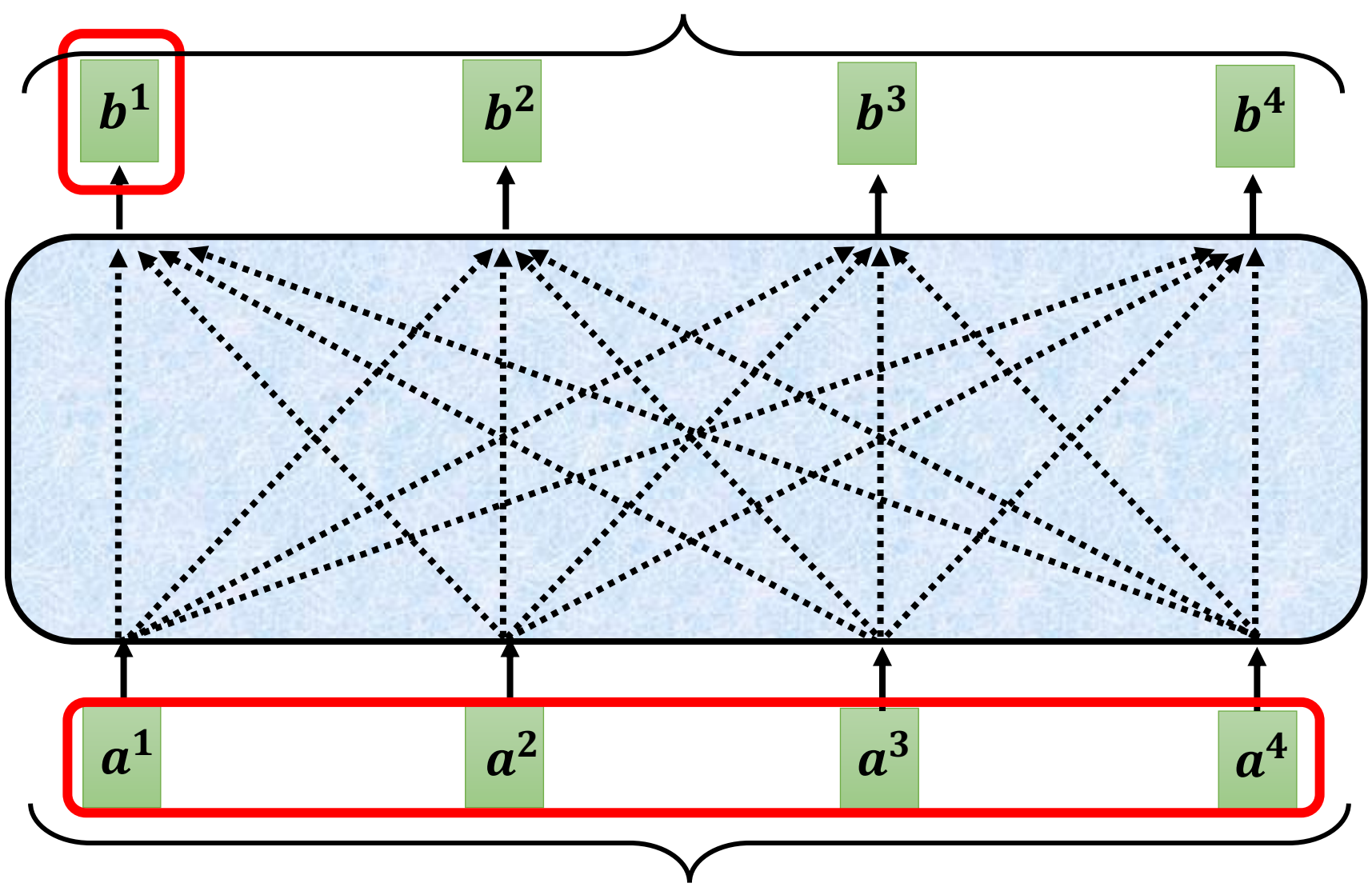
$$b^1 = \sum_i \alpha'_{1,i} v^i$$



还要学一个bias

$$v^1 = W^v a^1 \qquad v^2 = W^v a^2 \qquad v^3 = W^v a^3 \qquad v^4 = W^v a^4$$

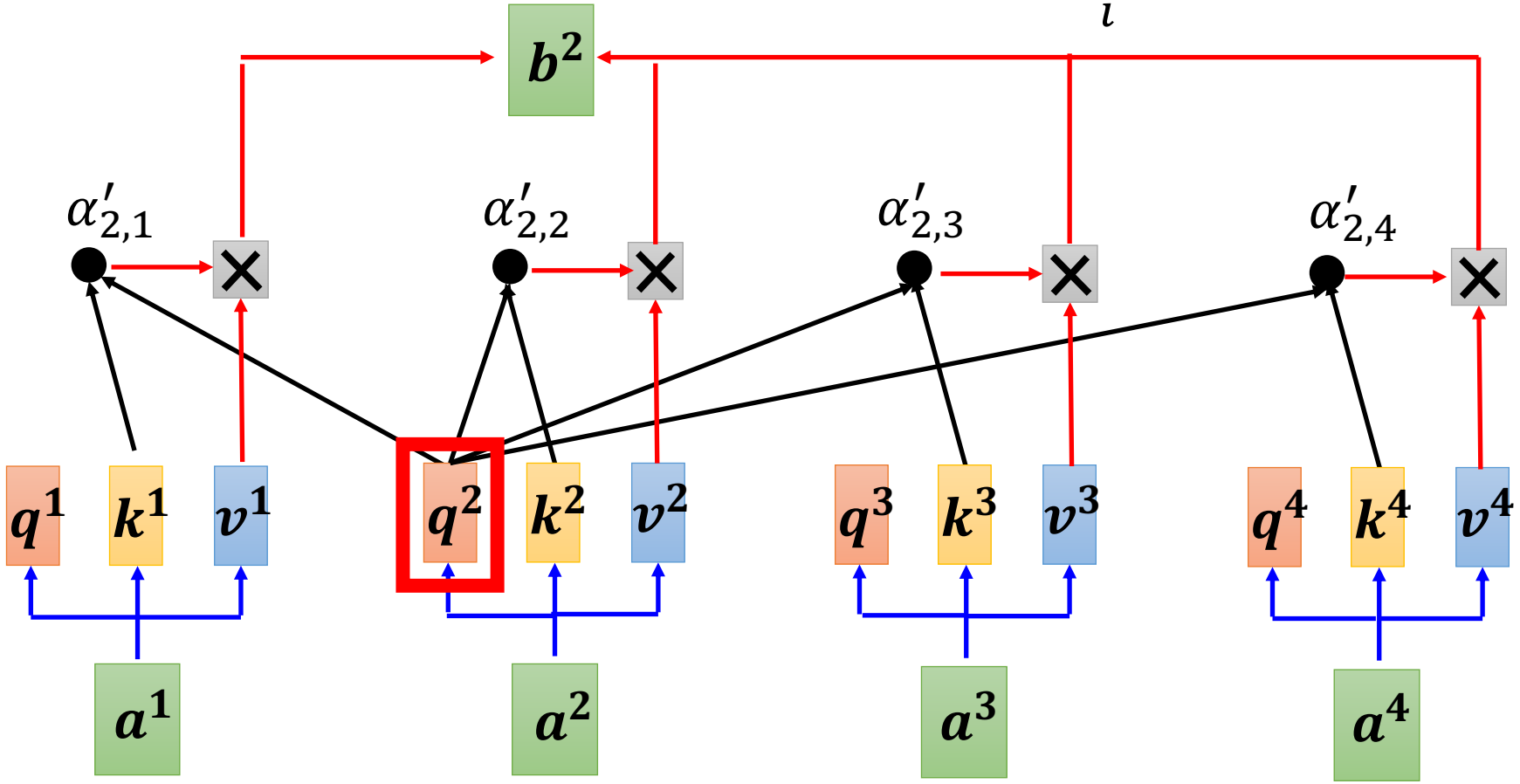
Self-attention



Can be either **input** or **a hidden layer**

Self-attention

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



Self-attention

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

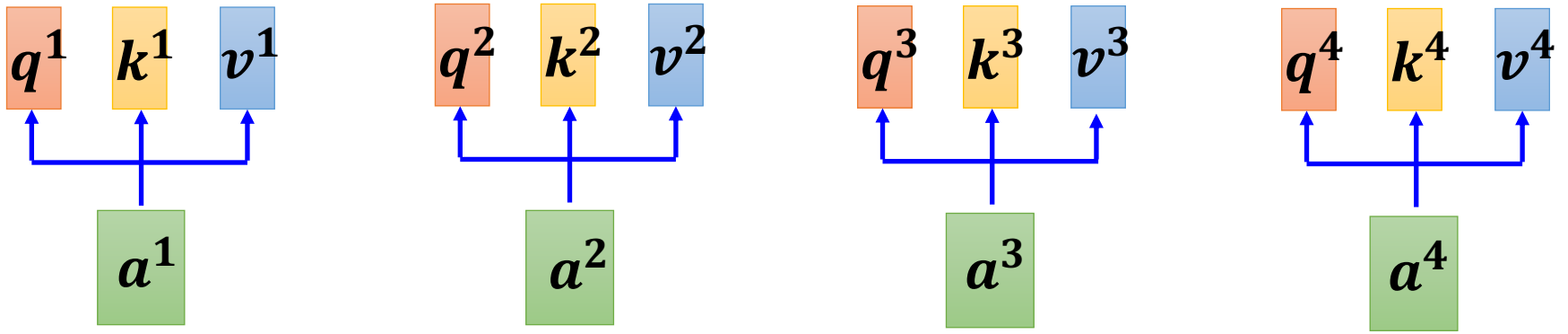
$$\underbrace{q^1 q^2 q^3 q^4}_{Q} = W^q \underbrace{a^1 a^2 a^3 a^4}_{I}$$

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

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

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

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

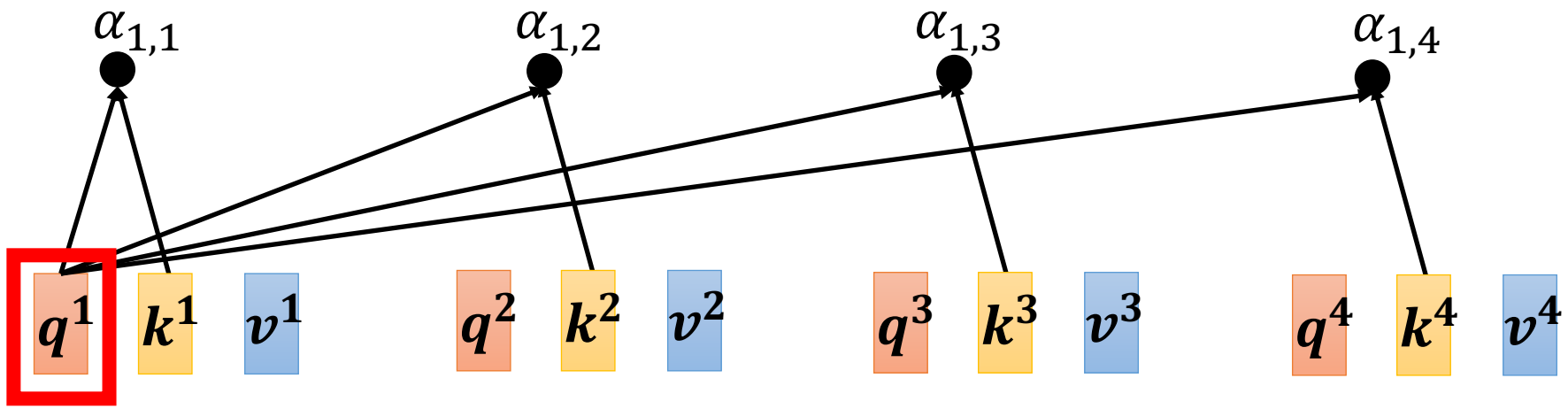


Self-attention

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

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

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

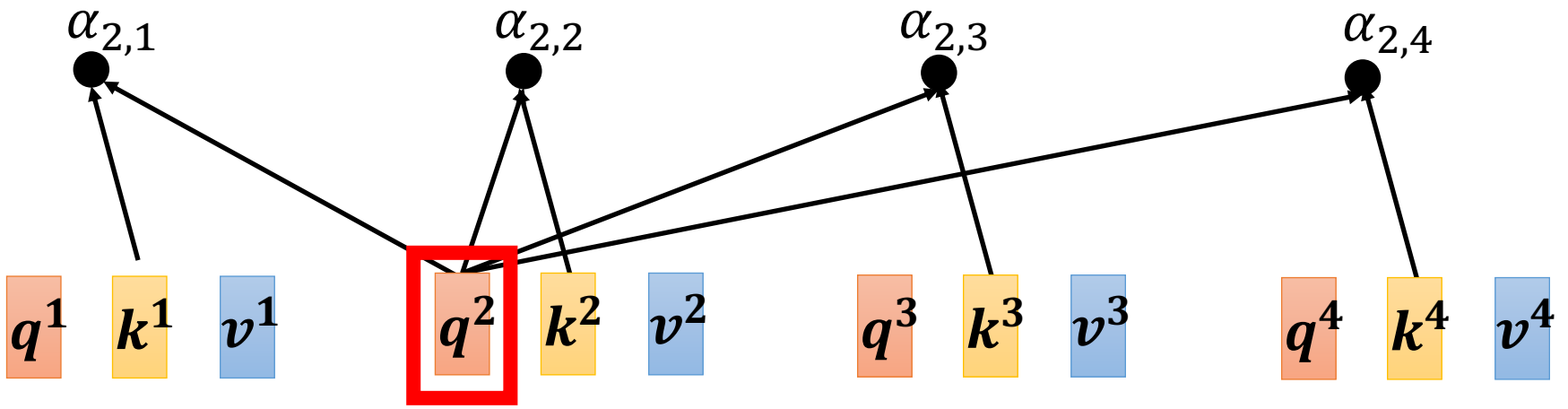


Self-attention

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

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

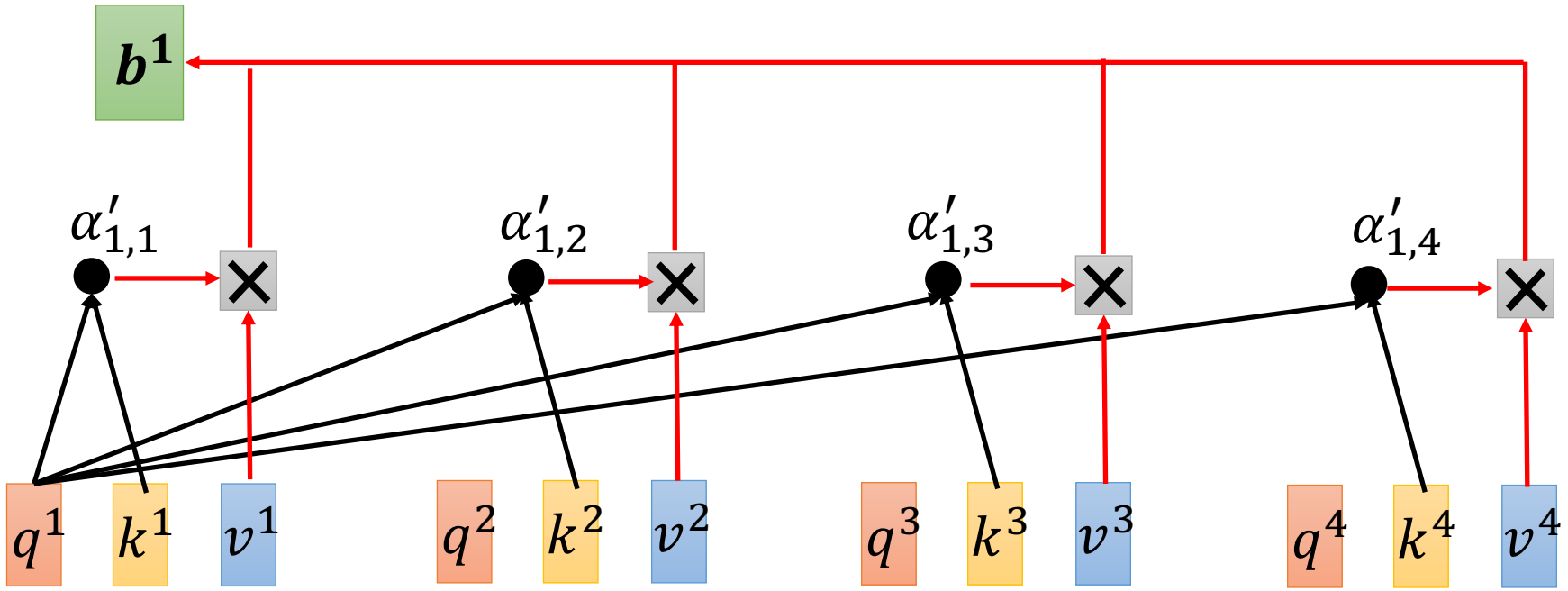
$$\begin{bmatrix}\alpha_{1,1}\\ \alpha_{1,2}\\ \alpha_{1,3}\\ \alpha_{1,4}\end{bmatrix} = \begin{bmatrix}\boldsymbol{k^1}\\ \boldsymbol{k^2}\\ \boldsymbol{k^3}\\ \boldsymbol{k^4}\end{bmatrix} \boldsymbol{q^1}$$



$$\begin{bmatrix}\alpha'_{1,1} & \alpha'_{2,1} & \alpha'_{3,1} & \alpha'_{4,1}\\ \alpha'_{1,2} & \alpha'_{2,2} & \alpha'_{3,2} & \alpha'_{4,2}\\ \alpha'_{1,3} & \alpha'_{2,3} & \alpha'_{3,3} & \alpha'_{4,3}\\ \alpha'_{1,4} & \alpha'_{2,4} & \alpha'_{3,4} & \alpha'_{4,4}\end{bmatrix} = A' \qquad \text{softmax} \qquad \begin{bmatrix}\alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1}\\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2}\\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3}\\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4}\end{bmatrix} = A$$

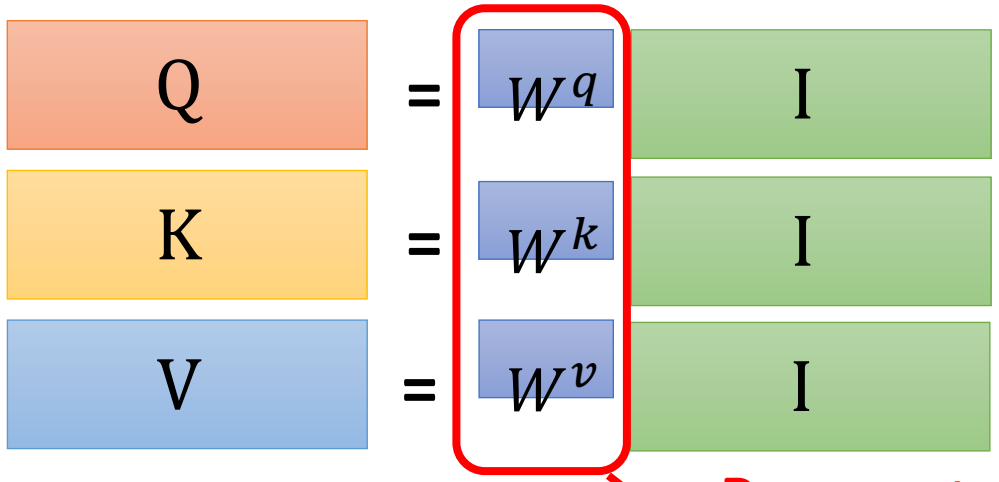
$$\begin{bmatrix}\boldsymbol{k^1}\\ \boldsymbol{k^2}\\ \boldsymbol{k^3}\\ \boldsymbol{k^4}\end{bmatrix} = K^T \qquad \begin{bmatrix}\boldsymbol{q^1} & \boldsymbol{q^2} & \boldsymbol{q^3} & \boldsymbol{q^4}\end{bmatrix} = Q$$

Self-attention

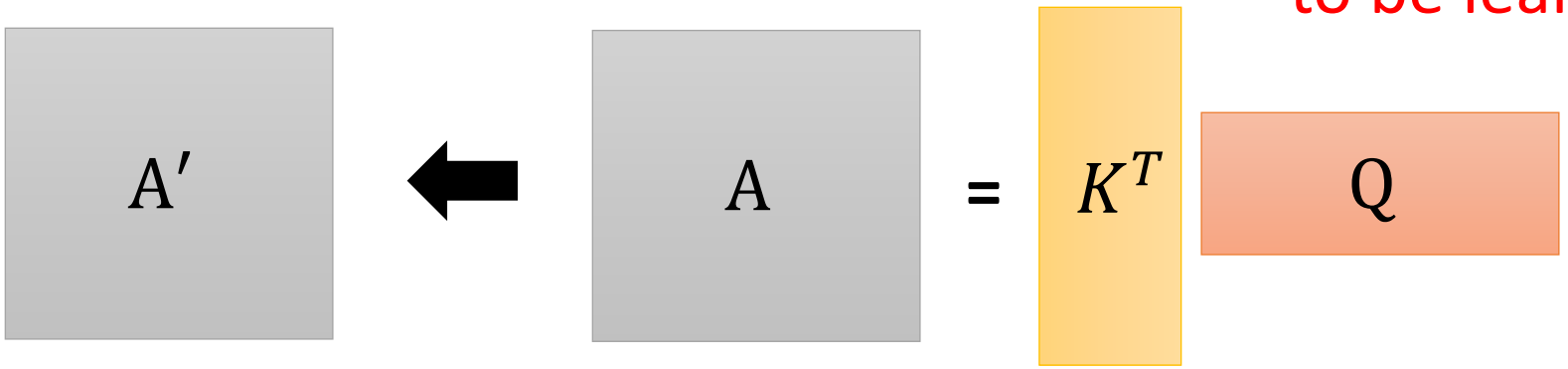


$$\underbrace{b^1\,b^2\,b^3\,b^4}_{O} = \underbrace{v^1\,v^2\,v^3\,v^4}_{V} \begin{bmatrix} \alpha'_{1,1} & \alpha'_{2,1} & \alpha'_{3,1} & \alpha'_{4,1} \\ \alpha'_{1,2} & \alpha'_{2,2} & \alpha'_{3,2} & \alpha'_{4,2} \\ \alpha'_{1,3} & \alpha'_{2,3} & \alpha'_{3,3} & \alpha'_{4,3} \\ \alpha'_{1,4} & \alpha'_{2,4} & \alpha'_{3,4} & \alpha'_{4,4} \end{bmatrix}_{A'}$$

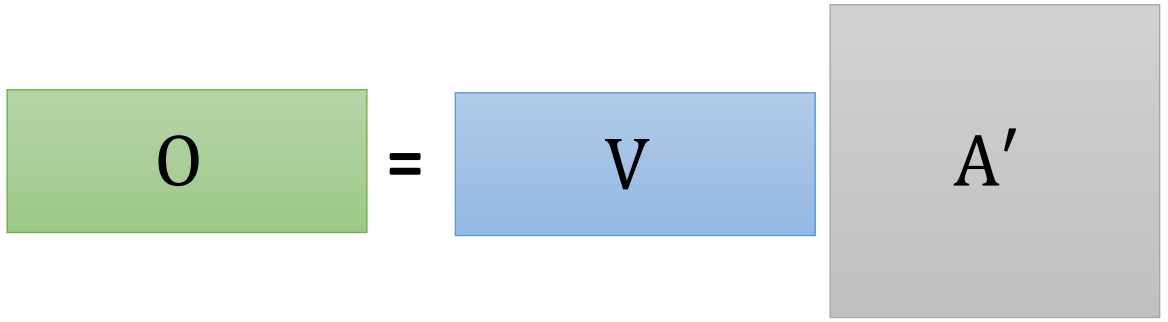
Self-attention



Parameters to be learned



Attention Matrix

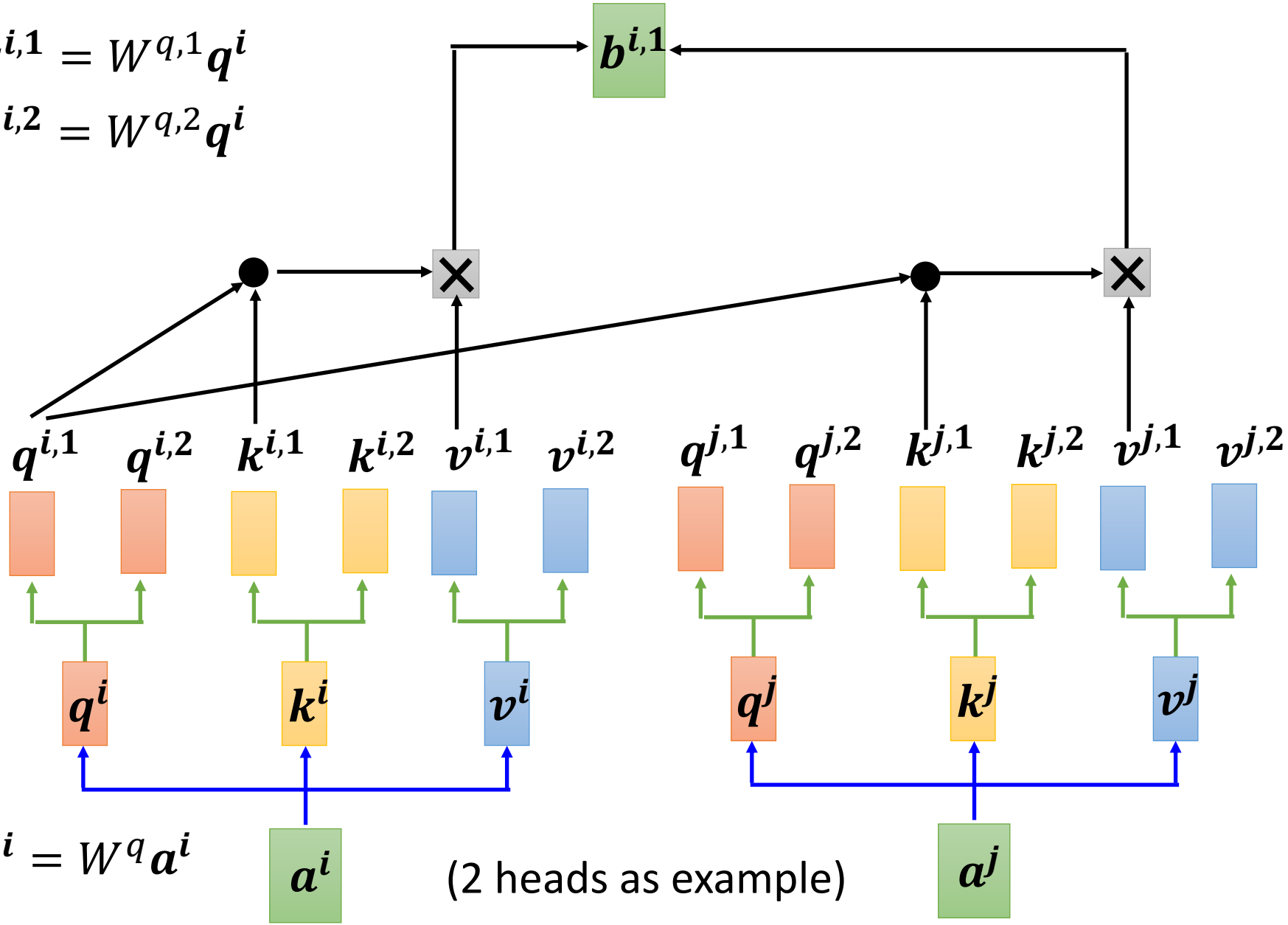


Multi-head Self-attention

Different types of relevance

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

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



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

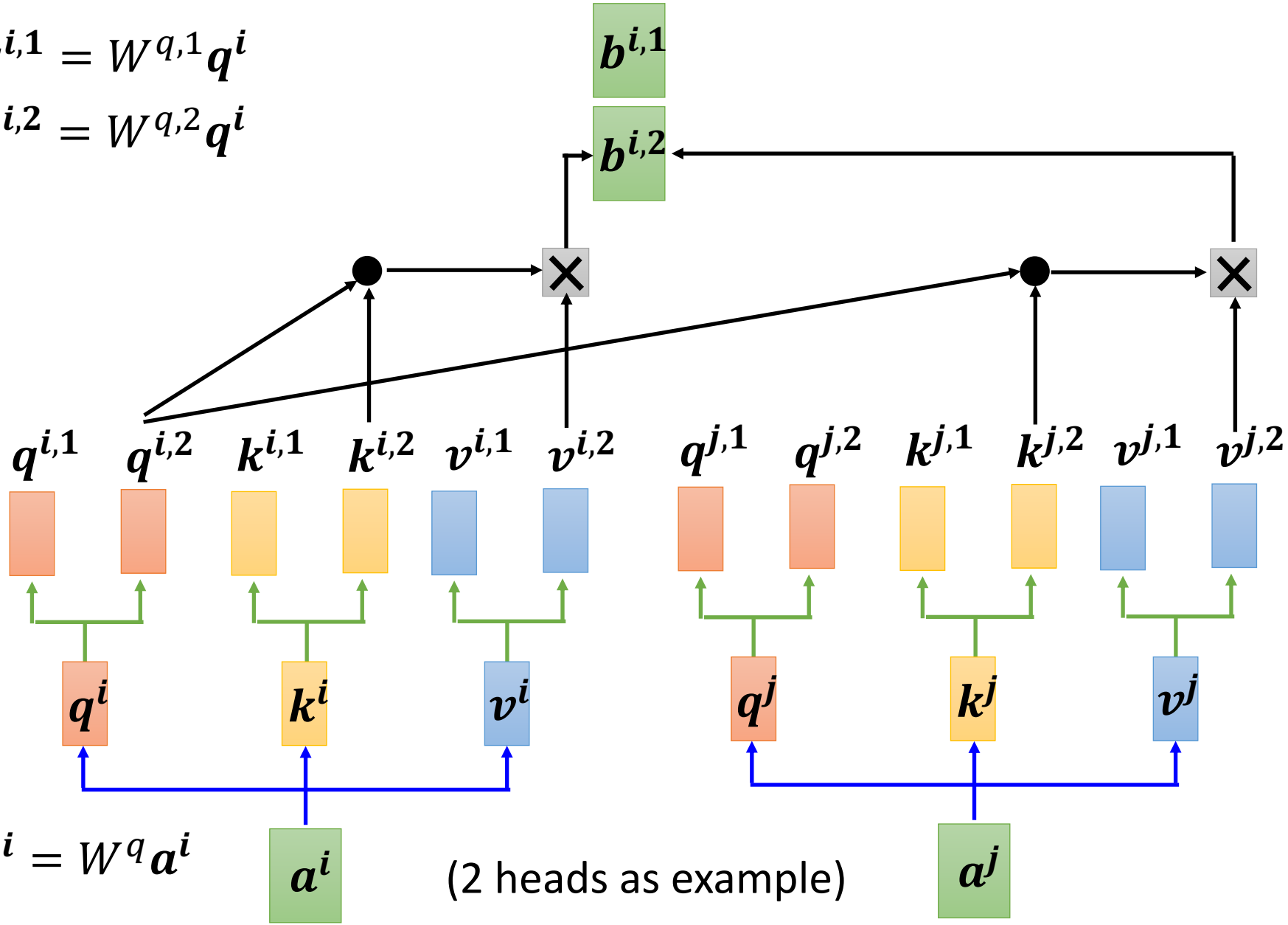
(2 heads as example)

Multi-head Self-attention

Different types of relevance

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

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

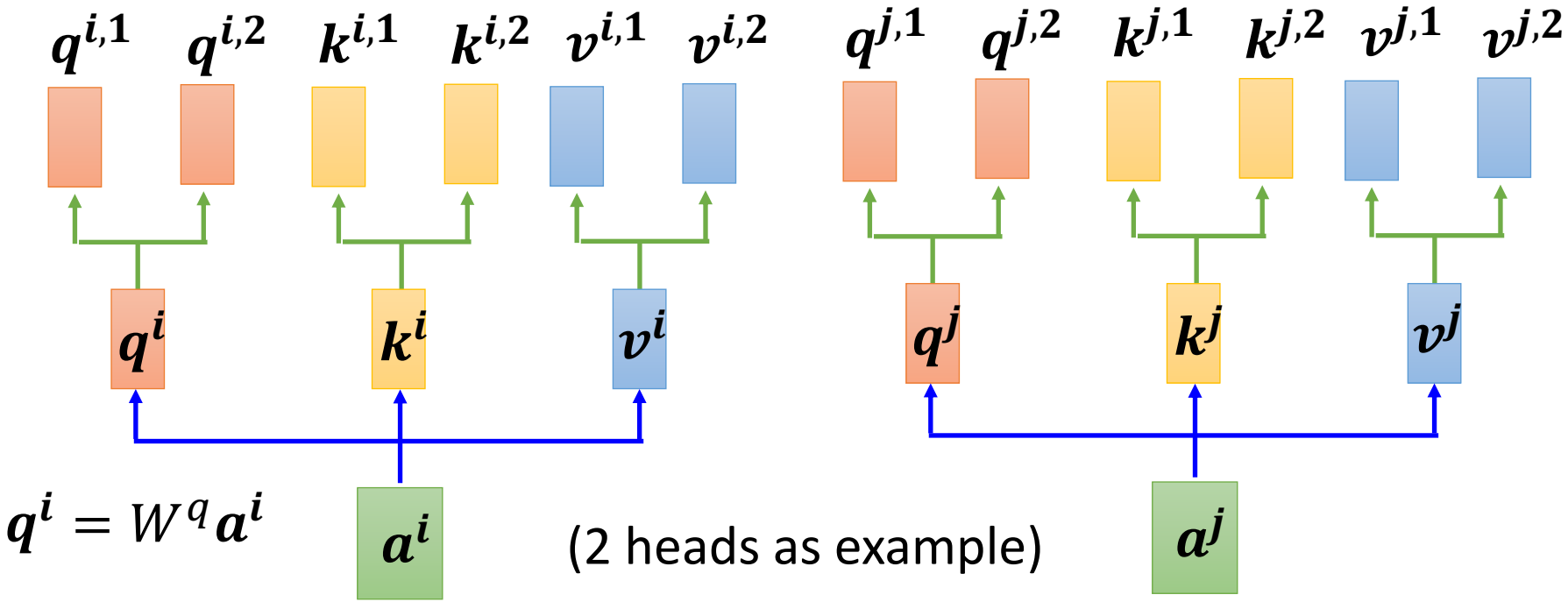


$$q^i = W^q a^i$$

(2 heads as example)

Multi-head Self-attention Different types of relevance

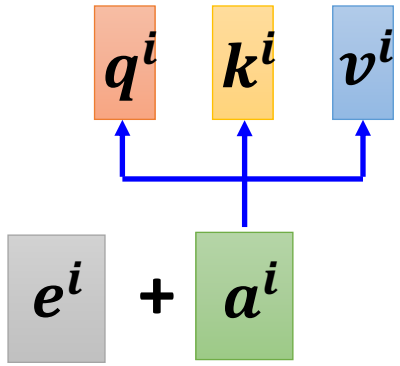
$$b^i = W^O \begin{bmatrix} b^{i,1} \\ b^{i,2} \end{bmatrix}$$



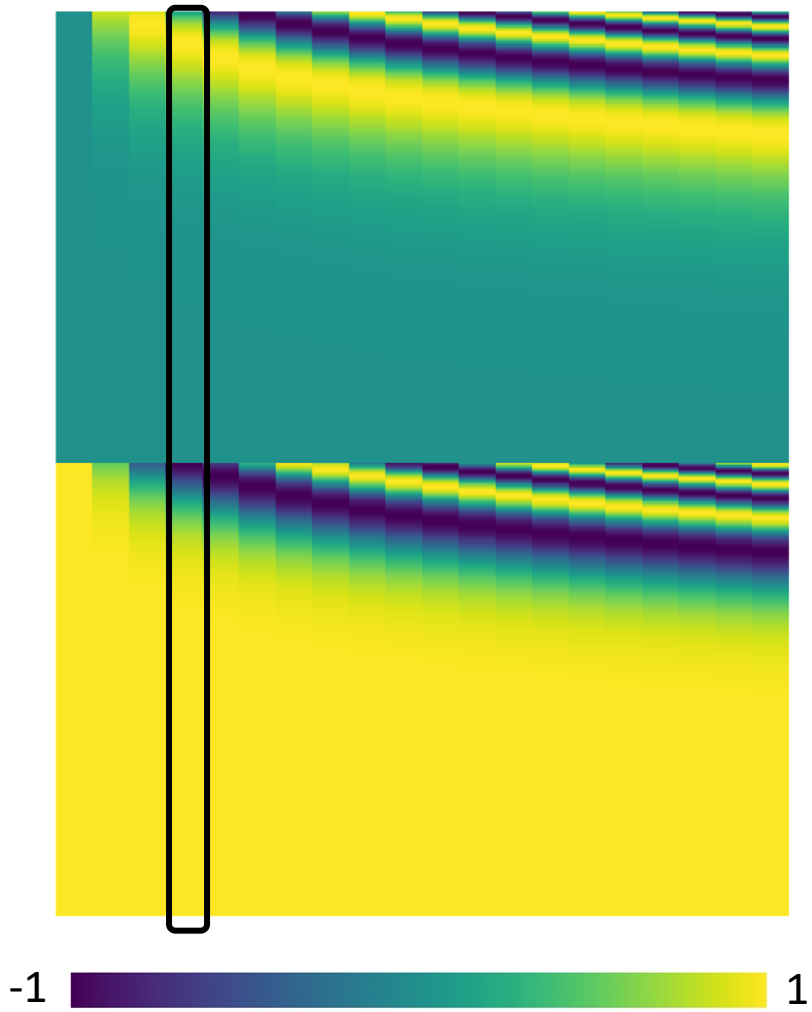
(2 heads as example)

Positional Encoding

- No position information in self-attention.
- Each position has a unique positional vector e^i
- **hand-crafted**
- **learned from data**



Each column represents a positional vector e^i



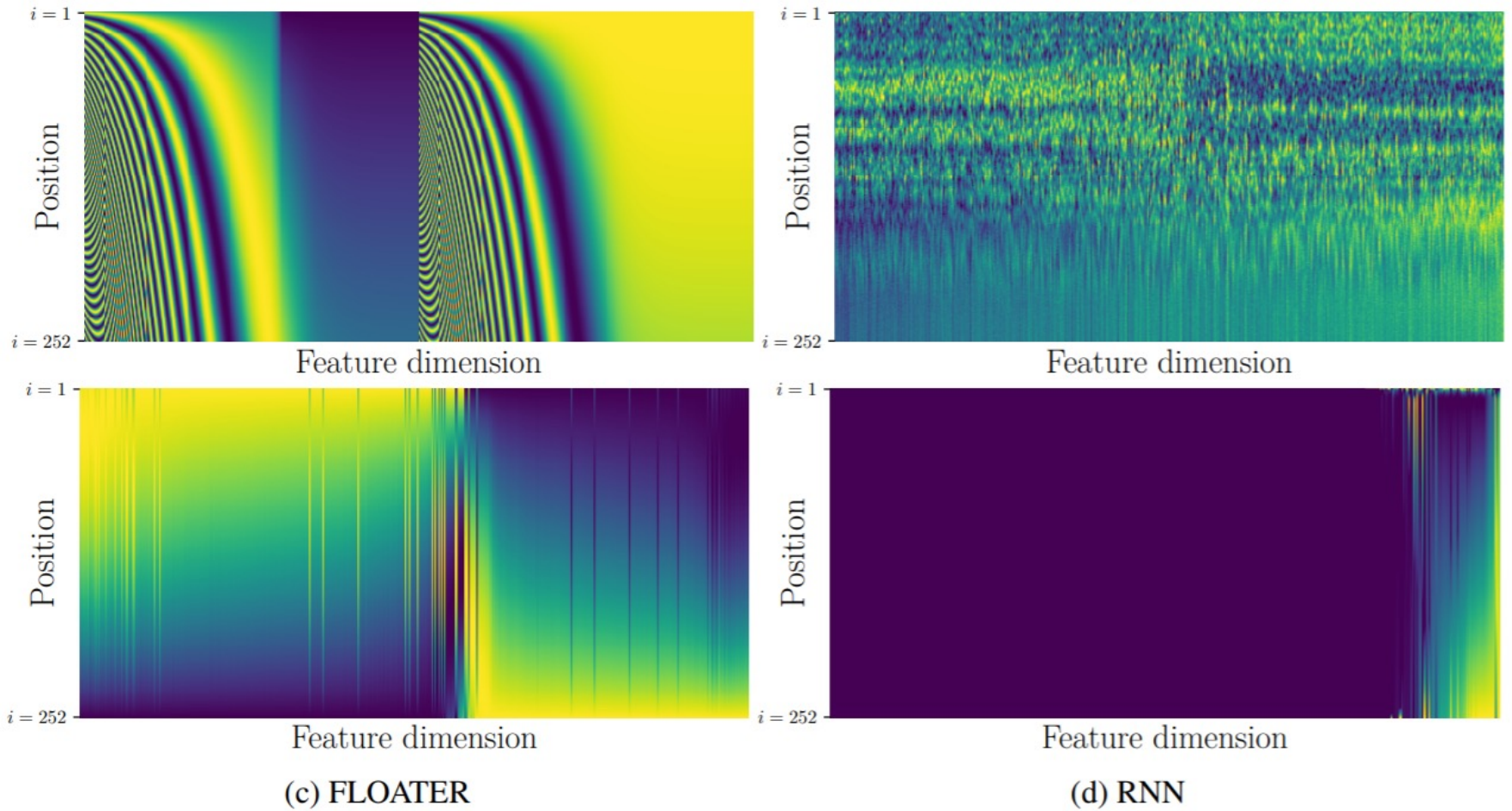
https://arxiv.org/abs/2003.09229

Table 1. Comparing position representation methods

Methods	Inductive	Data-Driven	Parameter Efficient
Sinusoidal (Vaswani et al., 2017)	✓	✗	✓
Embedding (Devlin et al., 2018)	✗	✓	✗
Relative (Shaw et al., 2018)	✗	✓	✓
This paper	✓	✓	✓

(a) Sinusoidal

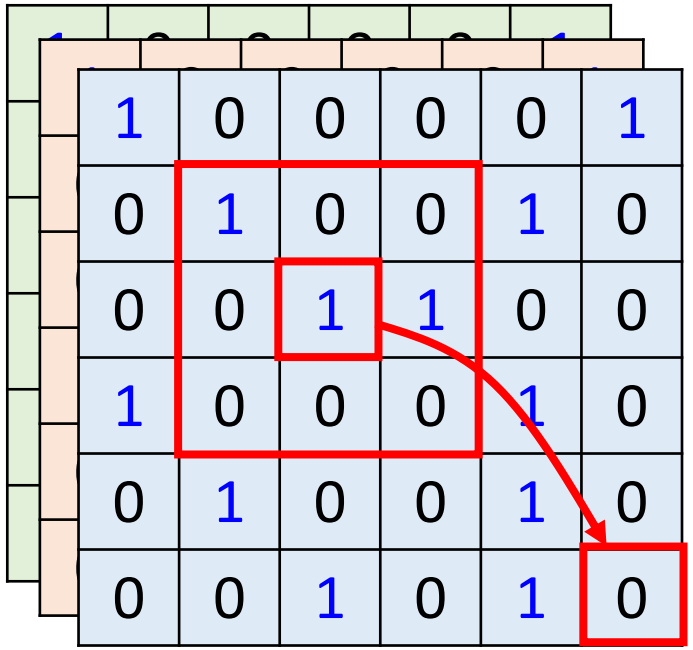
(b) Position embedding



(c) FLOATER

(d) RNN

Self-attention v.s. CNN



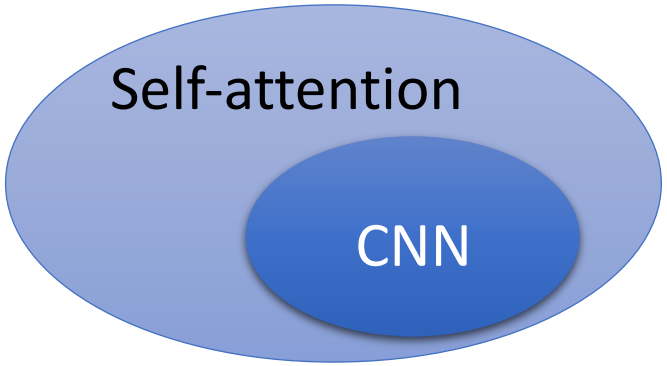
CNN: self-attention that can only attends in a receptive field

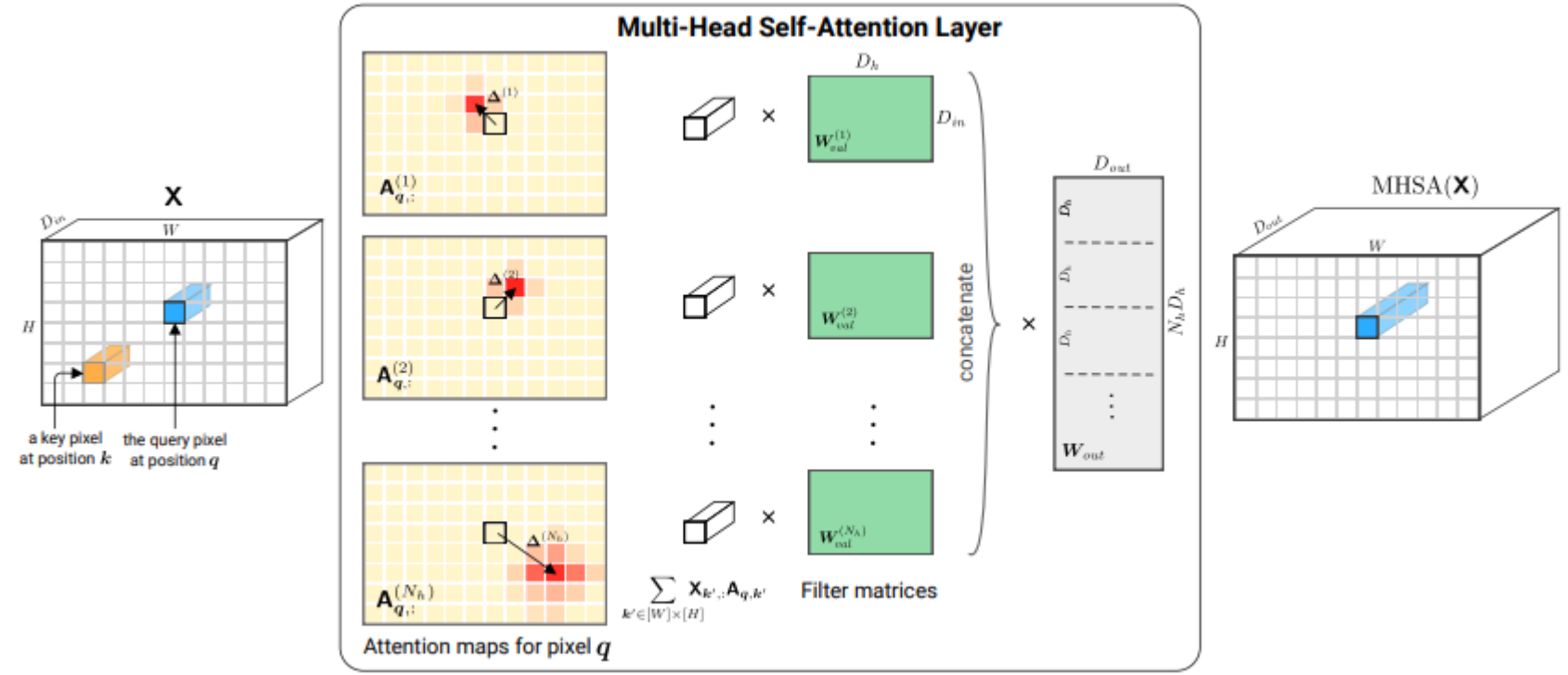
- CNN is simplified self-attention.

Self-attention: CNN with learnable receptive field

- Self-attention is the complex version of CNN.

Self-attention v.s. CNN





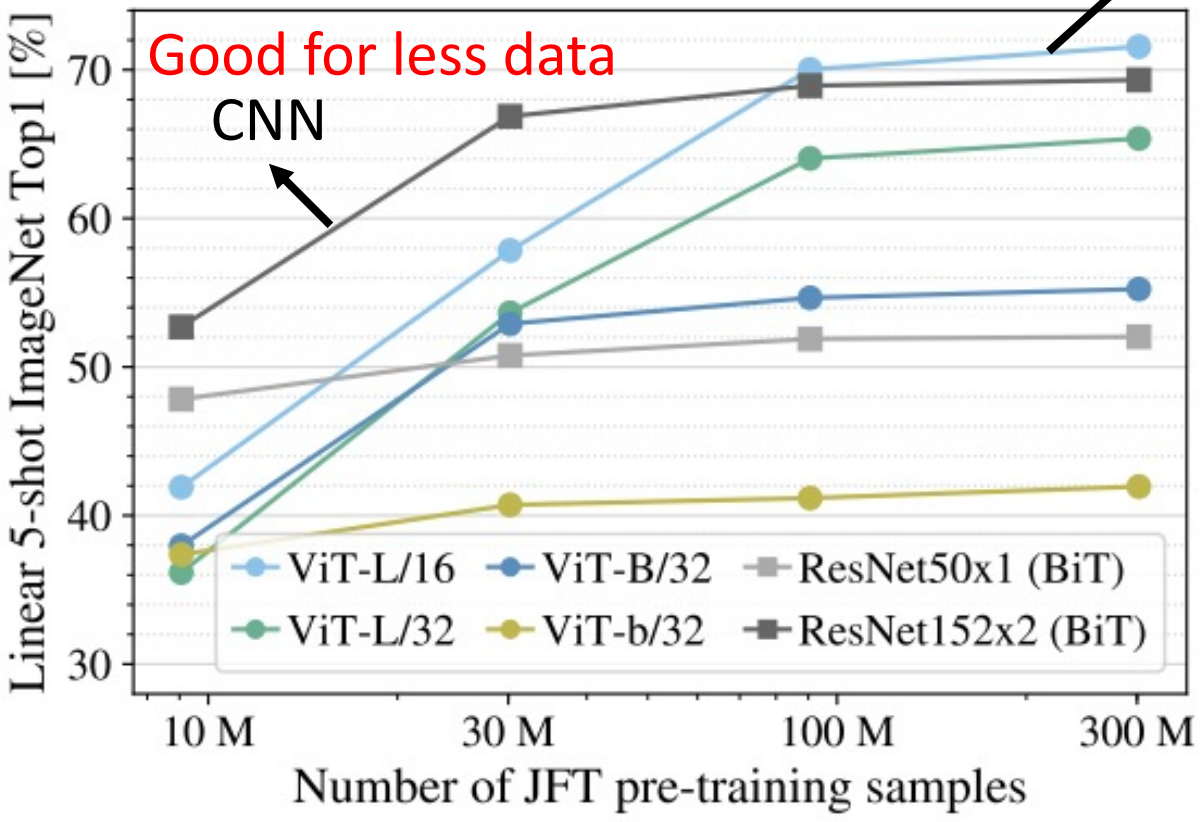
On the Relationship between Self-Attention and Convolutional Layers

https://arxiv.org/abs/1911.03584

Self-attention v.s. CNN

Good for more data

Self-attention

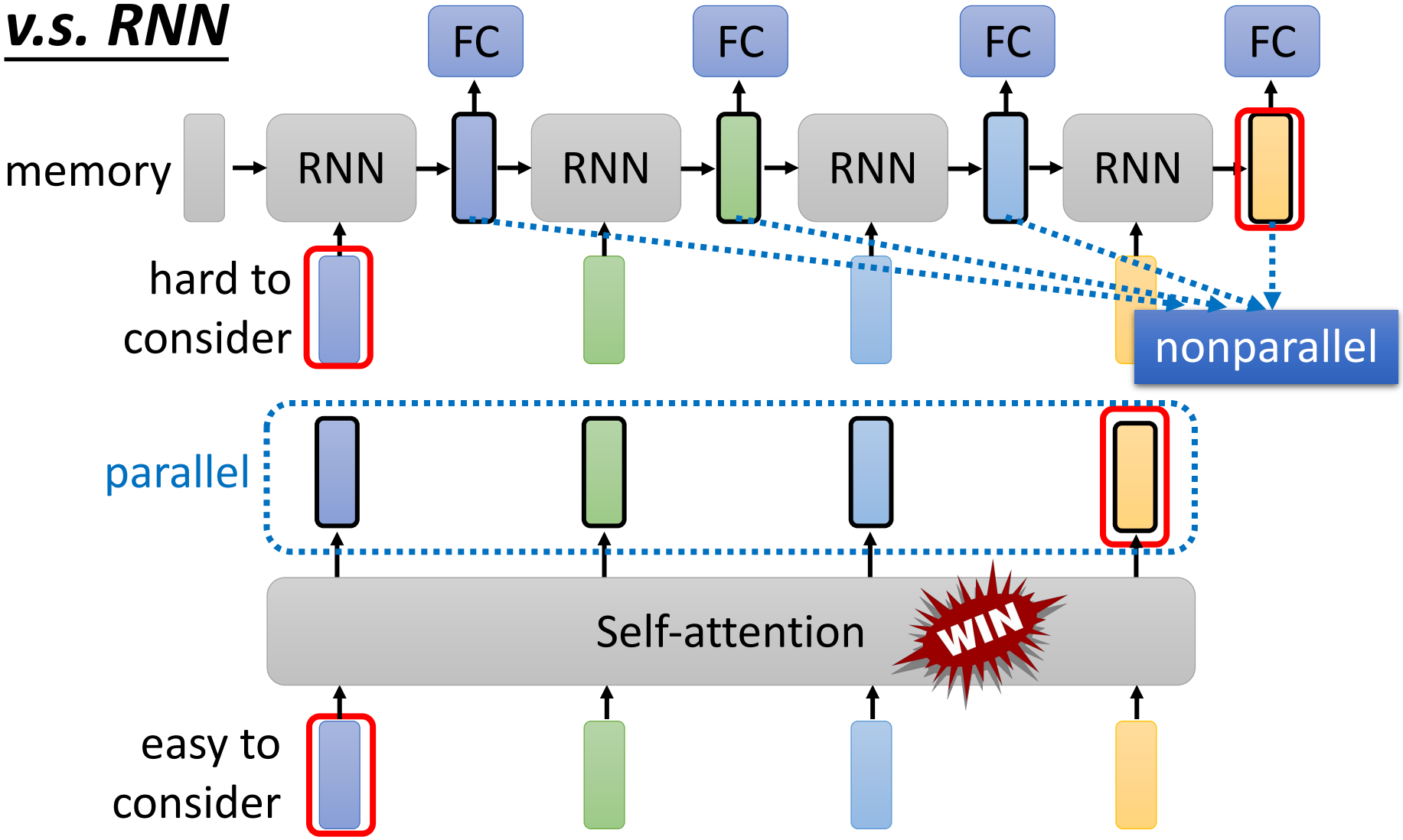


An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

https://arxiv.org/pdf/2010.11929.pdf

Self-attention v.s. RNN

Recurrent Neural Network (RNN)



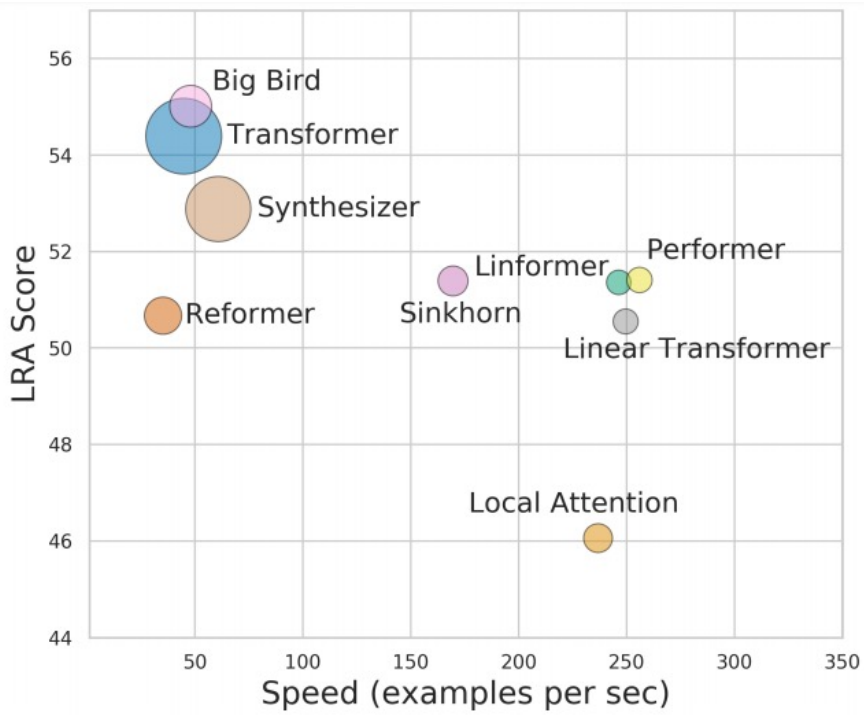
Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

https://arxiv.org/abs/2006.16236

To Learn More …

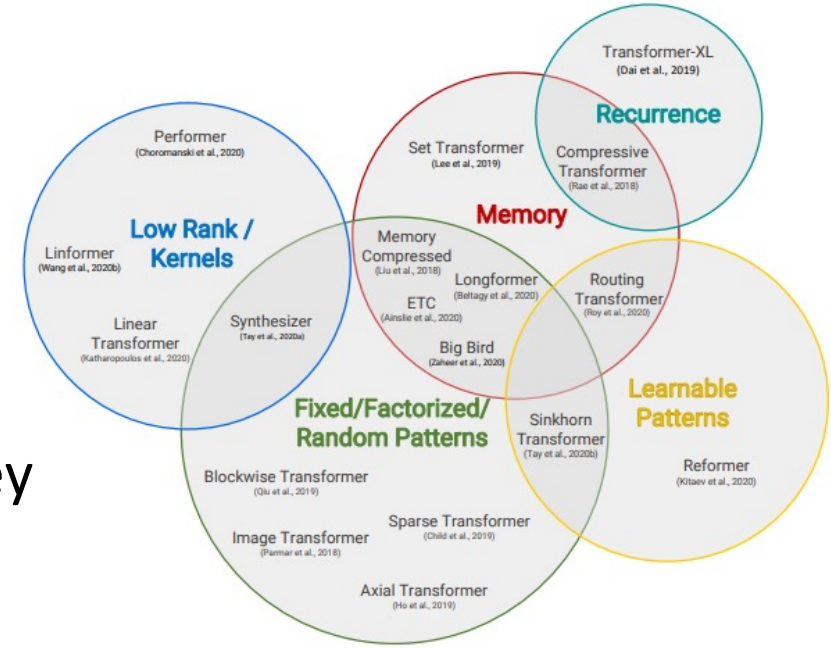
Long Range Arena: A Benchmark for Efficient Transformers

https://arxiv.org/abs/2011.04006

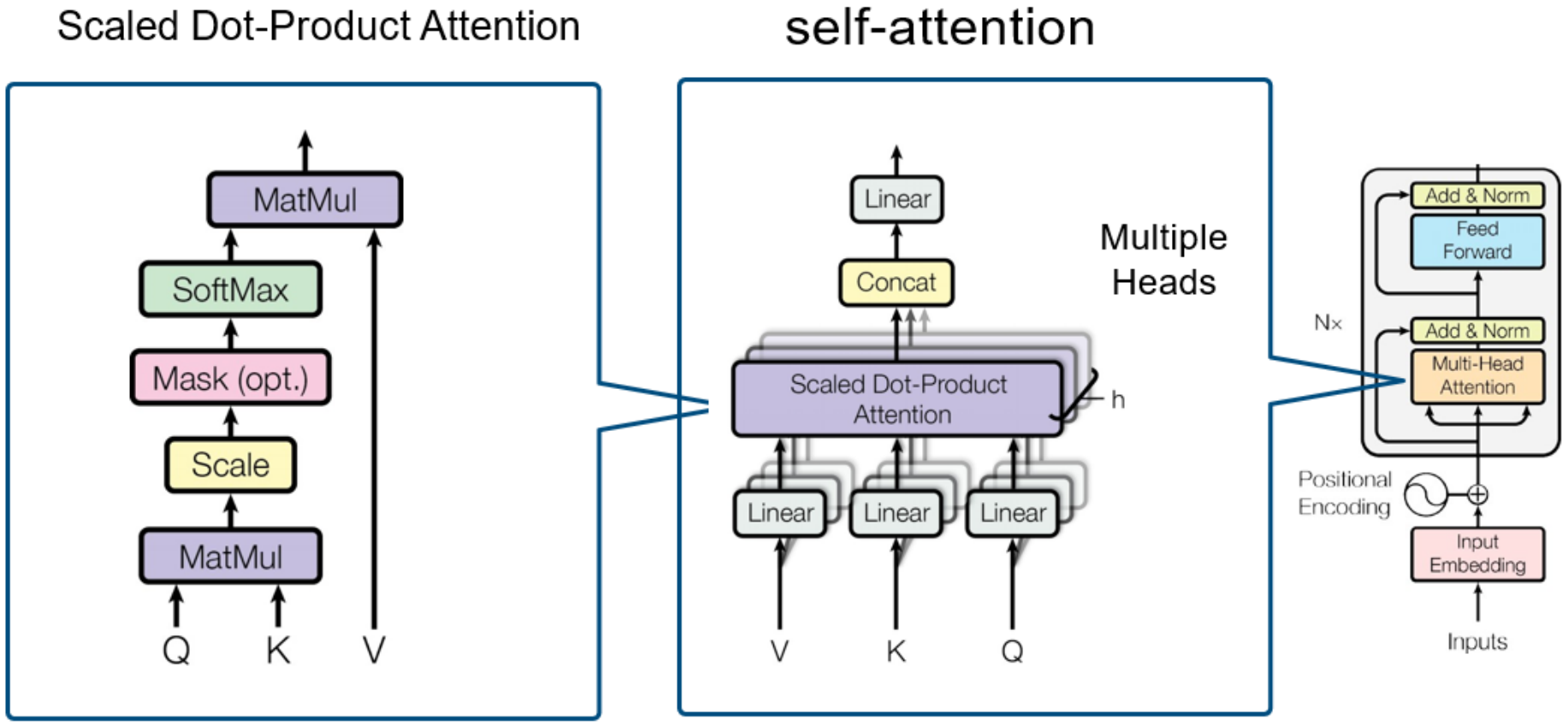


Efficient Transformers: A Survey

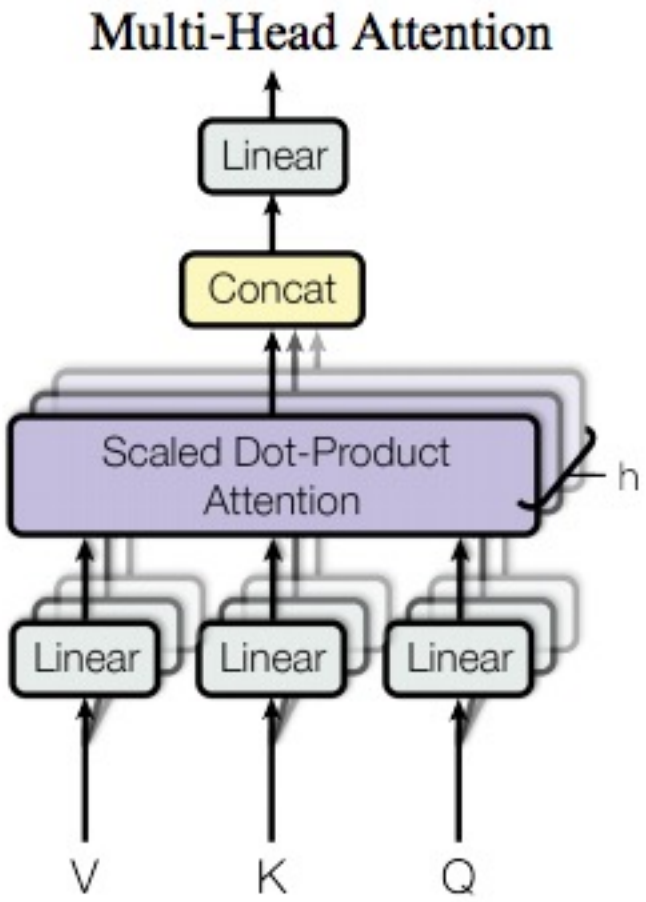
https://arxiv.org/abs/2009.06732



Multi-head self-attention



3. Text Processing — Transformer: Full Architecture



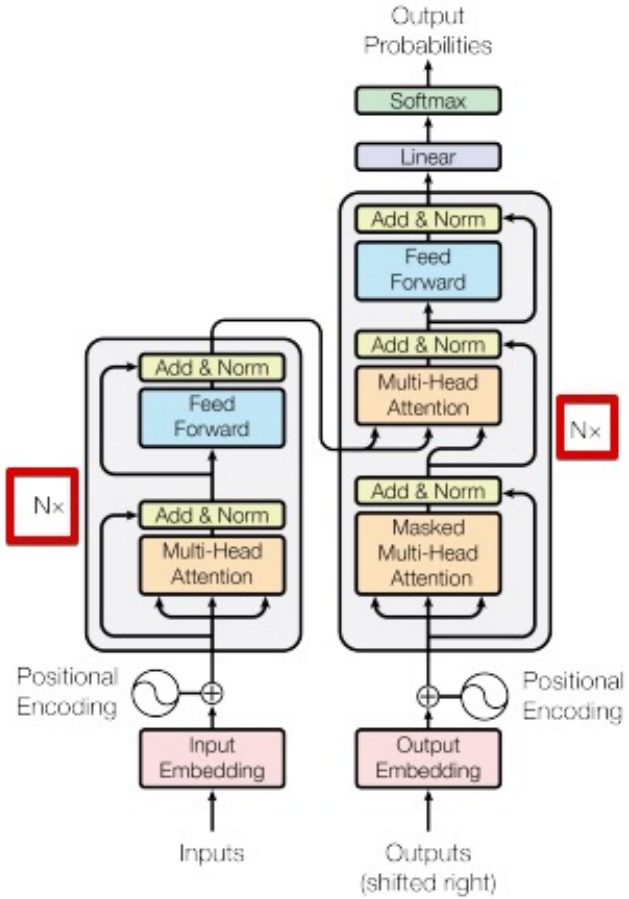
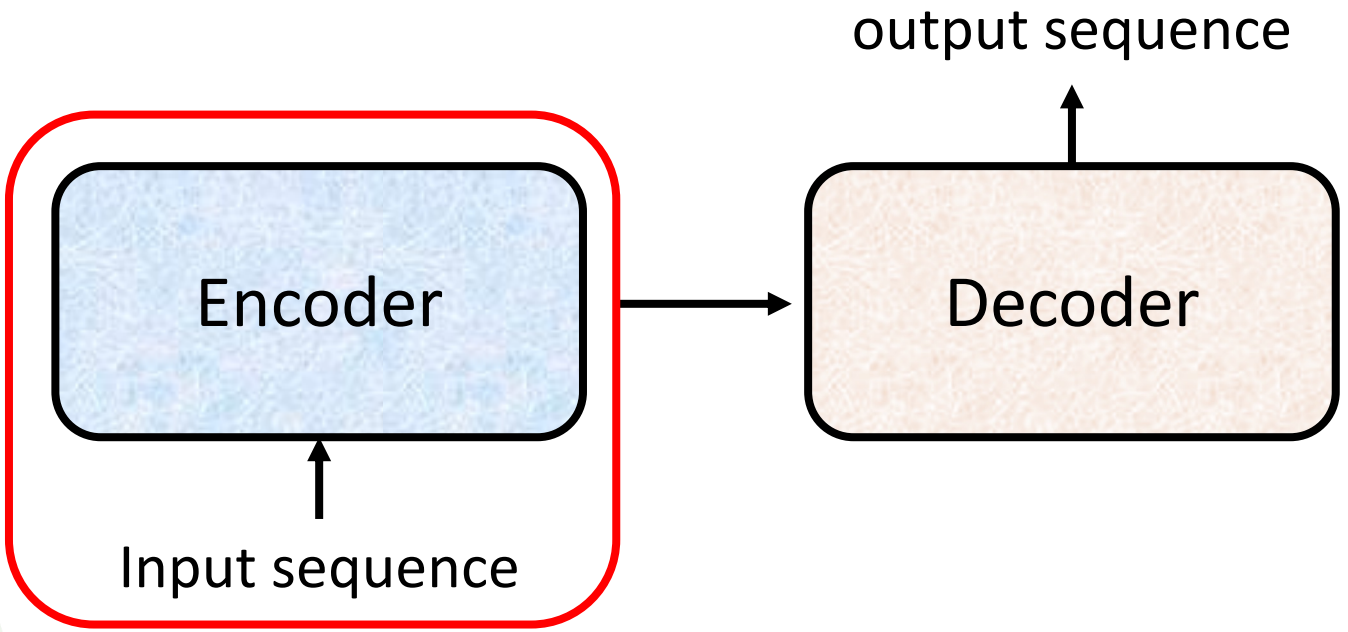


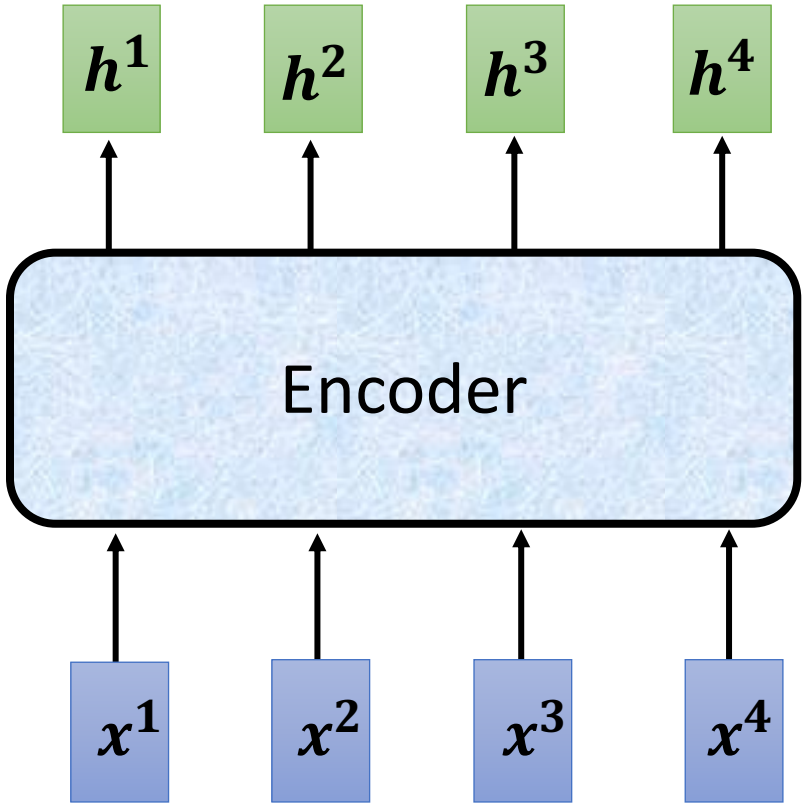
Figure 1: The Transformer - model architecture.

Encoder

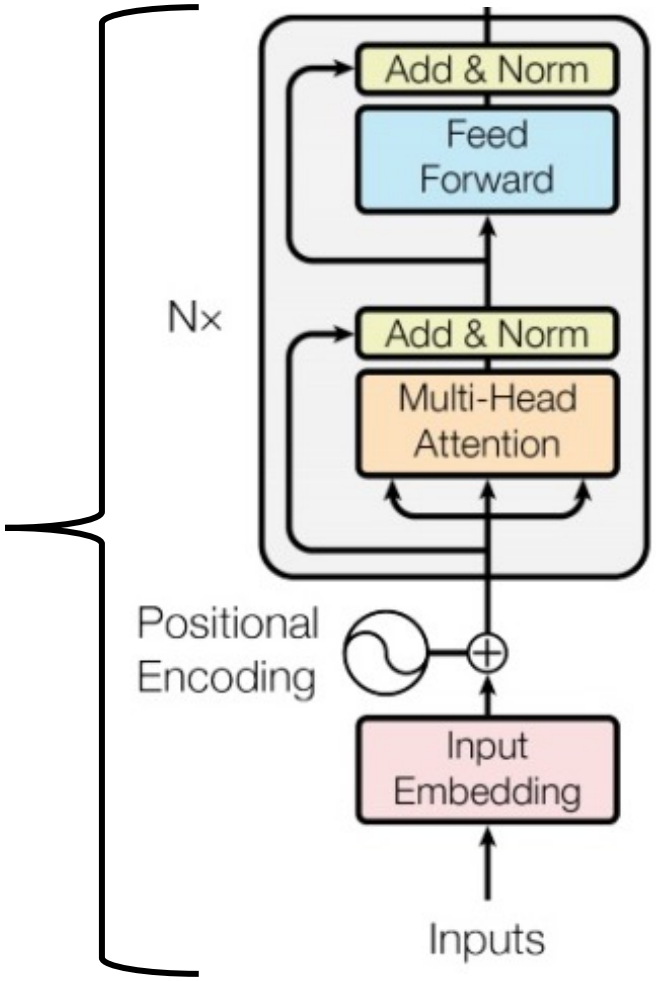


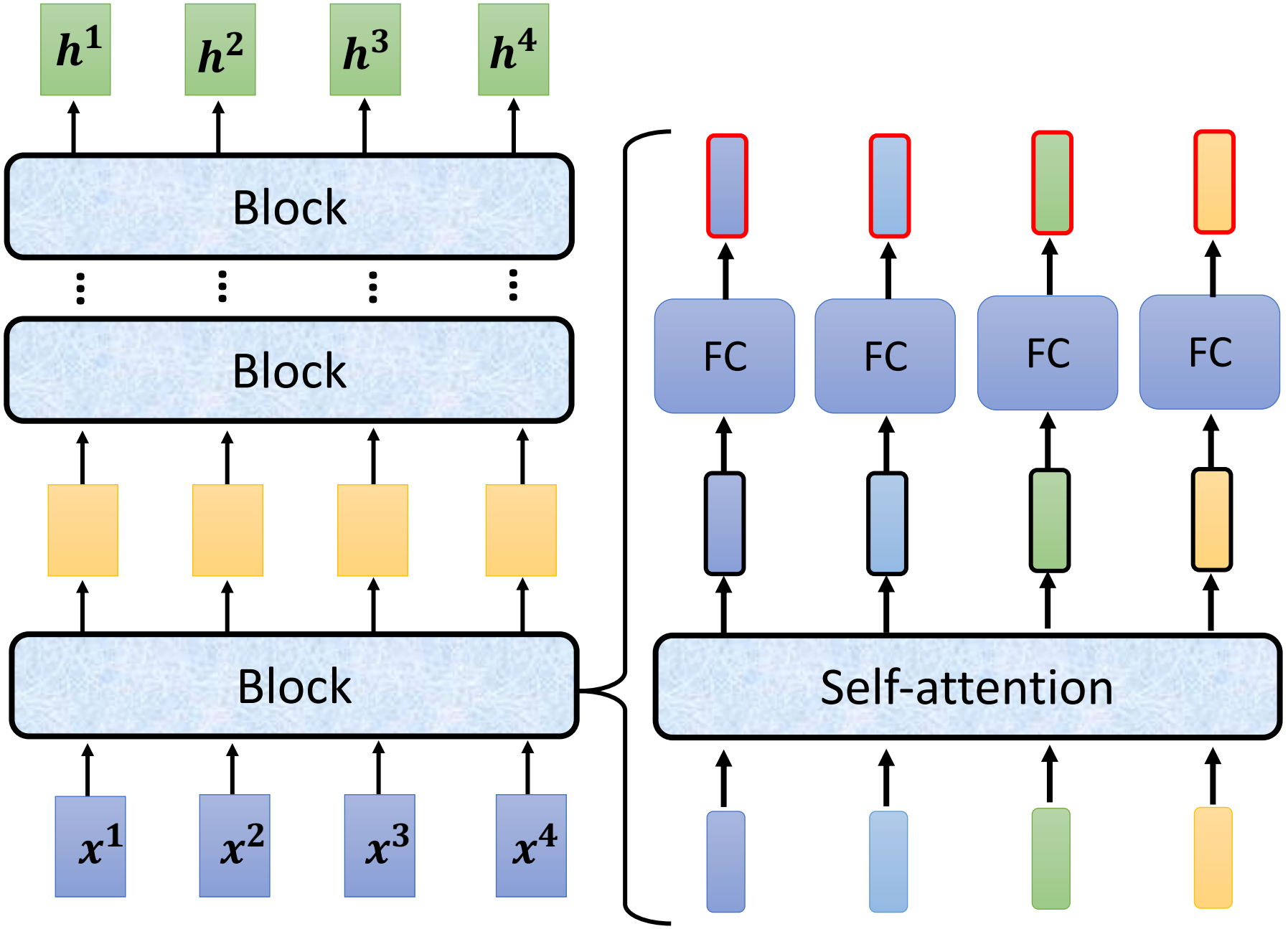
Encoder

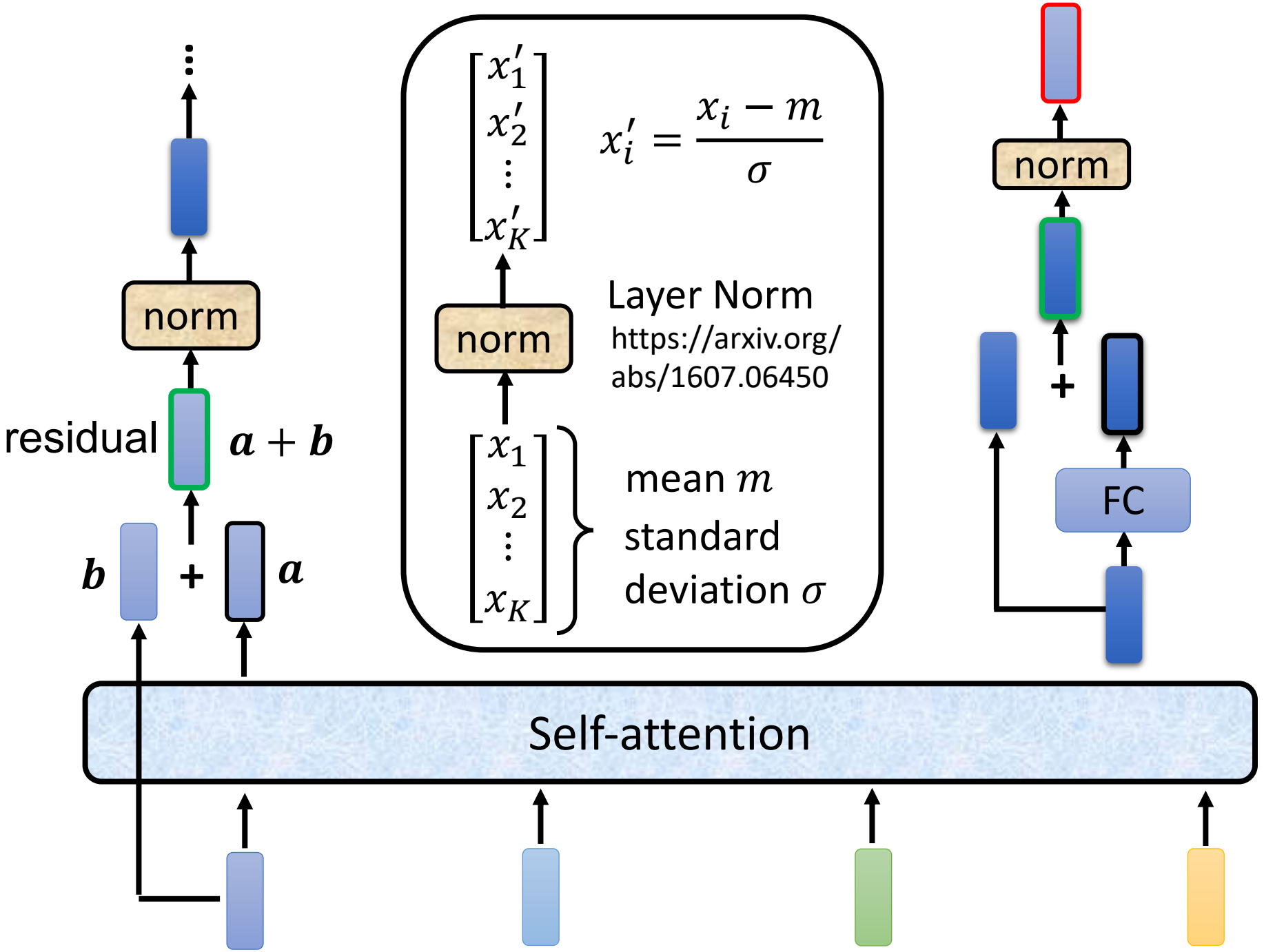
You can use **RNN** or **CNN**.



Transformer's Encoder

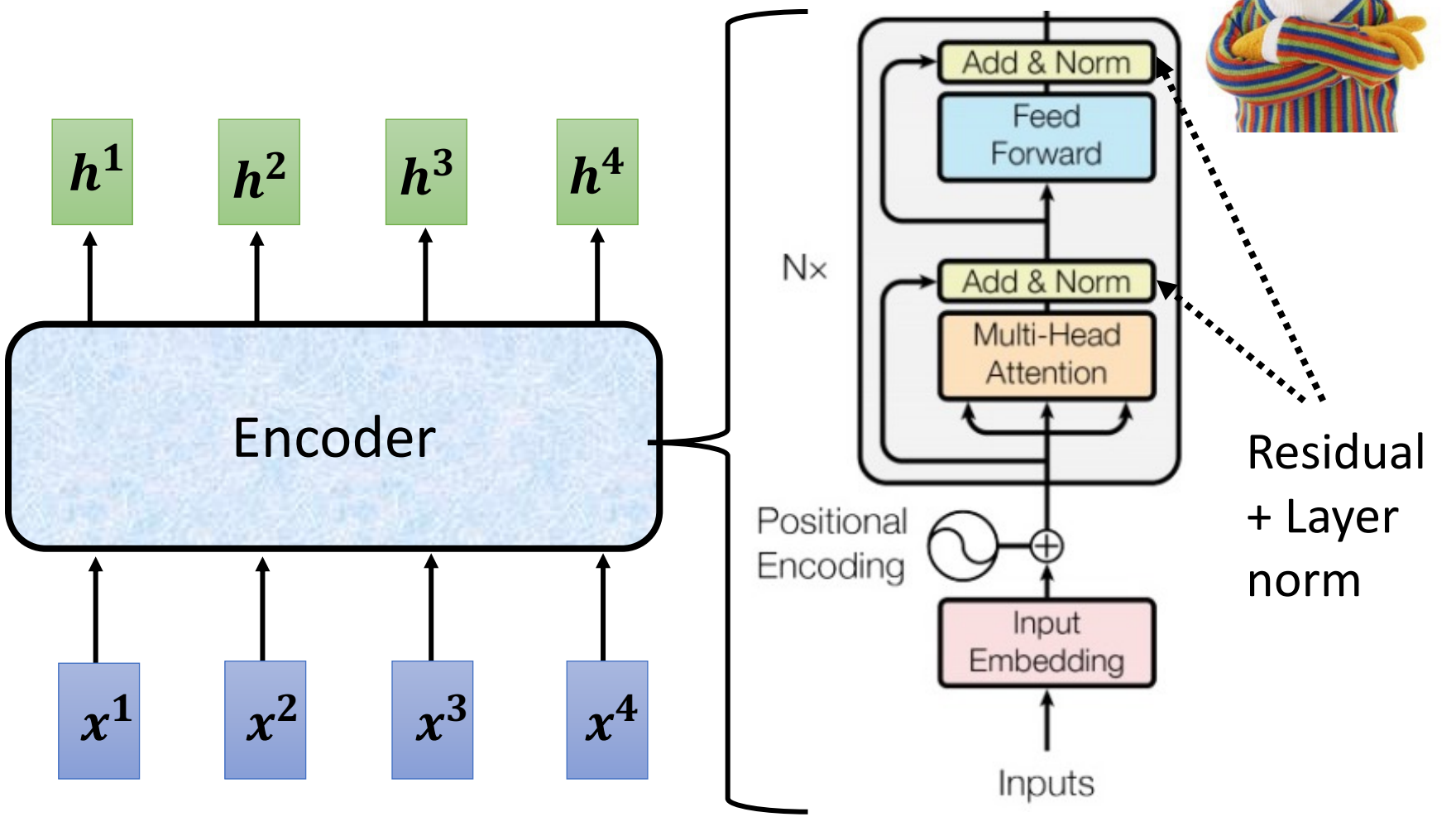



h^1
h^2
h^3
h^4
Block
Block
Block
x^1
x^2
x^3
x^4
FC
FC
FC
FC
Self-attention

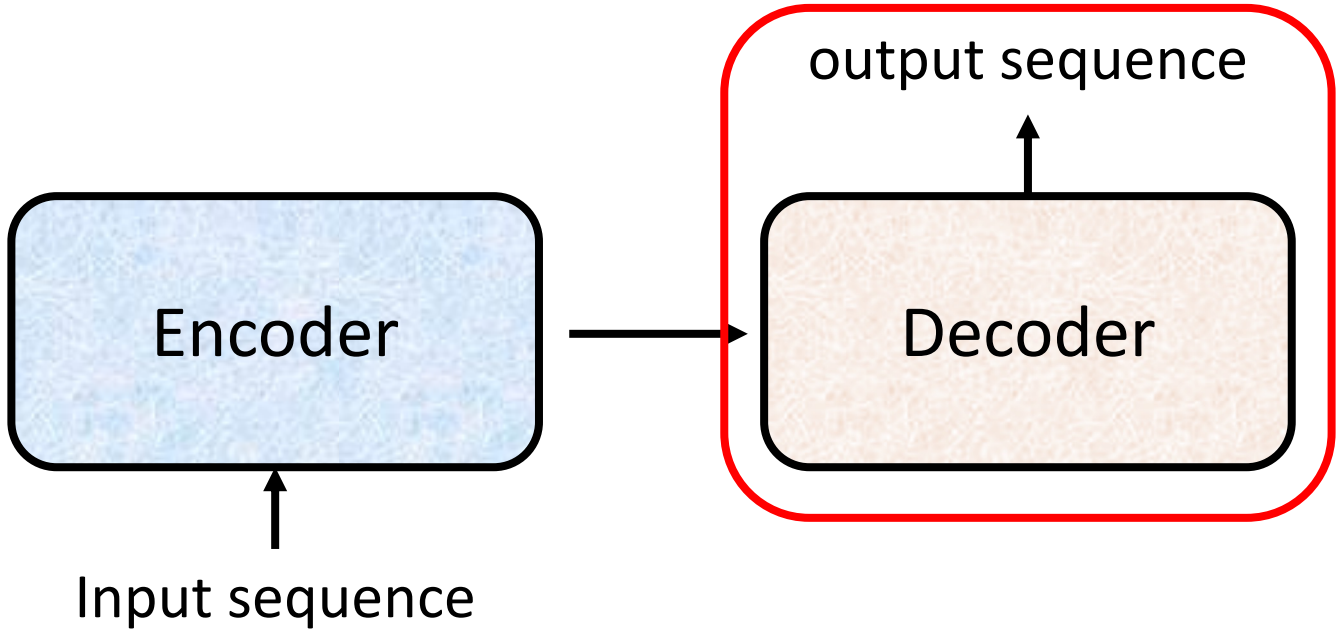

norm
residual
$a + b$
b
+
a
x'_1
x'_2
x'_K
$x_i' = \frac{x_i - m}{\sigma}$
norm
Layer Norm
https://arxiv.org/abs/1607.06450
x_1
x_2
x_K
mean m
standard deviation σ
norm
+
FC
Self-attention

BERT

I use the **same** network architecture as **transformer encoder**.

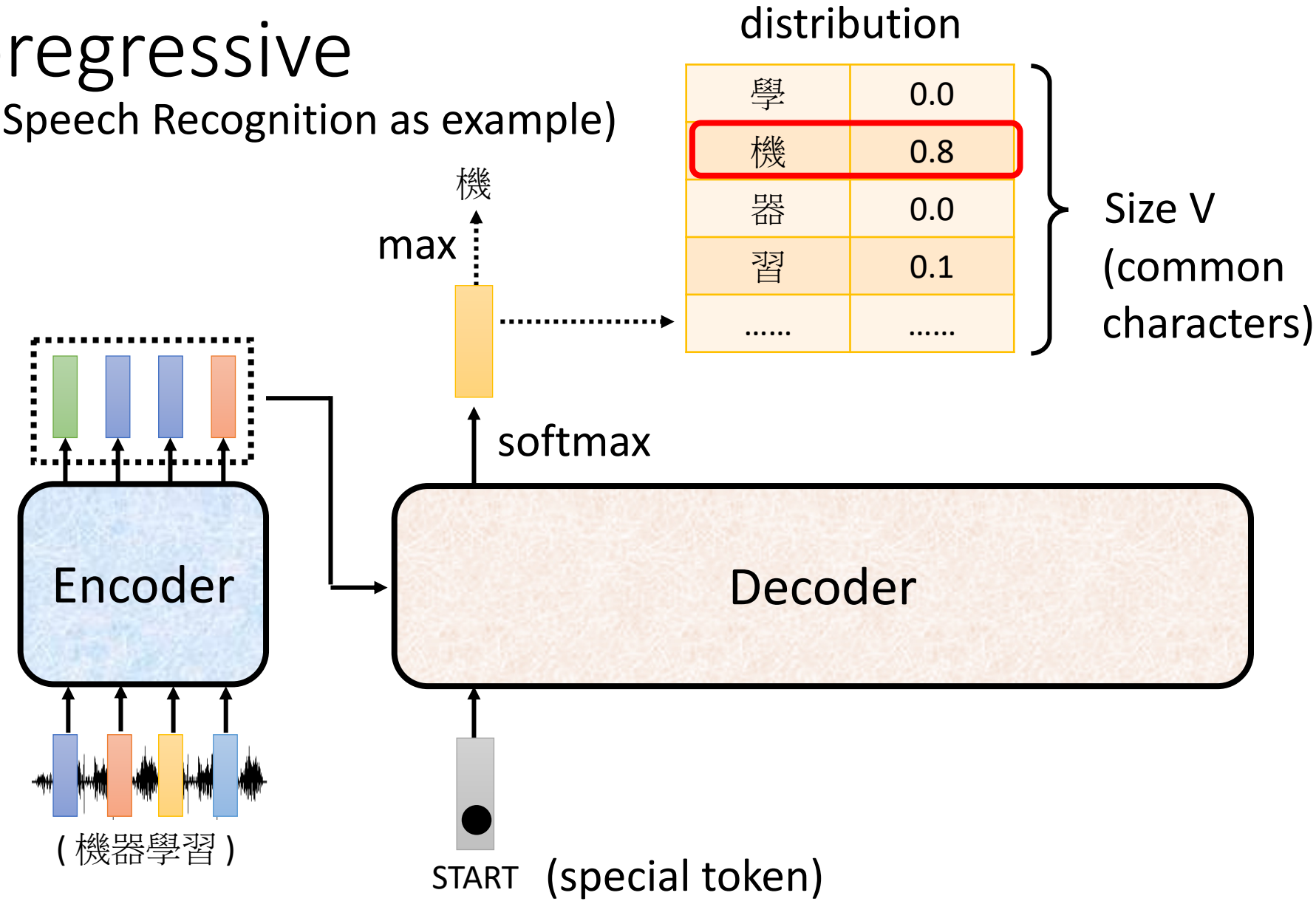


Decoder
– Autoregressive (AT)

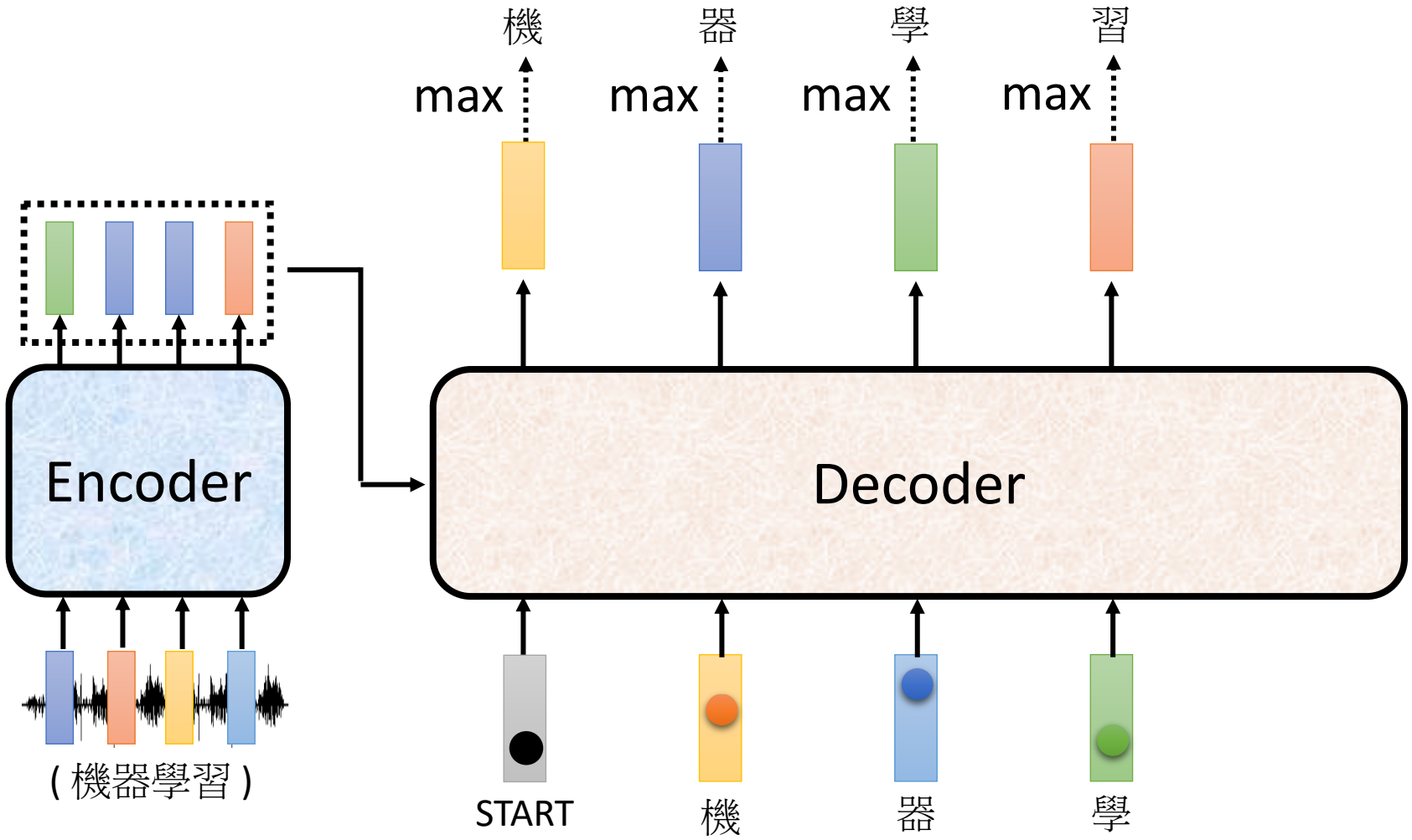


Autoregressive

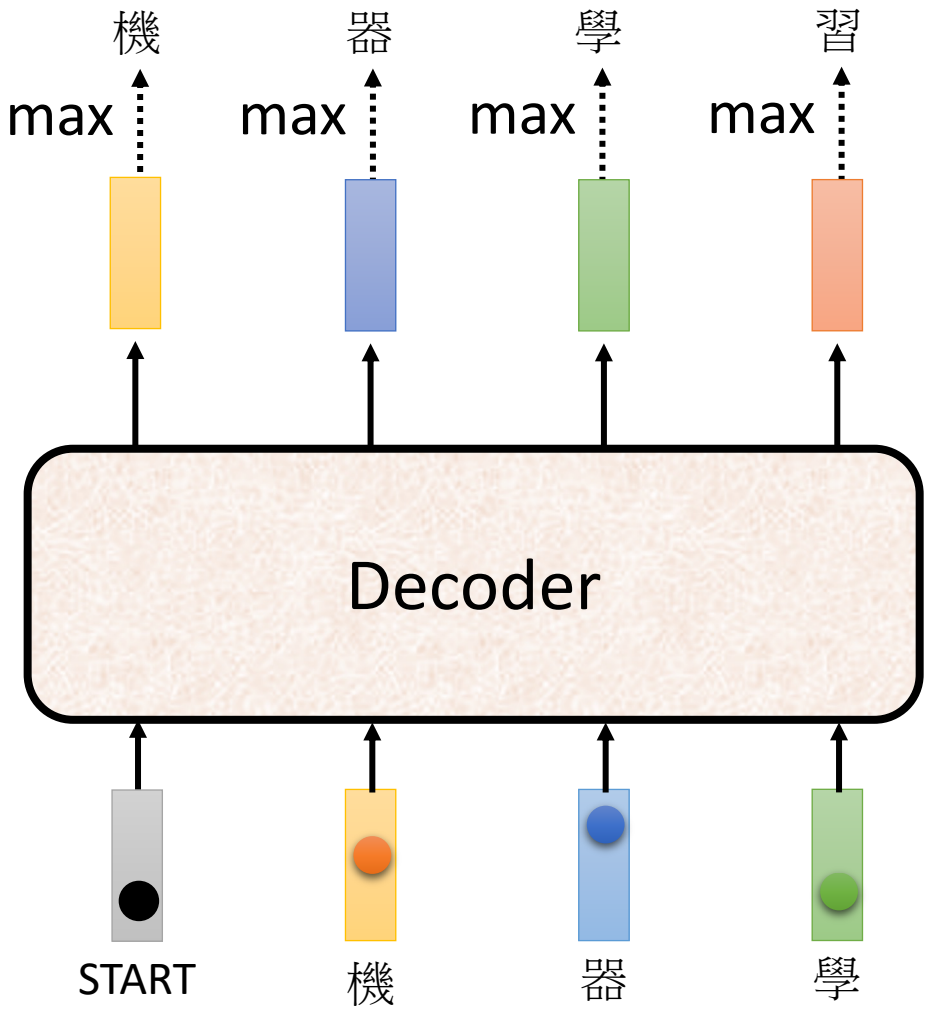
(Speech Recognition as example)

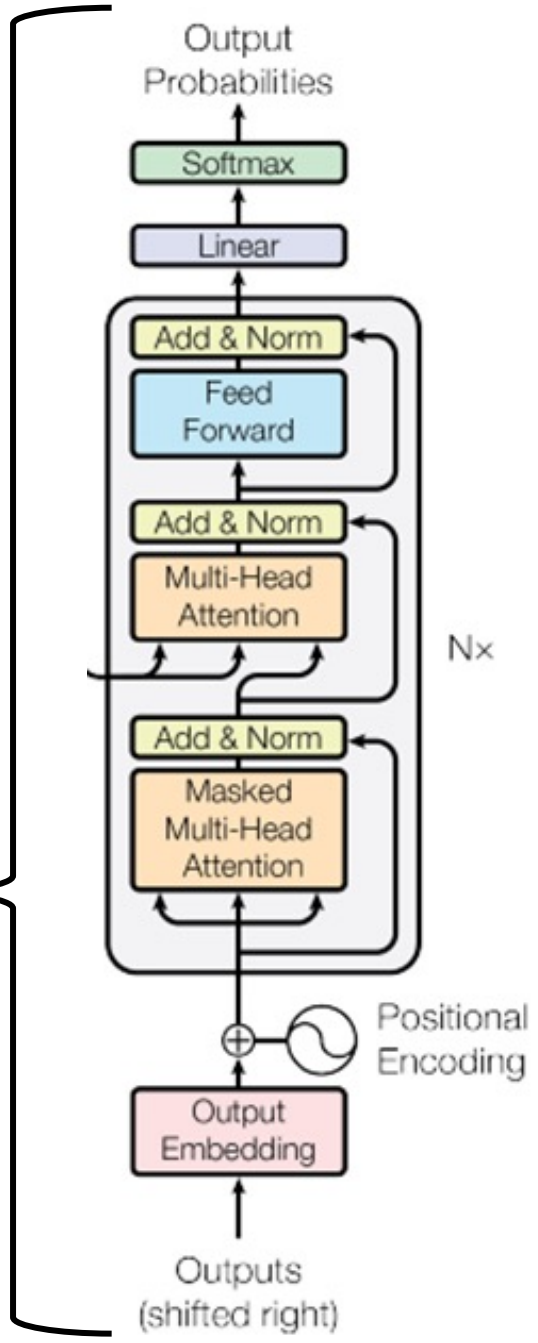


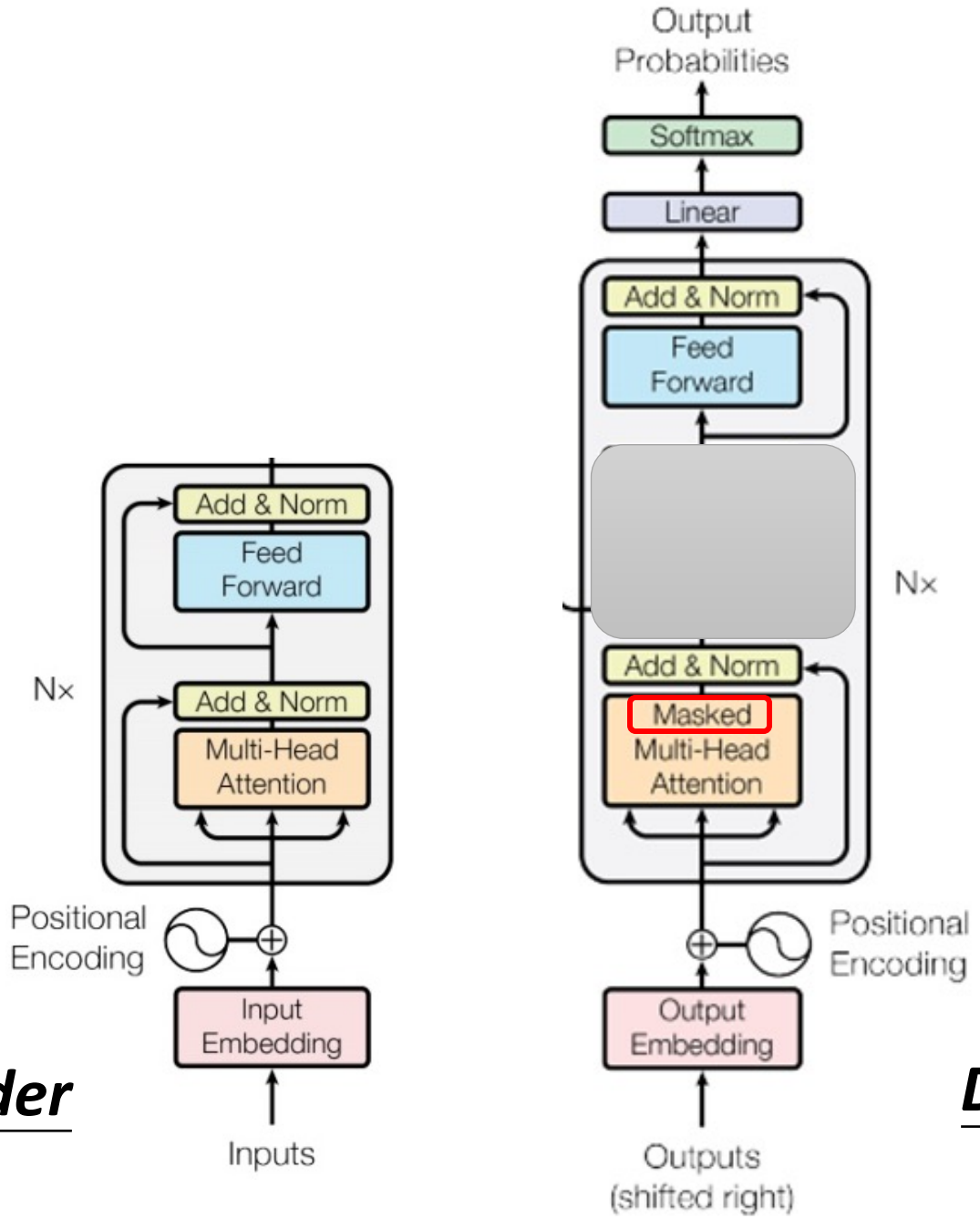
Autoregressive



ignore the input from the encoder here ☺



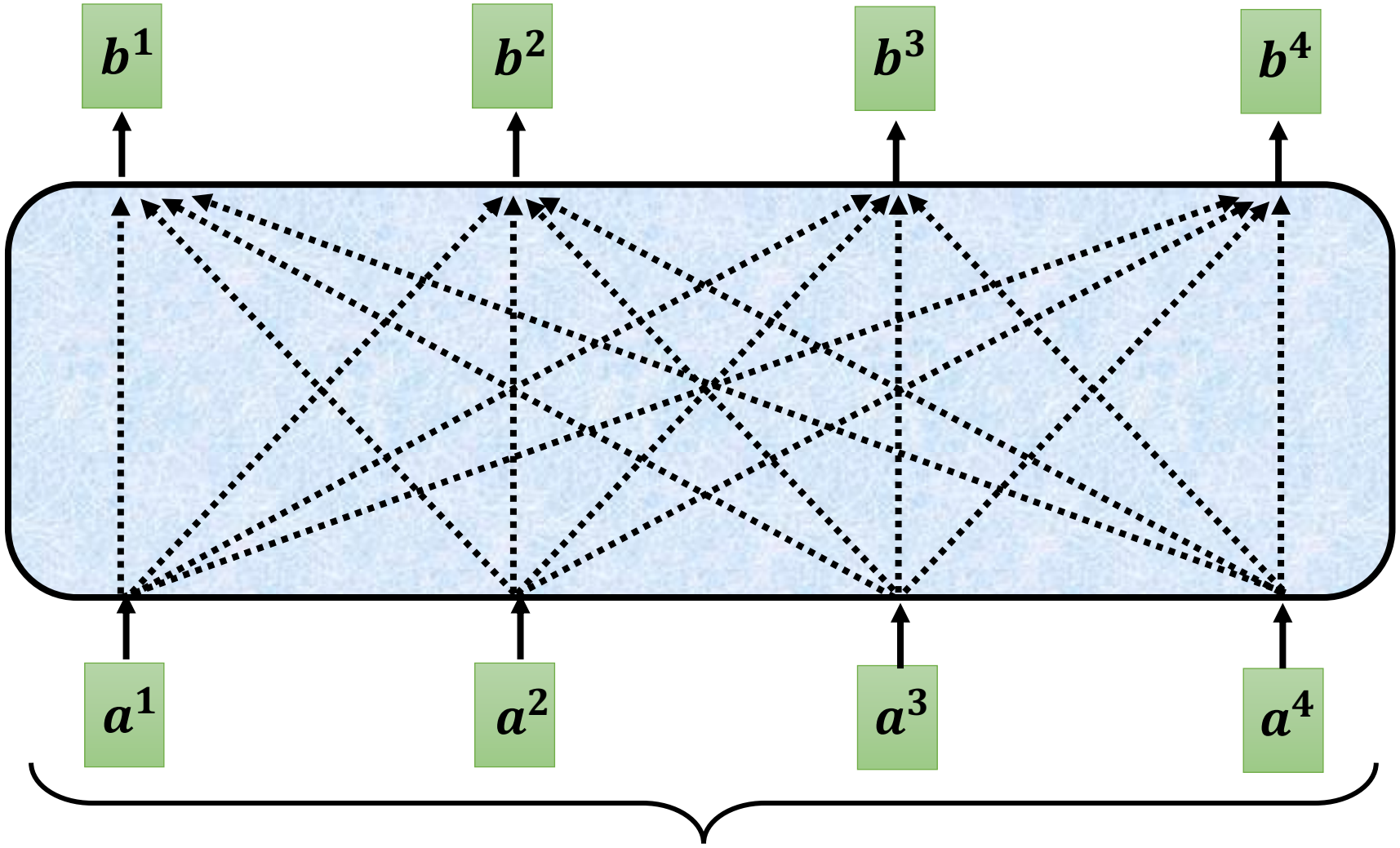




Encoder

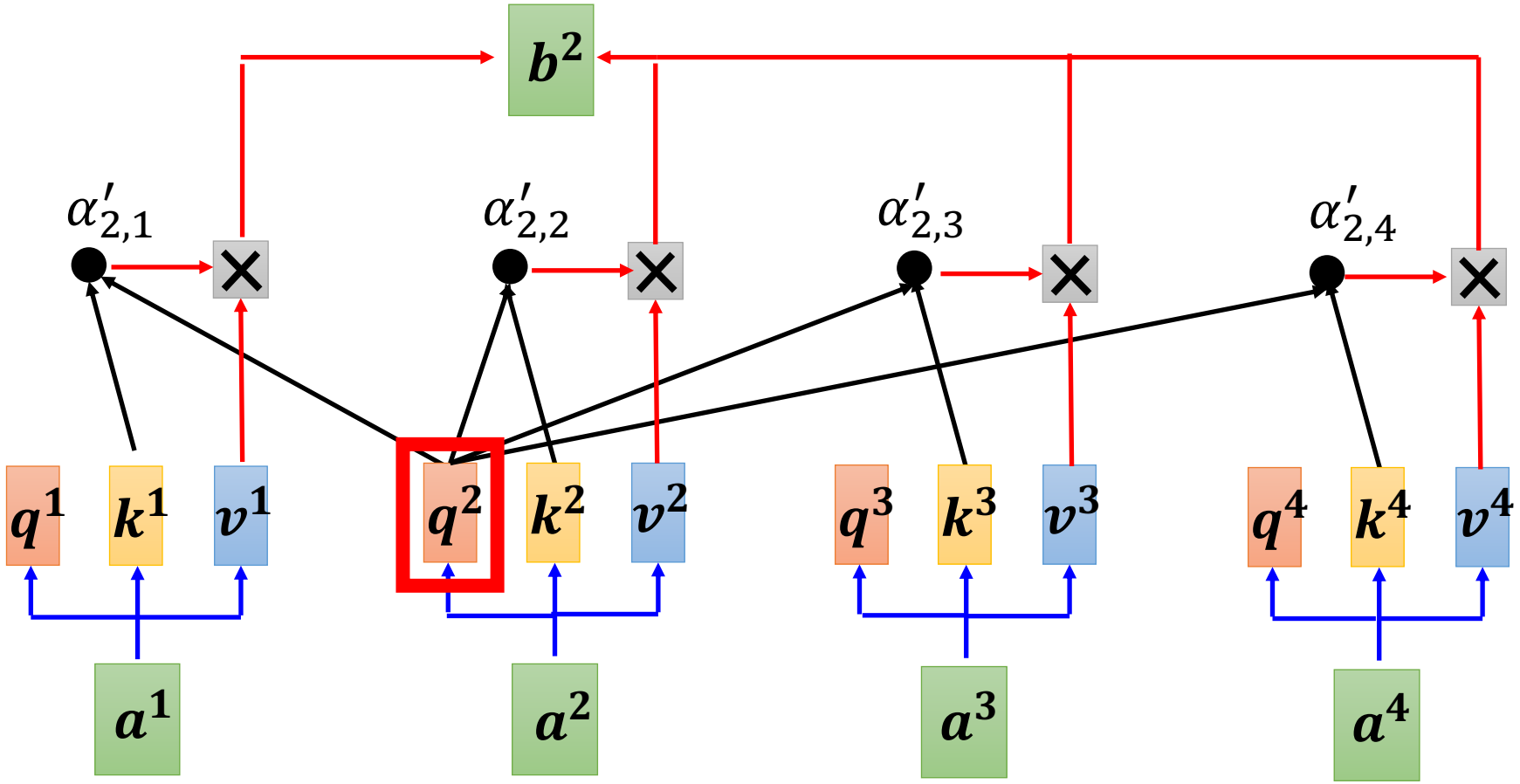
Decoder

Self-attention → Masked Self-attention



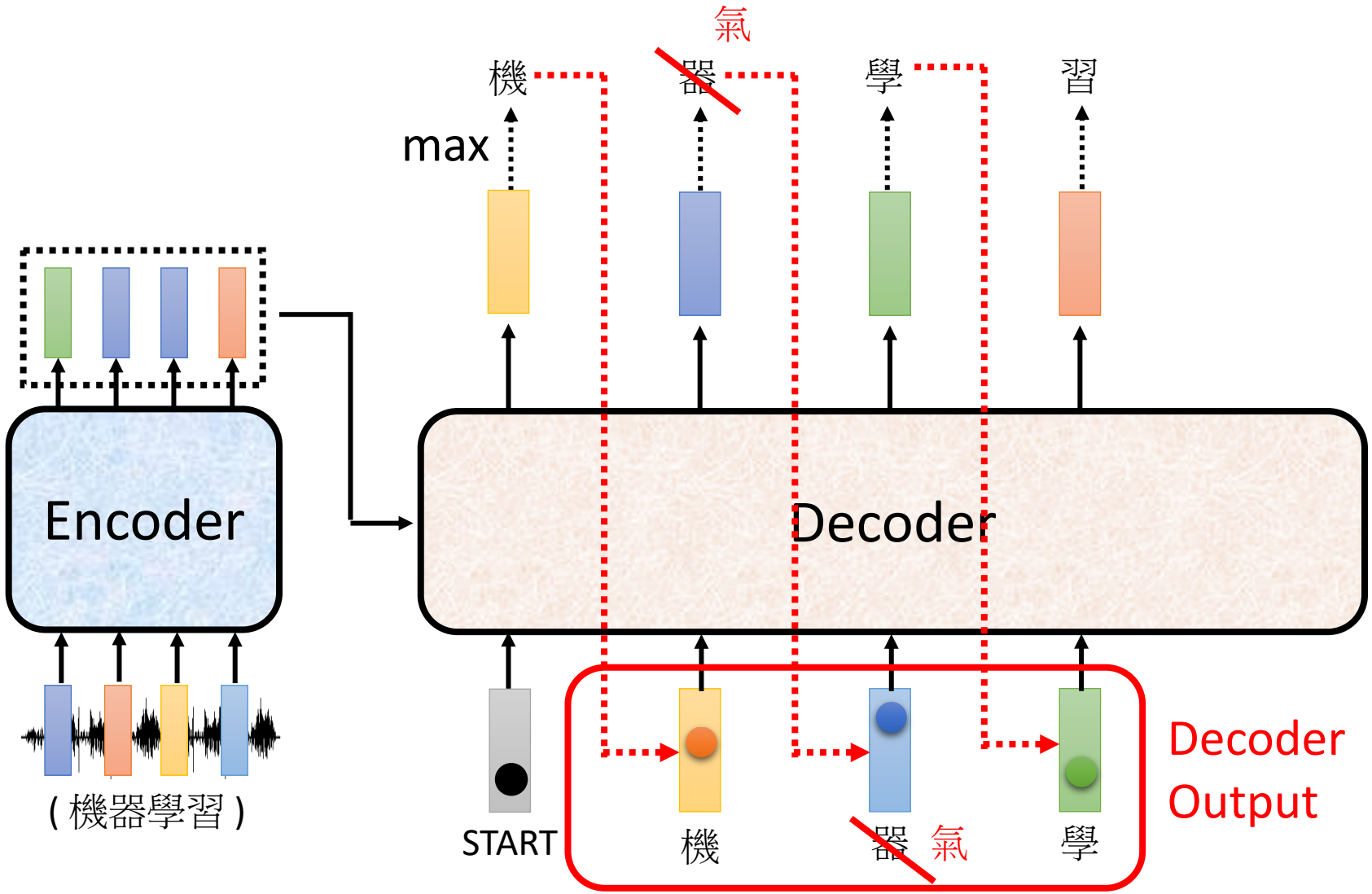
Can be either **input** or **a hidden layer**

Self-attention → Masked Self-attention



Why masked? Consider how does decoder work

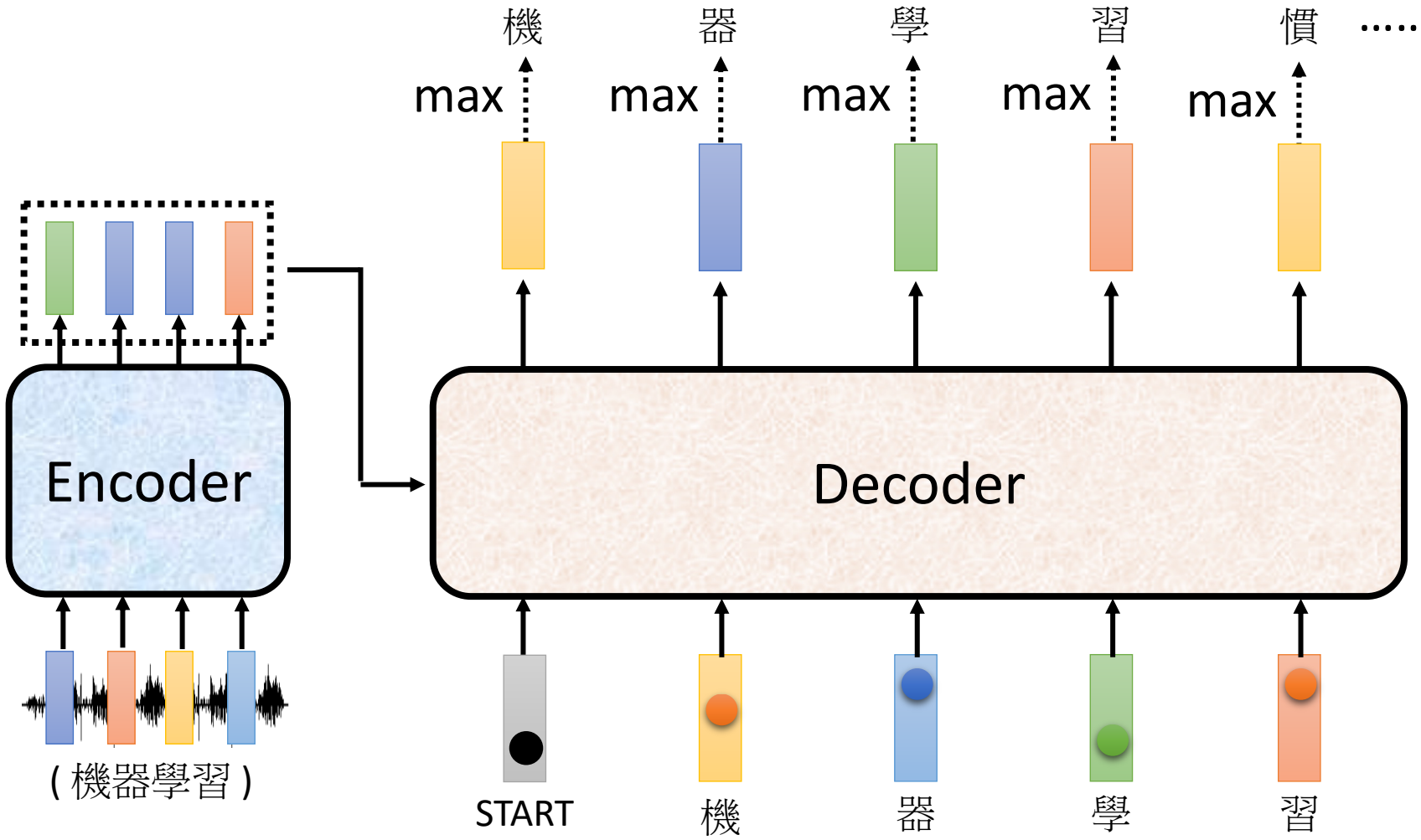
Autoregressive



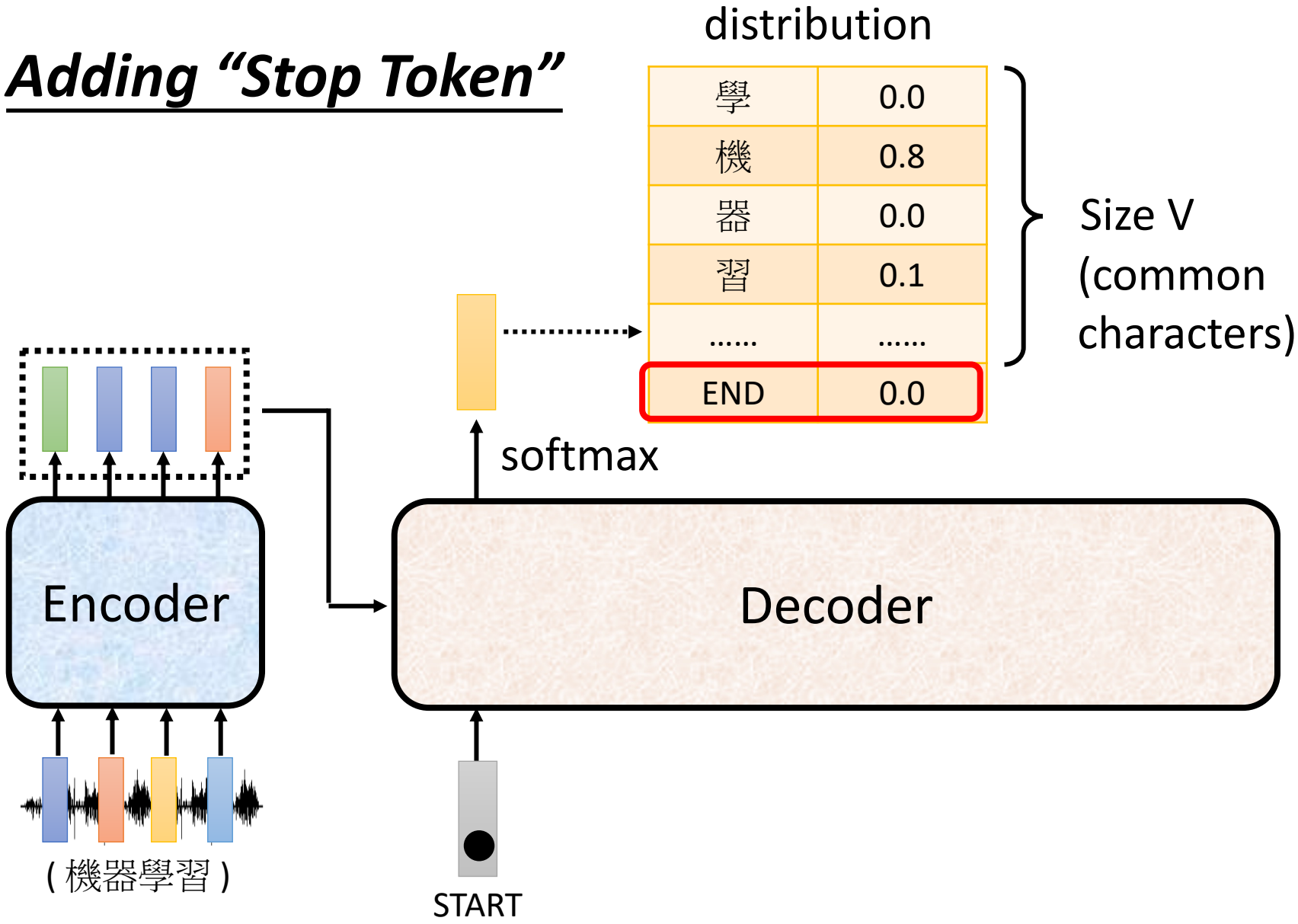
Autoregressive

We do not know the correct output length.

Never stop!

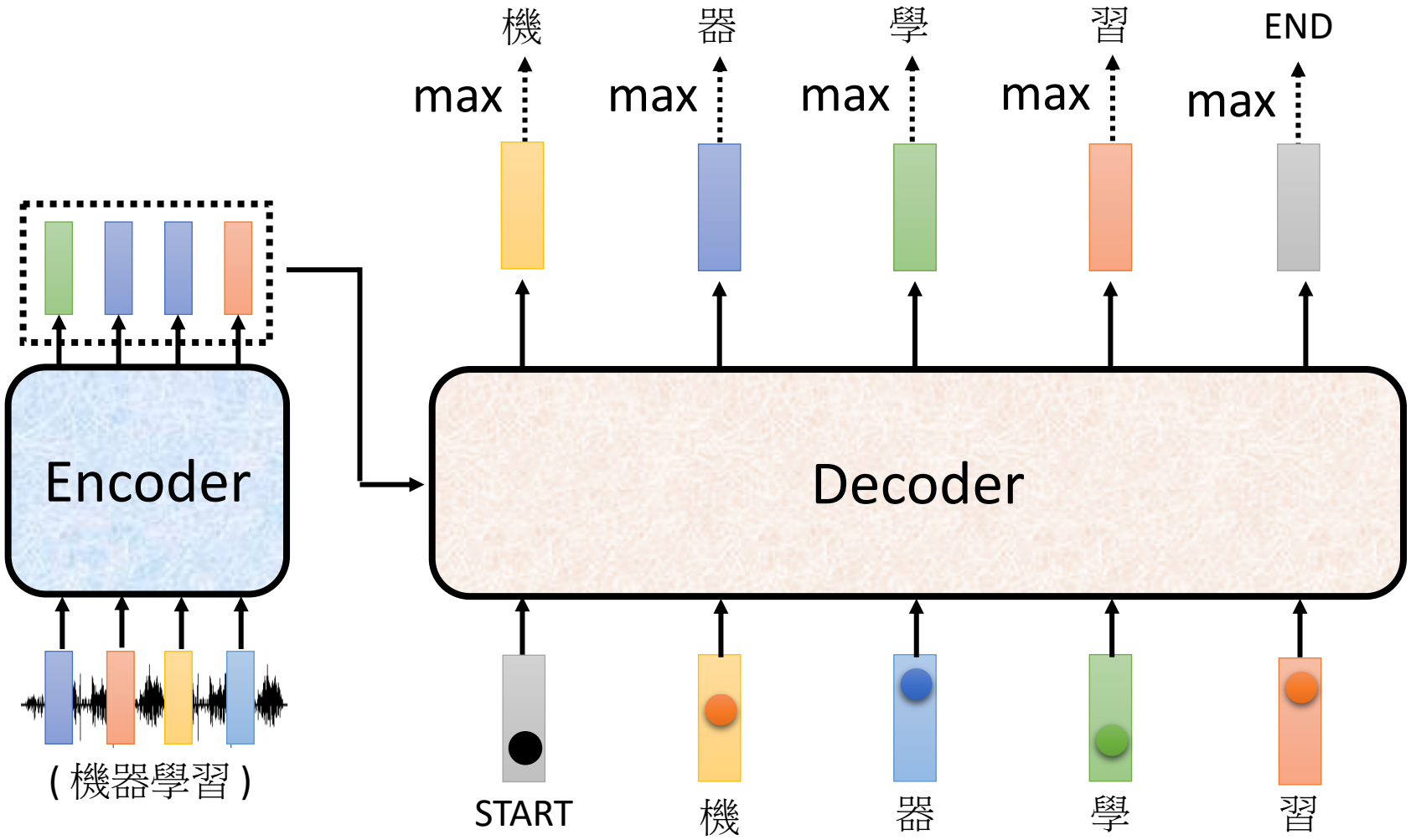


Adding "Stop Token"

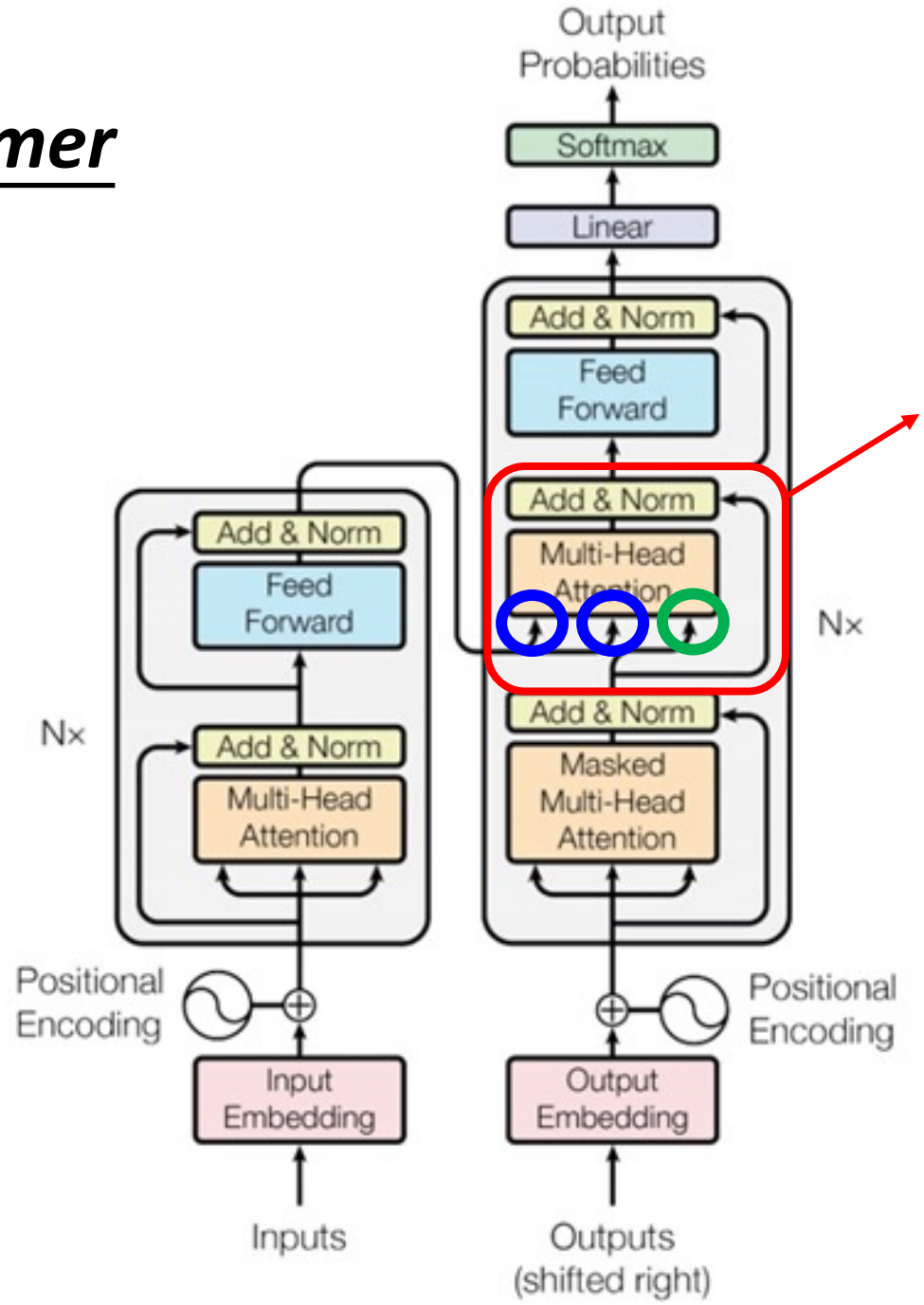


Autoregressive

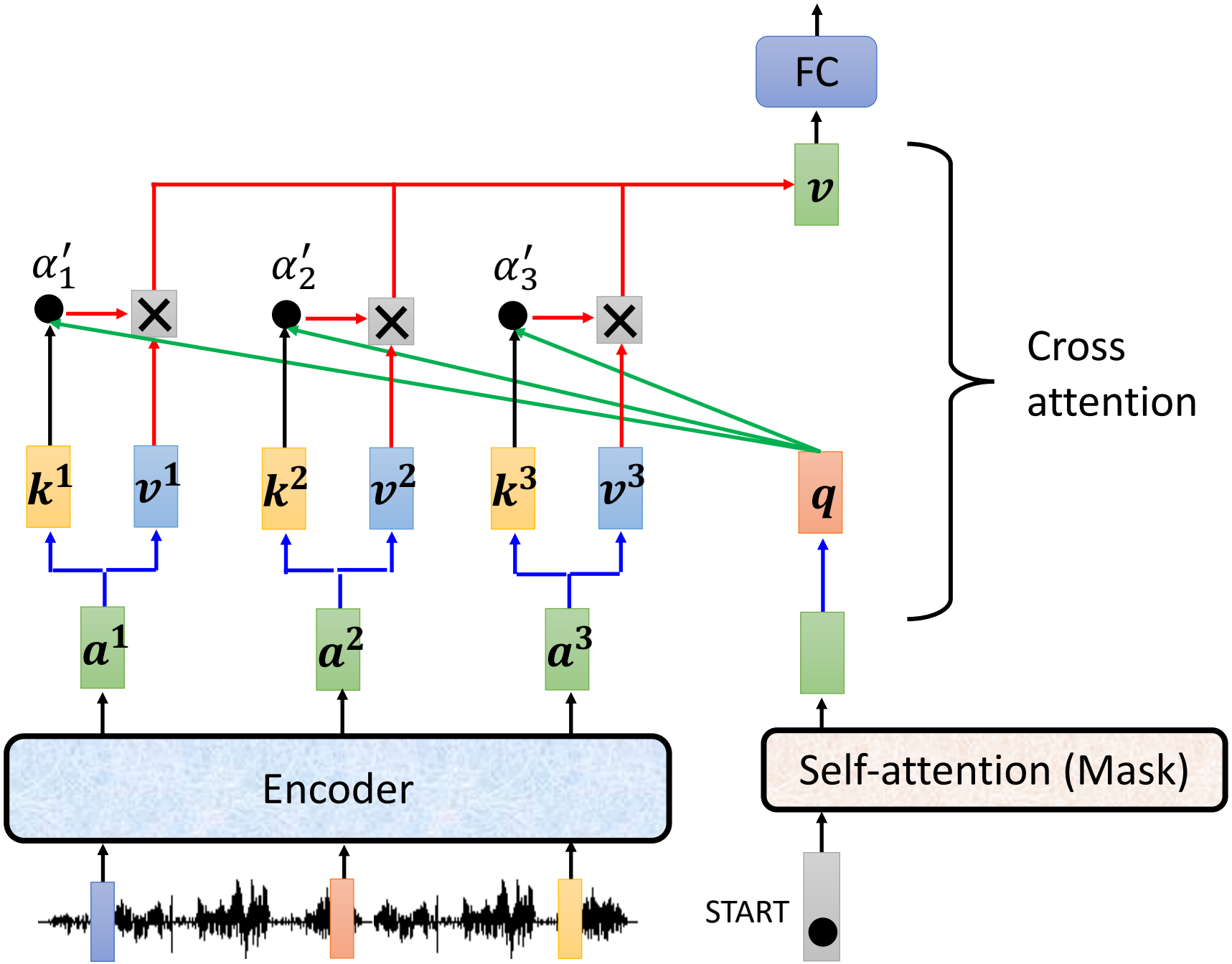
Stop at here!

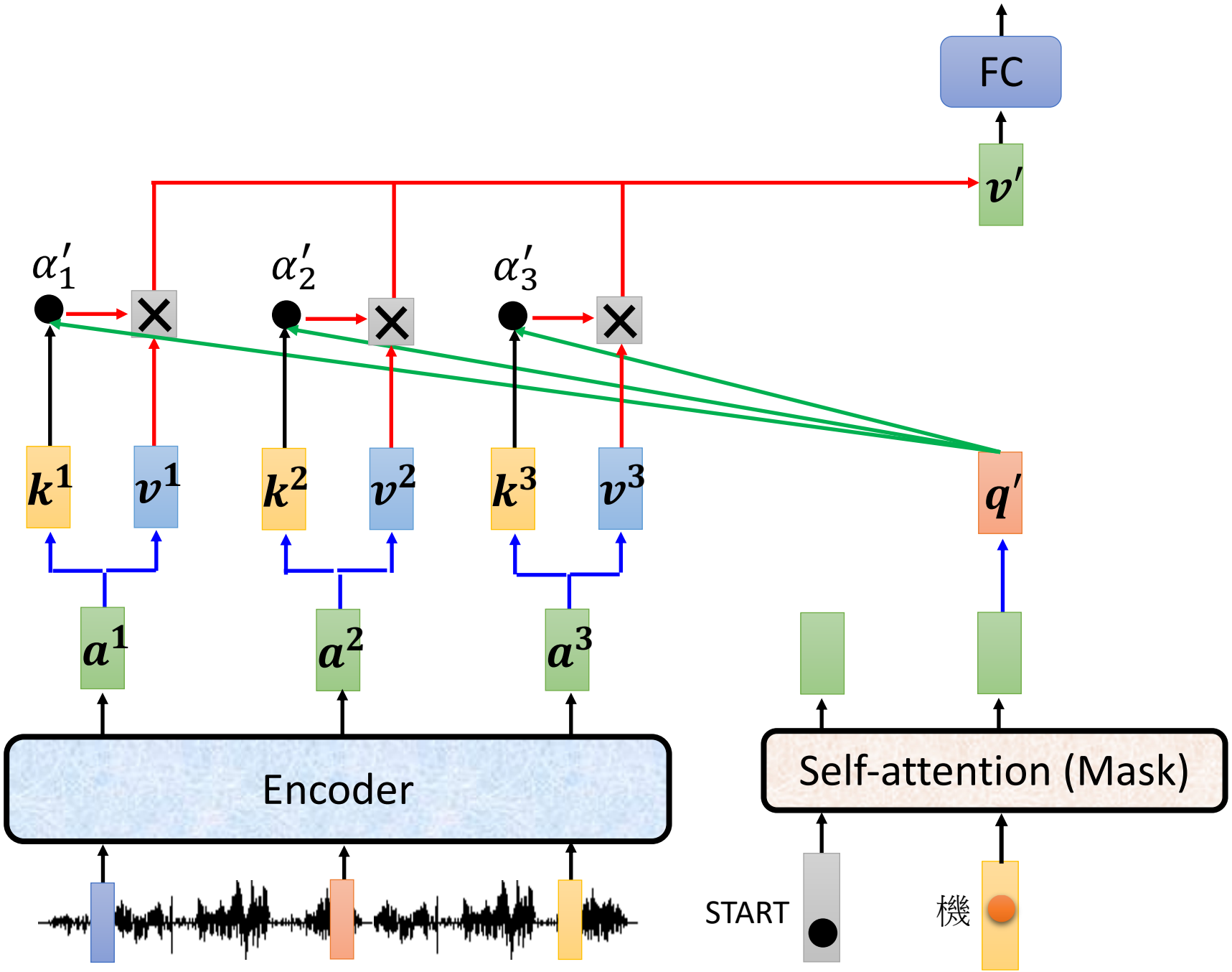


Transformer



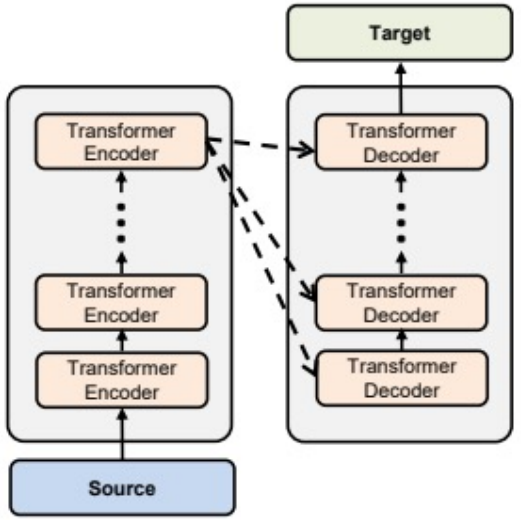
Cross attention


FC
v
α'_1
α'_2
α'_3
Cross attention
k^1
v^1
k^2
v^2
k^3
v^3
q
a^1
a^2
a^3
Encoder
Self-attention (Mask)
START

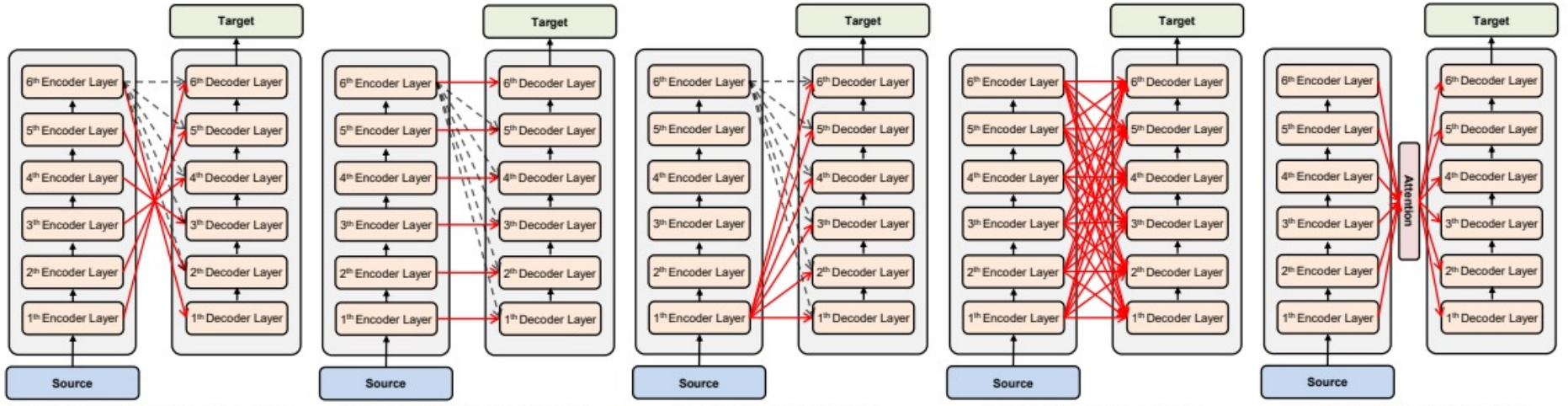

FC
v'
α'_1
α'_2
α'_3
k^1
v^1
k^2
v^2
k^3
v^3
q'
a^1
a^2
a^3
Encoder
Self-attention (Mask)
START
機

Cross Attention

Source of image:
https://arxiv.org/abs/2005.08081

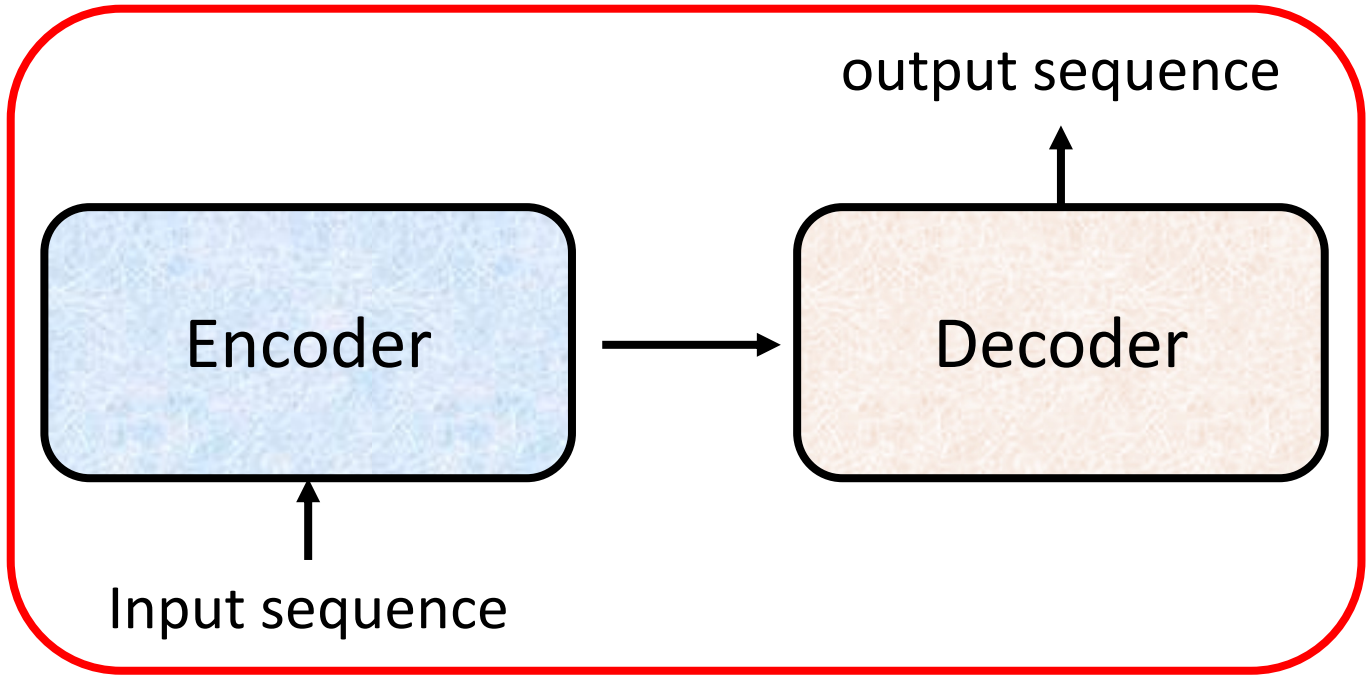


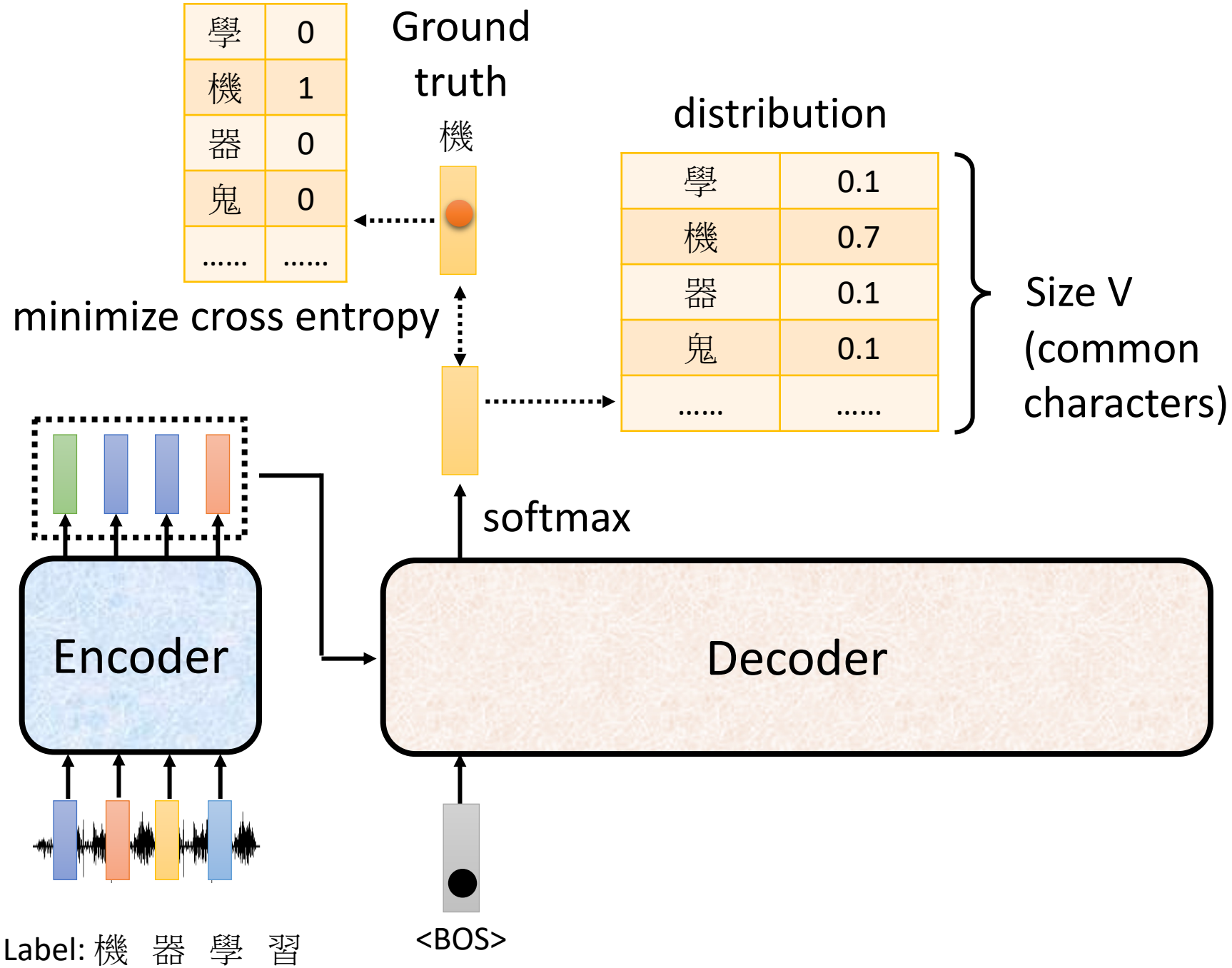
(a) Conventional Transformer



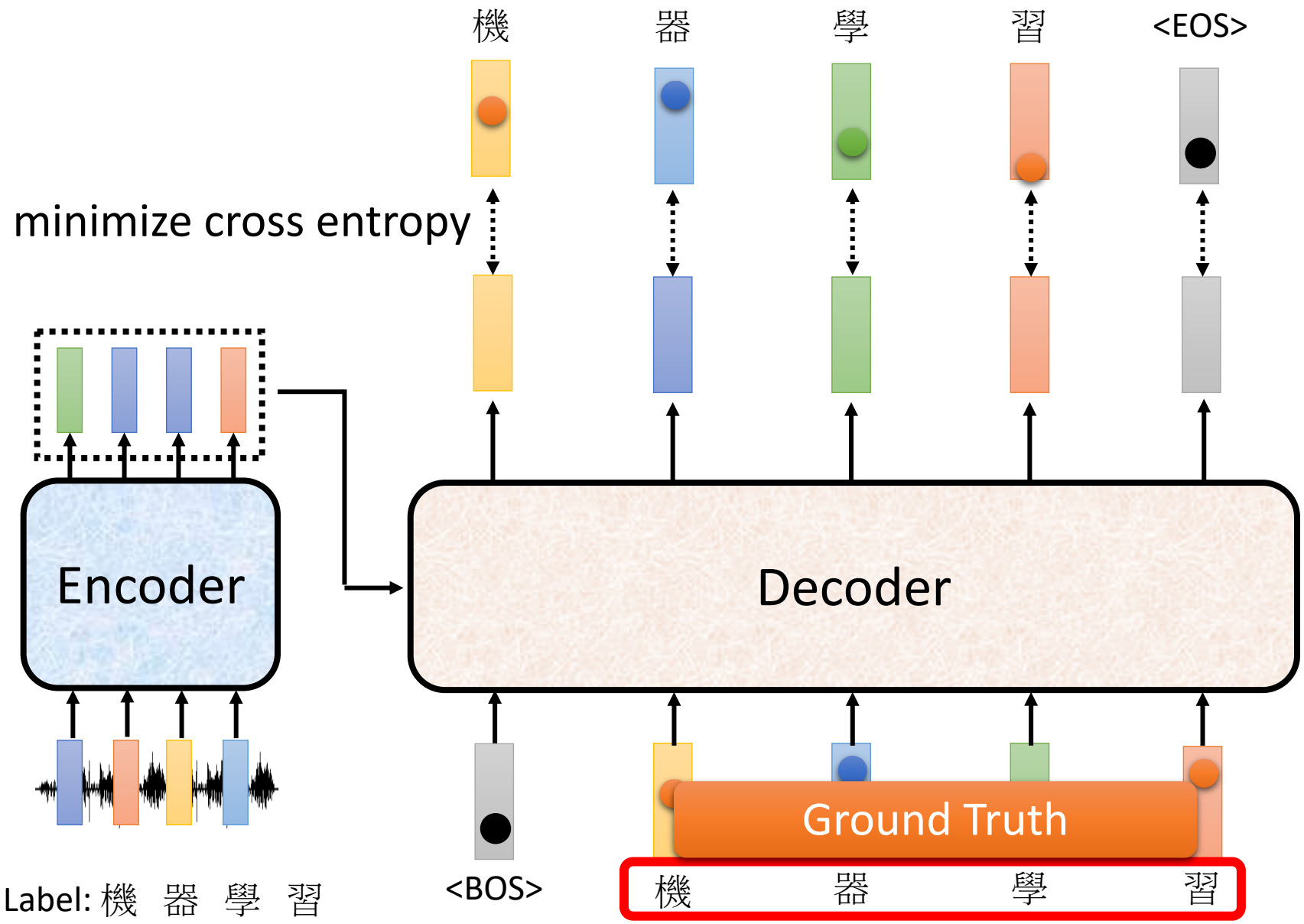
(a) Granularity Consistent Attention (b) Granularity Parallel Attention (c) Fine-Grained Attention (d) Full Matching Attention (e) Adaptive Matching Attention

Training

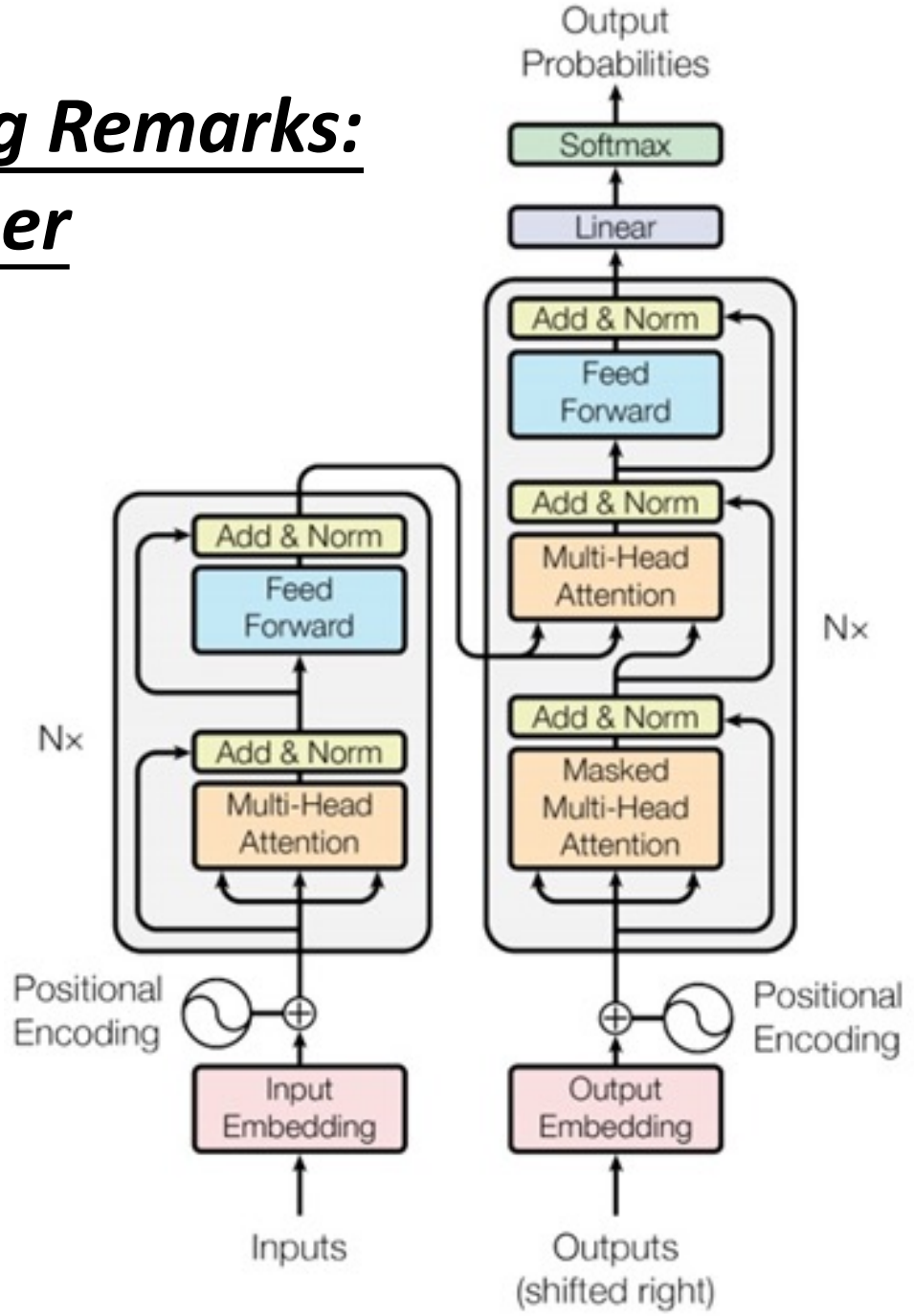



學 0
機 1
器 0
鬼 0
....
Ground truth
機
distribution
學 0.1
機 0.7
器 0.1
鬼 0.1
....
Size V (common characters)
minimize cross entropy
softmax
Encoder
Decoder
<BOS>
Label: 機 器 學 習

Teacher Forcing: using the ground truth as input.



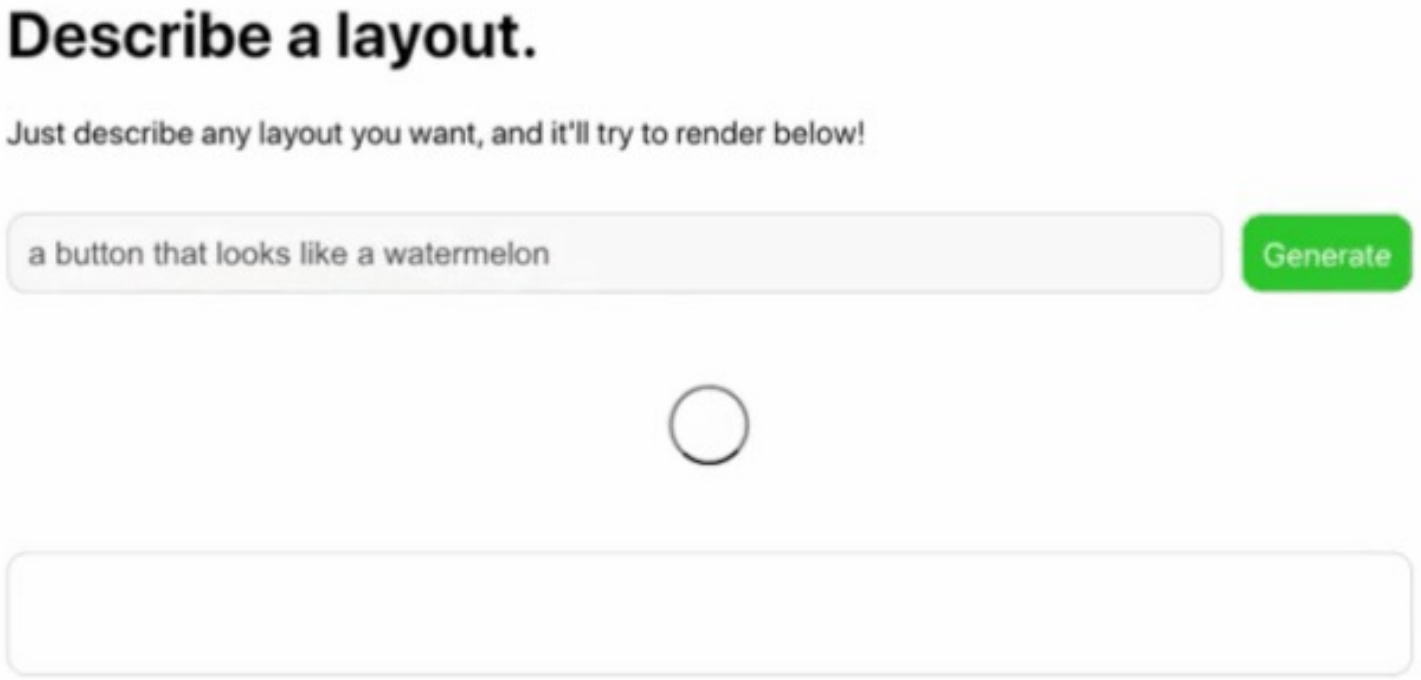
Concluding Remarks: Transformer



Overview of GPT-3

- **GPT-3 (Generative Pre-trained Transformer 3)**
  - the successor to GPT-2
  - the most recent(2020,May) language model coming from the OpenAI research lab team.

Describe HTML layout



3. Text Processing — GPT-3 Applications

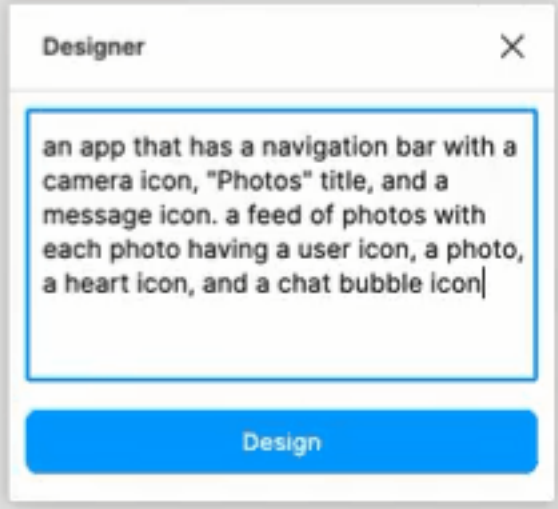
the ugliest emoji ever

Generate

```
<div style={{borderRadius: '100%', borderWidth: 1, border: '5px solid red',
padding: 20, width: 100, height: 100, backgroundColor: 'yellow'}}>😂</div>
```

Create UI

创建一个带照相机图标、Photo为标题、信息图标的导航栏，在应用中发送照片，每张照片有一个用户图标、一个点赞图标，和一个聊天泡泡图标



Excel Completion

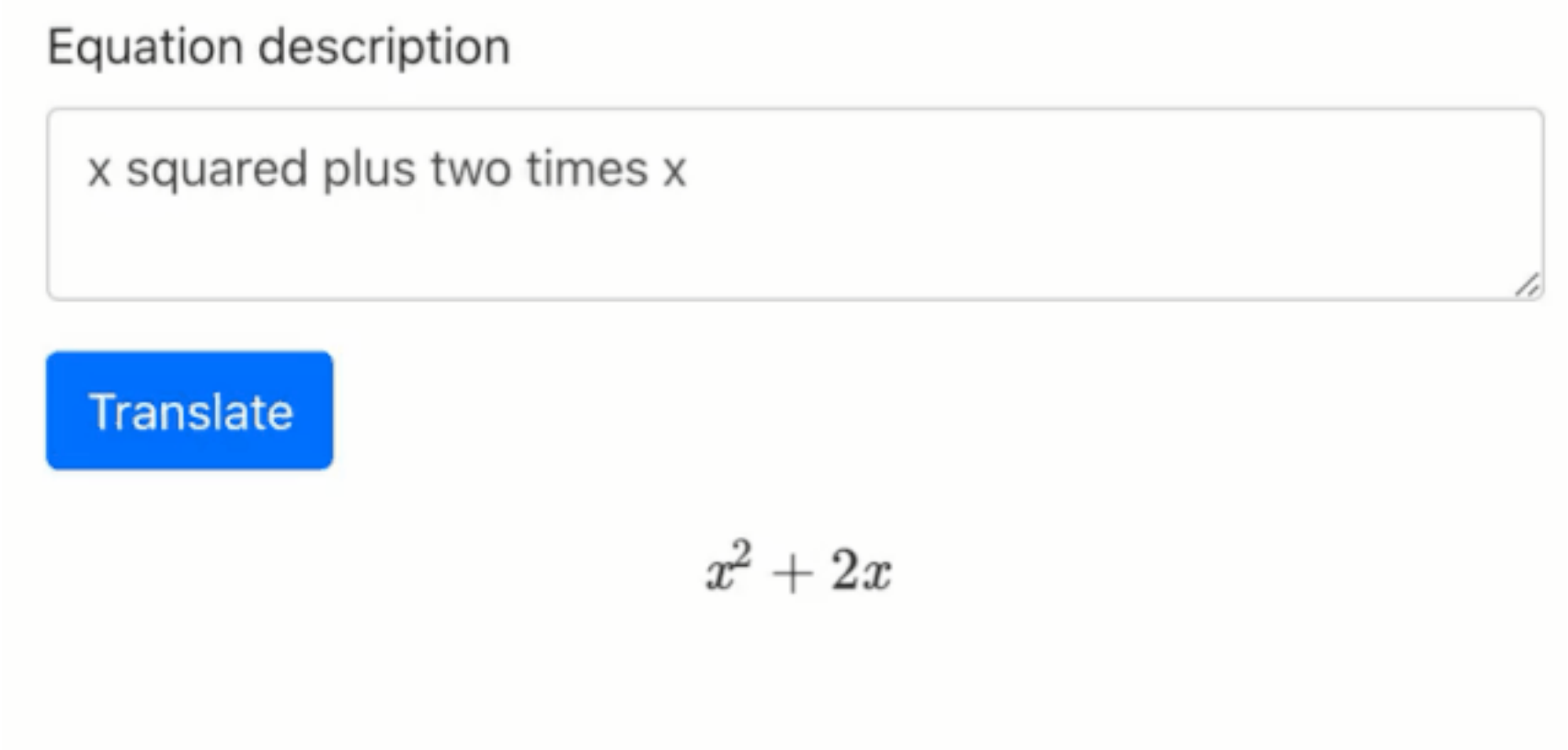
	A	B
1	State Name	Population
2	Illinois	12.67M
3	California	39.51M
4	Ohio	11.69M
5	Michigan	

3. Text Processing — GPT-3 Applications

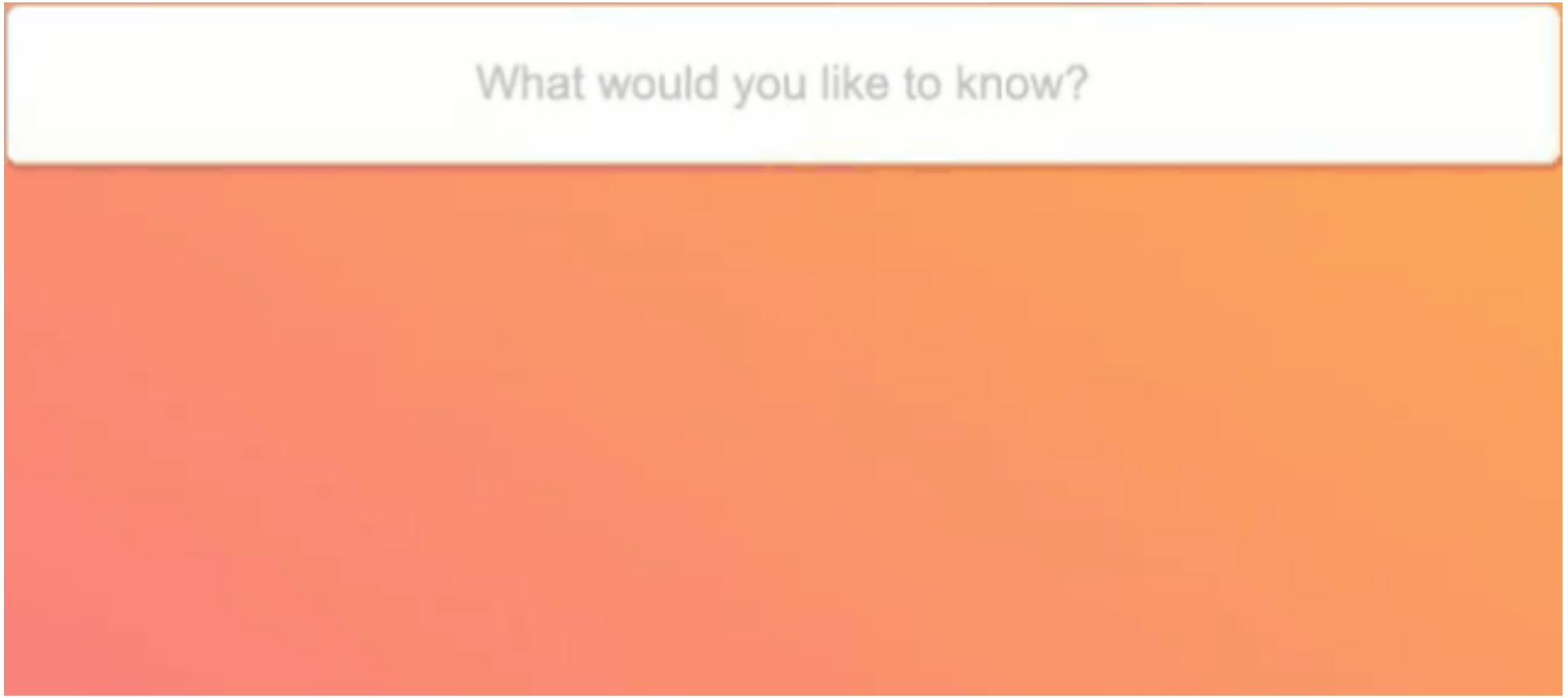
Python or Graph Generation: 对家庭成员的数据分析也不在话下。

这个应用更根据自然语言描述生成条形统计图，以及相应的Python代码。例如我们输入“在我家，我的姐姐5岁，我的妈妈46岁”，应用就能生成统计“姐姐”和“妈妈”岁数的条形图，如果再加上“我的爸爸比妈妈大6岁”，就能再生成“爸爸”52岁的条形图。

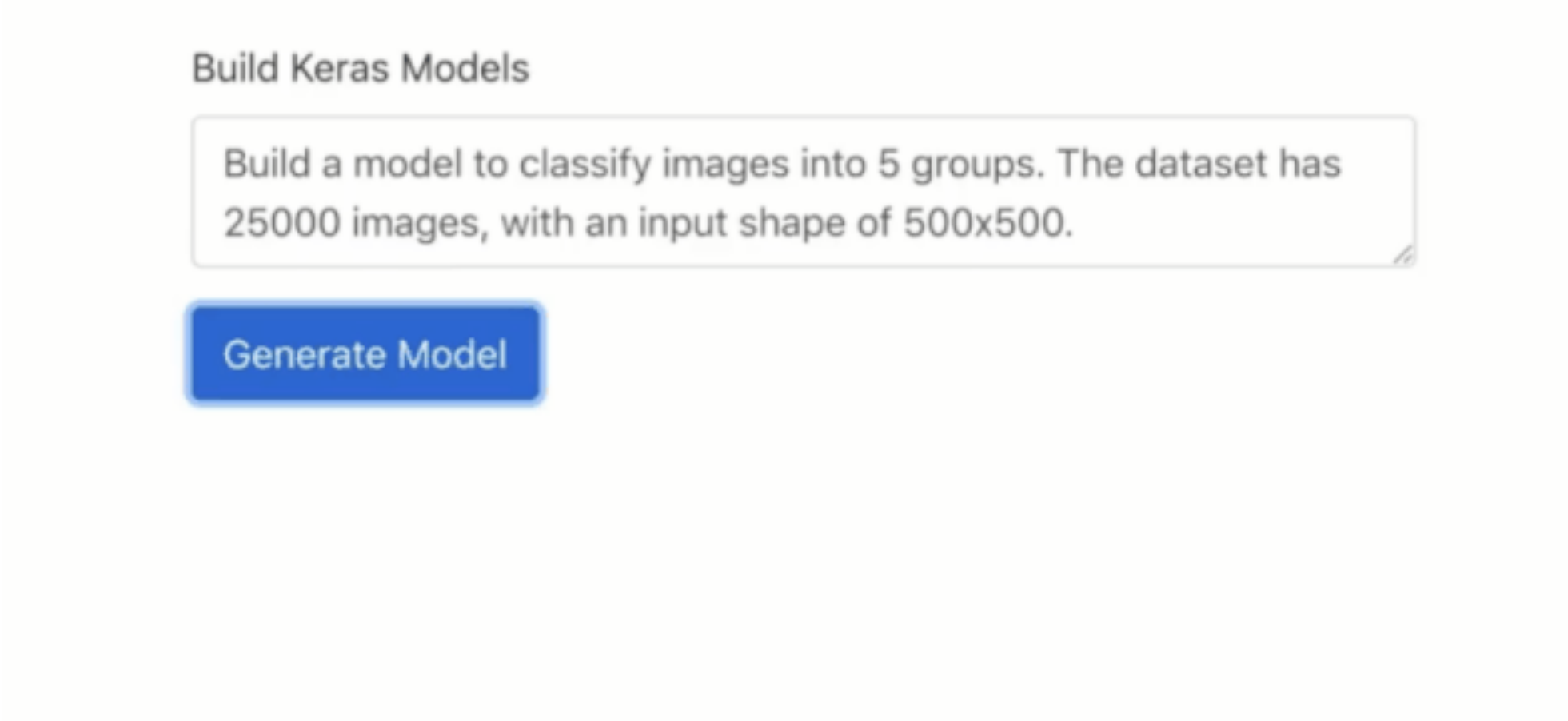
Latex Generation



SQL Generation



Quiz Solution?



Plain Language to Legal Language

Plain language: I don't know who the 10 people really are yet but they are all to blame.
Legal language: Defendants Does 1 through 10 are sued herein under fictitious names. Their true names and capacities are unknown to Plaintiff. When their true names and capacities are ascertained, Plaintiff will amend this complaint by inserting their true names and capacities herein. Plaintiff is informed and believes and thereon alleges that each of the fictitiously named Defendants is responsible in some manner for the occurrences herein alleged, and that Plaintiff's damages as herein alleged were proximately caused by those Defendants.
Plain language: Everyone I'm suing worked together.
Legal language: Plaintiff alleges on information and belief that Defendants, and each of them, are and at all times herein mentioned were the agents and servants of each other and in doing the things hereinafter alleged were acting within the scope of such agency and service.
Plain language: The people I'm suing are my landlords.
Legal language: The Defendants are, in some manner, the owners and lessors of the real property where Plaintiff resides.
Plain language: My landlord entered my apartment without my permission.
Legal language: Defendant entered Plaintiff's dwelling without Plaintiff's consent.
Plain language: My apartment had mold and it made me sick.
Legal language: Plaintiff's dwelling was infested with toxic and allergenic mold spores, and Plaintiff was rendered physically incapable of pursuing his or her usual and customary vocation, occupation, and/or recreation.
Plain language: My landlord told me he didn't want Black people living in the complex.
Legal language: Plaintiff is informed and believes and thereon alleges that the Defendant, by and through his agents and servants, has engaged in a pattern and practice of refusing to rent, and otherwise made unavailable, dwelling units to Black persons because of their race or color.
Plain language: My landlord didn't maintain the property.
Legal language: The Defendants have permitted the real property to fall into disrepair and have failed to comply with state and local health and safety codes and regulations.
Plain language: The apartment was covered by San Francisco rent control.
Legal language: Plaintiff's dwelling was a rent-controlled unit within the meaning of Section 37.2(b) of the San Francisco Rent Ordinance.
Plain language: My landlord didn't return my security deposit in the time allowed by California law.
Legal language: Defendant has failed to refund to Plaintiff all sums of money paid to Defendant as security deposit, within the time periods specified in California Civil Code section 1950.5.

Email Completion

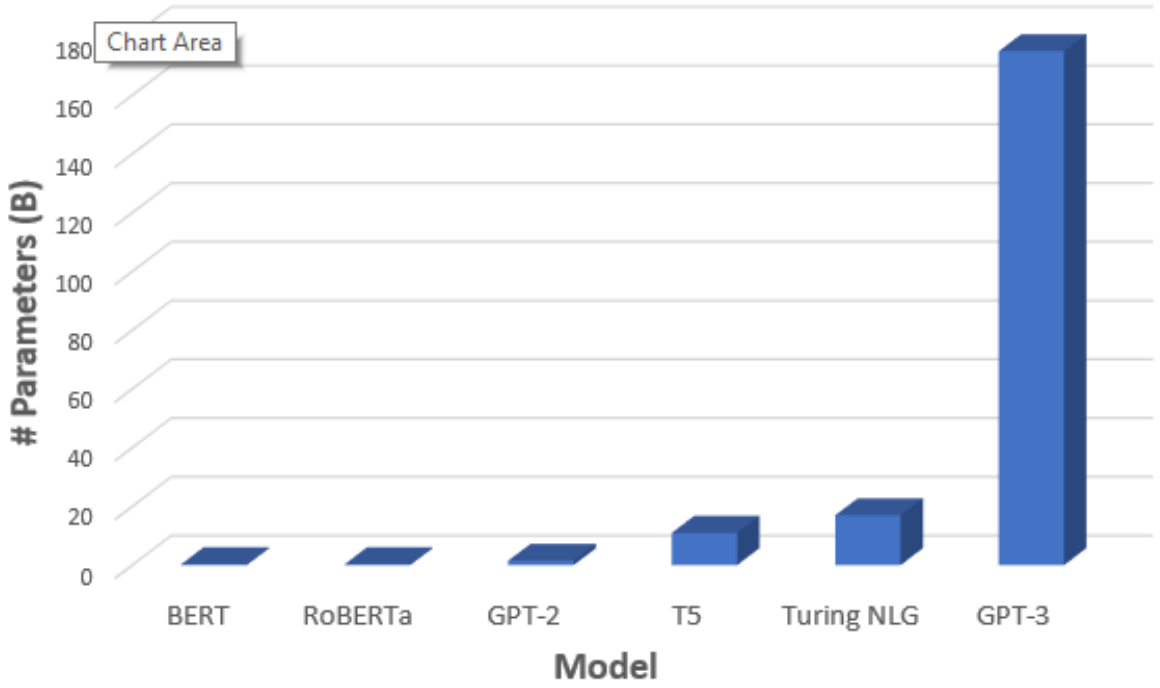
3. Text Processing — GPT-3 Over-estimated

GPT 3 vs BERT

- Unlike BERT models, GPT models are **unidirectional.**
- Major advantage: sheer volume of data were pretrained.

Largest Model Yet

- **175 billion** parameters
  - two orders of magnitude larger than its direct predecessor, GPT-2(1.5 billion parameters).
  - one order of magnitude larger than Microsoft’ s language model, Turing NLG(17 billion parameters， released in February 2020).



GPT vs BERT vs ELMo

GPT: language models are typically left-to-right

$$\text{too}\rightarrow\text{young}\rightarrow\text{too}\rightarrow\text{simple}\rightarrow\text{sometimes}\rightarrow[\text{naive}]$$

ELMO: two unidirectional LSTM(**biLM**)

$$\text{too}\rightleftharpoons\text{young}\rightleftharpoons\text{too}\ [\text{simple}]\ \text{sometimes}\rightleftharpoons\text{naive}$$

BERT: bidirectional transformer

$$\text{too}\Leftrightarrow[\text{mask1}]\Leftrightarrow\text{too}\Leftrightarrow[\text{mask2}]\Leftrightarrow\text{sometimes}\Leftrightarrow\text{naive}$$

3. Text Processing — GPT-3 v.s. BERT

GPT-3: 175 billion parameters——**fine-tuning is no longer necessary**

- **few-shot learning** (getting the model to learn what it needs to do using several training examples)
- **one-shot learning** (using a few or one example)
- **zero-shot learning** (getting the system to extrapolate from its previous training, but using no examples at all).