Data Mining Group Project

Part 3: BERT and Q&A robot

Group4

Teacher: Nicolas TURENNE

1730026058 Alvin LIANG

1730014072 Michael XU

1730026153 Hawthorn ZHAO

Data Science

Table of Contents

1 Motivation and background

In a Data Mining class, we asked Dr. Nicolas TURENNE which topic he was interested in, he said Text Mining. So we chose this. Of course, the emergence of BERT shocked academia and changed the research model of NLP. BERT also swept the score of 11 NLP tasks at the same time, making history. That's why we want to choose Text Mining.

2 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

- Pre-training

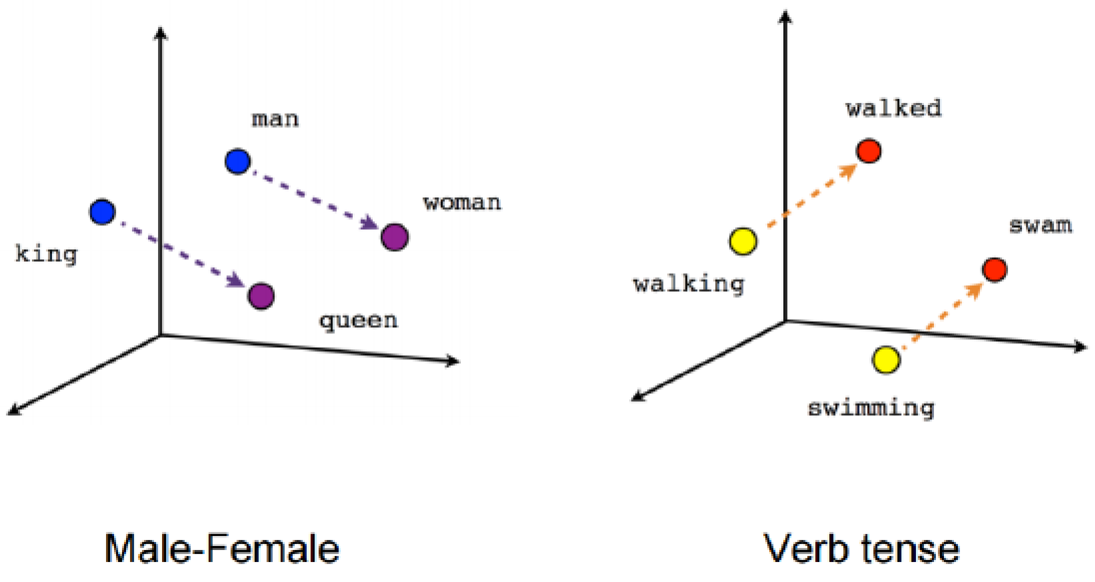
  Dur-ing pre-training, the model is trained on unlabeled data over different pre-training tasks. Learn a language with unsupervised learning

- Fine-tuning

  For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the param-eters are fine-tuned using labeled data from the downstream tasks. Each downstream task has sep-arate fine-tuned models, even though they are ini-tialized with the same pre-trained parameters. The question-answering example in Figure 1 will serve as a running example for this section.

A distinctive feature of BERT is its unified architecture across different tasks. There is mini-mal difference between the pretrained architecture and the final downstream architecture.

3 Review of common technology



3.1 Review: Bag-of-words

This model transforms each document to a fixed-length vector of integers. For example, given the sentences:

- `John likes to watch movies. Mary likes movies too.`
- `John also likes to watch football games. Mary hates football.`

The model outputs the vectors:

- `[1, 2, 1, 1, 2, 1, 1, 0, 0, 0, 0]`
- `[1, 1, 1, 1, 0, 1, 0, 1, 2, 1, 1]`

Each vector has 10 elements, where each element counts the number of times a particular word occurred in the document. The order of elements is arbitrary. In the example above, the order of the elements corresponds to the words: `["John", "likes", "to", "watch", "movies", "Mary", "too", "also", "football", "games", "hates"]`.

Bag-of-words models are surprisingly effective, but have several weaknesses.

First, they lose all information about word order: "John likes Mary" and "Mary likes John" correspond to identical vectors. There is a solution: bag of `n-grams <https://en.wikipedia.org/wiki/N-gram>` __ models consider word phrases of length n to represent documents as fixed-length vectors to capture local word order but suffer from data sparsity and high dimensionality.

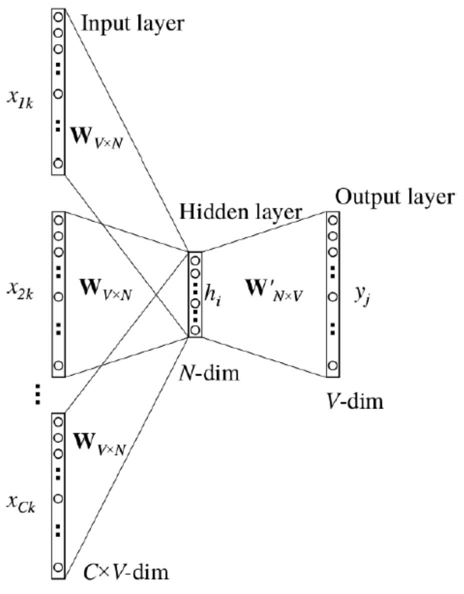
Second, the model does not attempt to learn the meaning of the underlying words, and as a consequence, the distance between vectors doesn't always reflect the difference in meaning. The `Word2Vec` model addresses this second problem.

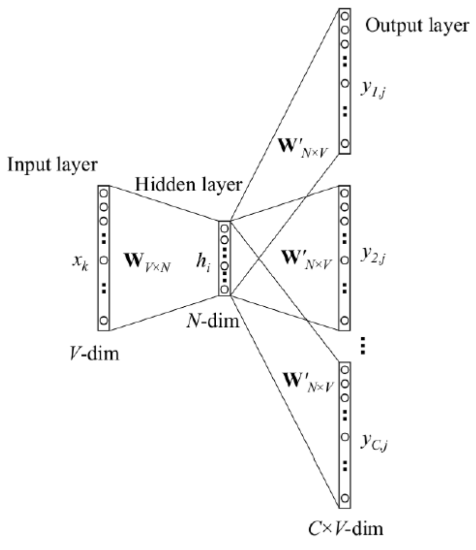
- He *deposited* his *money* in this **bank**.
- His soldiers were arrayed along the *river* **bank**.

The meaning of these two "banks" is not the same.

3.2 Review: `Word2Vec` Model

`Word2Vec` is a more recent model that embeds words in a lower-dimensional vector space using a shallow neural network. The result is a set of word-vectors where vectors close together in vector space have similar meanings based on context, and word-vectors distant to each other have differing meanings. For example, `strong` and `powerful` would be close together and `strong` and `Paris` would be relatively far.





Continuous Bag of Words(CBOW) and Skip-gram

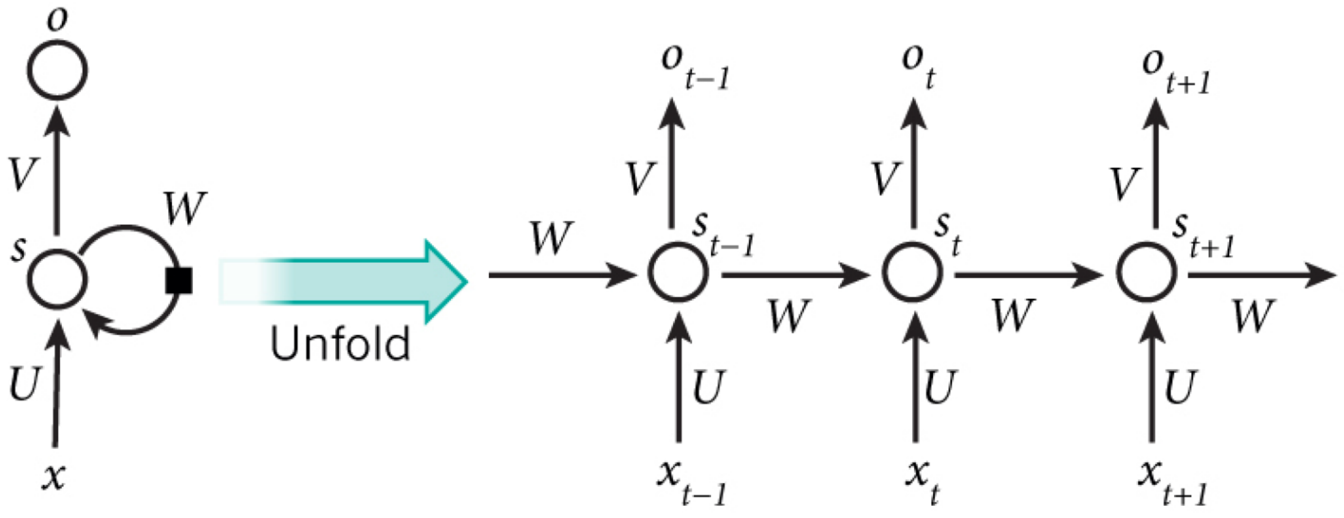
The goal of CBOW is to predict the probability of the current word based on the context.

Skip-gram is just the opposite: the probability of predicting context based on the current word

Both of the methods use artificial neural networks as their classification algorithms. At first, each word was a random N-dimensional vector. After training, the algorithm uses CBOW or Skip-gram methods to obtain the optimal vector for each word.Take a window of appropriate size as the context, read the words in the window in the input layer, and add their vectors (K-dimensional, initial random) together to form K nodes in the hidden layer. The output layer is a huge binary tree, and the leaf nodes represent all the words in the corpus (the corpus contains V independent words, then the binary tree has | V | leaf nodes). The algorithm for constructing the entire binary tree is the `Huffman tree` .

`word2vec` only performs "semantic analysis" based on the dimension of words, and does not have the "semantic analysis" ability of context.

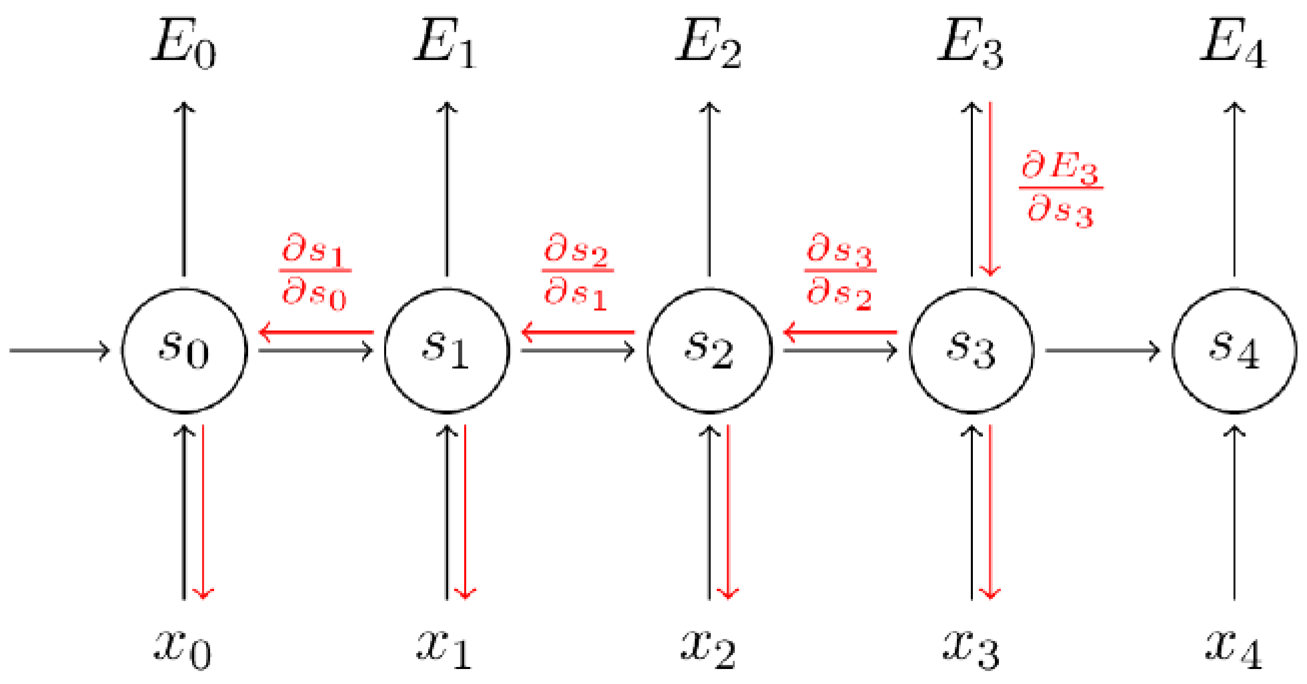
3.3 Review: `Vanilla RNN`



$$s_t = f(Ux_t + Ws_{t-1})$$

RNN has *"memory"* ability

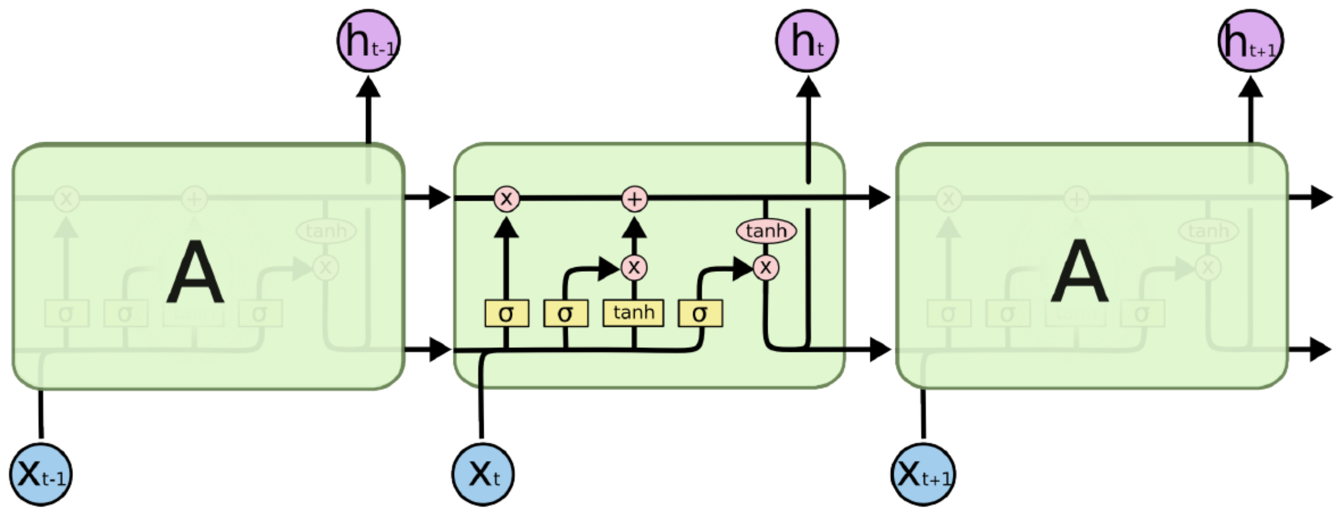
Like the picture, at S_t, not only c onsiderthe input X_t, but also consider the hidden state at S_t-1. It can be said that s_t contains all the semantics from the beginning to s_t. Ability to express in a certain context.



E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4

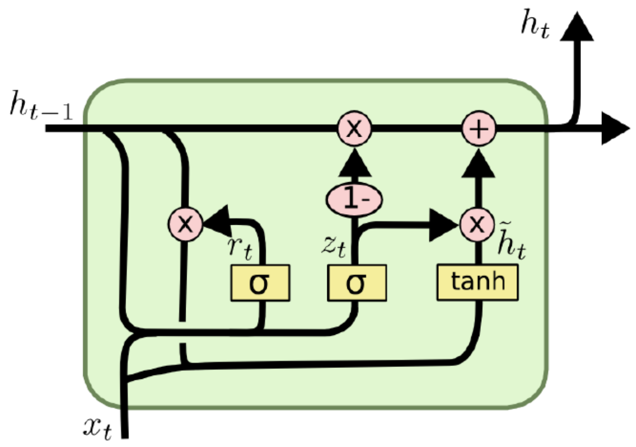
However in neural networks, they are all transmitted by gradient descent, and Loss and Error need to be passed layer by layer. But often after many neurons, these parameters are close to zero, there is a problem of gradient disappearance, and it is difficult to learn strong dependencies.

3.4 Review: LSTM



The first step in LSTM is to decide what information we will discard from the state of the cell. This decision is made through a layer called forget gate. The next step is to determine what new information is stored in the cell state. Then discard the information we determined needs to be discarded. Then get new candidate values, change according to the degree we decide to update each state. Finally, we need to determine what value to output. This output will be based on our cell state, but also a filtered version.

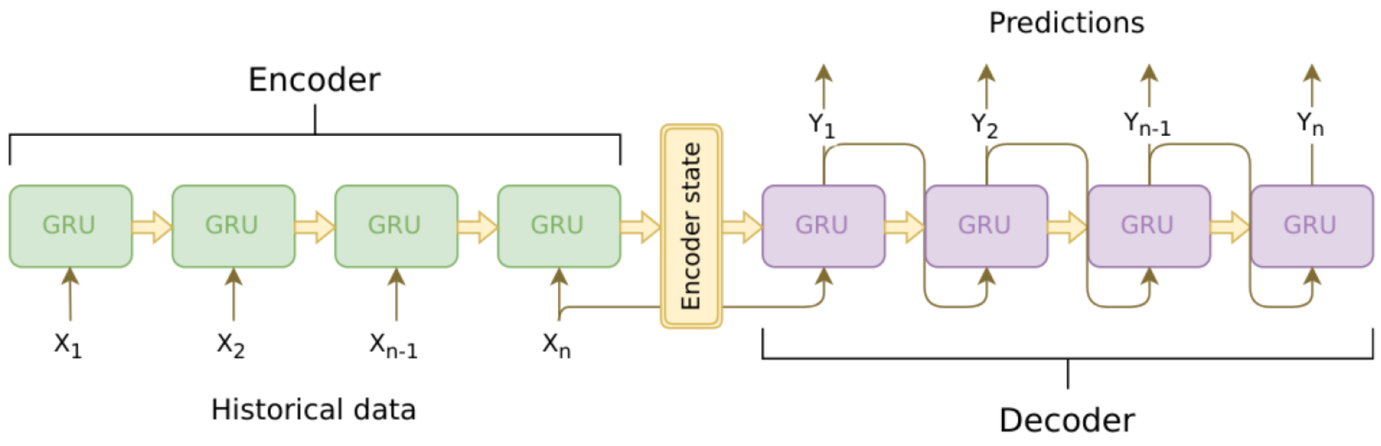
3.5 Review: GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU merges forget gate and input gate into one update gate

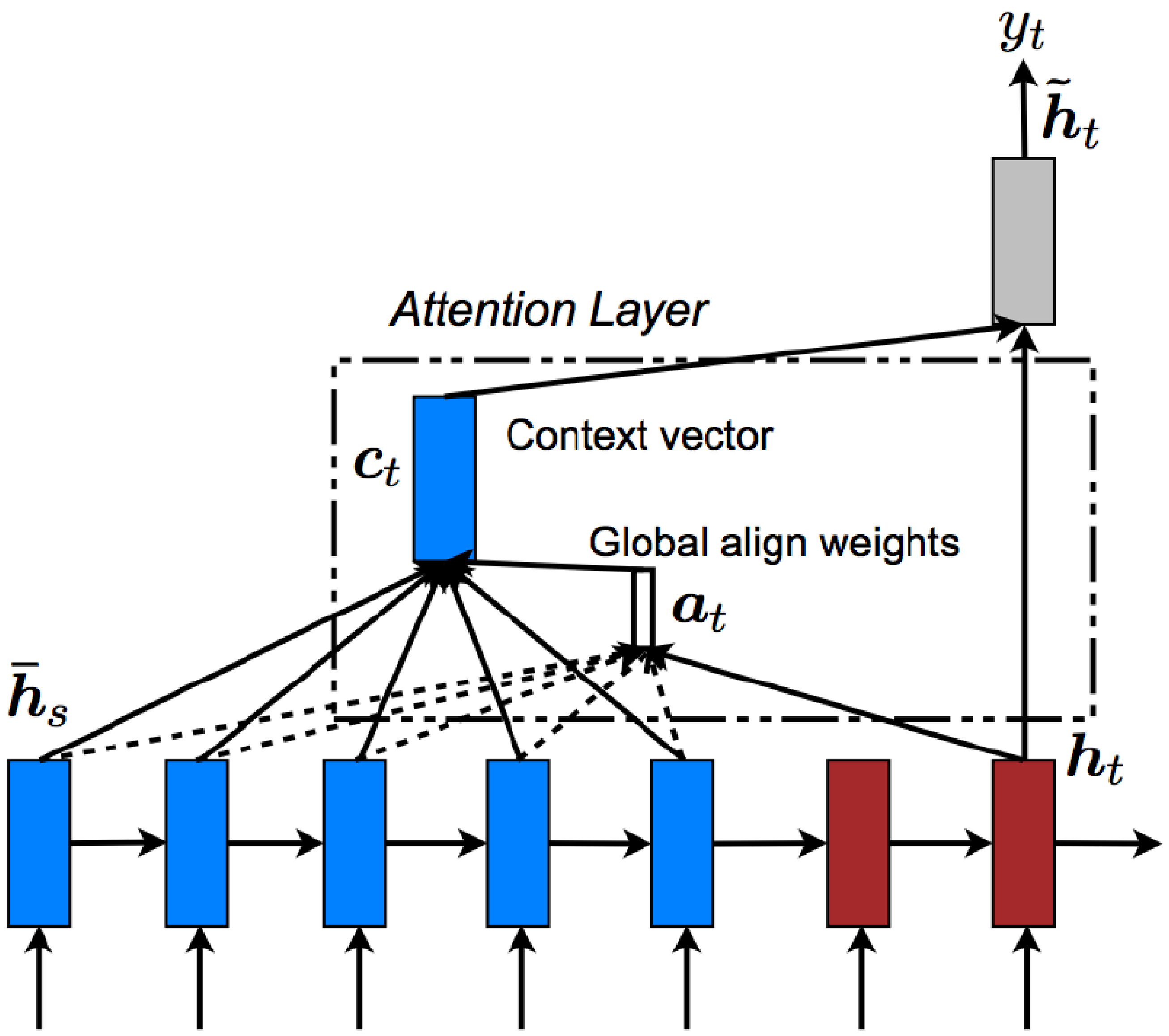
3.6 Review: Seq2Seq



Use two sets of GRUs to form a Seq2Seq , encode the input into a context vector, use this context vector to encode the semantics of the entire sentence, and then use the decoder to decode it. When predicting, the output of the first moment is used as the input of the second moment, of course, it will also bring the semantic state of the first moment. Until the last word.

But this is actually very difficult. If you use a context vector to encode all the semantics of a sentence. For example, if a human reads a 100-word sentence, you can remember the approximate meaning of the entire sentence, but you cannot remember every word and every detail.

3.7 Review: Attention



Attention can be said to be "pay attention to", related words when translating a certain word. In the figure, the blue block is equivalent to a sentence word, and each word is encoded into a vector by RNN. The red block is equivalent to Decoder. For example, h_t is the semantic we want to translate, but h_t may be a "rough meaning" rather than a "detailed meaning". So when we translate this h_t, what do we do when we only know a "rough meaning"? At this time, when we translate h_t, we must use h_t to accumulate all the semantics(all blue blocks), and then get the weight. Of course, this weight may be negative infinity to positive infinity, so we need a softmax function to turn it into a 0-1 probability. This can be seen as a way to extract information.

The essence of the attention mechanism is to get inspiration from the human visual attention mechanism (it can be said that it is 'people-oriented'). Roughly, when our vision perceives things, it is generally not a scene that looks at everything from beginning to end, but often it pays attention to a specific part according to the needs. And when we find that a scene often shows what we want to observe in a certain part, we will learn to focus on that part when similar scenes occur in the future. This can be said to be the essence of the attention mechanism. The Attention mechanism is actually a series of attention distribution coefficients, that is, a series of weight parameters.

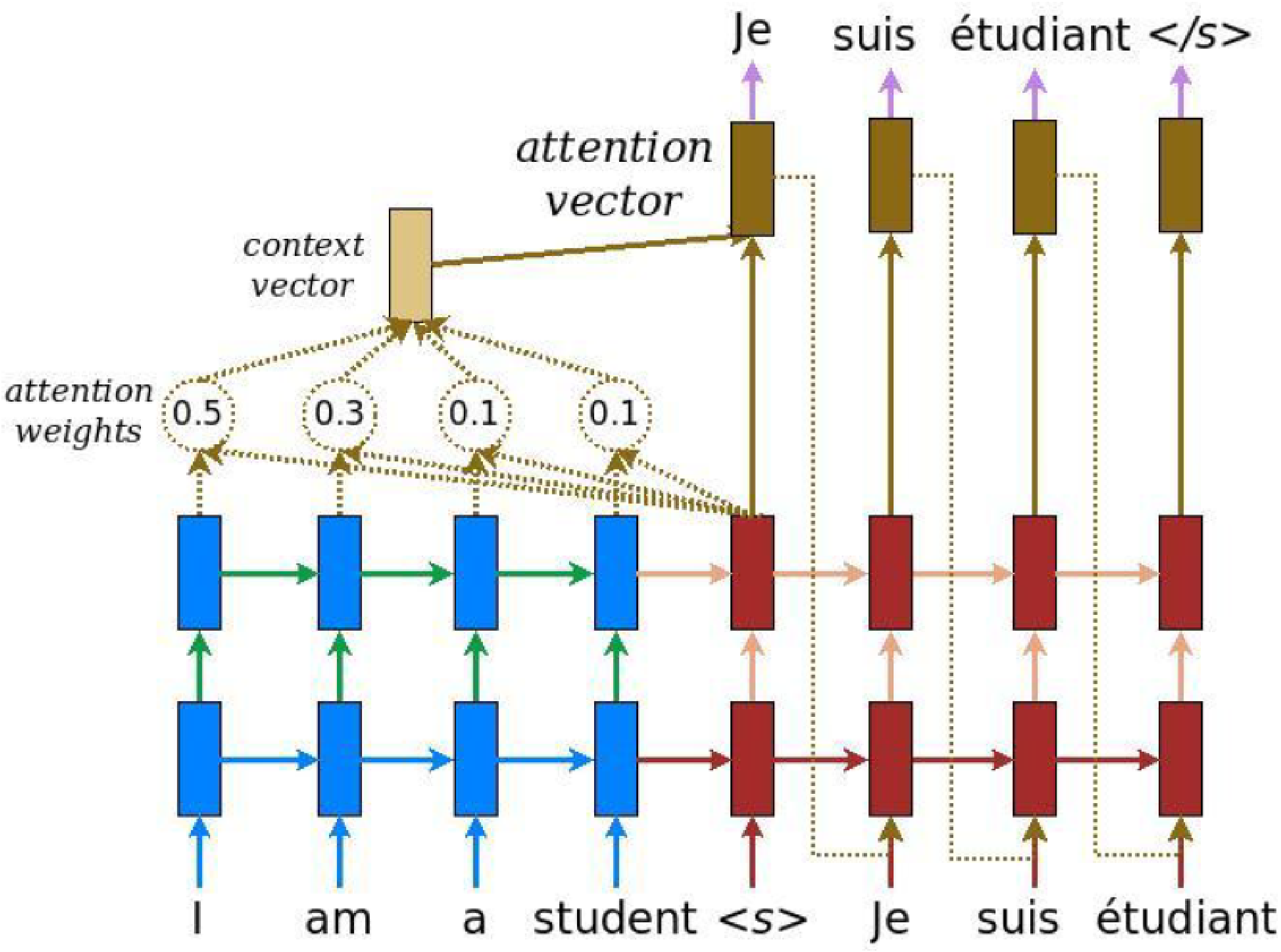
3.8 self-Attention

The above models are all based on RNN classes, among which there are still these problems:

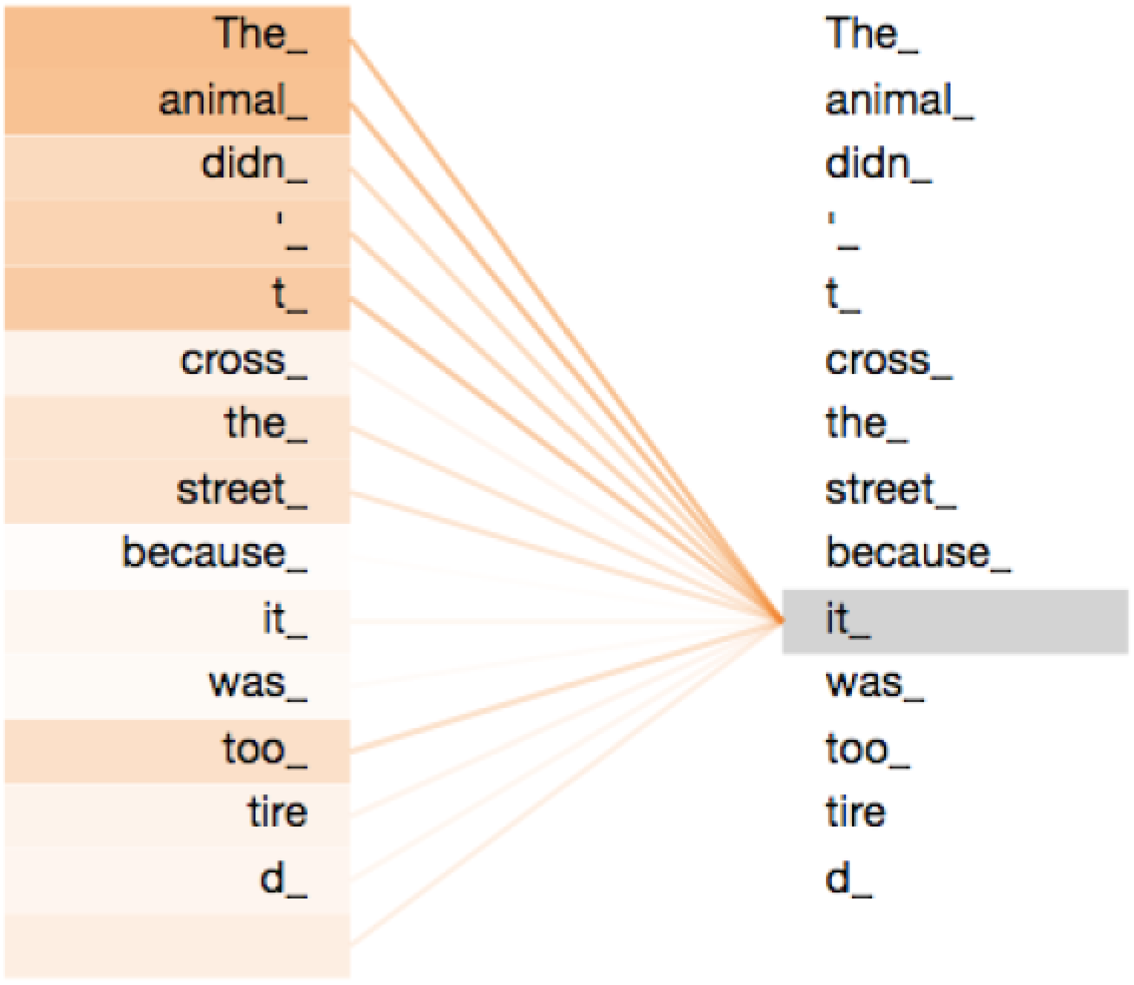
1. It cannot operate in parallel. Just like the RNN figure, you must calculate S_t-1 before you can calculate S_t. In this case, the training speed of the RNN is very slow, and it is difficult to train RNNs with many layers.
2. The problem of unidirectional information flow
    - The animal didn't cross the street because it was too tired.
        - `it` stands for animal, **animal is too tired.**
    - The animal didn't cross the street because it was too narrow.
        - `it` stands for street, **street is too narrow.**

```
- The animal didn't cross the street because `it?`
        - What is `it` stand for?
- `it?` was too tired.
        - What is `it` stand for?
```

In general, when we translate a sentence, it is not enough to just look at the front or back of the sentence, we must refer to the entire sentence.

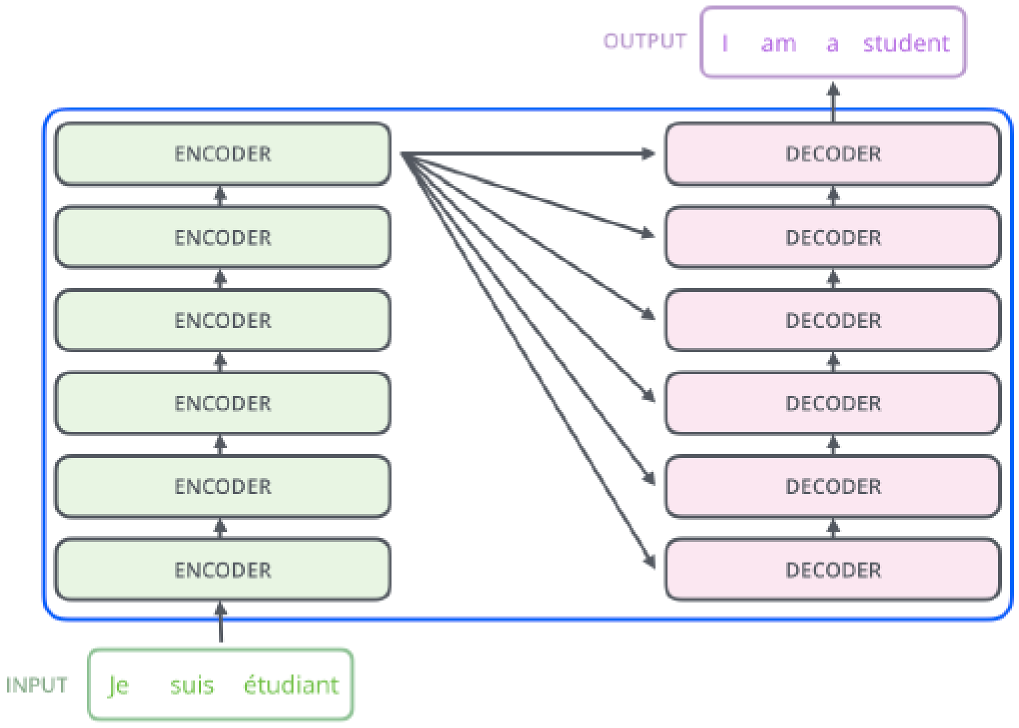


As mentioned earlier, Attention requires external driving and gives corresponding weight.

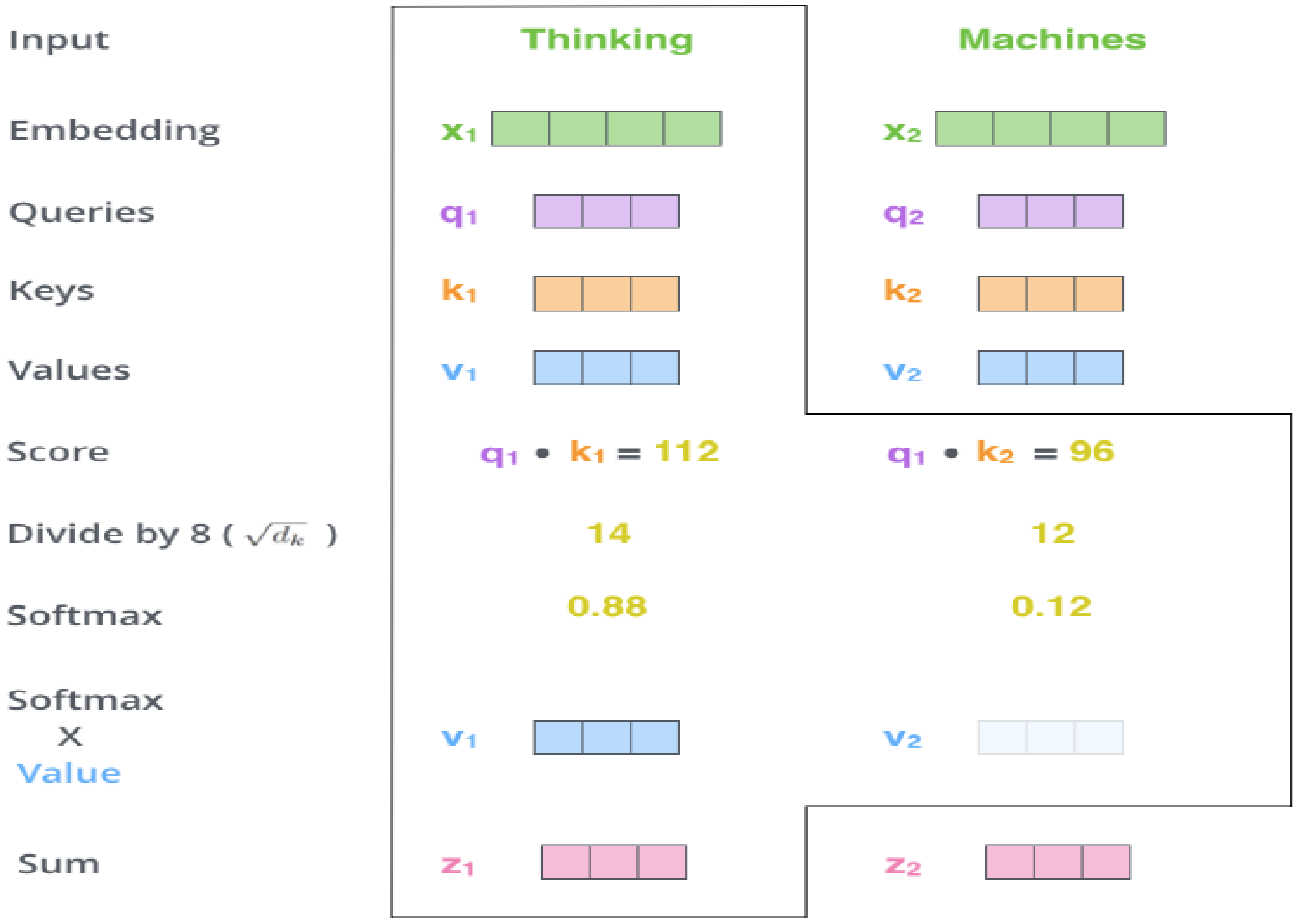


Self-attention will drive itself, for example, `it` is a pronoun, then it will look for nouns by itself. May be `animal` or `street` .

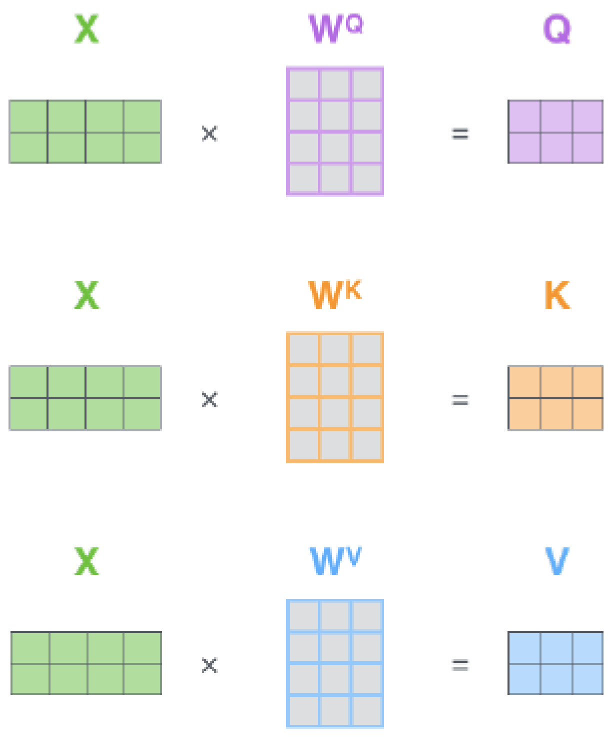
3.9 Transformer

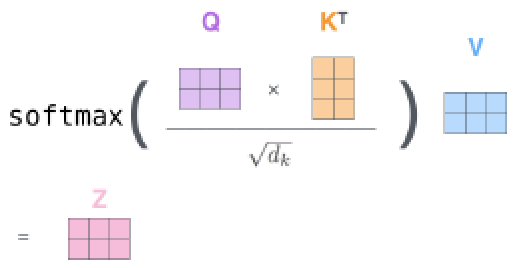


Multilayer Encoder-Decoder



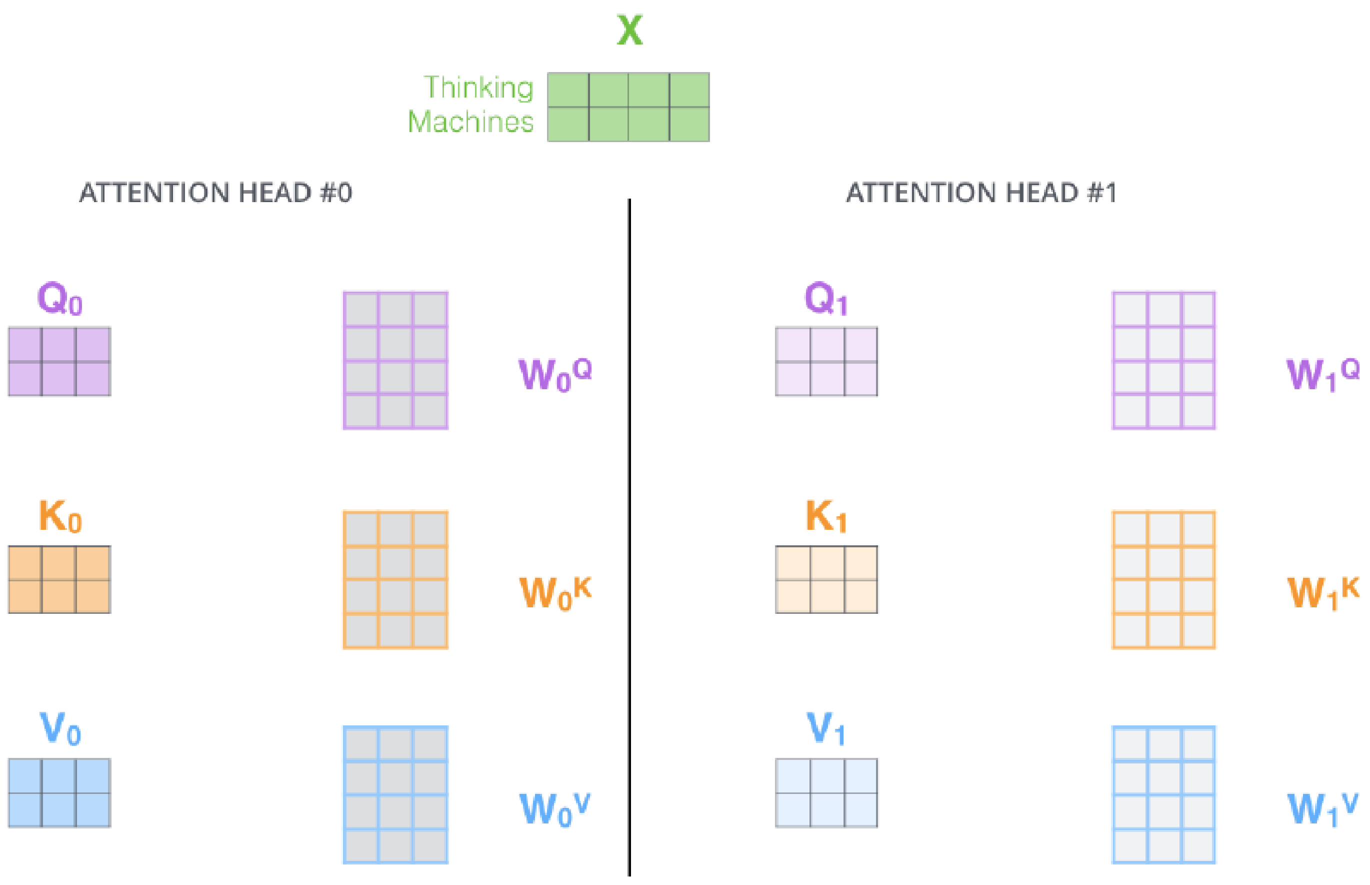
Among them, like X_1, if he is a word `bank` , it may encode a lot of its semantics. q can be seen as a vector used by this word to query other words. k can be seen as a vector used by others when querying itselves. Score refers to how much effort it takes to look at the word. For example, word `thinking` , it needs to focus 112 on the first word and 96 on the second. Finally, calculate v, and add up to sum.



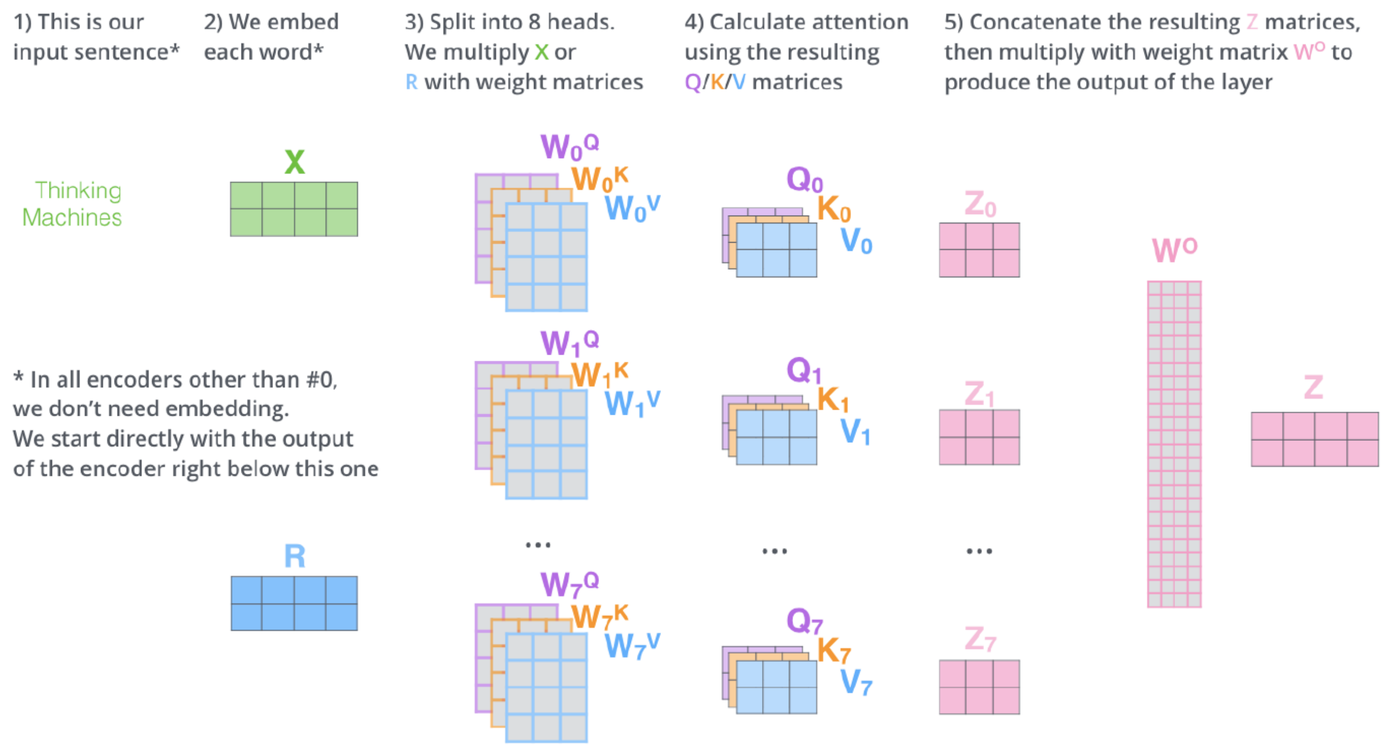

$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$

Unlike `RNN` , `transformers` can be calculated in parallel, and all the matrices are calculated at once, which is very fast.

- Multi-Heads



For example, a set of matrices is used to learn how to use `it` as a pronoun, another set of matrices is used to observe the upper and lower positions, and another set of matrices is the correspondence between the capital and the country. Different heads can learn different semantic relationships.


1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
R
W0Q W0K W0V
W1Q W1K W1V
W7Q W7K W7V
Q0 K0 V0
Q1 K1 V1
Q7 K7 V7
Z0
Z1
Z7
WO
Z

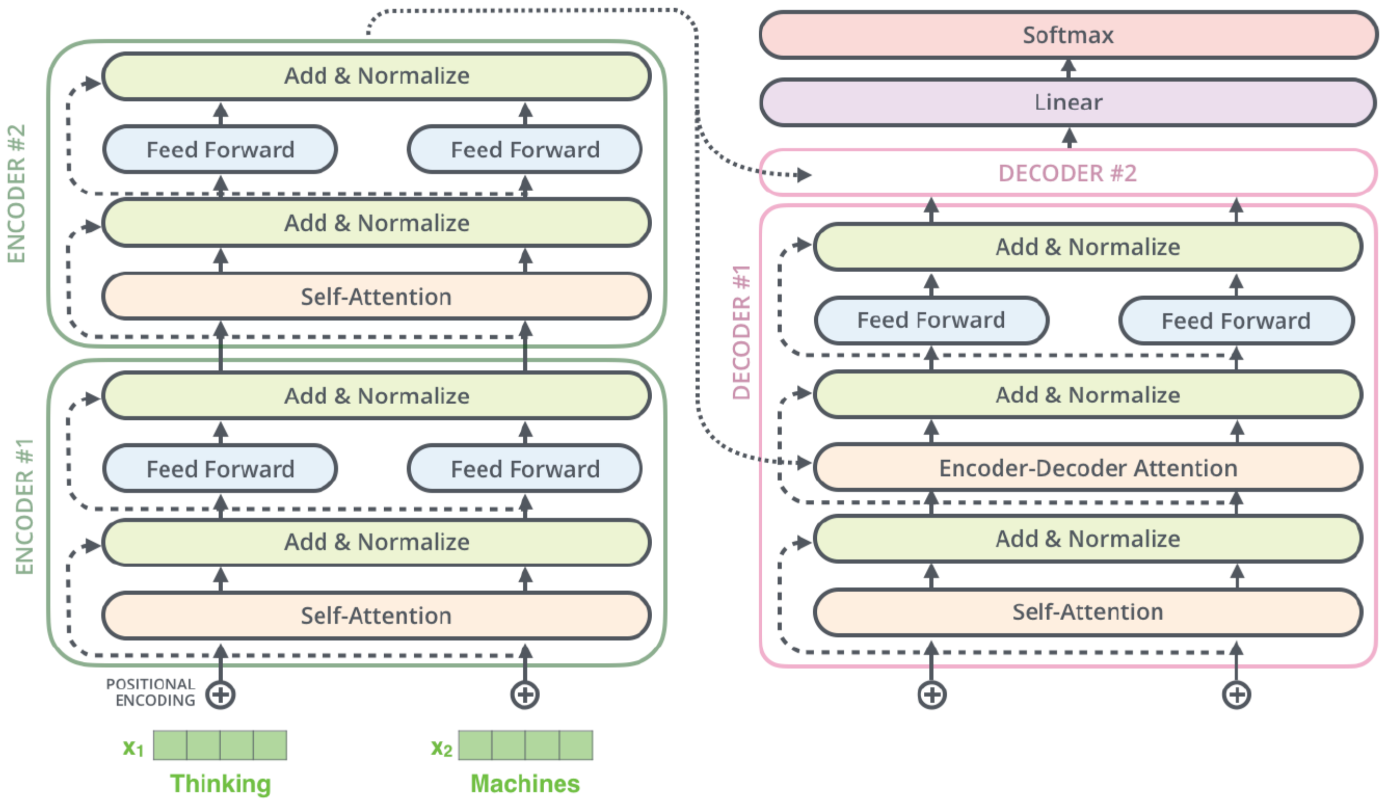
The picture above is the complete process

- Position coding

- Air tickets from `Guangzhou` to **Zhuhai**.
- Air tickets from `Zhuhai` to **Guangzhou**.

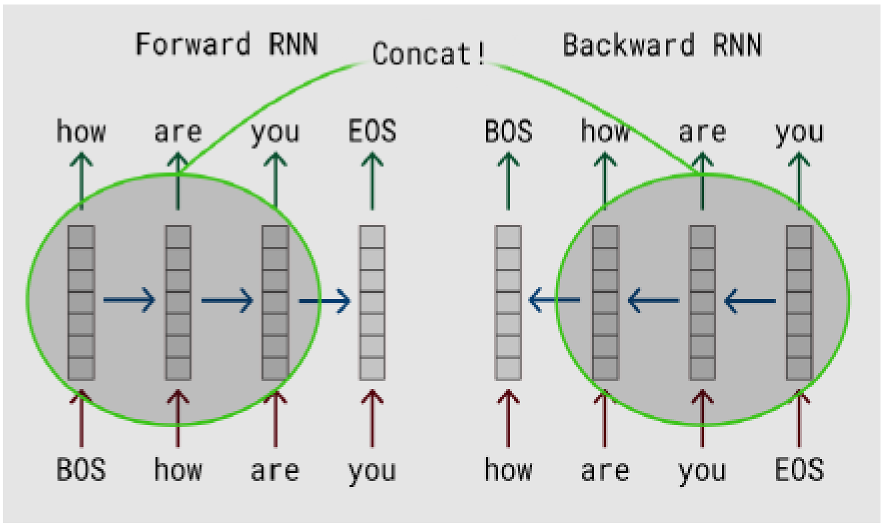
In traditional self-attention, there is no position encoding. In simple terms, the matrix compiled by `Guangzhou` in these two sentences is the same.

BERT adds position coding



ENCODER # 1 on the left is a complete encoder, adding Self-attention and a fully connected layer, residual connection, etc. Decoder is similar to Encoder, except that he added an `Encoder-Decoder Attention` . So in addition to its information flow from top to bottom, the output of the encoder will also be introduced to attention.

3.10 ELMo



$$ELMo_k^{task} = E(R_k; \Theta_{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{kj}^{LM}$$

ELMo uses multilayer bidirectional LSTM.

ELMo uses `contextual word embedding` as a feature, but context is learned through unsupervised corpus. Although it is still possible to learn something, there are still some differences from the corpus of real specific tasks. The context learned may not be appropriate for the specific task.

3.11 `OpenAI GPT`

- According to the mission `Fine-Tuning`
- Use Transformer instead of RNN / LSTM
- ...

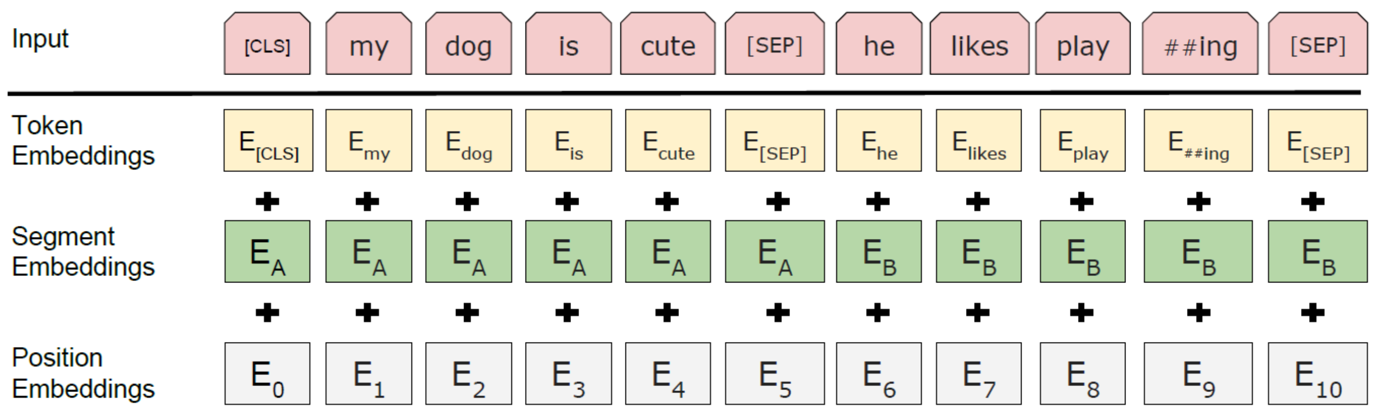
But...

- Sentences are masked, and you can only "look at" the front of the current word in the sentence. Can't "look at" the back.
- Pretraining does not match Fine-Tuning.

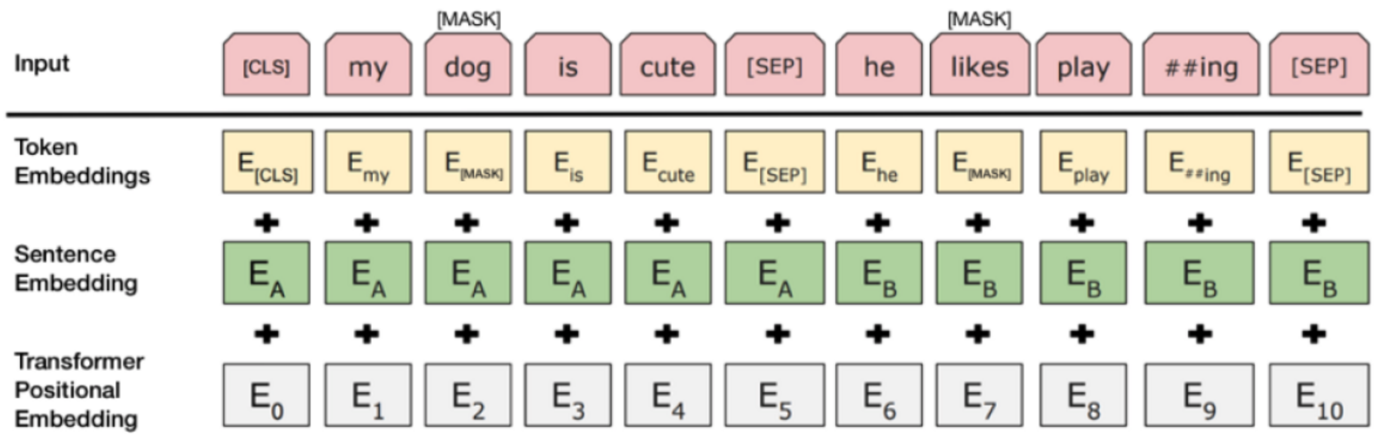
4 BERT

How to solve openAI GPT's problems?

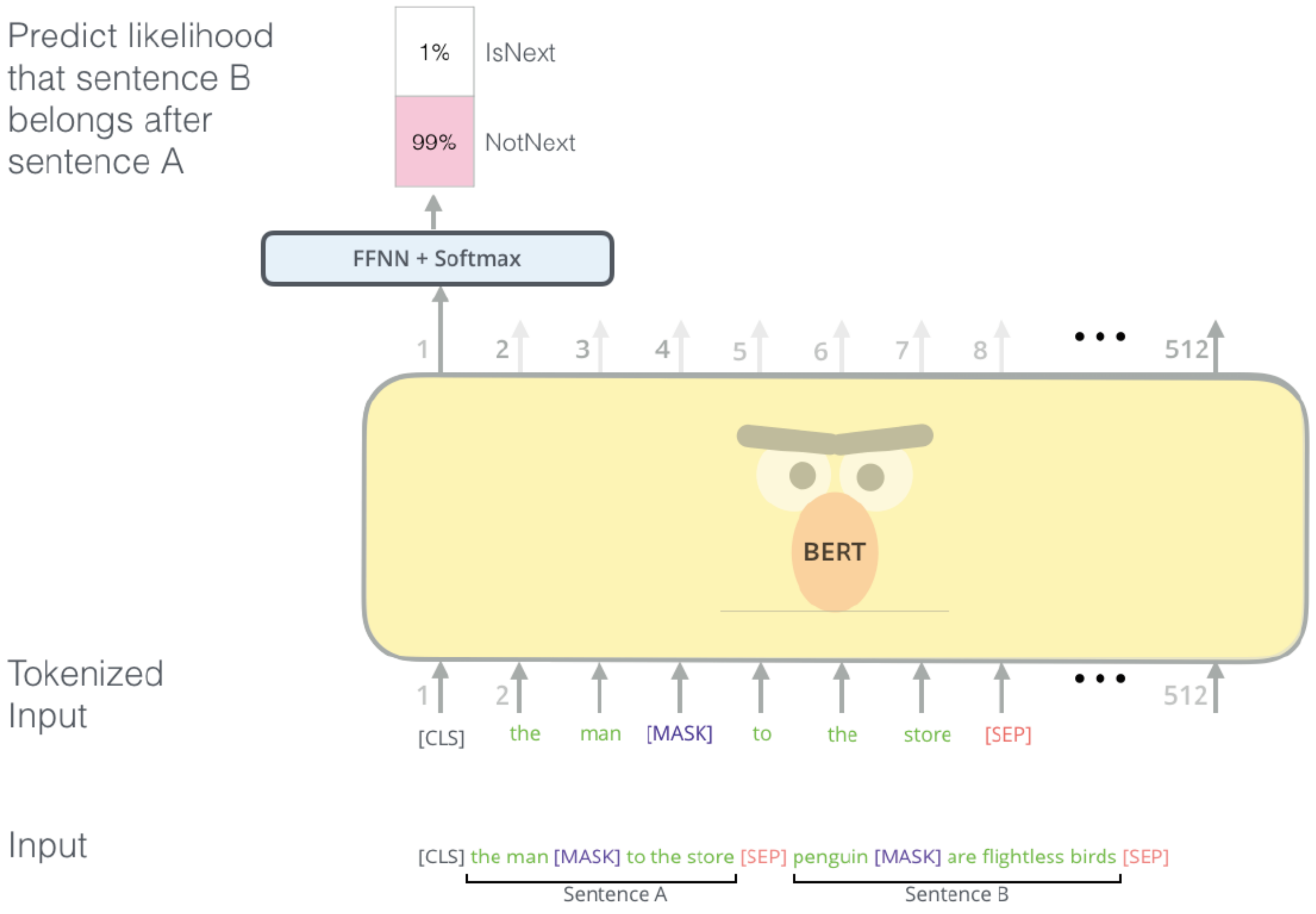
- Masked language model [Solving one-way information flow]
- Next sentence predict Multi-task Learning
- Encoder again


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

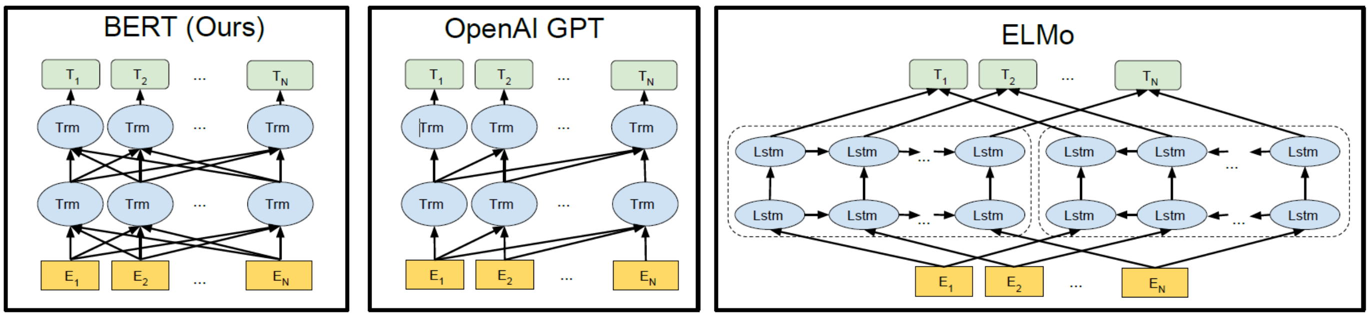
Use [CLS] to indicate the beginning of a sentence, and [SEP] to split the sentence. It use token embedding, segment embedding and position embedding.


[MASK]
[MASK]
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

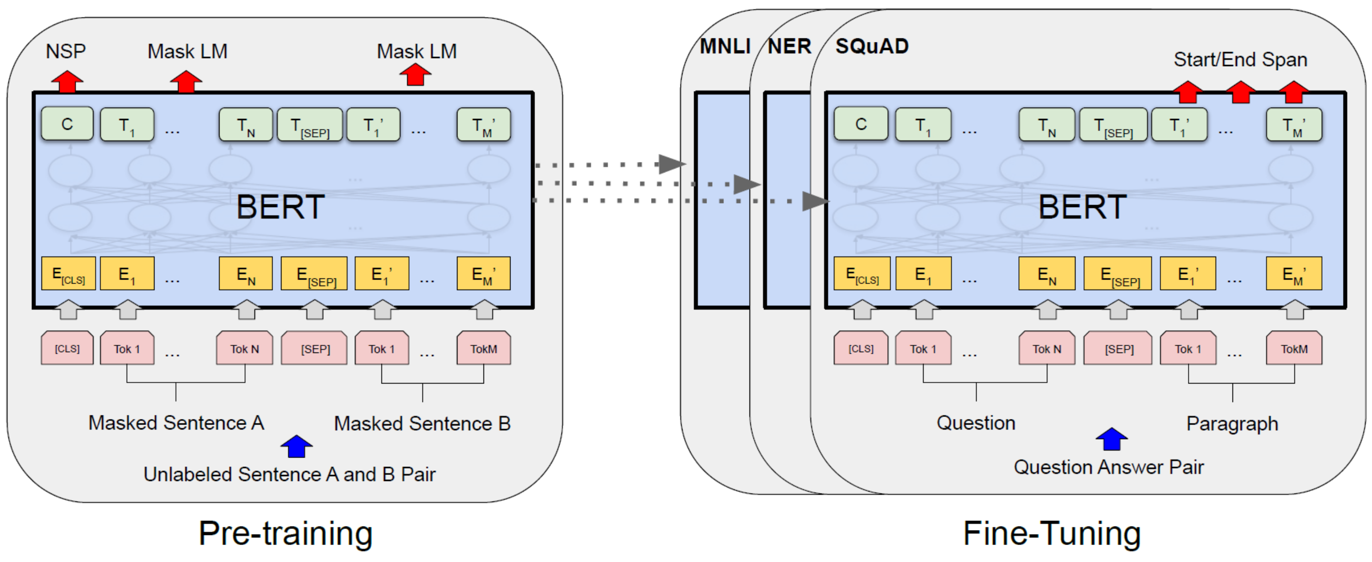
- Masked LM
  - Similar to cloze, random mask 15% of words, let Bert predict.


Predict likelihood that sentence B belongs after sentence A
1%
IsNext
99%
NotNext
FFNN + Softmax
1
2
3
4
5
6
7
8
512
BERT
Tokenized Input
[CLS]
the
man
[MASK]
to
the
store
[SEP]
Input
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]
Sentence A
Sentence B

- Introduce predictive sentence relationships to resolve the mismatch between Pre-traning and Fine-tuning



Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-toleft LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.



Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architec-tures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initializemodels for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a specialsymbol added in front of every input example, and [SEP] is a special separator token (e.g. separating ques-tions/answers).

For example, text classification and sentiment classification, we enter a sentence, first come first with BERT, encoding a lot of vectors. Then a fully connected layer is connected to the first vector to make a classification.

5 A simple Q&A robot

BERT can do many things, such as translation, sentiment analysis, question answering robot, etc.

The following is a Q & A robot based on the common questions of UIC admissions.

- Similarity calculation(like KNN)
- Intent classification

There are two modes, here we use the first mode.

In [1]:

```
import numpy as np
import pandas as pd
from bert_serving.client import BertClient
from termcolor import colored
# Import packages
```

executed in 472ms, finished 01:43:24 2019-12-23

In [2]:

```
bc = BertClient()
# Start bert client
```

executed in 106ms, finished 01:43:24 2019-12-23

In [3]:

```
# Encode the sentence
bc.encode(['I love UIC ds!']).shape
```

executed in 263ms, finished 01:43:25 2019-12-23

Out[3]:

```
(1, 768)
```

In [4]:

```
bc.encode(['I love UIC ds!'])
```

executed in 447ms, finished 01:43:25 2019-12-23

Out[4]:

```
array([[ 3.28549623e-01,  2.27723256e-01, -2.75719818e-02,
        -2.92795420e-01,  1.72237918e-01, -8.43800247e-01,
         5.86637139e-01, -4.82229710e-01, -1.36713862e-01,
         4.29402828e-01, -2.39786774e-01,  4.77168500e-01,
         1.98632516e-02, -4.04778957e-01,  1.32556212e+00,
        -4.47109729e-01,  3.39814752e-01,  1.39031231e-01,
        -2.32591927e-01,  1.28316090e-01,  4.22742546e-01,
         2.88237929e-01, -1.97668329e-01,  2.39096090e-01,
         7.38043547e-01,  2.52148449e-01, -3.10648948e-01,
         3.40460092e-02,  5.71894705e-01, -1.22017853e-01,
        -3.72085392e-01, -1.30522504e-01,  2.86171764e-01,
         4.44296688e-01, -1.37933359e-01,  3.02929431e-01,
         5.31697333e-01,  4.74559814e-01,  6.57850970e-03,
         1.18397474e-01,  3.88419956e-01, -2.32772186e-01,
        -3.19971442e-01,  8.37458670e-02,  2.16890037e-01,
         1.74603790e-01,  4.08070773e-01,  1.58761263e-01,
         2.19197273e-01,  4.33187746e-02,  4.46505696e-01,
         1.13296099e+01,  5.44061065e-01,  2.07740650e-01,
        -6.73959255e-02,  8.06004778e-02,  7.22105563e-01,
         2.70791501e-01, -4.42948118e-02, -3.20637017e-01,
        -2.20540568e-01, -4.74329472e-01,  1.41671762e-01,
         3.56895834e-01,  5.73533118e-01, -5.01369178e-01,
         4.62637782e-01, -3.42117220e-01,  2.07744643e-01,
        -4.52854425e-01,  1.56224936e-01,  1.06764361e-01,
         2.89475489e-02, -2.90381044e-01,  4.06534076e-01,
         2.50085413e-01, -4.46307123e-01,  3.27581167e-01,
        -4.17028606e-01, -9.46500450e-02, -1.03971064e-01,
         1.90162972e-01, -5.19163251e-01,  8.55259821e-02,
        -4.32216465e-01, -3.50687563e-01, -9.15935934e-02,
         3.66803035e-02,  5.95978200e-01,  4.88219559e-01,
         1.02233656e-01,  2.37998486e-01, -4.48038101e-01,
         7.12403715e-01,  1.42301947e-01, -4.82749790e-02,
         1.67693615e-01,  5.84440231e-01,  2.08570987e-01,
        -1.11731037e-01,  3.47616374e-02,  2.30510905e-01,
        -1.82862014e-01,  1.37807969e-02,  1.38918176e-01,
         1.61982083e+00,  5.40993512e-01, -2.79009193e-01,
        -3.01001430e-01, -3.53738099e-01, -4.93926406e-01,
         6.58815682e-01, -6.19612157e-01, -1.12212636e-01,
        -6.37638196e-02,  1.89625010e-01,  1.73202977e-01,
         5.47646403e-01,  5.99820316e-01, -3.86773765e-01,
         6.13597035e-01,  2.88248956e-01, -7.21784830e-02,
         5.50119460e-01,  3.83267067e-02, -1.03008360e-01,
         1.00366190e-01, -6.57688558e-01, -2.30412275e-01,
        -1.82446569e-01, -2.14210242e-01,  2.13708077e-02,
         1.92601323e-01, -4.21557218e-01,  1.40106365e-01,
        -3.09527904e-01, -4.05751690e-02,  2.19185561e-01,
        -3.66068602e-01, -1.69022113e-01, -2.00766891e-01,
        -3.49453062e-01, -6.05925858e-01, -6.25327006e-02,
         1.01104558e-01,  2.24908784e-01,  3.65139693e-01,
        -5.66324368e-02,  5.85523322e-02,  1.25852838e-01,
        -1.14241898e-01,  1.73904002e-02,  1.26608744e-01,
        -3.92084755e-03,  1.82298198e-01, -2.38636076e-01,
        -5.96504509e-01, -3.72820467e-01,  1.49434224e-01,
         4.03284609e-01, -4.25227344e-01,  2.01483205e-01,
         4.09208387e-01, -3.75849307e-01,  2.59342849e-01,
        -6.52602971e-01,  5.28280914e-01,  1.12749875e-01,
        -1.55065954e-01,  3.26563984e-01, -5.86794972e-01,
         3.09750468e-01,  3.53835016e-01,  2.75833189e-01,
        -2.19006389e-01,  3.05192359e-02, -7.51397848e-01,
         5.26784420e-01, -8.42048973e-02, -6.69605657e-02,
        -2.33093023e-01,  8.61496627e-02,  2.40022212e-01,
         2.17204869e-01,  1.28764063e-01, -1.65950045e-01,
        -2.08552212e-01, -2.82558829e-01,  3.02110314e-01,
         1.91108644e-01, -1.09476373e-01,  3.46561253e-01,
        -2.46590860e-02, -5.65864563e-01,  5.68153501e-01,
         5.14347017e-01, -1.35222897e-01, -1.83513820e-01,
        -7.38913640e-02, -1.71567798e-01, -2.22591713e-01,
        -5.21128960e-02, -4.07989353e-01,  8.49345699e-02,
        -2.49075145e-01,  6.75192662e-03, -4.19181705e-01,
         2.66924024e-01,  2.85524845e-01,  2.46724591e-01,
         7.20366761e-02,  4.25197989e-01, -1.08135454e-01,
         4.94723581e-02, -1.23918690e-01,  1.50335521e-01,
         1.20935701e-01, -3.20054591e-04,  8.57920796e-02,
        -3.77259135e-01,  6.51447415e-01, -2.77201563e-01,
        -7.32878223e-02, -1.84300199e-01, -1.59692869e-01,
         1.01989798e-01,  1.26649380e-01,  1.76367223e-01,
        -4.86593097e-01,  1.73106253e-01, -3.00621808e-01,
         5.83416484e-02,  3.71458024e-01,  8.20653021e-01,
        -2.22030327e-01,  7.76889250e-02, -5.10151386e-01,
         7.21124113e-02,  6.36578083e-01, -1.25402644e-01,
        -1.31336227e-01, -7.66628981e-02,  1.40042916e-01,
         9.21094865e-02,  5.65674126e-01,  4.76570316e-02,
         3.68906796e-01,  1.03326455e-01,  1.44267023e-01,
         4.22425002e-01, -1.99510992e-01,  3.49605083e-01,
         2.27927059e-01, -4.77794856e-01, -8.47362280e-02,
        -1.42981663e-01,  4.46568102e-01, -2.74389964e-02,
         2.69596428e-01, -1.13093041e-01,  1.75375223e-01,
        -8.73007402e-02, -7.25596845e-01, -2.52320737e-01,
         7.66266167e-01,  4.11202669e-01, -4.22305390e-02,
        -4.76828516e-01,  5.12913644e-01,  4.50352669e-01,
        -6.41672686e-02, -2.69682519e-03, -2.27311149e-01,
         4.17348780e-02,  5.91454692e-02, -2.27780953e-01,
        -1.55739766e-02,  3.91678035e-01,  2.58432388e-01,
        -4.01400588e-03,  1.11546002e-01,  3.49253118e-01,
        -3.35335769e-02, -6.44968897e-02,  6.33937299e-01,
        -3.29468191e-01, -1.45871878e-01, -2.03033119e-01,
         9.77446958e-02, -3.36466521e-01, -8.08516502e-01,
        -2.66384810e-01,  6.58659711e-02, -5.49935818e-01,
         3.45297337e+00,  4.34985235e-02,  1.76466346e-01,
        -1.32800847e-01,  1.38083756e-01, -8.16208776e-03,
        -3.05661500e-01,  1.72746658e-01, -3.36983353e-01,
        -4.02056247e-01,  7.09539056e-01, -1.64167792e-01,
         6.53635859e-02,  7.48579875e-02, -2.30351344e-01,
         1.46811053e-01, -3.86966646e-01, -5.84334955e-02,
         2.04716250e-01, -3.09912145e-01,  1.61669925e-01,
        -1.85244411e-01, -1.91433206e-01, -6.38241768e-01,
        -2.01692715e-01,  9.28444490e-02, -3.77047986e-01,
        -3.98688376e-01,  1.06628753e-01,  1.18738338e-01,
        -4.76340145e-01,  2.61432081e-01,  4.85472500e-01,
        -1.73364356e-01,  1.40869334e-01,  4.24290970e-02,
        -4.28669937e-02,  4.43771690e-01, -5.09139180e-01,
         1.88167274e-01,  2.65893161e-01,  4.84403819e-01,
         1.18716322e-01, -2.20032290e-01, -1.94402233e-01,
        -4.91736531e-02,  1.92710251e-01, -8.56739402e-01,
        -4.43858027e-01, -4.00264949e-01, -5.27888119e-01,
        -2.83466190e-01,  2.65673418e-02, -2.87437052e-01,
         2.24175602e-01, -7.53489137e-02, -6.69193268e-02,
         1.83154210e-01, -2.59288661e-02, -4.09272671e-01,
        -4.09531482e-02, -3.29131037e-01, -2.94846117e-01,
         2.97739506e-01,  6.07821718e-02,  3.40939984e-02,
         6.54370070e-01,  2.62647003e-01, -1.71577886e-01,
        -4.64521497e-02,  1.38814542e-02,  2.51465440e-01,
        -7.24549517e-02, -1.79207414e-01, -1.43724710e-01,
        -1.60570800e-01,  1.61282837e-01, -2.09994495e-01,
         3.57493162e-01,  5.02839446e-01, -7.05890730e-02,
         2.49251410e-01,  3.95884588e-02, -2.96806455e-01,
         2.65842825e-01, -1.04167961e-01, -2.32111126e-01,
        -2.43378893e-01, -3.93210381e-01, -1.69128150e-01,
         1.60781726e-01,  1.72875270e-01,  1.94405913e-01,
        -7.95987010e-01,  1.02050647e-01, -4.78948474e-01,
         8.60799104e-03,  4.41793978e-01, -4.29276258e-01,
         1.89361021e-01, -2.80394644e-01, -9.63330343e-02,
         4.43745106e-02, -3.84687543e-01,  1.56413689e-01,
        -5.09506285e-01,  1.54464051e-01,  9.22494680e-02,
        -8.64794254e-02,  2.04302579e-01, -5.23931459e-02,
         3.65529686e-01, -5.02646923e-01, -4.74538207e-01,
         7.63906896e-01, -2.00036854e-01, -2.67037064e-01,
         1.03515112e+00,  3.46397758e-01, -2.43670538e-01,
         4.50500458e-01, -1.68355346e-01,  5.18702745e-01,
        -6.82818532e-01,  6.24017239e-01,  3.85404602e-02,
        -4.09946501e-01,  3.68629098e-01,  4.96054441e-02,
        -2.33114943e-01,  4.85444993e-01, -1.08563080e-01,
         7.40344301e-02,  7.33888030e-01, -3.63181271e-02,
         2.62143075e-01,  4.39116418e-01, -6.83567524e-01,
         4.89041358e-02,  7.91769743e-01, -4.17364001e-01,
         6.51185036e-01, -3.52027088e-01, -5.82288057e-02,
        -2.13537039e-03,  1.39677539e-01, -6.08139992e-01,
        -1.67434126e-01,  3.79813701e-01, -2.67010033e-02,
         2.19746023e-01,  2.10642487e-01, -1.50555134e-01,
        -1.05163269e-01, -4.75701272e-01,  4.39117521e-01,
         4.93339896e-01, -3.96165580e-01, -5.87925017e-01,
        -1.02103166e-01,  1.78058088e-01,  3.02955210e-01,
        -4.89433706e-01,  1.61032304e-01,  1.22519443e-02,
        -5.89005090e-02,  1.86853200e-01, -7.18616620e-02,
        -1.33315116e-01,  4.56385791e-01,  4.26852018e-01,
         2.06931278e-01,  6.77767117e-03, -3.04977745e-01,
         7.81877786e-02, -2.29707938e-02,  1.70535550e-01,
         2.87278473e-01,  3.66455376e-01, -6.39986917e-02,
        -3.21979702e-01, -1.15096800e-01,  6.37380108e-02,
        -3.45741212e-01, -8.72737467e-01, -3.55792850e-01,
        -6.25388995e-02, -1.63068667e-01,  6.37575313e-02,
        -2.68151075e-01, -6.82171760e-03, -4.36101347e-01,
         1.09982751e-01, -6.27055690e-02, -4.45118062e-02,
        -2.80822486e-01, -9.27583694e-01, -9.45184156e-02,
         3.04895848e-01, -4.79713917e-01, -3.27210933e-01,
         9.45839584e-02,  5.23242474e-01, -4.74489748e-01,
         9.73453522e-01, -2.67334264e-02,  7.02126920e-02,
         2.13770419e-01,  6.22142792e-01, -3.57764840e-01,
        -4.72865254e-01, -1.19014040e-01,  1.80458173e-01,
         9.78040844e-02, -3.81565243e-01,  5.58672287e-02,
        -2.04942107e-01, -6.07924700e-01, -1.55688912e-01,
        -8.88803825e-02,  1.19029045e-01,  2.04908907e-01,
         1.95243314e-01, -1.35292202e-01,  3.09995841e-02,
        -1.43518567e-01,  2.78380901e-01,  4.91197735e-01,
         3.15296352e-01,  4.20068428e-02,  2.56695002e-01,
         4.86028671e-01,  1.31467044e-01,  7.48941973e-02,
         1.83462873e-01,  2.44619548e-01, -2.16858774e-01,
         3.59668881e-01, -2.71159172e-01,  5.15354693e-01,
        -8.01234767e-02, -1.31602749e-01,  1.14667073e-01,
         4.14321512e-01,  3.36664140e-01,  9.51737225e-01,
        -4.68294203e-01, -5.47929525e-01,  9.12783563e-01,
         7.00822711e-01, -3.96858484e-01,  1.19019508e-01,
         1.93363547e-01, -3.60500775e-02, -2.74685264e-01,
         1.06229983e-01,  1.62600502e-01,  2.96904773e-01,
         2.45514870e-01,  5.07883787e-01, -1.21607371e-02,
        -1.20399110e-01, -1.87768899e-02, -4.79419589e-01,
         2.84719050e-01,  1.00585416e-01, -9.79611456e-01,
         3.79275888e-01, -2.45932825e-02,  2.97707003e-02,
         8.89373645e-02, -1.60210878e-02, -2.18441978e-01,
         1.04107313e-01, -1.59560904e-01, -2.75983751e-01,
         4.49743956e-01,  1.10609077e-01, -5.40195942e-01,
         6.43462121e-01,  2.31721729e-01, -3.22686553e-01,
        -8.28095227e-02, -5.98991990e-01,  2.89050937e-01,
        -7.50098944e-01, -1.82394296e-01, -2.12628990e-01,
         7.83567354e-02,  2.82168627e-01,  2.63438970e-01,
         3.20612699e-01,  2.34754115e-01,  1.83150351e-01,
         2.85457283e-01,  2.29576856e-01, -2.06654742e-01,
         2.44399130e-01,  4.39147592e-01,  1.56023383e-01,
        -1.38305768e-01, -6.53927028e-03,  7.66566753e-01,
        -2.07285374e-01,  9.19575766e-02, -9.53285843e-02,
        -4.35106486e-01, -1.22965373e-01, -7.89854050e-01,
         2.94244677e-01, -6.30085170e-02, -2.62077264e-02,
         4.19278145e-01, -1.35393023e-01,  2.64319897e-01,
         3.91130507e-01, -4.41352397e-01, -1.68643013e-01,
         3.66803914e-01, -2.97639102e-01,  4.51262474e-01,
        -1.84189171e-01, -1.01875745e-01,  2.37352327e-01,
        -2.73293525e-01, -3.23818445e-01, -1.46860018e-01,
        -6.54657364e-01,  6.51524484e-01,  4.80685234e-01,
         3.26039970e-01,  8.93502589e-03,  4.86799330e-02,
        -3.64679098e-01,  2.61035949e-01, -2.78794348e-01,
        -3.81026119e-01, -3.04455757e-01,  1.25283912e-01,
         1.83533013e-01, -2.48191744e-01,  1.68629080e-01,
         5.69876432e-01, -3.80468100e-01, -1.48211448e-02,
        -3.46368581e-01, -4.08346474e-01, -1.06749028e-01,
        -2.08240405e-01,  2.61651158e-01, -3.39851305e-02,
         8.63851383e-02,  3.95763993e-01, -9.67696384e-02,
         2.57095456e-01,  5.38815498e-01,  2.59793758e-01,
        -4.97534215e-01, -9.39170569e-02, -9.86275524e-02,
        -2.28326470e-01,  4.07424057e-03,  3.98913890e-01,
        -9.14186984e-02,  2.70864218e-01, -1.18923597e-01,
         1.67921588e-01,  4.39366341e-01,  6.96266770e-01,
        -6.72626257e-01,  6.08640075e-01,  7.56917298e-02,
        -3.37288946e-01, -3.11383873e-01, -2.59985179e-01,
         2.53157109e-01, -5.18727720e-01,  5.65564148e-02,
         1.99406579e-01, -2.77464688e-01, -1.22484401e-01,
         3.70863169e-01,  1.15844578e-01, -5.56815155e-02,
        -3.81392300e-01, -1.43527493e-01,  2.15361014e-01,
        -3.75189036e-01, -1.49016380e-02, -4.58493441e-01,
        -1.76458001e-01,  4.11861867e-01, -7.78632939e-01,
        -1.53032541e-01,  2.73693800e-01, -1.21389501e-01,
        -3.77456218e-01,  4.99894261e-01,  1.44745201e-01,
        -2.72670627e-01,  1.57100007e-01, -1.18754416e+01,
         2.19080567e-01,  1.63526595e-01, -2.29906052e-01,
         1.45944551e-01,  1.19521938e-01,  2.22915784e-01,
        -2.73774043e-02, -3.55248690e-01,  1.29800066e-01,
        -5.85893512e-01, -1.87939018e-01,  7.56833553e-01,
         1.32106483e-01,  3.90221447e-01,  4.51841615e-02,
         2.53038198e-01,  2.01215878e-01, -3.56657863e-01,
         2.12585777e-02, -3.37050594e-02, -9.12581608e-02,
        -7.99443722e-02,  3.58428687e-01, -6.48412943e-01,
         4.72221166e-01, -3.82086188e-01,  4.32691932e-01,
        -4.06255536e-02,  5.19373834e-01,  6.95409596e-01,
        -6.30341530e-01, -2.26631775e-01,  2.35245060e-02,
        -2.62398094e-01,  1.59063116e-01, -4.42464352e-01,
         3.76208067e-01, -2.86560088e-01,  8.76452774e-02,
        -1.46660469e-02, -1.78098395e-01,  4.99708354e-01,
         1.36729047e-01, -1.07371978e-01, -1.28253764e-02,
        -6.34662211e-02,  3.48384261e-01,  2.00628694e-02,
        -7.19070673e-01, -1.60351187e-01,  9.35811624e-02,
         7.31709525e-02, -5.14569022e-02,  1.09936702e+00,
         1.52229249e-01,  4.41794187e-01,  5.50786376e-01,
         4.65815306e-01, -2.66599923e-01,  7.43036494e-02,
        -9.20686245e-01, -6.84893608e-01, -5.06588668e-02,
        -3.97043340e-02,  3.25588524e-01,  2.63692942e-02,
         5.98061681e-01, -1.16336815e-01, -5.40668905e-01,
         2.78817713e-01,  7.68379420e-02, -1.48884803e-01]], dtype=float32)
```

In [5]:

```
question_file = 'q.csv'
questions = pd.read_csv(question_file, header=None)

answer_file = 'a.csv'
answers = pd.read_csv(answer_file, header=None)
# Load Q&A template from UIC Admission Office

questions_list = list()
for i in questions.values:
    questions_list.append(i[0])

answers_list = list()
for i in answers.values:
    answers_list.append(i[0])
# Data processing
```

executed in 18ms, finished 01:43:25 2019-12-23

In [6]:

```
doc_vecs = bc.encode(questions_list)
# Encode questions
```

executed in 534ms, finished 01:43:26 2019-12-23

In [12]:

```
topk = 5
# top k matched question
query = input(colored('Your question: ', 'green'))
#Input your question here.
query_vec = bc.encode([query])[0]

# compute normalized dot product as score
score = np.sum(query_vec * doc_vecs, axis=1) / np.linalg.norm(doc_vecs, axis=1)
topk_idx = np.argsort(score)[::-1][:topk]
for idx in topk_idx:
    print('>%s\nQuestion:%s\n%s' % (colored(score[idx], 'red'),
                                    colored(questions_list[idx], 'yellow'),
                                    colored(answers_list[idx], 'green')))
```

executed in 4.83s, finished 01:44:48 2019-12-23

```
Your question: 假期课程
>16.205692
Question:什么是暑期课程?
暑期课程是UIC学子在大一至大三的暑假期间的一个短期校外留学项目，一般为期3-5周，项目根
据不同国家和院校设计了很多不同专业科目和语言背景的课程。暑期课程分为香港浸会大学暑期
课程和海外暑期课程。
>15.973037
Question:在UIC暑假，学生只是去修暑期课程吗?
除了暑期课程，学生还可选择参加UIC全人教育暑期实践活动、中国语言文化中心台湾游学营、
香港浸会大学“大都会体验计划”（海外实习）等。UIC全人教育办公室与国内外各类机构（如
户外体育运动协会，非政府组织、慈善公益组织、自然保护管理区、学校等）合作共同组织并开
展体验全人教育模块特色的体验式学习活动，为学生提供丰富多彩的寒暑期项目。每个项目一般
持续2-3个星期，传统项目有：柬埔寨，泰国等暑期义工服务项目，上海真爱梦想教练计划，中
国西北部沙漠草原环境之行，台湾潜水&海洋文化游、瑞士户外体育探险夏令营等。
>15.9349575
Question:副修课程有哪些?
目前学校共推出五个副修课程，每个副修课程学生最多可修得15个学分，副修课程分别为：音
乐、工商管理、财务学、公共关系与广告学以及应用心理学。
>15.910742
Question:暑期课程是必须参加的吗?
暑期课程为学生自愿申请参加，2016年约有700人参加海外或香港的暑期课程。
>15.607442
Question:学生会被退学吗?
我校有一些严格的学术规定，学术处分适用于所有本科生
a) 学术警告：适用于某一学期平均绩点（GPA）在1.67-1.99之间的学生；
b) 留校察看：适用于某一学期的平均绩点（GPA）在1.67以下的学生；
c) 勒令退学：如学生连续两学期的平均绩点（GPA）低于1.67或其他学术原因，由教务议会（Se
nate）作出决定。
```

In [ ]:

```
```