NLP Advanced

Wenqian Zhang

Xi'an Jiaotong University

2194510944@stu.xjtu.edu.cn

March 2nd, 2022

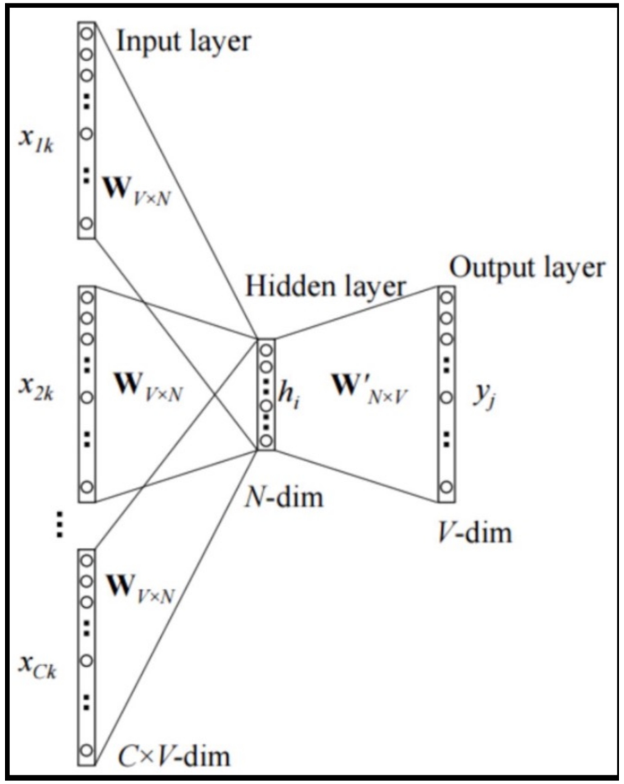
Contents

- Revisited
- Attention and Transformers
- Self-supervised Learning
- Pre-trained Language Models

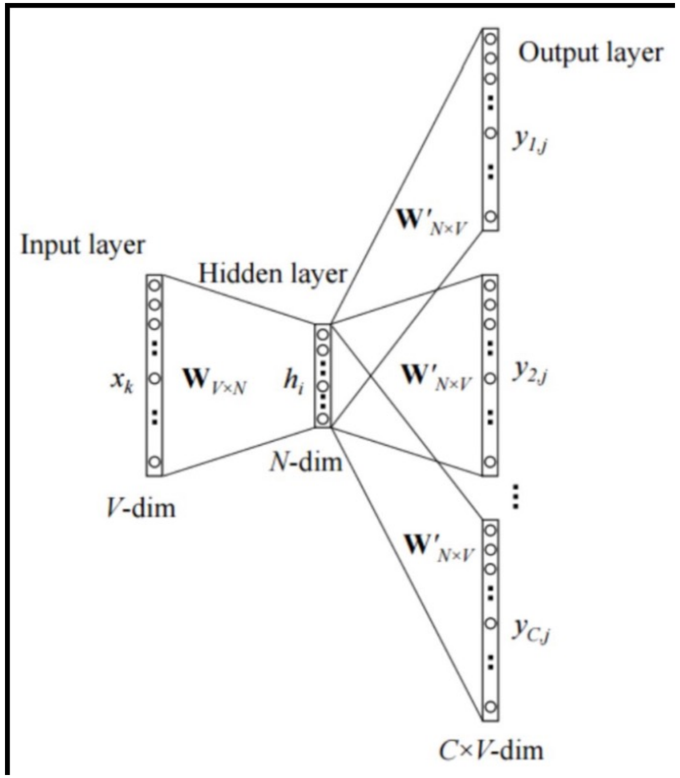
Revisit: Word2Vec

- “you shall know a word by the company it keeps”

- CBOW



Skip-Gram



Revisit: Language Model

- A language model tries to **predict the next word(token)** given the previous token sequence.

$$P(x^{(t+1)}|x^{(t)}, \dots, x^{(1)})$$

- RNN Language Model

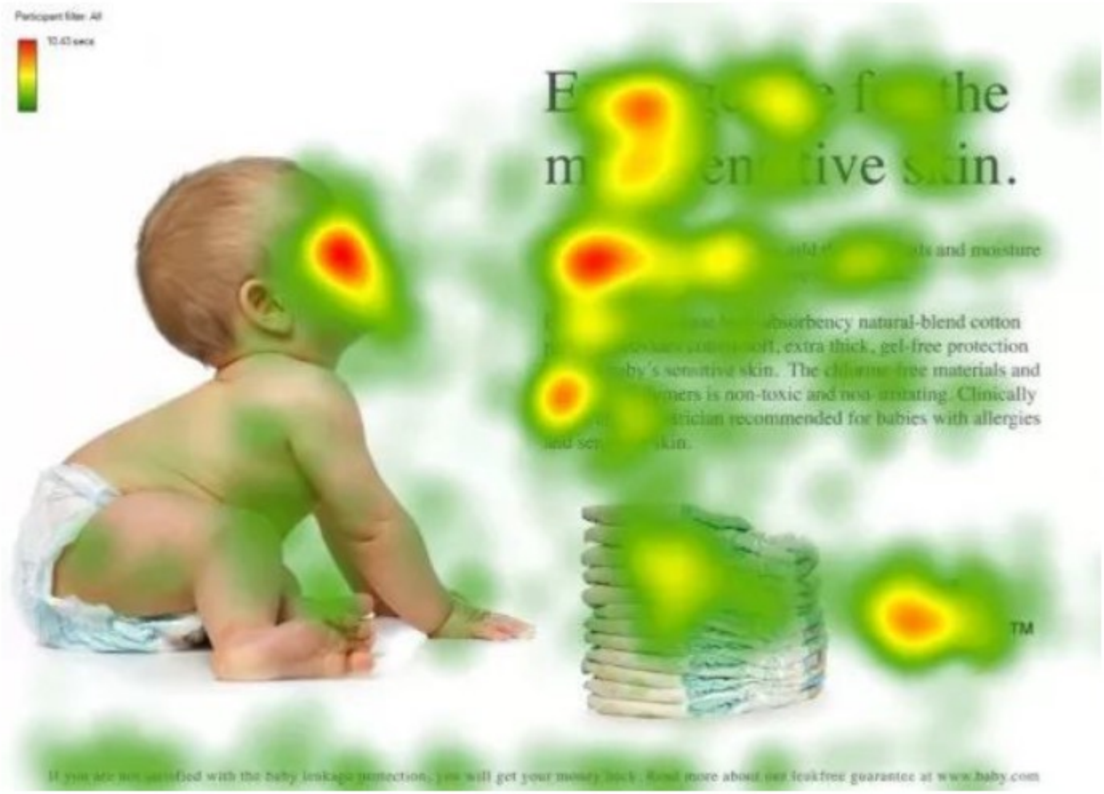
- How to train an RNN language model?

Contents

- Revisited
- Attention and Transformers
- Self-supervised Learning
- Pre-trained Language Models

1. What is attention?

- we tend to focus on something meaningful, containing much information
- Is there a same mechanic for computer?



- attention aids machine to perceive meanings more precisely

It is often the case in translation that a target word can be inferred by only a few words in the source!

e.g. Tom chase Jerry

汤姆 追赶 杰瑞

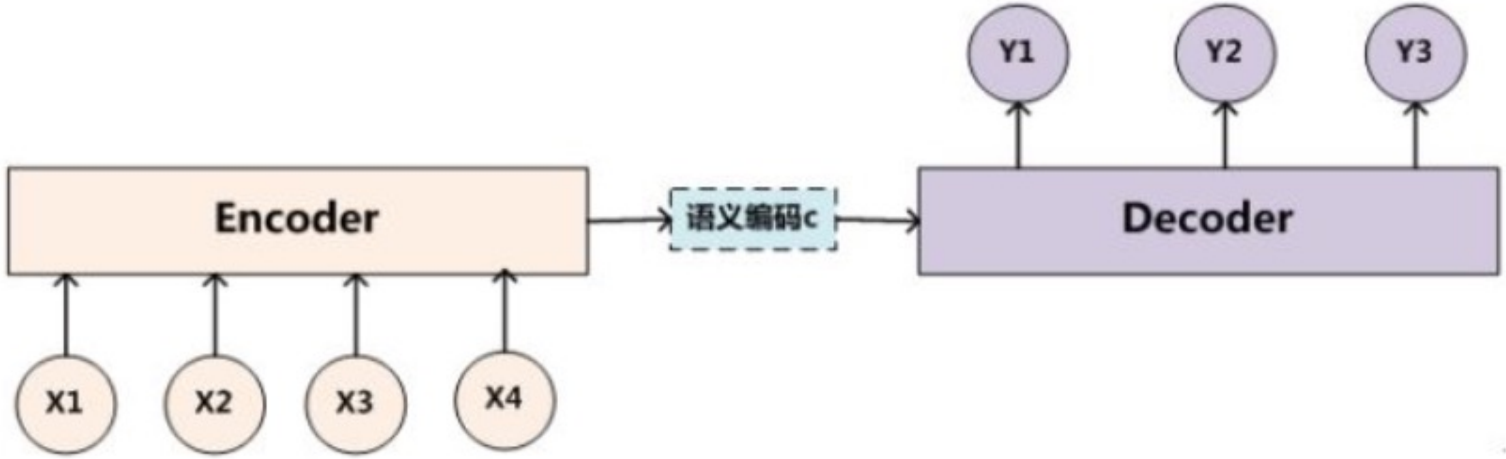
An apple a day

一天 一个 苹果

When we get the result ‘追赶’，'chase' plays a great role.

Now, let's begin with the tradtional one

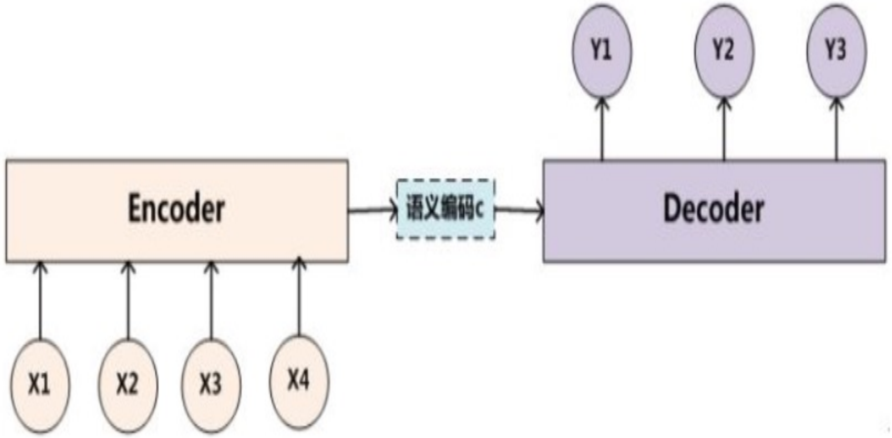
- Encoder---Decoder



In NLP, encoder---decoder model is widely used;
It is natural to use RNN is both encoder and decoder

First, embedding and encode

- Source = $\langle x_1, x_2, \cdots\cdots, x_n \rangle$
- Target = $\langle y_1, y_2, \cdots\cdots, y_m \rangle$
- The function of encoder: C = F(x1,x2,....,xn) and this vector C is pass to decoder



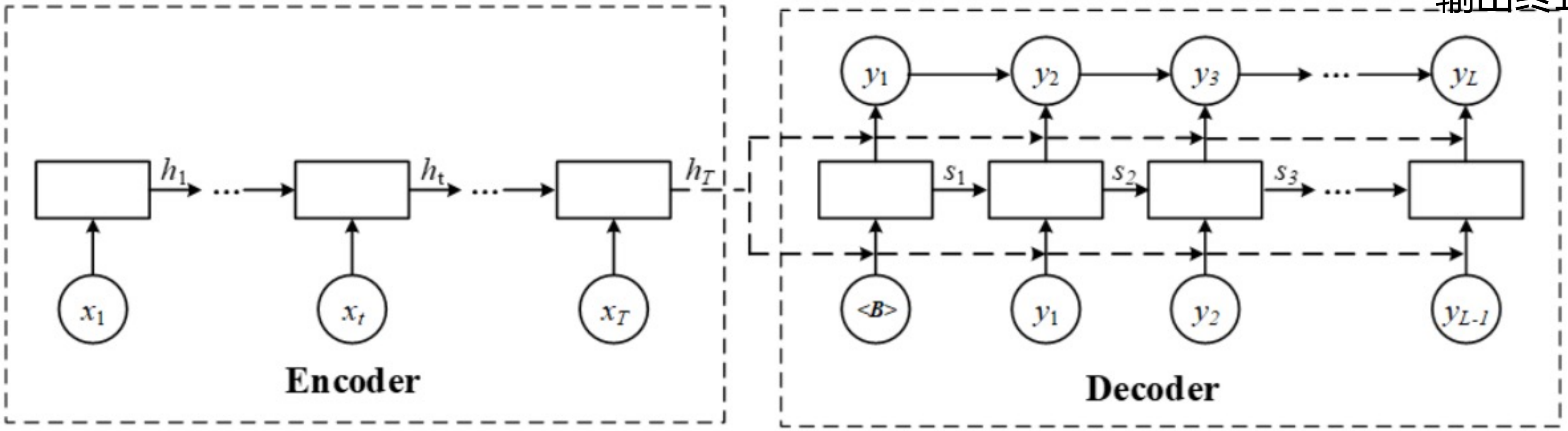
Then, with the connotation vector, decode

- Given C, and (Start), get y1;
- Given C, and y1, get y2;
- Given C, y1 and y2, get y3;
- C, y1,y2,....ym-1 are known, get ym=EOS, the translation process ended.

More precisely, we bring RNN into play

- The following picture shows how it works:

输出终止位



- where $S_t = f(S_{t-1}, y_{t-1}, h_T)$
- and $y_t = g(y_{t-1}, S_t, h_T)$

Problems in the model above

- when we get the translation result yi, the whole source sentense is used
- yi may only rely on x1, x3, x4, but we have used all words xj

Tom chase Jerry

汤姆　追赶　杰瑞

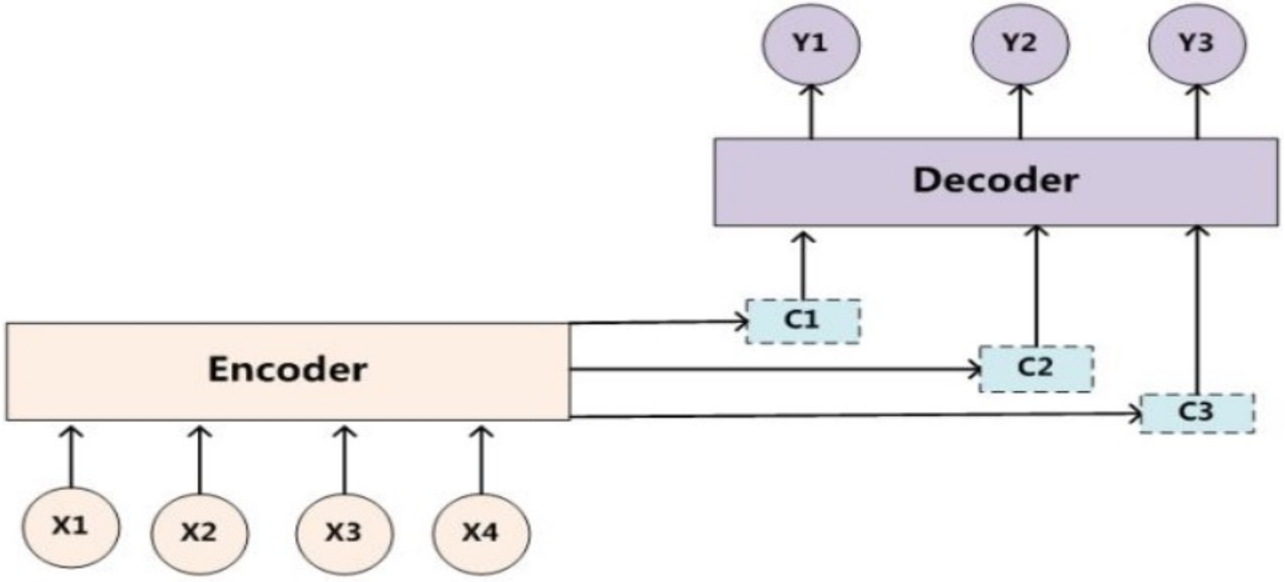
although we can hardly let computer to articulate which xi corresponds to which yj

It will be much better if we attach a weight to each word

e.g. (Tom: 0.2) (chase: 0.2) (Jerry: 0.6) when translate '杰瑞'

Add attention into the model

- After analysis, it is better to use different context vector Ci when translate each yi
- The new m



- y1 = F(C1)   y2 = F(C2, y1)   y3 = F(C3, y1, y2)

How to define such Ci's

- One idea to get Ci is weighted-sum

  Namely: $C_i = \sum_{j=1}^{L} a_{ij} h_j$

  Here aij are some coefficients , and hj is the embedding vector of xj

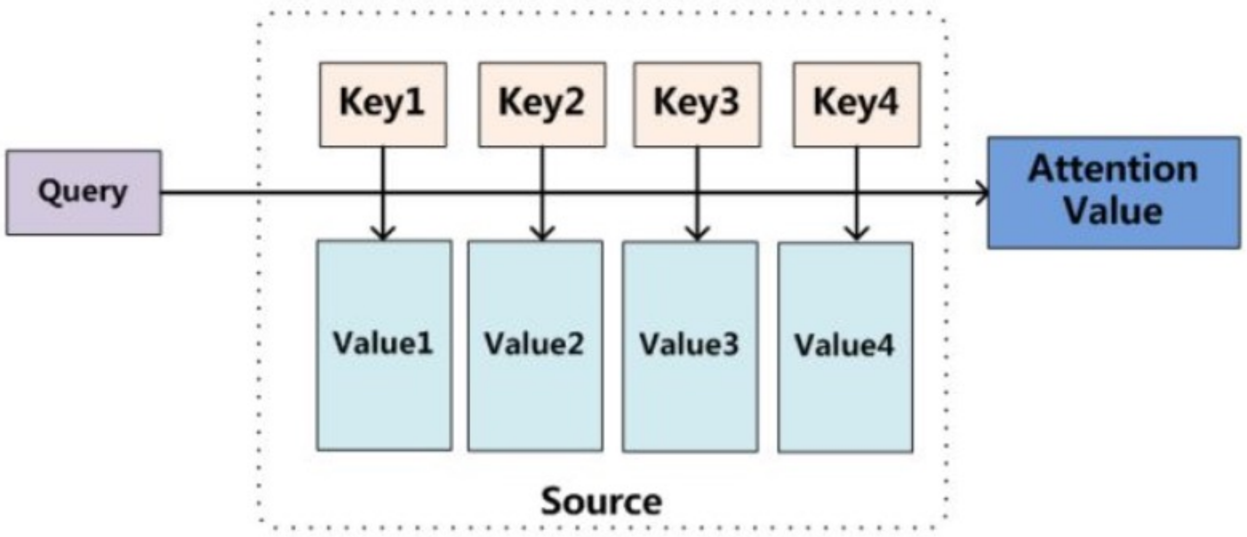
苹果

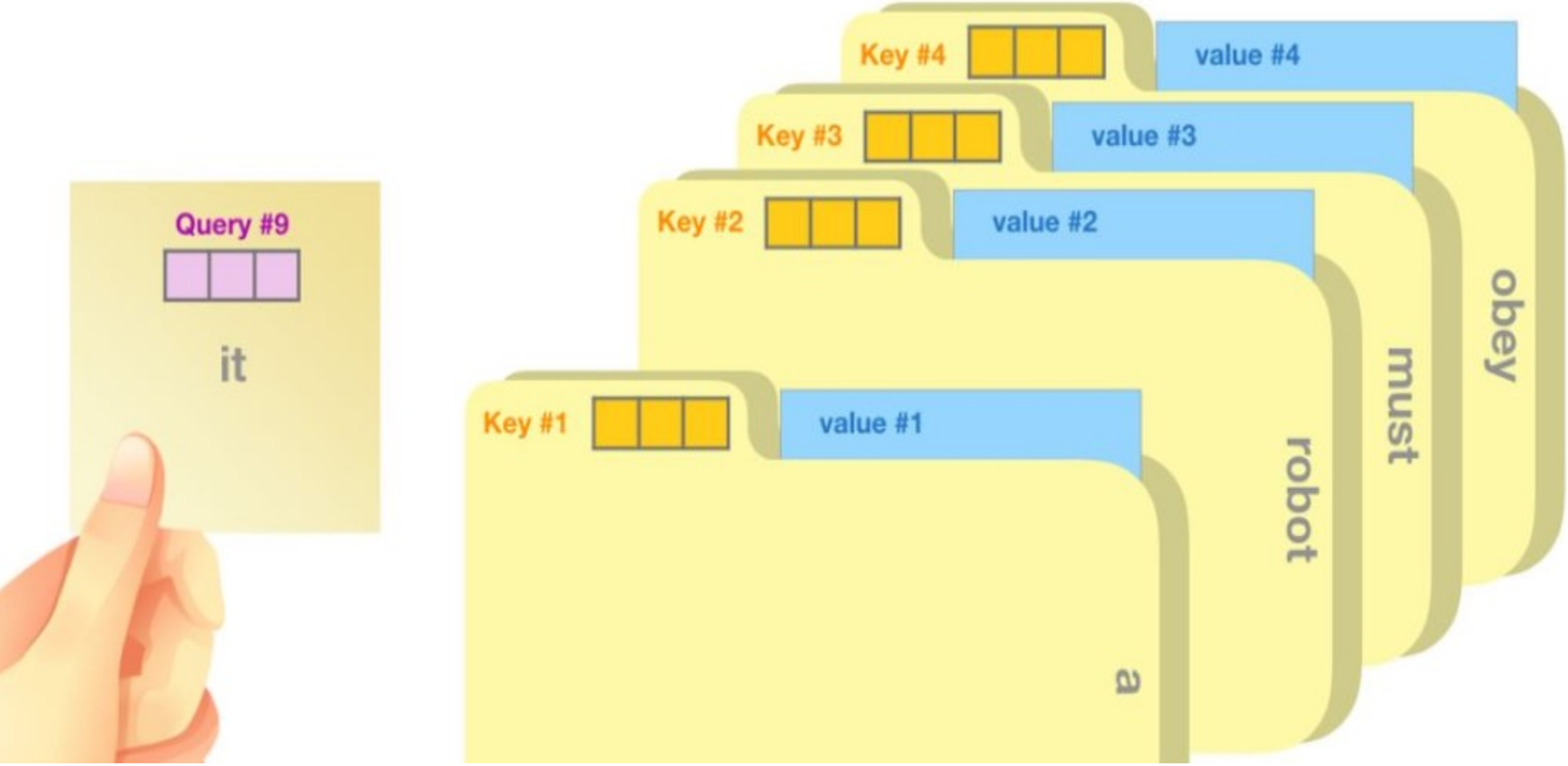
a1j: (0.1) (0.7) (0.1) (0.1)

An apple a day

Soft-addressing

- Now we are going to view the 'attention' in a different way
- What is soft-addressing?
- Attention is a special case of soft-addressing




Query #9
it
Key #1
value #1
a
Key #2
value #2
robot
Key #3
value #3
must
Key #4
value #4
obey

Key = Value in attention

- the formula is as below:
- Attention(Query, Source) = $\sum_{i=1}^{L}$Similarity(Query, Key(i)) * Value(i)

- (1) dot-product: Similarity(Query, Key(i)) = Query * Key(i)
- (2) Cosine-sim: Similarity(Query, Key(i)) = Query * Key(i)/||Query||*||Key(i)||
- (3) scaled dot-product: Similarity(Query, Key(i)) = Query * Key(i) / squareroot(d)

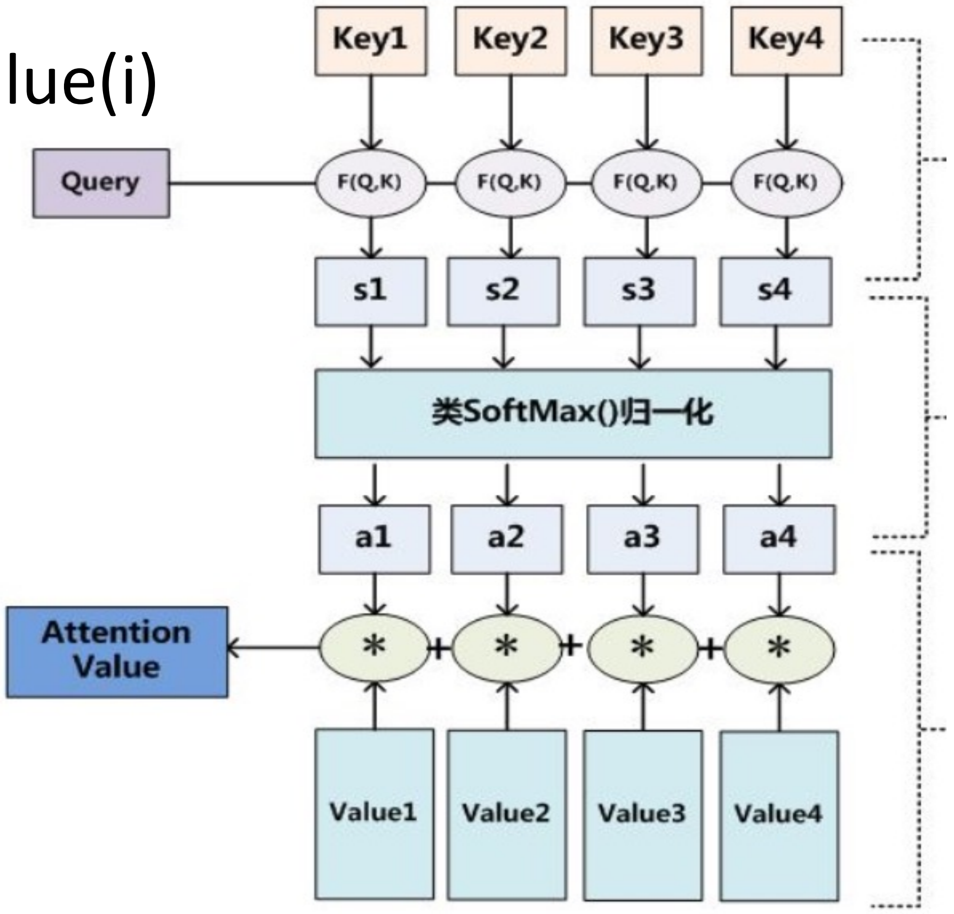
Get the attention vector

- then , ai = softmax(Sim(i)).
- Attention(Query, Source) = $\sum_{i=1}^{L}$ ai * Value(i)

$$f(Q,K_i)=\begin{cases} Q^T K_i & dot \\ Q^T W_a K_i & general \\ W_a[Q;K_i] & concat \\ v_a^T \tanh(W_a Q + U_a K_i) & perceptron \end{cases}$$

$$a_i = soft\max(f(Q,K_i)) = \frac{\exp(f(Q,K_i))}{\sum_j \exp(f(Q,K_j))}$$

$$\text{Attention}(Q,K,V)=\sum_i a_i V_i$$



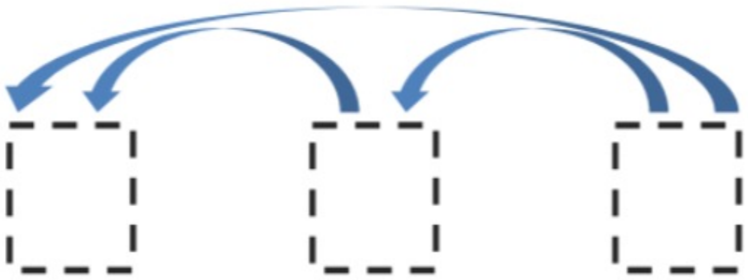
Self-attention

- What is self attention?
- In K=key, Q=Query, V=Value, self-attention takes K=V=Q
- Use self-attention to embed sentence in R^d
- It can catch long-distance dependence
- It is parallizable, we use a matmul to get the result
- constant path length between any two position

Masked self-attention

- the normal self-attention is fully connected:

- which will be used in encoder
- the masked self-attention is semi-connected:



the later word only depend
on the former ones

Multi-head attention

- Instead of compute the attention function once for all,
  we divide the total dimension d=d(model) into h many 'pieces
- That is , project the total matrix onto h-many subspaces
  Each subspace has dimension d'=d/h
- It is beneficial to learn h-many sub-repesentations, and
  sum them together to learn the final attention

Formula for multiheaded

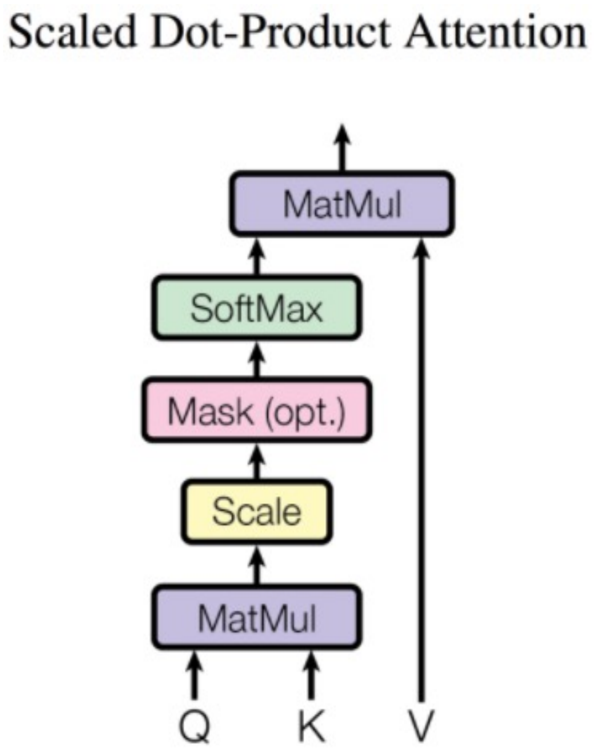
- traditional is :
- Attention(Q, K, V) = softmax(Q*K'/squareroot(d))*V
- multihead splits each matrix, by projecting them onto subspaces:
- written as: $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

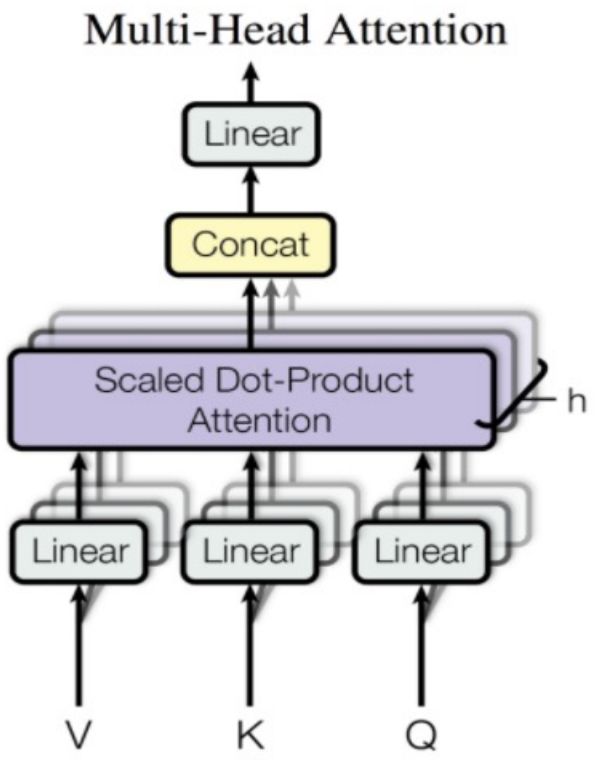
and then: $MultiHead(Q, K, V) = Concat(head_1, head_2, \cdots head_h)W^o$

where: $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v} \quad W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$

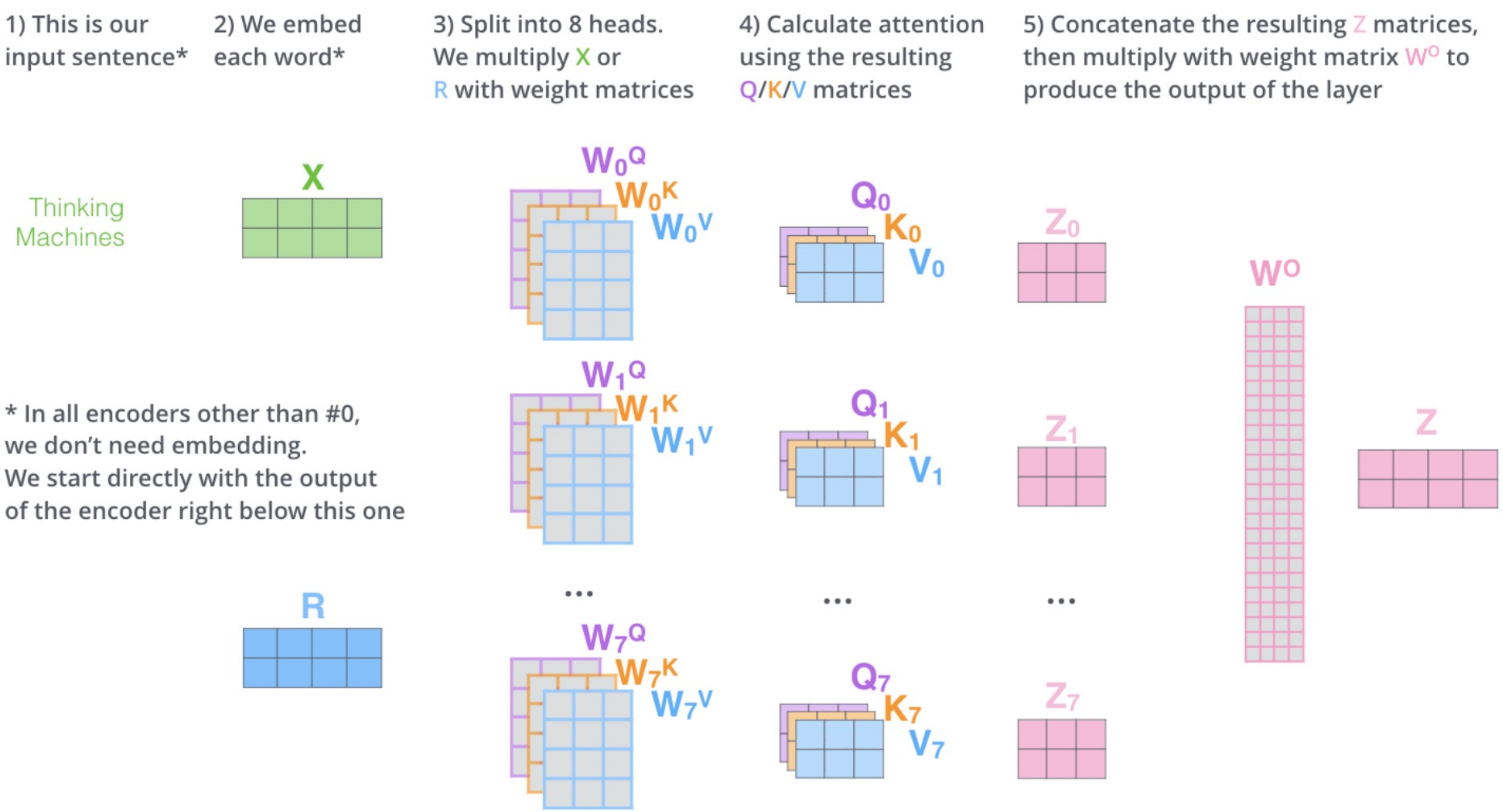
- e.g. d(model) = 512, h = 8, then dv=dk=d/h=64

Some pictures





Some pictures



multihead + self attention

- multihead attention allows the model to jointly attend
  to information from different representation subspaces
  at different positiions

- self attention is good at catching long-distance dependency
  we can use this to learn word representations

- maybe just attention is good enough !
  '' Attention is all you need''   --- Google, 2017

illustrate

Attention Visualizations

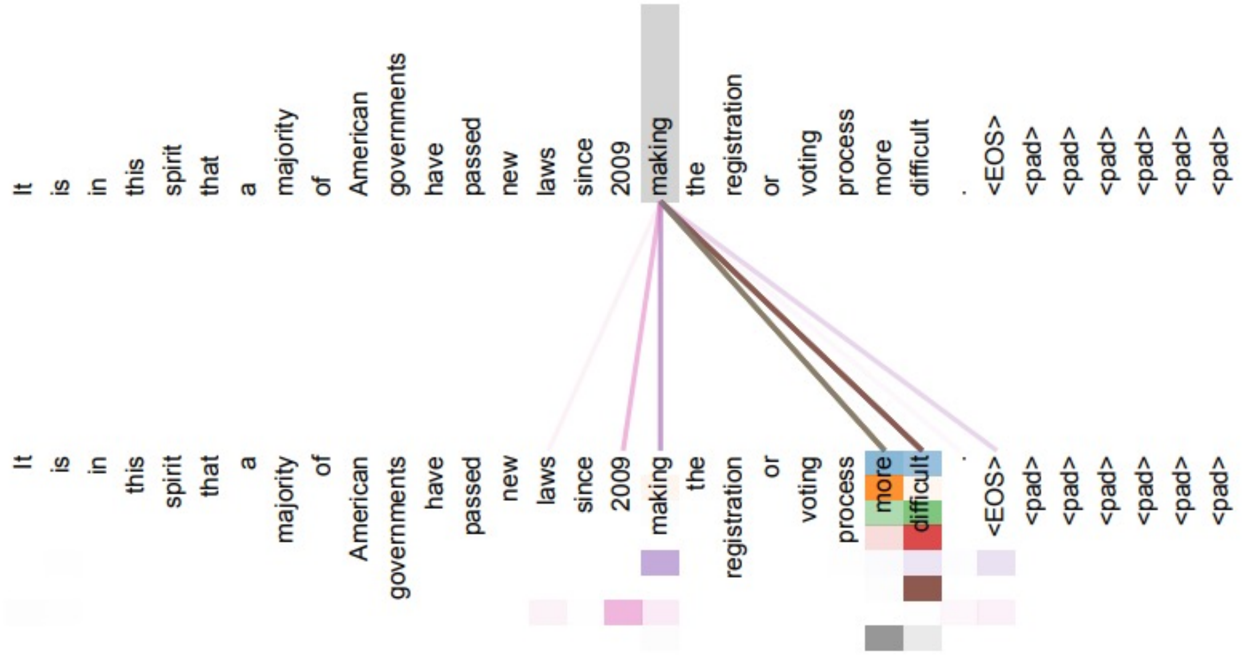


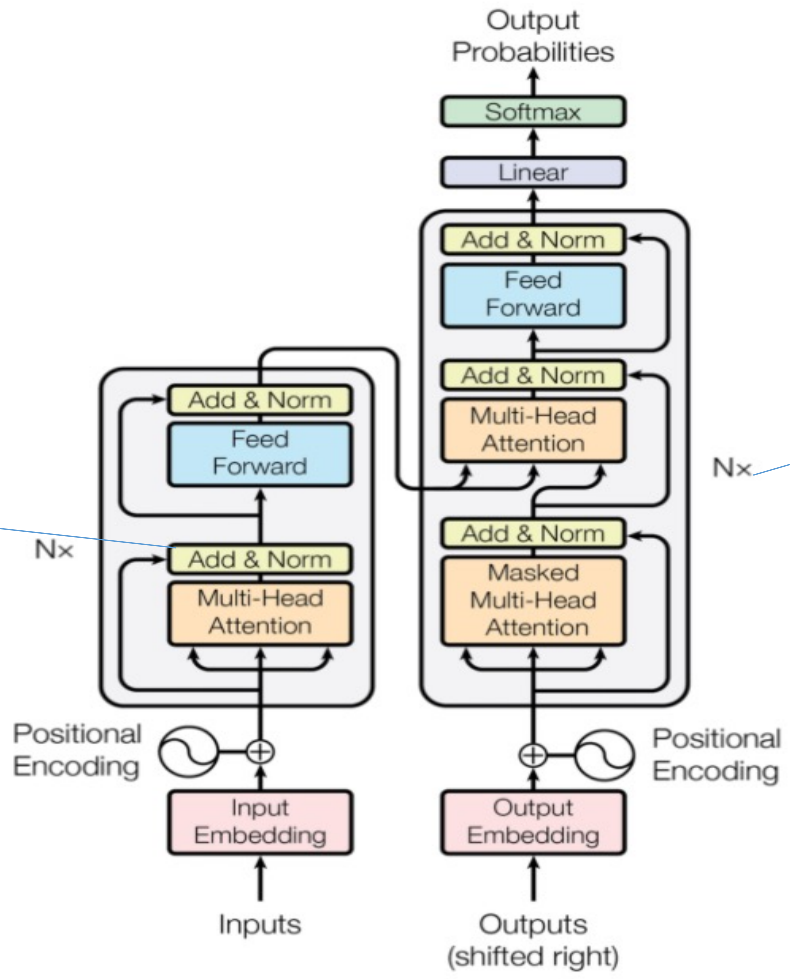
Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

Only attention

- In 2017, Google successfully built a transformer, using attention mechanism only.
- Without any RNN or CNN in its encoder and decoder
- But the result is cheerful, biting any other translator
- Let's see their model architecture

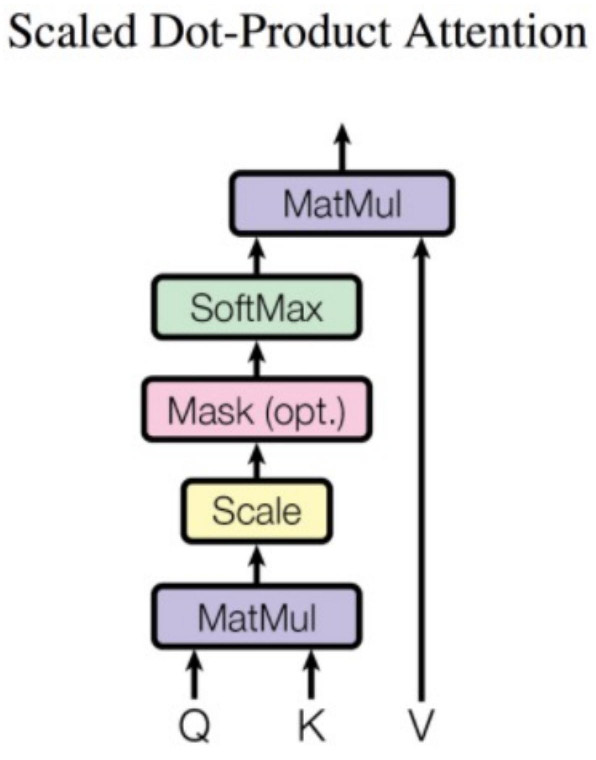
Transformers

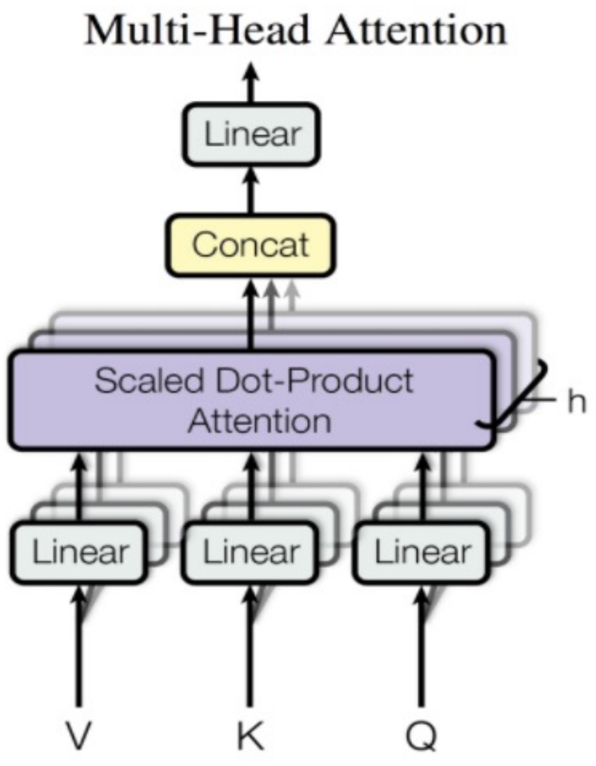
$LayerNorm(x + Sublayer(x))$



Here, N=6
each multihead
has 8 heads

Some pictures





Summary

- QKV attention
- Self-attention and in NLP
- Multi-head attention
- transformers

Contents

- Revisited
- Attention and Transformers
- Self-supervised Learning
- Pre-trained Language Models

Motivation

- Data annotation has always been the soft spot…

- Few-shot learning
- Unsupervised leanring
- Semi-supervised learning

- Self-supervised learning

Definition

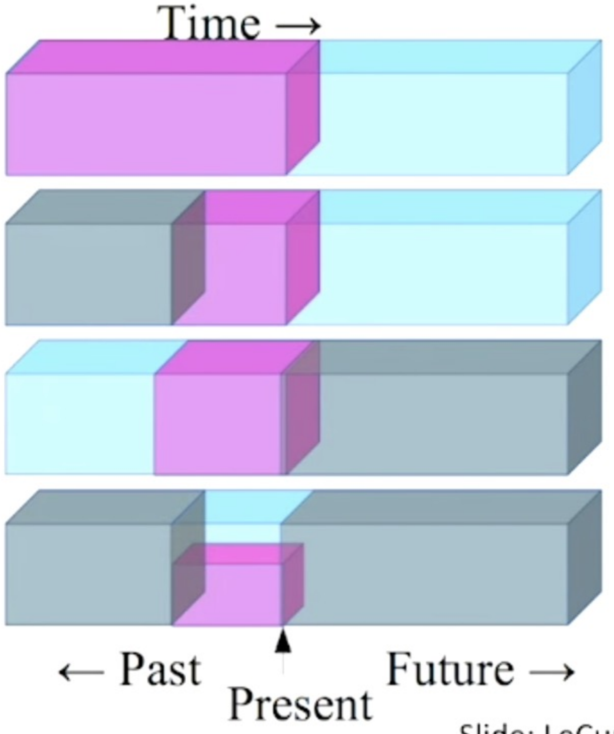
- Framing a supervised learning task in a special form where **predict only a subset of information using the rest**

- In which way, both inputs and labels are provided by the dataset

But

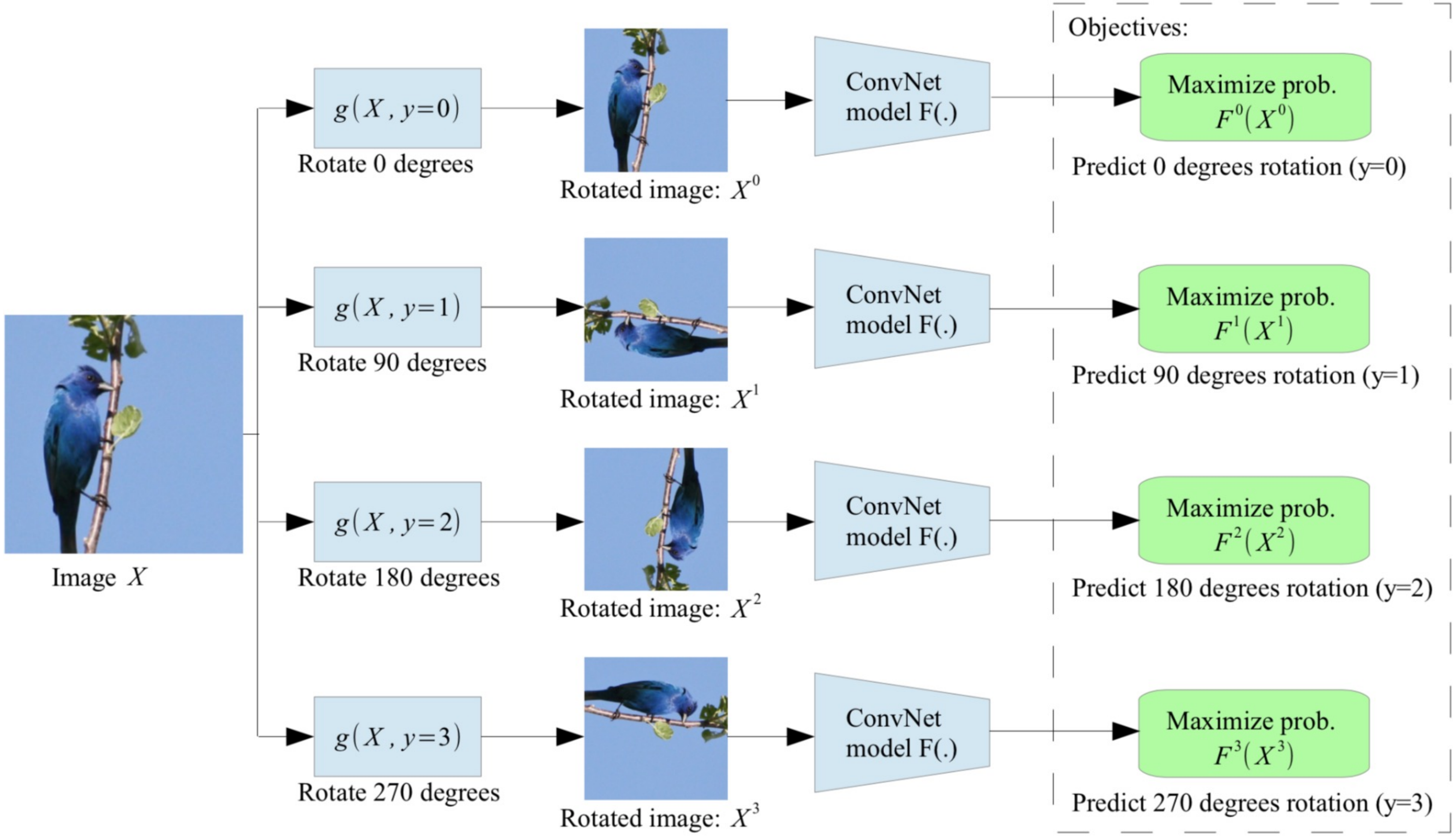
- How do we decide which to predict?

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.



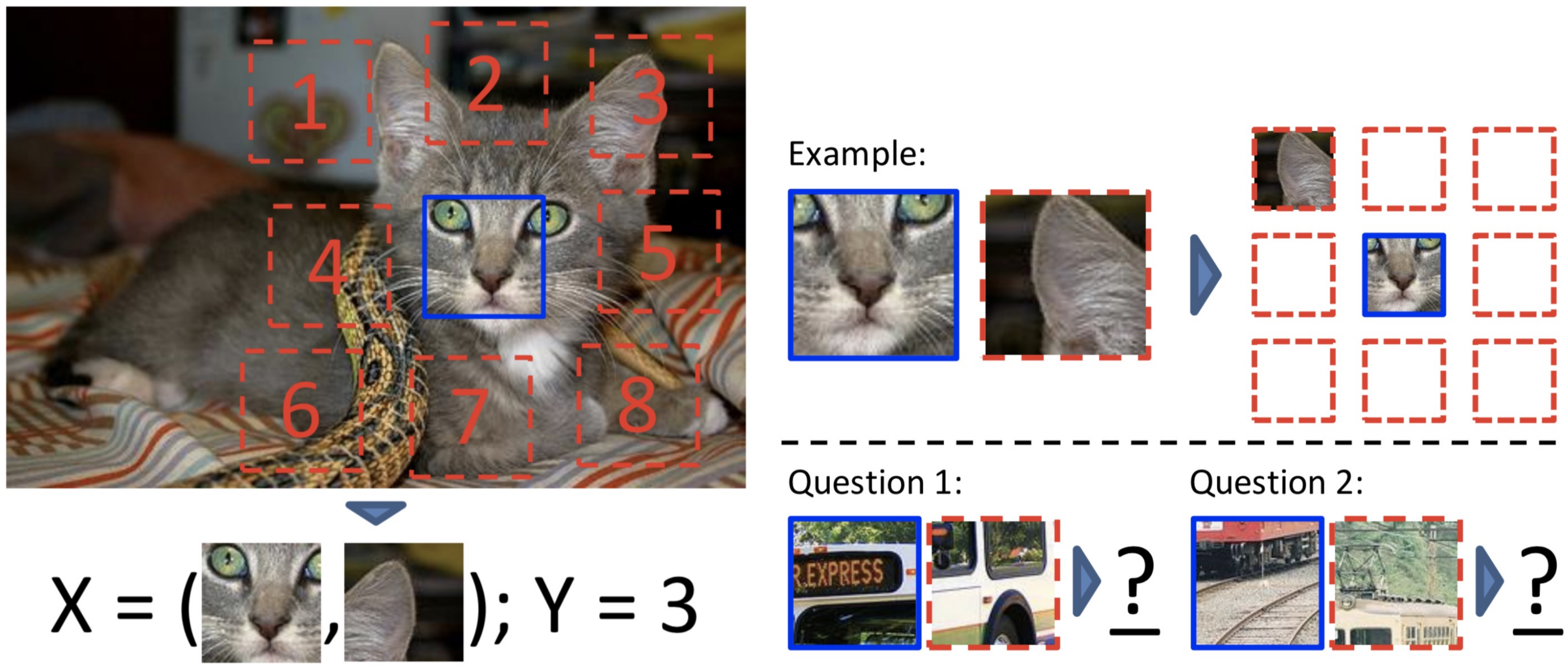
Slide: LeCun

Image: Rotation



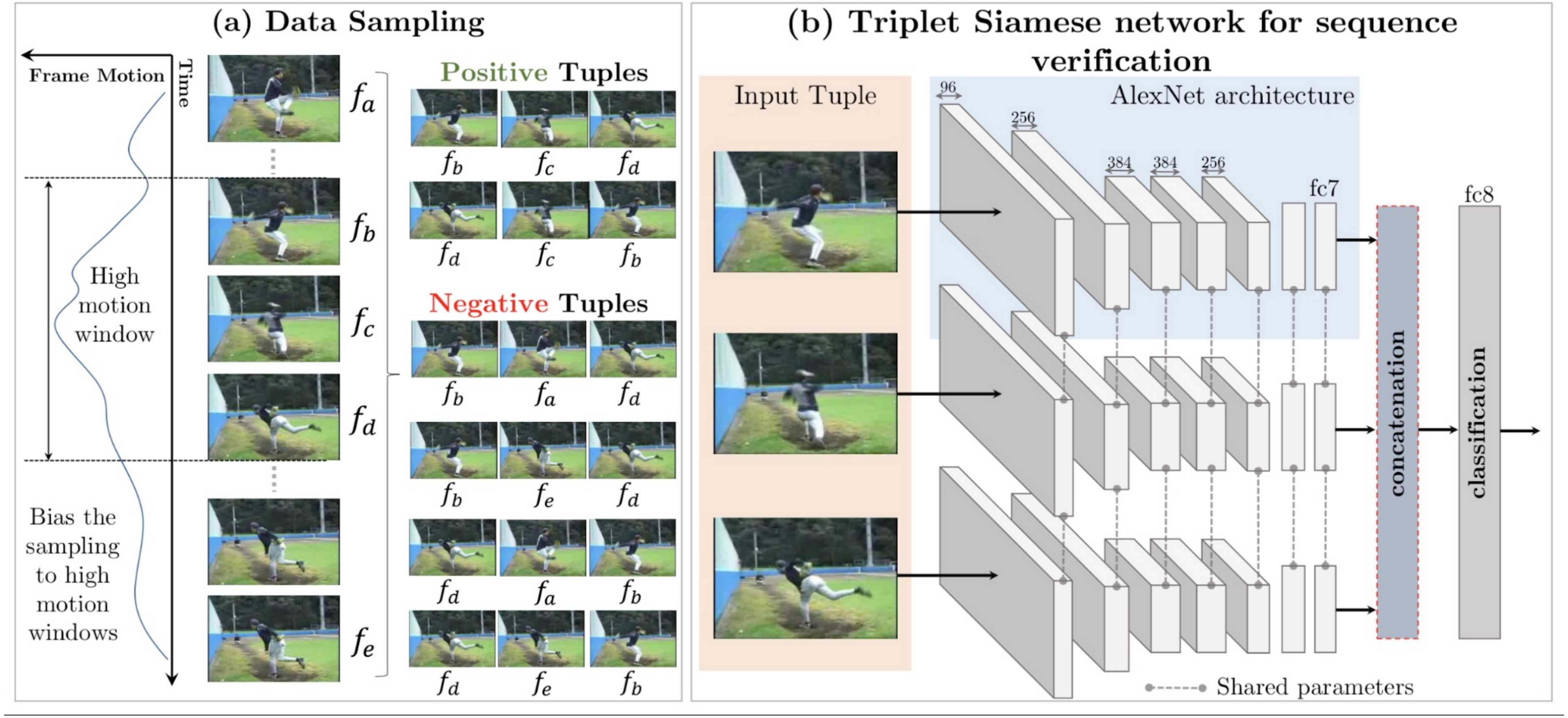
- Learn image semantics in the process

Image: Patch



- Gap between patches is crucial…

Video: Frame Sequence



NLP

- Predict next word of a sentence
  - Language Model

- Predict words within the context window
  - word2vec

Why self-supervision works

- In the process of solving auxiliary tasks, the network learns **representations** of inputs
- that could be utilized to solve downstream tasks

- **Representation Learning**
  - In comparison to **feature engineering**
  - A system automatically discover features (e.g. word embedding)

- ICLR

However

- The self-supervised task is not my objective!

- Self-supervised applications serve as a pre-training method, that captures semantic/context/correlation, learns **generic knowledge**

- Fine-tuning for the downstream tasks

- Which brings to transfer learning, specifically pre-train and fine-tune

Motivation

- Train a network from scratch?
  - Limited data, especially supervised
  - Limited computational resources (maybe u don't feel it now lol)
- Observation: tasks that are similar in context
  - image classification, HUGE dataset and lots of efforts
  - Image style transfer (e.g. horse 2 zebra)
    - Is there any supervised data at all?
    - Same context: understanding images

Transfer Learning

- Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

- Often,
  - The first task is rich in data(supervised in CV & self-supervised in NLP)
  - The first model is computationally expensive
  - The second task lacks data or accurate annotation
  - Both tasks are similar in underlying generic knowledge

TL: Feature Extraction Approach

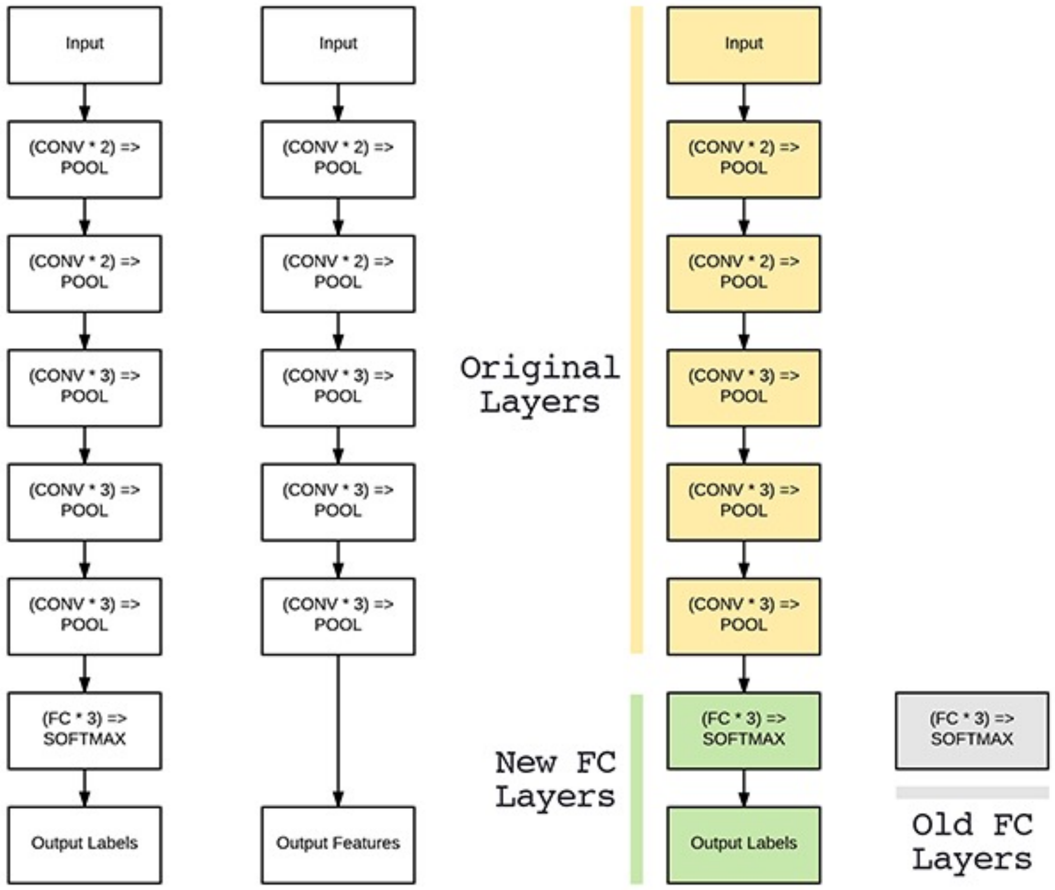
- Extract the data representation from the first approach as the input for the second model, keeps the first model param unchanged

- For instance
  - Word2vec in NLP

- Question: Are features discovered in the first task exactly what we want in solving the second task?

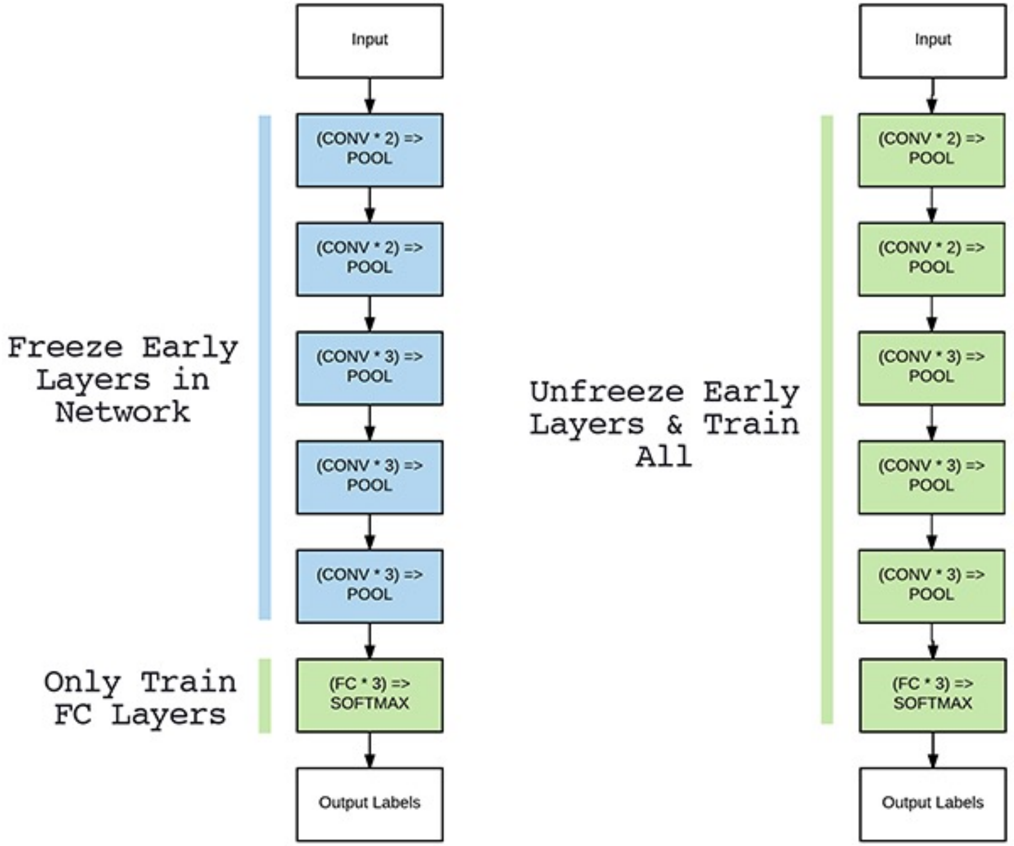
TL: Fine-Tuning Approach

- #1 Architecture Change
- MOST COMMON APPROACH
- **Truncate the last softmax layer**
- **and replace it with a new one**
- Retain original layer's params



TL: Fine-Tuning Approach

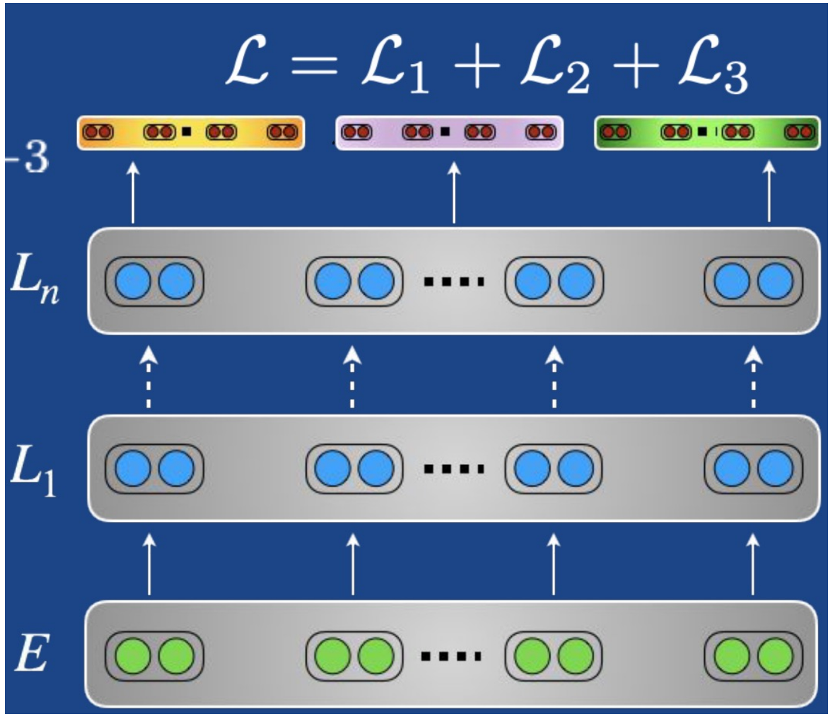
- #2 Optimization Scheme: Shallow vs Deep FT
- Tricks in deep FT
  - Reduced learning rate
  - Gradual unfreezing (direction?)
  - Tougher regularization
  - warmup



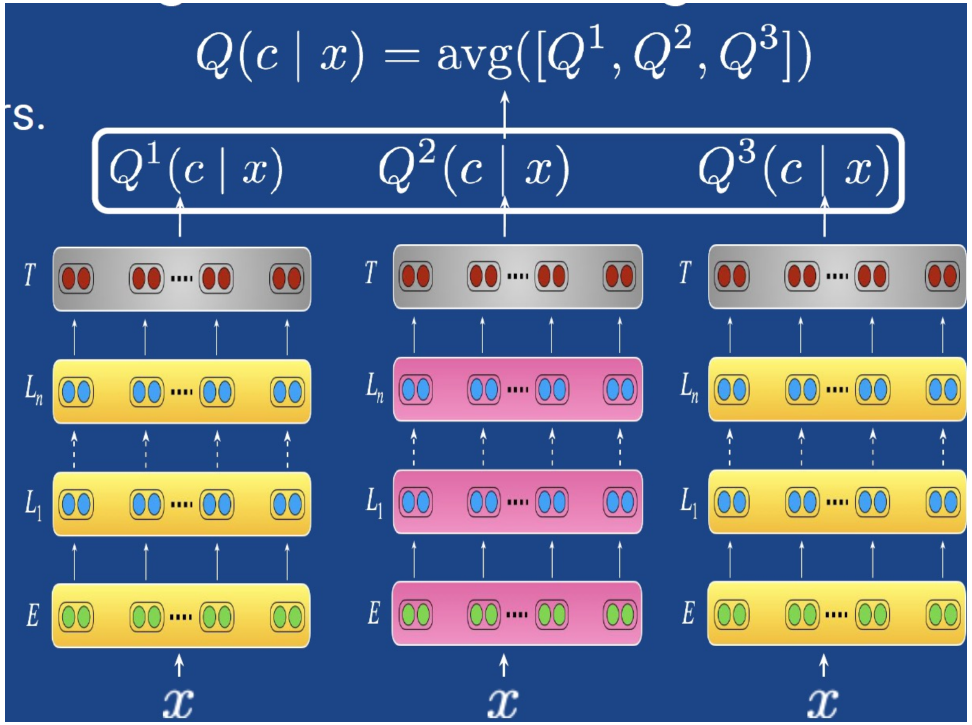
TL: Fine-Tuning Approach

- #3 Getting more signal

- Multi-task fine-tuning



Ensembling



Takeaway till now

- Attention
- Self-supervised learning
- Transfer learning
- Pre-training
- Fine-tuning

Contents

- Revisited
- Attention and Transformers
- Self-supervised Learning
- Pre-trained Language Models

BERT

- **B**idirectional **E**ncoder **R**epresentation from **T**ransformers

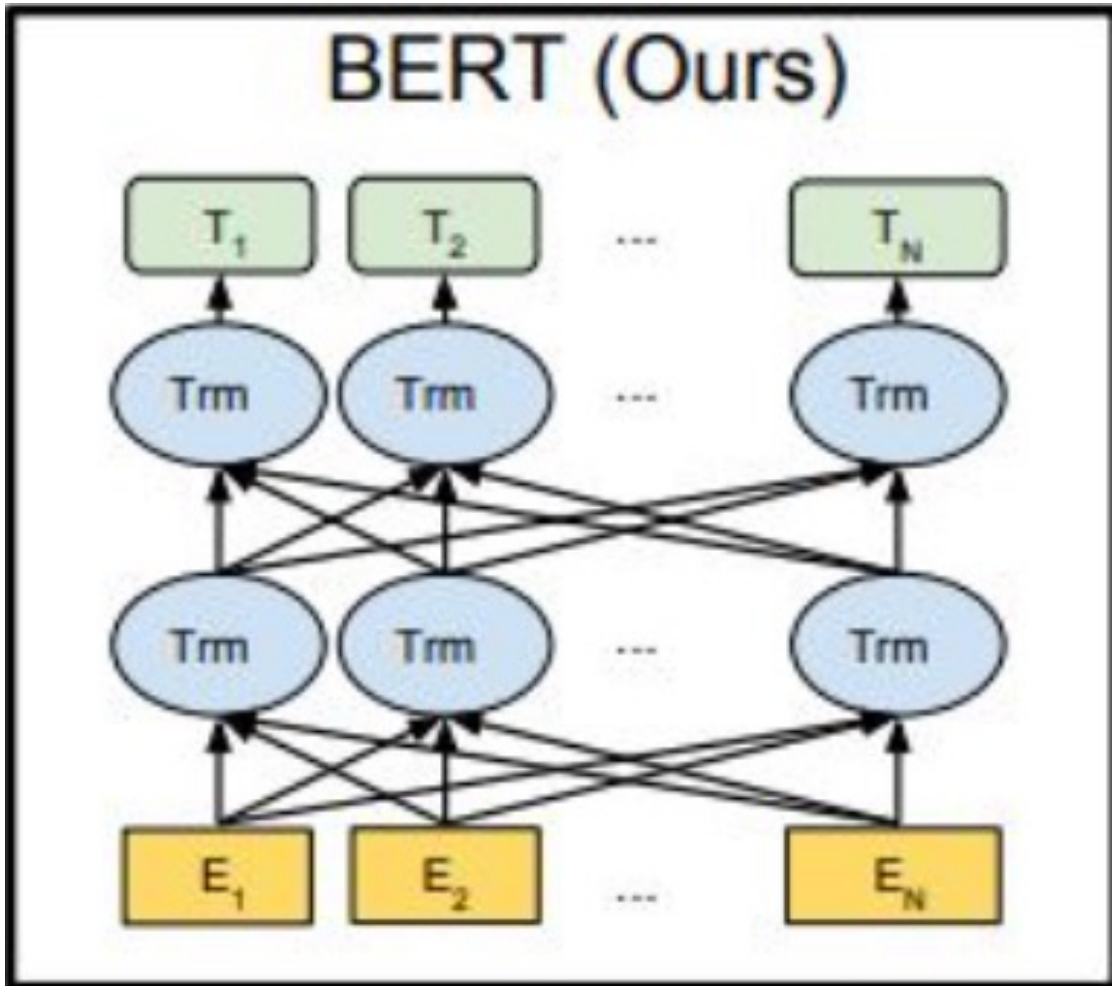
- After transformers, another shocking breakthrough from Google to the NLP community

- A self-supervised pre-training Language Model
  - Architecture?
  - Self-supervised pre-training tasks?
  - How do I use it in fine-tuning downstream tasks?

BERT Architecture

- Only encoder of transformer

- Bidirectional? Undirectional
  - Read the entire sequence at once
  - The ‘context window’ is total



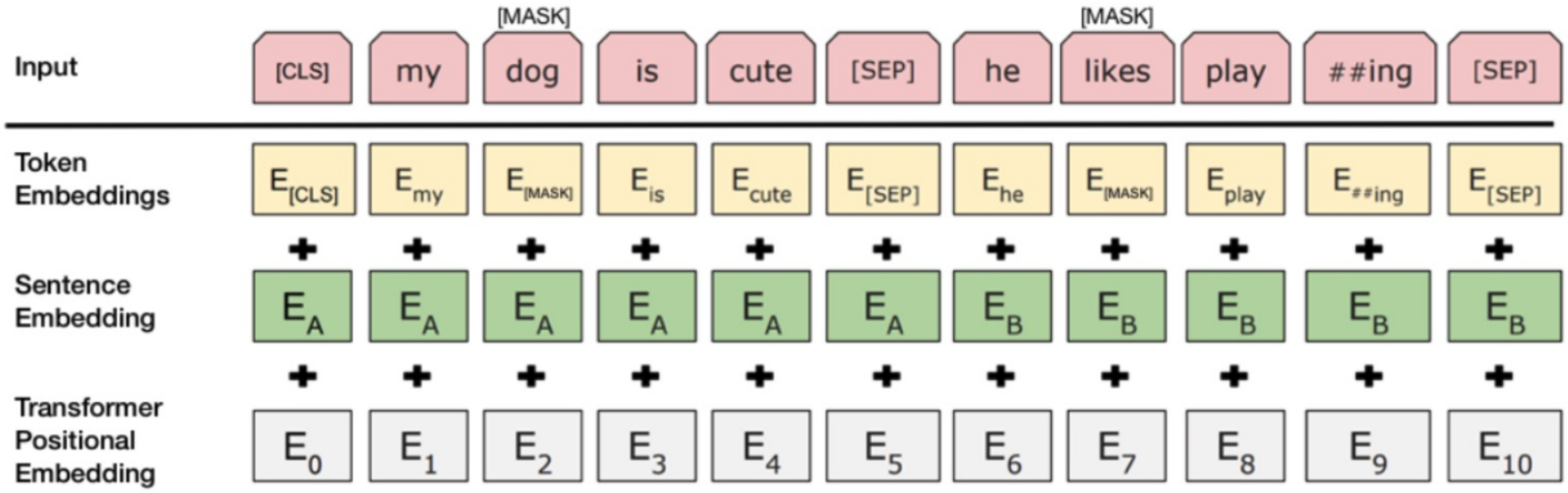
BERT Pre-training Tasks

- #1 Masked Language Model
- Pick 15% tokens and replace with
  - 80% [MASK], 10% random token, 10% the original one
  - Why not all [MASK]? [MASK] is not in fine-tuning process

$$P(w_i | w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n)$$

- #2 Next Sentence Prediction
- Each sequence is a concatenation of two sentences A & B
  - 50% B follows A
  - 50% B is a random sentence in the corpus
  - Binary classification

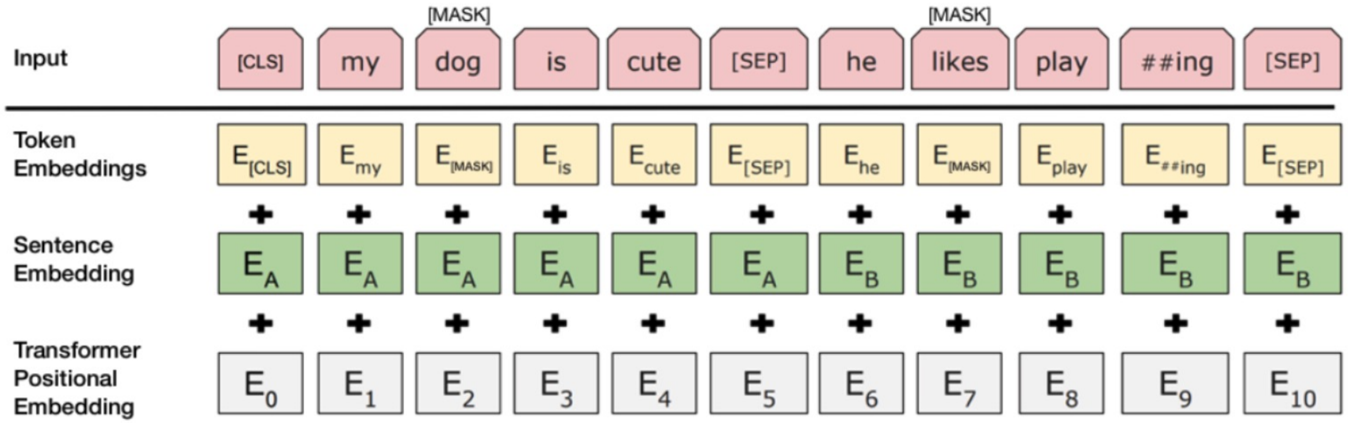
Joint Pre-training Input

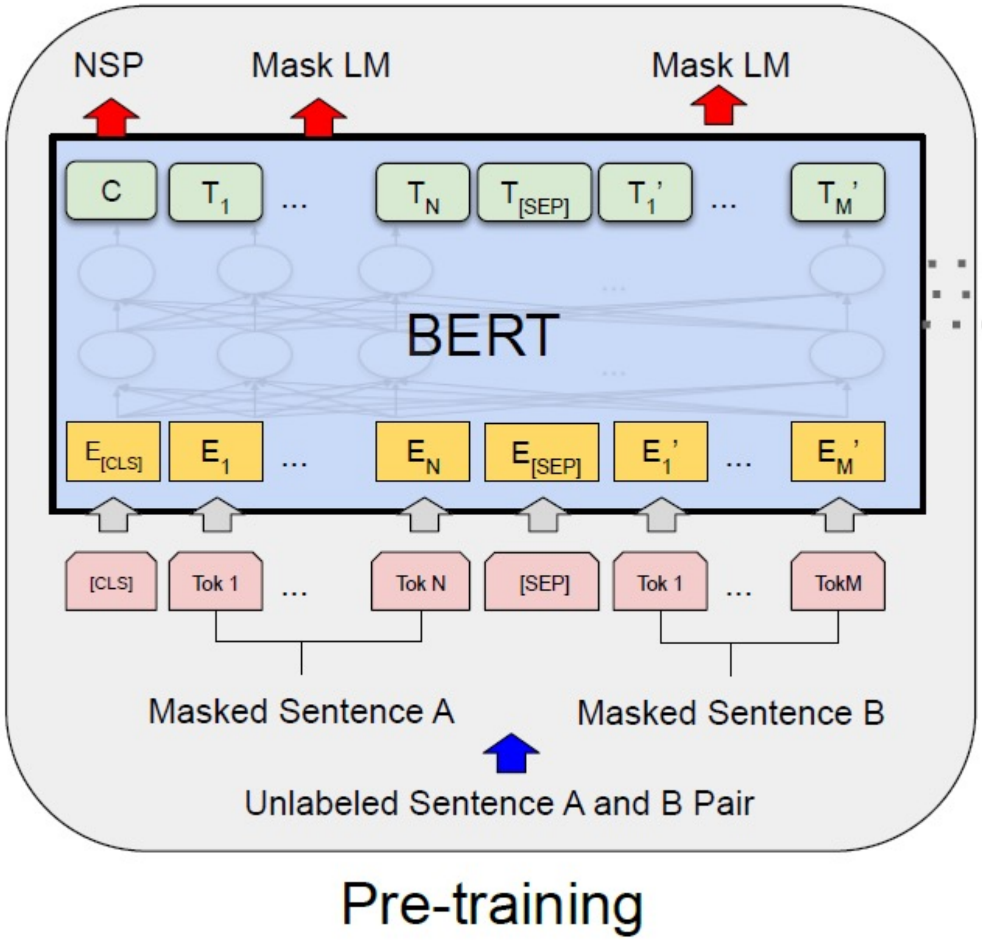


- [CLS]: prompter for next sentence prediction task
- [SEP]: sentence separater
- [MASK]: this word is masked due to Masked LM task

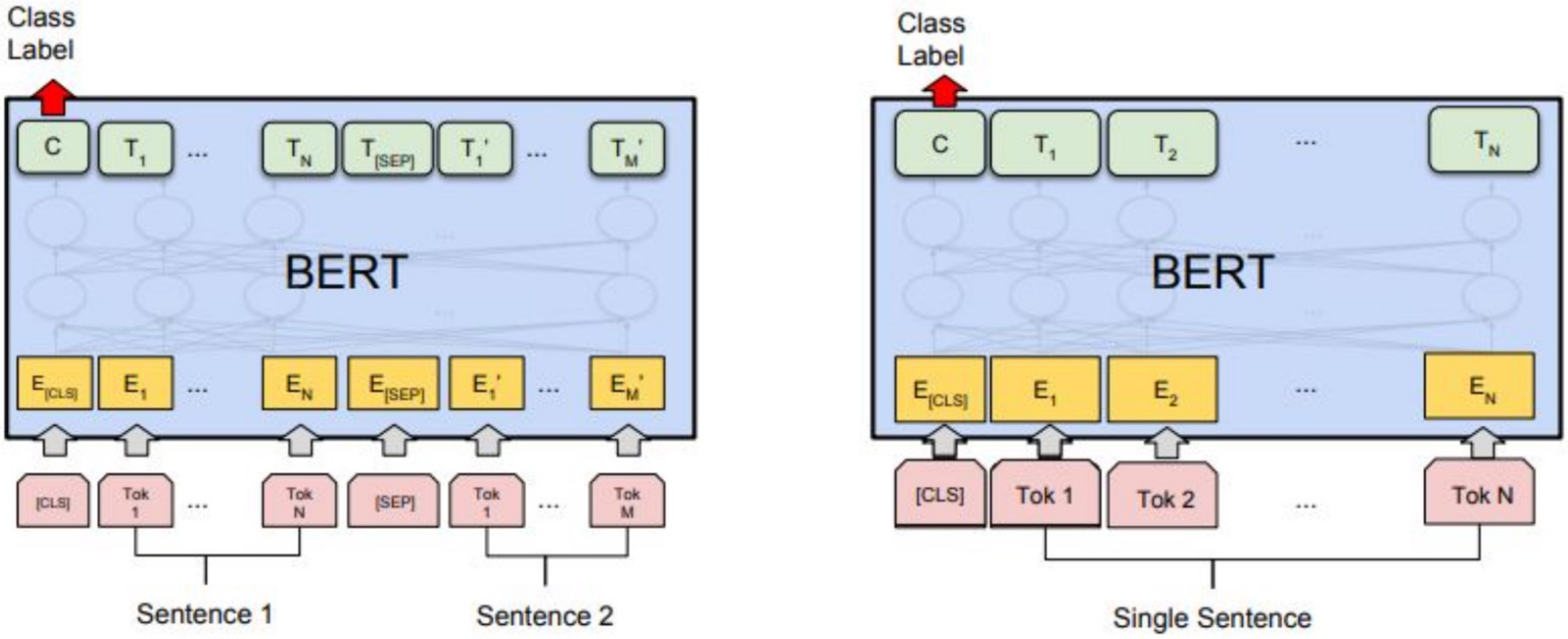
Joint Pre-training

- Next sentence prediction
  - Softmax @ [CLS] token
- Masked Language Model
  - Softmax @ any [MASK] token





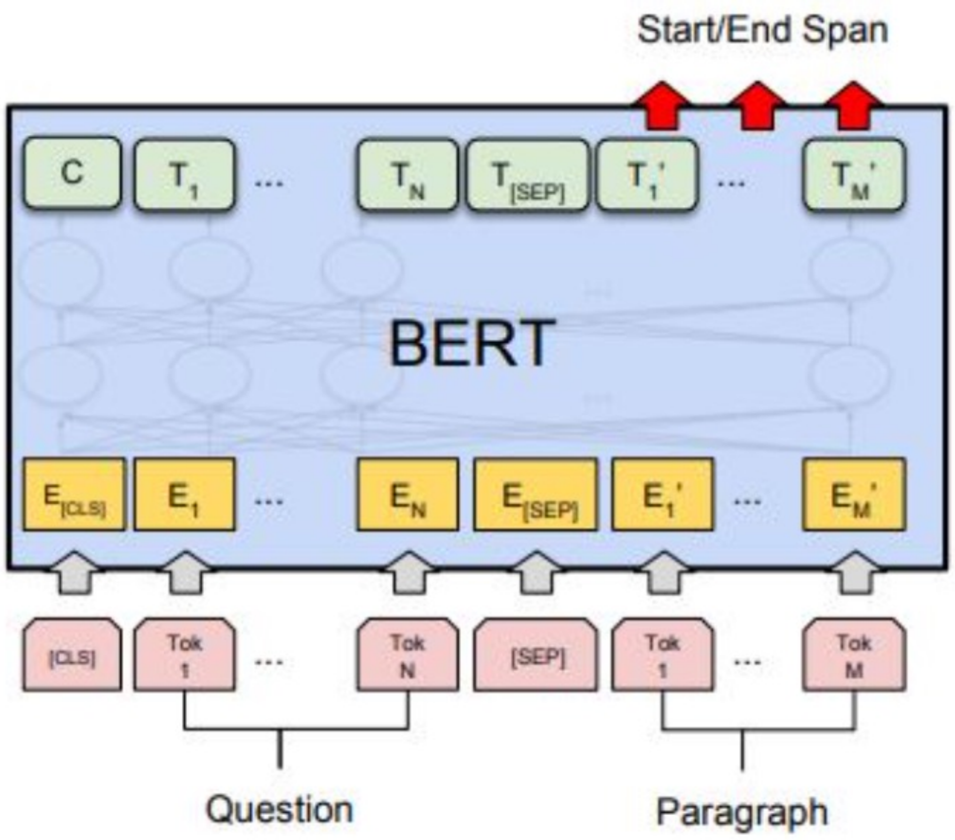
Fine-tuning on Downstream NLP Tasks



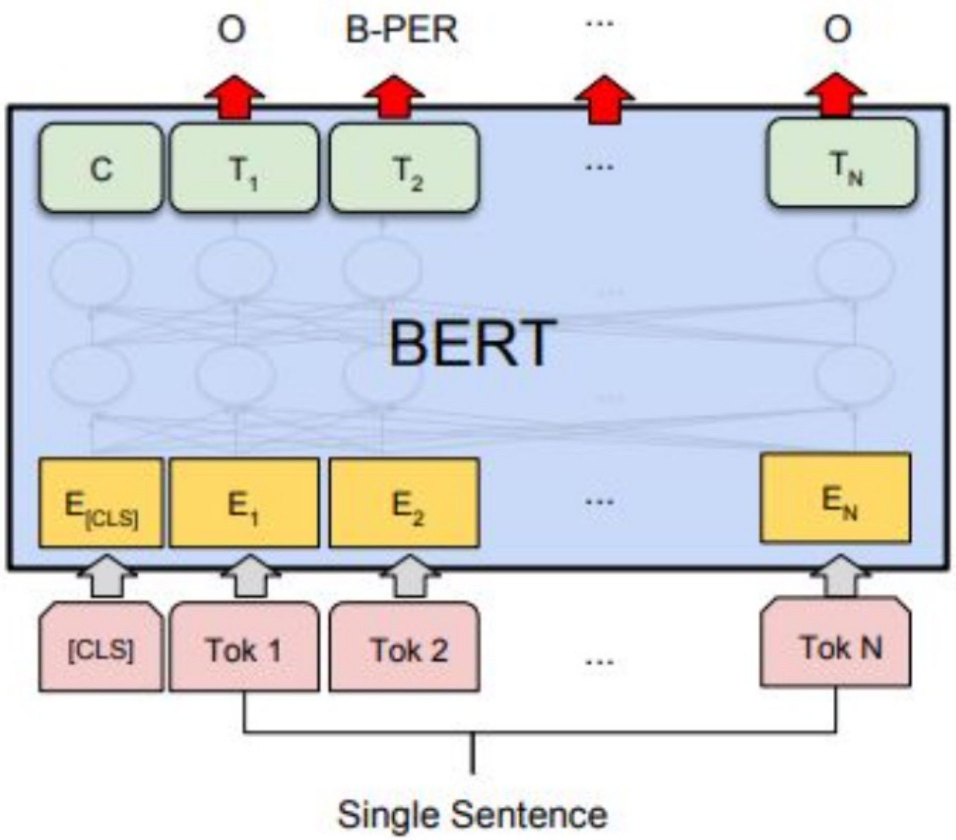
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

Fine-tuning on Downstream NLP Tasks



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

BERT Contributions

- #1 A deep model could also succeed in NLP

- #2 Representation Learning will be ubiquitous in NLP
  - Word2vec: one word, one embedding
  - BERT & more: context-aware embedding

- #3 Pre-training & Fine-tuning schema is substantiated
  - In fact, use pre-trained models whenever u can!

RoBERTa

- Robustly Optimized BERT Pre-training Approach
  - Shorter pre-train time, better downstream performance

- Change #1: Remove Next Sentence Prediction
  - Better performance on downstream tasks
- Change #2: Bigger batch sizes & longer sequences
  - Augment parallelization & end-task acc
- Change #3: Dynamically change [MASK] strategy
  - Re[MASK] every 4 epochs, 10 different [MASK]ed seqs in 40 epochs

GPT-3

- “Language Models are Few-shot Learners” by OpenAI
- 175 billion parameters, 12million $ for pre-training once

- Outperforms on Cloze, Q&A, Translation, Coreference Resolution, Reasoning, Reading Comprehension, Standard Test,…

- Few-shot, one-shot & zero-shot nature further substantiates the efficacy of pre-training to capture generic knowledge in text

Summary

- Self-supervised Learning
  - Learn to predict part of itself using the rest
- Pre-training & Fine-tuning
  - Pre-train captures generic knowledge, fine-tune for downstream
  - Architecture, Optimization tricks & More Signal
- BERT
  - Architertuce, Pre-Training Tasks, how to pre-train and fine-tune
- RoBERTa & GPT-3

Thx for Attention