Introduction to Deep Learning (IT3320E)

10 - Language Models

Hung Son Nguyen

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Nov. 22, 2022

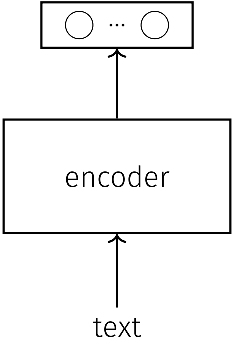
Agenda

1. Encoder-Decoder Models
2. BERT
3. NLP tasks
4. Deep learning prerequisites
5. Neural Machine Translation
6. Attention “is all you need”
7. Out-of-vocabulary words
8. Multi-task learning
9. “Unsupervised” Pre-Training
10. BERT
11. What do we know about how BERT works?

Section 1

Encoder-Decoder Models

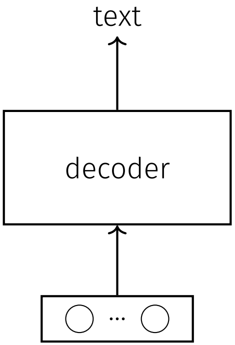
A neural model to transform a text into a vector in an embedding space.



Different types of neural encoders are

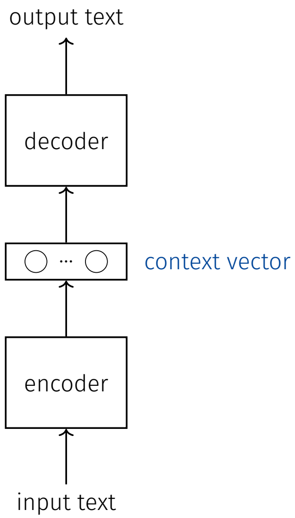
- pretrained word embeddings
- MLPs, CNNs, RNNs, Transformers, ...

A neural model to transform a vector from an embedding space to a text.

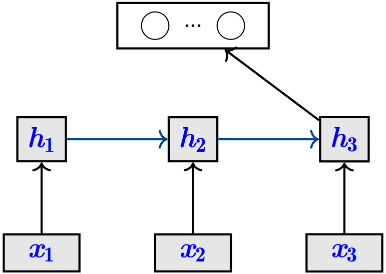


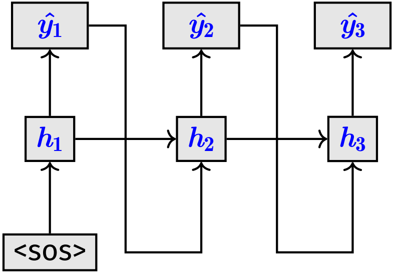
Different types of language models can be used as decoders

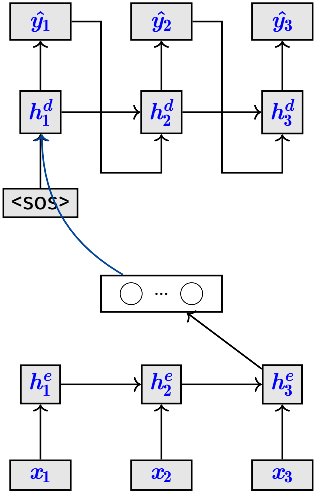
- RNN-based LMs
- Transformer-based LMs


output text
decoder
...
context vector
encoder
input text

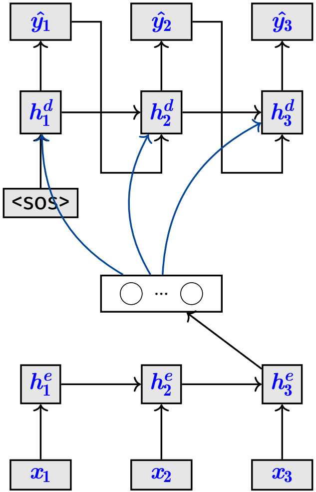
RNN Encoder




$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
h_1
h_2
h_3
<sos>

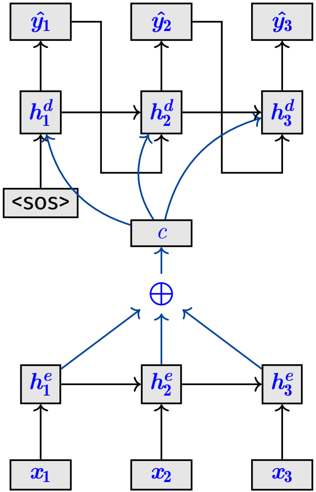

$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
h_1^d
h_2^d
h_3^d
<sos>
...
h_1^e
h_2^e
h_3^e
x_1
x_2
x_3

- The context vector is used to initialize the hidden state of the decoder.
- Its impact vanishes at the last steps of the decoder.



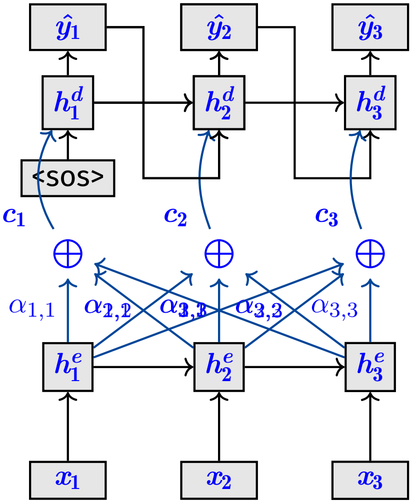
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
h_1^d
h_2^d
h_3^d
<sos>
...
h_1^e
h_2^e
h_3^e
x_1
x_2
x_3

- The output of the encoder is known as the context vector.
- The dimensionality of the context vector is fixed.
- However, different input texts might have different length.
- So, considering the hidden state of the RNN encoder may not capture the entire input text.
- This is a problem especially for long input texts.



$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
h_1^d
h_2^d
h_3^d
<SOS>
c
$\oplus$
h_1^e
h_2^e
h_3^e
x_1
x_2
x_3

- The other problem is that context vector is unique for all decoding steps.
- The encoder treats all tokens of the input sentence equally important to produce a context vector.
- However, at any decoding step, the decoder should focus on tokens of the input sentence differently.


$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
h_1^d
h_2^d
h_3^d
<SOS>
c_1
c_2
c_3
$\alpha_{1,1}$
$\alpha_{3,3}$
h_1^e
h_2^e
h_3^e
x_1
x_2
x_3

$$\boldsymbol{c_t} = \sum_{k=1}^{N} \alpha_{t,k} \boldsymbol{h_k^e}$$

$$\alpha_{t,k} = \frac{exp(\text{score}(\boldsymbol{h_{t-1}^d}, \boldsymbol{h_k^e}))}{\sum_{k'=1}^{N} exp(\text{score}(\boldsymbol{h_{t-1}^d}, \boldsymbol{h_{k'}^e}))}$$

$$\text{score}(\boldsymbol{h}_t^d, \boldsymbol{h}_k^e) = \boldsymbol{\mathtt{cosine}}(\boldsymbol{h}_t^d, \boldsymbol{h}_k^e)$$

Additive Attention (Bahdanau et al., 2015)

$$\mathrm{score}(\boldsymbol{h}_t^d, \boldsymbol{h}_k^e) = \mathit{tanh}\big([\boldsymbol{h}_t^d; \boldsymbol{h}_k^e]\,\boldsymbol{W}^{(h)}\big)\,\boldsymbol{W}^{(s)}$$

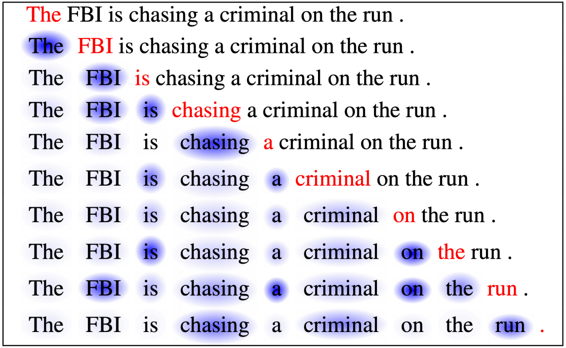
$$\text{score}(\boldsymbol{h}_t^d, \boldsymbol{h}_k^e) = \boldsymbol{softmax}\left(\boldsymbol{h}_t^d \boldsymbol{W}^{(s)}\right)$$

Scaled Dot-Product Attention (Vaswani et al., 2017)

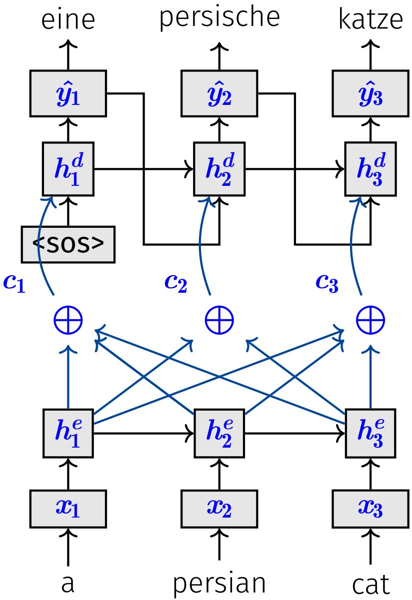
$$\text{score}(\boldsymbol{h}_t^d, \boldsymbol{h}_k^e) = \frac{\boldsymbol{h}_k^e \, \boldsymbol{trans}(\boldsymbol{h}_t^d)}{\sqrt{n}}$$

- The scaling factor $\frac{1}{\sqrt{n}}$ is motivated by the concern when the input is large, the softmax function may have an extremely small gradient.
- Small gradients yields difficulties in learning.

- An attention mechanism to relate different tokens of an input sequence to compute a representation of the sequence itself.
- For example, the self-attention mechanism enables a model to learn the relations between a word of an input sentence and its previous words.

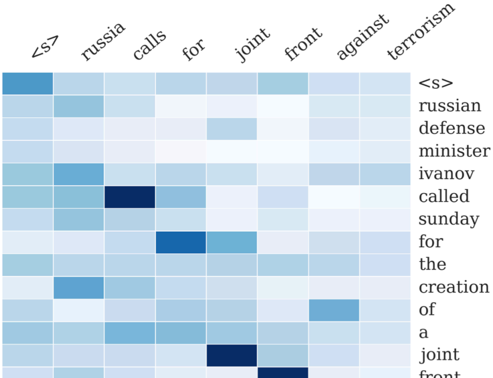


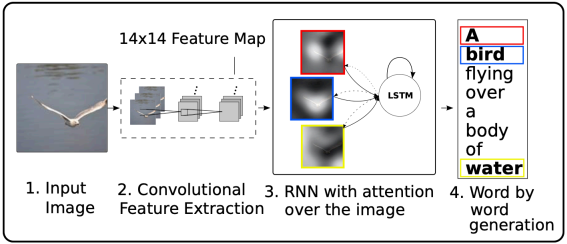
(Taken from Cheng et al., (2016))


eine
persische
katze
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
h_1^d
h_2^d
h_3^d
<sos>
c_1
c_2
c_3
h_1^e
h_2^e
h_3^e
x_1
x_2
x_3
a
persian
cat

Input $(\mathbf{x}_1, \ldots, \mathbf{x}_{18})$. First sentence of article:
russian defense minister ivanov called sunday for the creation of a joint front for combating global terrorism

Output $(\mathbf{y}_1, \ldots, \mathbf{y}_8)$. Generated headline:
russia calls for joint front against ***terrorism*** $\Leftarrow$ $g(\textit{terrorism}, \mathbf{x}, \textit{for, joint, front, against})$





14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

- lemmatization: g e s p i e l t $\longrightarrow$ s p i e l e n
- Spelling correction: i _ l v o e _ u $\longrightarrow$ i _ l o v e _ y o u

- An encoder-decoder model that transforms an input sequence to itself.
- It learns the identity function F(x) = x.
- It usually add some noise to the input, then the model learns to remove the noise.
- It is used for dimensionality reduction, representation learning, and unsupervised learning.
- The encoder and decoder can be used individually to solve other tasks.

- Encoders and Decoders
- Attention mechanism
- Their applications in NLP

Section 2

BERT

BERT[1] — The ”gamechanger”

Best paper award at NAACL 2019

State-of-the-art results on various NLP tasks

Directly applicable to other domains and languages

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

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language repre-

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that

[1]Devlin.et.al.2019.NAACL

Figure 2: To spice up the lectures, the lecturer is dressed in an ELMo costume

artemova-etal-2021-teaching

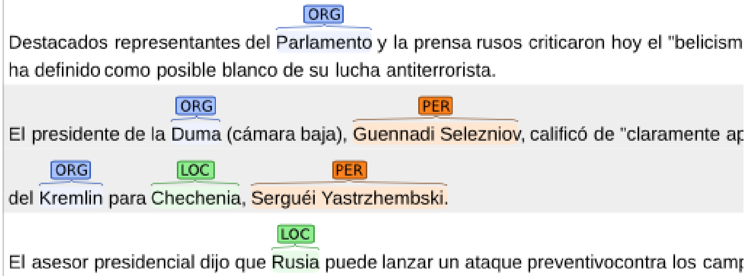
Section 3

NLP tasks

Single-sentence ”tagging” tasks, such as

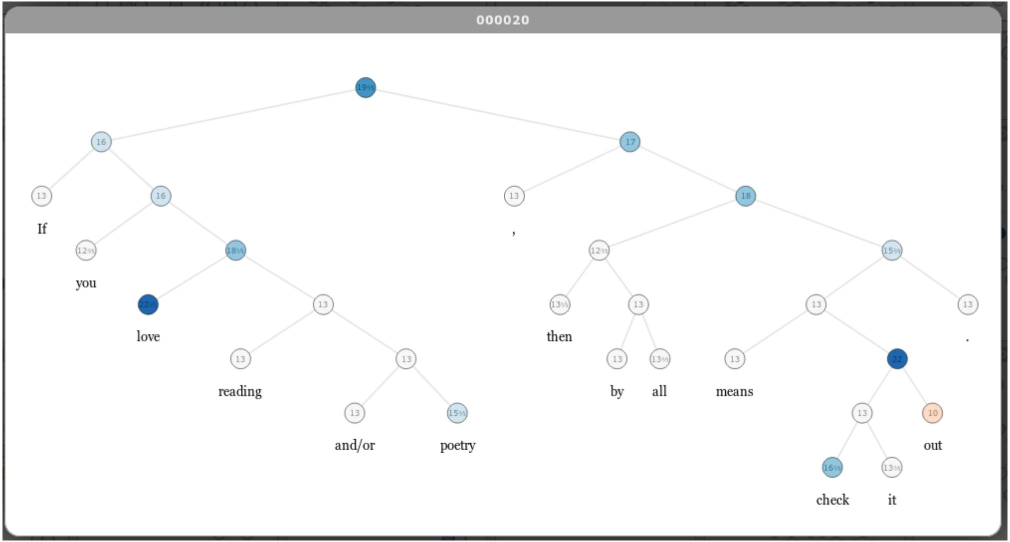
– Part of speech tagging (not in BERT paper)

– Named Entity Recognition



Short recap of "NLP tasks" – Single sentence

– Sentiment of a sentence[2]



[2] https://nlp.stanford.edu/sentiment/treebank.html

Reasoning about two sentences: Natural Language Inference[3]

Text	Judgments	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction C C C C C	The man is sleeping
An older and younger man smiling.	neutral N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	contradiction C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	entailment E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	neutral N N E C N	A happy woman in a fairy costume holds an umbrella.

[3] https://nlp.stanford.edu/projects/snli/

Question answering

– Natural language questions with locations of their answers in Wikipedia articles

Although some methods can ”exploit” artifacts in data,[4] the tasks can be truly solved only by

- Understanding meaning of words (semantics)
- Understanding relations between meanings
- Understanding syntax (negations, quantifiers, etc.)
- Reasoning about the world

[4] Gururangan.et.al.2018.NAACL.short

NLP tasks ✓

- Long-range dependencies, hard to represent meaning

Section 4

Deep learning prerequisites

We know the ”nail”, let’s take the hammer

Prerequisites: We know

- Neural network basics (layers, activations, softmax, convolutions)
- Where are the learnable parameters (”weight matrices” and biases),
- What are loss functions (e.g., cross-entropy for classification)
- How to train them (back-propagation, batches, SGD or Adam)
- Word embeddings (dense semantic representation)

NLP tasks ✓

- Long-range dependencies, hard to represent meaning

Neural networks ✓

- Learn non-linear dependencies, learn representations

Embeddings ✓

- Dense token representation

Section 5

Neural Machine Translation

Why machine translation here?

BERT builds upon techniques from MT

What is machine translation?

- Another popular NLP task
- Many large-scale parallel corpora available

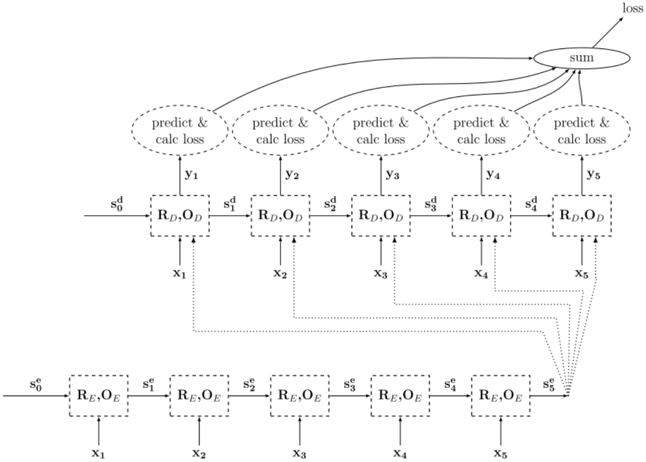
MT is a challenging task!

Image source: https://languagelog.ldc.upenn.edu/nll/?p=3978

Traditionally **encoder-decoder** architectures

- One recurrent neural network processes the entire input and generate its dense representation (**encoder**)
- Other recurrent network produces one token at the time conditioned on the previous states and generated tokens (**decoder**)

Long short-term memory (LSTM) / GRU networks



Encoder-decoder RNN.[5]

Inherently **sequential** nature

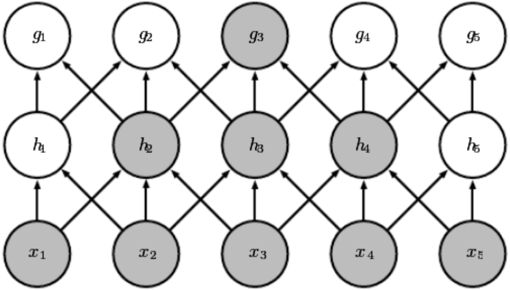
- No parallelization
- Big memory footprint (you must ”remember” the entire sequence)
- Long-range dependencies modeling: Distance plays a role!

...but when the goal is to learn a good representation of the input sequence, why not use...

- Convolutional neural networks?

One particular property of CNNs

- Modeling dependencies for a **local context**, but by **stacking layers**, one exactly controls the context size



Receptive field of units in deeper layers is larger. Source: **Goodfellow.et.al.2016.book**

CNNs competitive with RNNs for MT[6]

- Input tokens as word embeddings (not new) or sub-words (will be explained later)
- Fixed-length input? Set-up a maximum length and use *<PAD>*ding
- But positional information of tokens is lost...

[6] **Gehring.et.al.2017a.ICML** (from Facebook AI Research)

Solution: Positional embeddings

- For each input position n, train another embedding vector P_n:
  $P_1 = (1.12, -78.6, \dots), P_2, \dots, P_N$
- Word embeddings and position embeddings are simply summed up for each input token
- Why? The model knows with which part of the input/output is dealing with
  - Notice: Removing positional embeddings $\rightarrow$ only slightly worse performance

State-of-the-art results and **9.3–21.3$\times$ faster** than LSTMs on GPU

NLP tasks ✓

- Long-range dependencies, hard to represent meaning

Neural networks ✓

- Learn non-linear dependencies, learn representations

Embeddings ✓

- Dense token representation

Neural machine translation ✓

- Sequence to sequence models
- Positional embeddings

Section 6

Attention “is all you need”

Recap: How to model long-range dependencies in input?

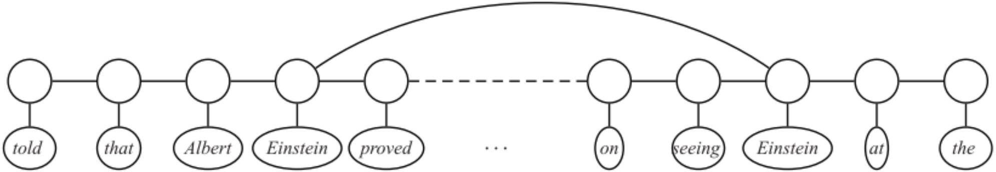
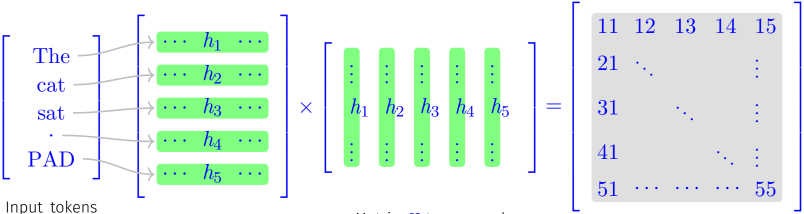


Figure 1: An example of the label consistency problem. Here we would like our model to encourage entities *Albert Einstein* and *Einstein* to get the same label, so as to improve the chance that both are labeled *PERSON*.

- RNNs or stacking CNNs
- **Self-Attention**: Utilize associations between all input word pairs

Figure source: **Krishnan.Manning.2006**



Input tokens

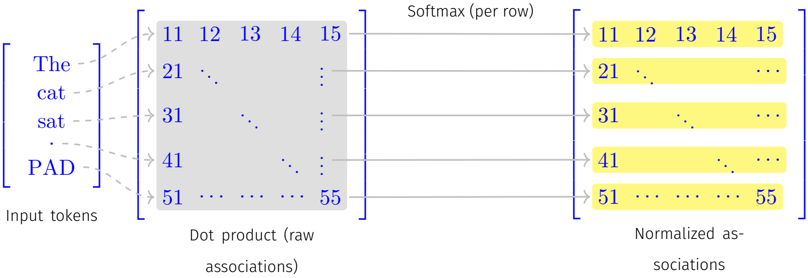
Matrix H

(latent representation)

Matrix H transposed

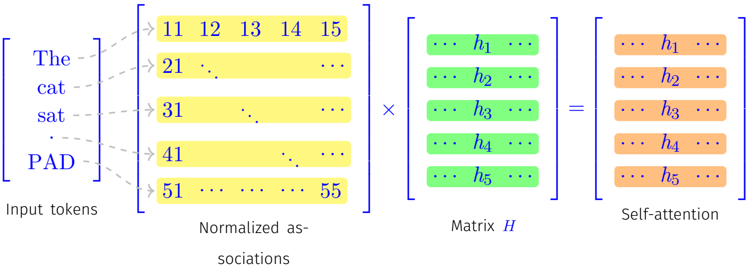
Dot product (raw associations)

Scale each row by $\sqrt{|h|}$



- Each row corresponds to an input token
- Each row sums up to 1
- Each cell shows the ”association strength” with all other tokens

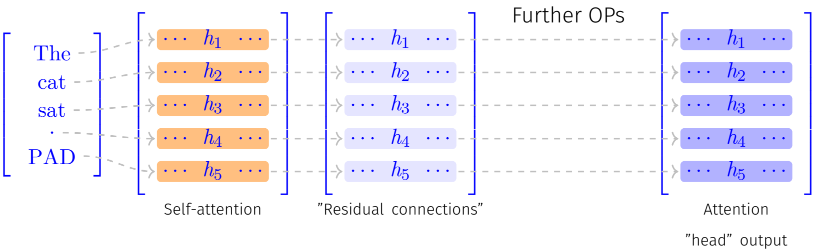
Self-Attention in detail (3)



Each position in the latent representation of a token is weighted by the association strength with other tokens

Self-Attention in detail (4)

$$\underbrace{\begin{bmatrix} \cdots & h_1 & \cdots \\ \cdots & h_2 & \cdots \\ \cdots & h_3 & \cdots \\ \cdots & h_4 & \cdots \\ \cdots & h_5 & \cdots \end{bmatrix}}_{\text{Self-attention}} + \underbrace{\begin{bmatrix} \cdots & h_1 & \cdots \\ \cdots & h_2 & \cdots \\ \cdots & h_3 & \cdots \\ \cdots & h_4 & \cdots \\ \cdots & h_5 & \cdots \end{bmatrix}}_{\text{Matrix } H} = \underbrace{\begin{bmatrix} \cdots & h_1 & \cdots \\ \cdots & h_2 & \cdots \\ \cdots & h_3 & \cdots \\ \cdots & h_4 & \cdots \\ \cdots & h_5 & \cdots \end{bmatrix}}_{\text{"Residual connections"}}$$



Further operations

- Layer normalization
- Feed-forward layer with ReLU
- Another residual connection and layer normalization

- Run N attention ”heads” in parallel and concatenate
- Stack on top of each other M-times

Why self-attention?

- Self-attention layer connects all positions with a constant number of sequentially executed operations
- Recurrent layer requires O(n) sequential operations
- Self-attention layers are **fast**

Vaswani.et.al.2017

NLP tasks ✓

- Long-range dependencies, hard to represent meaning

Neural networks ✓

- Learn non-linear dependencies, learn representations

Embeddings ✓

- Dense token representation

Neural machine translation ✓

- Sequence to sequence models
- Positional embeddings

Attention ✓

- Efficient long-range dependencies

Section 7

Out-of-vocabulary words

- MT often with fixed word vocabularies
  - Even though translation is fundamentally an open vocabulary problem (names, numbers, dates etc.).
  - Initially, the most frequent words were used, and all others *<UNK>*[7]
- Translation of out-of-vocabulary (OOV) words
  - Rare words (OOV) handled with a back-off dictionary, or simply copied 1:1 from source to target

[7] Koehn.2017

Sub-words for voice search (Japanese, Korean)[8]

- Too large vocabularies for these two languages would produce way too many OOVs

Later known as WordPiece model

- Adapted by Google's Neural Machine Translation[9] and eventually by BERT

[8] Schuster.Nakajima.2012
[9] Wu.et.al.2016.GoogleMT

But why should sub-word units give better translations than copying or back-off dictionary?

– Open-vocabulary MT better by representing rare and unseen words as a sequence of subword units[10]

[10] Sennrich.et.al.2016.ACL

What are WordPiece units?

- Similar to a vocabulary: A list of all known (sub-)words, including characters
  - Each word is either entirely a WordPiece unit, or can be split into several WordPiece units
- Splitting a text into the trained WordPiece model shipped along with BERT:

```
tokenizer.tokenize("All human beings are born free and equal in
dignity and rights.")
['all', 'human', 'beings', 'are', 'born', 'free', 'and', 'equal',
'in', 'dignity', 'and', 'rights', '.']
```

WordPiece units: Multilingual

- print(tokenizer.tokenize("Alle Menschen sind frei und gleich an Würde und Rechten geboren."))
- ['all', '##e', 'men', '##schen', 'sin', '##d', 'fr', '##ei', 'und', 'g', '##lei', '##ch', 'an', 'wu', '##rde', 'und', 'rec', '##ht', '##en', 'ge', '##bor', '##en', '.']
- BERT WordPiece tokenizer: Lower casing, punctuation removal
- More languages?
- tokenizer.tokenize("Все люди рождаются свободными и равными в своем достоинстве и правах.")
  tokenizer.tokenize("Všichni lidé se rodí svobodní a sobě rovní co do důstojnosti a práv.")
  tokenizer.tokenize("ყველა ადამიანი იბადება თავისუფალი და თანასწორი თავისი ღირსებითა და უფლებებით.")

- ['в', '##с', '##е', 'л', '##ю', '##д', '##и', 'р', '##о', '##ж', '##д', '##а', '##ю', '##т', '##с', '##я', 'с', '##в', '##о', '##б', '##о', '##д', '##н', '##ы', '##м', '##и', 'и', 'р', '##а', '##в', '##н', '##ы', '##м', '##и', 'в', 'с', '##в', '##о', '##е', '##м', 'д', '##о', '##с', '##т', '##о', '##и', '##н', '##с', '##т', '##в', '##е', 'и', 'п', '##р', '##а', '##в', '##а', '##х', '.']
  ['vs', '##ich', '##ni', 'lid', '##e', 'se', 'rod', '##i', 'sv', '##ob', '##od', '##ni', 'a', 'sob', '##e', 'ro', '##vn', '##i', 'co', 'do', 'dust', '##oj', '##nos', '##ti', 'a', 'pr', '##av', '.']
  ['[UNK]', 'ა', '##დ', '##ა', '##მ', '##ი', '##ა', '##ნ', '##ი', 'ი', '##ბ', '##ა', '##დ', '##ე', '##ბ', '##ა', '[UNK]', 'დ', '##ა', '[UNK]', 'თ', '##ა', '##ვ', '##ი', '##ს', '##ი', '[UNK]', 'დ', '##ა

Training WordPiece inventory

1. Init the WordPiece inventory with all characters (in all alphabets)
2. For each possible tuple of known WordPieces
   - Create a new candidate WordPiece from the tuple (simply concatenate)
   - Build a language model and compute likelihood on the corpus
3. Select the candidate with the maximum likelihood increase and add to the WordPiece inventory; Go back to 2 or finish, if WordPiece inventory has the desired size

Schuster.Nakajima.2012

NLP tasks ✓

- Long-range dependencies, hard to represent meaning

Neural networks ✓

- Learn non-linear dependencies, learn representations

Embeddings ✓

- Dense token representation

Neural machine translation ✓

- Sequence to sequence models
- Positional embeddings

Attention ✓

- Efficient long-range dependencies

Out-of-vocabulary words ✓

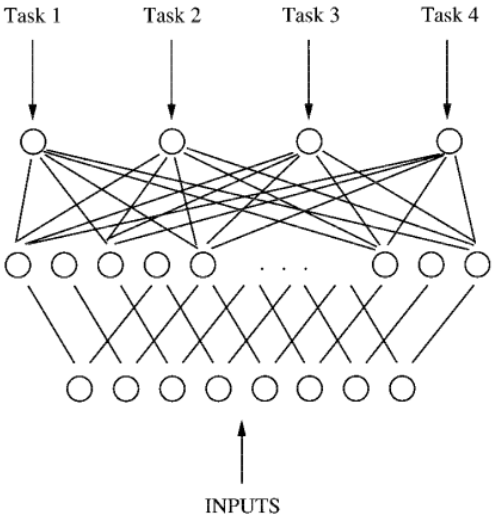
- WordPiece sub-word units can be truly multi-lingual and prevent OOV

Section 8

Multi-task learning

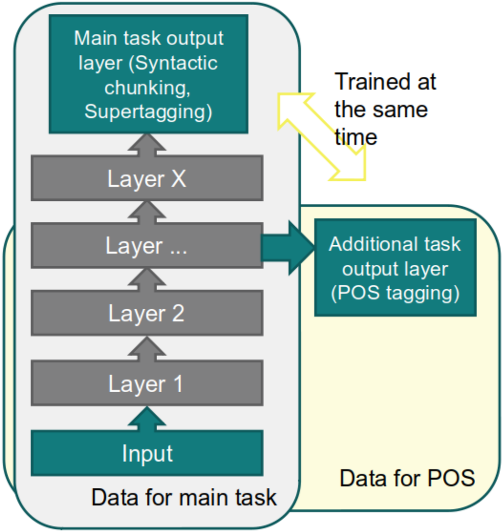
Approach to inductive transfer that improves generalization

By learning tasks in parallel while using a shared representation



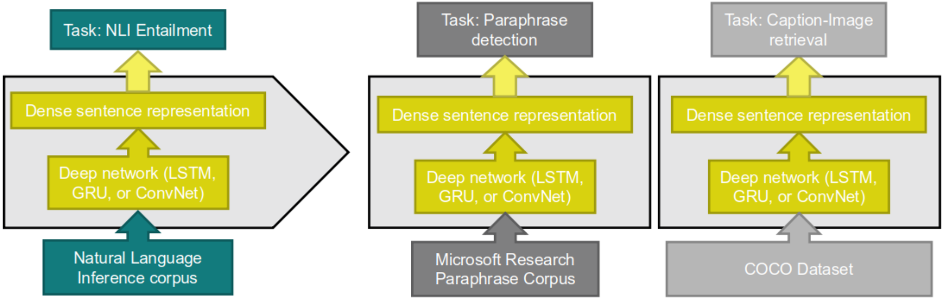
Caruana.1997

"In case we suspect the existence of a hierarchy between the different tasks, we show that it is worth-while to incorporate this knowledge in the MTL architecture's design, by making lower level tasks affect the lower levels of the representation."



Sogaard.Goldberg.2016

Learn a sentence representation on a different task



"Models learned on NLI can perform better than models trained in unsupervised conditions or on other supervised tasks."[11]

[11] Conneau.et.al.2017.EMNLP

NLP tasks ✓

- Long-range dependencies, hard to represent meaning

Neural networks ✓

- Learn non-linear dependencies, learn representations

Embeddings ✓

- Dense token representation

Neural machine translation ✓

- Sequence to sequence models
- Positional embeddings

Attention ✓

- Efficient long-range dependencies

Out-of-vocabulary words ✓

- WordPiece sub-word units can be truly multi-lingual and prevent OOV

Multi-task learning ✓

- Shared representation improves generalization; transfer learning

Section 9

“Unsupervised” Pre-Training

”Un-supervised” Pre-Training

- Deep neural nets are trained with full supervision
  - Even autoencoders are supervised by the reconstruction error
- ”Unsupervised” training scenario usually means:
  - I don’t have any labeled data for my target task (e.g., no labels for ”word similarity”)
  - But I can design a proxy supervised task (e.g., ”given a context of a missing word, predict that word”)
  - And create positive and negative instances by exploiting a large unlabeled corpus (e.g., words in their context as positive, and randomly swapped words with their context as negative)

NLP tasks ✓

- Long-range dependencies, hard to represent meaning

Neural networks ✓

- Learn non-linear dependencies, learn representations

Embeddings ✓

- Dense token representation

Neural machine translation ✓

- Sequence to sequence models
- Positional embeddings

Attention ✓

- Efficient long-range dependencies

Out-of-vocabulary words ✓

- WordPiece sub-word units can be truly multi-lingual and prevent OOV

Multi-task learning ✓

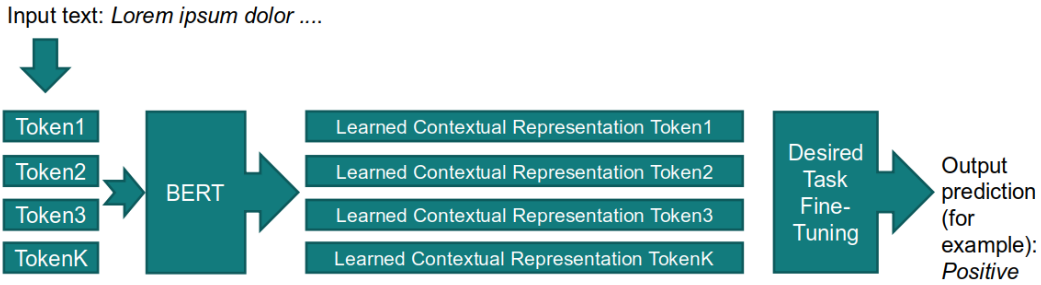
- Shared representation improves generalization; transfer learning

”Unsupervised” Pre-Training ✓

- Proxy task and unlimited data from unlabeled corpora

Section 10

BERT


Input text: *Lorem ipsum dolor ....*
Token1
Token2
Token3
TokenK
BERT
Learned Contextual Representation Token1
Learned Contextual Representation Token2
Learned Contextual Representation Token3
Learned Contextual Representation TokenK
Desired Task Fine-Tuning
Output prediction (for example): *Positive*

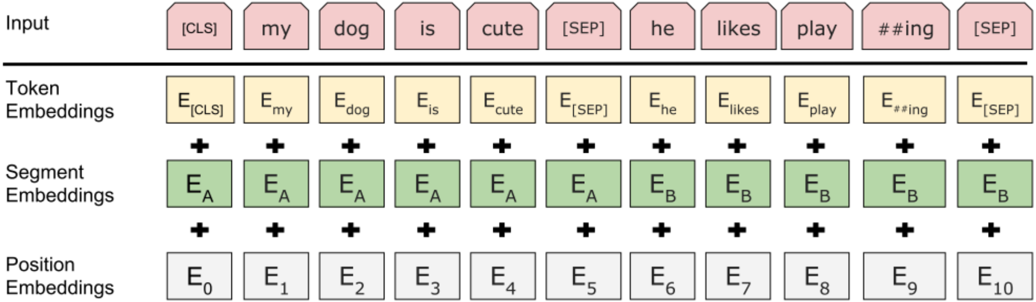
BERT: Tokenization

Tokenizing into a multilingual WordPiece inventory

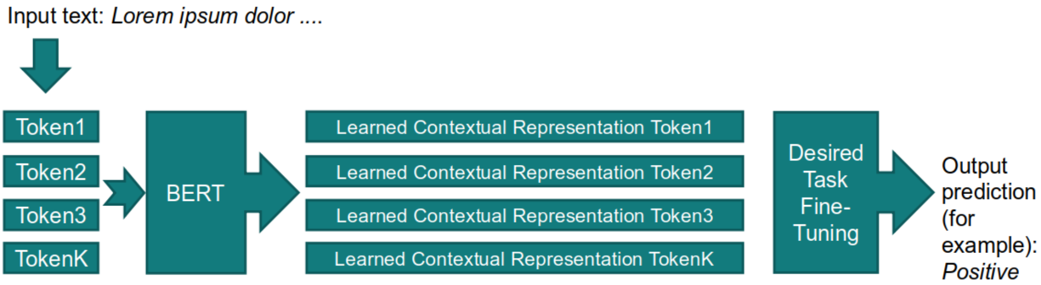
- Recall that WordPiece units are sub-word units
- 30,000 WordPiece units (newer models 110k units, 100 languages)

Implications: BERT can ”consume” any language

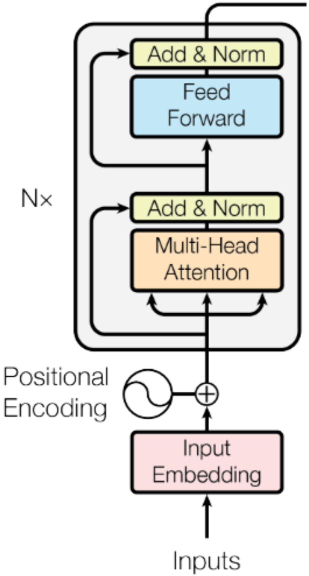
- Each WordPiece token from the input is represented by a **WordPiece embedding** (randomly initialized)
- Each position from the input is associated with a **positional embedding** (also randomly initialized)
- Input length limited to **512** WordPiece tokens, using *<PAD>*ding
- Special tokens
    - The fist token is always a special token **[CLS]**
    - If the task involves two sentences (e.g., NLI), these two sentences are separated by a special token **[SEP]**; also special two **segment position embeddings**


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}

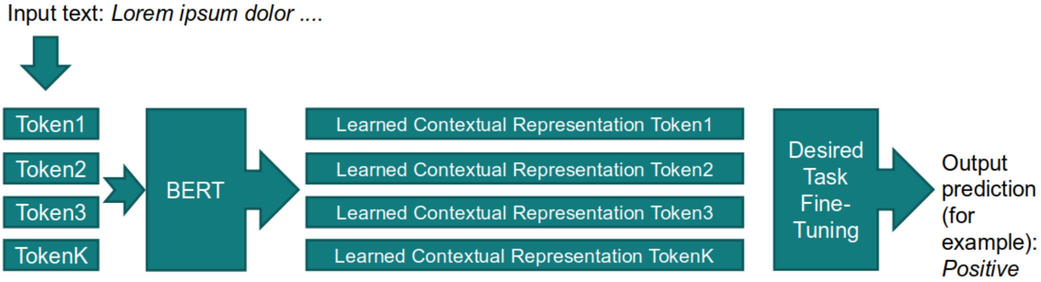
BERT: Very abstract view



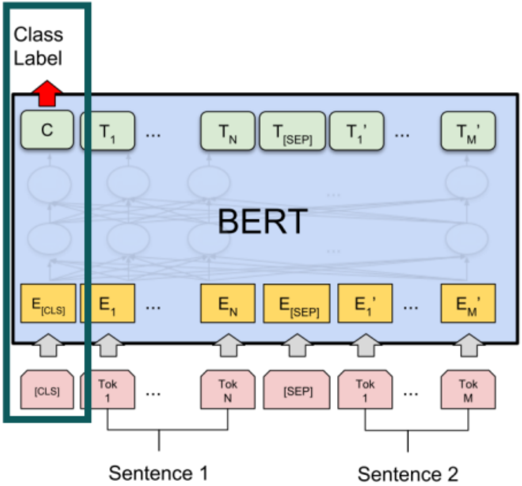
- The good-old-friend ”Self-Attention”
- Multiple parallel attention ”heads” (16 heads)
- With residual connections
- With layer normalization
- Stacked on top of each other (24-times)
- 310,000,000 trainable parameters
- ...we’ve seen that already



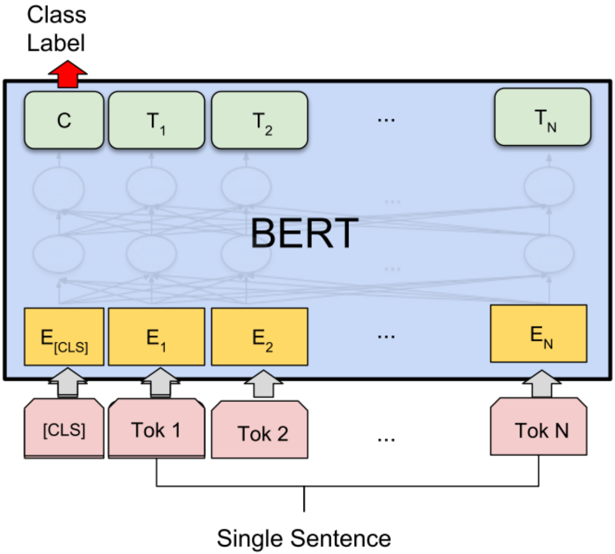
BERT: Very abstract view



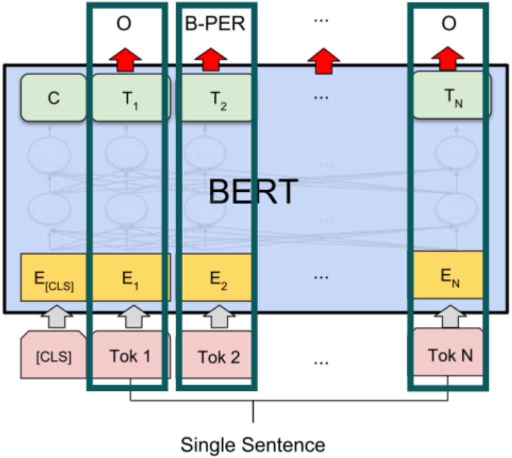
BERT: Representing various NLP tasks



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



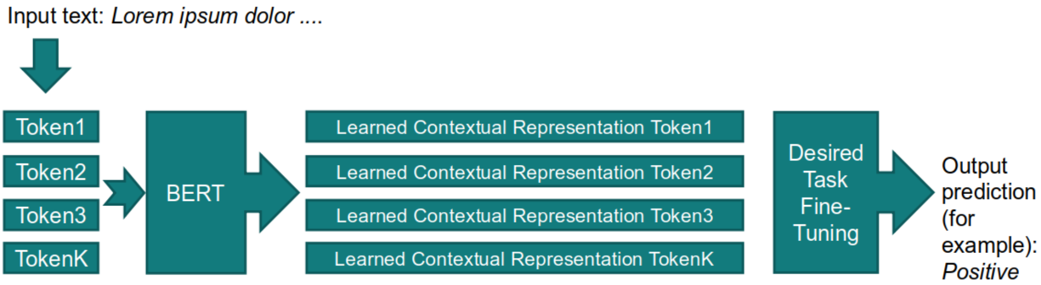
(b) Single Sentence Classification Tasks: SST-2, CoLA



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

Not conditioned on surrounding predictions

BERT: Very abstract view



BERT: "Unsupervised" multi-task pre-training

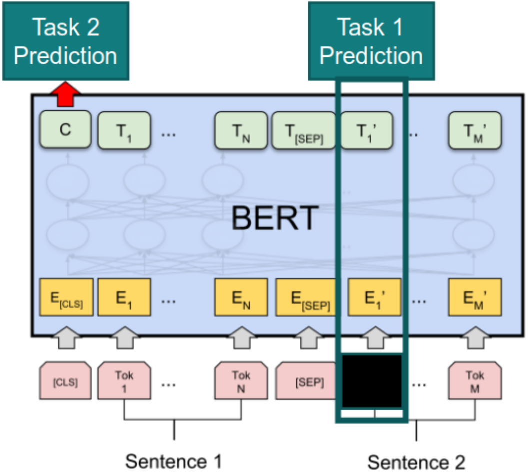
Prepare two auxiliary tasks that need no labeled data

Task 1: Cloze-test task

- Predict the masked WordPiece unit (multi-class, 30k classes)

Task 2: Consecutive segment prediction

- Did the second text segment appeared after the first segment? (binary)



Take the entire Wikipedia (in 100 languages; 2,5 billion words)

To generate a single training instance, sample two segments (max combined length 512 WordPiece tokens)

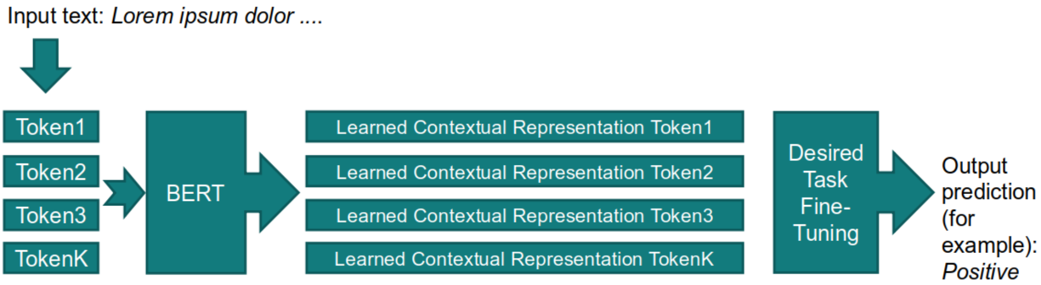
- For Task 2, replace the second segment randomly in 50% (negative samples)
- For Task 1, choose random 15% of the tokens, and in 80% replace with a [MASK]

BERT: Pre-training data – Simplified example

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

- <PAD>ding is missing
- The actual segments are longer and not necessarily actual sentences (just spans)
- The WordPiece tokens match full words / morphology well in this English text, but recall the ones we have seen before



Pretraining this monster took them 4 days on 64 TPU chips (estimated $500 USD)

Once pre-trained, transfer and ”fine-tune” on your small-data task and get state-of-the-art results :)

NLP tasks ✓

- Long-range dependencies, hard to represent meaning

Neural networks ✓

- Learn non-linear dependencies, learn representations

Embeddings ✓

- Dense token representation

Neural machine translation ✓

- Sequence to sequence models
- Positional embeddings

Attention ✓

- Efficient long-range dependencies

Out-of-vocabulary words ✓

- WordPiece sub-word units can be truly multi-lingual and prevent OOV

Multi-task learning ✓

- Shared representation improves generalization; transfer learning

”Unsupervised” Pre-Training ✓

- Proxy task and unlimited data from unlabeled corpora

BERT stays on the shoulders of many clever concepts and techniques, mastered into a single model

Section 11

What do we know about how BERT works?

Highly recommended reading

"BERTology has clearly come a long way, but it is fair to say we still have more questions than answers about how BERT works."

Rogers.et.al.2020.BERT