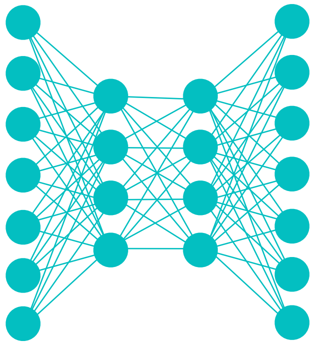
Lecture Notes for

Neural Networks and Machine Learning

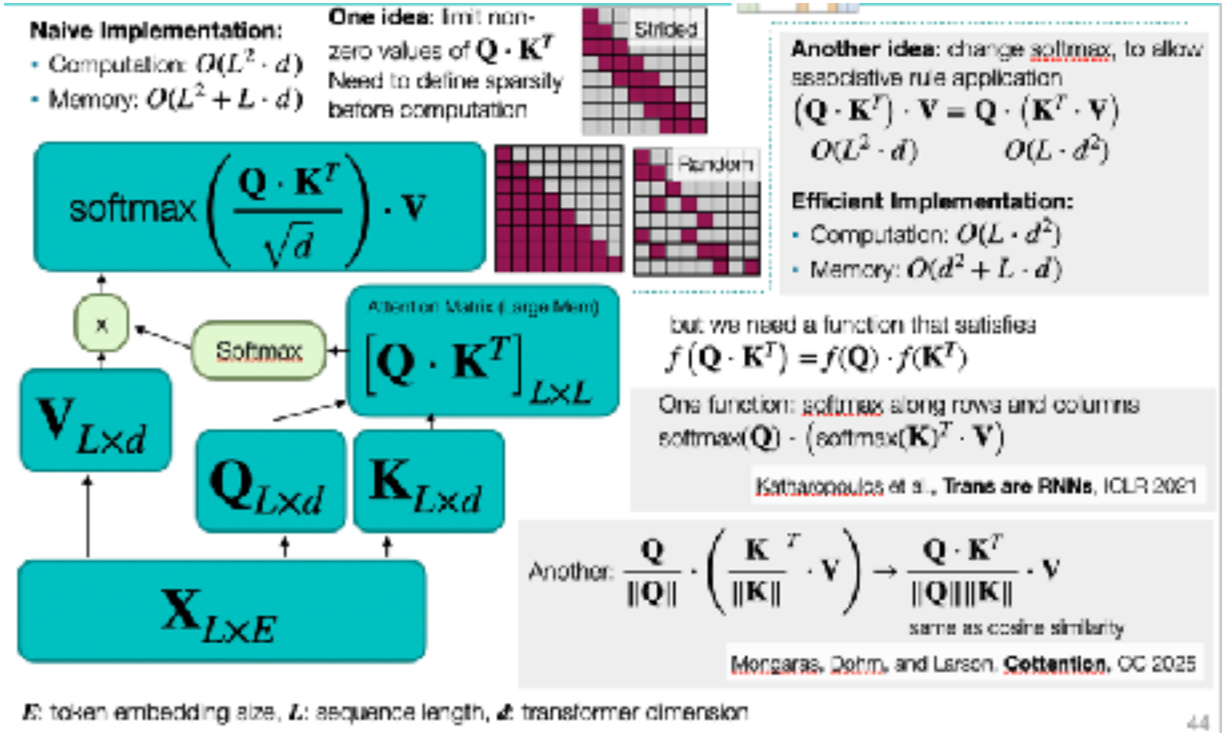
Practical Transformers

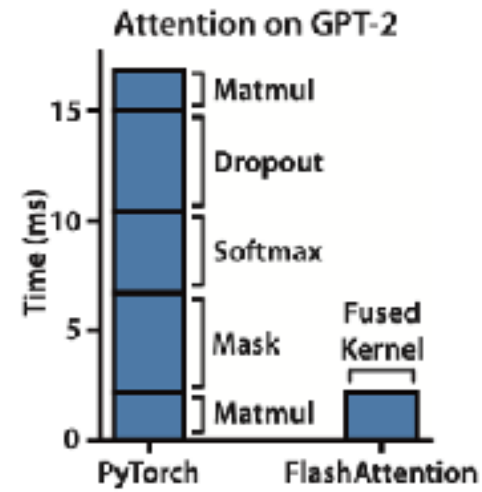


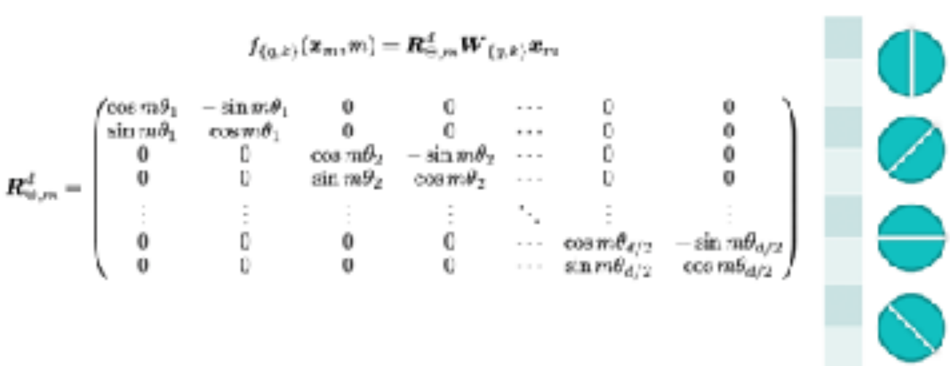
Logistics and Agenda

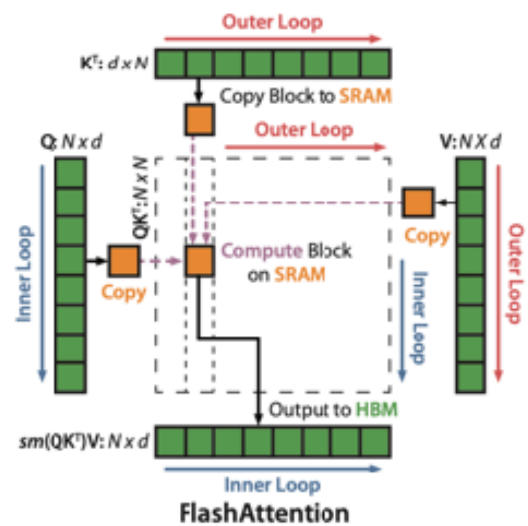
- Logistics
  - None!
- Agenda (probably two lectures)
  - Positional Encoding Review
  - Student Paper Presentation
  - Common Transformers

Last Time: Transformers

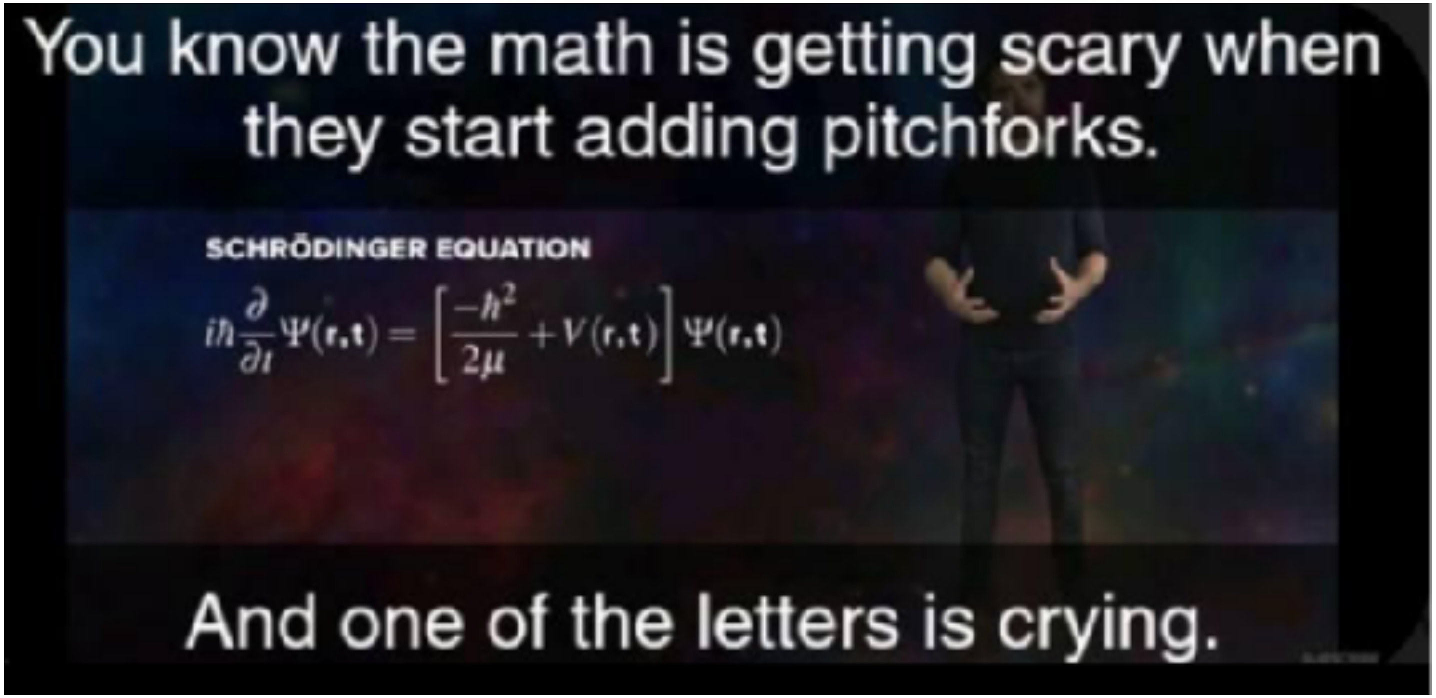






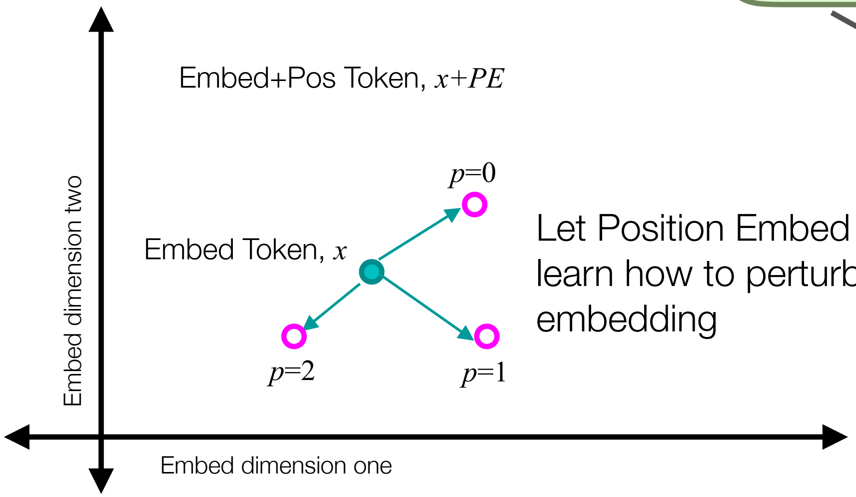


Position Encode Review

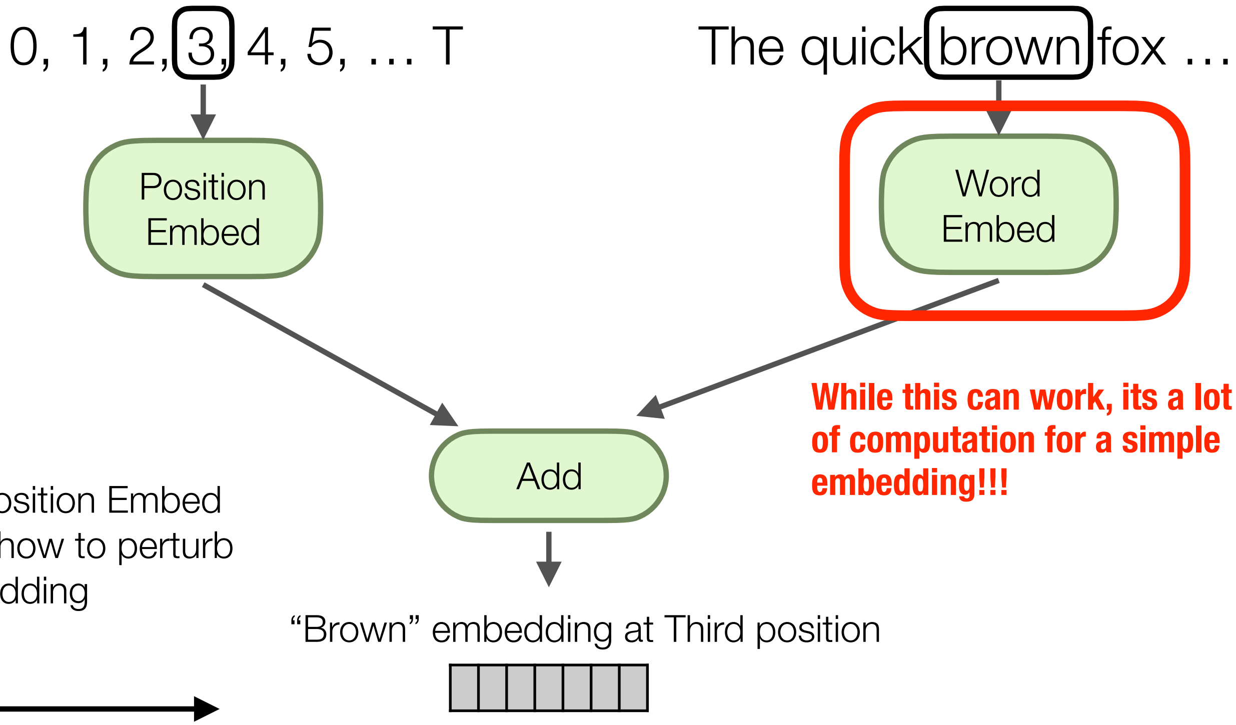


Transformer: Positional Embedding

- Objective: add notion of position to embedding
- Attempt in original paper: add sin/cos to embedding
- **But could be anything that encodes position, like:**



Let Position Embed learn how to perturb embedding



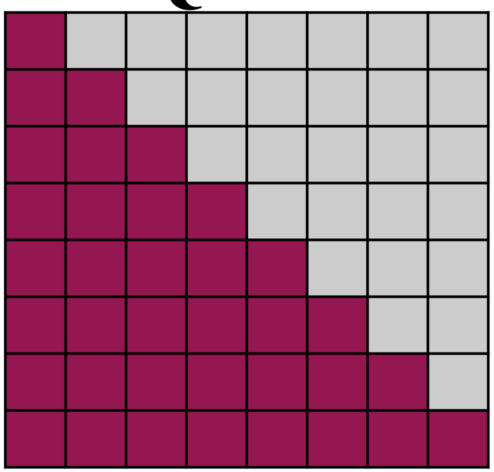
While this can work, its a lot of computation for a simple embedding!!!

Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Relative Positional Encoding

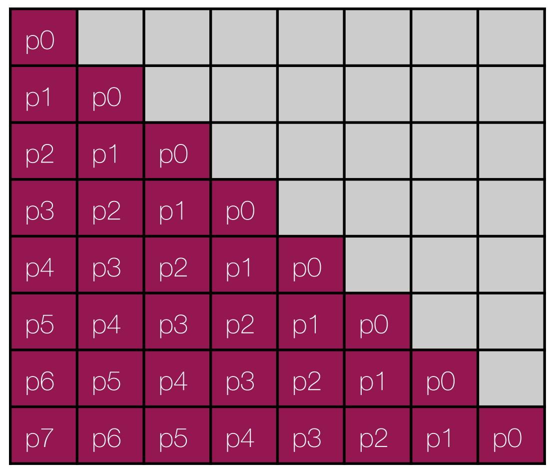
- Relative position encoding:
  add relative words differences into $\mathbf{Q} \cdot \mathbf{K}^T$

$\mathbf{Q} \cdot \mathbf{K}^T$



+

Relative Positional bias



- (+) nicely structured position information
- (-) Slow, more memory
- (-) fragments ops further, more KV cache misses

- **How might we still encode relative position, without all the overhead?**

Smart relative position encoding

- *Ideally*, if $\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_L\right]$ then position sensitive embedding between a and b vector is given by:

$$f(\mathbf{x}_a, \mathbf{x}_b, b-a) = \left(R_a \cdot \underbrace{\mathbf{W}^{(q)} \cdot \mathbf{x}_a}_{\mathbf{q}_a}\right)^T \left(R_b \cdot \underbrace{\mathbf{W}^{(k)} \cdot \mathbf{x}_b}_{\mathbf{k}_b}\right)$$

$$= \mathbf{q}_a^T \cdot R_a R_b \cdot \mathbf{k}_b$$

$$= \mathbf{q}_a^T \cdot R_{a-b} \cdot \mathbf{k}_b$$

- Sensitive to relative position
- and R_{b-a} can be decoupled into $R_a R_b$ for fast attention

$$R_a = e^{j \cdot \theta a} \qquad R_b = e^{-j \cdot \theta b} \qquad R_a R_b = e^{j \cdot \theta(a-b)} = R_{a-b}$$

These are rotations in the complex plain

Smart relative position encoding

- but practically $R_a R_b = e^{j\cdot\theta(a-b)} = R_{a-b}$ requires complex valued arithmetic and we only want real valued tensors. So we can get the same benefit via:

$$f(\mathbf{x}_a, \mathbf{x}_b, a-b) = \text{Re}[\mathbf{q}_a^T \cdot R_a R_b \cdot \mathbf{k}_b] = \mathbf{q}_a^T \cdot \text{Re}[R_a R_b] \cdot \mathbf{k}_b$$

- which in the 2D case reduces to the rotation matrix:

$$R_a \cdot \mathbf{q}_a = \begin{bmatrix} \cos(a\cdot\theta) & -\sin(a\cdot\theta) \\ \sin(a\cdot\theta) & \cos(a\cdot\theta) \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \quad R_b \cdot \mathbf{k}_b = \begin{bmatrix} \cos(b\cdot\theta) & -\sin(b\cdot\theta) \\ \sin(b\cdot\theta) & \cos(b\cdot\theta) \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

- and we effectively get relative position encoding, but with decoupled operations!!!
- but, expanding beyond 2D starts to make the operation too computational… so let's only do operation in pairs along $\mathbf{q}$ and along $\mathbf{k}$ separately (lots of 2D rotations)

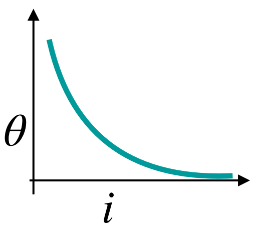
Rotary Position Encoding, RoPE

- Now we can finally understand this operation:

In general, produces better results (mostly) while being not too computational

$$R_a = \begin{bmatrix} \cos(a\cdot\theta_1) & -\sin(a\cdot\theta_1) & 0 & 0 & \dots & 0 & 0 \\ \sin(a\cdot\theta_1) & \cos(a\cdot\theta_1) & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \cos(a\cdot\theta_2) & -\sin(a\cdot\theta_2) & \dots & 0 & 0 \\ 0 & 0 & \sin(a\cdot\theta_2) & \cos(a\cdot\theta_2) & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \cos(a\cdot\theta_{L/2}) & -\sin(a\cdot\theta_{L/2}) \\ 0 & 0 & \dots & 0 & 0 & \sin(a\cdot\theta_{L/2}) & \cos(a\cdot\theta_{L/2}) \end{bmatrix}$$

Lots of pairwise rotations, each preserving the property of relative position encoding



where $\theta_i = 10000^{-2(i-1)/L}$ defines the range of increasing rotations

Fast to implement with two point wise vector multiplies and addition (low overhead, still parallel)

$$R_a \cdot \mathbf{q}_a = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ \vdots \\ q_{L-1} \\ q_L \end{bmatrix} \cdot \begin{bmatrix} \cos(a\cdot\theta_1) \\ \cos(a\cdot\theta_1) \\ \cos(a\cdot\theta_2) \\ \cos(a\cdot\theta_2) \\ \vdots \\ \cos(a\cdot\theta_{L/2}) \\ \cos(a\cdot\theta_{L/2}) \end{bmatrix} + \begin{bmatrix} -q_2 \\ q_1 \\ -q_4 \\ q_3 \\ \vdots \\ -q_L \\ q_{L-1} \end{bmatrix} \cdot \begin{bmatrix} \sin(a\cdot\theta_1) \\ \sin(a\cdot\theta_1) \\ \sin(a\cdot\theta_2) \\ \sin(a\cdot\theta_2) \\ \vdots \\ \sin(a\cdot\theta_{L/2}) \\ \sin(a\cdot\theta_{L/2}) \end{bmatrix}$$

Large Angle, sensitive to position

Transformer learns to encode positionally sensitive meaning in high frequency indices…

Small angle, less sensitive to position

Su et al., RoFormer, ACM Neurocomputing (2023) https://arxiv.org/pdf/2104.09864

Paper Presentation

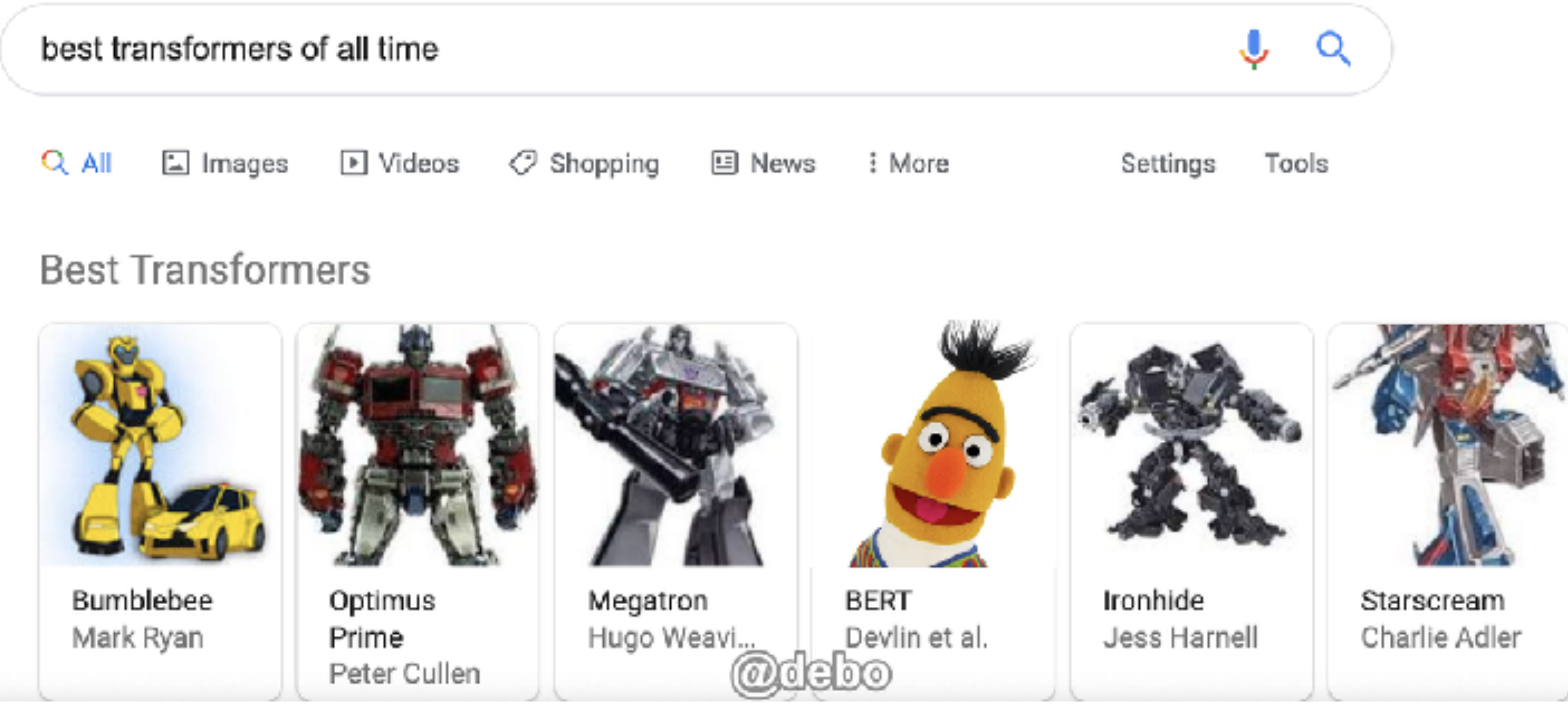
Are transformers effective for time series forecasting?

AUTHORs: Ailing Zeng, Muxi Chen, Lei Zhang, Qiang Xu | Authors Info & Claims

AAAI'23/IAAI'23/EAAI'23: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence
Article No.: 1248, Pages 11121 - 11128 • https://doi.org/10.1609/aaai.v37i9.26317

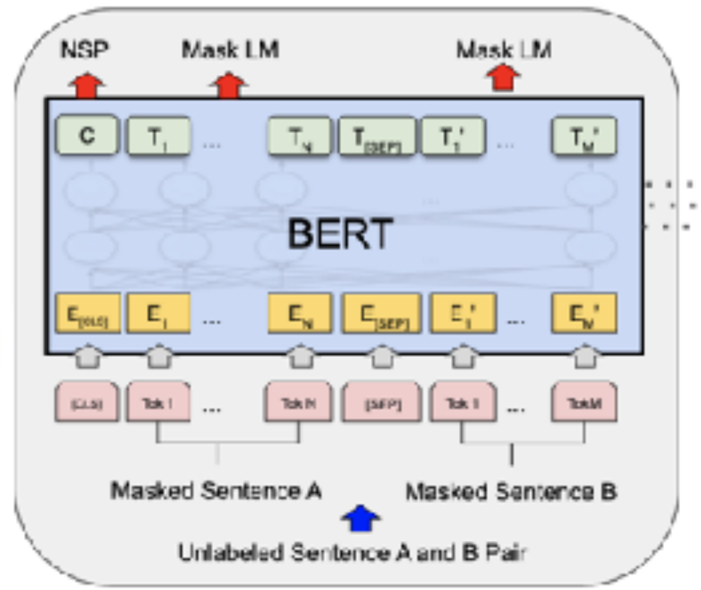
Published: 07 February 2023 Publication History

Encoder Transformers



Bidirectional Encoder Representation

- Google, 2018. Vocab: 30k words
- Bidirectional (non-causal attention)
- BERT$_{\text{Base}}$
  - 12 encoder layers, 12 heads/layer
  - 110M parameters
- BERT$_{\text{Large}}$
  - 24 encoder layers, 16 heads/layer
  - 340M parameters



Pre-training

Masked Language Modeling (Mask LM)

“I am **[MASK1]** in CS8321 at SMU. This class is **[MASK2]**”

MASK1: **“enrolled”** MASK2: **“great”**

Next Sentence Prediction (NSP)

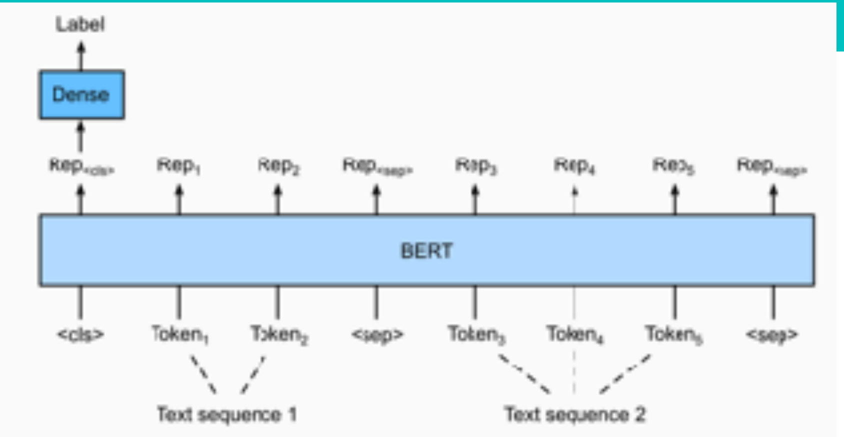
"[CLS] Dr. Larson is a professor [SEP] his class examples are great” → **Label “IsNext”**

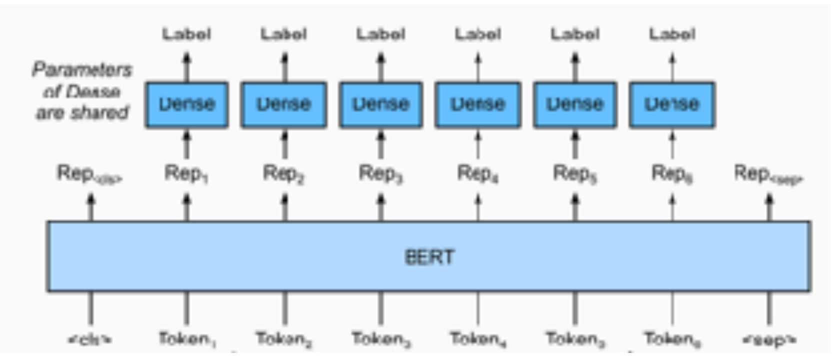
"[CLS] Dr. Larson is a professor [SEP] do you like bread” → **Label “NotNext”**

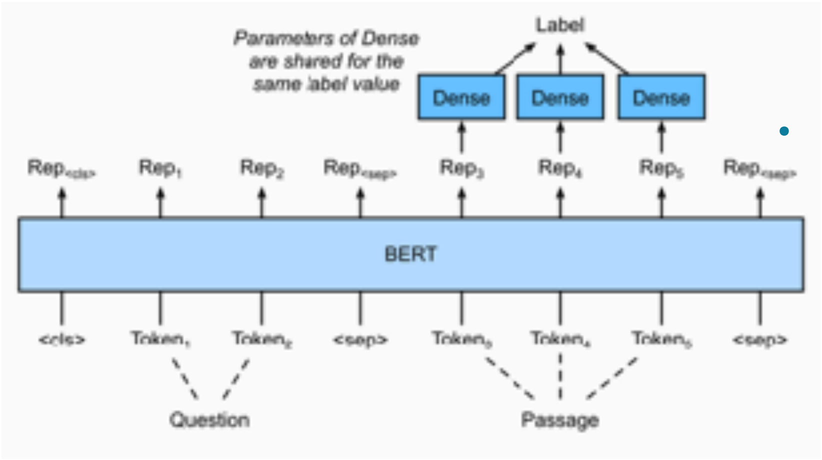
Devlin et al. ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". arXiv:1810.04805v2

Fine Tuning BERT

- Sentence predict: like Text Similarity
  - Make use of NSP
  - Two sentences, do they belong?
- Part of speech tagging
  - Make use of Masked LM
  - Shared dense layer for each Rep







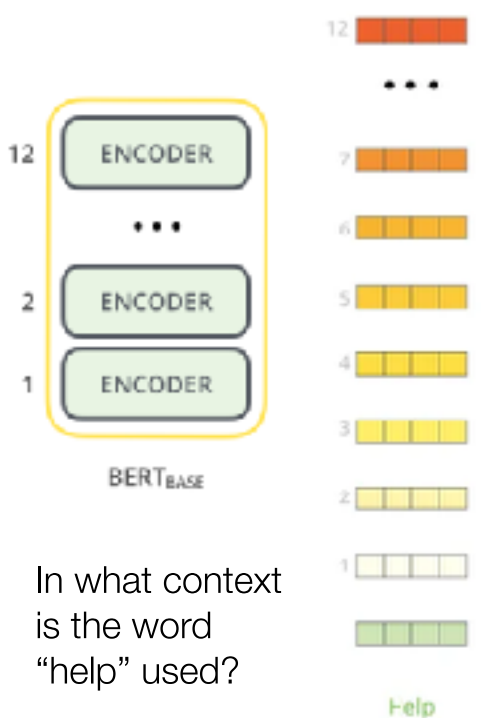
- Question Answering (Stanford QA Dataset, SQuaD)
  - Make use of Masked LM
  - Highlight passage text that answers given question

Q: Who currently teaches machine learning at SMU?
P: “Machine learning was first offered at SMU in the 1990’s. **Dr. Larson** has been teaching the course since 2014 and has changed it into a neural networks course, despite its origins. ”

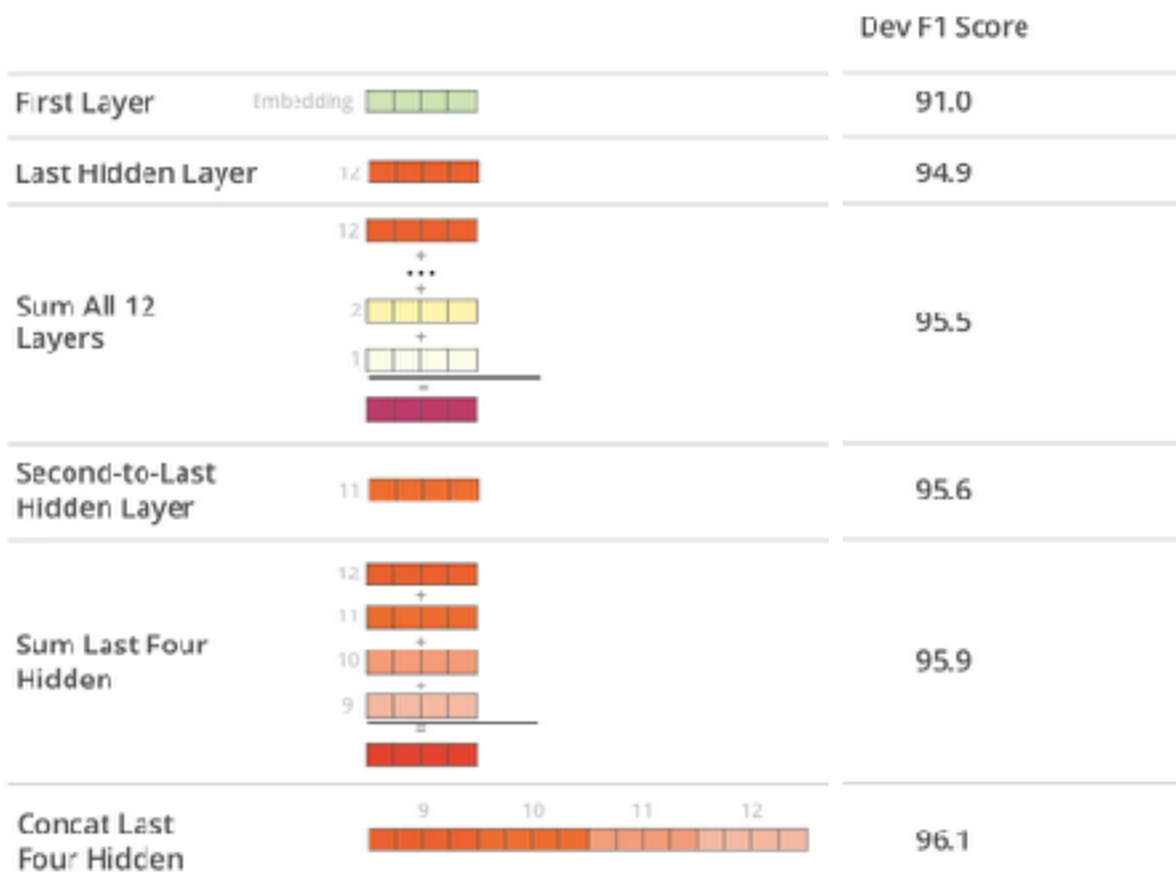
https://classic.d2l.ai/chapter_natural-language-processing-applications/finetuning-bert.html

Fine Tuning BERT

- Could we use more than just the final output layers?



Each word has 12 layer outputs in $\text{BERT}_{\text{Base}}$



In what context is the word "help" used?

Noun, verb, adverb, etc.

Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". arXiv:1810.04805v2

MELVA Results (my lab)

- Measuring English Language Vocabulary Acquisition
- Or results from my lab:
  - Using science terms in sentence?
  - Collect/transcribe responses
- Collected about 6000 sentences

- Transfer learn based upon LM output
  - Without transformer LM: ~75%
  - With transformer LM: ~84%

L@S '23, July 20–22, 2023, Copenhagen, Denmark

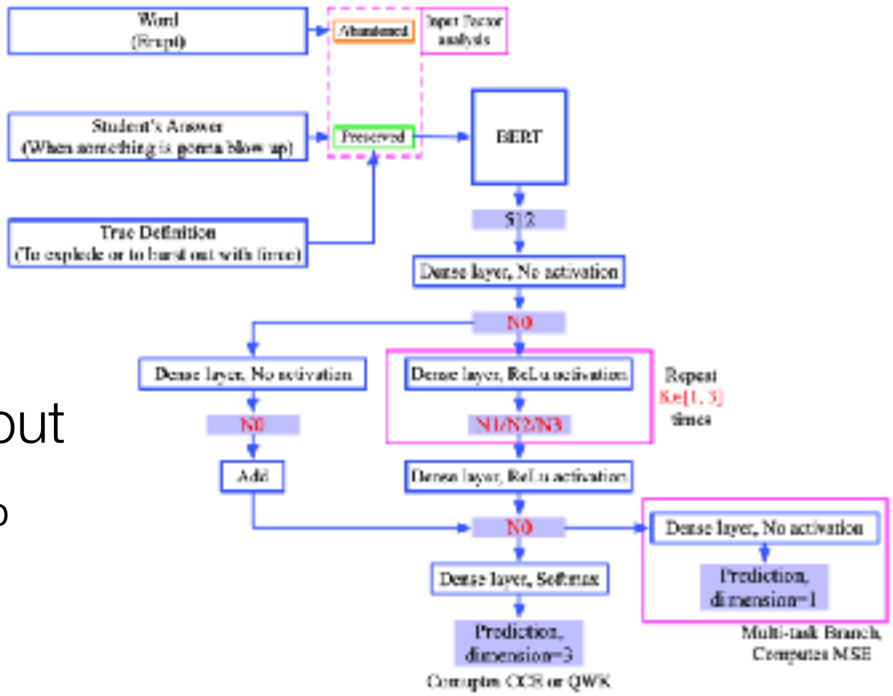


Figure 1: Example of the end-to-end pipeline of the network. Variables marked in red are found through hyperparameter search.

Zhongdi Wu, Larson, E., Makoto Sano, Doris Baker, Akihito Kamata, & Nathan Gage (2023) Towards Scalable Vocabulary Acquisition Assessment with BERT. Learning at Scale, 5. 10.1145/3573051.3596170

Fine Tuning BERT

eclarson Eric Larson

20 News Groups

Main Repository:

    02 BERT Transfer[experimnetal].ipynb

Since we are using hugging face for this, its better to use PyTorch …