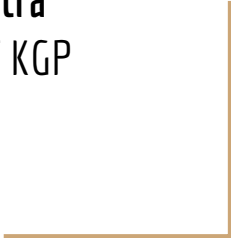




Pretrained Models in NLP

Presenter - Bishal Santra
CNeRG Reading Group, IIT KGP
July 18, 2019



Codes

Transformer-XS

Jupyter-Notebook

huggingface/pytorch-transformer

[LINK TO COLAB DEMO](#)

Tutorial on Building a Transformer LM: [Transformer-XS](#), [Jupyter-Notebook](#)

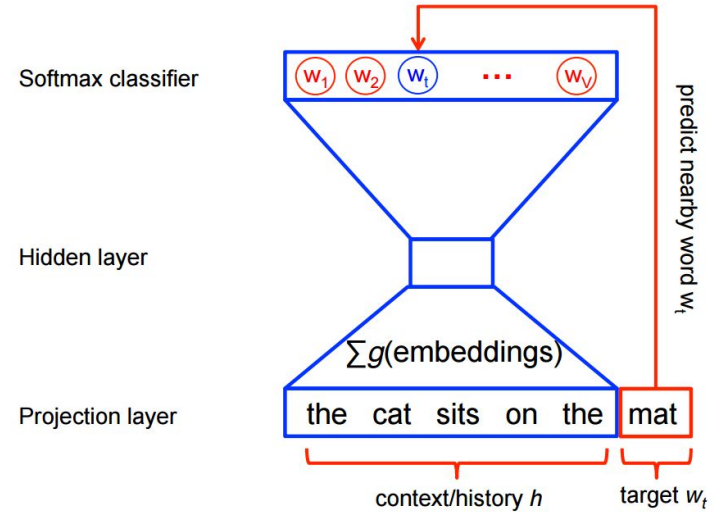
A Brief History of Pretraining

The main idea behind pretraining is to transfer knowledge from an already trained model. It comes in many flavours, e.g.

- **Word2Vec** - Pretrained word embeddings; Fixed Embedding Matrix
- **ImageNet** - Pretrained feature extraction layers; Final layers fine tunable
- **Infersent** - Fixed RNN Encoder for sentences; Fixed Model; Pretraining done on supervised task NLI
- **ELMo** - Pretrained model for Contextual Word Embedding; use as it is or fine tune linear layers on top.

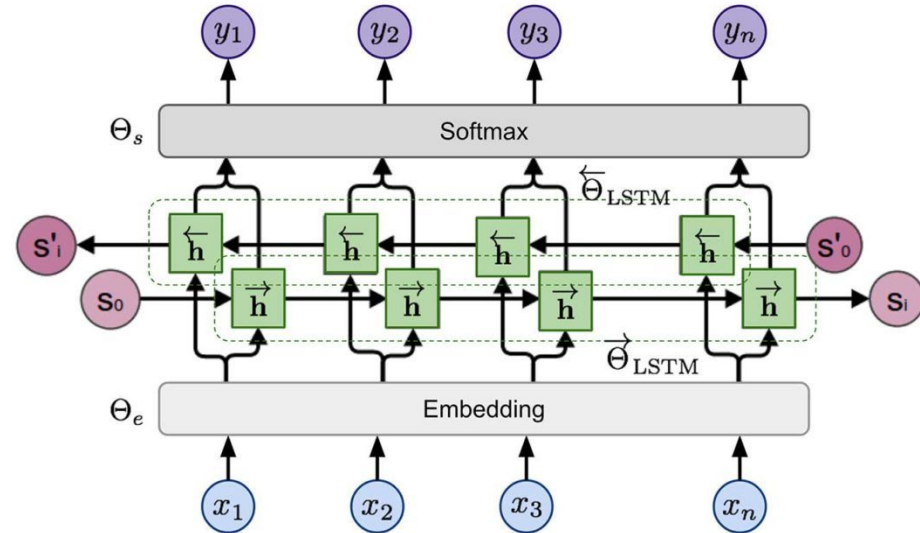
Word2Vec

- Default choice for word representation
- Learn word vectors from a large corpus
 - *Generalized representation* of words
- What's the catch?
 - Represents word meanings, a necessity for other NLU tasks
 - Word relations encoded upto certain extent as $w_i - w_j$



ELMo

- Pretrain forward and backward language models simultaneously
- *Contextual word embedding* - Hidden states from different layers
- *Sentence Embedding* - Final hidden states, s_1 and s_1'
- Add linear layers on top of above embedding and fine tune

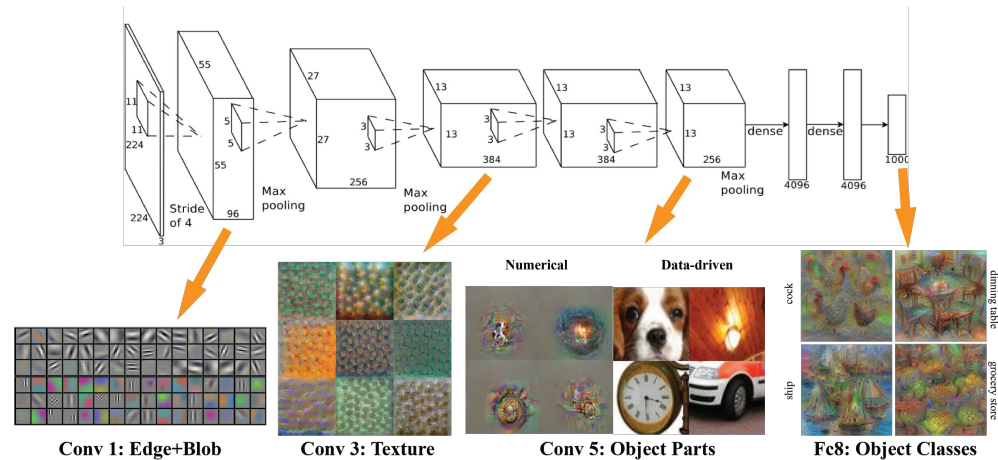


$$\sum_{k=1}^N (\log p(t_k | t_1, \dots, t_{k-1}; \Theta_x, \vec{\Theta}_{LSTM}, \Theta_s) + \log p(t_k | t_{k+1}, \dots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)).$$

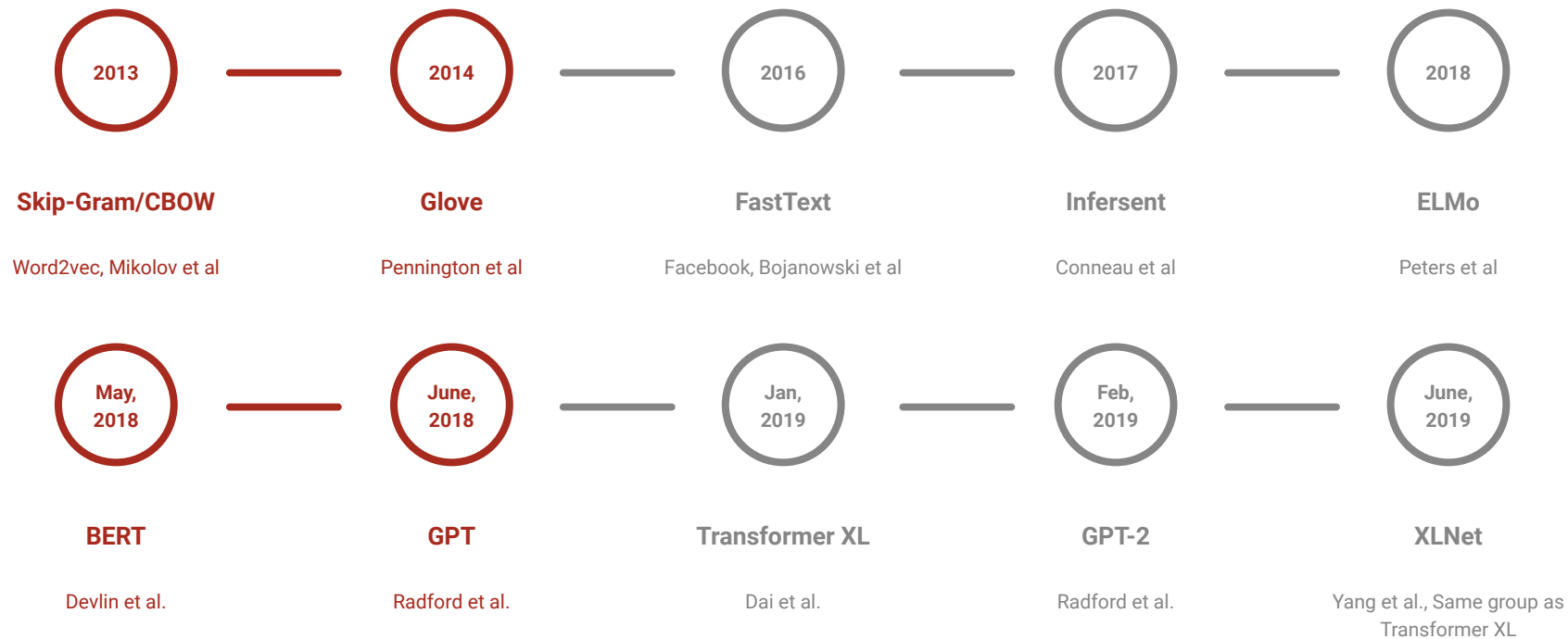
ELMo's Loss function

AlexNet for Imagenet Dataset

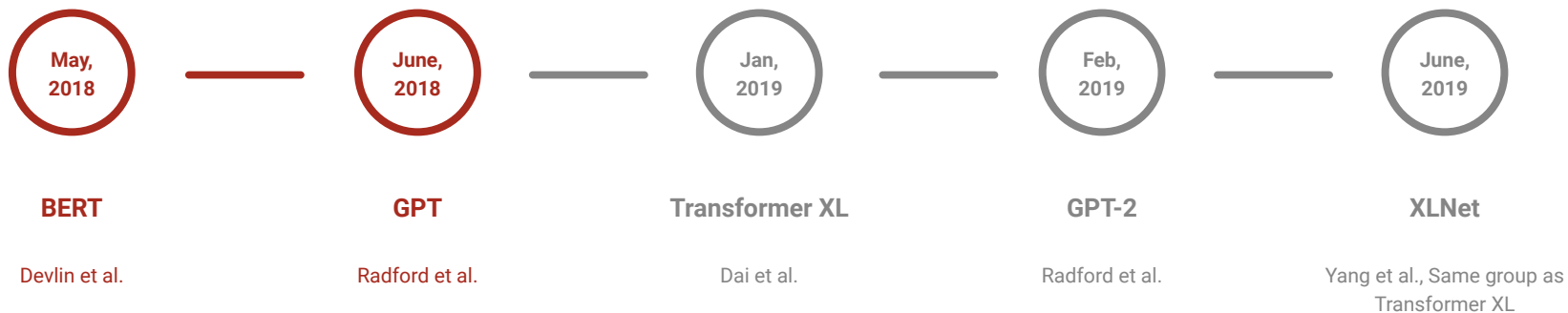
- Idea of pretraining is widely used in CV
- Pretrain a model for image classification, a task which has relatively more data
- Reuse initial layer weights in a more difficult task, e.g. object detection
 - Only fine-tune final layers.
- Lower level features doesn't change task to task.



Timeline of pre-training methods in NLP



Timeline of pre-training methods in NLP



- All of these last 5 models are based on an architecture called Transformer
- Transformer - Model architecture entirely based on self-attention

Enter Self-Attention

- Today's main focus. Self-attention based methods
- Let's review it.
- What's wrong with Seq2Seq model?
 - A critical and apparent disadvantage of this fixed-length context vector design
 - Incapability of remembering long sentences
 - Attention mechanism was born (Bahdanau et al., 2015) to resolve this problem
 - Mostly cross attention
- Self-Attention
 - Attention mechanism over a *single sequence*
 - We can just use the attention mechanism
 - Drop the recurrence component from Attention-Seq2Seq
 - Without the recurrence it's parallelizable

Attention is all you need!

- We all have seen this image.
- But doesn't really clear things up.
- Questions?
 - Intuition behind the model?
 - Why use this instead of RNNs?
 - How do I code this? Or where is this useful?

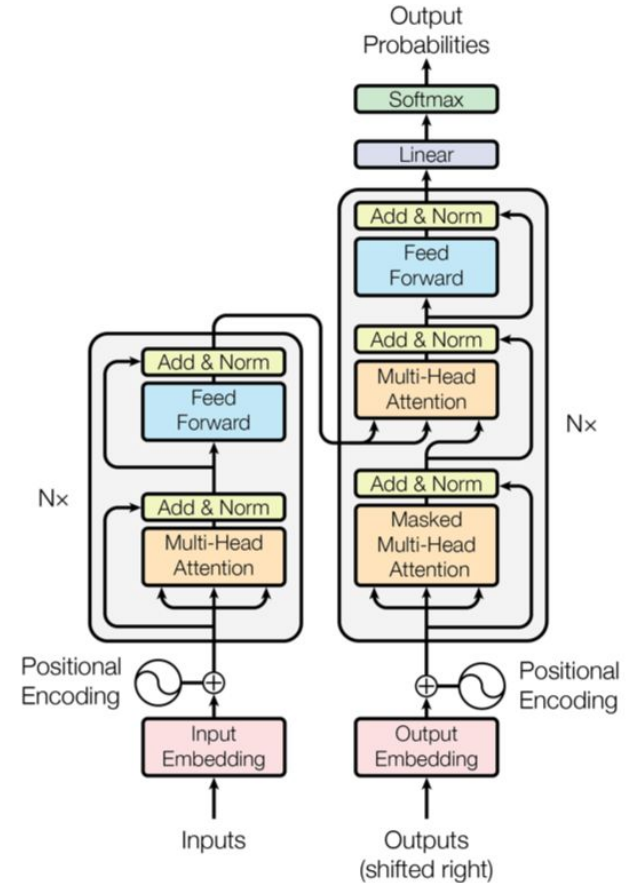
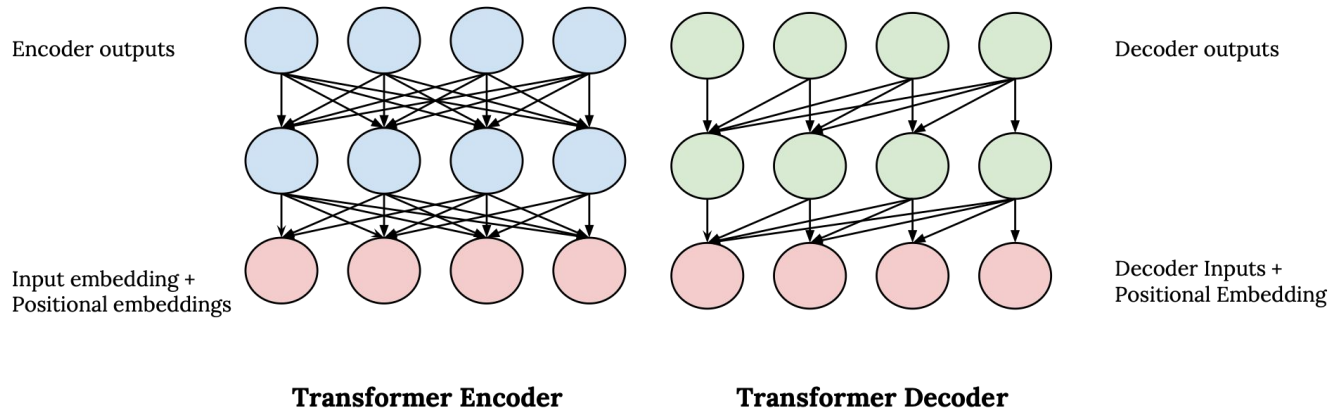


Figure 1: The Transformer - model architecture.

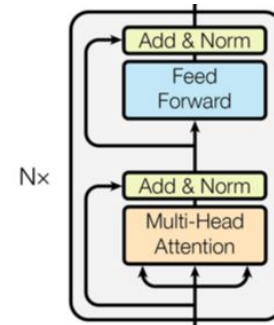
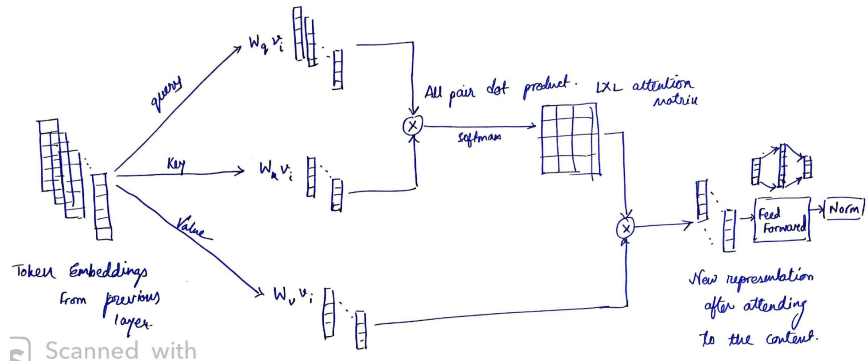
Inside the Transformer

- Multiple layers of “attention and combine”, stacked on top of each other.
- It’s like extended-convolution.

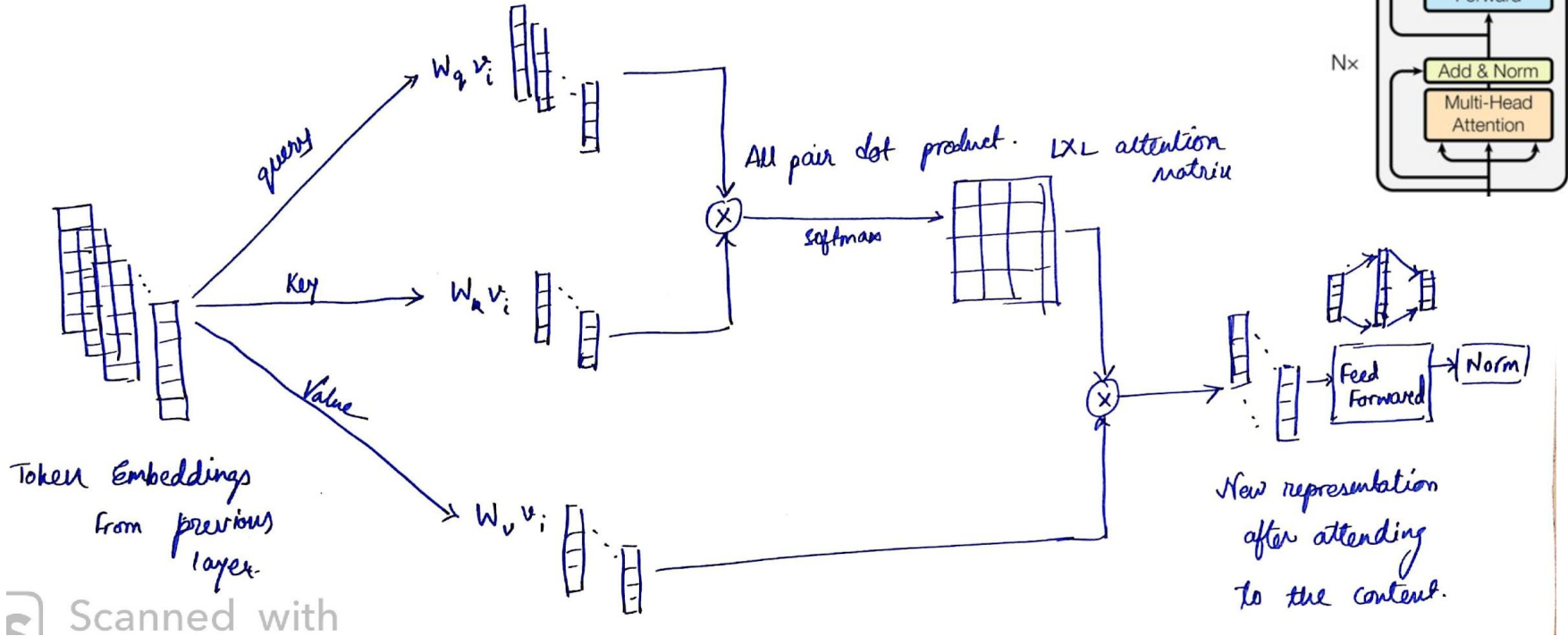


Inside the Transformer

- But that's not all, there's more to it.
- Each attention layer applies multi-head attention
 - On all tokens from previous layer
- The image below is single attention.
- Create copies of it within a single layer. We have multi-head attention.



Inside the Transformer



Inside the Transformer

- Understanding encoder and decoder is easy
- Encoder - Self attention on input sequence
- Decoder - Self attention on input sequence (*saved-memory*) and input sequence till now.

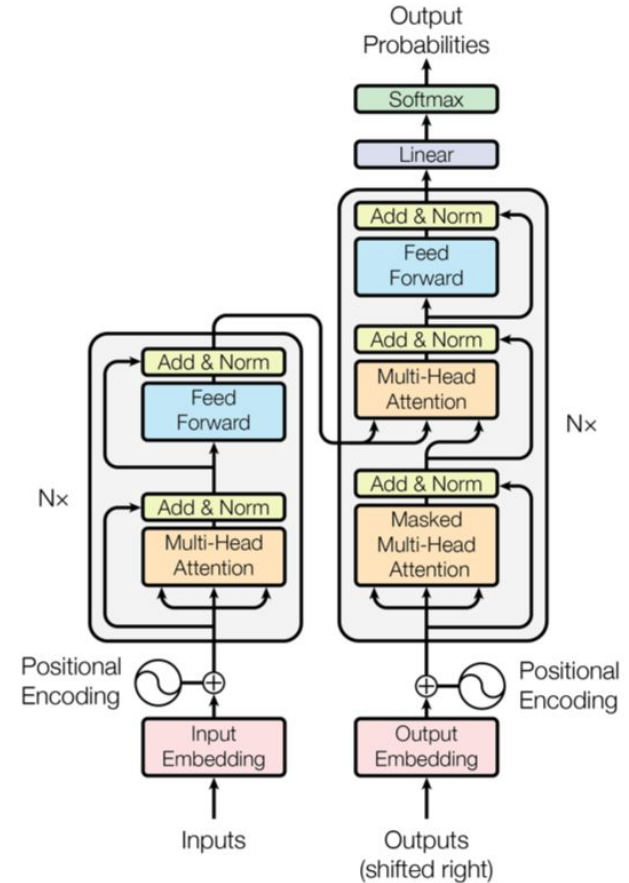
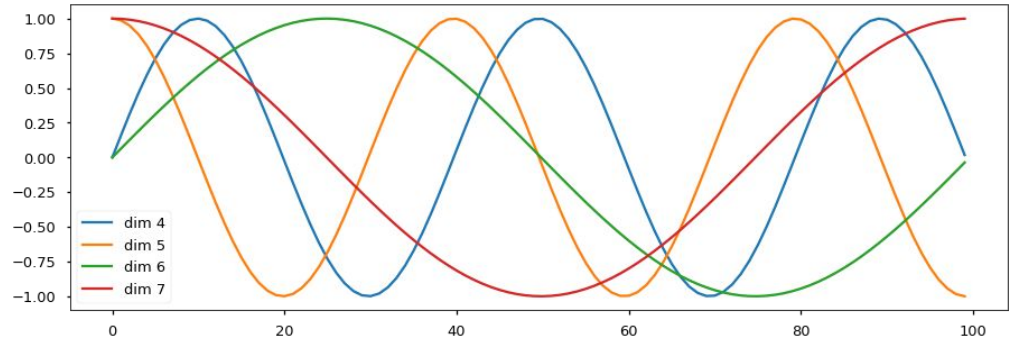


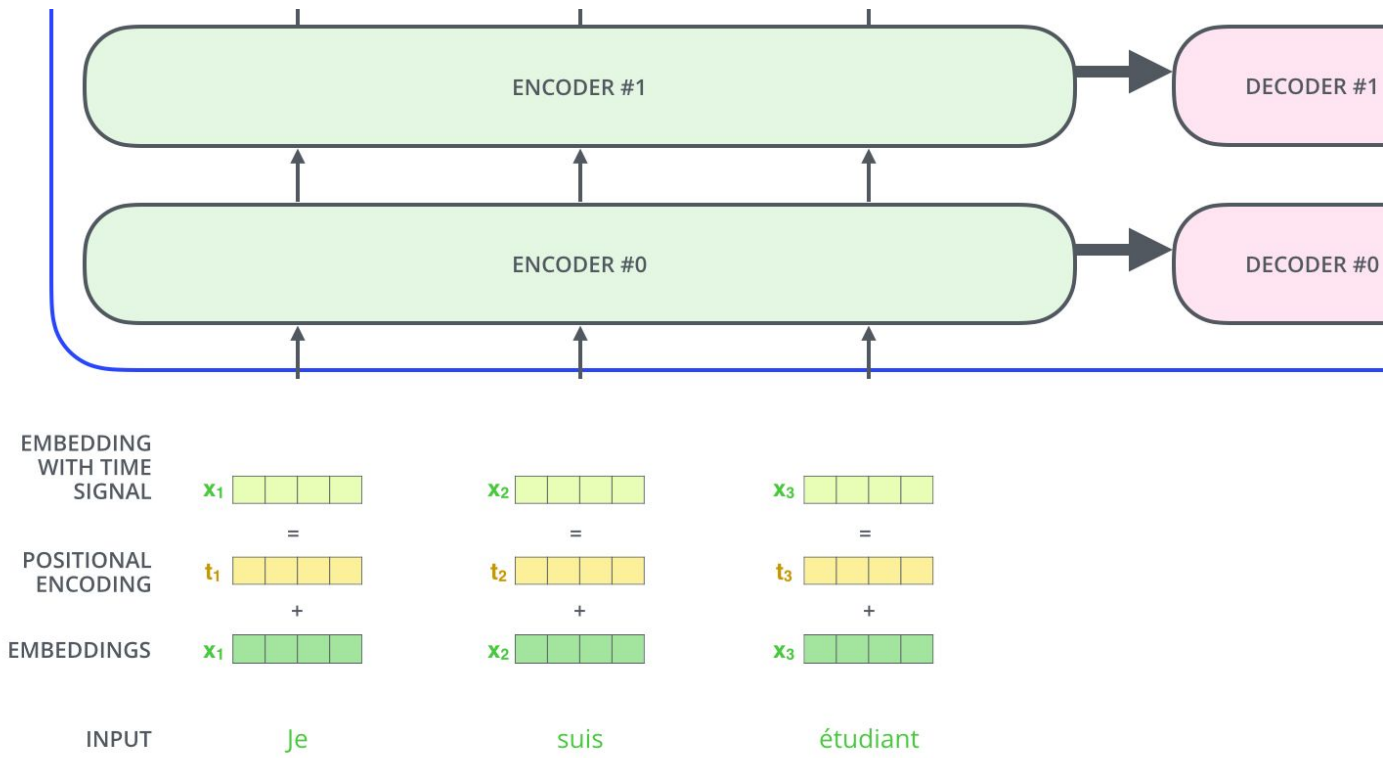
Figure 1: The Transformer - model architecture.

Positional Encoding

- But in self-attention all inputs are equivalent.
- Model will interpret the input as *Bag-of-words* instead of a *Sequence*.
- Need to tell the position of each token explicitly.
- Create a positional embedding matrix of size $L \times d$ and add it to word embedding matrix.
- Add sinusoids of different freq.



Positional Encoding



Sources: The Illustrated Transformer

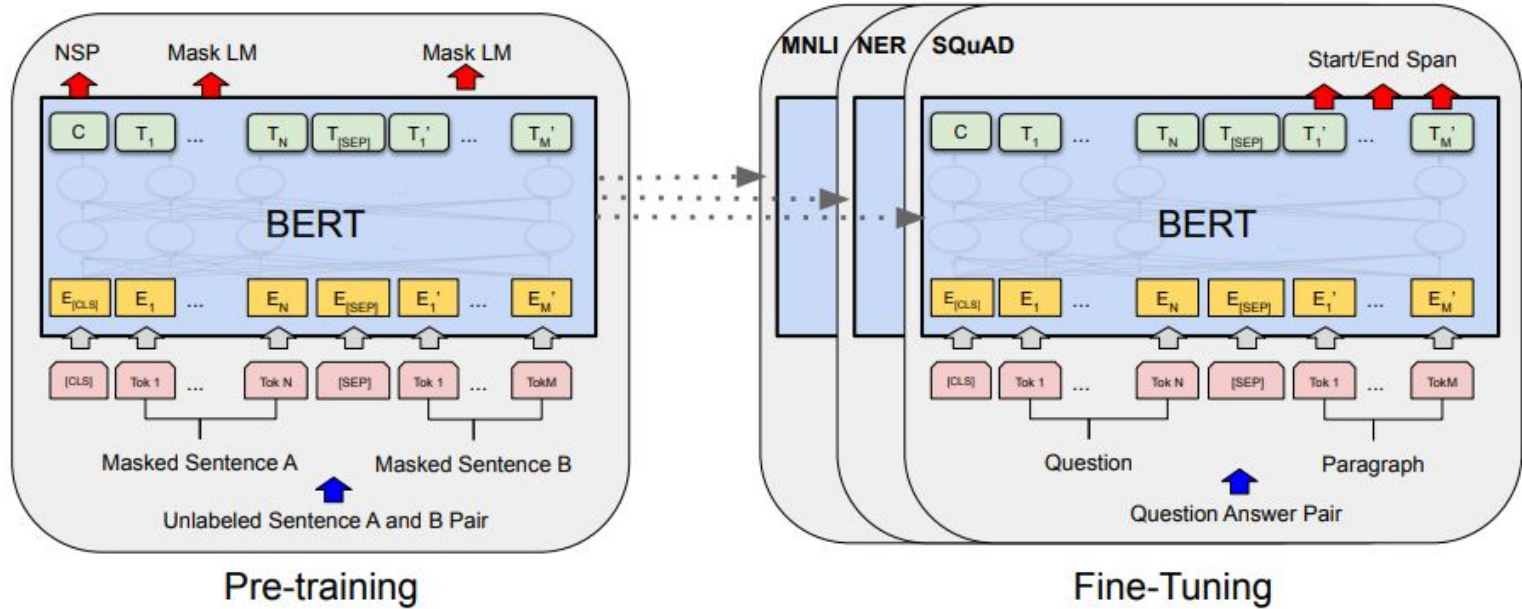
Back to Pretraining

- Types of pre-training in NLP
 - Unsupervised Learning methods
 - Autoregressive Language Model
 - ELMo, Transformer-XL
 - AutoEncoder Language Model
 - BERT, XLNet
 - Supervised Learning Methods
 - Multi-Task Learning and Transfer Learning

BERT

- Pretrained model for **Contextual-word Embeddings**
- **Pre-training Tasks**
 - Masked LM
 - Next Sentence Prediction
- Training Dataset
 - BookCorpus (800M Words)
 - Wikipedia English (2,500M Words)
- Training Settings
 - Billion Word Corpus was not used to avoid using shuffled sentences in training.
 - Training with long contiguous contexts

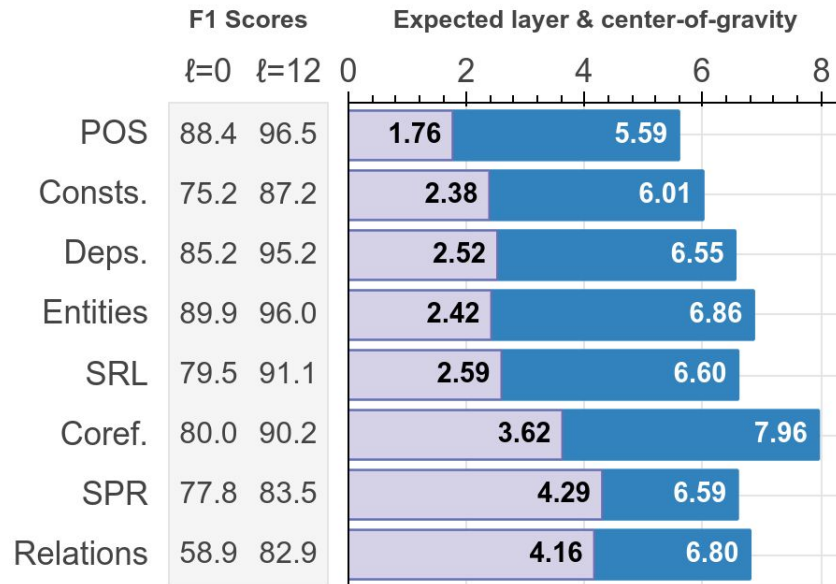
BERT



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

Looking Inside BERT

- Similar to what has been observed in Deep CNN models in CV, BERT also learns a hierarchy of features
- BERT also learns elementary features required for NLU
- This shows why we should use such architectures.



OpenAI GPT

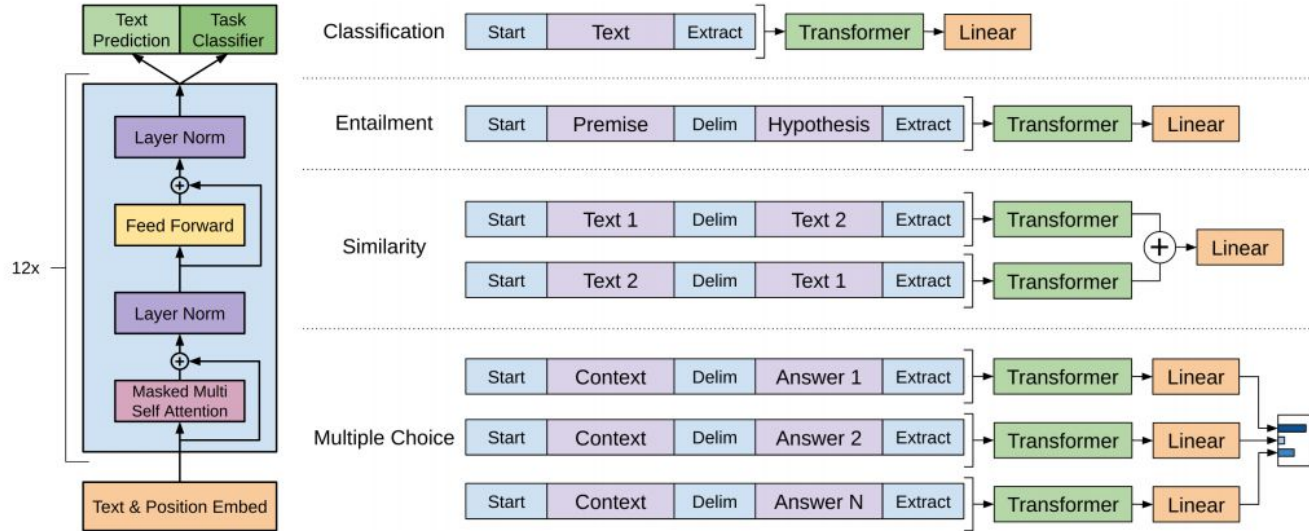


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

OpenAI GPT

Training Dataset

- BookCorpus
- Not 1B word benchmark

Model Settings

- 768 dim, 12 attention heads
- Gaussian Error Linear Unit (GELU) as Activation
- ***Learned position embedding matrix*** instead of the sinusoidal version

Comparison

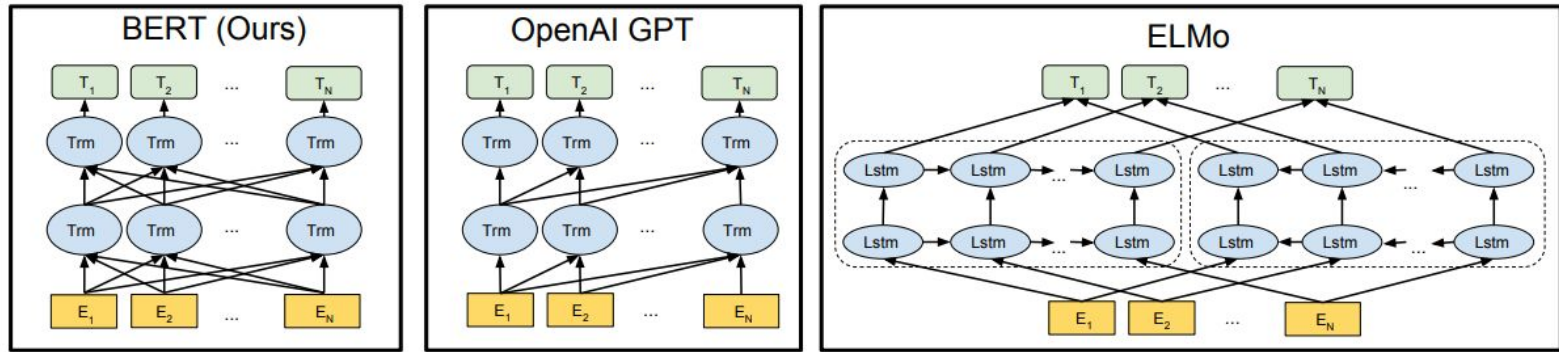


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT 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.

OpenAI GPT-2

- Extended version of OpenAI GPT
- Vocabulary size expanded to 50,257
- Trained on larger context of size 1024 instead of 512
- **Dataset**
 - This is the most important change
 - Trained on **40 GB** of web crawl data
 - **WebText: 8 Million documents**
 - Wikipedia not included to prevent overlapping data
 - Filtering Good and Bad webpages
 - Outbound links from Reddit posts or comments with more than 3 karma (votes)
 - Heuristic indicator of what other users found interesting, educational or funny

Transformer XL (eXtra Long)

- Problem of Transformer - Fixed Length Context
- Can't carry over information beyond the length
- Context Fragmentation
 - Training batches from corpus doesn't respect semantic boundary
 - It's not easy to detect such context boundaries either
- Solution
 - Add Recurrence
 - Relative Positional Encoding Scheme

Transformer XL

Vanilla Transformer LM looks like

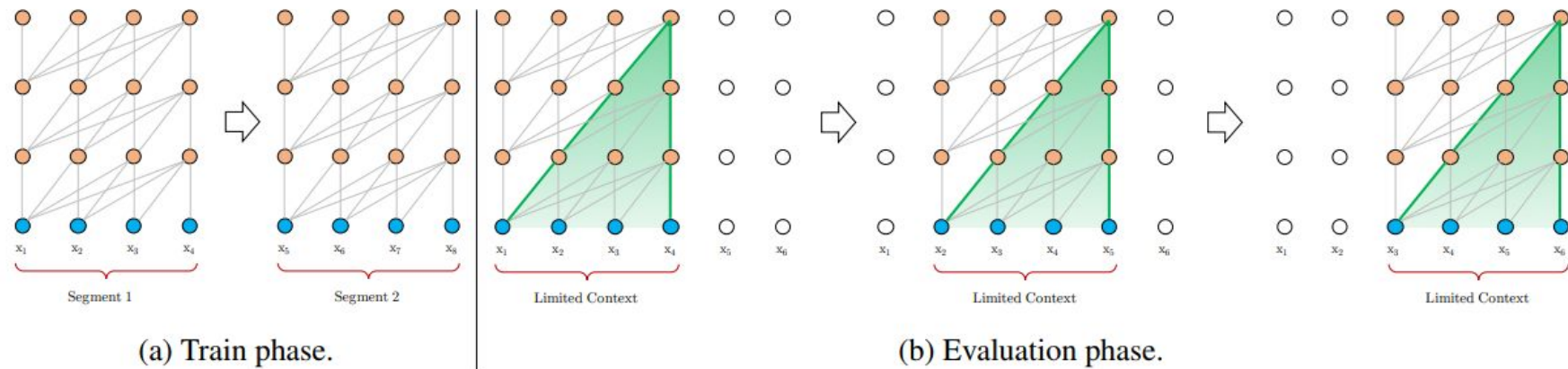


Figure 1: Illustration of the vanilla model with a segment length 4.

Sources:

Transformer XL

Add recurrence and we have Transformer-XL.

- Segment Level Recurrence

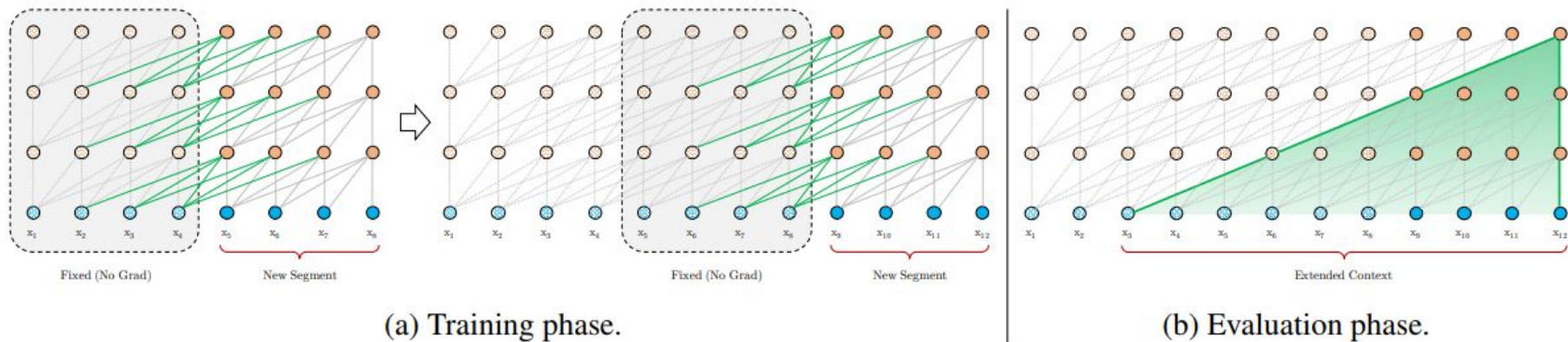
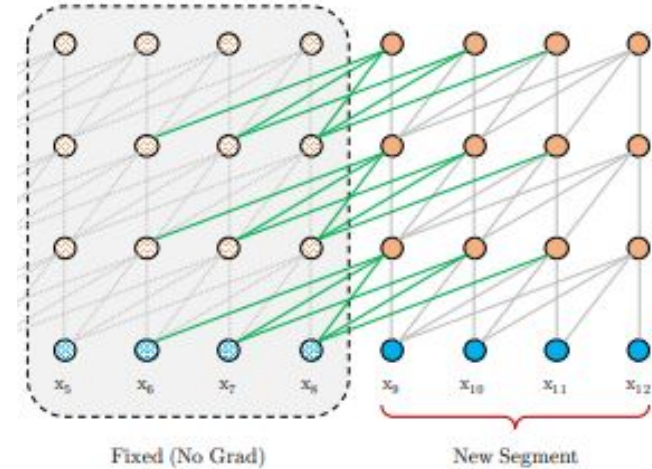


Figure 2: Illustration of the Transformer-XL model with a segment length 4.

Transformer XL

- No gradient is propagated to the previous segment
- 2L keys and values. (L = Segment Length)
- Only L queries.
- Only calculate states for new segment
- Crazy Fast Evaluation
 - 1800x times faster than vanilla Transformer
 - Reuse previous states instead of recomputing at every step



$$\tilde{\mathbf{h}}_{\tau+1}^{n-1} = [\text{SG}(\mathbf{h}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}],$$

$$\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n = \mathbf{h}_{\tau+1}^{n-1} \mathbf{W}_q^\top, \tilde{\mathbf{h}}_{\tau+1}^{n-1} \mathbf{W}_k^\top, \tilde{\mathbf{h}}_{\tau+1}^{n-1} \mathbf{W}_v^\top,$$

$$\mathbf{h}_{\tau+1}^n = \text{Transformer-Layer}(\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n).$$

Transformer XL

Relative Positional Encodings

- But reusing states in a transformer has a challenge.
- Positional encoding aren't translation invariant.

$$\mathbf{A}_{i,j}^{\text{abs}} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} \\ + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}.$$

$$\mathbf{A}_{i,j}^{\text{rel}} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} \\ + \underbrace{\mathbf{u}^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{v}^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}.$$

Transformer XL

- Trained and tested on
 - Wikitext-103,
 - enwiki8,
 - text8,
 - One Billion Word corpus
- Can reproduce really long texts. Best *(Relative) Effective Context Length of 900*
- SoTA on LM

Model	#Param	PPL
Grave et al. (2016b) - LSTM	-	48.7
Bai et al. (2018) - TCN	-	45.2
Dauphin et al. (2016) - GCNN-8	-	44.9
Grave et al. (2016b) - LSTM + Neural cache	-	40.8
Dauphin et al. (2016) - GCNN-14	-	37.2
Merity et al. (2018) - QRNN	151M	33.0
Rae et al. (2018) - Hebbian + Cache	-	29.9
Ours - Transformer-XL Standard	151M	24.0
<hr/>		
Baevski and Auli (2018) - Adaptive Input [◇]	247M	20.5
Ours - Transformer-XL Large	257M	18.3

Table 1: Comparison with state-of-the-art results on WikiText-103. [◇] indicates contemporary work.

Model	$r = 0.1$	$r = 0.5$	$r = 1.0$
Transformer-XL 151M	900	800	700
QRNN	500	400	300
LSTM	400	300	200
<hr/>			
Transformer-XL 128M	700	600	500
- use Shaw et al. (2018) encoding	400	400	300
- remove recurrence	300	300	300
Transformer	128	128	128

Table 8: Relative effective context length (RECL) comparison. See text for the definition of RECL and r . The first three models and the last four models are compared as two *model groups* when we calculate RECL (RECL is computed on a model group rather than a single model). Each group has the same parameter budget.

Transformer XL (Example Generation)

=== Kansas City Royals ===

==== 2012 season ====

During spring training, Kershaw played very well. He was selected to spring training as a relief pitcher for the Royals for the 2012 season. After an injury to closer Javier Vázquez, he was activated on April 29 to replace Matt Holliday in the Royals' starting rotation. In his only start with the Royals, on August 6, 2012, Kershaw struck out five batters in seven innings pitched to help the Royals to their first victory in franchise history. On September 27, 2012, it appeared Kershaw was going to pitch a complete game shutout against the Detroit Tigers, but did not manage to do so since the Tigers won 3 – 1. At the conclusion of the season, Kershaw was named Major League Baseball's Most Valuable Player, was chosen to the All-Star Game at Busch Stadium and was named to the All-Star Game as the starting pitcher at shortstop. The Royals announced on February 4, 2013 that Kershaw would spend 2013 as starting pitcher, though he was expected to miss the season as a result of a shoulder injury.

==== 2013 season ====

On May 17, 2013, Kershaw sustained another back injury and did not start in August and October 2013. He appeared in 22 starts, all starts, finishing with a strikeout-to-walk ratio of 1.50 and a 2.91 ERA. He also had the third most strikeouts in the league: 10. On May 20, 2013, he

Two Paradigms of Unsupervised LM Pretraining

BERT - A **Denoising AutoEncoder** approach to pretraining. Add noise to i/p sentence by [MASK]ing ~15% of tokens.

Issues:

1. Independence Assumption
 - Assumes all masked tokens are independent -> Reconstructed independently
 - Prohibits timely learning
2. Input Noise -> pretrain-finetune discrepancy
 - [MASK] is a token that always occur in training text but never during inference.
3. Context Dependency

Two Paradigms of Unsupervised LM Pretraining

ELMo - An **AutoRegressive** approach to pretraining.

$$p(t_1, t_2, \dots, t_N) = \prod_{k=1}^N p(t_k | t_{k+1}, t_{k+2}, \dots, t_N).$$

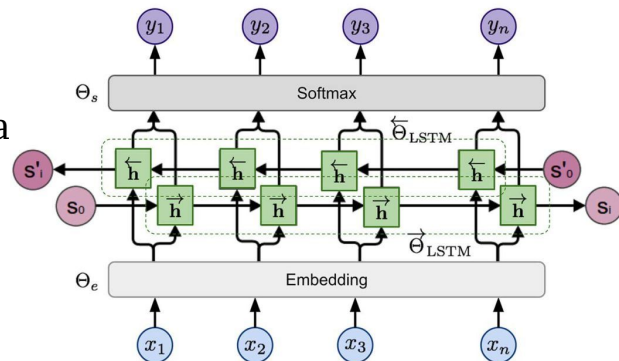
$$p(t_1, t_2, \dots, t_N) = \prod_{k=1}^N p(t_k | t_1, t_2, \dots, t_{k-1}).$$

Issues:

1. Context Dependency:

- Either of the forward or backward LMs is able to look at a single direction
- Access to bidirectional features isn't possible

Sources: XLNet, topbots,



XLNet solves it all!

- Introduces Permutation Language Modelling
- No [MASK]ing
- Bi-directional feature access

\mathbf{z} : some permutation of $\{1,2,3,\dots,L\}$

$$\max_{\theta} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_T} \left[\sum_{t=1}^T \log p_{\theta}(x_{z_t} \mid \mathbf{x}_{\mathbf{z} < t}) \right].$$

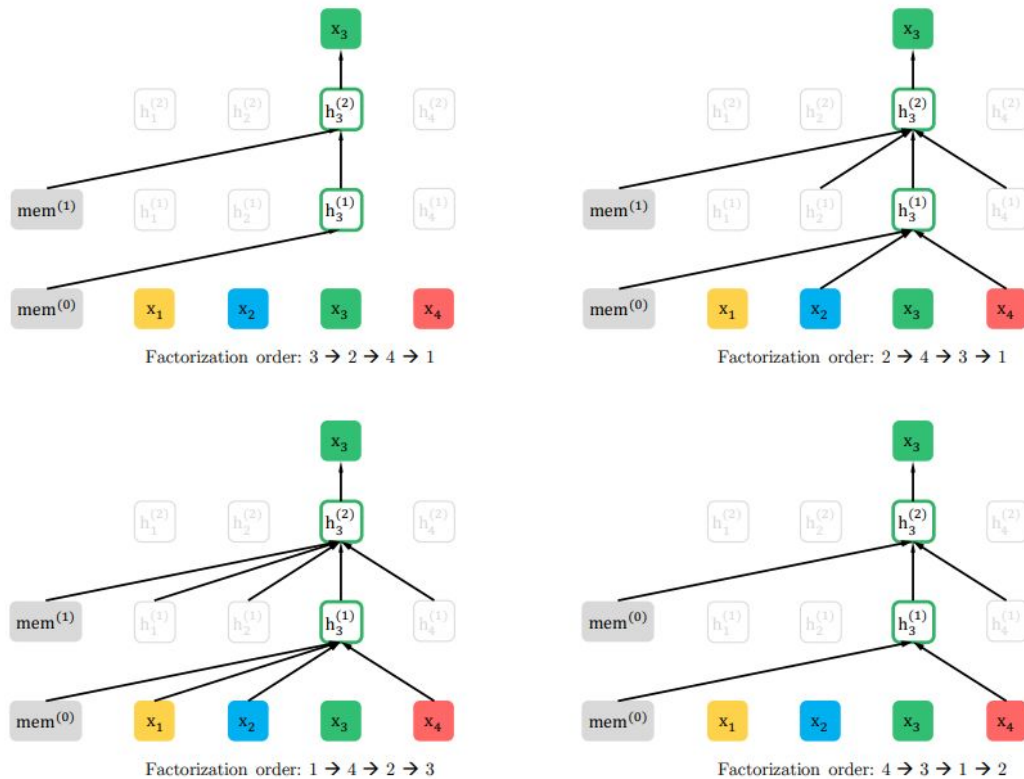


Figure 1: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence \mathbf{x} but with different factorization orders.

XLNet

- Note that the sequence order doesn't change.
- Only the factorization order is change.
- This is possible because seq order is fed trough positional encoding

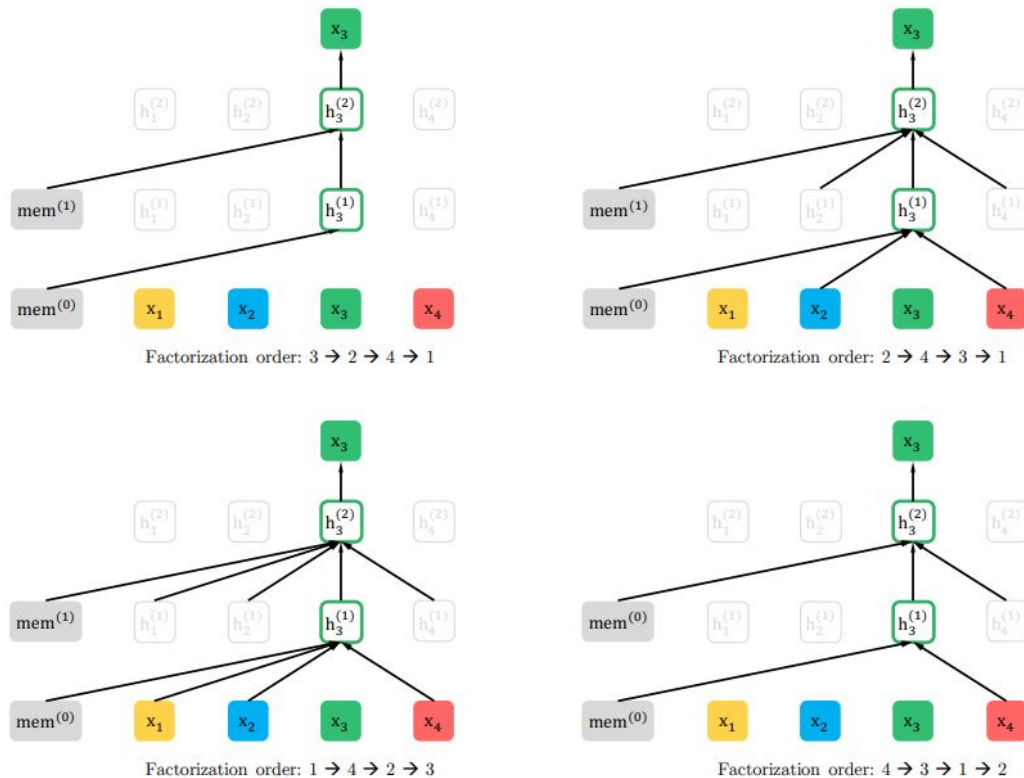


Figure 1: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence x but with different factorization orders.

XLNet

- But it creates a new issue: How to tell the model which token of the sequence is being decoded?
- **Two-Stream Self-Attention for Target-Aware Representations**
- **Content Representation: h**
- **Query Representation: g**
- Two separate embedding channels
- Next word prediction will be done using g which doesn't only have positional encoding and no token information
- First Layer
 - $h^{(0)}$ is initialized with vectors from token embedding matrix
 - $g^{(0)}$ is initialized with w , a trainable matrix

XLNet

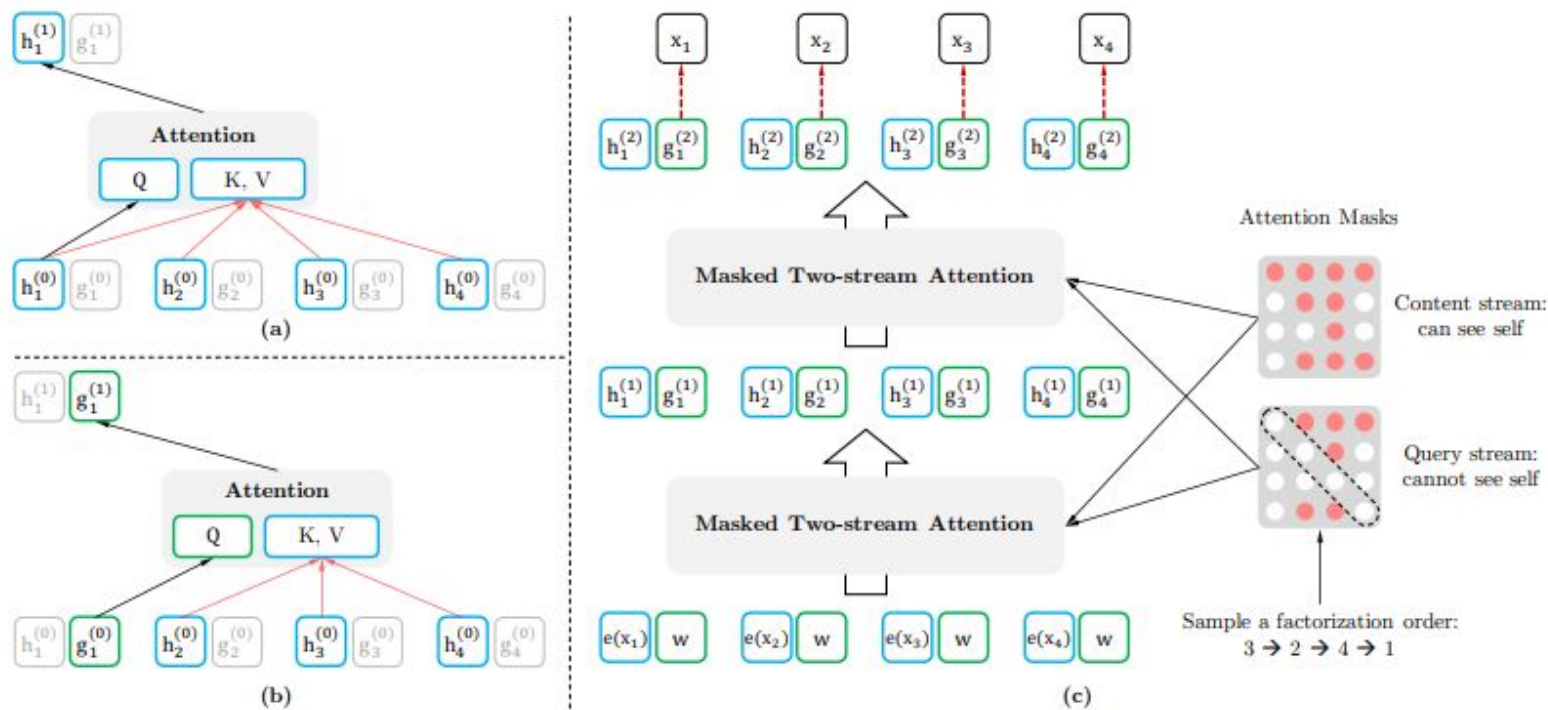


Figure 2: (a): Content stream attention, which is the same as the standard self-attention. (b): Query stream attention, which does not have access information about the content x_{z_t} . (c): Overview of the permutation language modeling training with two-stream attention.

Using these Models

- The best source of all pretrained transformer models
 - PyTorch huggingface/pytorch-transformers
 - All the pretrained models are available through this package.
- Example Case:
 - BERT for NLI, QA, Reading Comprehension etc.
 - OpenAI-GPT for predicting follow-up text
 - Transformer XL for generating even longer text
 - XLNet to beat SoTA of BERT

Using these Models

Scenario 1 - Classification

- For classification or sequence labelling task.
- Use the [CLS] tag for sentence embedding and then a linear layer on top of it.
- Use a [SEP] tag to denote boundary between multiple sentence inputs.
- NLI
 - [CLS] S1 [SEP] S2
 - Final [CLS] embedding -> Linear -> Softmax
 - Fine tune using Cross-Entropy loss

Using these Models

Scenario 2 - Sequence Tagging

- POS Tagging, Entity Detection, Semantic Role Labelling etc.
- Apply a linear classifier on the contextual embedding of all the word in sentence.
- Fine tune on task specific dataset

Using these Models

Scenario 3 - Abstractive Summarization

Summarization - Fine tune top layers of the network with a small dataset in a application specific format

Training	Inference
[Snippet] ... [TL;DR] ... [End of Summary]	[Snippet] ... [TL;DR] - Then greedy sample until [End of Summary is produced] or MAX_LEN is reached

Sources: Language Models are Unsupervised Multitask Learners (GPT-2)

Using these Models

Scenario 4 - Translation (from a Language Model)

Training	Inference
<code><english sentence> = <french sentence></code>	<code><english sentence> =</code>

Using these Models

Scenario 5 - Question Answering

No Extra Training Required! “WebText” has enough data to learn from. Fine tune with seed samples to help generate short answers.

Language Models are Unsupervised Multitask Learners			
Question	Generated Answer	Correct	Probability
Who wrote the book the origin of species?	Charles Darwin	✓	83.4%
Who is the founder of the ubuntu project?	Mark Shuttleworth	✓	82.0%
Who is the quarterback for the green bay packers?	Aaron Rodgers	✓	81.1%
Panda is a national animal of which country?	China	✓	76.8%
Who came up with the theory of relativity?	Albert Einstein	✓	76.4%
When was the first star wars film released?	1977	✓	71.4%
What is the most common blood type in sweden?	A	✗	70.6%
Who is regarded as the founder of psychoanalysis?	Sigmund Freud	✓	69.3%
Who took the first steps on the moon in 1969?	Neil Armstrong	✓	66.8%
Who is the largest supermarket chain in the uk?	Tesco	✓	65.3%
What us state forms the western boundary of montana?	Montana	✗	52.3%
Who plays ser davos in game of thrones?	Peter Dinklage	✗	52.1%
Who appoints the chair of the federal reserve system?	Janet Yellen	✗	51.5%

Sources: Language Models are Unsupervised Multitask Learners (GPT-2)

Using these Models

Scenario 5 - Question Answering

- In this particular tasks authors made sure same questions haven't appeared in WebText in the exact form
- Train-test overlap was verified using Bloom Filters built with 8-grams

Using these Models

Scenario 6 - Dialogue Response Generation

Training	Inference
[U1] ... [U2] ... [U1] ... [U2] ...	[U1] ... [U2] ... [U1] ... [U2] ... <Prompt> [U1] <i>Then do greedy sampling.</i>

What else can we do with these?

It's up to your imaginations!

Talktotransformer.com

Completion

Machine Learning has revolutionized the way we play video games by providing an intuitive way to play a game, allowing you to quickly recognize and identify enemies, collect weapons, and kill enemies in a real time. And now we've added an all new gameplay method that includes the ability to move around while killing enemies and quickly get to the final battle.

It's easy to jump straight to "play now" without having to dig into the game details. You can watch a video demo and try it for yourself to see how the new weapon works, as well as see the final battle from the perspective of the player who wins.

New Mechanics:

New weapons have been added to the game, allowing players to get the most out of all three modes.

What else can we do these?

Deep TabNine

Deep TabNine is based on **GPT-2**, which uses the Transformer network architecture.

```
// Determines whether the connection is alive
func (c *Conn) IsAlive() bool {
func (c *Conn) IsAlive() bool
func (
func IsAlive
func (c *Conn)
```

```
func (c *Conn)
func IsAlive
```

Sources: <https://tabnine.com/blog/deep>,

```
1 import os
2 import sys
3
4 # Count lines of code in the given directory, separated by file extension
5 def main(directory):
6     line_count = {}
```

```
16
17
18
19
20
21
```


Links to Papers

1. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). **Bert: Pre-training of deep bidirectional transformers for language understanding**. arXiv preprint arXiv:1810.04805.
2. Tenney, I., Das, D., & Pavlick, E. (2019). **Bert rediscovers the classical nlp pipeline**. arXiv preprint arXiv:1905.05950.
3. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). **Improving language understanding by generative pre-training**. [URL](#).
4. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). **Language models are unsupervised multitask learners**. OpenAI Blog, 1(8).
5. Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). **Transformer-xl: Attentive language models beyond a fixed-length context**. arXiv preprint arXiv:1901.02860.
6. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). **XLNet: Generalized Autoregressive Pretraining for Language Understanding**. arXiv preprint arXiv:1906.08237.

Thank you!

Any Questions?
