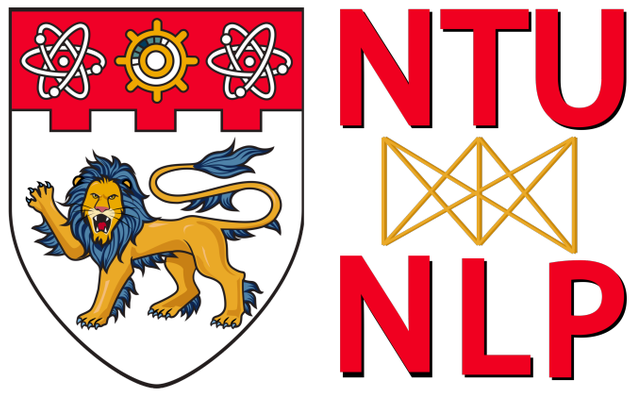
# Deep Learning for Natural Language Processing

Shafiq Joty




**Lecture 10: Self-supervised Learning**

# Announcements

- Assignment 3 has been uploaded
- Project final report guidelines are also uploaded

**https://ntunlpsg.github.io/ce7455_deep-nlp-20/**

# Where we are

NTU NLP

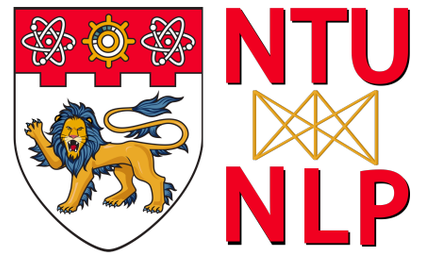
## Models/Algorithms

- Linear models
- Feed-forward Neural Nets (FNN)
- Window-based methods
- Convolutional Nets
- Recurrent Neural Nets

## NLP tasks/applications

- Word meaning
- Language modelling
- Sequence tagging
- Sequence encoding
- Parsing

# Where we are




## Models/Algorithms

- Seq2Seq
  - + Attention
  - + Subword
- Seq2Seq Varints
- Transformer Seq2Seq

## NLP tasks/applications

- Machine Translation
- Summarization
- Parsing
- Dialogue generation

# Today's Plan

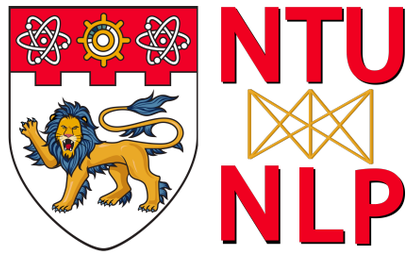
NTU NLP

- **Self-supervised pretraining**

A. Cove
B. ELMO
C. GPT
D. BERT
E. XLNET
F. ROBERTa
G. ALBERT
H. BART
I. XLM

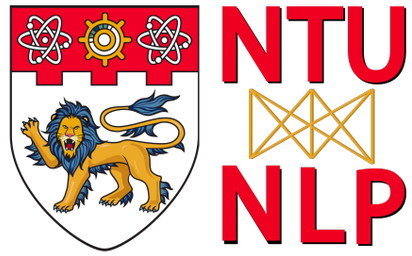
NLP tasks/applications

1. NLI (+XNLI)
2. GLUE (+Superglue)
3. QA (SQuAD)
4. Machine Translation
5. Summarization




# Contextual Word Representations

# Contextual Word Representations



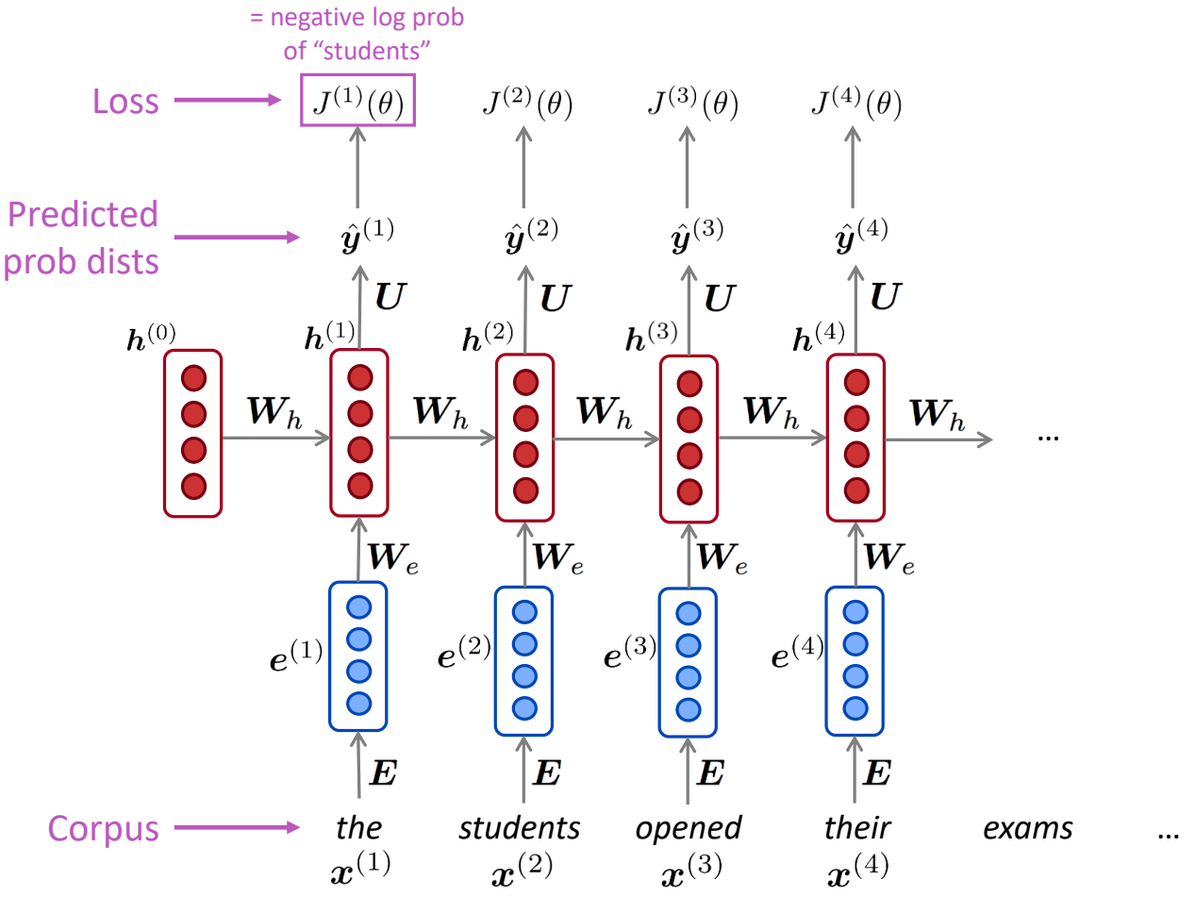

- So far, we have one (fixed) representation of words:
  - Word2vec, GloVe, fastText

- These have two problems:
  - Always the same representation for a **word type** regardless of the context in which a **word token** occurs
    - We might want very fine-grained word sense disambiguation

- We just have one representation for a word, but words have different **aspects**, including semantics, syntactic behavior, and register/connotations

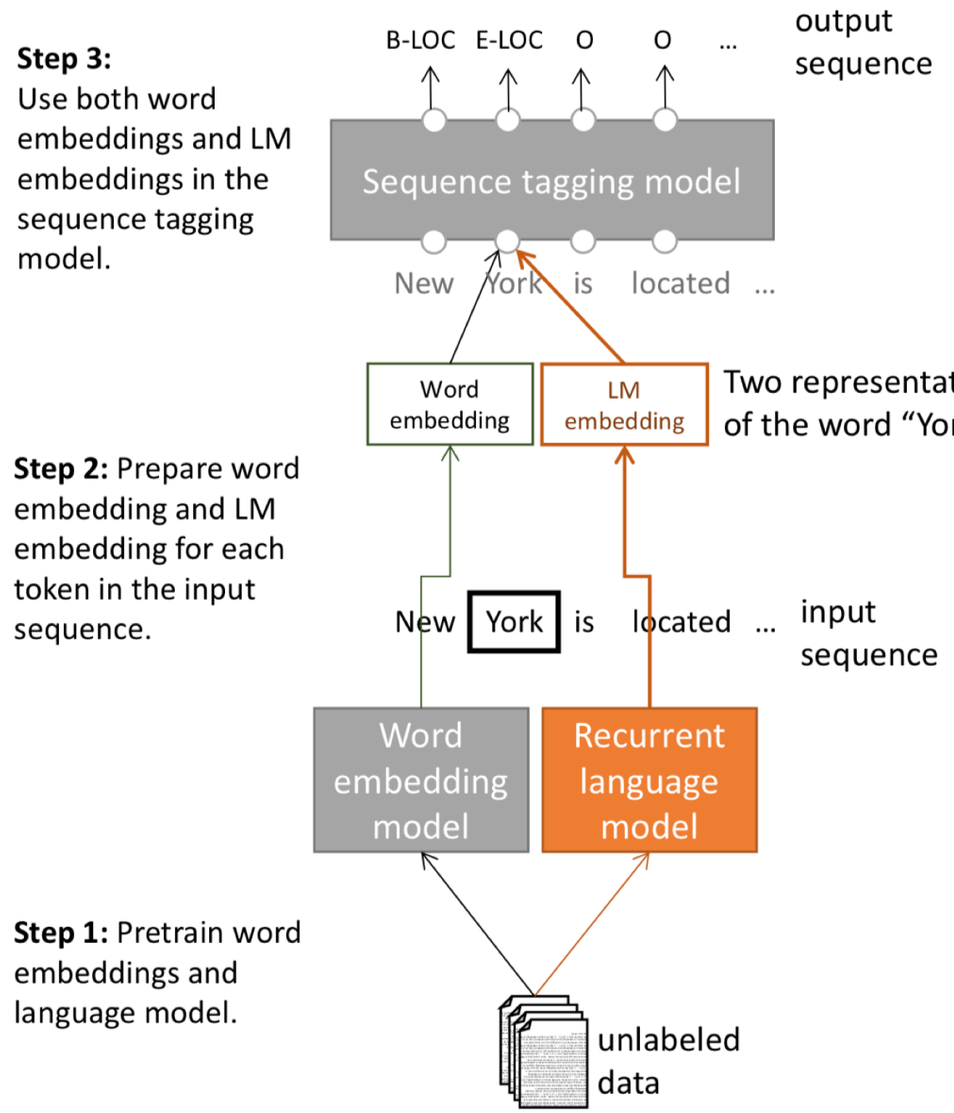
# Do we have a solution already?

- In an NLM, we immediately stuck word vectors through LSTM layers
- Those LSTM layers are trained to predict the next word
- These LMs are producing context-specific word representations at each position



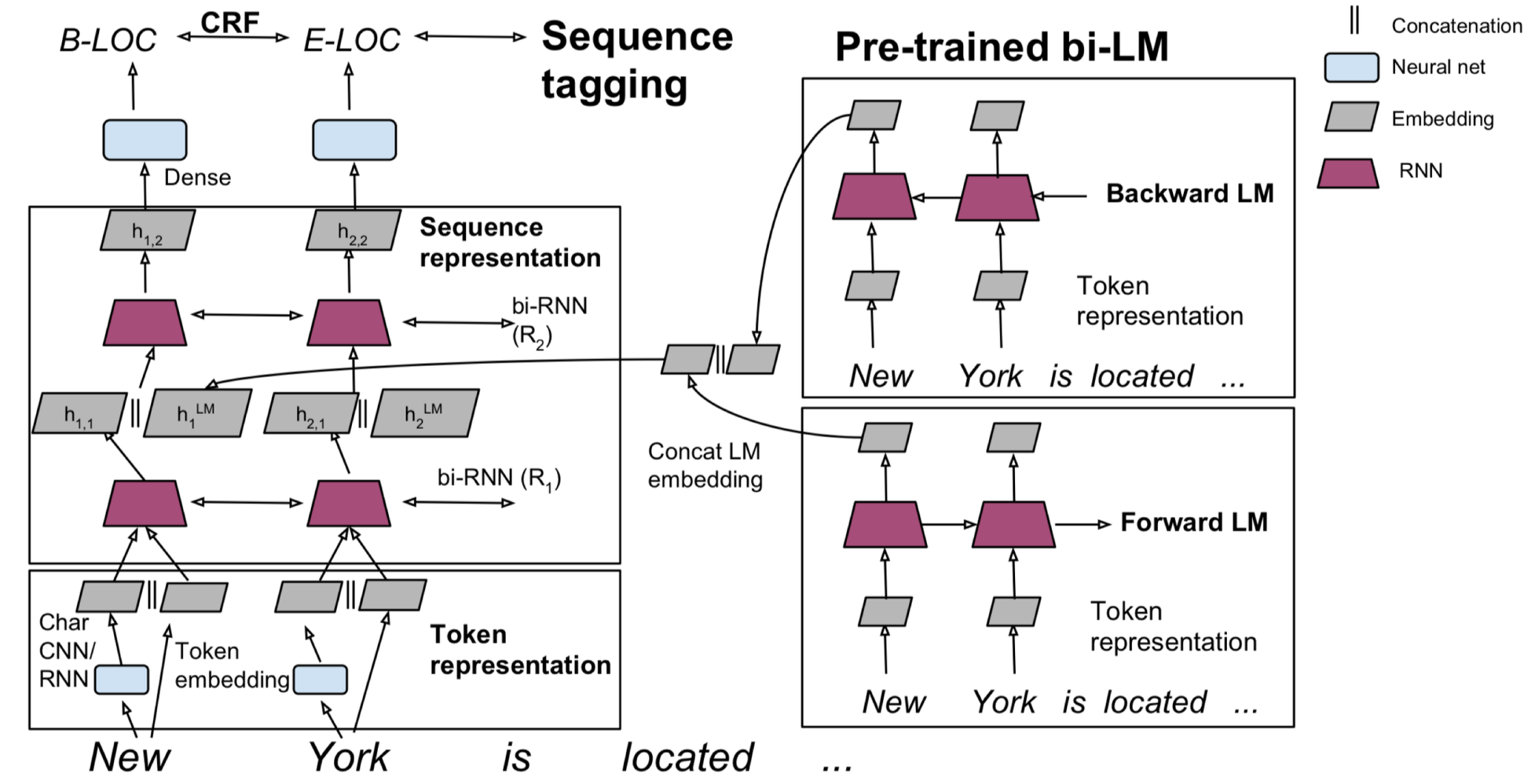
# TagLM – "Pre-ELMo"

- Idea: Want meaning of word in context, but standardly learn task RNN only on small task-labeled data (e.g., NER)
- Why don't we do semi-supervised approach where we train NLM on large unlabeled corpus, rather than just word vectors?




https://arxiv.org/pdf/1705.00108.pdf

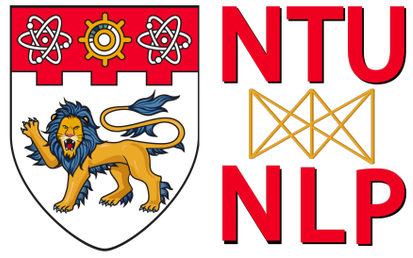
# TagLM - "Pre-ELMo"




$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}]$$

https://arxiv.org/pdf/1705.00108.pdf

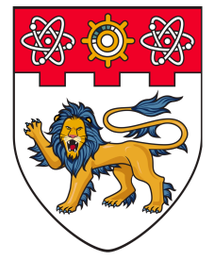
# McCann et al. 2017: CoVe




- Similar idea of using a trained sequence model to provide context to other NLP models
- **Machine translation** is meant to preserve meaning, so maybe that's a good objective?
- Use a 2-layer bi-LSTM that is the encoder of seq2seq + attention NMT system as the context provider
- The resulting CoVe vectors do outperform GloVe vectors on various tasks
- But, the results aren't as strong as the simpler NLM training described in the rest of these slides so seems abandoned.
  - Maybe NMT is just harder than language modeling?
  - Maybe someday this idea will return?

https://arxiv.org/pdf/1705.00108.pdf

# ELMo: Embeddings from Language Models

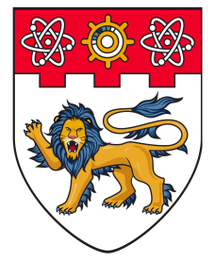


NTU NLP

- Deep **contextualized** word representations.
- Breakout version of word token vectors or contextual word vectors
- Learn word token vectors using long contexts not context windows (here, whole sentence, could be longer)
- Learn a deep Bi-NLM and use all its layers in prediction

https://arxiv.org/abs/1802.05365

# ELMo: Embeddings from Language Models

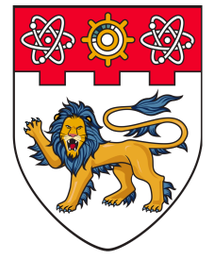


NTU NLP

- Train a bidirectional LM
- Aim at performant but not overly large LM:
  - Use 2 biLSTM layers
  - Use character CNN to build initial word representation (only)
    - 2048 char n-gram filters and 2 highway layers, 512 dim projection
  - Use 4096 dim hidden/cell LSTM states with 512 dim projections to next input
  - Use a residual connection
  - Tie parameters of token input and output (softmax) and tie these between forward and backward LMs

https://arxiv.org/abs/1802.05365

# ELMo: Embeddings from Language Models



NTU NLP

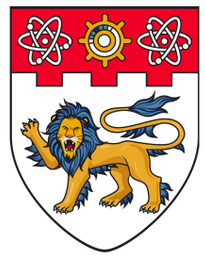
- ELMo learns task-specific combination of biLM representations
- This is an innovation that improves on just using top layer of LSTM stack

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

$$
\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}
$$

- $\gamma^{\text{task}}$ scales overall usefulness of ELMo to task
- $s_j^{task}$ are softmax-normalized mixture model weights

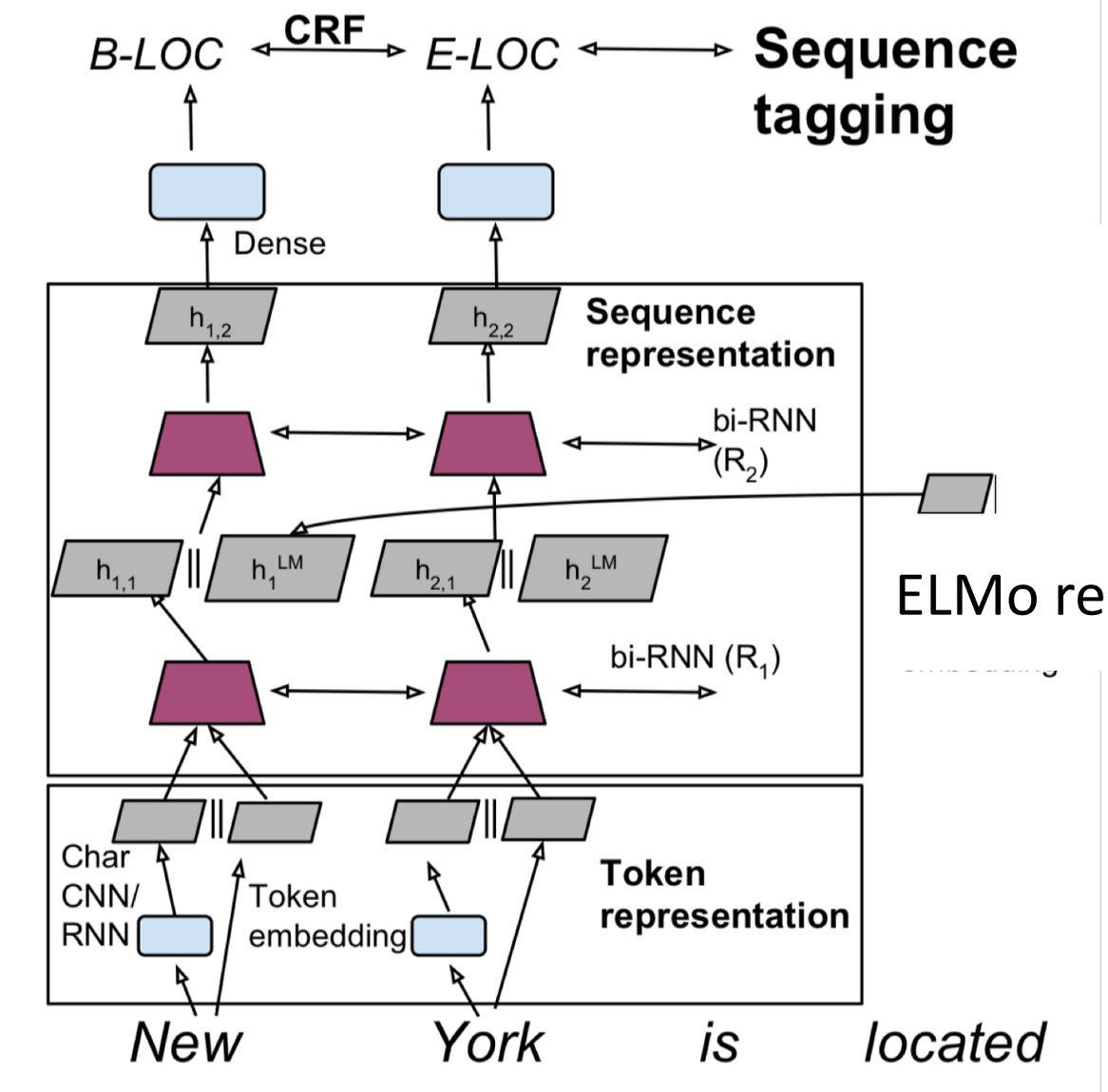
# ELMo: Embeddings from Language Models



NTU NLP

- First run biLM to get representations for each word
- Then let (whatever) end-task model use them
  - Freeze weights of ELMo for purposes of supervised model
  - Concatenate ELMo weights into task-specific model
- Details depend on task
  - Concatenating into intermediate layer as for TagLM is typical
  - Can provide ELMo representations again when producing outputs, as in a question answering system

# ELMo: Embeddings from Language Models




ELMo representation

$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}]$$

# NER Results

| | | | |
|---|---|---|---|
| ELMo | ELMo in BiLSTM | 2018 | 92.22 |
| TagLM Peters | LSTM BiLM in BiLSTM tagger | 2017 | 91.93 |
| Ma + Hovy | BiLSTM + char CNN + CRF layer | 2016 | 91.21 |
| Tagger Peters | BiLSTM + char CNN + CRF layer | 2017 | 90.87 |
| Ratinov + Roth | Categorical CRF+Wikipeda+word cls | 2009 | 90.80 |
| Finkel et al. | Categorical feature CRF | 2005 | 86.86 |
| IBM Florian | Linear/softmax/TBL/HMM ensemble, gazettes++ | 2003 | 88.76 |
| Stanford | MEMM softmax markov model | 2003 | 86.07 |

# Great on other tasks as well

| Task | Previous SOTA | | Our baseline | ELMo + baseline | Increase (absolute/ relative) |
|---|---|---|---|---|---|
| SQuAD | Liu et al. (2017) | 84.4 | 81.1 | 85.8 | 4.7 / 24.9% |
| SNLI | Chen et al. (2017) | 88.6 | 88.0 | 88.7 ± 0.17 | 0.7 / 5.8% |
| SRL | He et al. (2017) | 81.7 | 81.4 | 84.6 | 3.2 / 17.2% |
| Coref | Lee et al. (2017) | 67.2 | 67.2 | 70.4 | 3.2 / 9.8% |
| NER | Peters et al. (2017) | 91.93 ± 0.19 | 90.15 | 92.22 ± 0.10 | 2.06 / 21% |
| SST-5 | McCann et al. (2017) | 53.7 | 51.4 | 54.7 ± 0.5 | 3.3 / 6.8% |

# Some findings

The two biLSTM NLM layers have differentiated uses/meanings

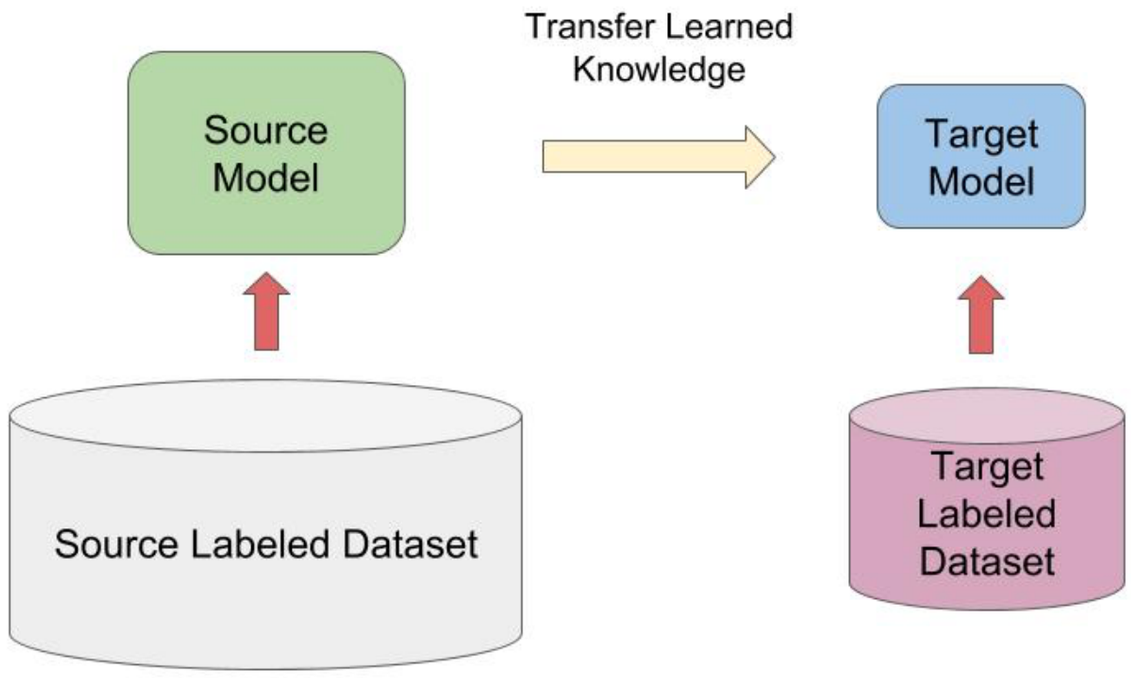
- Lower layer is better for lower-level syntax, etc.
  - Part-of-speech tagging, syntactic dependencies, NER
- Higher layer is better for higher-level semantics
  - Sentiment, Semantic role labeling, question answering, SNLI

This seems interesting, but it'd seem more interesting to see how it pans out with more than two layers of network

# ULMfit

Howard and Ruder (2018) Universal Language Model Fine-tuning for Text Classification.

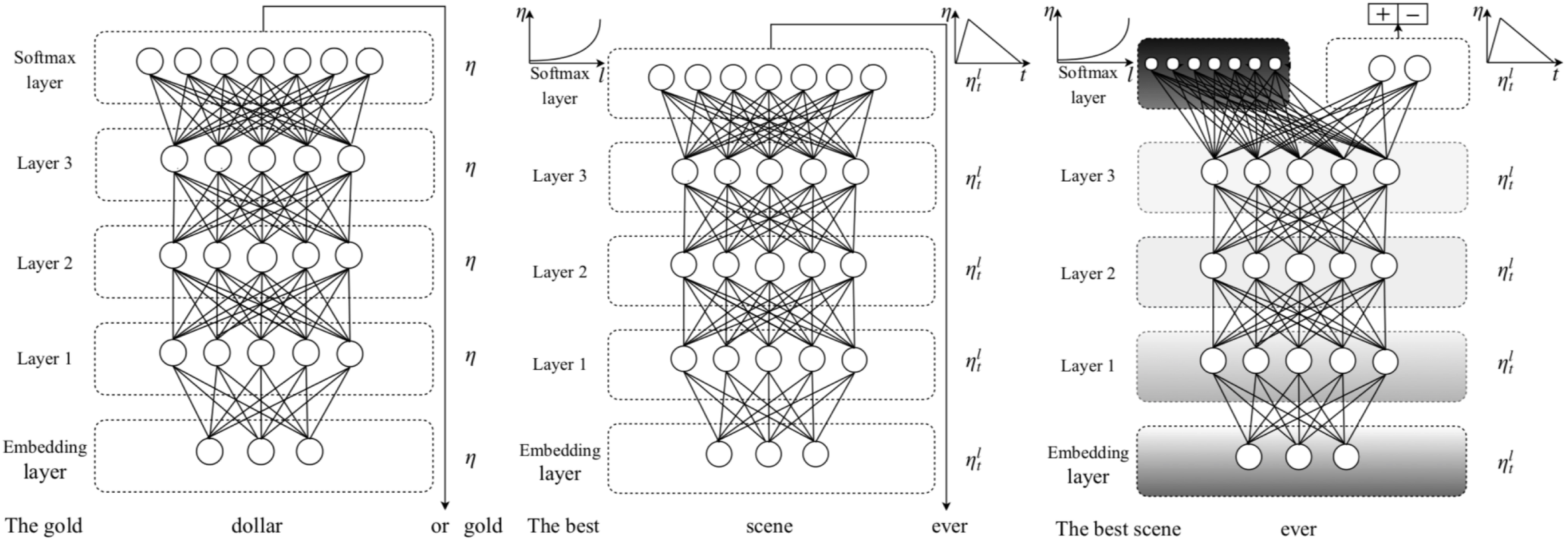
- Same general idea of transferring NLM knowledge
- Here applied to text classification




https://arxiv.org/pdf/1801.06146.pdf

# ULMfit

- Train LM on big general domain corpus (use biLM)
- Tune LM on target task data
- Fine-tune as classifier on target task




(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

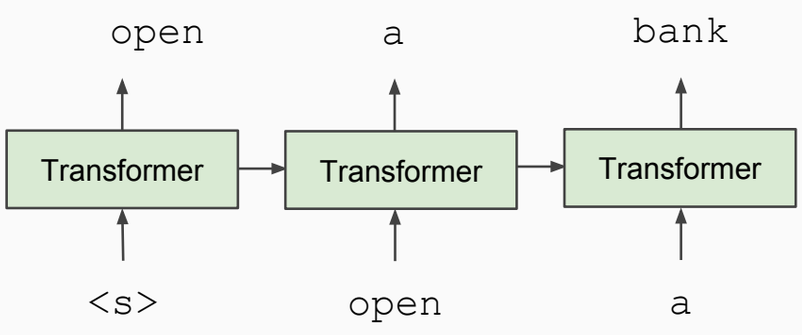
# Scaling Up!

| ULMfit | GPT | BERT | GPT-2 |
|---|---|---|---|
| Jan 2018 | June 2018 | Oct 2018 | Feb 2019 |
| Training: 1 GPU day | Training 240 GPU days | Training 256 TPU days ~320–560 GPU days | Training ~2048 TPU v3 days according to a reddit thread |
| fast.ai | OpenAI | Google AI | OpenAI |

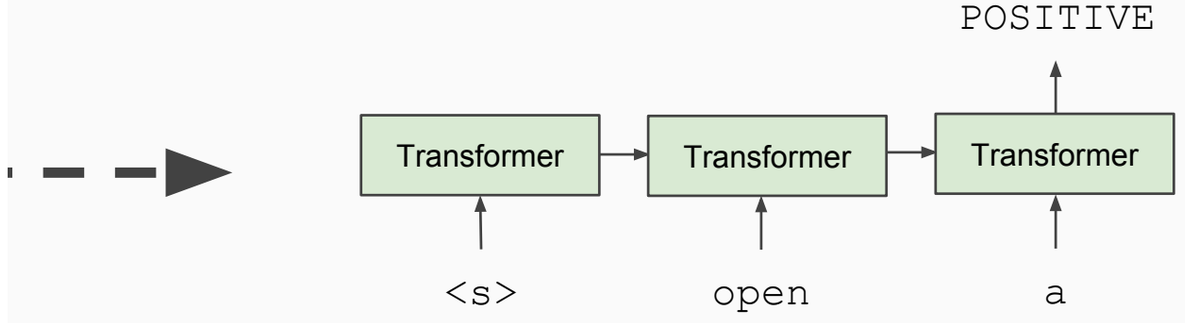
# GPT & GPT-2

Main Idea: Train a Transformer Decoder as conditional LM

**Train Deep (12-layer) Transformer LM**




**Fine-tune on Classification Task**




*Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018

# GPT-2 language model (cherry-picked) output

| | |
|---|---|
| SYSTEM PROMPT (HUMAN-WRITTEN) | *In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.* |
| MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES) | The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.<br><br>Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.<br><br>Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.<br><br>Pérez and the others then ventured further into the valley. … |

METRO NEWS... BUT NOT AS YOU KNOW IT

NEWS SPORT ENTERTAINMENT SOAPS MORE TRENDING

UK WORLD WEIRD TECH

# Elon Musk's OpenAI builds artificial intelligence so powerful it must be kept locked up for the good of humanity

Jasper Hamill Friday 15 Feb 2019 10:06 am

272 SHARES

**Elon Musk's scientists have announced the creation of a terrifying artificial intelligence that's so smart they refused to release it to the public.**

OpenAI's GPT-2 is designed to write just like a human and is an impressive leap forward capable of penning chillingly convincing text.

It was 'trained' by analysing eight million web pages and is capable of writing large tracts based upon a 'prompt' written by a real person.

But the machine mind will not be released in its fully-fledged form because of the risk of it being used for 'malicious purposes' such as generating fake news, impersonating people online, automating the production of spam or churning out 'abusive or faked content to post on social media'.

OpenAI wrote: 'Due to our concerns about malicious applications of the technology, we are not releasing the trained model.

Elon Musk @elonmusk

Follow

Replying to @georgezachary

To clarify, I've not been involved closely with OpenAI for over a year & don't have mgmt or board oversight

8:19 PM - 16 Feb 2019

500 Retweets 14,573 Likes

229 500 15K

# Scaling Up!

All of these models are Transformer architecture models; we already know Transformer

| ULMfit | GPT | BERT | GPT-2 |
|---|---|---|---|
| Jan 2018 | June 2018 | Oct 2018 | Feb 2019 |
| Training: 1 GPU day | Training 240 GPU days | Training 256 TPU days ~320–560 GPU days | Training ~2048 TPU v3 days according to a reddit thread |
| fast.ai | OpenAI | Google AI | OpenAI |

# BERT

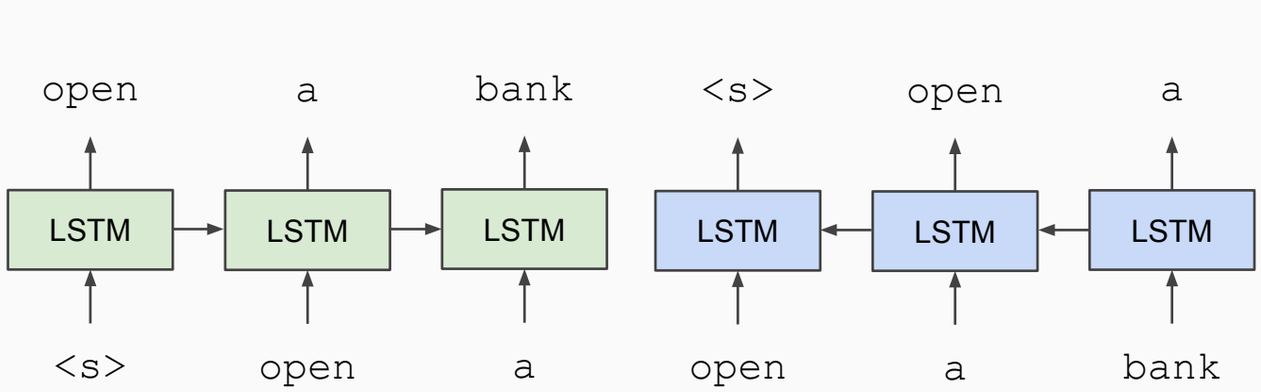
# (Bidirectional Encoder Representations from Transformers)

## Devlin, Chang, Lee, Toutanova (2018)

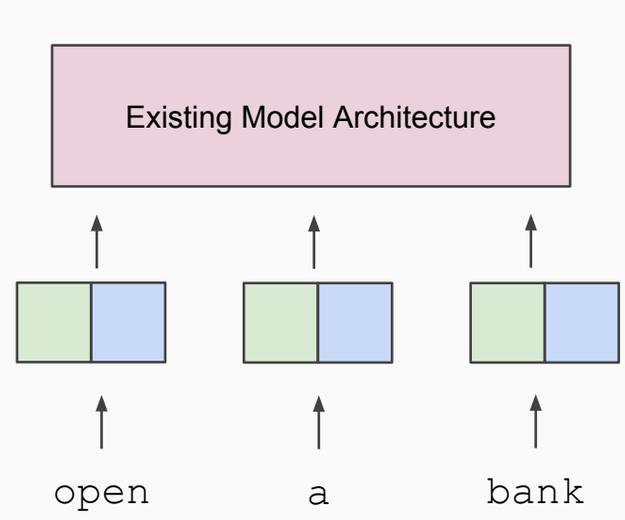
# ELMo: Embeddings from Language Models

NTU NLP

## Train Separate Left-to-Right and Right-to-Left LMs




## Apply as "Pre-trained Embeddings"

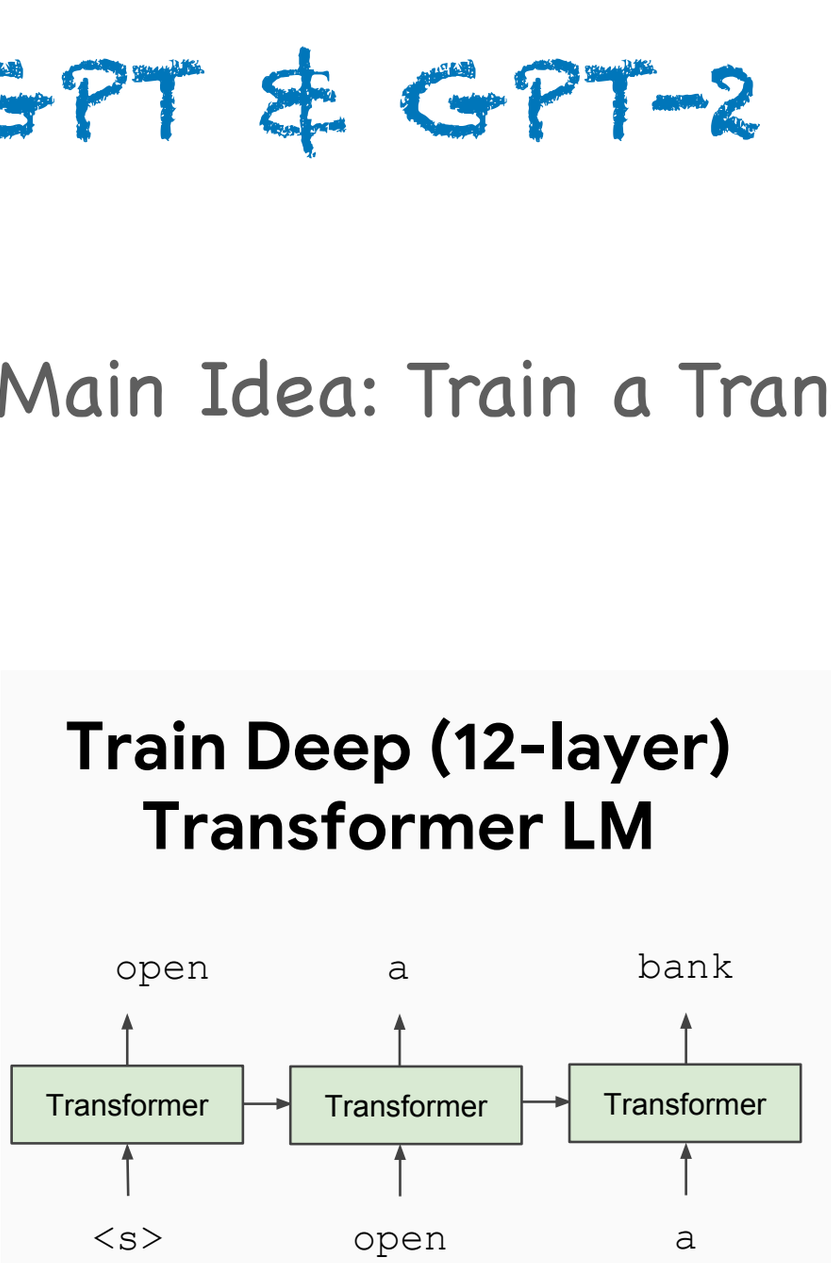



source: Devlin et al. 2019

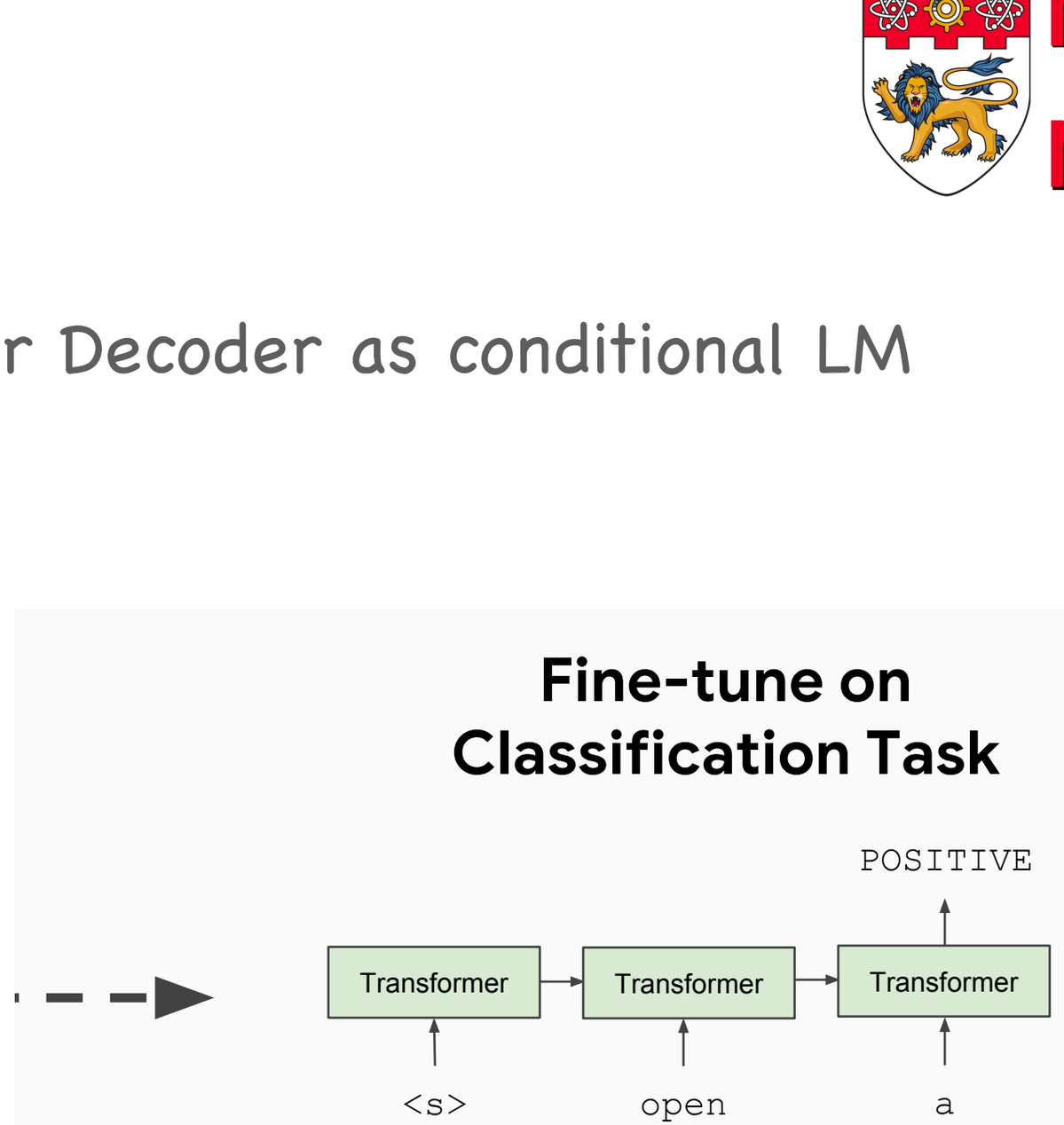
# GPT & GPT-2

Main Idea: Train a Transformer Decoder as conditional LM

**Train Deep (12-layer) Transformer LM**



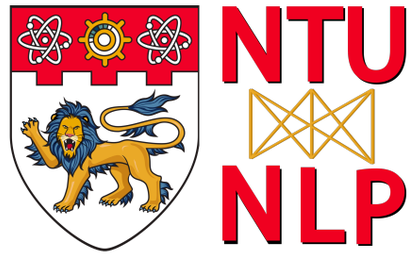

**Fine-tune on Classification Task**




*Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018

source: Devlin et al. 2019

# Why Bidirectionality?

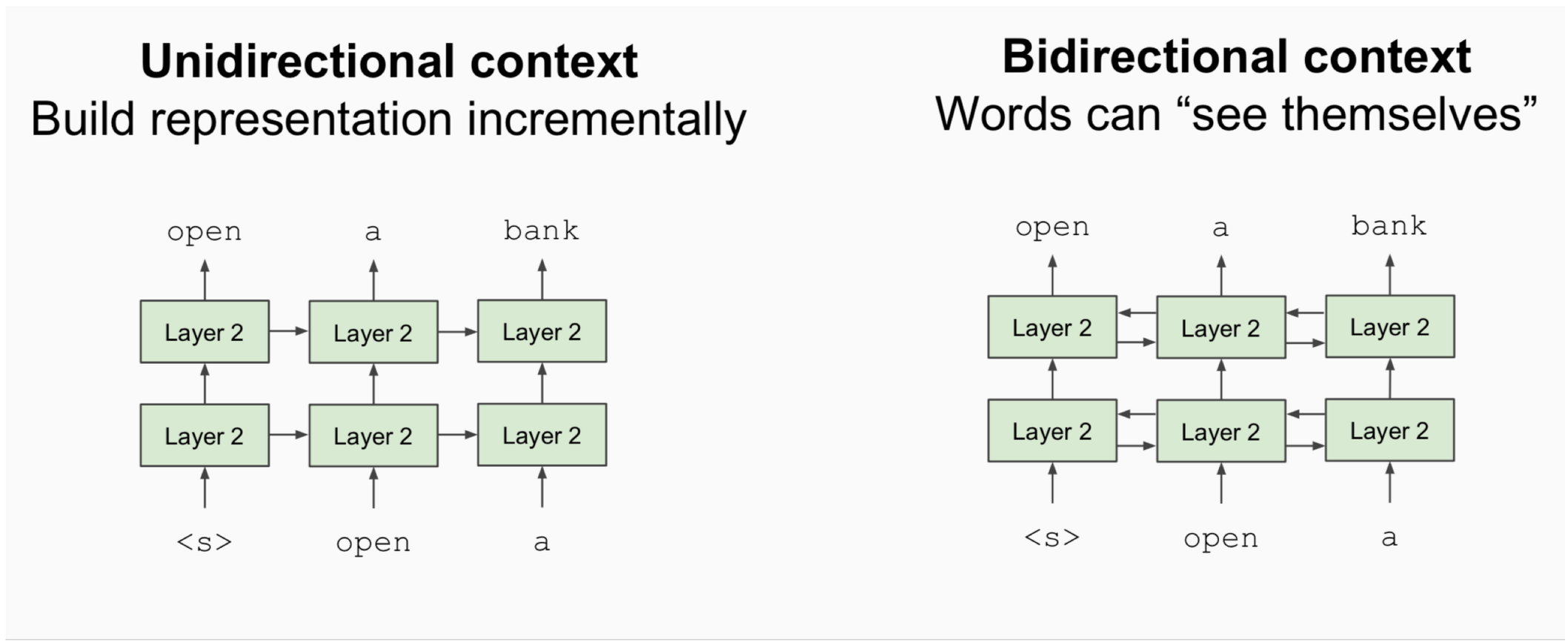



- **Problem**: Language models only use left context or right context, but language understanding is bidirectional.

- Why are LMs unidirectional?
- Reason 1: Directionality is needed to generate a well-formed probability distribution.
  - We don't care about this.
- Reason 2: Words can "see themselves" in a bidirectional encoder.

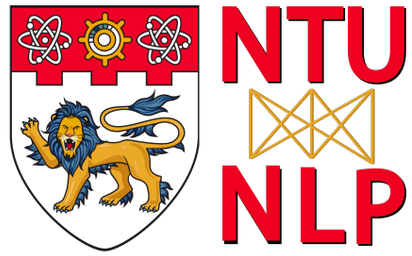
source: Devlin et al. 2019

# Bidirectionality?




source: Devlin et al. 2019

# Masked LM




- **Solution**: Mask out k% of the input words, and then predict the masked words
  - They always use k = 15%

store gallon

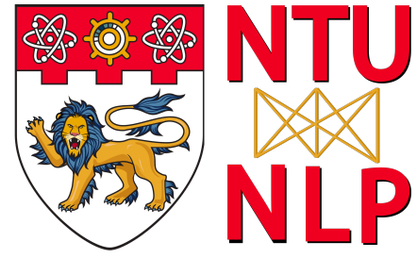
↑ ↑

the man went to the [MASK] to buy a [MASK] of milk

- Too little masking: Too expensive to train
- Too much masking: Not enough context

source: Devlin et al. 2019

# Masked LM




- Problem: Mask token never seen at fine-tuning
- Solution: 15% of the words to predict, but don't replace with [MASK] 100% of the time. Instead:
  - 80% of the time, replace with [MASK]

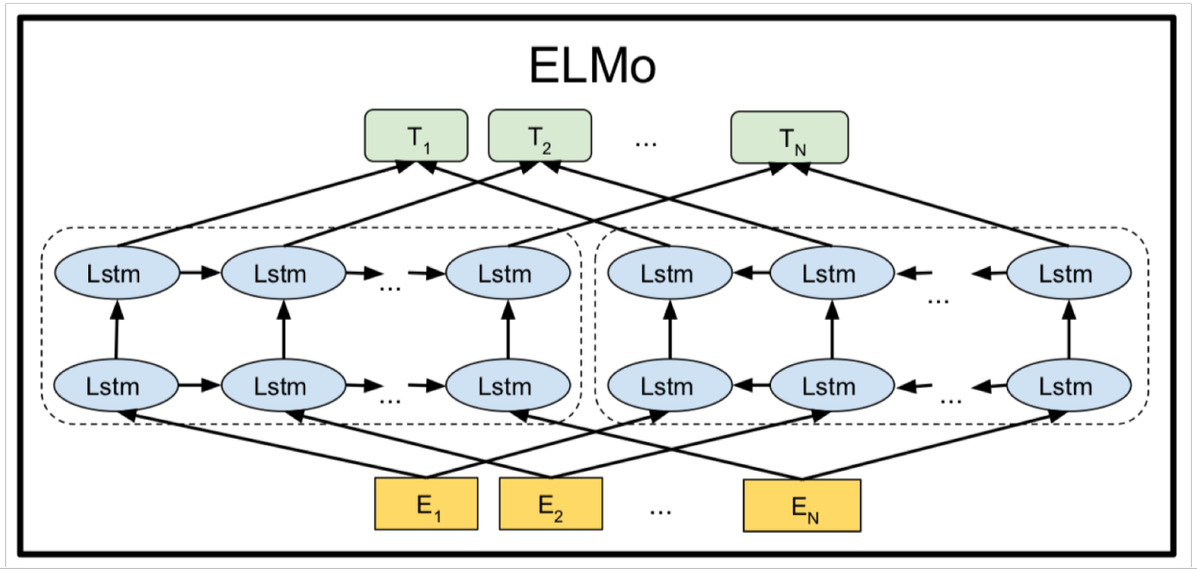
    went to the store → went to the [MASK]
  - 10% of the time, replace random word

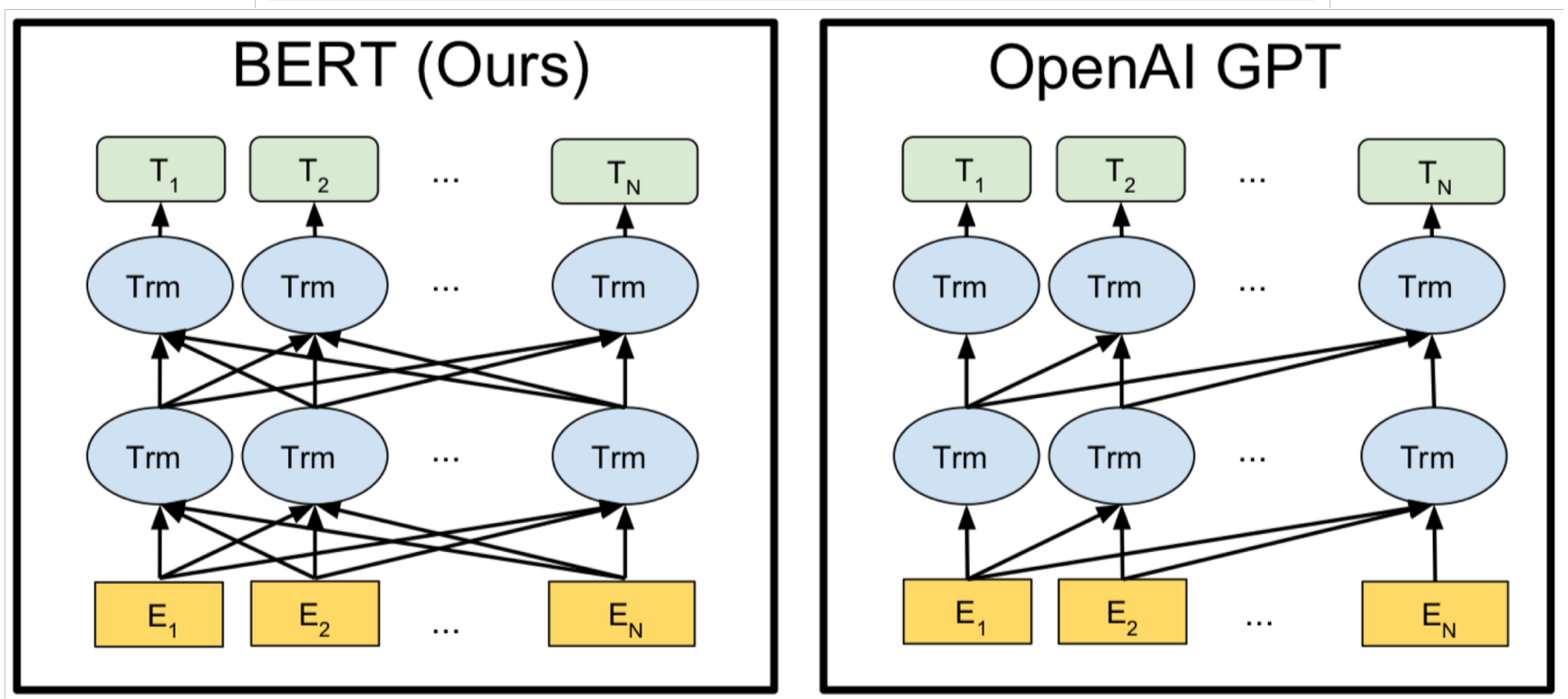
    went to the store → went to the running
  - 10% of the time, keep the same

    went to the store → went to the store

source: Devlin et al. 2019

# Bidirectionality



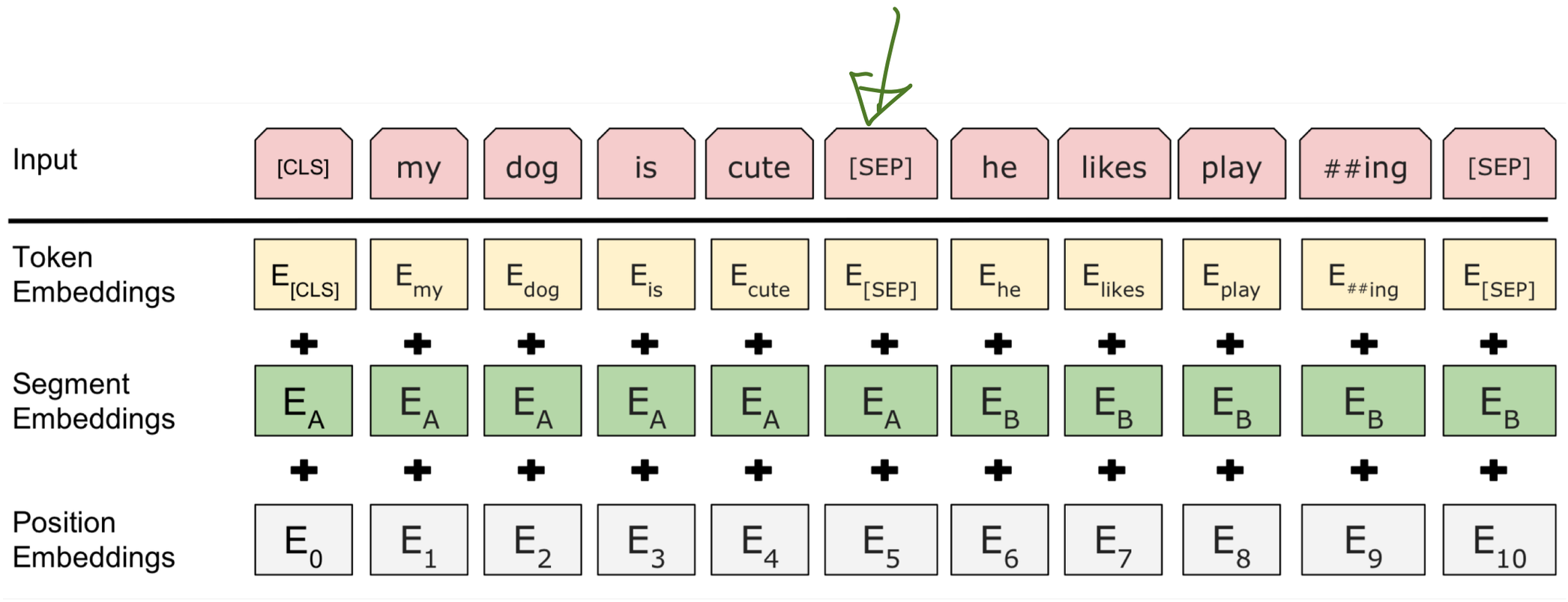



# Next sentence prediction

- To learn relationships between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

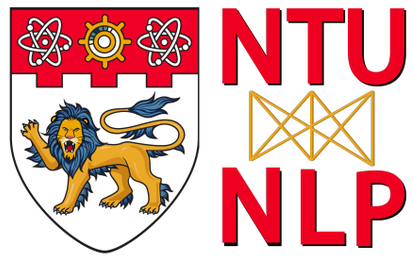
| | |
|---|---|
| **Sentence A** = `The man went to the store.`<br>**Sentence B** = `He bought a gallon of milk.`<br>**Label** = `IsNextSentence` | **Sentence A** = `The man went to the store.`<br>**Sentence B** = `Penguins are flightless.`<br>**Label** = `NotNextSentence` |

# BERT sentence pair encoding




- Token embeddings are word pieces
- Learned segmented embedding represents each sentence
- Positional embedding is as for other Transformer architectures

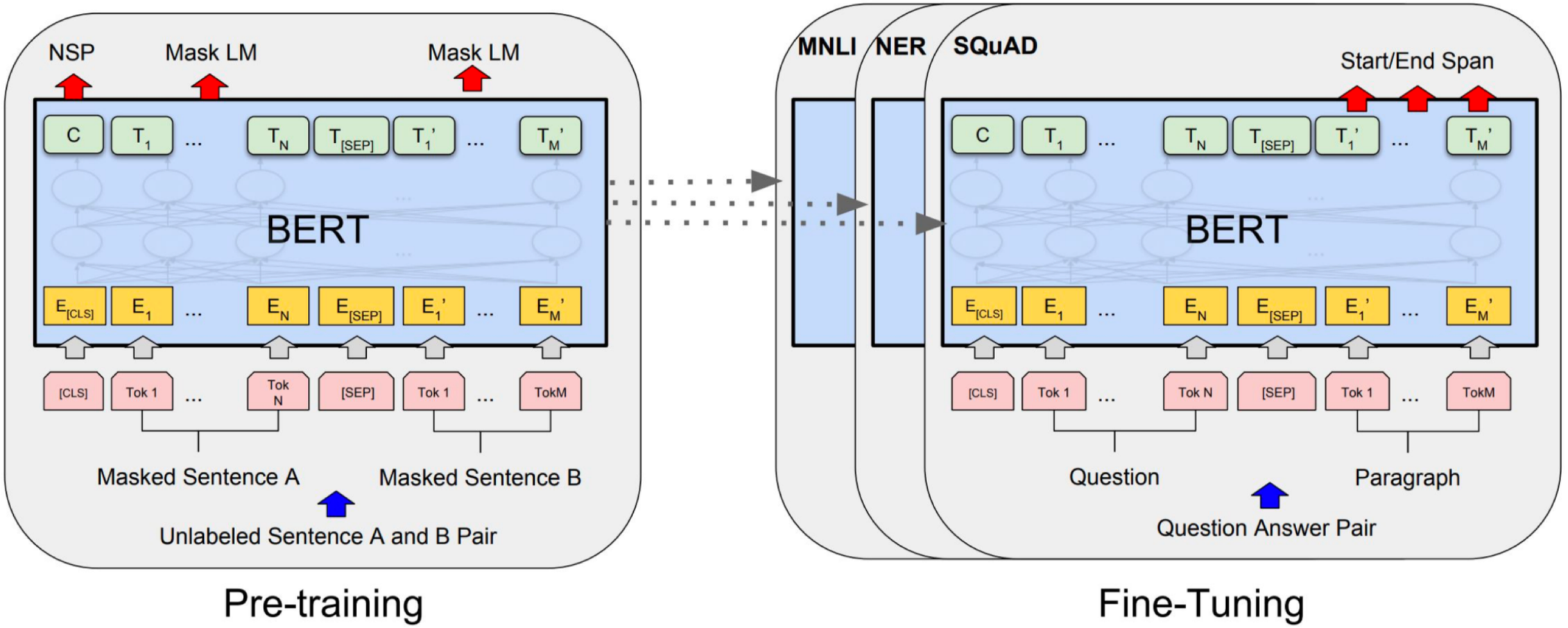
# BERT Model Details



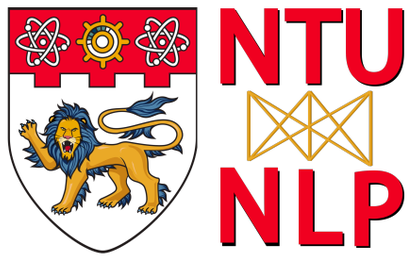

- Transformer encoder (as before)
- Self-attention ⇒ no locality bias
  - Long-distance context has "equal opportunity"
- Single multiplication per layer ⇒ efficiency on GPU/TPU
- Train on Wikipedia + BookCorpus
- Train 2 model sizes:
  - BERT-Base: 12-layer, 768-hidden, 12-head
  - BERT-Large: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPU slice for 4 days

# BERT Fine-tuning

Simply learn a classifier built on the top layer for each task that you fine tune for



# BERT Results (GLUE tasks)




| System | MNLI-(m/mm) 392k | QQP 363k | QNLI 108k | SST-2 67k | CoLA 8.5k | STS-B 5.7k | MRPC 3.5k | RTE 2.5k | **Average** - |
|---|---|---|---|---|---|---|---|---|---|
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.9 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 88.1 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.2 |
| $\text{BERT}_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.1 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| $\text{BERT}_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **91.1** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **81.9** |

- GLUE benchmark is dominated by natural language inference tasks, but also has sentence similarity and sentiment
- MultiNLI
  - Premise: Hills and mountains are especially sanctified in Jainism.
    Hypothesis: Jainism hates nature.
    Label: Contradiction
- CoLa
  - Sentence: The wagon rumbled down the road. Label: Acceptable
    Sentence: The car honked down the road. Label: Unacceptable

# BERT Results (NER)

## CoNLL 2003 Named Entity Recognition (en news testb)

| Name | Description | Year | F1 |
|---|---|---|---|
| Flair (Zalando) | Character-level language model | 2018 | 93.09 |
| BERT Large | Transformer bidi LM + fine tune | 2018 | 92.8 |
| CVT Clark | Cross-view training + multitask learn | 2018 | 92.61 |
| BERT Base | Transformer bidi LM + fine tune | 2018 | 92.4 |
| ELMo | ELMo in BiLSTM | 2018 | 92.22 |
| TagLM Peters | LSTM BiLM in BiLSTM tagger | 2017 | 91.93 |
| Ma + Hovy | BiLSTM + char CNN + CRF layer | 2016 | 91.21 |
| Tagger Peters | BiLSTM + char CNN + CRF layer | 2017 | 90.87 |
| Ratinov + Roth | Categorical CRF+Wikipeda+word cls | 2009 | 90.80 |
| Finkel et al. | Categorical feature CRF | 2005 | 86.86 |
| IBM Florian | Linear/softmax/TBL/HMM ensemble, gazettes++ | 2003 | 88.76 |

# BERT Results (SQuAD 1.1)

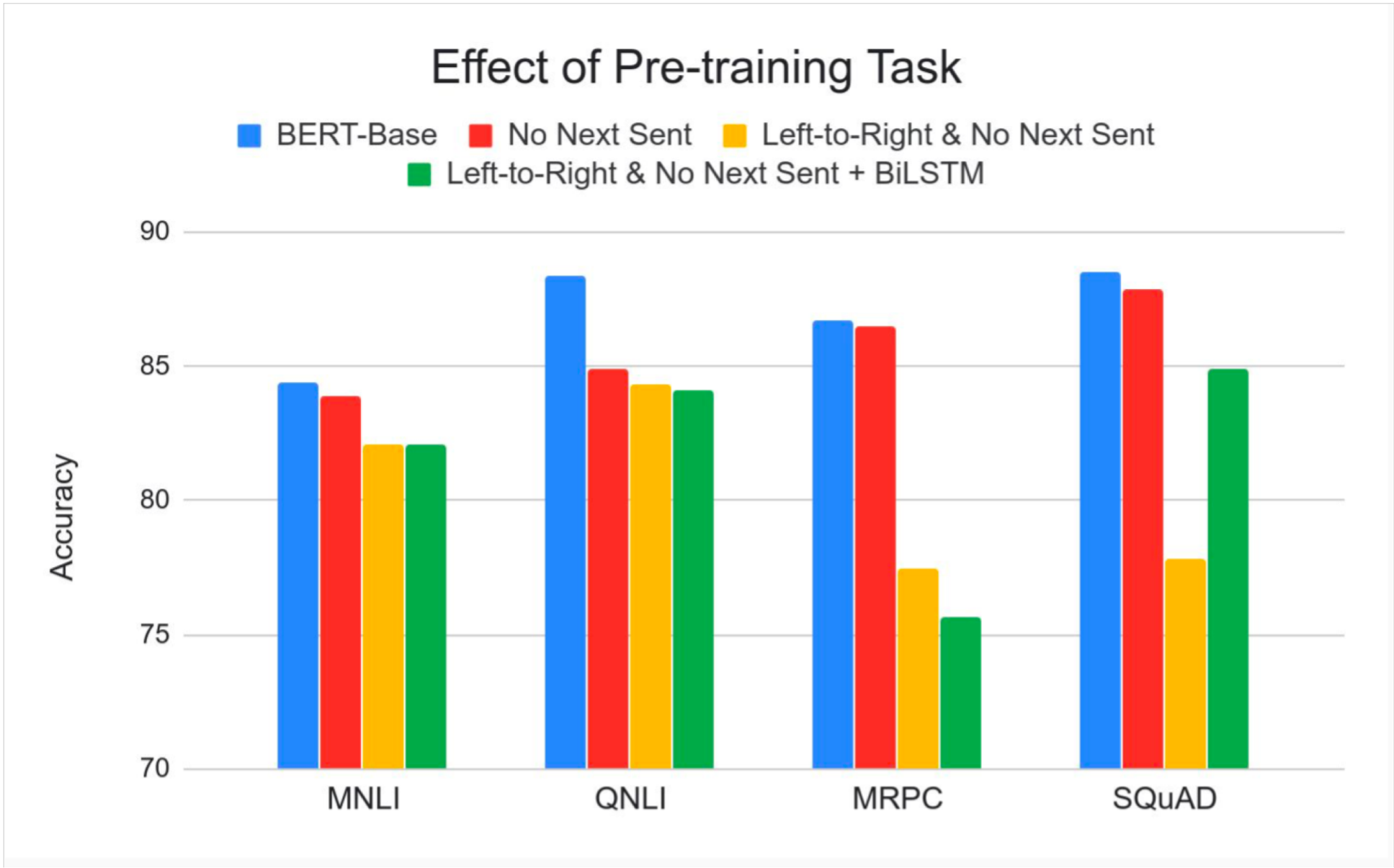
| Rank | Model | EM | F1 |
|---|---|---|---|
| | Human Performance<br>*Stanford University*<br>(Rajpurkar et al. '16) | 82.304 | 91.221 |
| 1<br>Oct 05, 2018 | BERT (ensemble)<br>*Google AI Language*<br>https://arxiv.org/abs/1810.04805 | **87.433** | **93.160** |
| 2<br>Oct 05, 2018 | BERT (single model)<br>*Google AI Language*<br>https://arxiv.org/abs/1810.04805 | 85.083 | 91.835 |
| 2<br>Sep 26, 2018 | nlnet (ensemble)<br>*Microsoft Research Asia* | 85.954 | 91.677 |
| 5<br>Sep 09, 2018 | nlnet (single model)<br>*Microsoft Research Asia* | 83.468 | 90.133 |
| 3<br>Jul 11, 2018 | QANet (ensemble)<br>*Google Brain & CMU* | 84.454 | 90.490 |

# BERT Results (SQuAD 2.0)

| Rank | Model | EM | F1 |
|---|---|---|---|
| | Human Performance<br>*Stanford University*<br>(Rajpurkar & Jia et al. '18) | 86.831 | 89.452 |
| 1<br>Jan 15, 2019 | BERT + MMFT + ADA (ensemble)<br>*Microsoft Research Asia* | **85.082** | **87.615** |
| 2<br>Jan 10, 2019 | BERT + Synthetic Self-Training (ensemble)<br>*Google AI Language*<br>https://github.com/google-research/bert | 84.292 | 86.967 |
| 3<br>Dec 13, 2018 | BERT finetune baseline (ensemble)<br>*Anonymous* | 83.536 | 86.096 |
| 4<br>Dec 16, 2018 | Lunet + Verifier + BERT (ensemble)<br>*Layer 6 AI NLP Team* | 83.469 | 86.043 |
| 4<br>Dec 21, 2018 | PAML+BERT (ensemble model)<br>*PINGAN GammaLab* | 83.457 | 86.122 |
| 5<br>Dec 15, 2018 | Lunet + Verifier + BERT (single model)<br>*Layer 6 AI NLP Team* | 82.995 | 86.035 |

# BERT Results

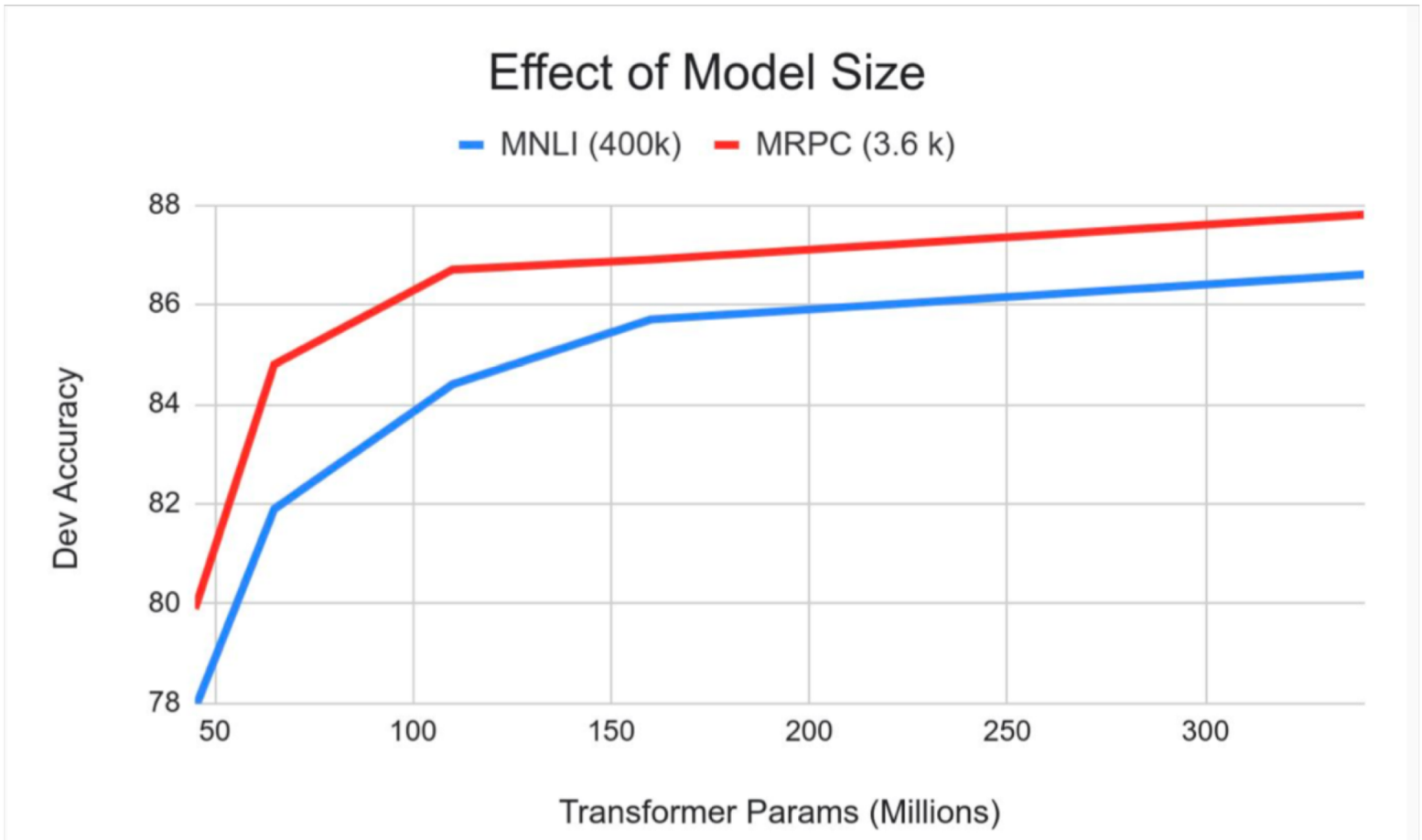
## Effect of pre-training task



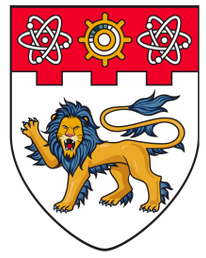
# BERT Results

## Size matters

- Going from 110M to 340M parameters helps a lot
- Improvements have not yet asymptoted



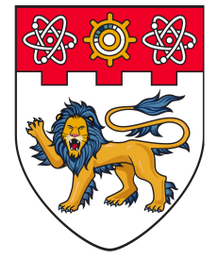
# ROBERTa (Robustly Optimized BERT Pretraining Approach)



NTU NLP

- Improves BERT with better hyperparameter tuning and more (CC) data
  - Training it longer, with bigger batches, over more data;
  - Removing the next sentence prediction objective
  - Training on longer sequences; and
  - Dynamically changing the masking pattern applied to the training data.

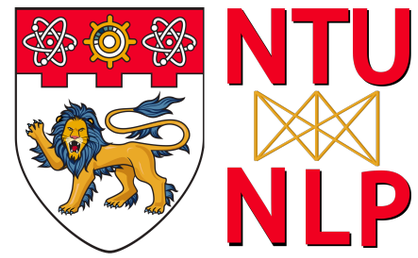
They show that BERT's MLM training is competitive with other recently proposed training objectives such as perturbed autoregressive language model (XLNET)



NTU NLP

# Transformer XL & XLNET

# Transformer



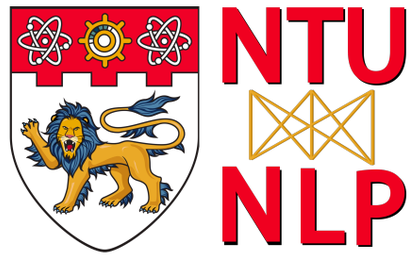

## Simplified pseudocode

```
embeddings = word_embs + pos_embs
h = [embeddings] + [None for _ in attention_layers]

for i, attention in enumerate(attention_layers):
    h[i+1] = attention(queries=h[i], keys=h[i], values=h[i])
```

- Pros: No sequential encoding; parallelizeable
- Cons: Sequence length needs to be fixed; no memory

# Transformer XL (Dai et al.)



Main Idea: Add recurrence between segment

- Cache the hidden states of the previous sequence and pass them as keys/values for processing the current sequence.

```
memory = init_memory()
h = [embeddings] + [None for _ in attention_layers]

for i, attention in enumerate(attention_layers):
    ext_h_i = concat([h[i], memory[i])
    h[i+1] = Attention(queries=h[i], keys=ext_h_i, values=ext_h_i)

memory = h
```

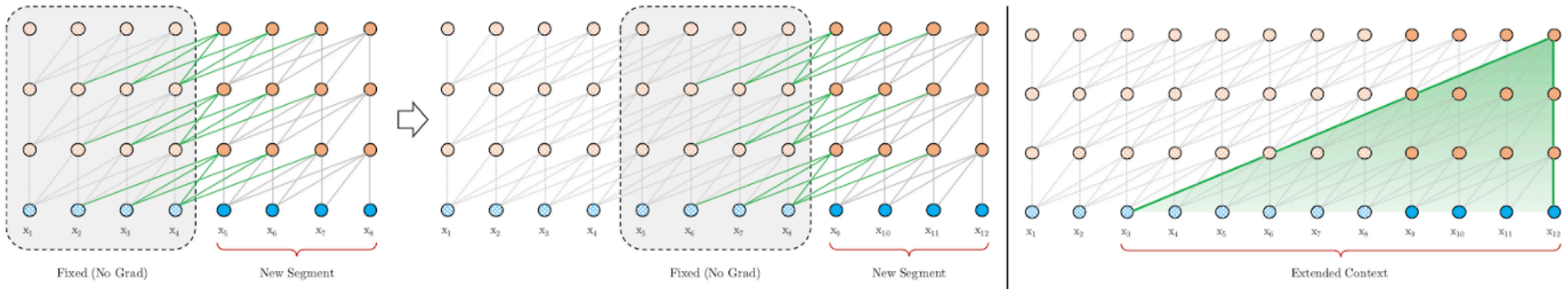
- Use relative positional encoding inside attention computation

- Pros: Allows to model longer sequences, where computation inside a sequence is still parallelizeable

# Transformer XL (Dai et al.)

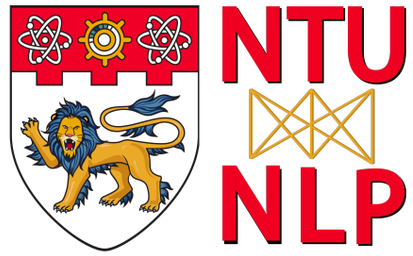
Main Idea: Add recurrence between segment

- Cache the hidden states of the previous sequence and pass them as keys/values for processing the current sequence.




(a) Training phase. (b) Evaluation phase.

# Transformer XL (Dai et al.)



Main Idea: Add recurrence between segment

- Cache the hidden states of the previous sequence and pass them as keys/values for processing the current sequence.

```
memory = init_memory()
h = [embeddings] + [None for _ in attention_layers]

for i, attention in enumerate(attention_layers):
    ext_h_i = concat([h[i], memory[i])
    h[i+1] = Attention(queries=h[i], keys=ext_h_i, values=ext_h_i)

memory = h
```

- Use relative positional encoding inside attention computation

# Relative Positional Representation (Shaw et al, 2018)

NTU NLP

- Transformer

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V)$$

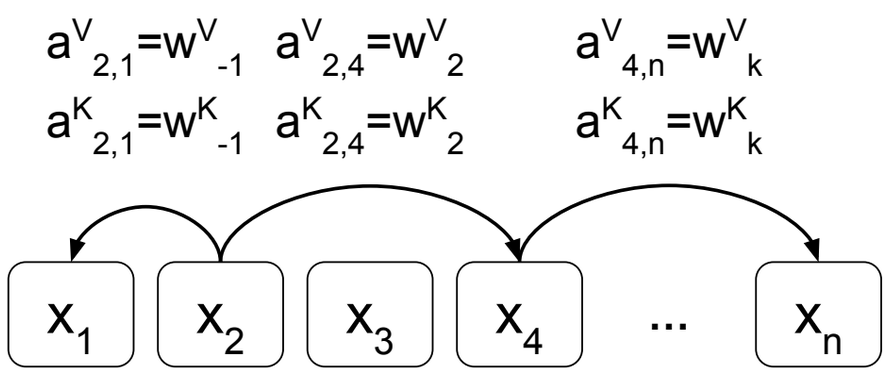
$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}}$$

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}}$$

- Transformer + Relative position

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V)$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}}$$



# Issues with BERT

- Independent prediction of masked tokens

store gallon

↑ ↑

the man went to the [MASK] to buy a [MASK] of milk

Assumes the prediction of "store" and "gallon" are conditionally independent given the unmasked tokens

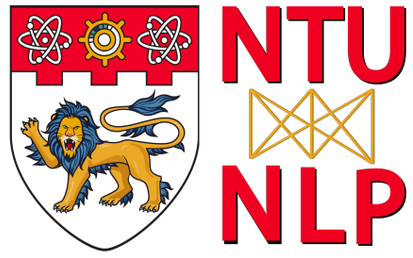
*I went to [MASK] [MASK] and saw the [MASK] [MASK] [MASK].*

*I went to **New York** and saw the **Empire State building**.*

*I went to **San Francisco** and saw the **Golden Gate bridge**.*

*I went to **San Francisco** and saw the **Empire State building***

# Issues with BERT



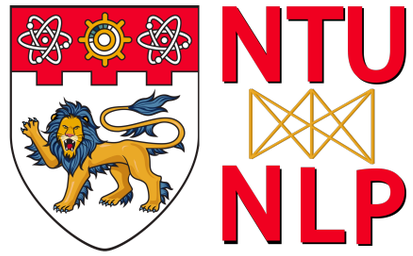

- Mask token never seen at fine-tuning: training-finetune mismatch
  - 10% of the 15% [MASK] time, BERT keeps the same

    went to the **store** → went to the **store**

    If not controlled, the prediction is trivial (just copy the word)

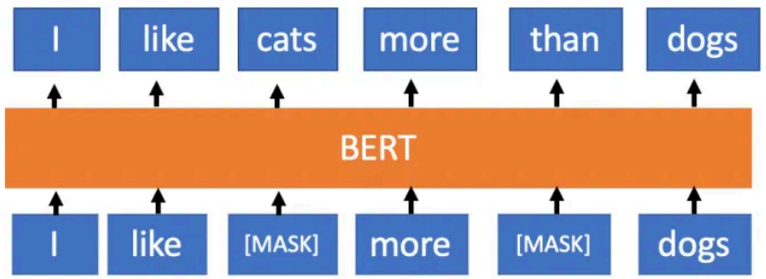
- Auto-regressive LMs (e.g., GPT) do not have these two issues but they lack bidirectionality.
- XLNet attempts to combine the best of both worlds.

# A new Objective: Permutation Language Model

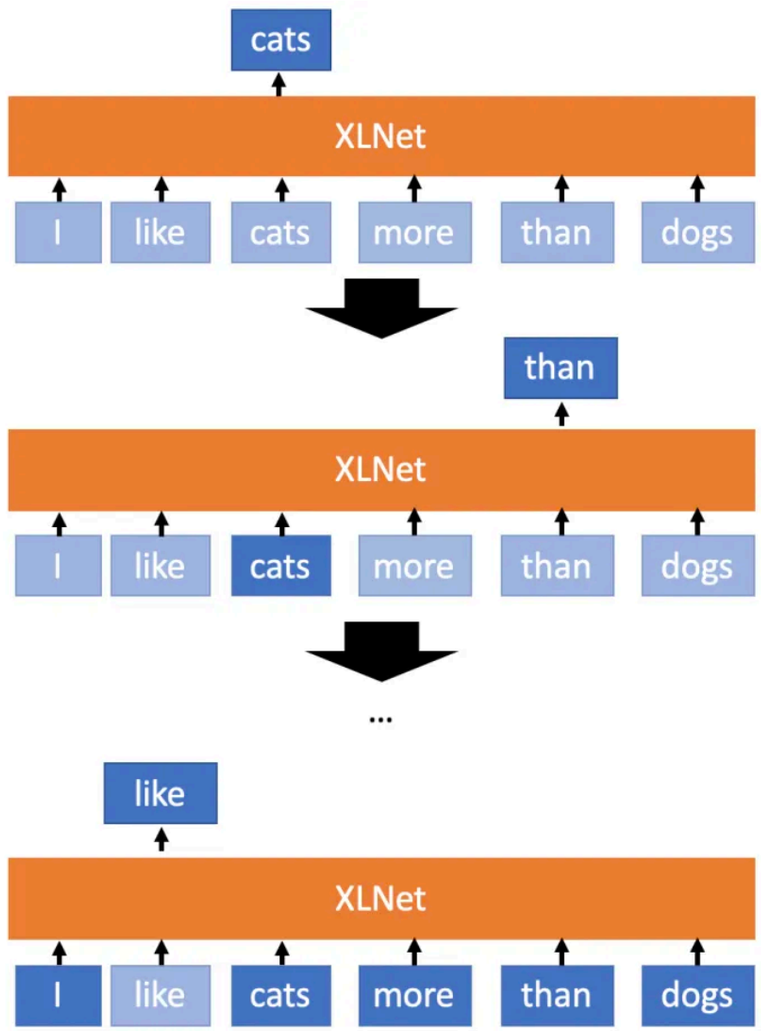



- Order of **prediction/factorisation (not input)** is not necessarily left to right and is sampled randomly instead

*"I", "like", "cats", "more", "than", "dogs"*




*"cats", "than", "I", "more", "dogs", "like"*

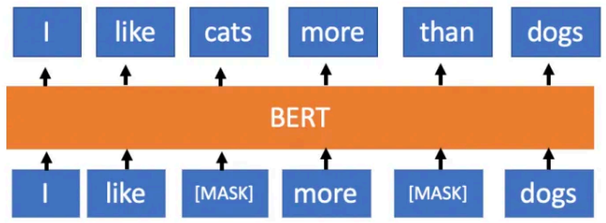


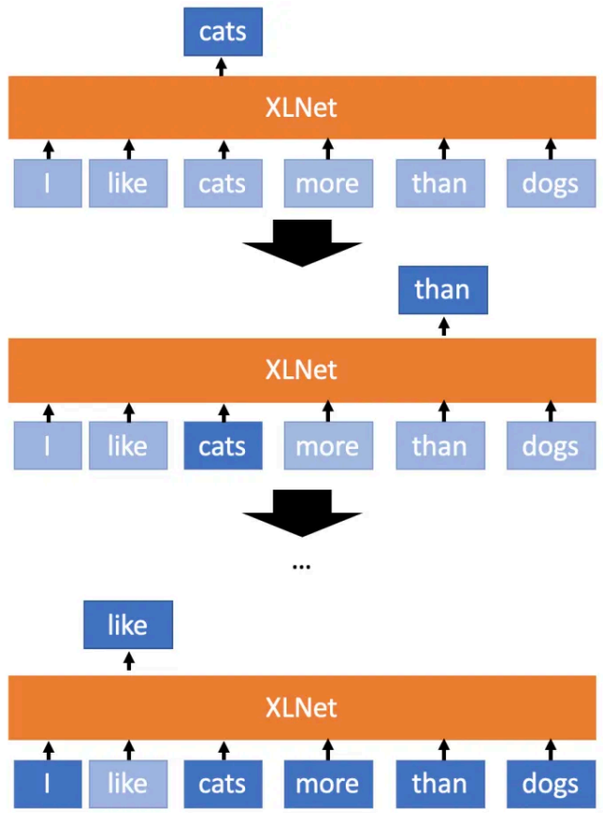

"*than*" is conditioned on seeing "*cats*", "*I*" is conditioned on seeing "*cats*, *than*" and so on.

# A new Objective: Permutation Language Model

NTU NLP

- Order of **prediction/factorisation (not input)** is not necessarily left to right and is sampled randomly instead



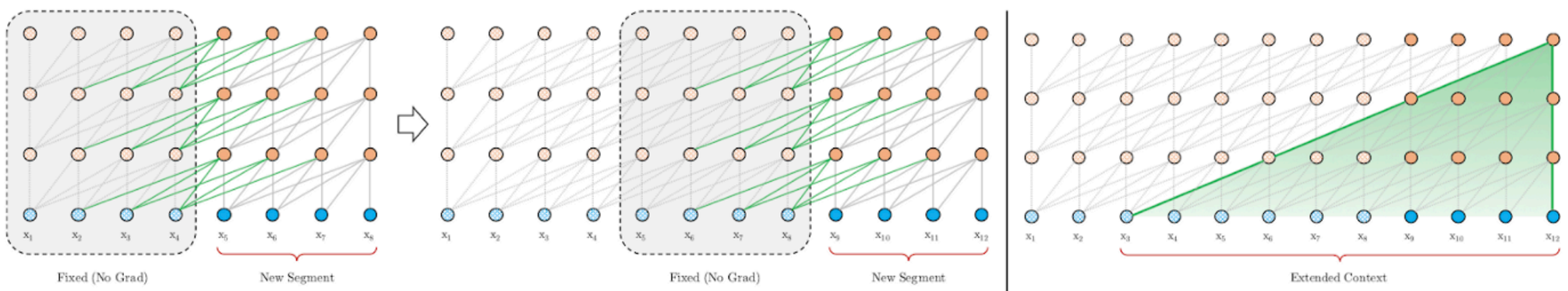

- Notice the order of input (with positional encoding of words) does not change
- In expectation, the model should learn to model the dependencies between all combinations of inputs

# XLNet (Yong et al.)

- Uses **permutation language modelling** objective
- Uses **Transformer XL architecture**

    1. Relative positional embeddings.
    2. Recurrence mechanism.

The hidden states from the previous segment are cached and frozen while conducting the permutation language modeling for the current segment.



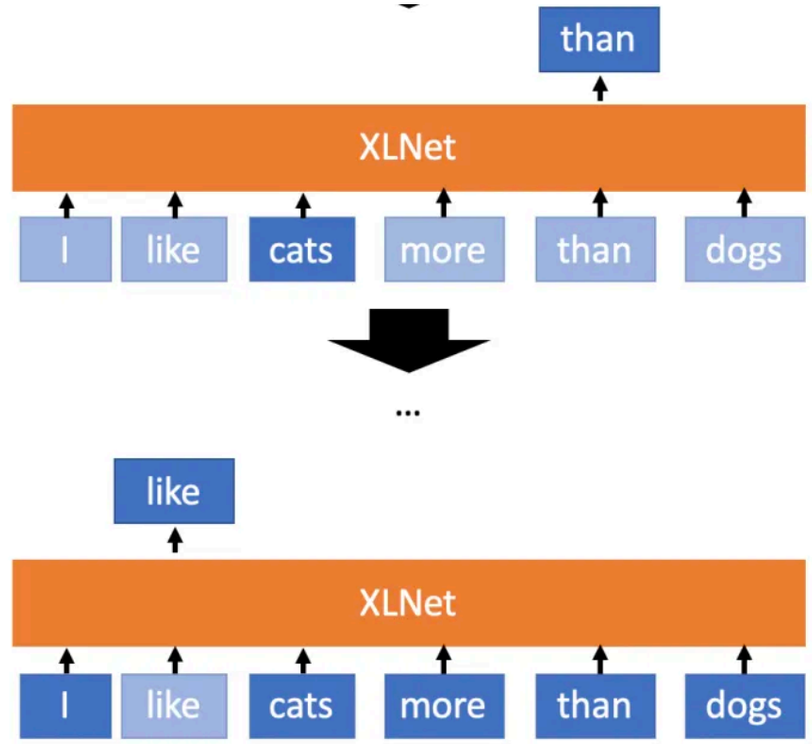

(a) Training phase. (b) Evaluation phase.

# XLNet: Implementation Details

- Standard Transformer parameterization is problematic

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{z_{<t}}) = \frac{\exp\big(e(x)^\top h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})\big)}{\sum_{x'} \exp\big(e(x')^\top h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})\big)}$$

Same for all positions $z_t$



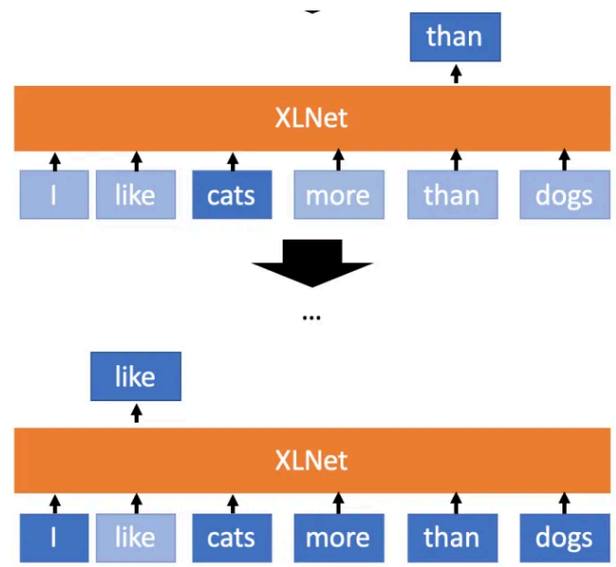

- We need to take the position of the predicted token into consideration

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{z_{<t}}) = \frac{\exp\left(e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}{\sum_{x'} \exp\left(e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}$$

# XLNet: Implementation Details

- Need to take the position of **the predicted token** into consideration

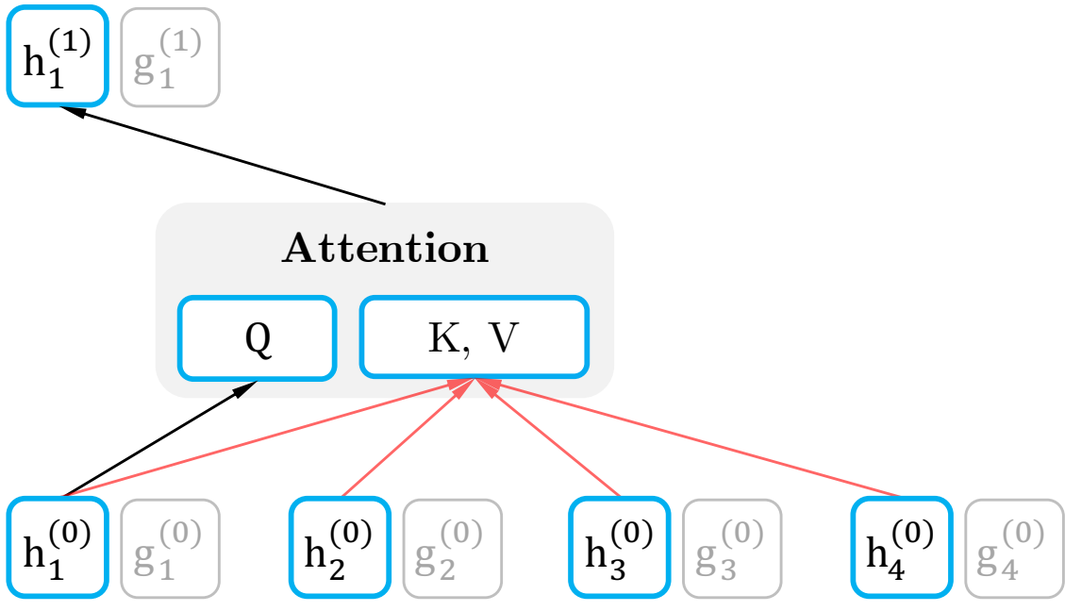
$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{z_{<t}}) = \frac{\exp\left(e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}{\sum_{x'} \exp\left(e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}$$




- However, $g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ should only use the position $z_t$ and not the content $x_{z_t}$, otherwise the objective becomes trivial
- However, to predict the other tokens $x_{z_j}$ with $j>t$, $g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ should also encode the content $x_{z_t}$ to provide full contextual information

# XLNet: Two-Stream Self-Attention

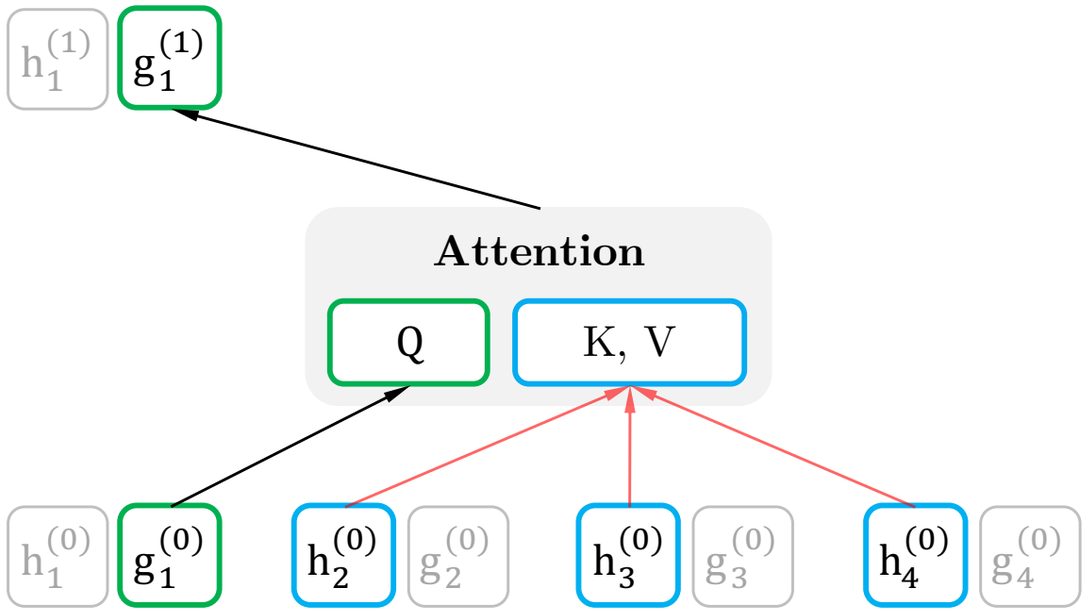
**Content stream** (same as the standard self-attention)




Sample a factorization order:
3 → 2 → 4 → 1

# XLNet: Two-Stream Self-Attention

**Query stream** (does not use the content $x_{z_t}$)

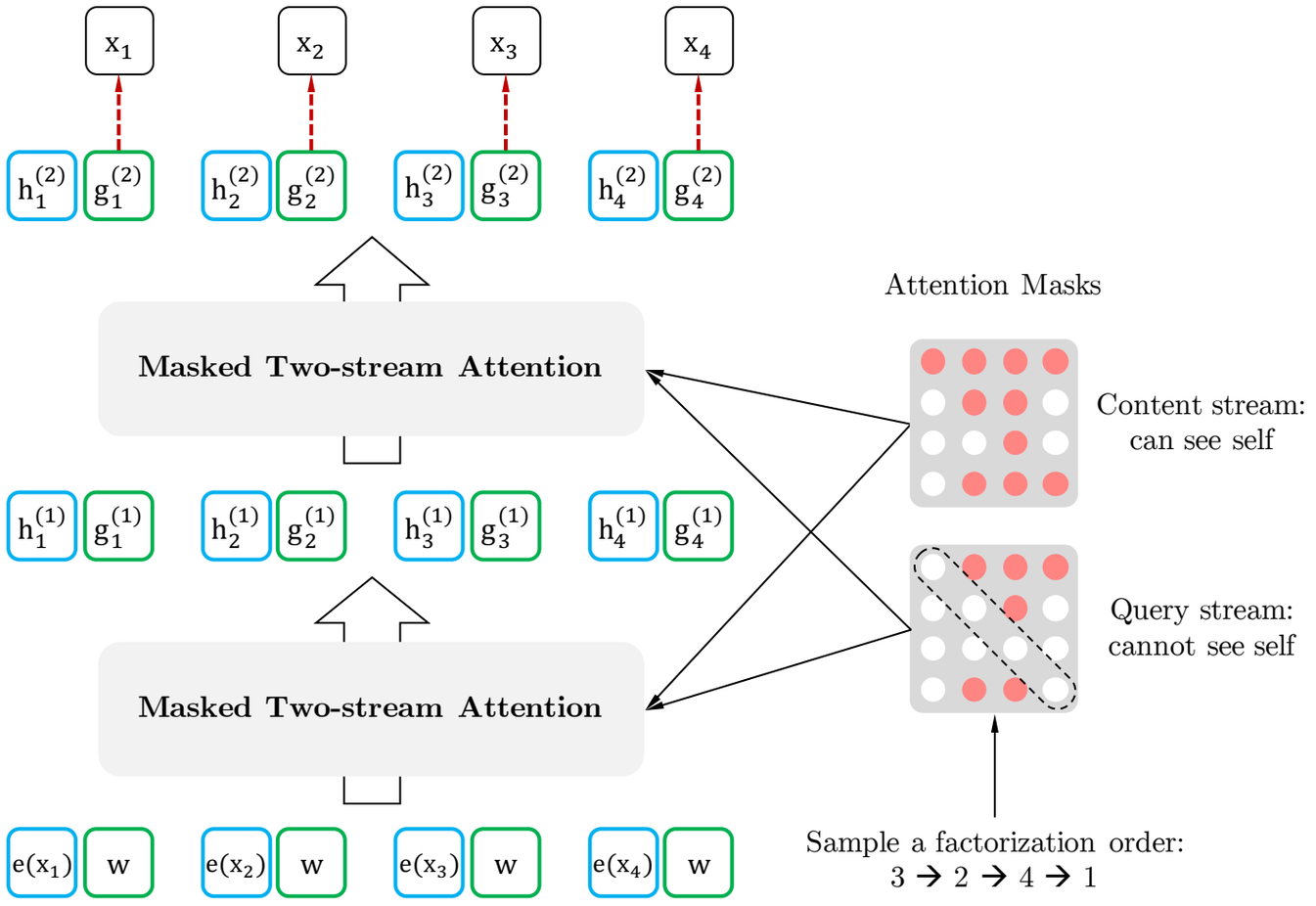



.

Sample a factorization order:

3 → 2 → 4 → 1

# XLNet: Two-Stream Self-Attention

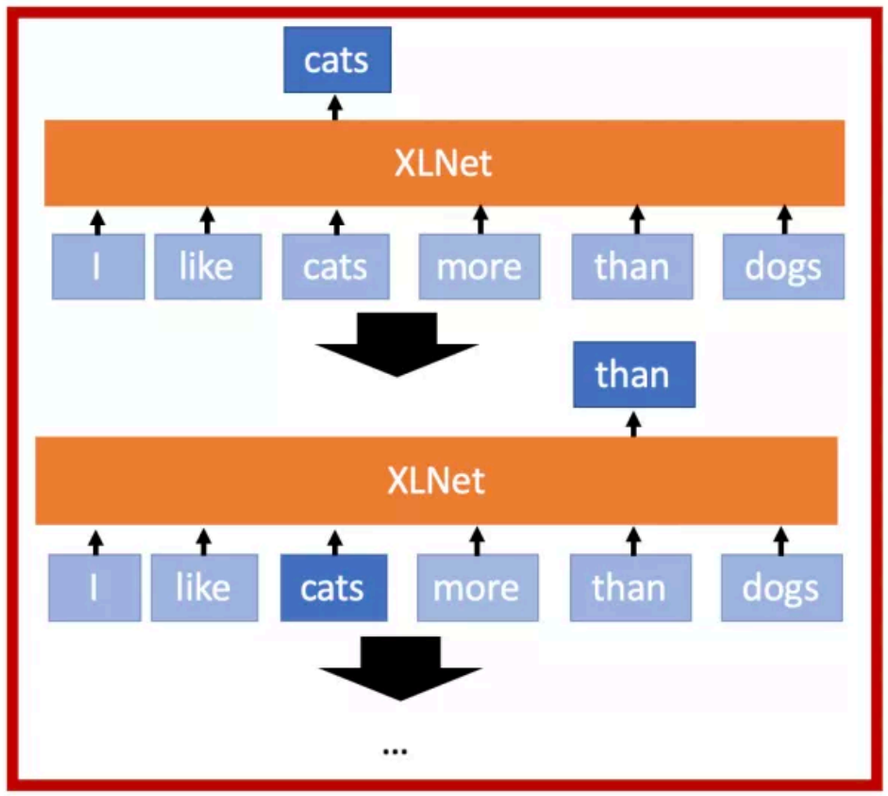



$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = g_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t})$$

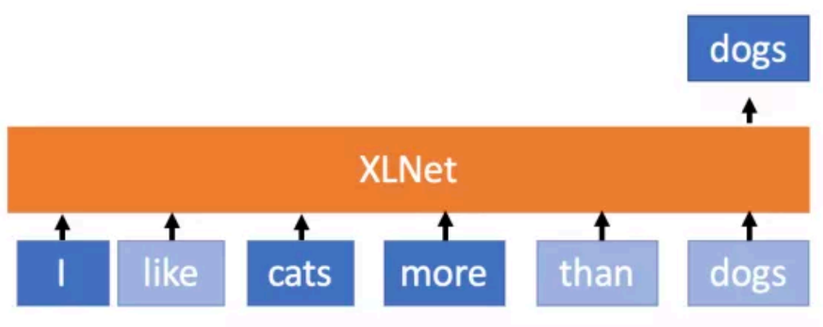
$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}).$$

# XLNet: Optimisation

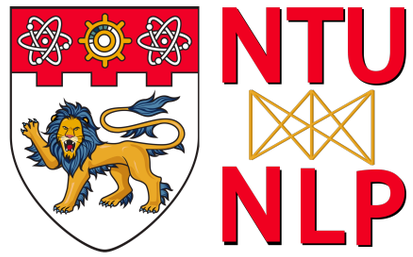
- PLM converges slowly
- Predict only last n words




Skip



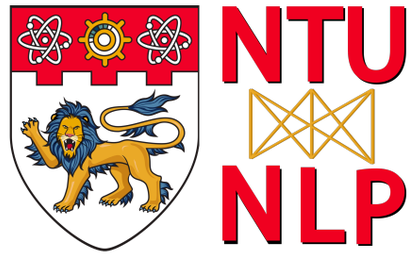
# XLNet Results (GLUE tasks)




| Model | MNLI | QNLI | QQP | RTE | SST-2 | MRPC | CoLA | STS-B | WNLI |
|---|---|---|---|---|---|---|---|---|---|
| *Single-task single models on dev* | | | | | | | | | |
| BERT [2] | 86.6/- | 92.3 | 91.3 | 70.4 | 93.2 | 88.0 | 60.6 | 90.0 | - |
| RoBERTa [21] | 90.2/90.2 | 94.7 | 92.2 | **86.6** | 96.4 | **90.9** | 68.0 | 92.4 | - |
| XLNet | **90.8/90.8** | **94.9** | **92.3** | 85.9 | **97.0** | 90.8 | **69.0** | **92.5** | - |
| *Multi-task ensembles on test (from leaderboard as of Oct 28, 2019)* | | | | | | | | | |
| MT-DNN* [20] | 87.9/87.4 | 96.0 | 89.9 | 86.3 | 96.5 | 92.7 | 68.4 | 91.1 | 89.0 |
| RoBERTa* [21] | 90.8/90.2 | 98.9 | 90.2 | 88.2 | 96.7 | 92.3 | 67.8 | 92.2 | 89.0 |
| XLNet* | **90.9/90.9**† | **99.0**† | **90.4**† | **88.5** | **97.1**† | **92.9** | **70.2** | **93.0** | **92.5** |

- MultiNLI
  - Premise: Hills and mountains are especially sanctified in Jainism.
    Hypothesis: Jainism hates nature.
    Label: Contradiction
- CoLa
  - Sentence: The wagon rumbled down the road. Label: Acceptable
    Sentence: The car honked down the road. Label: Unacceptable

# XLNet Results (Text Classification)



- Error rates

| Model | IMDB | Yelp-2 | Yelp-5 | DBpedia | AG | Amazon-2 | Amazon-5 |
|---|---|---|---|---|---|---|---|
| CNN [15] | - | 2.90 | 32.39 | 0.84 | 6.57 | 3.79 | 36.24 |
| DPCNN [15] | - | 2.64 | 30.58 | 0.88 | 6.87 | 3.32 | 34.81 |
| Mixed VAT [31, 23] | 4.32 | - | - | 0.70 | 4.95 | - | - |
| ULMFiT [14] | 4.6 | 2.16 | 29.98 | 0.80 | 5.01 | - | - |
| BERT [35] | 4.51 | 1.89 | 29.32 | 0.64 | - | 2.63 | 34.17 |
| XLNet | **3.20** | **1.37** | **27.05** | **0.60** | **4.45** | **2.11** | **31.67** |

# XLNet Results (MRC/QA)

| SQuAD2.0 | EM | F1 | SQuAD1.1 | EM | F1 |
|---|---|---|---|---|---|
| *Dev set results (single model)* | | | | | |
| BERT [10] | 78.98 | 81.77 | BERT† [10] | 84.1 | 90.9 |
| RoBERTa [21] | 86.5 | 89.4 | RoBERTa [21] | 88.9 | 94.6 |
| XLNet | **87.9** | **90.6** | XLNet | **89.7** | **95.1** |
| *Test set results on leaderboard (single model, as of Dec 14, 2019)* | | | | | |
| BERT [10] | 80.005 | 83.061 | BERT [10] | 85.083 | 91.835 |
| RoBERTa [21] | 86.820 | 89.795 | BERT* [10] | 87.433 | 93.294 |
| XLNet | **87.926** | **90.689** | XLNet | **89.898**‡ | **95.080**‡ |

# BART & mBART

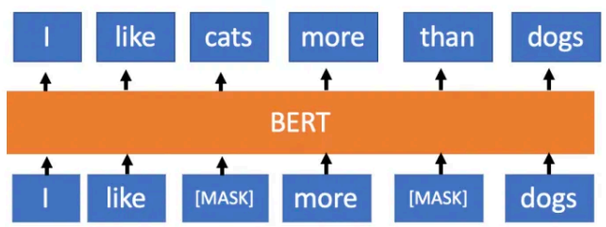
# BART: Denoising Sequence-to-Sequence Pre-training

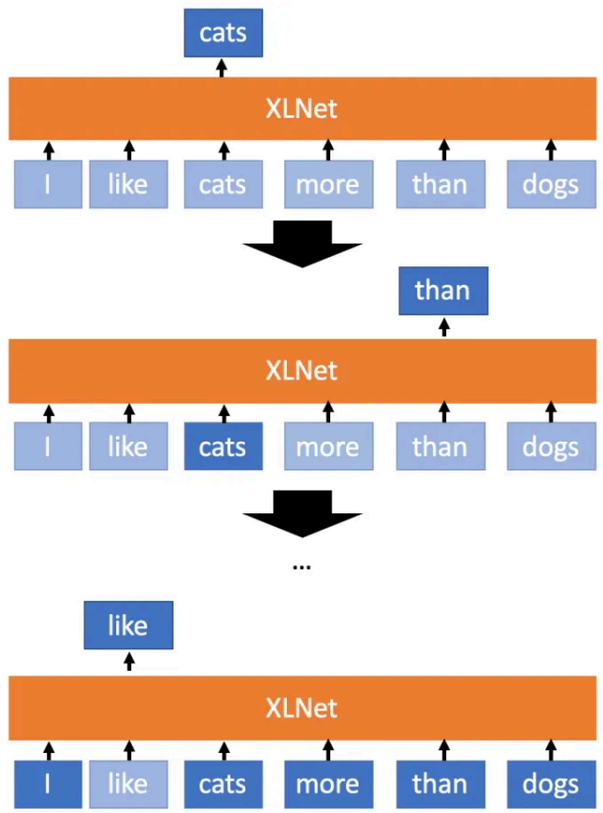
NTU NLP

Predictions are not made auto-regressively by a decoder, reducing the effectiveness of BERT-like models for generation tasks

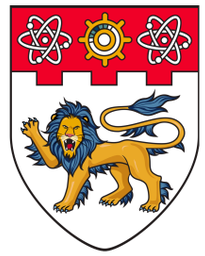
MLM Variants: BERT, RoBERTa, ALBERT, SpanBERT, XLM

PLM: XLNet (auto-regressive, encoder)



# BART: Denoising Sequence-to-Sequence Pre-training

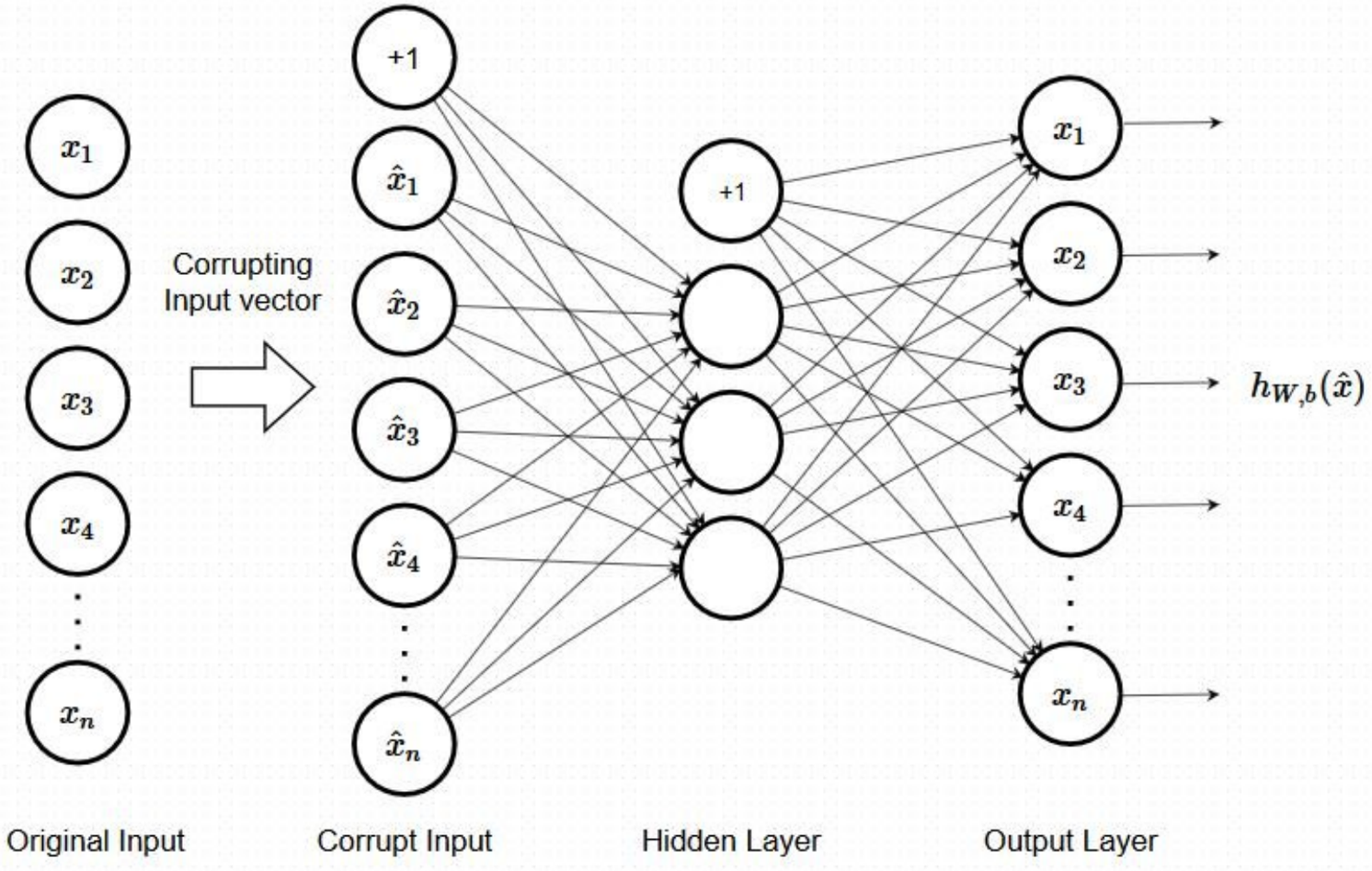


NTU NLP

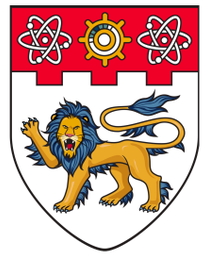
- Generalizes BERT, GPT, and many other more recent pretraining schemes
- Particularly effective when fine tuned for text generation but also works well for discriminative/comprehension tasks $\Longrightarrow$ matches the performance of RoBERTa on GLUE and SQuAD
- Achieves SOTA on a range of abstractive dialogue and question answering, and summarization tasks $\Longrightarrow$ For summarization tasks it improves performance by 6 ROUGE over previous work on XSum

# Denoising Auto-Encoder

- AE generally used for feature selection and extraction
- Denoising AE $\Longrightarrow$ corrupt input data on purpose during training



# BART: Denoising Sequence-to-Sequence Pre-training



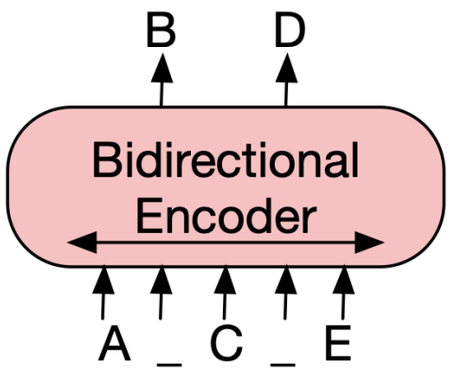
NTU NLP

- A denoising autoencoder built with a sequence-to-sequence model that is applicable to a very wide range of end tasks.
- Pretraining has two stages:
  1. Input Corruption: text is corrupted with an arbitrary noising function
  2. Reconstruction: a sequence-to-sequence model is learned to reconstruct the original text.

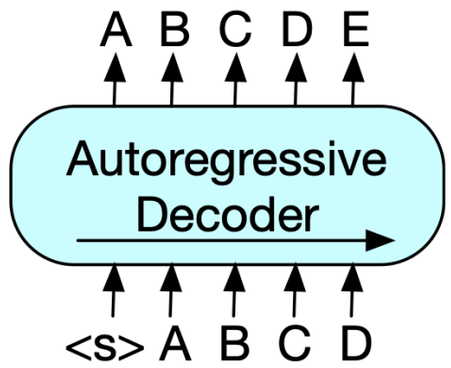
BART uses a standard Tranformer-based NMT architecture

$\Longrightarrow$ can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and many other more recent pretraining schemes

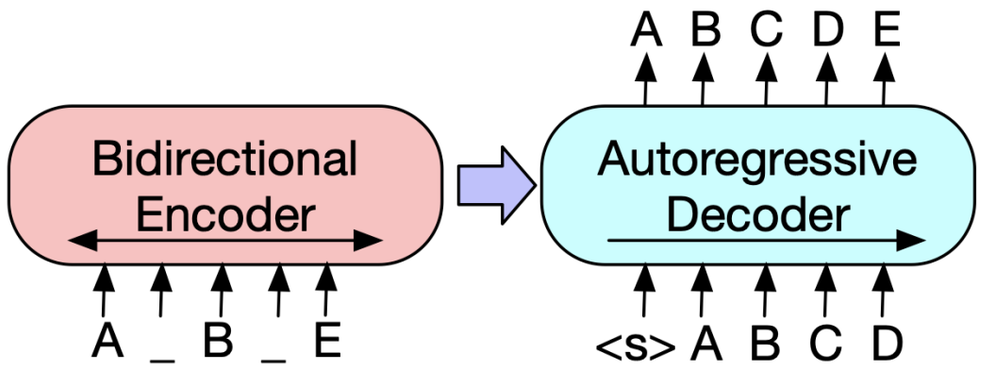
# BART: Denoising Sequence-to-Sequence Pre-training




(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



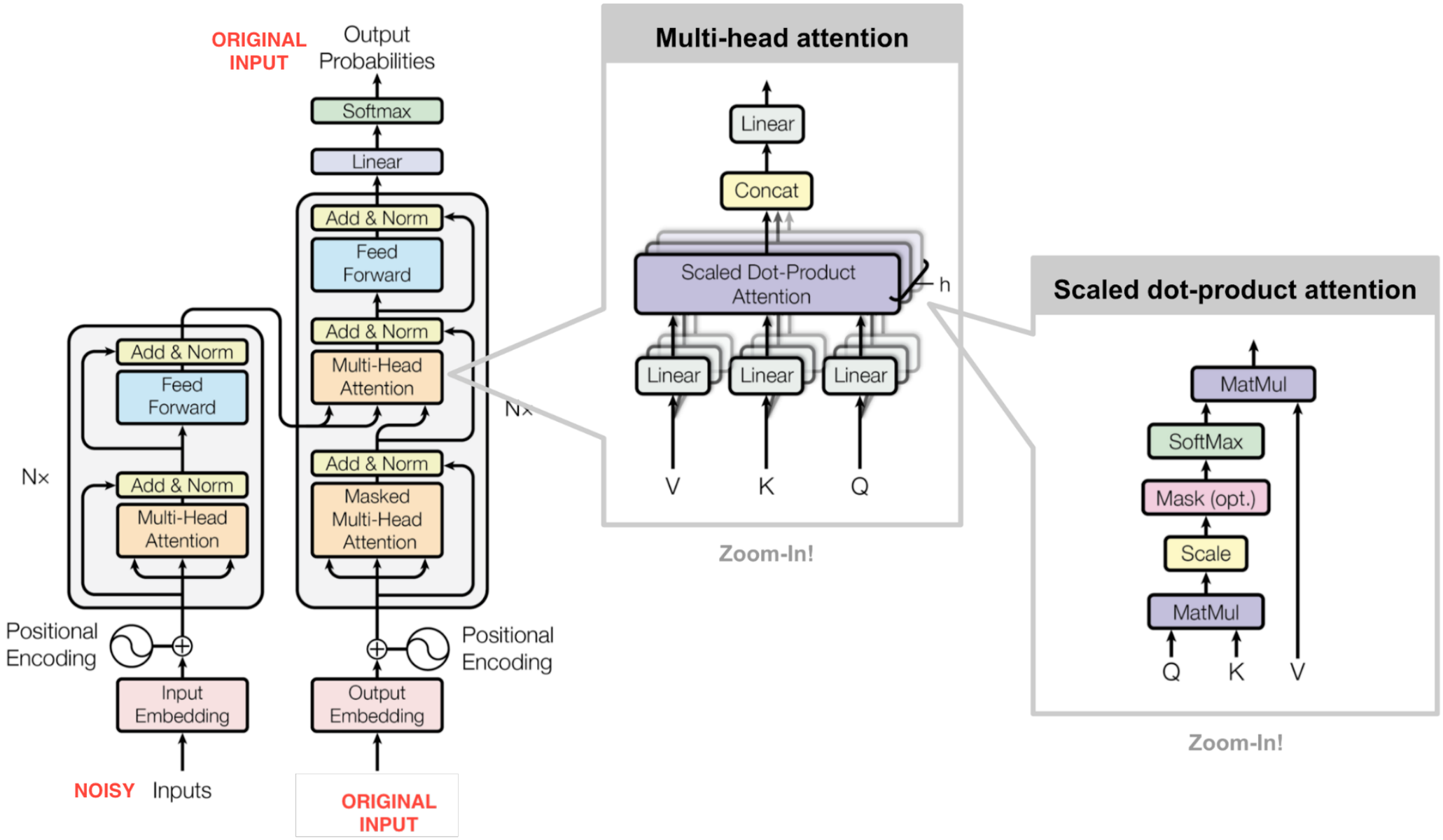

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



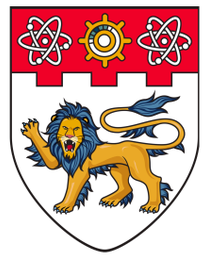

(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

# BART: Denoising Sequence-to-Sequence Pre-training

NTU NLP



# BART: Denoising Sequence-to-Sequence Pre-training



- Noising flexibility: arbitrary transformations can be applied to the original text, including changing its length
- Particularly effective when fine tuned for text generation but also works well for comprehension tasks.

# BART: Denoising Sequence-to-Sequence Pre-training

NTU NLP

- Uses standard seq2seq Transformer architecture
  $\Longrightarrow$ uses GeLU activation instead of ReLU
- Base model: 6 layers in encoder and decoder
  Large model: 12 layers in encoder and decoder

Architectural differences with BERT:

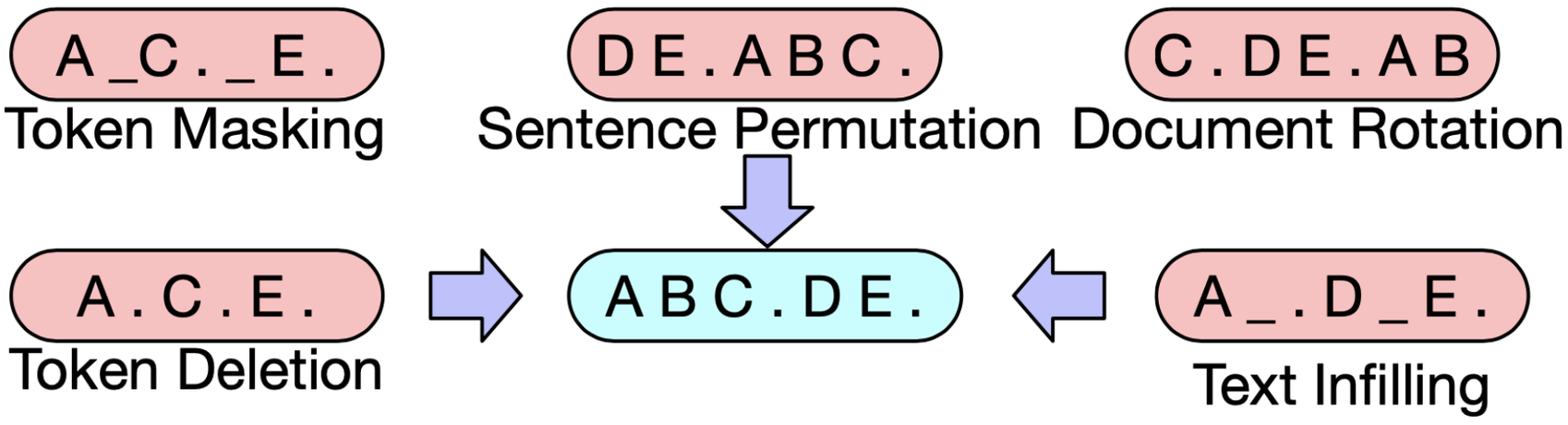
1. each layer of the decoder additionally performs cross-attention over the final hidden layer of the encoder
2. BERT uses an additional feed-forward network before word-prediction, which BART does not
3. BART contains roughly 10% more parameters than the equivalently sized BERT model

# BART: Pre-training

BART is trained by corrupting documents and then optimizing a reconstruction loss

**Reconstruction Loss**: cross-entropy between the decoder's output and the original document.

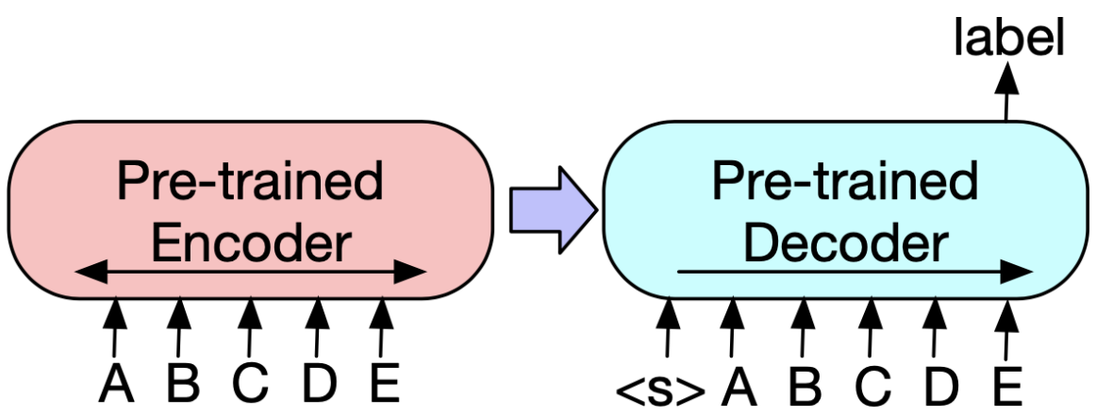
BART allows to apply any type of document corruption



# BART: Fine-training

**Sequence Classification Tasks**

- Same input is fed into the encoder and decoder
- Final hidden state of the final decoder token is fed into new multi-class linear classifier.

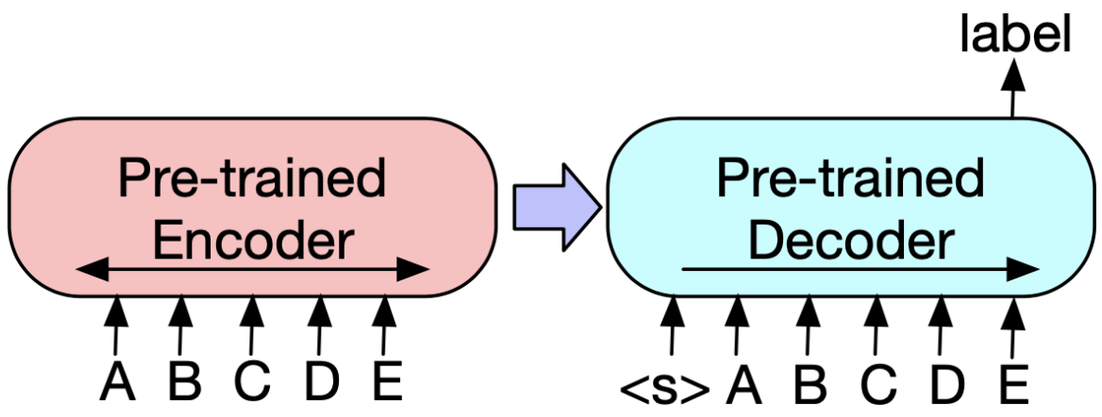



(a) To use BART for classification problems, the same input is fed into the encoder and decoder, and the representation from the final output is used.

# BART: Fine-training

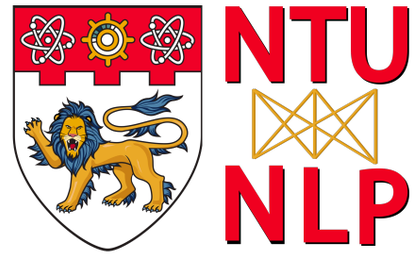
**Token Classification Tasks**

- Input processing is similar to sequence classification tasks
- Use the top hidden state of the decoder as a representation for each word which is used to classify the token.




(a) To use BART for classification problems, the same input is fed into the encoder and decoder, and the representation from the final output is used.

# BART: Fine-training




**Sequence Generation Tasks**

- Can be directly fine tuned for sequence generation tasks because of autoregressive decoder
- Input is manipulated $\Longrightarrow$ related to the denoising pre-training objective.
- Encoder input is the input sequence, and the decoder generates outputs autoregressively

# BART: Results

| Model | SQuAD 1.1 F1 | MNLI Acc | ELI5 PPL | XSum PPL | ConvAI2 PPL | CNN/DM PPL |
|---|---|---|---|---|---|---|
| BERT Base (Devlin et al., 2019) | 88.5 | **84.3** | - | - | - | - |
| Masked Language Model | 90.0 | 83.5 | 24.77 | 7.87 | 12.59 | 7.06 |
| Masked Seq2seq | 87.0 | 82.1 | 23.40 | 6.80 | 11.43 | 6.19 |
| Language Model | 76.7 | 80.1 | **21.40** | 7.00 | 11.51 | 6.56 |
| Permuted Language Model | 89.1 | 83.7 | 24.03 | 7.69 | 12.23 | 6.96 |
| Multitask Masked Language Model | 89.2 | 82.4 | 23.73 | 7.50 | 12.39 | 6.74 |
| BART Base | | | | | | |
| w/ Token Masking | 90.4 | 84.1 | 25.05 | 7.08 | 11.73 | 6.10 |
| w/ Token Deletion | 90.4 | 84.1 | 24.61 | 6.90 | 11.46 | 5.87 |
| w/ Text Infilling | **90.8** | 84.0 | 24.26 | **6.61** | **11.05** | 5.83 |
| w/ Document Rotation | 77.2 | 75.3 | 53.69 | 17.14 | 19.87 | 10.59 |
| w/ Sentence Shuffling | 85.4 | 81.5 | 41.87 | 10.93 | 16.67 | 7.89 |
| w/ Text Infilling + Sentence Shuffling | **90.8** | 83.8 | 24.17 | 6.62 | 11.12 | **5.41** |

# BART: Results

## Discriminative Tasks

| | SQuAD 1.1 EM/F1 | SQuAD 2.0 EM/F1 | MNLI m/mm | SST Acc | QQP Acc | QNLI Acc | STS-B Acc | RTE Acc | MRPC Acc | CoLA Mcc |
|---|---|---|---|---|---|---|---|---|---|---|
| BERT | 84.1/90.9 | 79.0/81.8 | 86.6/- | 93.2 | 91.3 | 92.3 | 90.0 | 70.4 | 88.0 | 60.6 |
| UniLM | -/- | 80.5/83.4 | 87.0/85.9 | 94.5 | - | 92.7 | - | 70.9 | - | 61.1 |
| XLNet | **89.0**/94.5 | 86.1/88.8 | 89.8/- | 95.6 | 91.8 | 93.9 | 91.8 | 83.8 | 89.2 | 63.6 |
| RoBERTa | 88.9/**94.6** | **86.5/89.4** | **90.2/90.2** | 96.4 | 92.2 | 94.7 | **92.4** | 86.6 | **90.9** | **68.0** |
| BART | 88.8/**94.6** | 86.1/89.2 | 89.9/90.1 | **96.6** | **92.5** | **94.9** | 91.2 | **87.0** | 90.4 | 62.8 |

Table 2: Results for large models on SQuAD and GLUE tasks. BART performs comparably to RoBERTa and XLNet, suggesting that BART's uni-directional decoder layers do not reduce performance on discriminative tasks.

# BART: Results

## Summarization

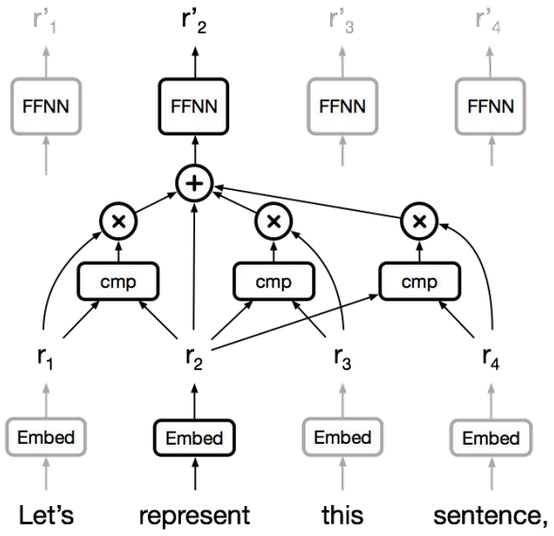
| | CNN/DailyMail | | | XSum | | |
|---|---|---|---|---|---|---|
| | R1 | R2 | RL | R1 | R2 | RL |
| Lead-3 | 40.42 | 17.62 | 36.67 | 16.30 | 1.60 | 11.95 |
| PTGEN (See et al., 2017) | 36.44 | 15.66 | 33.42 | 29.70 | 9.21 | 23.24 |
| PTGEN+COV (See et al., 2017) | 39.53 | 17.28 | 36.38 | 28.10 | 8.02 | 21.72 |
| UniLM | 43.33 | 20.21 | 40.51 | - | - | - |
| BERTSUMABS (Liu & Lapata, 2019) | 41.72 | 19.39 | 38.76 | 38.76 | 16.33 | 31.15 |
| BERTSUMEXTABS (Liu & Lapata, 2019) | 42.13 | 19.60 | 39.18 | 38.81 | 16.50 | 31.27 |
| BART | **44.16** | **21.28** | **40.90** | **45.14** | **22.27** | **37.25** |

# mBART: Results (Translation)

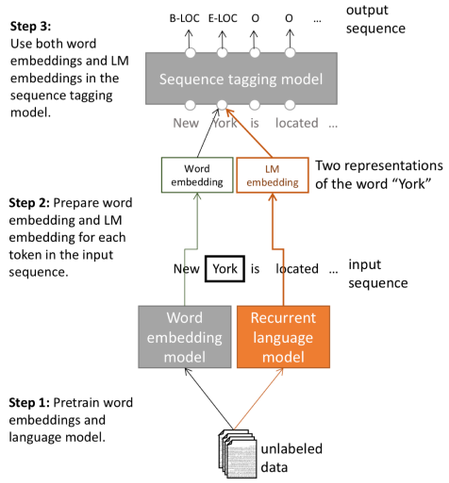
| | Monolingual | Nl-En | En-Nl | Ar-En | En-Ar | Nl-De | De-Nl |
|---|---|---|---|---|---|---|---|
| **Random** | None | 34.6 (-8.7) | 29.3 (-5.5) | 27.5 (-10.1) | 16.9 (-4.7) | 21.3 (-6.4) | 20.9 (-5.2) |
| **mBART02** | En Ro | 41.4 (-2.9) | 34.5 (-0.3) | 34.9 (-2.7) | 21.2 (-0.4) | 26.1 (-1.6) | 25.4 (-0.7) |
| **mBART06** | En Ro Cs It Fr Es | 43.1 (-0.2) | 34.6 (-0.2) | 37.3 (-0.3) | 21.1 (-0.5) | 26.4 (-1.3) | 25.3 (-0.8) |
| **mBART25** | All | **43.3** | **34.8** | **37.6** | **21.6** | **27.7** | **26.1** |

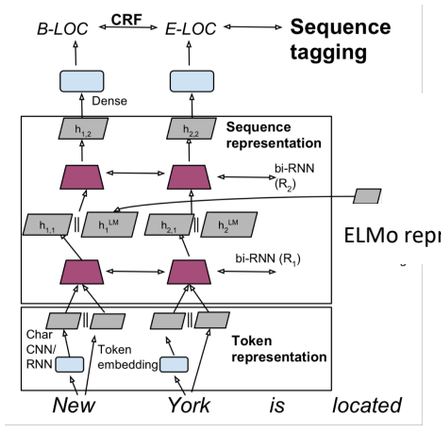
Table 7: **Generalization to Unseen Languages** Language transfer results, fine-tuning on language-pairs without pre-training on them. mBART25 uses all languages during pre-training, while other settings contain at least one unseen language pair. For each model, we also show the gap to mBART25 results.

# Summary




...




ELMo representation

$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}].$$