Transformers

Marco Moresi
Dialog Systems and Machine Learning Group

22.05.2020

Content

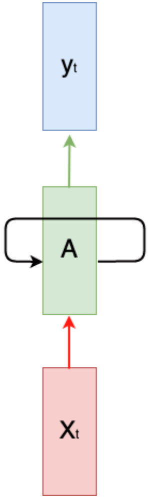
Heinrich Heine
Universität Düsseldorf

- Recurrent Neural Network (RNN)
  - What is a RNN?
  - Long term dependencies
- Transformers
  - Self Attention Mechanism
  - Multi Head Attention
- Transformer-based models
  - BERT
  - XLNet
  - Electra
- Applications in dialogue systems
- Conclusion

Recurrent Neural Network

Recurrent Neural Network (RNN)

- The idea behind RNNs is to make use of sequential information
- Computation takes into account historical information using the recurrence
- Weights are shared across time.
- Different architectures

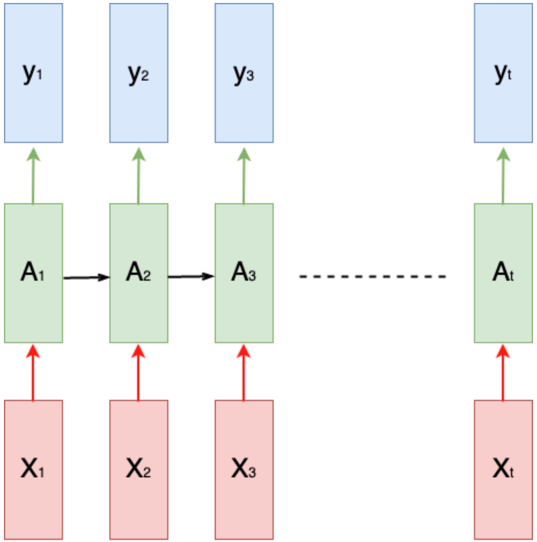


Recurrent neural network diagram

Recurrent Neural Network

Recurrent Neural Network (RNN)

- The idea behind RNNs is to make use of sequential information
- Computation takes into account historical information using the recurrence
- Weights are shared across time.
- Different architectures

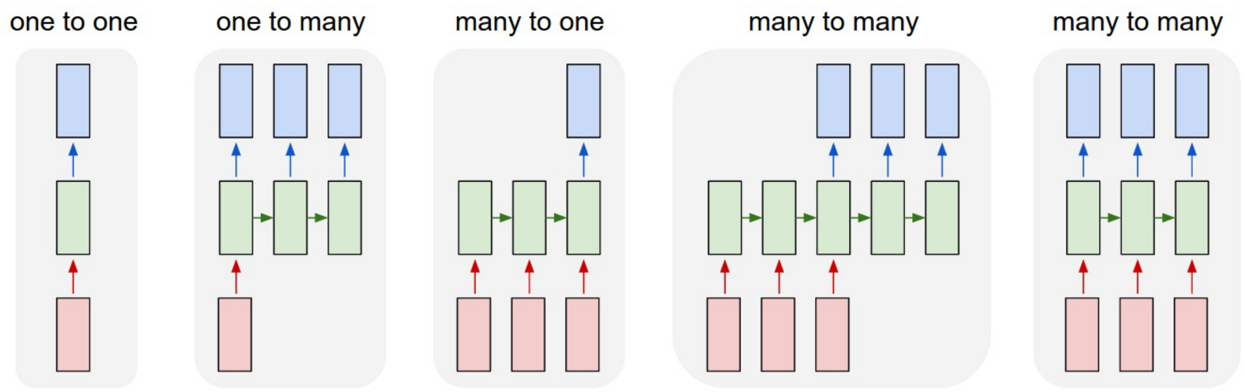


Unfolded recurrent neural network diagram

Recurrent Neural Network

Recurrent Neural

- The idea behind
of sequential info
- Computation tak
historical informa
recurrence
- Weights are sha
- Different architectures



Recurrent Neural Network

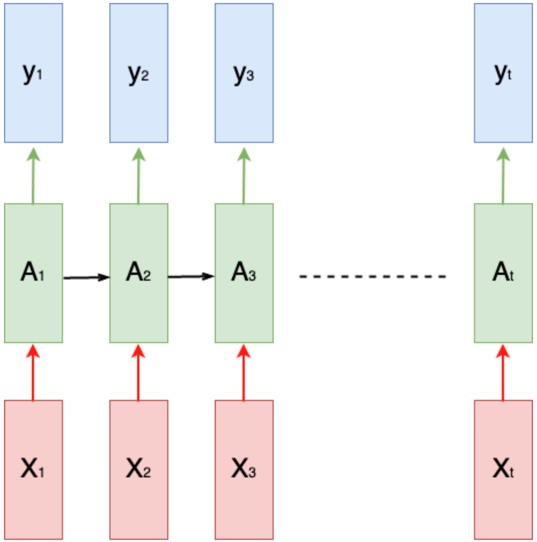
Recurrent Neural Network

- Loss Function

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}\left(\hat{y}^{<t>}, y^{<t>}\right)$$

- Backpropagation

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \left.\frac{\partial \mathcal{L}^{(T)}}{\partial W}\right|_{(t)}$$



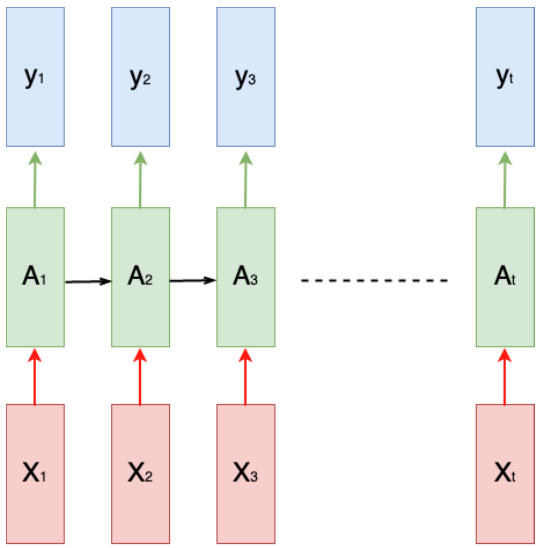
Unfolded recurrent neural network diagram

Recurrent Neural Network

Heinrich Heine
Universität Düsseldorf

Vanishing gradient problem

- The gradient value becomes small during backpropagation so does not affect the values at the beginning of the network.
- Many local influences. A value is mainly influenced by inputs that are somewhere close.
- Lack of long term dependencies



Unfolded recurrent neural network diagram

The cat, which ate a lot of food …., was full.
The cats, which ate a lot of food …., were full.

Transformers

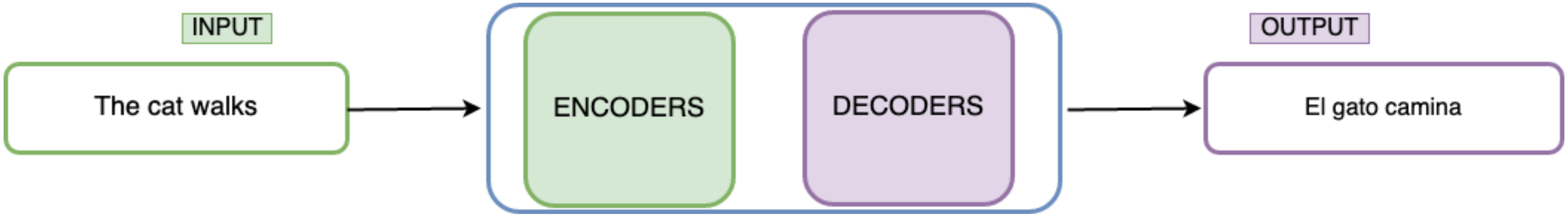
Heinrich Heine
Universität Düsseldorf

Architecture

[Vaswani, et al. Attention is all you need, 2017]

Transformers

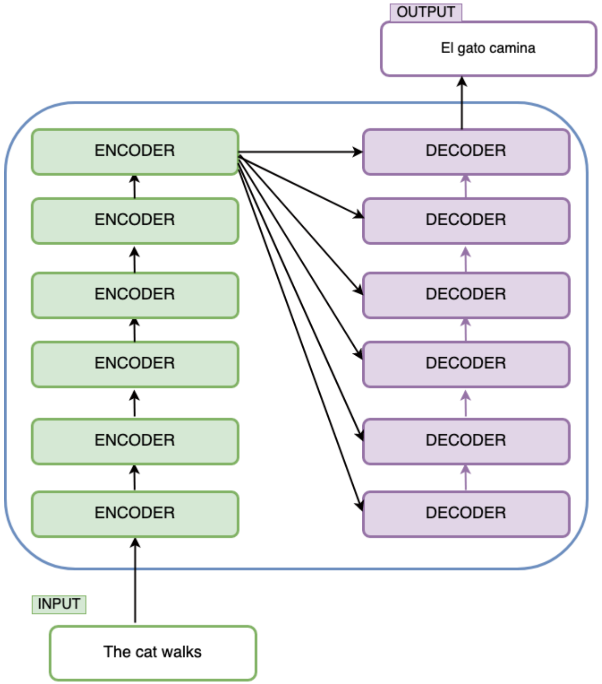
Architecture



[Vaswani, et al. Attention is all you need, 2017]

Transformers

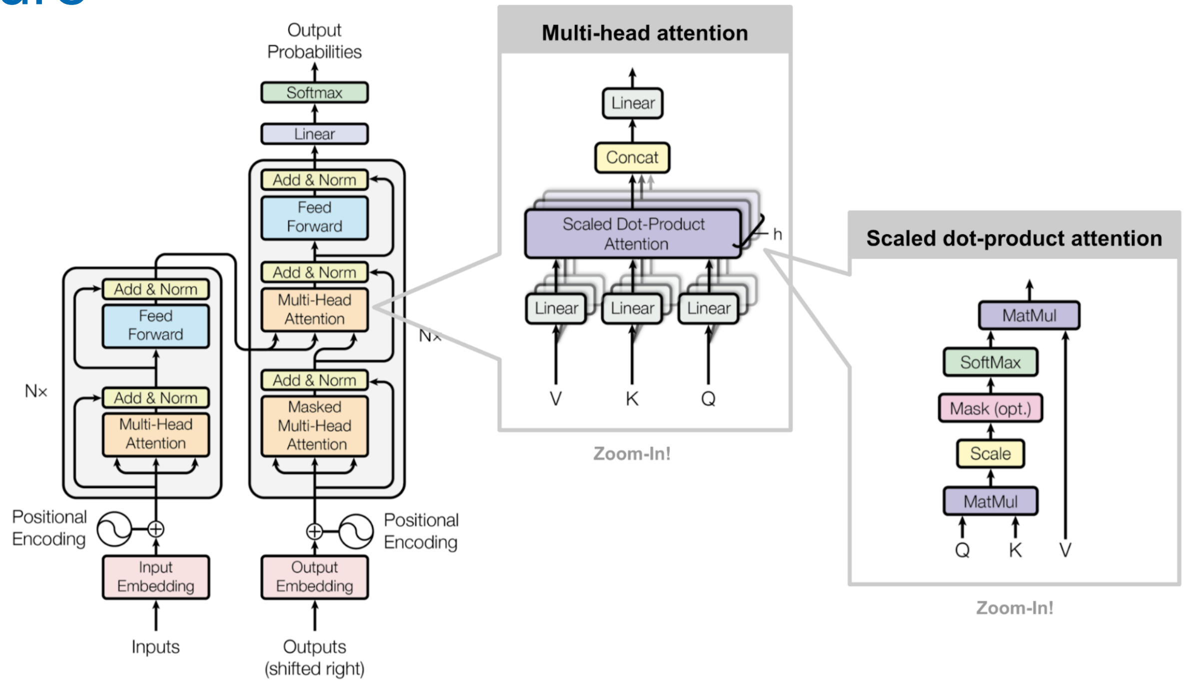
Architecture



[Vaswani, et al. Attention is all you need, 2017]

Transformer

Architecture



The full model architecture of the transformer. (Image source: Fig 1 & 2 in "Attention is all you need" Vaswani, et al., 2017)

Transformer

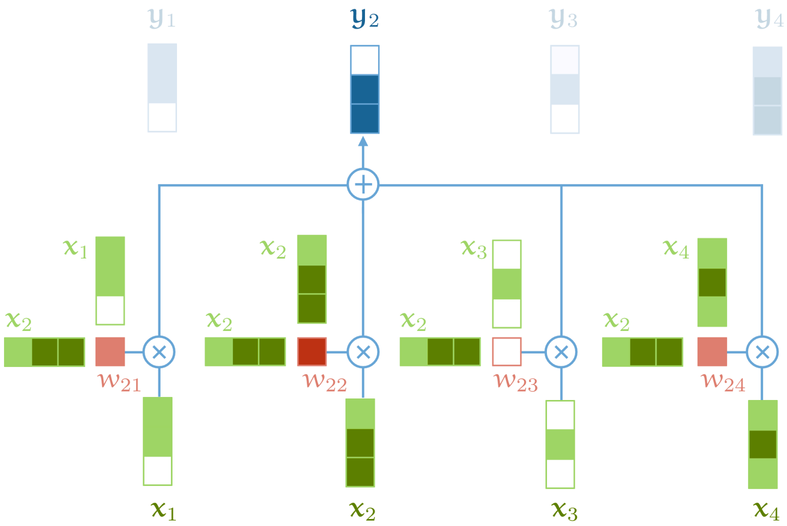
Self-Attention

- Sequence to Sequence Operation
- input vectors $\mathbf{x}_1,\mathbf{x}_2,\ldots\mathbf{x}_t$
- output vectors $\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_t$

$$\mathbf{y}_i = \sum_j w_{ij}\boldsymbol{x}_j$$

$$w'_{ij} = \boldsymbol{x}_i^\top \boldsymbol{x}_j$$

$$w_{ij} = \frac{\exp w'_{ij}}{\sum_j \exp w'_{ij}}$$



Transformer

Self-Attention in action

- We have the following sequence as input:
  - ” the, cat, walks, on, the, street ”
- Assign each word **t** in our sequence its corresponding embedding
  - V_{the} , V_{cat} ,V_{walks}, V_{on} ,V_{the} ,V_{street}
- Feed this sequence into a self-attention layer and the output looks as follow:
  - Y_{the} , Y_{cat} ,Y_{walks}, Y_{on} ,Y_{the} ,Y_{street}

  Where Y_{cat} is a weighted sum over all embeddings vectors in the first sequence, weighted by their dot-product with V_{cat}

Heinrich Heine Universität Düsseldorf

Queries, Keys and Values

- Every input vector $\mathbf{x}_i$ is used in three different ways in the self attention operation:
  - Query: It is compared to every other vector to establish the weights for its own output $\mathbf{y}_i$
  - Key: It is compared to every other vector to establish the weights for the output of the j-th vector $\mathbf{y}_j$
  - Value: It is used as part of the weighted sum to compute each output vector once the weights have been established.

Transformer

Queries, Keys and Values

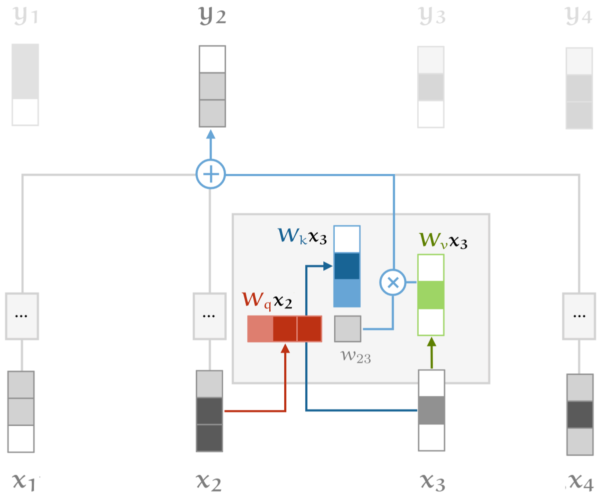
- In order to calculate the query, keys and values vectors we incorporate W_q W_k W_v matrices

$$\mathbf{q}_i = \mathbf{W}_q \mathbf{x}_i \quad \mathbf{k}_i = \mathbf{W}_k \mathbf{x}_i \quad v_i = \mathbf{W}_v \mathbf{x}_i$$

$$w'_{ij} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{k}}$$

$$w_{ij} = \text{softmax}(w'_{ij})$$

$$\mathbf{y}_i = \sum_j w_{ij} \boldsymbol{v}_j$$



Self-attention with query, key and values transformations

Multi-Head Attention

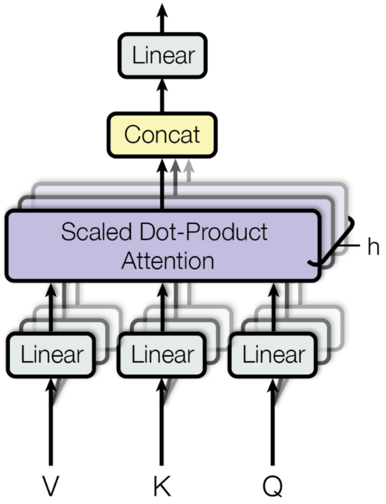
- Given the sequence:
  - “Juan gave roses to Susan”

- Words like ”gave” has different relations to different parts of the sentence.
- In a single Self-Attention operation all the information just gets summed together.
  - If “Susan gave roses to Juan” instead, the output vector Y_{gave} would be the same even though the meaning has changed.

- We can give the self attention greater power of discrimination, by combining several self attention mechanisms

Multi-Head Attention

- Multi-head attention allows the model to jointly attend to information from different representation subspaces at different position
- Each head *i* has its own matrices W_V^i, W_K^i, W_Q^i to do the projections into different subspaces
- Scaled Dot-Product Attention are calculated in parallel
- All the outputs are concatenated.
- Finally the concatenated output is projected with a weight matrix W^O that was trained jointly with the model.



Multi-Head Attention

Transformer

Multi-Head Attention (recap)

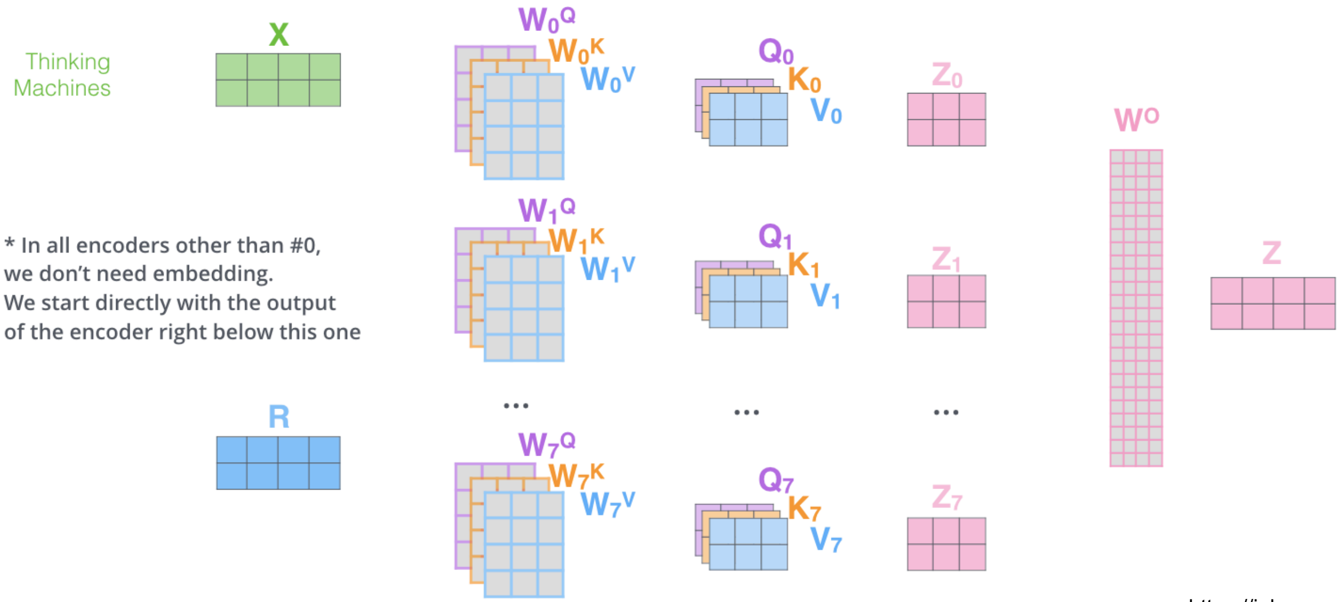
1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



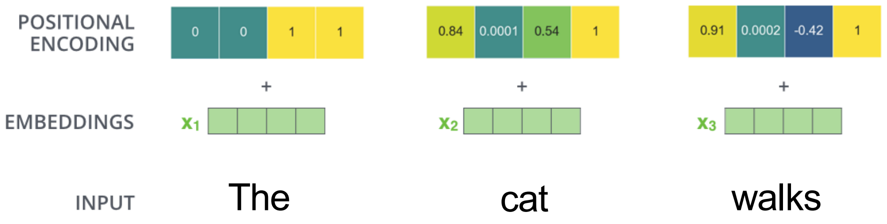
https://jalammar.github.io/illustrated-transformer/

Positional Encoding

- As attention system does not take into account the order of the sequence as RNN does, is necessary to add extra information.

- Positional Encoding

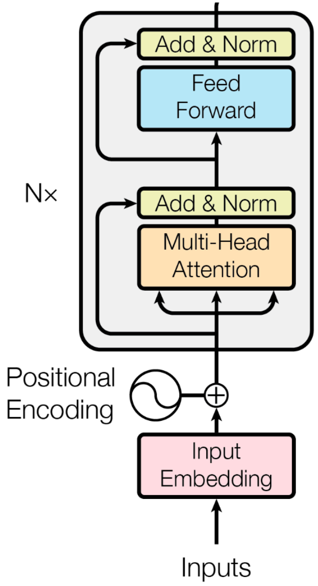
$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right)$$



Heinrich Heine
Universität Düsseldorf

The residuals

- Each sub-layer in each encoder has a residual connection around it, and is followed by a layer-normalization step.
- Normalization and residual connections are standard tricks used to help deep neural networks train faster and more accurately.
- The layer normalization is applied over the embedding dimension only.



Transformer decoder block

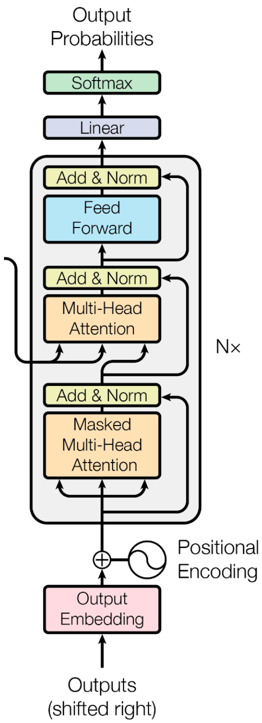
Transformer

Decoder side

- The self-attention layer is only allowed to attend to earlier positions in the output sequence. This is done by masking future positions (setting them to -inf) before the softmax step in the self-attention calculation.

- The Linear layer is a simple fully connected neural network that projects the vector produced by the stack of decoders, into a much, much larger vector called a logits vector.

- The softmax layer then turns scores of logits vector into probabilities.



BERT decoder

Transformer

Results

- Architecture Details:
  - Transformer base model
    - Stack of 6 encoder-decoder
    - 8 Attention heads
    - 512 hidden dimensions
  - Transformer big
    - Stack of 6 encoder-decoder
    - 16 Attention heads
    - 1024 hidden dimensions

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

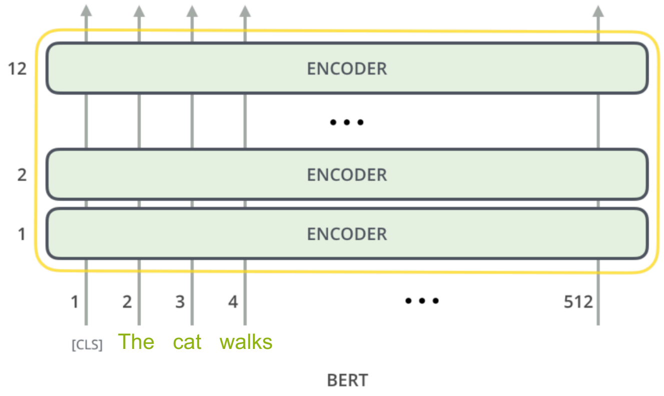
BLEU scores on the English-to-German and English-to-French newstest 2014

Transformer

Heinrich Heine
Universität Düsseldorf

BERT (Bidirectional Encoder Representation from Transformers)

- Designed to pretrain bidirectional representation from unlabeled text
- BERT-base: 12 Transformer encoder blocks, hidden dimension 768 and 12 attention heads.
- Fixed input size (512 tokens)
- Word embedding + Positional encoding +Sentence embedding
- Special tokens ([CLS], [SEP], sentence A/B embedding)
- Pre-training tasks:
  - Masked LM
  - Next Sentence Prediction



[Devlin, et al.. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.]

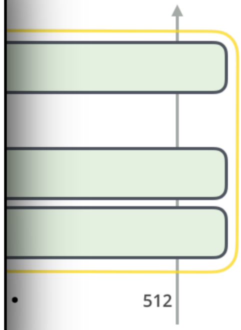
Heinrich Heine Universität Düsseldorf

BERT (Bidirectional Encoder Representation from Transformers)

- Designed to pretrai
  representation fron
- BERT-base: 12 Tra
  blocks, hidden dim
  attention heads.
- Fixed input size (51
- Word embedding
  +Sentence embe
- Special tokens (
  A/B embedding
- Pre-training tasks:
  - Masked LM
  - Next Sentence Prediction

Masked Language Model:

- Mask some percentage of the input tokens at random, and then predict those masked tokens.
- Last hidden vector corresponding to the masked tokens are fed into an output softmax layer over the vocabulary



[Devlin, et al.. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.]

Transformer

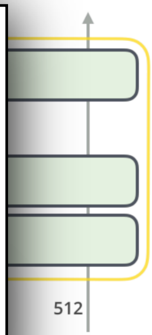
Heinrich Heine
Universität Düsseldorf

BERT (Bidirectional Encoder Representation from Transformers)

- Designed to pretrain bidirectional representation from unla
- BERT-base: 12 Transfor blocks, hidden dimensio attention heads.
- Fixed input size (512 tok
- Word embedding + Posi +Sentence embedding
- Special tokens ([CLS], A/B embedding)
- Pre-training tasks:
  - Masked LM
  - Next Sentence Prediction

Next Sentence Prediction:

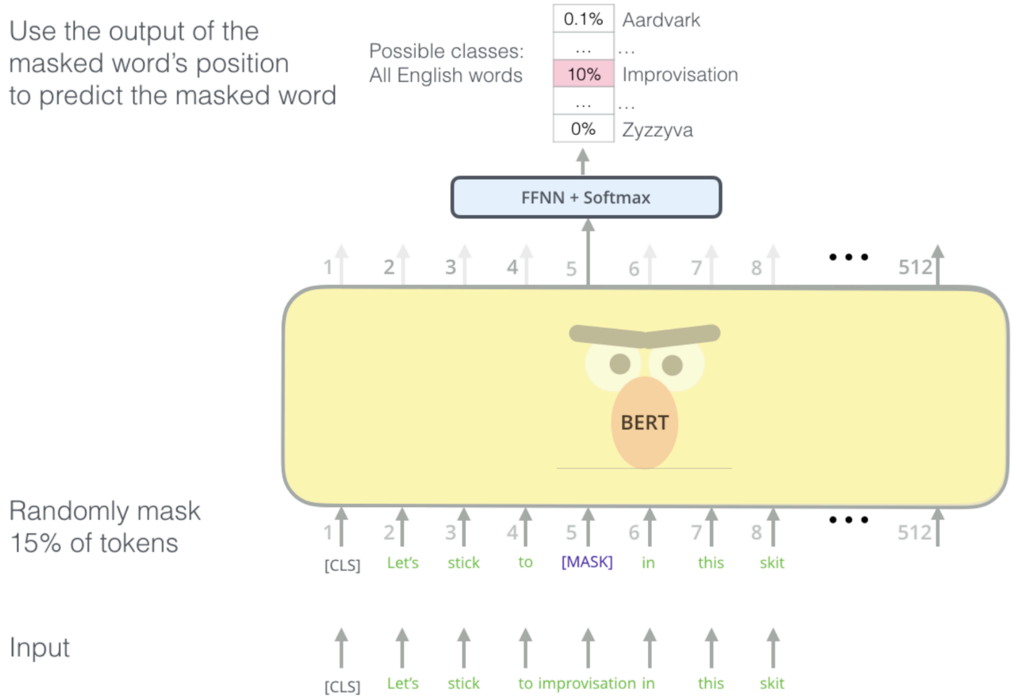
- Given a pair of two sentences (**A, B**)
- Learns to predict if the second sentence in the pair is the subsequent sentence in the original document



[Devlin, et al.. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.]

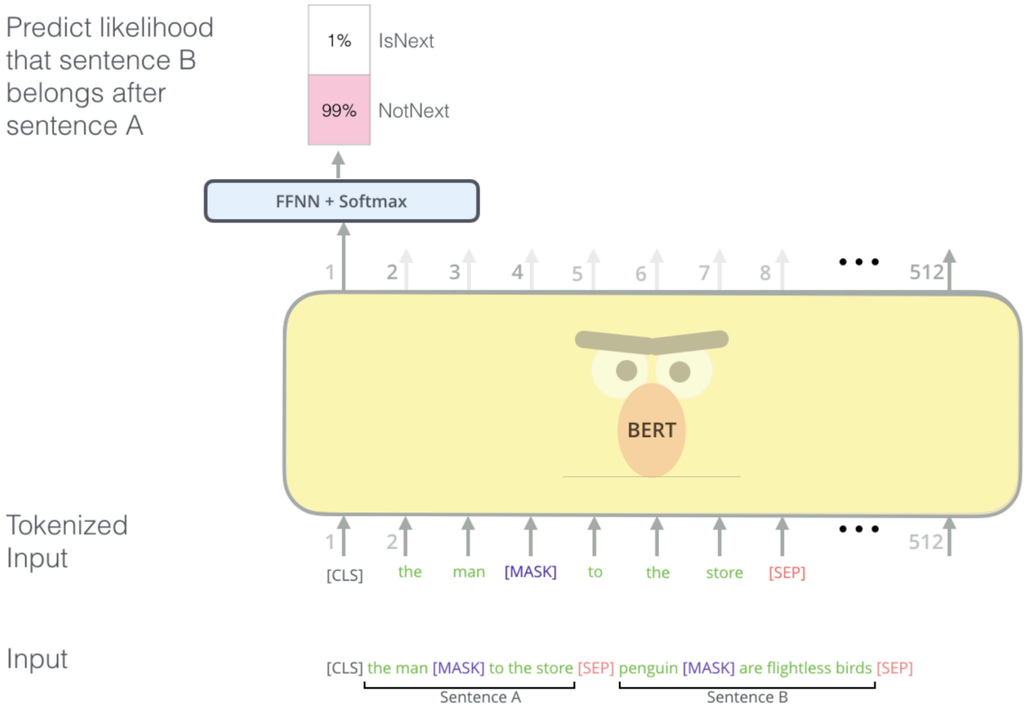
Transformer

Masked Language Model

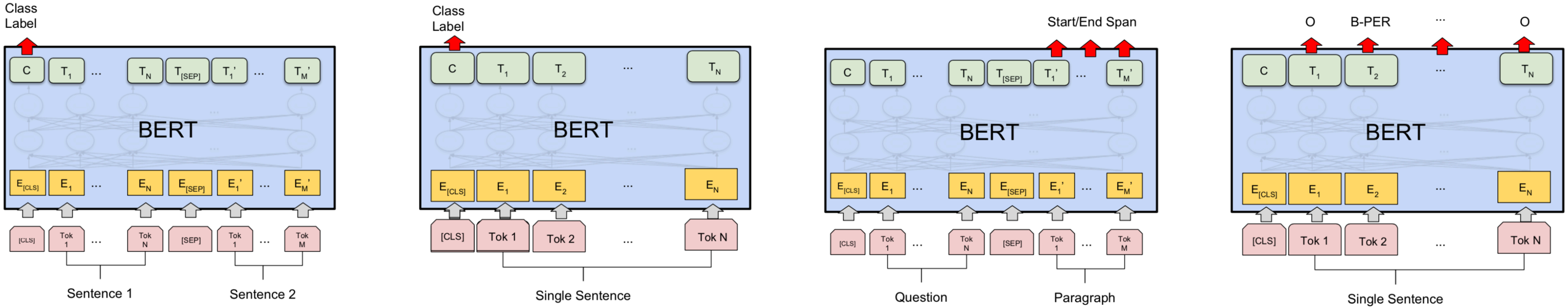


Transformer

Next Sentence Prediction



Fine Tuning



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

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

(c) Question Answering Tasks: SQuAD v1.1

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

Transformer

Benchmarks

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Average
Pre-OpenAI SOTA	80.6	82.3	66.1	61.7	93.2	86.0	35.0	81.0	74.0
BiLSTM+ELMo+Attn	76.4	79.8	64.8	56.8	90.4	84.9	36.0	73.3	71.0
OpenAI GPT	82.1	87.4	70.3	56.0	91.3	82.3	45.4	80.0	75.1
BERT Base	84.6	90.5	71.2	66.4	93.5	88.9	52.1	85.8	79.6
BERT Large	**86.7**	**92.7**	**72.1**	**70.1**	**94.9**	**89.3**	**60.5**	**86.5**	**82.1**

GLUE Test results

SQuAD1.1	EM	F1
BiDAF+ELMo	-	85.6
R.M. Reader	81.2	82.3
BERT Large	**84.1**	**90.9**

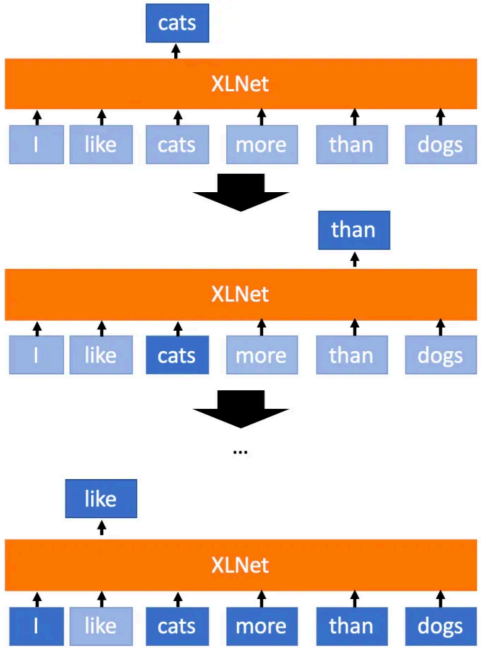
SQuAD 1.1 results

SQuAD2.0	EM	F1
unet	71.4	74.9
SLQA	71.4	74.4
BERT Large	**80.0**	**83.1**

SQuAD 2.0 results

XLNet (Generalized Autoregressive Pretraining for Language Understanding)

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurrence mechanism and relative encoding.
- Mitigate the problem of Masked Language Model



XLNet - Permutation Language Modeling

[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

Transformer

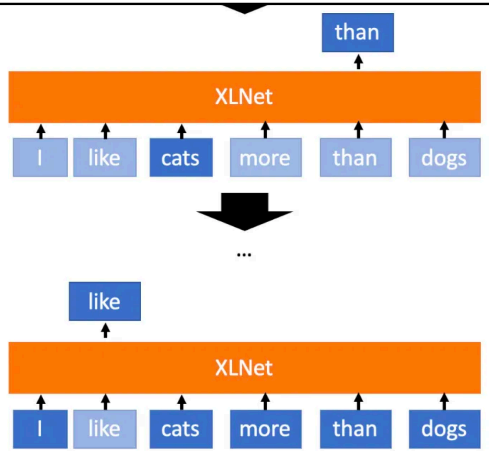
XLNet (Generalized Autoregres

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurrence mechanism and relative encoding.
- Mitigate the problem of Masked Language Model

Language Model:

$$P(W) = \prod_t P(w_t | w_0, w_1, \ldots w_{t-1})$$

$$W^* = \underset{W}{\operatorname{argmax}}\, P(W)$$



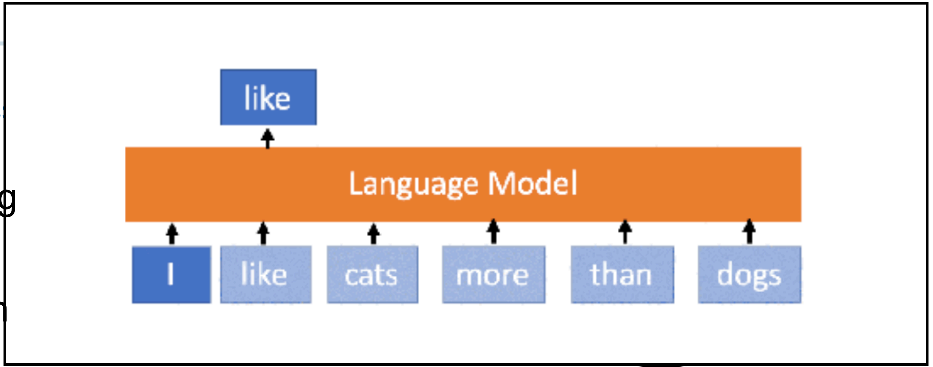
XLNet - Permutation Language Modeling

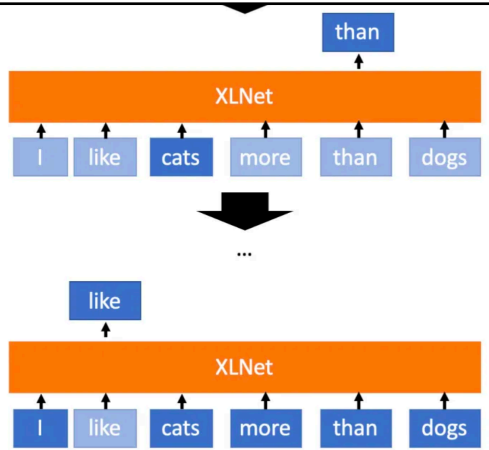
[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

Transformer

XLNet (Generalized Autoregres

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurrence mechanism and relative encoding.
- Mitigate the problem of Masked Language Model



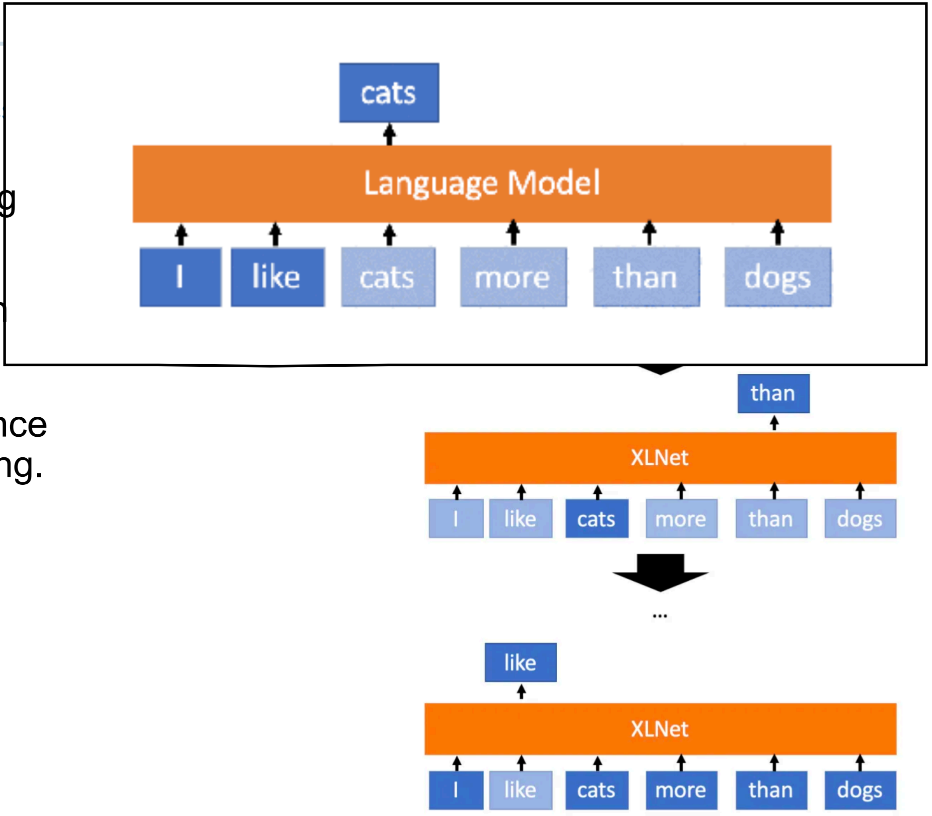


XLNet - Permutation Language Modeling

[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

XLNet (Generalized Autoregres

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurrence mechanism and relative encoding.
- Mitigate the problem of Masked Language Model



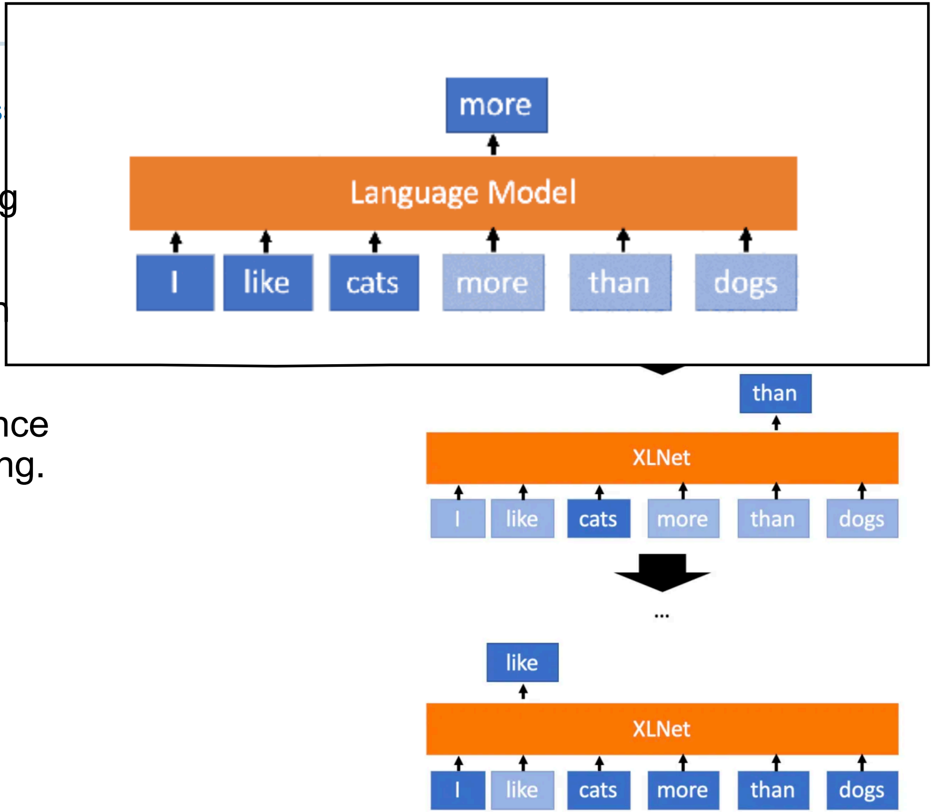
XLNet - Permutation Language Modeling

[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

Transformer

XLNet (Generalized Autoregres

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurrence mechanism and relative encoding.
- Mitigate the problem of Masked Language Model



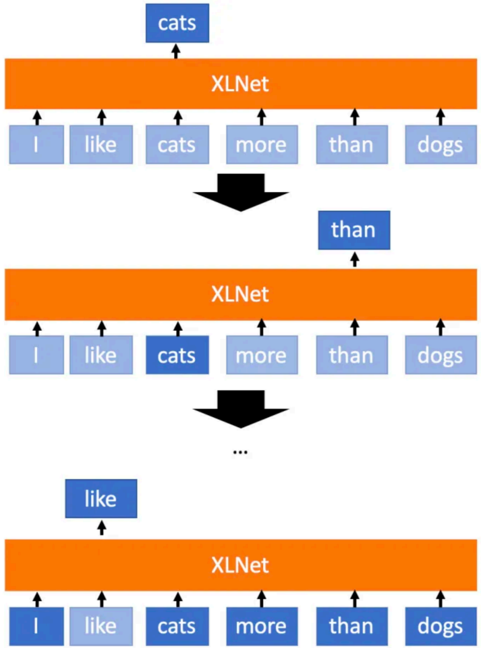
XLNet - Permutation Language Modeling

[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

Heinrich Heine Universität Düsseldorf

XLNet (Generalized Autoregressive Pretraining for Language Understanding)

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurrence mechanism and relative encoding.
- Mitigate the problem of Masked Language Model



XLNet - Permutation Language Modeling

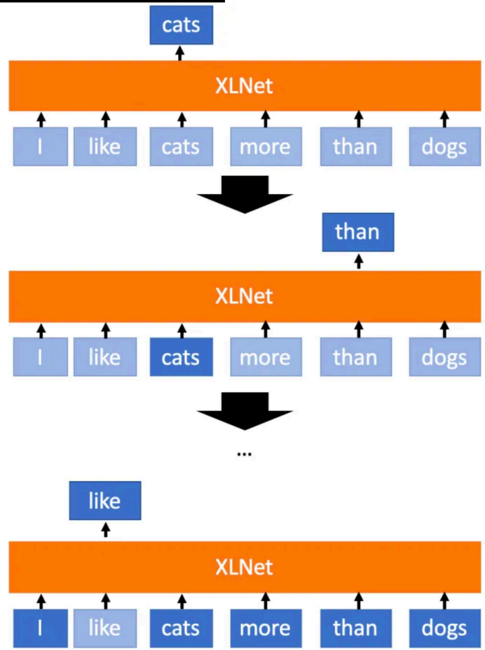
[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

Transformer

XLNet (Generalized Autoregress)

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

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurrence mechanism and relative encoding.
- Mitigate the problem of Masked Language Model



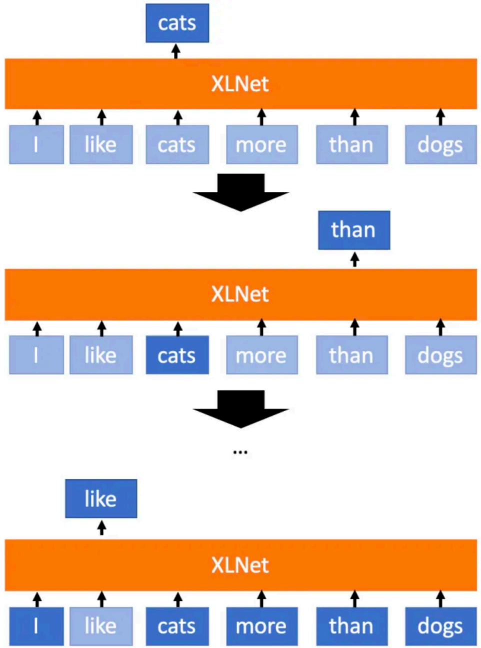
XLNet - Permutation Language Modeling

[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

Transformer

XLNet (Generalized Autoregressive Pretraining for Language Understanding)

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurrence mechanism and relative encoding.
- Mitigate the problem of Masked Language Model



XLNet - Permutation Language Modeling

[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

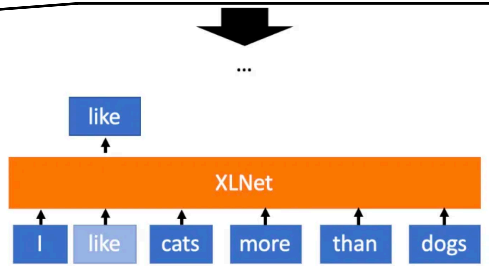
Heinrich Heine Universität Düsseldorf

XLNet (Generalized Autoregressive Pretraining for Language Understanding)

- Permutation Language Modeling
- Does not rely on data corruption
- Integrates the segment recurren mechanism and relative encodi
- Mitigate the problem of Masked Language Model

Example:
New York is a city
Target: Predict tokens New and York

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New} \mid \text{ is a city }) + \log p(\text{ York } \mid \text{is a city })$$
$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{ is a city }) + \log p(\text{ York } \mid \text{New, is a city })$$



XLNet - Permutation Language Modeling

[Yang, et al. Xlnet: Generalized autoregressive pretraining for languageunderstanding, 2019]

Transformer

Benchmarks

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	AVERAGE
BERT-Large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90	84.1
XLNet-Base	86.8	91.7	91.4	74.0	94.7	88.2	60.2	89.5	84.6
XLNet-Large	**89.8**	**93.9**	**91.8**	**83.8**	**95.6**	**89.2**	**63.6**	**91.8**	**87.5**

GLUE Test results

SQuAD1.1	EM	F1
BERT-Large	84.1	90.9
XLNet-Large	**89.7**	**95.1**

SQuAD 1.1 results

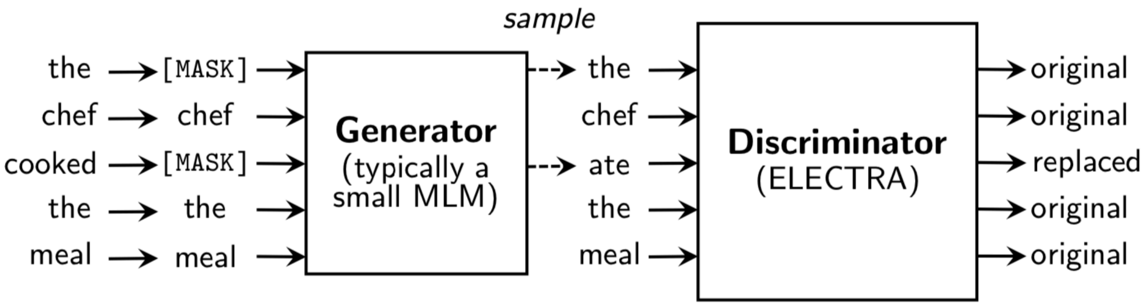
SQuAD2.0	EM	F1
BERT-Large	80.0	83.1
XLNet-Large	**87.9**	**90.6**

SQuAD 2.0 results

Transformer

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately)

- Mitigate the problem of Masked Language Model
- Replaced token detection as pre-training task
- More efficient pre training task
- Bi-directional representation
- Less computation consumption



An overview of replaced token detection. After pre-training, we throw out the generator and only fine-tune the discriminator (the ELECTRA model) on downstream tasks.

[Clark, et al. Electra: Pre-training text encoders as discriminators rather than generators, 2020]

Transformer

Benchmarks

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	AVERAGE
BERT	86.7	92.7	89.3	70.1	94.9	85.4	60.5	86.5	83.3
XLNET	90.9	93.9	90.4	**92.5**	**97.1**	90.5	70.2	90.4	89.5
ELECTRA	**91.3**	**95.8**	**90.8**	**92.5**	**97.1**	**90.7**	**71.7**	**90.8**	**90.1**

GLUE Test results

SQuAD1.1	EM	F1
BERT-Large	84.1	90.9
XLNet-Large	**89.7**	**95.1**
ELECTRA	**89.7**	94.9

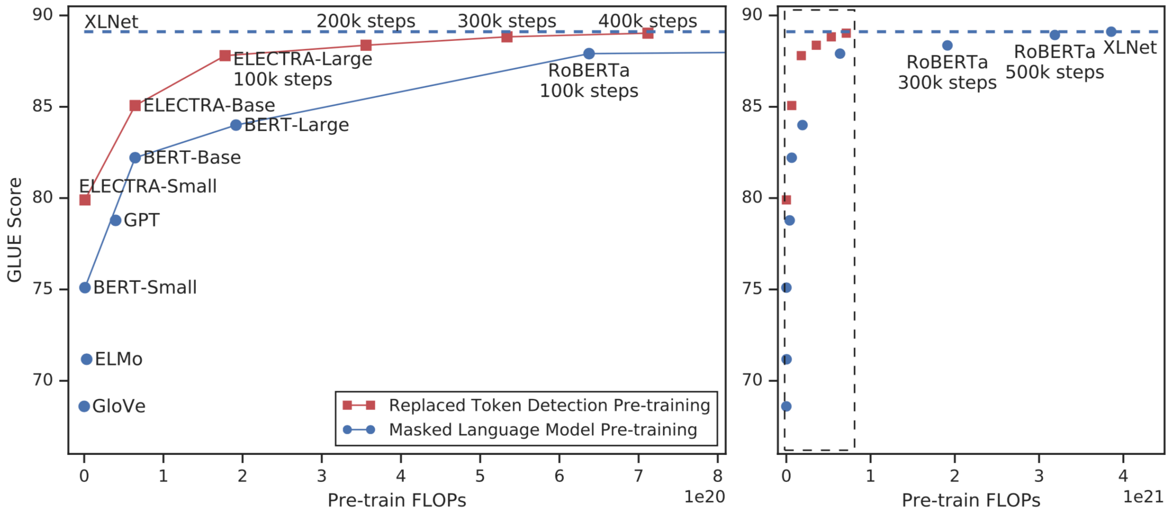
SQuAD 1.1 results

SQuAD2.0	EM	F1
BERT-Large	78.98	81.77
XLNet-Large	87.9	90.6
ELECTRA	**88.7**	**91.4**

SQuAD 2.0 results

Transformer

Benchmarks



Replaced token detection pre-training consistently outperforms masked language model pre-training given the same compute budget. The left figure is a zoomed-in view of the dashed box.

Transformer

Applications in Dialogue Systems

- Dialog State Tracking
  - TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking (Heck et al., 2020)
  - BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer (Chao et al., 2019)
- Natural Language Generation
  - Few-shot Natural Language Generation for Task-Oriented Dialog (Peng et al., 2020)
  - Semantically Conditioned Dialog Response Generation via Hierarchical Disentangled Self-Attention (Chen et al., 2019)
- Evaluation
  - BERTScore: Evaluating Text Generation with BERT (Zhang et al., 2019)
  - USR: An Unsupervised and Reference Free Evaluation Metric for Dialog Generation (Mehri et al., 2020)
- Sentiments in Dialog
  - Hierarchical Transformer Network for Utterance-level Emotion Recognition (Li et al., 2020)
- … and more.

Conclusion

Heinrich Heine
Universität Düsseldorf

- Pros
  - Tackle scarcity data problem
  - Bi-directional representation
  - Easy to fine-tune for a specific task
- Cons
  - Hard to start from the scratch
  - Catastrophic forgetting
  - More expensive to train than RNN

Heinrich Heine
Universität Düsseldorf

Thanks!

References

- TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking (Heck et al., 2020)
- BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer (Chao et al., 2019)
- Few-shot Natural Language Generation for Task-Oriented Dialog (Peng et al., 2020)
- Semantically Conditioned Dialog Response Generation via Hierarchical Disentangled Self-Attention (Chen et al., 2019)
- BERTScore: Evaluating Text Generation with BERT (Zhang et al., 2019)
- USR: An Unsupervised and Reference Free Evaluation Metric for Dialog Generation (Mehri et al., 2020)
- Hierarchical Transformer Network for Utterance-level Emotion Recognition (Li et al., 2020)
- Xlnet: Generalized autoregressive pretraining for language understanding (Yang et al., 2019)
- Attention is all you need, (Vaswani et al., 2017)
- Bert: Pre-training of deep bidirectional transformers for language understanding. (Devlin et al., 2018)
- Glue: A multi-task benchmark and analysis platform for natural language understanding. (Wang et al, 2018)