Transformers in 2021

Brandon Henry
November 11th, 2021
CPSC 677 Advanced NLP

Presentation Outline

Background

Paper: An Attentive Survey of Attention Models

Overview of the Transformer Architecture

Paper: ConViT - Inductive Bias and the Generality of Transformers

Paper: Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer

Paper: DeBERTa: Decoding-enhanced BERT with Disentangled Attention

Paper: Transformers: “The End of History” for Natural Language Processing?

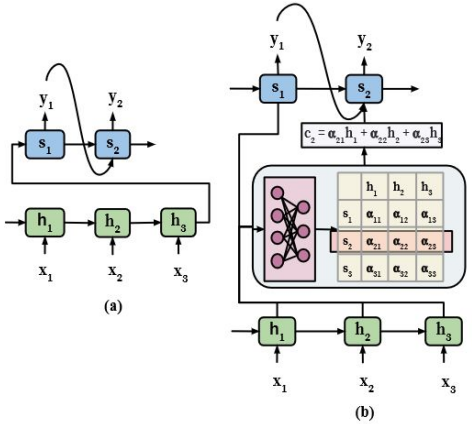
Background

NLP Before Transformers

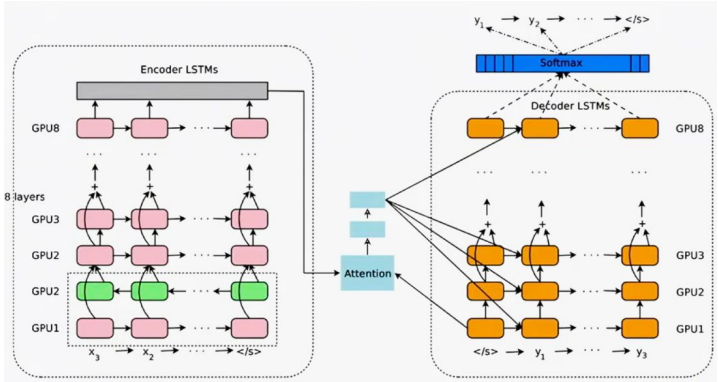
- RNN variants dominated models for sequential data, including most NLP tasks.
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)
- Problems:
  - Recurrent structure does not benefits from modern hardware advancements
  - Training stability in larger, deeper networks
  - Longer term dependencies are not well captured or represented.
  - Together limits the representation capacity that is practical to achieve

Sequence To Sequence

- Encoder - Decoder structure
- Token embeddings fed directly to LSTM layers
- Decoder LSTM layers attend to hidden states of final encoder LSTM layer



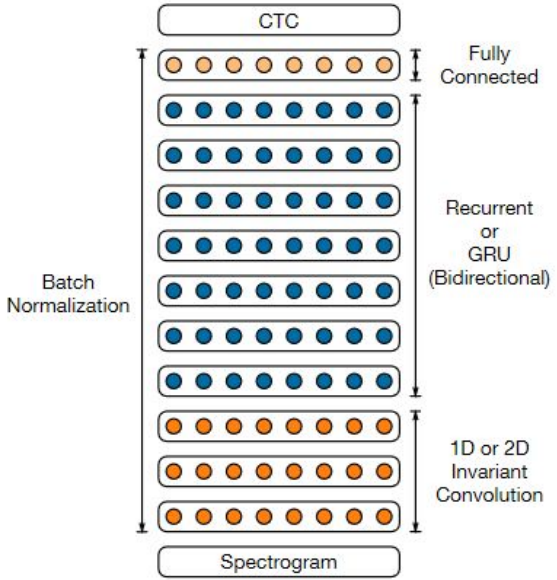
Encoder-decoder architecture: (a) traditional (b) with attention model
An Attentive Survey of Attention Models https://arxiv.org/pdf/1904.02874.pdf



Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation
https://arxiv.org/pdf/1609.08144.pdf

Automatic Speech Recognition (ASR)

- Acoustic features extracted from Mel-frequency spectrogram using convolution
- Relationship between phonemic features developed in LSTM layers.
- Alignment achieved via Connectionist Classification Temporal (CTC) loss.



Deep Speech 2: End-to-End Speech Recognition in English and Mandarin
https://arxiv.org/pdf/1512.02595.pdf

Attention

- Enables a weighting function of values to vary with the data
  - Fully connected and convolution layers are static weighting methods
  - Deep networks can approximate dynamism, but don’t explicitly model it.
- Many different flavors of attention
  - Most developed in the 2014-2017 era.
- Accomplishments:
  - Ameliorates the information bottleneck in traditional Seq2Seq models.
  - Imposes few constraints on the receptive field of the attending node. Context is accessible.
  - Usually amounts to matrix operations: highly scalable on modern hardware, especially GPU/TPU.

Attention Basics

Goal:

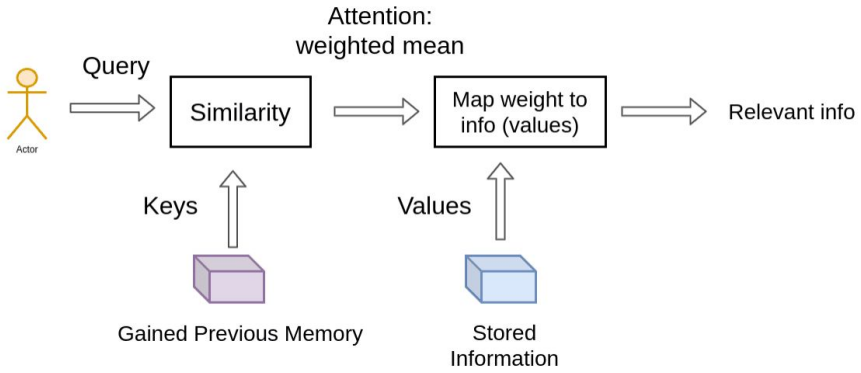
Extract most relevant information to a subject from a larger body of information.

Query: The subject about which relevant information is sought.

Keys: Information/meta-information from which relevance should be determined

Value: The information associated with the key.

How is relevance determined? That depends on the *type* of attention.



Attention as Nadaraya-Watson Kernel Regression

- Restates regression problem as : $\hat{y} = \sum_{i=1}^{n} y_i \alpha(x, x_i)$
- Enables the estimate $\hat{y}$ to be stated as of a function α which only depends on the covariates.
  - Can interpret $\alpha(x, x_i)$ as the relevance of x_i to predict for x; it's a weighting function.
  - In other words, not all data points are created equal as it pertains to predicting y.
- This functional space for which $\hat{y}$ can be represented as a function of the covariates is then contingent on the inner product space α characterizes
- Prespecification of a Gram Matrix is common in kernel SVM to characterize α.
- In NNets, we can relax this prespecification requirement and enable the model to learn the form of α.

Generalizing Attention

- In practice we decompose α into two functions, an alignment function **a** and a distribution function **p**. The composition **p**(**a**(x,x_i)) produces the attention weight for x_i.
- We can relax our interpretation of **a** to further accommodate the attention interpretation:
  - Let K, q, V be key, query, value vectors respectively. We can attend to the keys given a query as:

$$A(q, K, V) = \sum_i p(a(k_i, q)) * v_i$$

- In other words we can model the conditional expectation V|q,K = A(q,K,V) as a weighted average of the values, by attending to values whose weight is determined by the relevance of the query to the corresponding keys.
- The special case where we model elements of our query vector by attending to our query vector itself is known as self attention.
- Coattention is where attention weights are computed for several input sequences *jointly.*
  - Can be useful for Q-A; jointly attend to the question and the source documents.

Attention Mechanisms

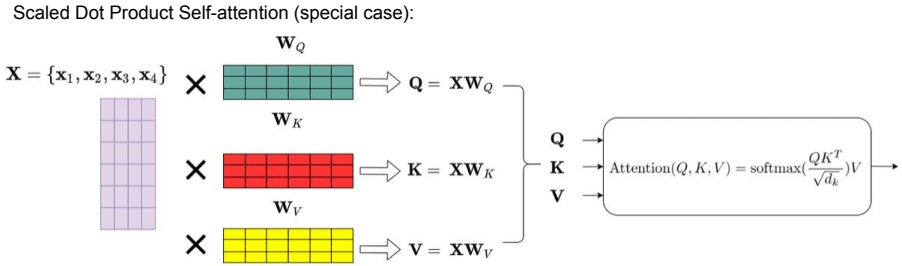
- Different alignment functions usually characterize the meaningful difference between attention mechanisms.
- Scaled dot product very common due to its success in transformers.
- Softmax is the most common distribution function
    - Sparsemax , sparse entmax have been used to yield sparse weights. Useful in Q-A, document summarization.

Function	Equation	References
similarity	$a(k_i, q) = sim(k_i, q)$	[Graves et al. 2014a]
dot product	$a(k_i, q) = q^T k_i$	[Luong et al. 2015a]
scaled dot product	$a(k_i, q) = \frac{q^T k_i}{\sqrt{d_k}}$	[Vaswani et al. 2017]
general	$a(k_i, q) = q^T W k_i$	[Luong et al. 2015a]
biased general	$a(k_i, q) = k_i(Wq + b)$	[Sordoni et al. 2016]
activated general	$a(k_i, q) = act(q^T W k_i + b)$	[Ma et al. 2017b]
generalized kernel	$a(k_i, q) = \phi(q)^T \phi(k_i)$	[Choromanski et al. 2021]
concat	$a(k_i, q) = w_{imp}^T act(W[q; k_i] + b)$	[Luong et al. 2015a]
additive	$a(k_i, q) = w_{imp}^T act(W_1 q + W_2 k_i + b)$	[Bahdanau et al. 2015]
deep	$a(k_i, q) = w_{imp}^T E^{(L-1)} + b^L$ $E^{(l)} = act(W_l E^{(l-1)} + b^l$ $E^{(1)} = act(W_1 k_i + W_0 q) + b^l$	[Pavlopoulos et al. 2017]
location-based	$a(k_i, q) = a(q)$	[Luong et al. 2015a]
feature-based	$a(k_i, q) = w_{imp}^T act(W_1 \phi_1(K) + W_2 \phi_2(K) + b)$	[Li et al. 2019a]

An Attentive Survey of Attention Models
https://arxiv.org/pdf/1904.02874.pdf

(Self) Attention in Practice (for Transformer)

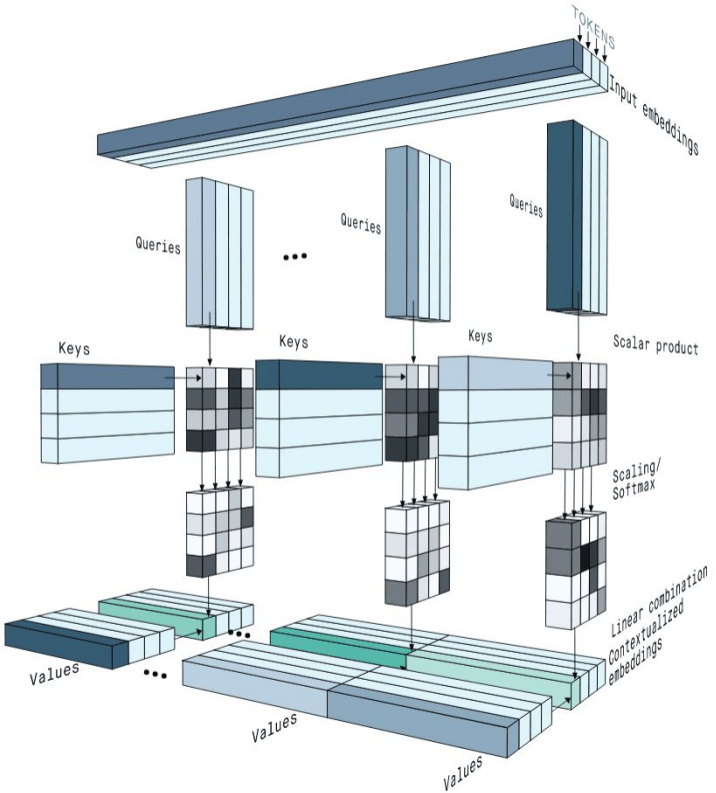
- Transformers use scaled dot product attention (mostly).
- Query, Key, and Value matrices are *learned* by projecting the inputs onto a unique weight matrix corresponding to each.
- This mechanism enables the model to learn the alignment function.



https://theaisummer.com/transformer/

Multi-Head Attention

- Apply a attention separately multiple times.
- Different attention heads could extract different features in a manner not well captured by a single head.
- Concatenate the results of each attention head.
- Must project concatenated result to original dimensions in case of self attention.



https://peltarion.com/blog/data-science/self-attention-video

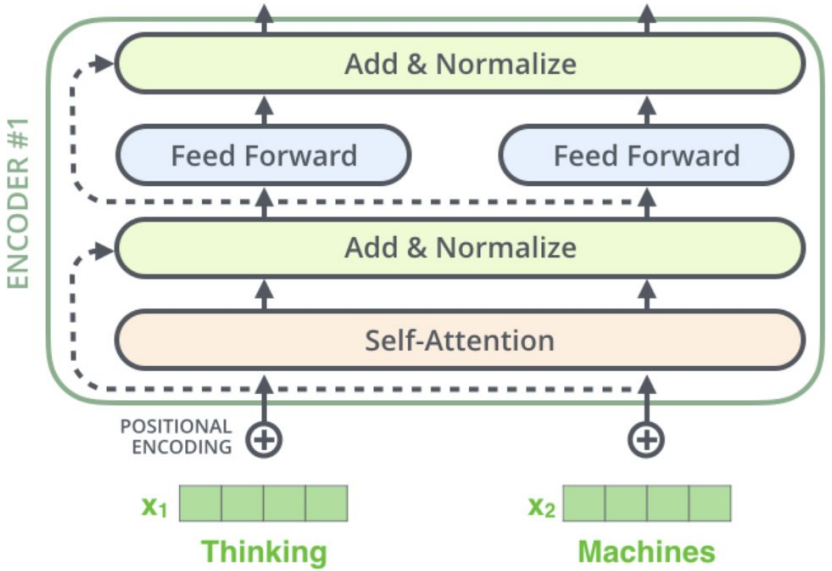
The Transformer

Overview

- Introduced by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin in the landmark paper *Attention is All You Need* at NIPS 2017
- Eliminated use of RNN type architectures entirely in favor of scaled dot product attention for feature extraction.
- Substantial improvements in most NLP tasks; widely regarded as the “ImageNet moment” for NLP.
- Its success in vanilla NLP tasks has inspired many to adapt the transformer to computer vision, speech recognition, and more.

The Transformer

- Use self attention mechanism for feature extraction.
- Multiple attention heads to enable representation of multiple unique features.
- Residual connections
- Layer Normalization
- Feed forward layers
- Compensate for use of location agnostic components by adding position encoding directly to the token embeddings.



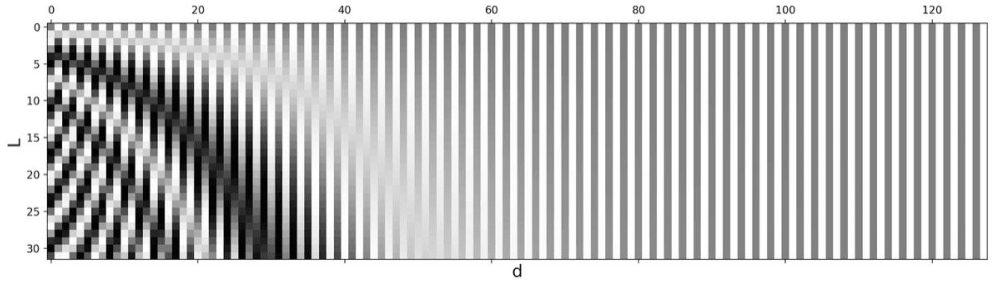
https://jalammar.github.io/illustrated-transformer/

Position Encoding

- The Transformer self-attention mechanism is permutation invariant; it does not capture position information.
- This information is essential for language tasks.
- Conventional solution:
  - Derive a position embedding for each token in the sequence.
  - Add this position embedding to the token embedding.
  - Use the result as input to the first transformer layer.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/512}}\right)$$

$$PE_{(pos,2*i+1)} = \cos\left(\frac{pos}{10000^{2i/512}}\right)$$



https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html

Model Archetypes

Encoder Only

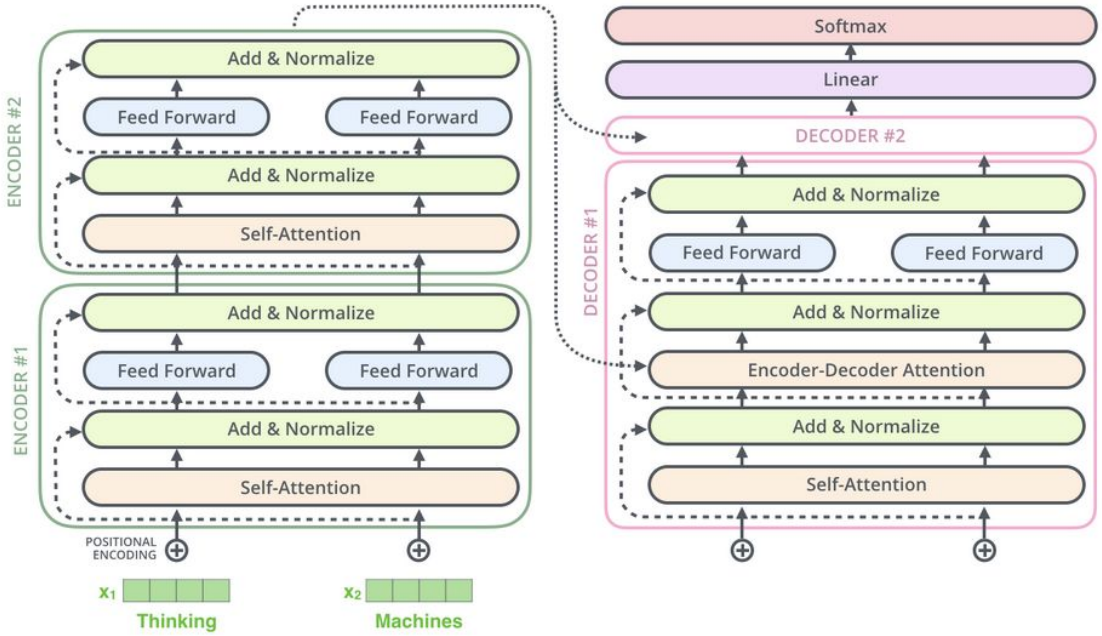
- BERT and Variants
- RoBERT
- ALBERT
- Longformer

Decoder Only

- GPT, GPT-2, GPT-3
- Transformer-XL
- XLNet

Encoder-Decoder

- T5
- BART
- Pegasus



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

ConViT- Improving Vision Transformers with Soft Convolutional Inductive Biases

Inductive Bias

- An inherent constraint or preference, exhibited in the model architecture or training process, that promotes (or requires) solutions of a particular kind.
  - CNN layers assume fine-grained feature locality
  - RNN assumes sequential relationships/causality.
  - Linear models assume linearity in parameters.
  - Etc etc.
- Models with appropriate inductive bias are often more consistent and sample efficient.
- Problems requiring greater representation power often benefit from less inductive biases.

Current State of Computer Vision

- Since AlexNet was introduced in 2012, CNN’s have dominated the field.
  - VGG
  - Resnet
  - Inception
- Success and generality of Transformers in NLP has inspired development of Vision Transformers (ViT)
- ViT have been shown to outperform pure CNN models in many situations.
- Contrastive pretraining techniques now dominate.

The Inductive Bias of the CNN

- Receptive field of a filter is learned and applied repeatedly over entire image. This imposes translation equivariance.
- Downsampling via max pooling imposes translation invariance
- Subsequent convolutional layers operate on downsampled representations of features locally extracted from early layers.
  - Features with long-range spatial dependencies are not well represented.
- Models that employ attention as the primary mechanism of feature extraction (transformers), do not impose significant (hard) inductive bias.

Feature Extraction and Inductive Bias

Architectural tradeoff:

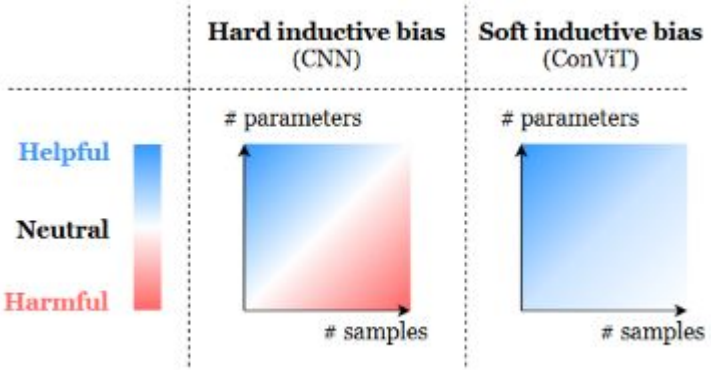
- More structured representation space -> Less parameters, sample efficient, more control over the nature of the representations found.
- Less structured representation space -> More parameters, less restrictions/biases imparted by the designer, greater representation capacity.
  - May require more data to see advantages.

Transformers impart relatively few restrictions on how features are to be extracted.

What is a “Soft” Inductive Bias?

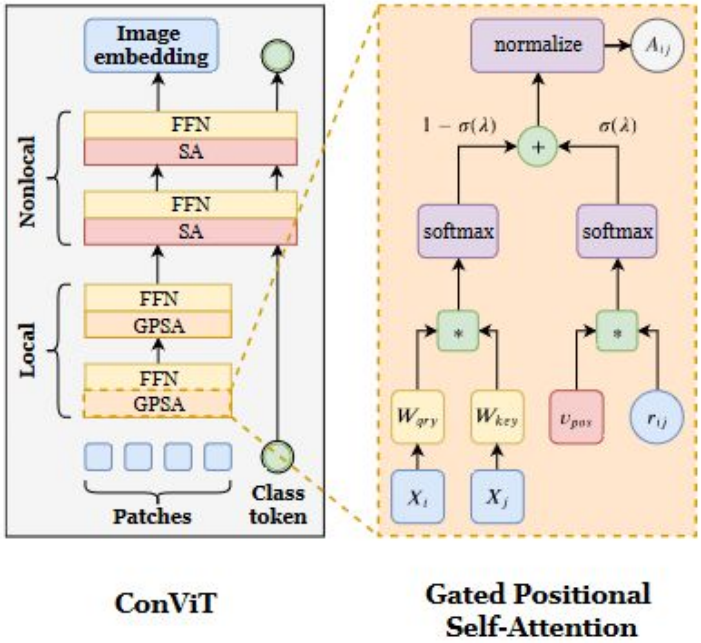
“In data-plentiful regimes, hard inductive biases can be overly restrictive and *learning* the most appropriate inductive bias can prove more effective. The practitioner is therefore confronted with a dilemma between using a convolutional model, which has a high performance floor but a potentially lower performance ceiling due to the hard inductive biases, or a self-attention based model, which has a lower floor but a higher ceiling. This dilemma leads to the following question: can one get the best of both worlds, and obtain the benefits of the convolutional inductive biases without suffering from its limitations”

- The authors explore a “hybrid model” in pursuit of a highly performant model with a high performance floor.



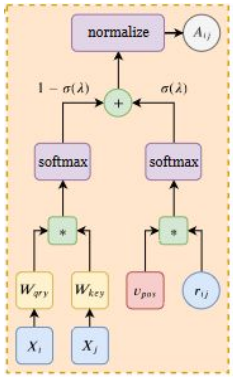
The ConViT Model

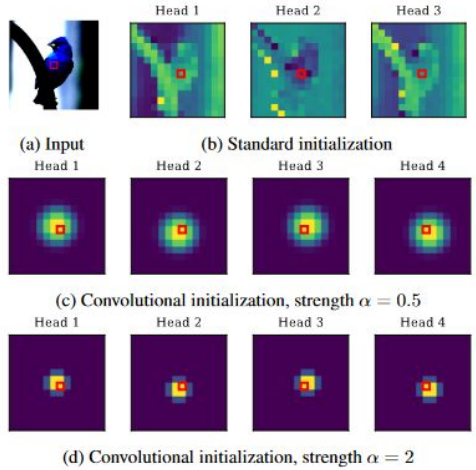
- Computer vision model; applied to ImageNet, a common supervised pre-training (image classification) dataset and benchmark.
- Appeared in ICML 38 in 2021
- Starts with a ViT (Dosovitskiy et al., 2020)
- Modifies early transformer blocks:
  - Replaces self attention (SA in diagram) with Gated Positional Self Attention (GPSA)
- GPSA layers, depending on the state of the parameters, can replicate standard convolution.



Gated Positional Self Attention

- In classic transformers positional information is added to content embedding before being passed to SA layer.
- Alternative is to encode patch positions, and learn appropriate positional embeddings.
- It can be shown that such Positional Self Attention is a form of generalized convolution.
- Adding gating mechanism with learnable λ parameter enforces scale consistency, and enables network to “choose” whether the layer behaves like a local convolution or a general attention layer.
    - Authors enforce the local regime at initialization for all GPSA layers.



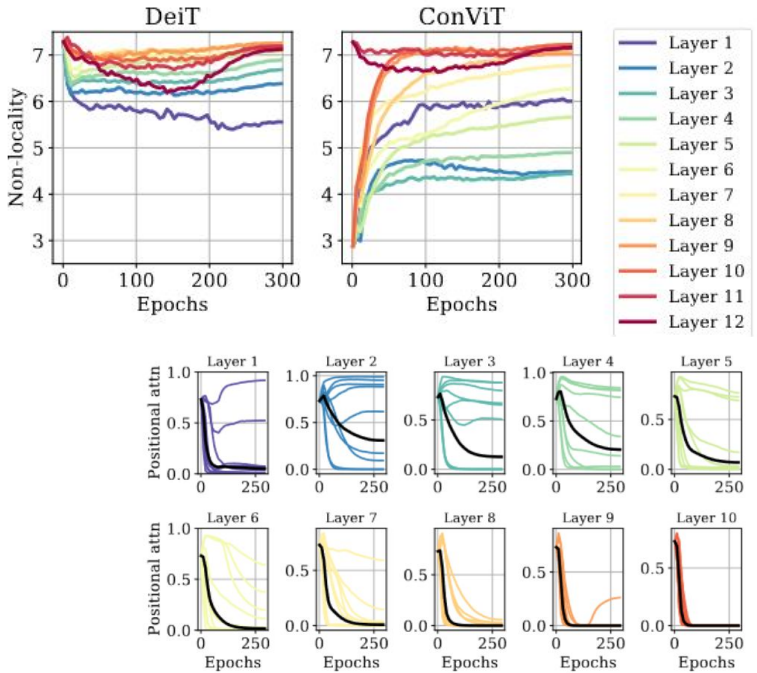


$$\begin{aligned}\mathrm{GPSA}_h(\boldsymbol{X}) &:= \mathrm{normalize}\left[\boldsymbol{A}^h\right]\boldsymbol{X}\boldsymbol{W}_{val}^h \\ \boldsymbol{A}_{ij}^h &:= (1-\sigma(\lambda_h))\,\mathrm{softmax}\left(\boldsymbol{Q}_i^h\boldsymbol{K}_j^{h\top}\right) \\ &\quad + \sigma(\lambda_h)\,\mathrm{softmax}\left(\boldsymbol{v}_{pos}^{h\top}\boldsymbol{r}_{ij}\right),\end{aligned}$$

Performance (ImageNet)

- When trained on entire dataset, performance improvements over DeiT are small.
- Smaller training datasets result in substantial outperformance, owing to the sample efficiency of the convolution-like GPSA layers in early iterations.
- Locality measures demonstrate early training of ConViT leveraging local receptive fields, particularly in first layers.

Train size	Top-1 DeiT	Top-1 ConViT	Top-1 Gap	Top-5 DeiT	Top-5 ConViT	Top-5 Gap
5%	34.8	**47.8**	37%	57.8	**70.7**	22%
10%	48.0	**59.6**	24%	71.5	**80.3**	12%
30%	66.1	**73.7**	12%	86.0	**90.7**	5%
50%	74.6	**78.2**	5%	91.8	**93.8**	2%
100%	79.9	**81.4**	2%	95.0	**95.8**	1%



Ablation Study

- The baseline is DeiT, where no GPSA layers, convolutional initialization, or gating parameter is used.
- The performance impact when only 10% of the training data is used is worse than for the full dataset.
- Using GPSA alone does not improve DeIT; it hurts performance.

Ref.	Train gating	Conv init	Train GPSA	Use GPSA	Full data	10% data
a (ConViT)	✓	✓	✓	✓	**82.2**	**59.7**
b	✗	✓	✓	✓	82.0	57.4
c	✓	✗	✓	✓	81.4	56.9
d	✗	✗	✓	✓	81.6	54.6
e (DeiT)	✗	✗	✗	✗	79.1	47.8
f	✗	✓	✗	✓	78.6	54.3
g	✗	✗	✗	✓	73.7	44.8

Discussion

- What soft inductive biases could be introduced to existing transformer architectures in NLP? Do any current models already do perhaps less deliberately?
- Would you expect the approach of ConViT to also affect sample efficiency for fine-tuning tasks?

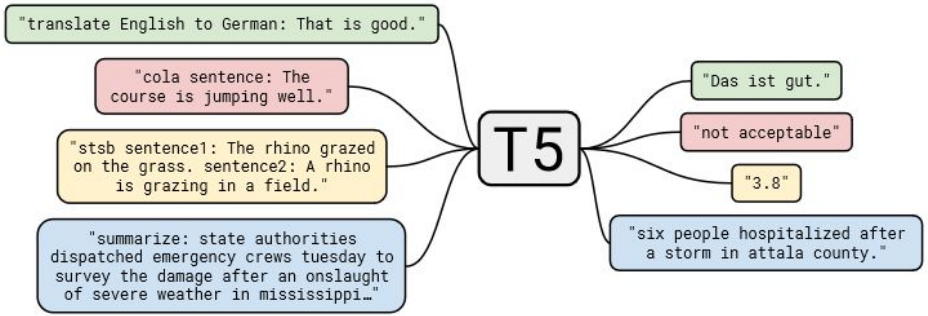
Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu

Google Research
Journal of Machine Learning Research 21 (2020)
https://arxiv.org/pdf/1910.10683.pdf

Overview

- Developed “unified framework” for pretraining by using text-text model that can be applied to many tasks.
- Unification enables rigorous evaluation of transfer learning methods.
- Develop the “Colossal Clean Crawled Corpus”
- Using a baseline model specification, conducted numerous experiments to identify key variables in transfer learning performance.
- Considering the results from such experiments, they developed T5
    - The largest variant is/was SOTA on several NLP tasks



“the text-to-text framework allows us to directly apply the same model, objective, training procedure, and decoding process to every task we consider”

The Colossal Clean Crawled Corpus (C4)

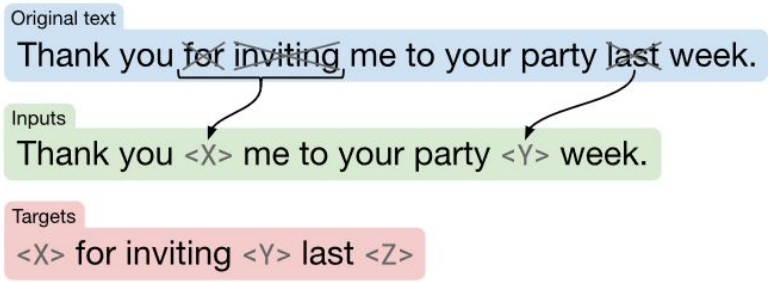
- Derived from the Common Crawl web archive
- Not all text is natural language, or otherwise suitable for NLP tasks that concerned the authors
- English language only
- Around 750 GB- significantly larger than other common pre-training datasets.

Modifications to the Common Crawl archive:

- We only retained lines that ended in a terminal punctuation mark (i.e. a period, exclamation mark, question mark, or end quotation mark).
- We discarded any page with fewer than 5 sentences and only retained lines that contained at least 3 words.
- We removed any page that contained any word on the "List of Dirty, Naughty, Obscene or Otherwise Bad Words".[6]
- Many of the scraped pages contained warnings stating that Javascript should be enabled so we removed any line with the word Javascript.
- Some pages had placeholder "lorem ipsum" text; we removed any page where the phrase "lorem ipsum" appeared.
- Some pages inadvertently contained code. Since the curly bracket "{" appears in many programming languages (such as Javascript, widely used on the web) but not in natural text, we removed any pages that contained a curly bracket.
- To deduplicate the data set, we discarded all but one of any three-sentence span occurring more than once in the data set.

Span Replacement

- For text-to-text to generalize to most NLP tasks, a general span ‘mask’ is used.
- As opposed to a single token mask widely employed in BERT-like model pretraining, a single span ‘sentinel’ token can represent an arbitrary number of tokens.
- There are no alignment issues that encoder only models present.
- The task context (“summarize”, “translate”, etc) is provided to the decoder as a prefix.



Experiments

- Text to Text Architectures
  - Encoder-decoder vs decoder only (LM) vs prefix decoder
- Unsupervised objectives
  - Denoising
- Pre-training dataset
- Training Strategy
  - Multi-task training
  - Fine tuning method
  - Multi-task + fine tuning method
- Performance scaling

NLP Tasks:

- Text classification (GLUE/SuperGLUE)
  - Includes: sentence acceptability, sentiment, sentence similarity, NL inference, coreference resolution, sentence completion, word sense disambiguation, question-answer.
- Question Answer (SQUAD)
- Translation
  - English to German, French, and Romanian
- Abstractive summarization (CNN/Daily Mail benchmark)

Experiments: Baseline Model

Pretraining:

- Designed to be similar to BERT base.
  - Encoder, decoder have 12 transformer blocks
  - 12 attention heads
  - 220mm parameters
- Train 524,288 steps on C4 dataset
- Standardize batch size so number of steps equates to ~34B tokens

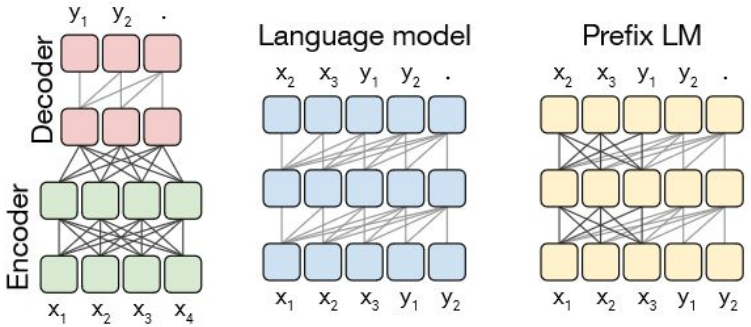
Fine Tuning:

- Additional 262,144 steps on all tasks.
- Fine tuned independently for all tasks
- Best checkpoint for each task chosen independently.

	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline average	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Baseline standard deviation	0.235	0.065	0.343	0.416	0.112	0.090	0.108
No pre-training	66.22	17.60	50.31	53.04	25.86	**39.77**	24.04

Experiments: Architecture Experiment

- Compared encoder-decoder, LM, to Prefix LM
- Prefix LM is like a standard LM, but the attention mask permits attending to subsequent tokens within the prefix.
- The denoising objective consistently outperformed the autoregressive LM task
- The encoder-decoder is difficult to compare due to either greater parameter count, or lesser computation cost.



Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

Experiments: Unsupervised Objectives

- BERT style masking outperformed shuffling and prefix LM.
- The span replacement and token deletion techniques performed best.
- After varying the token corruption rate, the authors selected 15% as their standard pre-training approach going forward.
- Corrupting spans of longer length, on average, produced best results at a span of 3.

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style (Devlin et al., 2018)	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

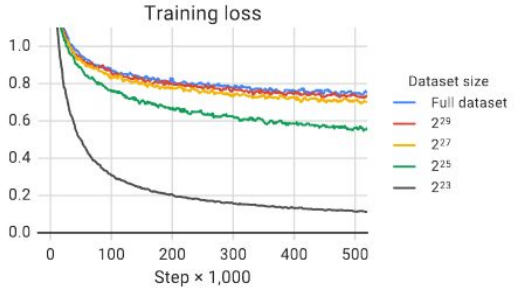
Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style (Song et al., 2019)	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (i.i.d.)	**83.28**	19.24	80.88	71.36	**26.98**	**39.82**	**27.65**
2	**83.54**	19.39	**82.09**	**72.20**	**26.76**	**39.99**	**27.63**
3	**83.49**	**19.62**	**81.84**	**72.53**	**26.86**	39.65	**27.62**
5	**83.40**	19.24	**82.05**	**72.23**	**26.88**	39.40	**27.53**
10	82.85	19.33	**81.84**	70.44	**26.79**	39.49	**27.69**

Experiments: Pre-training Dataset

- The C4 filtering process clearly improves performance.
- Baseline training time isn't long enough (or large enough?) to truly assess the value of C4 by leveraging its size advantage.
- In domain pre-training appears beneficial
- Some data repetition is not necessarily harmful



Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full data set	0	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
2^{29}	64	**82.87**	**19.19**	**80.97**	**72.03**	**26.83**	**39.74**	**27.63**
2^{27}	256	82.62	**19.20**	79.78	69.97	**27.02**	**39.71**	27.33
2^{25}	1,024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
2^{23}	4,096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

Experiments: Fine-Tuning Methods

- Gradual unfreezing: From the last layer to the first layer iteratively unfreeze during fine-tuning.
- Adapter layers: additional feed forward network block added after each transformer block.
  - Parameter d: dimensionality of the inner block of the FFN.
- Result: simply fine-tuning all parameters was universally superior for all tasks.

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Adapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
Adapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Adapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
Adapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Gradual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

Experiments: Multi-task Learning

- Instead of fine-tuning the model independently for each task, mix fine-tuning tasks in with the denoising pretraining task.
- Challenges arise: How to balance iterations spent on what tasks?
  - Avoid overfitting low resource tasks
  - Avoid undertraining high resource tasks
  - Huge dataset size imbalance.
- Completely failed for most tasks

Mixing strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (pre-train/fine-tune)	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Equal	76.13	19.02	76.51	63.37	23.89	34.31	26.78
Examples-proportional, $K = 2^{16}$	80.45	19.04	77.25	69.95	24.35	34.99	27.10
Examples-proportional, $K = 2^{17}$	81.56	19.12	77.00	67.91	24.36	35.00	27.25
Examples-proportional, $K = 2^{18}$	81.67	19.07	78.17	67.94	24.57	35.19	27.39
Examples-proportional, $K = 2^{19}$	81.42	**19.24**	79.78	67.30	25.21	36.30	**27.76**
Examples-proportional, $K = 2^{20}$	80.80	**19.24**	**80.36**	67.38	25.66	36.93	**27.68**
Examples-proportional, $K = 2^{21}$	79.83	18.79	79.50	65.10	25.82	37.22	27.13
Temperature-scaled, $T = 2$	81.90	**19.28**	79.42	69.92	25.42	36.72	27.20
Temperature-scaled, $T = 4$	80.56	**19.22**	77.99	69.54	25.04	35.82	27.45
Temperature-scaled, $T = 8$	77.21	19.10	77.14	66.07	24.55	35.35	27.17

Examples-proportional: sample in proportion to each tasks dataset size with maximum size K.

Experiments: Multi-task Learning + Fine Tuning

- Try the previous Multi-task approach, but then fine-tune.
- Rationelle is that “early exposure” to the task could improve task specific representation suitability.
- Results were comparable to the standard unsupervised pretraining + fine-tuning approach.
- Eliminating unsupervised C4 pretraining entirely yielded quite diverse results.

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

Experiments: Scaling

- Given 4x computational resources, what allocation of these resources produce greatest performance uplift?
- In general, it appears that model size is a key limiting factor here.
- Ensembling yielded tangible improvements.
    - The fine-tune only ensemble requires substantially less resources as it shares pretrained weights.

Scaling strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline	83.28	19.24	80.88	71.36	26.98	39.82	27.65
1× size, 4× training steps	85.33	19.33	82.45	74.72	27.08	40.66	27.93
1× size, 4× batch size	84.60	19.42	82.52	74.64	27.07	40.60	27.84
2× size, 2× training steps	**86.18**	19.66	**84.18**	77.18	27.52	**41.03**	28.19
4× size, 1× training steps	**85.91**	19.73	**83.86**	**78.04**	27.47	40.71	28.10
4× ensembled	84.77	**20.10**	83.09	71.74	**28.05**	40.53	**28.57**
4× ensembled, fine-tune only	84.05	19.57	82.36	71.55	27.55	40.22	28.09

The T5 Model

- Given the results of the previous experiments, the authors developed a model and training approach:
  - Encoder-decoder
  - Mean corruption span of 3, 15% of tokens
  - 1T pretraining tokens (32x baseline)
  - Multi-task pretraining + fine-tuning
  - Beam search instead of greedy decoding

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**90.3**	**71.6**	**97.5**	92.8	90.4	**93.1**	**92.8**

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	74.8[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	**75.1**	90.6	**92.2**	**91.9**	96.9	**92.8**	**94.5**

Model	SQuAD EM	SQuAD F1	SuperGLUE Average	BoolQ Accuracy	CB F1	CB Accuracy	COPA Accuracy
Previous best	90.1[a]	95.5[a]	84.6[d]	87.1[d]	90.5[d]	95.2[d]	90.6[d]
T5-Small	79.10	87.24	63.3	76.4	56.9	81.6	46.0
T5-Base	85.44	92.08	76.2	81.4	86.2	94.0	71.2
T5-Large	86.66	93.79	82.3	85.4	91.6	94.8	83.4
T5-3B	88.53	94.95	86.4	89.9	90.3	94.4	92.0
T5-11B	**91.26**	**96.22**	**88.9**	**91.2**	**93.9**	**96.8**	**94.8**

Model	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD Accuracy	RTE Accuracy	WiC Accuracy	WSC Accuracy
Previous best	84.4[d]	52.5[d]	90.6[d]	90.0[d]	88.2[d]	69.9[d]	89.0[d]
T5-Small	69.3	26.3	56.3	55.4	73.3	66.9	70.5
T5-Base	79.7	43.1	75.0	74.2	81.5	68.3	80.8
T5-Large	83.3	50.7	86.8	85.9	87.8	69.3	86.3
T5-3B	86.8	58.3	91.2	90.4	90.7	72.1	90.4
T5-11B	**88.1**	**63.3**	**94.1**	**93.4**	**92.5**	**76.9**	**93.8**

Model	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU	CNN/DM ROUGE-1	CNN/DM ROUGE-2	CNN/DM ROUGE-L
Previous best	**33.8**[e]	**43.8**[e]	**38.5**[f]	43.47[g]	20.30[g]	40.63[g]
T5-Small	26.7	36.0	26.8	41.12	19.56	38.35
T5-Base	30.9	41.2	28.0	42.05	20.34	39.40
T5-Large	32.0	41.5	28.1	42.50	20.68	39.75
T5-3B	31.8	42.6	28.2	42.72	21.02	39.94
T5-11B	32.1	43.4	28.1	**43.52**	**21.55**	**40.69**

Discussion

- Why does T5, as trained, not yield SOTA performance on the translation tasks?
- The unified text-to-text approach still required independent task fine-tuning to obtain the best results. Does this defeat the point?

DeBERTa: Decoding-enhanced BERT with Disentangled Attention

Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen
Microsoft Research
ICLR 2021

Overview

- BERT-like model
- Very recent model. A new top performer in GLUE benchmark
- ICLR 2021
- Microsoft Research
- Key architectural introductions:
  - Disentangled attention
  - Enhanced Mask Decoder
- Achieves SOTA performance on several benchmarks.

Disentangled Attention

- In standard BERT, input vectors are the sum of word embedding and position embedding
- DeBERTa does not add the embeddings; attention score is decomposed into
  - Content to content attention
  - Content to position attention
  - Position to content attention
- Softmax of the scaled sum of attention scores, projected onto the content values yields attention layer output.
- In largest model (1.5B), attention positional projection matrices are shared within the layer.

Conventional self-attention:

$$\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{q}}, \boldsymbol{K} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{k}}, \boldsymbol{V} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{v}}, \boldsymbol{A} = \frac{\boldsymbol{Q}\boldsymbol{K}^{\intercal}}{\sqrt{d}}$$

$$\boldsymbol{H}_{\boldsymbol{o}} = \mathrm{softmax}(\boldsymbol{A})\boldsymbol{V}$$

Disentangled self-attention:

$$\delta(i,j) = \begin{cases} 0 & \text{for} \quad i-j \leqslant -k \\ 2k-1 & \text{for} \quad i-j \geqslant k \\ i-j+k & \text{others.} \end{cases}$$

$$\boldsymbol{Q}_{\boldsymbol{c}} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{q},\boldsymbol{c}}, \boldsymbol{K}_{\boldsymbol{c}} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{k},\boldsymbol{c}}, \boldsymbol{V}_{\boldsymbol{c}} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{v},\boldsymbol{c}}, \boldsymbol{Q}_{\boldsymbol{r}} = \boldsymbol{P}\boldsymbol{W}_{\boldsymbol{q},\boldsymbol{r}}, \boldsymbol{K}_{\boldsymbol{r}} = \boldsymbol{P}\boldsymbol{W}_{\boldsymbol{k},\boldsymbol{r}}$$

$$\tilde{A}_{i,j} = \underbrace{\boldsymbol{Q}_{\boldsymbol{i}}^{\boldsymbol{c}}\boldsymbol{K}_{\boldsymbol{j}}^{\boldsymbol{c}\intercal}}_{\text{(a) content-to-content}} + \underbrace{\boldsymbol{Q}_{\boldsymbol{i}}^{\boldsymbol{c}}\boldsymbol{K}_{\boldsymbol{\delta}(\boldsymbol{i},\boldsymbol{j})}^{\boldsymbol{r}\intercal}}_{\text{(b) content-to-position}} + \underbrace{\boldsymbol{K}_{\boldsymbol{j}}^{\boldsymbol{c}}\boldsymbol{Q}_{\boldsymbol{\delta}(\boldsymbol{j},\boldsymbol{i})}^{\boldsymbol{r}\intercal}}_{\text{(c) position-to-content}}$$

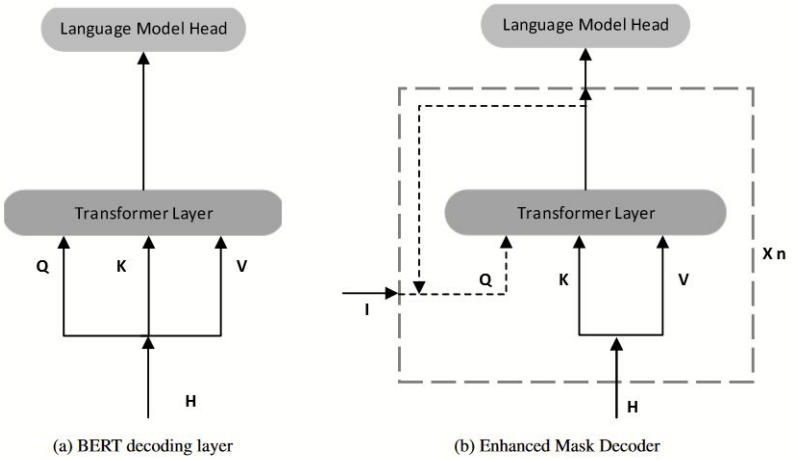
$$\boldsymbol{H}_{\boldsymbol{o}} = \mathrm{softmax}(\frac{\tilde{\boldsymbol{A}}}{\sqrt{3d}})\boldsymbol{V}_{\boldsymbol{c}}$$

Enhanced Mask Decoder (EMD)

- Actually incorporates *absolute* position embeddings before softmax layer
- Enables model to avoid conflating phrases with identical relative position;
  - Paper example:

    “A new store opened beside the new mall”

- BERT incorporates absolute position information at input.
- DeBERTa uses absolute position at the last transformer layer. This layer is the enhanced mask decoder.



(a) BERT decoding layer

(b) Enhanced Mask Decoder

Scale Invariant Fine Tuning (SiFT)

- Instance of a virtual adversarial training algorithm.
  - Seeks to achieve robustness to adversarial examples: achieve same output distribution on adversarial perturbations of task-specific examples.
- Applies perturbations to the normalized word embeddings
- Only applied to the largest model (1.5B parameters)
  - Sensitivity to perturbations increases with model size.
- Very little emphasis on this technique, authors indicate desire investigate more thoroughly in future work.

Data, Training

- English Wikipedia dump 12GB
- BookCorpus 6GB
- OPENWEBTEXT (reddit) 38GB
- STORIES (Common Crawl) 31GB

Total size after cleaning: 78GB

- Trained for 1mm steps
- Batch size of 2k

Model	Wiki+Book 16GB	OpenWebText 38GB	Stories 31GB	CC-News 76GB	Giga5 16GB	ClueWeb 19GB	Common Crawl 110GB
BERT	✓						
XLNet	✓				✓	✓	✓
RoBERTa	✓	✓	✓	✓			
DeBERTa	✓	✓	✓				
DeBERTa$_{1.5B}$	✓	✓	✓	✓			

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
General Language Understanding Evaluation (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
WNLI	NLI	634	71	146	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr
SuperGLUE						
WSC	Coreference	554k	104	146	2	Accuracy
BoolQ	QA	9,427	3,270	3,245	2	Accuracy
COPA	QA	400k	100	500	2	Accuracy
CB	NLI	250	57	250	3	Accuracy/F1
RTE	NLI	2.5k	276	3k	2	Accuracy
WiC	WSD	2.5k	276	3k	2	Accuracy
ReCoRD	MRC	101k	10k	10k	-	Exact Match (EM)/F1
MultiRC	Multiple choice	5,100	953	1,800	-	Exact Match (EM)/F1
Question Answering						
SQuAD v1.1	MRC	87.6k	10.5k	9.5k	-	Exact Match (EM)/F1
SQuAD v2.0	MRC	130.3k	11.9k	8.9k	-	Exact Match (EM)/F1
RACE	MRC	87,866	4,887	4,934	4	Accuracy
SWAG	Multiple choice	73.5k	20k	20k	4	Accuracy
Token Classification						
CoNLL 2003	NER	14,987	3,466	3,684	8	F1

Performance

- The 1.5B parameter model is the first model to surpasses human performance on SuperGLUE benchmark.
- Outperforms T5 on many subscores, despite using 9.5B fewer parameters.
- Despite being trained on less data, DeBERTa outperformed XLNet virtually everywhere.

Model	MNLI-m/mm (Acc)	SQuAD v1.1 (F1/EM)	SQuAD v2.0 (F1/EM)
RoBERTa$_{base}$	87.6/-	91.5/84.6	83.7/80.5
XLNet$_{base}$	86.8/-	-/-	-/80.2
DeBERTa$_{base}$	**88.8/88.5**	**93.1/87.2**	**86.2/83.1**

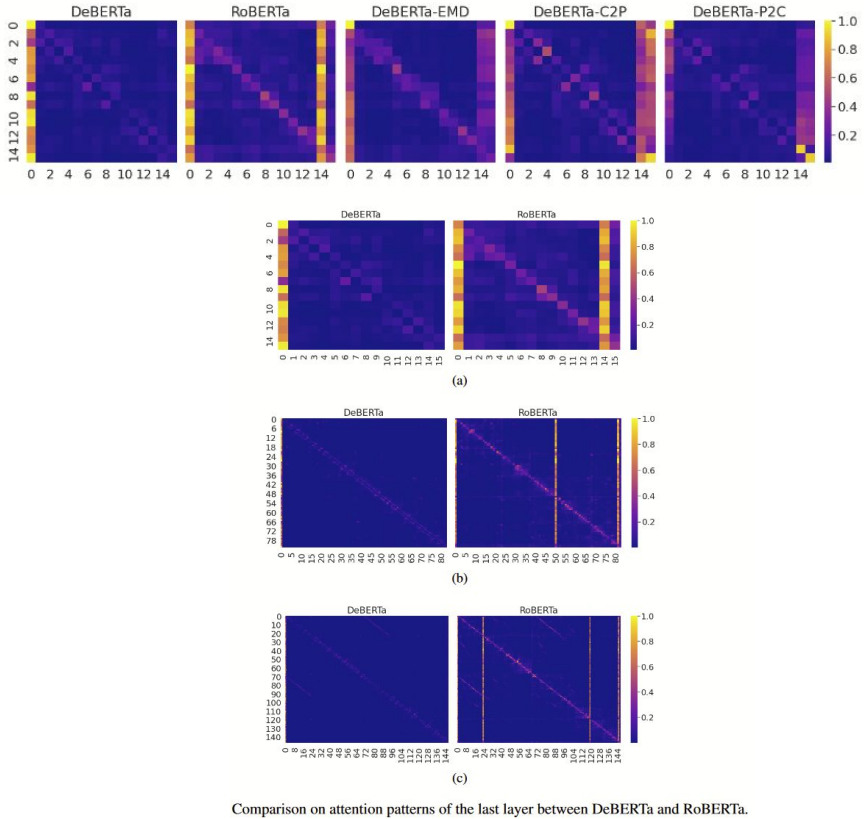
Model	MNLI-m/mm Acc	SQuAD v1.1 F1/EM	SQuAD v2.0 F1/EM	RACE Acc	ReCoRD F1/EM	SWAG Acc	NER F1
BERT$_{large}$	86.6/-	90.9/84.1	81.8/79.0	72.0	-	86.6	92.8
ALBERT$_{large}$	86.5/-	91.8/85.2	84.9/81.8	75.2	-	-	-
RoBERTa$_{large}$	90.2/90.2	94.6/88.9	89.4/86.5	83.2	90.6/90.0	89.9	93.4
XLNet$_{large}$	90.8/90.8	95.1/89.7	90.6/87.9	85.4	-	-	-
Megatron$_{336M}$	89.7/90.0	94.2/88.0	88.1/84.8	83.0	-	-	-
DeBERTa$_{large}$	**91.1/91.1**	**95.5/90.1**	**90.7/88.0**	**86.8**	**91.4/91.0**	**90.8**	**93.8**
ALBERT$_{xxlarge}$	90.8/-	94.8/89.3	90.2/87.4	86.5	-	-	-
Megatron$_{1.3B}$	90.9/91.0	94.9/89.1	90.2/87.1	87.3	-	-	-
Megatron$_{3.9B}$	91.4/91.4	95.5/90.0	91.2/88.5	89.5	-	-	-

Model	CoLA Mcc	QQP Acc	MNLI-m/mm Acc	SST-2 Acc	STS-B Corr	QNLI Acc	RTE Acc	MRPC Acc	Avg.
BERT$_{large}$	60.6	91.3	86.6/-	93.2	90.0	92.3	70.4	88.0	84.05
RoBERTa$_{large}$	68.0	92.2	90.2/90.2	96.4	92.4	93.9	86.6	90.9	88.82
XLNet$_{large}$	69.0	92.3	90.8/90.8	**97.0**	92.5	94.9	85.9	90.8	89.15
ELECTRA$_{large}$	69.1	**92.4**	90.9/-	96.9	92.6	95.0	88.0	90.8	89.46
DeBERTa$_{large}$	**70.5**	92.3	**91.1/91.1**	96.8	**92.8**	**95.3**	**88.3**	**91.9**	**90.00**

Model	BoolQ Acc	CB F1/Acc	COPA Acc	MultiRC F1a/EM	ReCoRD F1/EM	RTE Acc	WiC Acc	WSC Acc	Average Score
RoBERTa$_{large}$	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	84.6
NEXHA-Plus	87.8	94.4/96.0	93.6	84.6/55.1	90.1/89.6	89.1	74.6	93.2	86.7
T5$_{11B}$	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	89.3
T5$_{11B}$+Meena	**91.3**	**95.8/97.6**	97.4	88.3/63.0	94.2/93.5	92.7	**77.9**	95.9	90.2
Human	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	89.8
DeBERTa$_{1.5B}$+SiFT	90.4	94.9/97.2	96.8	**88.2/63.7**	**94.5/94.1**	**93.2**	76.4	**95.9**	89.9
DeBERTa$_{Ensemble}$	90.4	95.7/97.6	**98.4**	88.2/63.7	94.5/94.1	93.2	77.5	95.9	**90.3**

Effects of Disentangled Attention and the EMD

- Final layer in DeBERTa exhibits less tendency for a token to attend to itself.
- Vertical strips correspond to special tokens.
  - RoBERTa attends more to punctuation and functional words
  - DeBERTa only has vertical strip at [CLS] token.
  - Authors argue DeBERTa focus on [CLS] is desirable
- Removal of EMD or any disentangled attention component restored the diagonal attention weights seen in the figures.



Comparison on attention patterns of the last layer between DeBERTa and RoBERTa.

Ablation Study

- The removal of any new component introduced by the authors results in a performance drop
- RACE benchmark appears most sensitive.

Model	MNLI-m/mm Acc	SQuAD v1.1 F1/EM	SQuAD v2.0 F1/EM	RACE Acc
BERT_{base} Devlin et al. (2019)	84.3/84.7	88.5/81.0	76.3/73.7	65.0
RoBERTa_{base} Liu et al. (2019c)	84.7/-	90.6/-	79.7/-	65.6
XLNet_{base} Yang et al. (2019)	85.8/85.4	-/-	81.3/78.5	66.7
$\text{RoBERTa-ReImp}_{base}$	84.9/85.1	91.1/84.8	79.5/76.0	66.8
DeBERTa_{base}	**86.3/86.2**	**92.1/86.1**	**82.5/79.3**	**71.7**
-EMD	86.1/86.1	91.8/85.8	81.3/78.0	70.3
-C2P	85.9/85.7	91.6/85.8	81.3/78.3	69.3
-P2C	86.0/85.8	91.7/85.7	80.8/77.6	69.6
-(EMD+C2P)	85.8/85.9	91.5/85.3	80.3/77.2	68.1
-(EMD+P2C)	85.8/85.8	91.3/85.1	80.2/77.1	68.5

Discussion

- Disentangling the position encoding from the token embedding improves performance by more effectively propagating position information. Could a (gated) position self attention used in ConViT be effective in NLP? Why or why not?
- What might explain the greater parameter efficiency of DeBERTa vs T5?

Transformers: “The End of History” for Natural Language Processing?

Anton Chernyavskiy, Dmitry Ilvovsky, Preslav Nakov

HSE University, Qatar Computing Research Institute
arXiV preprint

Overview

- Transformers have proven to be game changing for NLP
- Research efforts have taken the basic transformer and evolved it to improve performance.
- Much effort has been devoted to understanding how attention layers capture linguistic features.
- Limited demonstration of whether capturing such features results in the effective utilization of such in downstream tasks.
- The authors observe common weaknesses intrinsic to the transformer architecture that have not been addressed.
- The authors propose and test basic ways of addressing the limitations and make additional recommendations.

General Weaknesses of Transformer Models

- Self attention patterns are developed in pre-training rather than in task-specific fine-tuning.
- The models is overparameterized.
- Insensitivity to negation
- Not robust to misspelling
- Vulnerable to adversarial attacks
- Challenging to tune hyperparameters.

Previous Improvements over BERT

RoBERTa: Removes next sentence pretraining, bigger batches, longer training, better hyperparameter tuning.

DistilBERT: Reduces overparameterization

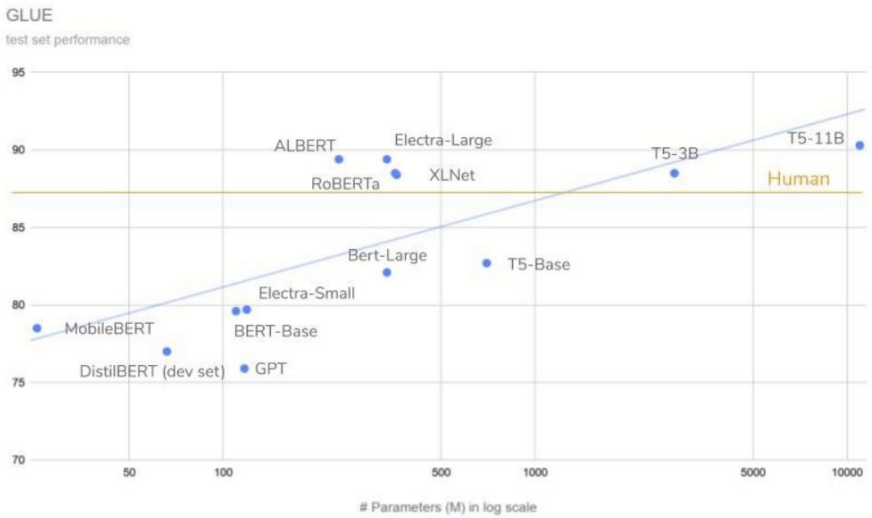
ALBERT: Parameter reduction via cross-layer sharing

Transformer-XL: Increases context by enabling access to previously computed hidden states.

XLNet: Models permutations of the input sequence; better captures bidirectional dependencies.

BERT-CRF: Applies conditional random field to BERT output.

KnowBERT: Provides knowledge external to the input.



https://towardsdatascience.com/gpt-3-transformers-and-the-wild-world-of-nlp-9993d8bb1314?gi=f00d37529cba

Test Environment: Tasks

- Propaganda Detection
  - Subtasks
    - Span identification: Identify all propaganda spans. Cast as a NER problem.
    - Technique classification: classify each span into one of 14 propaganda techniques
  - Rationale:
    - Includes labeling and classification tasks
    - “Specificity necessary for our research”
    - Spans can be nested
- Keyphrase Extraction
  - Subtasks
    - Keyphrase identification
    - Keyphrase classification (3 class)
  - Rationelle:
    - Length of phrases is an important feature
    - Phrases can be nested
    - Phrase duplication across articles.

Test Environment: Models, Data

Model Baseline:

- Best performer of BERT, RoBERTa, ALBERT, XLNet chosen for each task:
  - Propaganda detection: RoBERTa
  - Keyphrase Extraction: XLNet
- Use 3 initialization ensemble, using the intersection of identified spans.

Data:

Propaganda detection: SemEval 2020 task 11

- 371/75 development, 90 testing
- Split development 80:20 train:validation
- F1 score evaluation

Keyphrase Extraction: SemEval 2017 task 10

- 350/50/100 train/dev/test English articles

Identification Tasks: Model Changes

Problem 1:

Spans of “positive” tokens must follow a “beginning” token, but RoBERTa will frequently ignore this rule.

Proposed Solution:

Replace classification head with Conditional Random Field (CRF) layer to help enforce tagging rules between neighboring tokens.

Problem 2:

Insufficient attention weights to punctuation and [SEP] tokens.

Proposed Solution:

Enforce the requirement that a span cannot begin or end with punctuation unless enclosed in quotation marks. Add these in post-processing.

Classification Tasks: Model Changes

Problem: BERT cannot do explicit word/subword counting. Tagging task is compromised

Solution: Add length feature to [CLS] token embedding. Concatenate the associated RoBERTA embedding with the span embedding before classification.

Problem: Propaganda techniques contain common words; such knowledge is inaccessible to the model.

Solution: Create an externally constructed “gazetteer” that incorporates training set class distributions. Use in post-processing to avoid overfitting.

Problem: Model produces nested classes, while there are no such occurrences in the training data.

Solution: Again postprocessing by computing the empirical probability of such a nesting occurring, and multiplying the predicted occurrence probability by such empirical probability.

Classification Tasks: Model Changes (cont.)

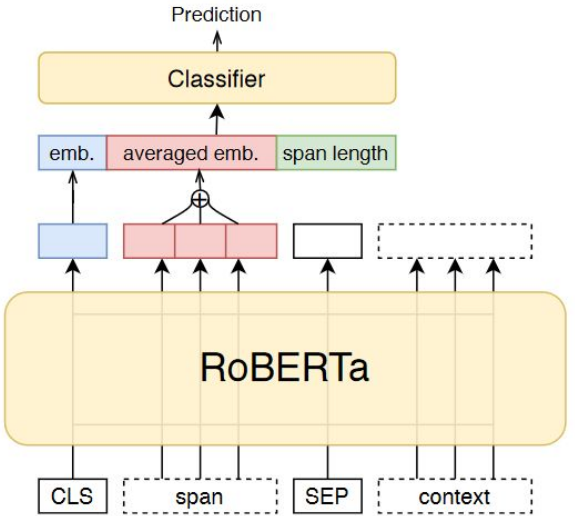
Problem: The ‘repetition’ propaganda technique requires entire document context.

Solution: Post-processing. Adjust class with repetitions k with probability p by:

$$\hat{p} = \begin{cases} 1, & \text{if } k \geq 3 \text{ or } (k = 2 \text{ and } p \geq t_1) \\ 0, & \text{if } k = 1 \text{ and } p \leq t_2 \\ p, & \text{otherwise} \end{cases}$$

Problem: Many spans can have multiple labels

Solution: Post-processing. Classify according to the top k scores produced by the classifier if the span occurs multiple times in the target spans.



Identification Task Results

- All Model variants benefited from the inclusion of a CRF.
  - XLNet benefited most when compared to the other model architectures.

Task	Approach	F1
SI	RoBERTa (BIO encoding)	46.91
	+ CRF	48.54 $_{\uparrow 1.63}$
	+ punctuation post-processing	47.54 $_{\uparrow 0.63}$
	Overall	48.87 $_{\uparrow 1.96}$
	XLNet (BIO encoding)	46.47
	+ CRF	46.68 $_{\uparrow 0.21}$
	+ punctuation post-processing	46.76 $_{\uparrow 0.29}$
	Overall	47.05 $_{\uparrow 0.58}$
KI	RoBERTa (BIO encoding)	57.85
	+ CRF	58.59 $_{\uparrow 0.74}$
	XLNet (BIO encoding)	58.80
	+ CRF	60.11 $_{\uparrow 1.31}$

Table 1. Analysis of RoBERTa and XLNet modifications for sequential classification tasks: span identification and keyphrase identification. *Overall* is the simultaneous application of two improvements.

Classification Task Results

- Repetition post-processing provides the most improvement for Technique classification.
- Gazetteer knowledge was minimally helpful for technique classification.
- XLNet, on average benefits from these post-processing techniques more than RoBERTa for technique classification.

Technique Classification		
Approach	**RoBERTa**	**XLNet**
Baseline+multi-label	63.78	59.27
+ length	64.72 $_{\uparrow 0.94}$	60.68 $_{\uparrow 1.41}$
+ averaged span embededding	64.25 $_{\uparrow 0.47}$	60.77 $_{\uparrow 1.50}$
+ gazetteer post-processing	63.87 $_{\uparrow 0.09}$	59.36 $_{\uparrow 0.09}$
+ *repetition* post-processing	67.54 $_{\uparrow 3.76}$	63.50 $_{\uparrow 4.23}$
+ class insertions	63.69 $_{\downarrow 0.09}$	58.89 $_{\downarrow 0.38}$

Table 4. Analysis of the improvements for the TC task using RoBERTa and XLNet on the development set in the multi-label mode. Shown is micro-F_1 score.

Technique Classification		
Approach	**RoBERTa**	**XLNet**
Baseline	62.75	58.23
+ length	63.50 $_{\uparrow 0.75}$	59.64 $_{\uparrow 1.41}$
+ averaged span embededding	62.94 $_{\uparrow 0.19}$	59.64 $_{\uparrow 1.41}$
+ multi-label	63.78 $_{\uparrow 1.03}$	59.27 $_{\uparrow 1.04}$
+ gazetteer post-processing	62.84 $_{\uparrow 0.10}$	58.33 $_{\uparrow 0.10}$
+ *repetition* post-processing	66.79 $_{\uparrow 4.04}$	62.46 $_{\uparrow 3.67}$
+ class insertions	62.65 $_{\downarrow 0.10}$	57.85 $_{\downarrow 0.38}$

Table 2. Analysis of the improvements using RoBERTa and XLNet for the TC task on the development set. Shown is micro-F_1 score.

Keyphrase Classification		
Approach	**RoBERTa**	**XLNet**
Baseline	77.18	78.50
+ length	77.38 $_{\uparrow 0.20}$	78.65 $_{\uparrow 0.15}$
+ gazetteer post-processing	77.43 $_{\uparrow 0.25}$	78.69 $_{\uparrow 0.19}$
+ class insertions	77.82 $_{\uparrow 0.64}$	78.69 $_{\uparrow 0.19}$

Table 3. Analysis of improvements for the KC task using RoBERTa and XLNet on the development set in the multi-label mode. Shown is micro-F_1 score.

Discussion Questions

- Most of the authors' improvements involve post-processing to enforce known rules (prior information) that are task specific. Are there any other methods of incorporating this information without such hand-tuning?
- What are possible causes of the shortcomings the authors identified? Are they intrinsic to transformers or could they be ameliorated by minor architectural modifications, training methods, different pre-training tasks, etc?

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, 2017.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016

Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. Transformers "The End of History" for Natural Language Processing? arXiv:2105.00813v2

Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An Attentive Survey of Attention Models arXiv:1904.02874v3