

# Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Presenter: Huangrui Chu

CPSC 670: Topics in Natural Language Processing

Instructor: Arman Cohan

# Basic Information

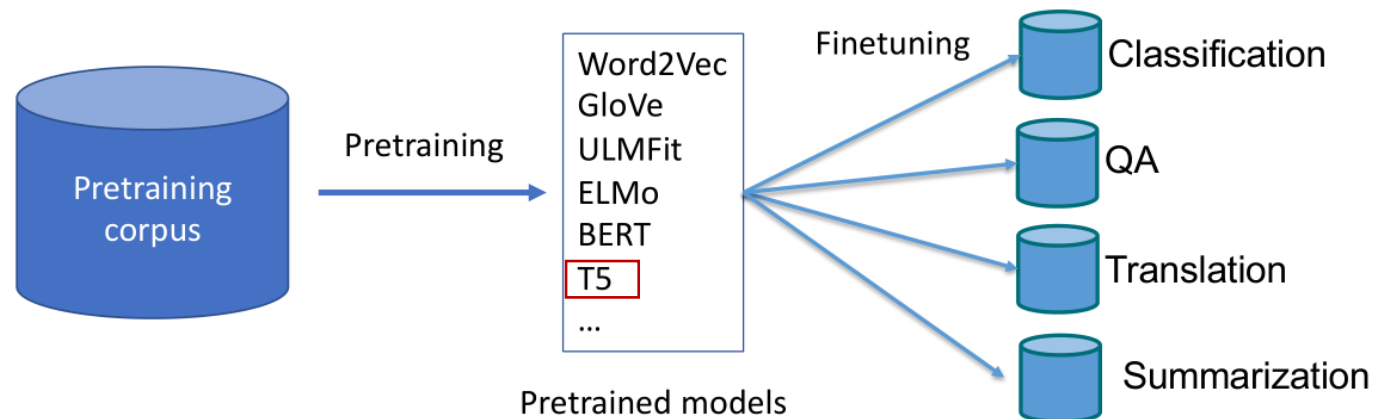
---

- 44 pages
- Total time: around 30 mins
- Basic Information
- General Idea
- Some details
- Reference:  
[https://zhuanlan.zhihu.com/p/88438851?utm\\_medium=social&utm\\_oi=949958707937865728&utm\\_psn=1600265640807415808&utm\\_source=wechat\\_session](https://zhuanlan.zhihu.com/p/88438851?utm_medium=social&utm_oi=949958707937865728&utm_psn=1600265640807415808&utm_source=wechat_session)

# What we learned from previous lecture

## Transfer learning in NLP

- The general transfer learning refers:
  - Training a model to perform one task/dataset/set of tasks
  - Then transfer to another task/dataset/set of tasks
  - Often refers to training a language model then transferring to downstream tasks





# Problem

---

It is often possible to achieve better performance simply by training a larger model on a larger data set.

---

a great deal of recent work developing transfer learning methodology for NLP

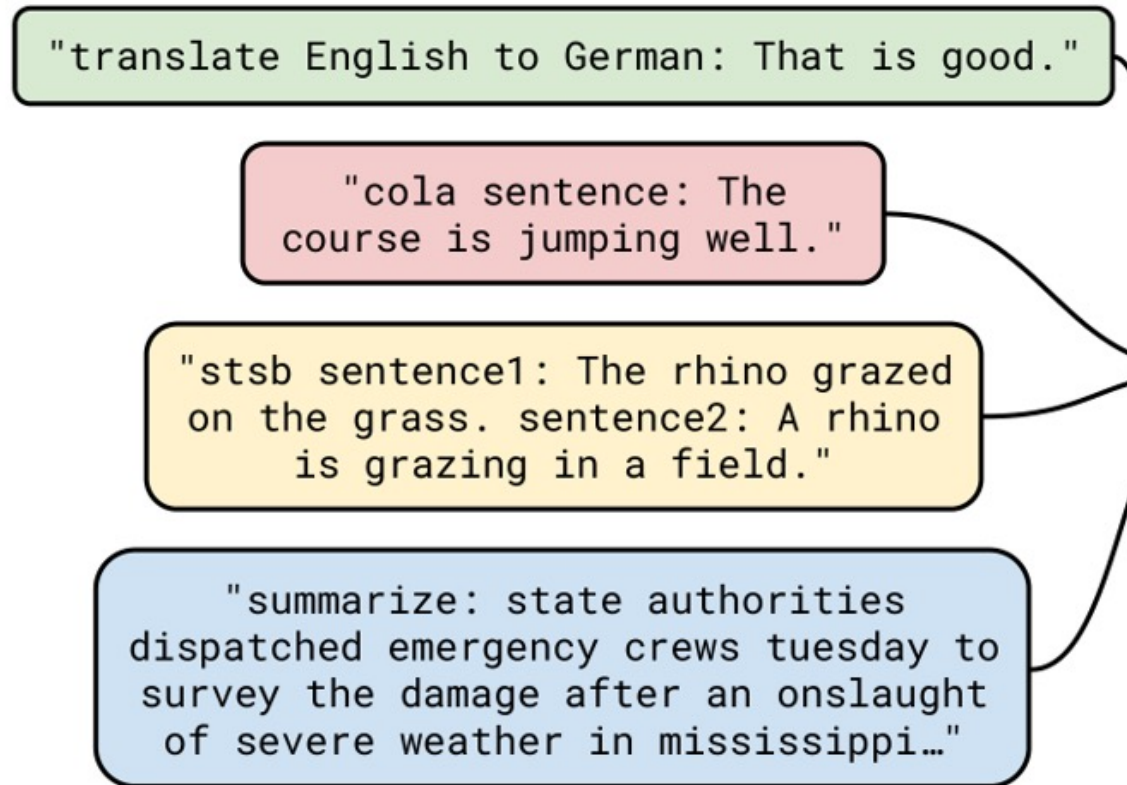
---

difficult to compare different algorithms, tease apart the effects of new contributions, and understand the space of existing methods for transfer learning.

# How?

---

unified framework that  
converts all text-based  
language problems into a  
text-to-text format



# Tasks

---

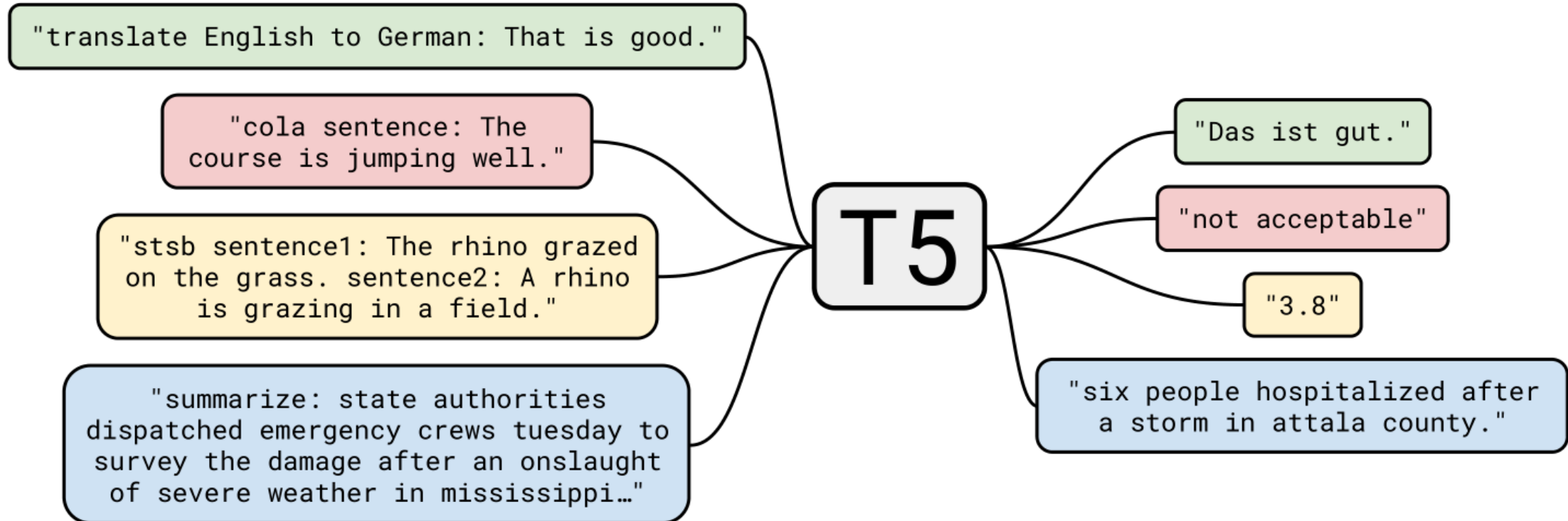
- Translation: easy to understand
- Cola sentence: Corpus of Linguistic Acceptability (CoLA) annotated for acceptability
- Stsb sentence1 sentence2 This dataset provides pairs of sentences and a score of their similarity
- Summarize: easy to understand

# All Tasks

- Sentence acceptability judgment (CoLA ([Warstadt et al., 2018](#)))
- Sentiment analysis (SST-2 ([Socher et al., 2013](#)))
- Paraphrasing/sentence similarity (MRPC ([Dolan and Brockett, 2005](#)), STS-B ([Cer et al., 2017](#)), QQP ([Iyer et al., 2017](#)))
- Natural language inference (MNLI ([Williams et al., 2017](#)), QNLI ([Rajpurkar et al., 2016](#)), RTE ([Dagan et al., 2005](#)), CB ([De Marneff et al., 2019](#)))
- Coreference resolution (WNLI and WSC ([Levesque et al., 2012](#)))
- Sentence completion (COPA ([Roemmele et al., 2011](#)))
- Word sense disambiguation (WIC ([Pilehvar and Camacho-Collados, 2018](#)))
- Question answering (MultiRC ([Khashabi et al., 2018](#)), ReCoRD ([Zhang et al., 2018](#)), BoolQ ([Clark et al., 2019](#)))



# Text-to-Text Transfer Transformer (T5)





# Objectives

---

- unsupervised objective:
  - a basic language modeling objective (causal language modeling objective)
    - **The task of predicting the token after a sequence of tokens is known as causal language modeling.**
  - baseline denoising objective (also called “masked language modeling” )
    - In a denoising objective, the model is trained to predict missing or otherwise corrupted tokens in the input.

# Benefits of T5

---

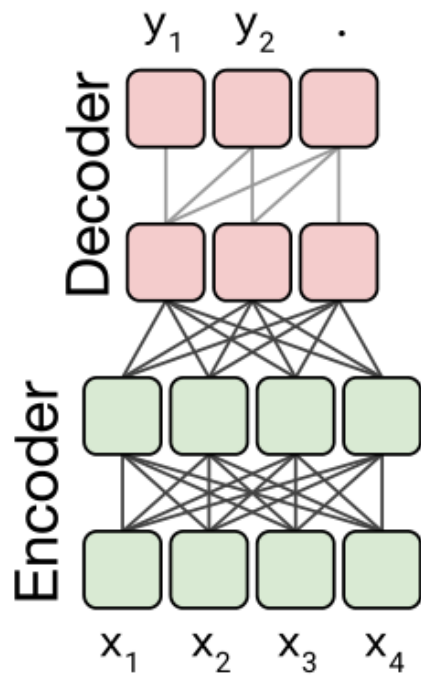
- This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey.
- This idea is also mentioned in GPT2 :” When a large language model is trained on a sufficiently large and diverse dataset it is able to perform well across many domains and datasets”
- For example, automatic summarization is done by feeding in a document followed by the text “TL;DR:” (short for “too long, didn’ t read” , a common abbreviation) and then the summary is predicted via autoregressive decoding

# Difference Between GPT2 and T5

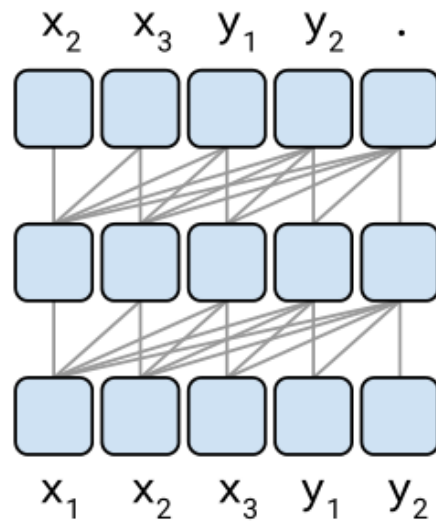
---

- T5 mainly considers models that explicitly process an input with an encoder before generating an output with a separate decoder
- T5 focuses on transfer learning rather than zero-shot learning(gpt2)

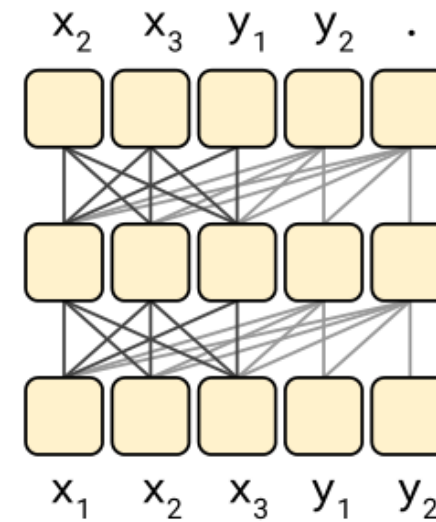
# Proposed Model Structures



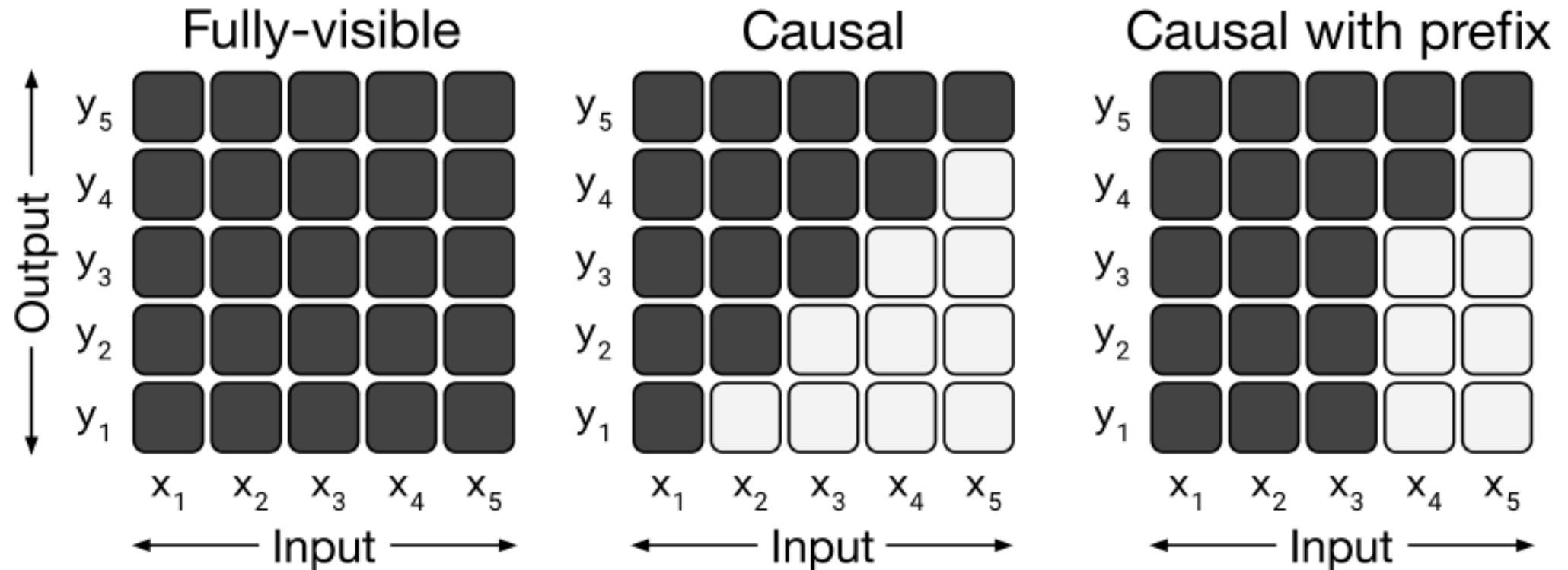
Language model



Prefix LM



# Matrices representing different attention mask patterns



# Annotations

---

- Wherever a baseline configuration appears, we will mark it with a (as in the first row of Table 1). We also will **boldface** any score that is within two standard deviations of the maximum (best) in a given experiment. ★

# Model Structure Result

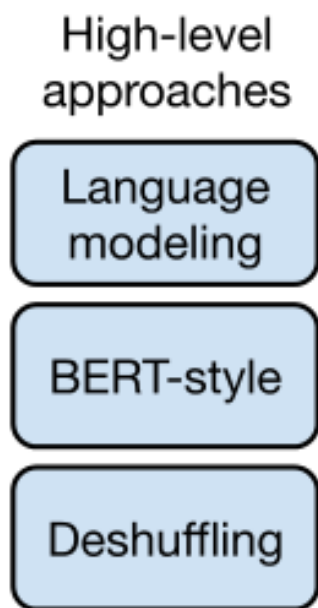
Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	$M$	<b>83.28</b>	<b>19.24</b>	<b>80.88</b>	<b>71.36</b>	<b>26.98</b>	<b>39.82</b>	<b>27.65</b>
Enc-dec, shared	Denoising	$P$	$M$	82.81	18.78	<b>80.63</b>	<b>70.73</b>	26.72	39.03	<b>27.46</b>
Enc-dec, 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



# Structure Difference

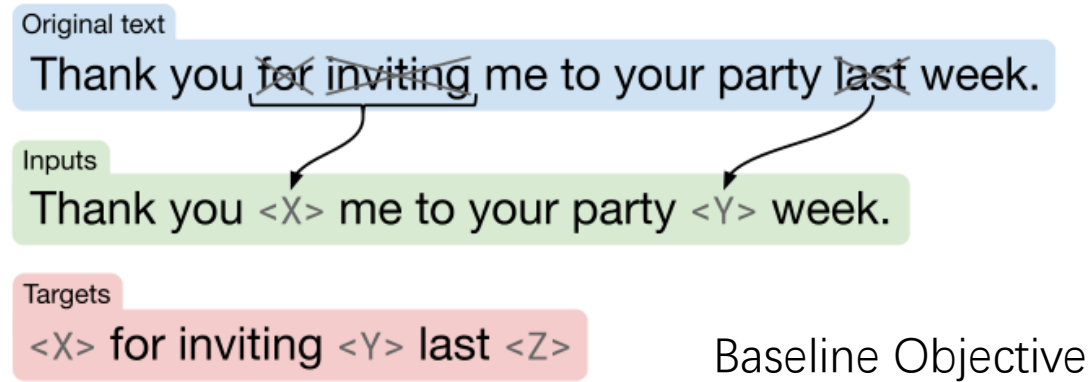
- T5 is roughly equivalent to the original Transformer proposed by Vaswani et al. (2017)
- Removing the Layer Norm bias,
- placing the layer normalization outside the residual path,
- Using a different position embedding scheme.

# A Flow Chart of T5 Exploration of Unsupervised Objectives



# High-Level Approaches

- Prefix language modeling: left to right
- BERT-style: mask and recover
- Deshuffling : shuffle the input and recover



Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style <a href="#">Devlin et al. (2018)</a>	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)

different

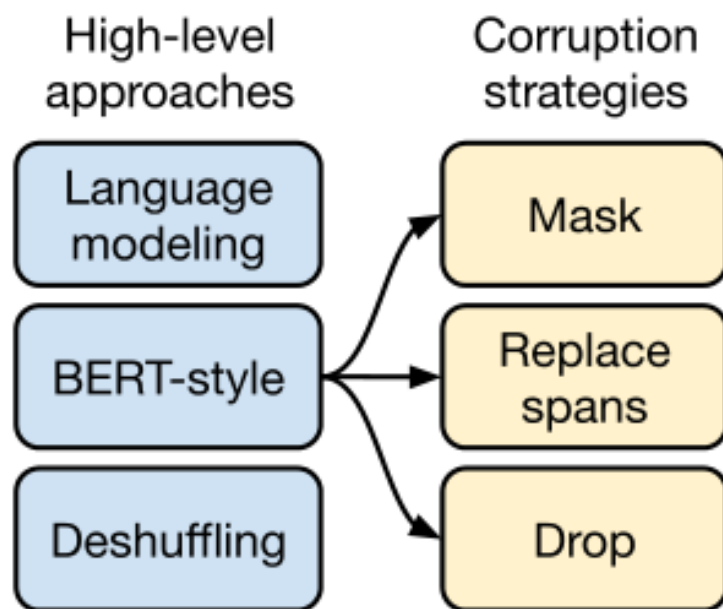
We talked about it in last class

# The performance of these three objectives

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

Table 4: Performance of the three disparate pre-training objectives described in Section 3.3.1.

# A Flow Chart of T5 Exploration of Unsupervised Objectives



# Corruption Strategies

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style <a href="#">Devlin et al. (2018)</a>	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 <a href="#">Song et al. (2019)</a>	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>

## Simplifying the BERT Objective

consecutive spans of dropped-out tokens are replaced by a single sentinel token. Each sentinel token is assigned a token ID that is unique to the sequence.

Our choices to mask consecutive spans of tokens and only predict dropped-out tokens were made to reduce the computational cost of pre-training.

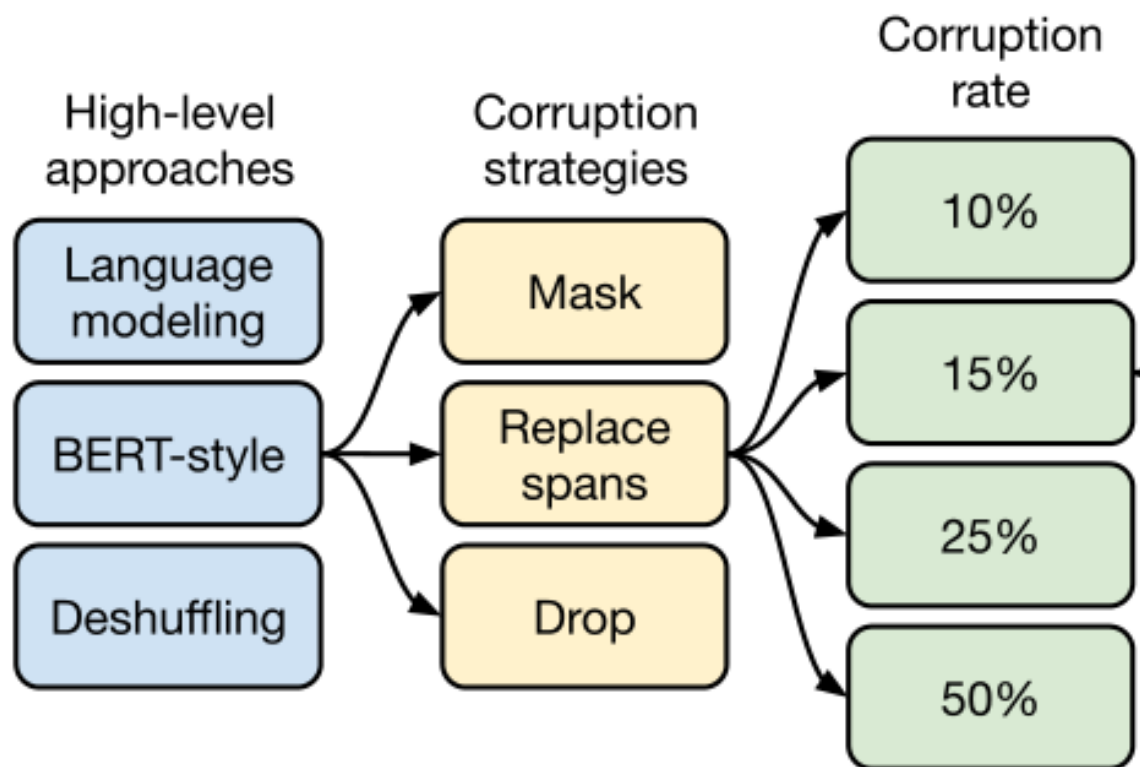
# The performance

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

Table 5: Comparison of variants of the BERT-style pre-training objective. In the first two variants, the model is trained to reconstruct the original uncorrupted text segment. In the latter two, the model only predicts the sequence of corrupted tokens.



# A Flow Chart of T5 Exploration of Unsupervised Objectives



# Varying the Corruption Rate

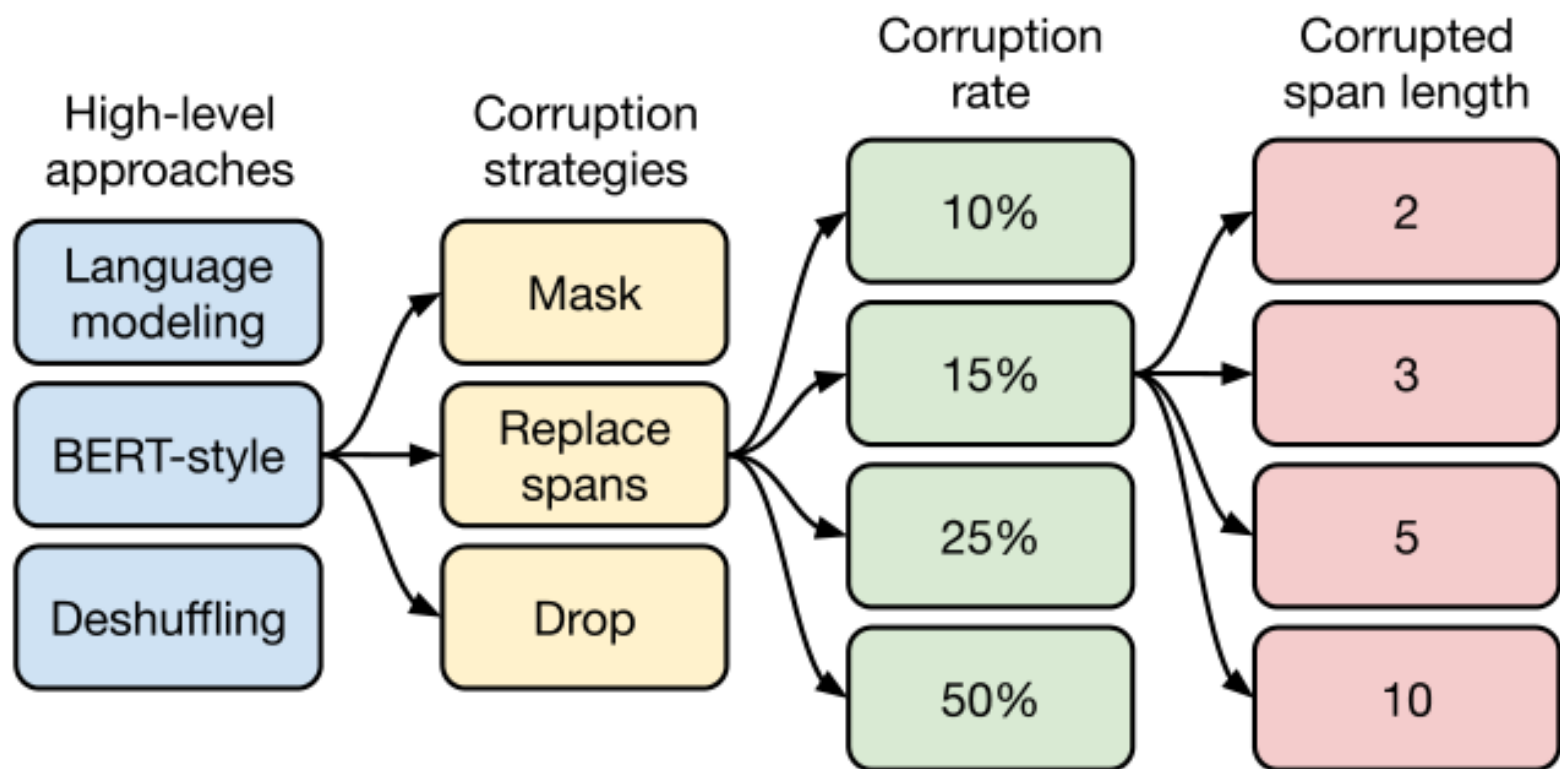
Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	<b>82.82</b>	19.00	<b>80.38</b>	69.55	<b>26.87</b>	39.28	<b>27.44</b>
★ 15%	<b>83.28</b>	19.24	<b>80.88</b>	<b>71.36</b>	<b>26.98</b>	<b>39.82</b>	<b>27.65</b>
25%	<b>83.00</b>	<b>19.54</b>	<b>80.96</b>	70.48	<b>27.04</b>	<b>39.83</b>	<b>27.47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

Table 6: Performance of the i.i.d. corruption objective with different corruption rates.

The training data generator chooses 15% of the token positions at random for prediction. If the  $i$ -th token is chosen, we replace the  $i$ -th token with (1) the [MASK] token 80% of the time (2) a random token 10% of the time (3) the unchanged  $i$ -th token 10% of the time. Then,  $T_i$  will be used to predict the original token with cross entropy loss. We compare variations of this procedure in Appendix C.2.

Do You find the number familiar?  
BERT

# A Flow Chart of T5 Exploration of Unsupervised Objectives



# Corrupting Spans

- For example, if we are processing a sequence of 500 tokens and we have specified that 15% of tokens should be corrupted and that there should be 25 total spans, then the total number of corrupted tokens would be
- $500 \times 0.15 = 75$
- and the average span length would be
- $75 / 25 = 3$

# Corrupting Spans

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

T5

Encoder-Decoder Structure

BERT-style Corruption  
Method

Replace Span corruption  
strategy

15 % corruption rate

Corrupted span of length 3



# Dataset


- Colossal Clean Crawled Corpus (C4)
- C4 was designed to be able to create extremely large pre-training data sets.
- The access to so much data allows us to pre-train our models without repeating examples.





# Question

Whether repeating examples during pre-training would be helpful or harmful to downstream performance?

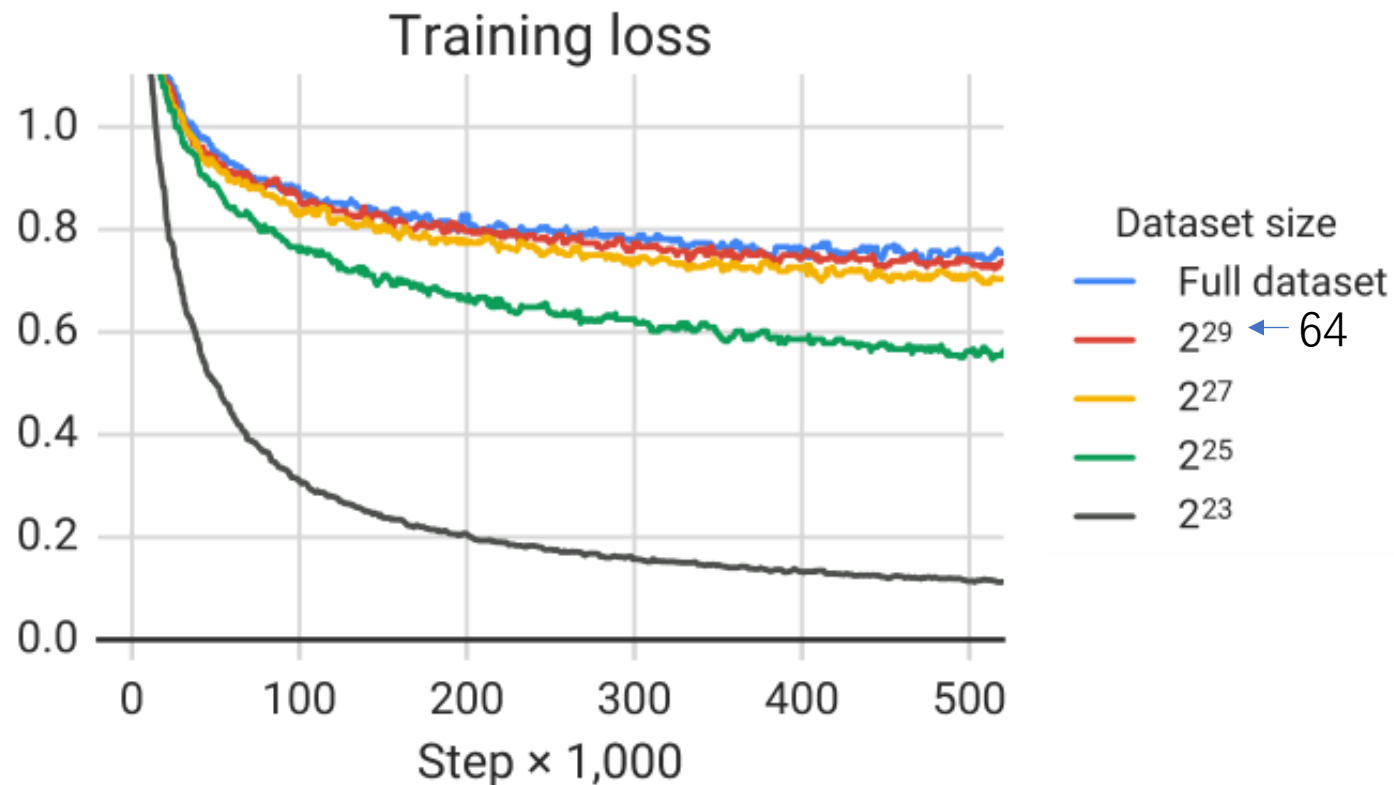


# Measuring the effect of repeating data during pre-training

	Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
$2^{35}$ ← ★	Full data set	0	<b>83.28</b>	<b>19.24</b>	<b>80.88</b>	<b>71.36</b>	<b>26.98</b>	<b>39.82</b>	<b>27.65</b>
	$2^{29}$	64	<b>82.87</b>	<b>19.19</b>	<b>80.97</b>	<b>72.03</b>	<b>26.83</b>	<b>39.74</b>	<b>27.63</b>
$2^{35}$ ←	$2^{27}$	256	82.62	<b>19.20</b>	79.78	69.97	<b>27.02</b>	<b>39.71</b>	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

Table 9: Measuring the effect of repeating data during pre-training. In these experiments, we only use the first  $N$  tokens from C4 (with varying values of  $N$  shown in the first column) but still pre-train over  $2^{35}$  tokens. This results in the data set being repeated over the course of pre-training (with the number of repeats for each experiment shown in the second column), which may result in memorization (see Figure 6).

# Evidence of the model begins to memorize the pre-training data set.



These effects are limited when the pre-training data set is repeated only 64 times. →  
Some amount of repetition of pre-training data might not be harmful.

However, given that additional pre-training can be beneficial and that obtaining additional unlabeled data is cheap and easy, we suggest using large pre-training data sets whenever possible.

“We also note that this effect may be more pronounced for larger model sizes, i.e. a bigger model may be more prone to overfitting to a smaller pre-training data set.”

# Training

---



# Training Strategy

---

- “adapter layers”
  - Adapter layers are additional dense-ReLU-dense blocks that are added after each of the preexisting feed-forward networks in each block of the Transformer.
- “gradual unfreezing”
  - more and more of the model’s parameters are fine-tuned over time.



# Multi-task Learning

---

- We relax this goal somewhat and instead investigate methods for training on multiple tasks at once in order to eventually produce separate parameter settings that perform well on each individual task.
- Try to compare this unsupervised learning result and supervised learning result.
  - Equal mixing
  - Examples-proportional mixing
  - Temperature-scaled mixing

# Multi-task Learning

---

“The primary concern in multi-task learning is setting the proportion of each task to train on. We ultimately did not find a strategy for setting mixing proportions that matched the performance of the basic approach of unsupervised pre-training followed by supervised fine-tuning. However, we found that fine-tuning after pre-training on a mixture of tasks produced comparable performance to unsupervised pre-training.”

--page 42



# Scaling

- Using a bigger model,
- training the model for more steps
- ensembling.

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	<b>86.18</b>	19.66	<b>84.18</b>	77.18	27.52	<b>41.03</b>	28.19
4× size, 1× training steps	<b>85.91</b>	19.73	<b>83.86</b>	<b>78.04</b>	27.47	40.71	28.10
4× ensembled	84.77	<b>20.10</b>	83.09	71.74	<b>28.05</b>	40.53	<b>28.57</b>
4× ensembled, fine-tune only	84.05	19.57	82.36	71.55	27.55	40.22	28.09

Training  
detail (In  
case  
someone  
asks)

---

Standard maximum likelihood, i.e.  
using teacher forcing (Williams and  
Zipser, 1989)

---

Cross-entropy loss

---

AdaFactor (Shazeer and Stern, 2018).

# Citations

---

- *Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. 21, 1, Article 140 (January 2020), 67 pages.*

# Prepared Question 1:

---

- STS-B is a regression task where the goal is to predict a similarity score between 1 and 5.
- How could we translate this task to text?

# Prepared Answer 1:

---

- We found that most of these scores were annotated in increments of 0.2, so we simply rounded any score to the nearest increment of 0.2 and converted the result to a literal string representation of the number (e.g. the floating-point value 2.57 would be mapped to the string “2.6” ).
- At test time, if the model outputs a string corresponding to a number between 1 and 5, we convert it to a floating-point value; otherwise, we treat the model’s prediction as incorrect. This effectively recasts the STS-B regression problem as a 21-class classification problem.

# Prepared Question 2:

---

- How can we use “Cross Entropy” as the loss for the text generating task? Isn't word the string?

# Prepared Answer 2:

---

- “Thank you for inviting me to your party last week .” Note that all of our objectives process tokenized text. For this particular sentence, all words were mapped to a single token by our vocabulary. We write
- (original text) as a target to denote that the model is tasked with reconstructing the entire input text.  $\langle M \rangle$  denotes a shared mask token and  $\langle X \rangle$  ,  $\langle Y \rangle$  , and  $\langle Z \rangle$  denote sentinel tokens that are assigned unique token IDs. The BERT-style objective (second row) includes a corruption where some tokens are replaced by a random token ID.

# Prepared Question 3:

---

- How do the model know when to stop, like how many words there should be in the answer?



# Appendix: Fine-tune Result

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	<b>83.28</b>	<b>19.24</b>	<b>80.88</b>	<b>71.36</b>	<b>26.98</b>	<b>39.82</b>	<b>27.65</b>
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	<b>70.79</b>	26.71	39.02	26.93

Table 10: Comparison of different alternative fine-tuning methods that only update a subset of the model’s parameters. For adapter layers,  $d$  refers to the inner dimensionality of the adapters.