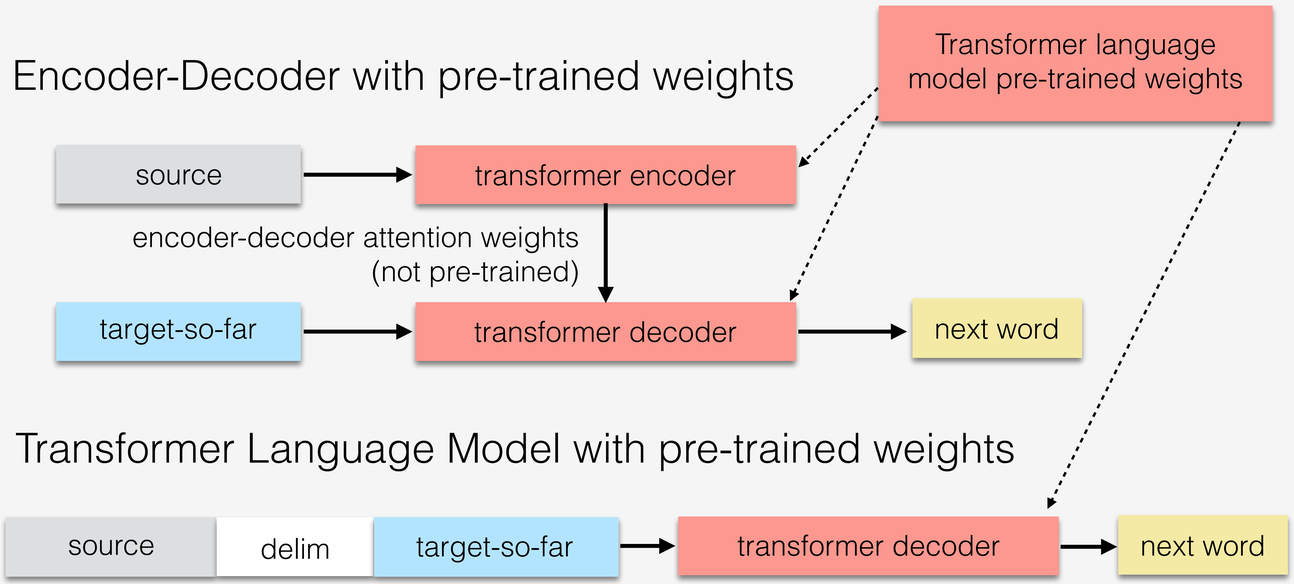
Sample Efficient Text Summarization Using a Single Pre-Trained Transformer

Urvashi Khandelwal[1], Kevin Clark[1], Dan Jurafsky[1], Lukasz Kaiser[2]
Stanford University[1], Google Brain[2]

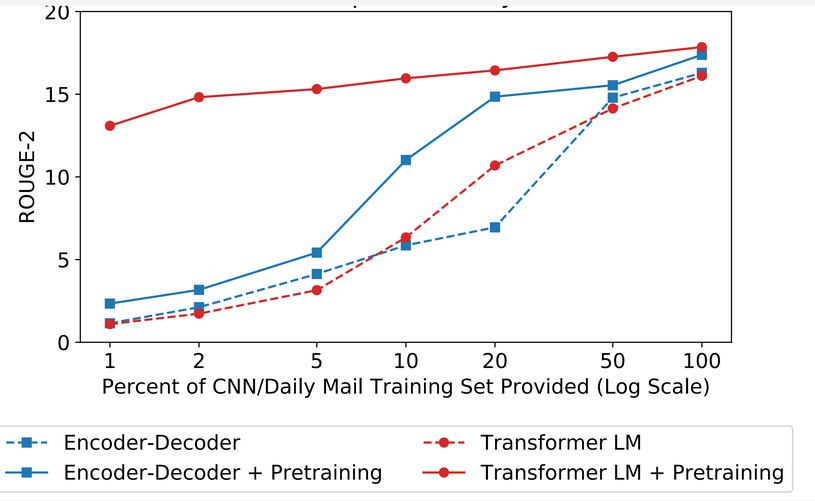
Given the pre-trained weights of a Transformer language model, replacing the Transformer encoder-decoder with a Transformer decoder only model for fine-tuning on text summarization is highly sample efficient since there are no more non-pre-trained encoder-decoder attention weights.

Fine-tuning on Text Summarization



Simpler model, fewer parameters and no non-pre-trained attention weights

Sample Efficiency



Outputs from models fine-tuned on 1% data

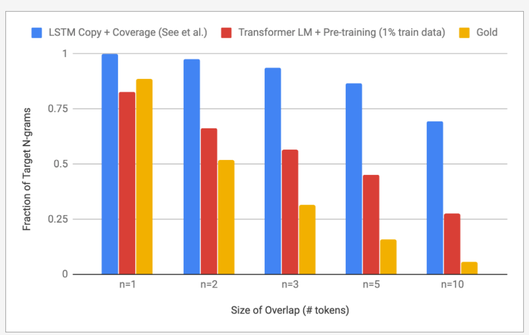
Ground truth: *A man in suburban Boston is selling snow online to customers in warmer states. For $89, he will ship 6 pounds of snow in an insulated Styrofoam box.*

Encoder-Decoder + Pre-training: *NEW: A snowfall of is forecast for New England. NEW: The Massachusetts-based company hopes to sell more than 30,000 bottles of snow. The company says it will use snow from as far as Canada.*

Transformer LM + Pre-training: *Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says. But not if you live in New England or surrounding states.*

The Transformer LM fine-tuned on 1% data successfully extracts facts from the source, while the encoder-decoder hallucinates facts/ generates gibberish.

How much of the sample efficiency is due to copying?



This Transformer LM is more abstractive (copies less) than the Pointer-Generator model, likely due to lack of the copy mechanism.
A drawback of this abstractiveness is subtle factual inaccuracies!

References
[1] Peter J Liu, et al. 2018. *Generating Wikipedia by summarizing long sequences.* In ICLR.
[2] Alec Radford et al. 2018. *Improving language understanding by generative pre-training.*
[3] Abigail See et al. 2017. *Get to the point: Summarization with pointer-generator networks.* In ACL.

Acknowledgements
This work was started and, in part, carried out during the first author's internship at Google Brain. We gratefully acknowledge support of the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF15-1-0462 and the NSF via grant IIS-1514268. Kevin is supported by the Google PhD Fellowship.

Code: https://github.com/tensorflow/tensor2tensor