Leveraging Pre-trained Checkpoints for Sequence Generation Tasks

Sascha Rothe
Google Research
rothe@google.com

Shashi Narayan
Google Research
shashinarayan@google.com

Aliaksei Severyn
Google Research
severyn@google.com

Abstract

Unsupervised pre-training of large neural models has recently revolutionized Natural Language Processing. Warm-starting from the publicly released checkpoints, NLP practitioners have pushed the state-of-the-art on multiple benchmarks while saving significant amounts of compute time. So far the focus has been mainly on the Natural Language Understanding tasks. In this paper, we present an extensive empirical study on the utility of initializing large Transformer-based sequence-to-sequence models with the publicly available pre-trained BERT and GPT-2 checkpoints for sequence generation. We have run over 300 experiments spending thousands of TPU hours to find the recipe that works best and demonstrate that it results in new state-of-the-art results on Machine Translation, Summarization, Sentence Splitting and Sentence Fusion.

1 Introduction

Unsupervised and self-supervised pre-training methods, such as ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), and more recently BERT (Devlin et al., 2018), GPT and GPT-2 (Radford et al., 2018, 2019) and XLNet (Yang et al., 2019) have established a qualitatively new level of baseline performance for many widely used Natural Language Understanding (NLU) benchmarks including some of the most popular Glue (Williams et al., 2017) and Squad (Rajpurkar et al., 2018).

The most appealing part about this massive shift towards using large architectures pre-trained on large collections of texts is that the pre-trained checkpoints along with the inference code are made freely available. This saves hundreds of TPU/GPU hours as warm-starting a model from a pre-trained checkpoint typically requires orders of magnitude fewer fine-tuning steps while delivering significant performance boosts. More importantly, the ability to bootstrap from a state-of-the-art performing model such as BERT (Devlin et al., 2018) motivates the community to greatly speed up the progress towards developing better and easily reusable NLU systems. For example, as of July 2019, 66 out of the top 84 systems on the Squad leaderboard[1] build on top of BERT.

While we continue to observe an increasing number of papers building on top of BERT and/or GPT (Generative Pre-Training) models reporting encouraging improvements on Glue, Squad, and other similar benchmarks, very little attention has been paid to using these pre-trained models to warm-start sequence-to-sequence (seq2seq) models. It has been argued that the pre-training objective used by BERT is not well suited for tasks that require decoding texts, e.g., conditional text generation in machine translation and summarization (Yang et al., 2019). In contrast, GPT and XLNet do not exhibit such limitations as they employ a language model objective during pre-training. Nevertheless, it remains unclear to what extent employing such large models pre-trained on large collections of text can be beneficial to warm-start sequence-to-sequence generation models.

In this paper we aim to provide an empirical answer to the following research question: *what is the best way to leverage publicly available pre-trained checkpoints from BERT and GPT-2 for warm-starting seq2seq models?* For example, one could imagine using BERT checkpoint to initialize the encoder and choosing GPT-2 model as the decoder or vice versa. Taking the full Cartesian product produces lots of model combinations that are expensive and time-consuming to train. One of the main contribution of this paper is that we rigorously experiment with a large number of different settings to combine BERT and GPT pre-trained checkpoints in a large Transformer sequence-2-sequence model. To put these models at test, we report the results on three canonical conditional text generation tasks of increasing complexity:

[1] https://rajpurkar.github.io/SQuAD-explorer/

sentence fusion (DiscoFuse (Geva et al., 2019)) and sentence splitting (WikiSplit (Botha et al., 2018)), WMT14 En↔De machine translation using most common eval sets: newstest2014 and newstest2016, and abstractive summarization using three datasets: Gigaword (Napoles et al., 2012), CNN and DailyMail (Hermann et al., 2015) and BBC extreme (Narayan et al., 2018).

Overall, we have run over 300 experiments spending thousands of TPU v3 hours to determine which approach works best. We believe that researchers and NLP practitioners will derive actionable insights from our findings when tackling various seq2seq tasks.

In all of the tasks, we report significant improvements over randomly initialized models demonstrating the benefit of leveraging unsupervised pre-trained models. More importantly, this simple strategy results in several new state-of-the-art results.

2 Models

Our blueprint is a sequence-2-sequence architecture where both encoder and decoder are composed of Transformer layers. Note that, in order to properly warm-start a model from a public BERT and/or GPT-2 checkpoint, we use the reference BERT and GPT-2 Transformer layer implementations where appropriate, as they differ slightly from each other (with respect to activation function, dropout, etc.) as well as from the canonical Transformer layer by Vaswani et al. (2017). All models have 12 layers, a hidden size of 768, filter size of 3072, and 12 attention heads unless mentioned otherwise.

Depending on the experiment, we use one of the following publicly available BERT checkpoints: BERT-Base Cased, BERT-Base Uncased, BERT-Base Multilingual Cased (Devlin et al., 2018).[2] The first two checkpoints have a vocabulary size of around ∼30k wordpieces, whereas the multilingual checkpoint has a much larger vocabulary size of ∼110k. BERT also trains positional embeddings for up to 512 positions. This will be the maximum input and output length in all of our experiments.

We also use the publicly available GPT-2 checkpoint with 12 layers (Radford et al., 2019).[3] Note that, while this checkpoint is frequently called 117M, which suggests the same number of parameters, we count 124M parameters in the checkpoint available to us. This is the smallest architecture they trained, and the number of layers, hidden size, and filter size are comparable to BERT-Base. The model was trained mainly on English data but does contain some foreign language. The vocabulary size is ∼50k.

While GPT-2 has positional embeddings for up to 1024 position, we only use the first 512 to make the results comparable with BERT.

We experiment with the following model combinations:

RND2RND A BERT Transformer encoder-decoder architecture in which all the weights are initialized randomly.

BERT2RND A BERT-initialized encoder paired with a randomly initialized decoder. Note that both encoder and decoder share the embeddings initialized from a checkpoint.

RND2BERT A randomly initialized encoder paired with a BERT-initialized decoder. We mask the bidirectional self-attention mechanism of BERT to look only at the left context, to be able to perform autoregressive decoding.

BERT2BERT A BERT-initialized encoder paired with a BERT-initialized decoder. All weights are initialized from a public BERT checkpoint. The only variable that is initialized randomly is the encoder-decoder attention.

BERTSHARE Like BERT2BERT, but the parameters between encoder and decoder are shared. This greatly reduces the memory footprint of the model (136M vs. 221M parameters). Additionally, we experimented with a layer-wise attention mechanism (He et al., 2018), but got nearly identical numbers on most tasks.

RND2GPT A randomly initialized encoder paired with a GPT-2 decoder. We warm start the decoder and the embeddings with a public GPT-2 checkpoint.

BERT2GPT A BERT-compatible encoder paired with a GPT-2-compatible decoder. We warm start both sides with the two public checkpoints. We use the BERT vocabulary for the input and the GPT-2 vocabulary for the output.

GPT A decoder-only architecture. We treat the input as a conditioning prefix of a language model. The decoder is warm started with a public GPT-2 checkpoint. Similarly to BERTSHARE, the memory footprint of this model is smaller compared to an encoder-decoder setup (124M parameters).

[2] https://github.com/google-research/bert

[3] https://github.com/openai/gpt-2

	total	embed.	init.	random
RND2RND	221M	23M	0	221M
BERT2RND	221M	23M	109M	112M
RND2BERT	221M	23M	109M	112M
BERTSHARE	136M	23M	109M	27M
BERT2BERT	221M	23M	195M	26M
RND2GPT	238M	39M	124M	114M
GPT	124M	39M	124M	0
BERT2GPT	260M	62M	234M	26M

Table 1: The number of total trainable parameters, the number of embedding parameters, the number of parameters initialized from the checkpoint vs. randomly.

DiscoFuse	100% Exact	100% SARI	10% SARI	1% SARI
(Geva et al., 2019)	51.1	84.5	–	–
RND2RND	58.3	86.9	81.5	69.3
BERT2RND	**63.9**	**89.3**	**86.1**	80.3
RND2BERT	60.0	87.6	82.1	72.8
BERTSHARE	**63.9**	89.2	86.0	80.8
BERT2BERT	**63.9**	**89.3**	**86.1**	**81.2**
RND2GPT	57.6	86.5	81.4	70.6
GPT	60.4	88.0	82.9	74.5
BERT2GPT	61.5	88.4	84.1	70.2

Table 2: Results of different models and initialization techniques on DiscoFuse and subsampled training sets.

All models were trained using Adam with a learning rate of 0.05. We used a linear learning rate warmup with 40k steps, normalization by the square root of the hidden size, and a square root decay. We did not perform any tuning of these hyperparameters (except for Section 4.2). The batch size and the number of training steps will be reported for each task individually.

3 Experiments

This section reports the results on three canonical conditional text generation tasks of increasing complexity: sentence fusion with the DiscoFuse (Geva et al., 2019) dataset, sentence splitting with WikiSplit (Botha et al., 2018), WMT14 En↔De machine translation with the most common evaluation sets: newstest2014 and newstest2016, and abstractive summarization on three datasets: Gigaword (Napoles et al., 2012), CNN, and DailyMail (Hermann et al., 2015)

3.1 Sentence Fusion

Sentence Fusion is the problem of combining multiple sentences into a single coherent sentence. We use the balanced Wikipedia portion of the DiscoFuse dataset by Geva et al. (2019) for our experiments with 4.5M fusion examples in the training set. The evaluation set has 50k example. Due to the size of this evaluation set, even small changes are statistically significant. For this reason, we have solely chosen this dataset for additional experiments described at the end of the paper.

Training was done for 300k steps with a global batch size of 256. The input and output is padded to a length of 128, which covers 100% of the training, evaluation and test data. We report SARI[4] and the exact match accuracy. The results can be seen in Table 2. Previous state-of-the-art results by Geva et al. (2019) used the vanilla transformer model by Vaswani et al. (2017), with only 7 layers. All models with BERT-initialized encoder outperform the baseline by a large margin, with a SARI score of 89.3 compared to 86.9. To measure the effect on smaller training sets, we randomly subsample the training data down to 10% and 1%, i.e. 450k and 45k training examples, respectively. First we notice, that performance comparable to the baseline is achieved even when training on only 10% of the training data. Secondly models, with fewer parameters (BERTSHARE and GPT) perform slightly better on fewer training data. This is most visible when comparing BERT2GPT with GPT, where the former has to learn two embedding matrices, while the latter has only one, and all its variables are pre-trained (see Table 1).

3.2 Split and Rephrase

The reverse task of sentence fusion is the split-and-rephrase task, which requires rewriting a long sentence into two or more coherent short sentences. We use the WikiSplit dataset (Botha et al., 2018), which consists of 1M examples of sentence splits extracted from the Wikipedia edit history, and follow the training/test split suggested by the authors. Training was done for 300k steps with a global batch size of 256. The input and output is padded to a length of 128, which covers 100% of the training, evaluation and test data. As in Botha et al. (2018), we report corpus-level Bleu[5], the exact match accuracy, and SARI score. Previous state-of-the-art results by Botha et al. (2018) used

[4]SARI implementation is available at: https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/sari_hook.py

[5]We use NLTK v3.2.2 with case sensitive scoring to estimate Bleu scores.

	WikiSplit		
	Exact	SARI	Bleu
(Botha et al., 2018)	14.3	61.5	76.4
RND2RND	14.6	61.7	76.3
BERT2RND	15.9	63.1	76.9
RND2BERT	15.2	61.8	76.5
BERTSHARE	**16.3**	**63.5**	**77.2**
BERT2BERT	15.6	63.2	77.0
RND2GPT	14.2	61.3	76.2
GPT	14.2	61.1	75.8
BERT2GPT	14.6	62.4	76.5

Table 3: Results of different models and initialization setups on WikiSplit.

a bi-directional LSTM with a copy mechanism (Aharoni and Goldberg, 2018). Analogous to the DiscoFuse task we observe that initializing the encoder with BERT improves the model the most (Table 3). The shared encoder-decoder setup of BERTSHARE outperforms all other setups.

3.3 Machine Translation

We test our setups on the most commonly used benchmark in machine translation – WMT 2014 English ↔ German task using newstest2014 and newstest2016 eval sets. We use the same hyper-parameter settings as in the previous experiments. We limit the input and decoding length to 128 tokens. Since the task involves one non-English language, we use the BERT-Base Multilingual Cased checkpoint to initialize the encoder and the decoder, where both share the same vocabulary. This checkpoint has been pre-trained on 108 languages using a multilingual Wikipedia dump. To support all these languages, the vocabulary is composed of 110k wordpieces. As the larger vocabulary increases the memory footprint, we used a global batch size of 256. Training was done for 30 epochs, which takes 2-3 days on a 4x4 TPU v3. We report uncased Bleu-4 scores.[6] Decoding was done with the beam size of 4 and the default value for the sentence length penalty $\alpha = 0.6$.

In Table 4 we first report the baseline scores for the Transformer model by Vaswani et al. (2017) and our Transformer implementation[7] with the same hyper-parameter setting (number of hidden layers, hidden layer size, number of heads, etc.), which obtains slightly higher scores. Note that here we use the 32k wordpiece vocabulary extracted from the WMT14 training set. Both models have 6 layers in both the encoder and the decoder.

The middle section of Table 4 reports the results for various initialization schemas using BERT and GPT-2 pre-trained checkpoints. Note that here all models have 12 layers in both the encoder and the decoder. For BERT models we use the 110k wordpiece vocabulary that comes with the public multilingual BERT checkpoint. First, we observe that initializing the model with the BERT checkpoint is most beneficial on the encoder side, which further supports previous observations (Yang et al., 2019) that the pre-training objective of BERT is not very well suited for sequence generation tasks. Furthermore, models initialized with the BERT checkpoint receive a significant boost: BERT2RND compared to the no-initialization RND2RND setup scores higher by +4 points on En→De and +3.6 points on De→En on newstest2014. Contrary to the WikiSplit and DiscoFuse task, sharing the encoder and decoder variables did not give an additional boost. This is most likely because a) model capacity is an important factor in MT and b) encoder and decoder have to deal with different grammar and vocabulary.

GPT-based models (RND2GPT, GPT, and BERT2GPT) do not perform nearly as well, especially when GPT is used as the decoder and the target language is German. This is because the public GPT model comes with an English vocabulary and has been pre-trained mainly on English text.

Customized BERT checkpoint. Additionally, we observe that our implementation of the baseline Transformer, as well as RND2RND setup which uses no initialization, perform weaker on newstest2014 compared to the Transformer baselines we report in the top section of Table 4. We conjecture that the differences might be due to the larger 110k wordpiece vocabulary trained to handle 104 languages from Wikipedia dump which is suboptimal for WMT14 data and leads to inferior results. To verify this conjecture, we perform the following experiment: we use the 32k wordpiece vocabulary extracted from the WMT14 En ↔ De training

[6] We use the script from the Tensorflow Official Transformer implementation https://github.com/tensorflow/models/tree/master/official/transformer Note that, differently from the tensor2tensor/utils/get_ende_bleu.sh used by (Vaswani et al., 2017) this script does not split noun compounds, but we normalize utf-8 quotes to ascii quotes as we noted that our pre-processed training set contains only ascii quotes.

[7] We use Transformer layers from the official BERT implementation which have small differences from (Vaswani et al., 2017).

	newstest2014		newstest2016	
	En→De	De→En	En→De	De→En
Baselines				
(Vaswani et al., 2017)	27.3	–	–	–
Transformer (ours)	28.1	31.4	33.5	37.9
KERMIT (Chan et al.)	28.7	31.4	–	–
(Shaw et al., 2018)	29.2	–	–	–
(Edunov et al., 2018)*	**35.0** (33.8)	–	–	–
Initialized with public checkpoints and vocabulary				
Transformer (ours)	23.7	26.6	31.6	35.8
RND2RND	26.0	29.1	32.4	36.7
BERT2RND	30.1	32.7	34.4	39.6
RND2BERT	27.2	30.4	33.2	37.5
BERTSHARE	29.6	32.6	34.4	39.6
BERT2BERT	30.1	32.7	34.6	39.3
RND2GPT	19.6	23.2	24.2	28.5
GPT	16.4	21.5	22.4	27.7
BERT2GPT	23.2	31.4	28.1	37.0
Initialized with a custom BERT checkpoint and vocabulary				
BERT2RND	**30.6**	33.5	35.1	**40.2**
BERTSHARE	30.5	**33.6**	**35.5**	40.1

Table 4: Uncased Bleu-4 scores on WMT14 English ↔ German newstest2014 and newstest2016 test sets. Models in the middle section use the 110k wordpiece vocabulary that comes with the multilingual BERT checkpoint. In the bottom section we use the native 32k wordpiece vocabulary extracted from WMT14 train set and a BERT checkpoint pre-trained only on English and German subset of Wikipedia. * leveraging a large number of additional parallel sentence pairs obtained with back-translation; we include this score as a reference to the highest achieved result on newstest2014.

set (same as used in the top section of Table 4) and pre-train a BERT model on the English and German subset of the Wikipedia dump in the same way as the multilingual BERT checkpoint was obtained. We initialize our best-performing setups, BERT2RND and BERTSHARE, with this checkpoint (bottom section of Table 4). This provides a further +0.5 (En ↔ De) and +0.8 (De ↔ En) improvements on newstest2014 and +1.1 and +0.7 on newstest2016, yielding an overall very strong performance on the challenging WMT14 task.

Note that Edunov et al. (2018) report better results when they augment the training set with a massive amount of back-translated sentence pairs. To the best of our knowledge, among the approaches that only leverage parallel data from WMT14, our scores are state-of-the-art across both newstest2014 and newstest2016.

3.4 Abstractive Summarization

Document summarization is the task of producing a short version of a document while preserving its salient information content. We evaluate our setups on three different summarization datasets of varying characteristics: Gigaword (Napoles et al., 2012), CNN and DailyMail (Hermann et al., 2015), and BBC extreme (Narayan et al., 2018). The Gigaword dataset focuses on abstractive sentence summarization with a total of 3.8M sentence-summary training pairs. The other two datasets focus on single-document summarization: the CNN/DailyMail dataset consists of 287k document-summary pairs, whereas the BBC dataset consists of 204k document-summary pairs. The CNN/DailyMail summaries are in the form of bullet-point story highlights and exhibit a high degree of extraction, requiring the models to learn to copy from the source documents. The BBC summaries, on the other hand, are extreme, in that the documents are summarized into single-sentence summaries. These summaries demonstrate a high level of abstractiveness, and generating them automatically requires document-level inference, abstraction, and paraphrasing.

In all three cases, we did not anonymize entities. We worked on the original cased versions of CNN/DailyMail and BBC datasets. For Gigaword we used the lowercased version to match the requirements of the publicly available lowercased test set. During training and at inference time, the input documents were truncated to 512 tokens for CNN/DailyMail and BBC, and to 128 tokens for Gigaword. Similarly, the length of the summaries was limited to 128 tokens for CNN/DailyMail, 64

	Gigaword			CNN/Dailymail			BBC XSum		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
PtGen	–	–	–	39.53	17.28	36.38	29.70	9.21	23.24
ConvS2S	35.88	17.48	33.29	–	–	–	31.89	11.54	25.75
MMN	–	–	–	–	–	–	32.00	12.10	26.0
Bottom-Up	–	–	–	41.22	18.68	38.34	–	–	–
GPT-2	–	–	–	29.34	8.27	26.58	–	–	–
MASS	*38.73*	19.71	*35.96*	–	–	–	–	–	–
TransLM	–	–	–	39.65	17.74	36.85	–	–	–
UniLM	–	–	–	*43.47*	*20.30*	*40.63*	–	–	–
RND2RND	36.94	18.71	34.45	35.77	14.00	32.96	30.90	10.23	24.24
BERT2RND	37.71	19.26	35.26	38.74	17.76	35.95	38.42	15.83	30.80
RND2BERT	37.01	18.91	34.51	36.65	15.55	33.97	32.44	11.52	25.65
BERTSHARE *	**38.13**	***19.81***	**35.62**	**39.25**	**18.09**	**36.45**	***38.52***	***16.12***	***31.13***
BERT2BERT	38.01	19.68	35.58	39.02	17.84	36.29	37.53	15.24	30.05
RND2GPT	36.21	18.39	33.83	32.08	8.81	29.03	28.48	8.77	22.30
GPT	36.04	18.44	33.67	37.26	15.83	34.47	22.21	4.89	16.69
BERT2GPT	36.77	18.23	34.24	25.20	4.96	22.99	27.79	8.37	21.91

Table 5: Summarization results of different models and their initialization setups. We compare our setups (the bottom block) against both non-pre-trained (the top block) and pre-trained (the middle block) models on various datasets: PtGen (See et al., 2017), ConvS2S (Gehring et al., 2017), MMN (Kim et al., 2018), Bottom-Up (Gehrmann et al., 2018), GPT-2 (Radford et al., 2019), MASS (Song et al., 2019), TransLM (Khandelwal et al., 2019) and UniLM (Dong et al., 2019). PtGen and ConvS2S results on the BBC dataset are taken from Narayan et al. (2018). Our best results are **boldfaced** and the best results on the datasets are *italicized*. See the text for more details. * For CNN/DailyMail, we report BERTSHARE with a layer-wise attention mechanism (He et al., 2018), see details in Section 2.

for BBC, and 32 for Gigaword. We used a global batch size of 128 document-summary pairs for CNN/DailyMail and BBC, and 256 document-summary pairs for Gigaword. We adapted to different number of training steps depending on the training data sizes. Models were trained for 500k, 300k and 200k steps for the Gigaword, CNN/DailyMail and BBC summarization datasets respectively. In all cases we used the standard publicly available test sets; these consists of 1951 instances for Gigaword, 11490 for CNN/DailyMail and 11334 for BBC. We report on the Rouge F_1 scores (Lin and Hovy, 2003); in particular, we report on Rouge-1 and Rouge-2 for informativeness and Rouge-L for fluency. The results from our summarization experiments are presented in Table 5.

Document understanding with BERT. All BERT encoder based setups (i.e., BERT2RND, BERTSHARE, and BERT2BERT) outperform the baseline RND2RND by a large margin. The improvements of the RND2BERT setup, where only the decoder is initialized, are not significant. These results overall validate the significance of document representation in the encoder-decoder framework for summarization. On the BBC extreme summarization in particular, these three models achieve on average +6.42 point improvement in Rouge-L compared to the RND2RND setup. Our results demonstrate that the models with better document representations are better in generating extreme summaries that require document-level inference and abstraction. For the extractive highlights in the CNN/DailyMail dataset, these models show an improvement of +3.3 Rouge-L points over the RND2RND baseline. For Gigaword, where the input is a single sentence, the improvements are minimal (average of +0.99 Rouge-L points). Interestingly, for CNN/DailyMail, BERTSHARE with a layer-wise attention mechanism (He et al., 2018) performed better than only top layer attention. The layerwise attention probably better captures the lexical information required to generate extractive summaries of the CNN/DailyMail documents.

The BERTSHARE setup with shared encoder and decoder parameters achieves better performance than BERT2BERT where the encoder and decoder parameters are not shared, on all three datasets. The gains are larger on the BBC dataset than on the Gigaword and CNN/DailyMail datasets. This is probably because the BBC summary sentences follow a distribution that is similar to that of the sentences in the document, whereas this is not necessarily the case for the Gigaword headlines and the CNN/DailyMail bullet-point highlights.

Summarization with GPT checkpoints. GPT (decoder-only) performs better than RND2GPT

or BERT2GPT (encoder-decoder models) by a large margin for generating CNN/Daily extracts, but poorer for generating BBC abstracts. We hypothesize that the encoder-decoder architecture where the input document is modeled separately is better equipped for document level abstraction than the decoder-only architectures where the input document is a conditioning prefix of a language model. Initialization with different checkpoints, e.g., encoder with BERT and decoder with GPT in BERT2GPT, is not effective, especially for the smaller training sets. We observe that the performance of BERT2GPT is inferior to the performance of RND2GPT on the CNN/DailyMail and BBC datasets with 287k and 204k training instances. However, this is not the case with the Gigaword dataset, which has 3.8M training instances; BERT2GPT performs better than RND2GPT on Rouge-1 and Rouge-L scores.

The BERTSHARE setup performs best and is on par with the current state-of-the-art MASS model (Song et al., 2019) on the Gigaword dataset. The MASS model has an advantage of pre-training encoder-decoder attention from scratch, our proposed models use the publicly available pre-trained checkpoints and only fine-tune on the target task. It is not obvious how the masked seq2seq pre-training objective for sentence generation in the MASS model will be beneficial for tasks like document summarization. Our proposed models provide a generic alternative and can be easily adapted to various text generation tasks. The BERTSHARE setup sets a new state-of-the-art, outperforming all existing baselines by a large margin on the BBC extreme summarization task. The best model on the CNN/DailyMail dataset performs comparable to the Pointer Generator network (See et al., 2017) and the pre-trained single-decoder model with TransformerLM (Khandelwal et al., 2019). Our model, however, lags behind the Bottom-Up system (Gehrmann et al., 2018) with a task-specific module for content selection along with the copy mechanism (Gu et al., 2016) and the UniLM model (Dong et al., 2019) with BERT-Large pre-trained for Bidirectional, unidirectional and seq2seq language modeling objectives.

4 Further Setups & Negative Results

4.1 Using Larger Models

We performed the bulk of our experiments on the 12-layer checkpoints of BERT and GPT, assuming that the findings will also hold for the 24-layer checkpoints. We chose the best-performing model BERTSHARE to also collect numbers using the 24-layer checkpoint. The results can be seen in Table 6. This setup achieves new state-of-the-art results on DiscoFuse, Wikisplit, and BBC XSum.

	metric	BERTSHARE 12 layer	BERTSHARE 24 layer
DiscoFuse	SARI	89.3	**89.9**
WikiSplit	SARI	63.5	**63.7**
WikiSplit	Bleu	**77.2**	77.0
Gigaword	Rouge-1	38.1	38.4
Gigaword	Rouge-2	**19.8**	**19.8**
Gigaword	Rouge-L	35.6	35.7
CNN/Dailymail	Rouge-1	39.3	39.8
CNN/Dailymail	Rouge-2	18.1	17.7
CNN/Dailymail	Rouge-L	36.5	37.0
BBC XSum	Rouge-1	38.5	**38.9**
BBC XSum	Rouge-2	16.1	**16.4**
BBC XSum	Rouge-L	31.1	**31.5**

Table 6: Results of the setups with 24-layer checkpoints. For summarization datasets, we report ROUGE F_1 scores. Bold numbers represent new state-of-the-art results.

We also experimented with the GPT-2 counterpart GPT and the public checkpoint with 24 layers and 345M parameters. We found almost no change on the WikiSplit task and even a quality loss on the DiscoFuse task.

4.2 Tuning GPT-2 Based Models

Our intuition was that the natural language generation part plays a big role in the overall performance of a given setup. Given that GPT-2 was trained as a language model on a large corpus, we were surprised that setups using this checkpoint performed relatively poorly. To ensure that this was not due to an unfortunate choice of hyperparameters, we tuned the learning rate, the warmup steps and the optimizer $\in$ {Adam, Adafactor} for the GPT-2 based setups (RND2GPT, GPT, BERT2GPT) on the DiscoFuse task. Naturally, this gave us slightly higher numbers but not at a magnitude that would suggest a previously suboptimal setting. Specifically, we got a SARI score of 88.8 compared to 88.4 for BERT2GPT, 88.1 compared to 88.0 for GPT and 87.7 compared to 86.5 for RND2GPT.

4.3 Initializing only Embeddings

In this experiment, we want to investigate the impact of the non-contextualized BERT and GPT-2 embeddings. This means we are initializing the transformer model with only the embedding

matrices. The advantage of this setup would be that we could freely choose the model architecture and size and adapt it to a specific task. We found almost no improvement over the fully randomly initialized model RND2RND. Concretely, we compute a SARI score of 87.1 using the BERT embeddings and 87.0 using the GPT-2 embeddings, compared to 86.9 of the RND2RND baseline. The numbers on WikiSplit are 61.1 for BERT and 61.2 for GPT-2, compared to 61.0 of the RND2RND baseline. We observe slightly higher improvements of up to 2 percentage points when training on only 10% of the training data.

4.4 Initializing only Layers

Contrary to the previous section, we want to investigate the effect of initializing everything but the word embedding matrix. The embedding matrix accounts for only 10-31% of all learnable parameters and sometimes the vocabulary given from a public checkpoint might not be optimal for a certain task. In these cases, it would be nice to redefine the vocabulary while still leveraging the checkpoint. First, we remove the embeddings matrices from the warm started variables and observe a drop of 1.7 points using the BERTSHARE setup and 11 points using the GPT setup (Table 7). The latter is probably due to the large vocab of the GPT-2 model which now remains random initialized. We then train a new BPE model with 16k tokens using the DiscoFuse training data (Kudo and Richardson, 2018; Sennrich et al., 2015). We observe almost no change on BERTSHARE, suggesting that the BERT vocabulary was already optimal for DiscoFuse. GPT however, showed a significant improvement using this much smaller vocabulary but is still behind the fully initialized setup. Finally, we experimented with a more sensitive way of training the model, meaning that we fix all warm started variables for 100k steps. During this pre-training phase, we only train the new word embeddings. After the pre-training, we fine-tune the entire model for another 300k steps. This training scheme resulted in an improvement of 0.5 for the BERTSHARE setup, but overall the number is still way behind the fully initialized setup. For GPT, this training scheme did not result in a satisfying training curve.

4.5 Initializing a Subset of Layers

Motivated by the results of using 24 layers in Section 4.1, we want to investigate if only a subset of these 24 layers can be used. To account for the larger hidden layer size (1024 vs. 768) and filter size (4096 vs. 3072) we limit ourselves to using only 10 layers and the embedding matrix of this model. This model still has more parameters then the base model (324M vs. 221M for BERT2BERT, 198M vs. 136M for BERTSHARE) but we were able to train it with the same batch size, in a comparable amount of time (3 min/1000 iterations). As an initial experiment, we used the first 10 layers out of the large BERT checkpoint to initialize the BERTSHARE setup. This gave us a SARI score of 88.2 on DiscoFuse, compared to 89.3 of using the base checkpoint and compared to 87.0 of using the embeddings only (see Section 4.3). We then performed a hyperparameter search on the evaluation set using CMA-ES (Hansen, 2016) to find an optimal subset of layers to use. The best setup used the following layers: 9, 10, 13-18, 23, 24; and achieved a SARI score of 89.1. While this is a remarkable improvement over using the first 10 layers, this setup is still outperformed by the base BERT model.

	BERTSHARE	GPT
DiscoFuse	89.3	88.0
- embeddings from checkpoint	87.5	77.0
+ task specific SentencePieces	87.5	84.2
+ pre-training SentencePieces	88.0	69.7

Table 7: SARI scores on the Discofuse dataset when experimenting with different embedding setups. Each row also includes the setups of all previous rows.

5 Related Work

Representation learning. Starting around 2013, word embeddings like word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) became popular as they were easy to train in an unsupervised fashion on raw text and they improved several downstream tasks when used as features. These word embeddings are invariant to the context the word is in. There has been work to contextualize these embeddings, mainly to account for synonyms (e.g. (Huang et al., 2012; Rothe and Schütze, 2015)) before, but only in 2018 did training of the contextualized embeddings using large deep neural networks and an unsupervised training scheme become popular.

While ELMo (Peters et al., 2018) and ULMFiT (Howard and Ruder, 2018) are based on LSTMs (Hochreiter and Schmidhuber, 1997), BERT and

GPT are based on the transformer architecture (Vaswani et al., 2017). This architecture outperforms LSTMs on several NLP tasks and we therefore concentrated on these two pre-trained models. The contextualized embedding for each input token is given by the corresponding output of the last encoder layer.

Pre-training models. One can also see these models as pre-trained models (Dai and Le, 2015), which are then fine-tuned for a downstream task. This is the conceptual view we adopted for this paper. Why unsupervised pre-training helps deep learning was investigated by Erhan et al. (2010). While the unsupervised pre-training strategies are different from those used in our paper, we expect the findings to still hold. They show that unsupervised pre-training is not simply a way of getting a good initial marginal distribution, that classical regularization techniques cannot achieve the same performance as unsupervised pre-training, and that the effect of unsupervised pre-training does not go away with more training data. An extensive study of pre-training was done by Bowman et al. (2018). This study compares single sentence classification, sentence pair classification, sequence to sequence and language modeling tasks for pre-training and measures the effect on GLUE. The primary results support the use of language modeling. Peters et al. (2019) explore whether it is preferable to fine-tune the entire model on a specific task or to use the learned representations as features, i.e. freezing the pre-trained model. Their results suggest that the relative performance of fine-tuning vs. feature extraction depends on the similarity between the pre-training and the target tasks. Wang et al. (2019) propose a combination of both, were first the model is trained with the BERT parameters being frozen and then the entire model is fine-tuned. This is the training scheme we used in Subsection 4.4.

Pre-training for sequence generation. Pre-training for seq2seq learning was first done by Ramachandran et al. (2016). They used a language model to pre-train the encoder and decoder of an RNN seq2seq model. Their method improved Bleu scores on newstest2014 by 3 points and Rouge-L on CNN/Dailymail also by 3 points. However their Bleu score of 24.7 on newstest2014 En→De, compared to 30.6 in this work, and 29.4 Rouge-L on CNN/Dailymail, compared to 36.33 also show the superiority of the transformer model as well as the masked language model objective of BERT. MASS (Song et al., 2019) is a BERT-inspired method of pre-training sequence to sequence models. One advantage of this method is that, in contrast to our setups (except for GPT), the encoder-decoder attention mechanism is also pre-trained. The downside of this approach is that the pre-trained model is task-specific and not as general as BERT or GPT-2. UniLM (Dong et al., 2019) also unifies bidirectional, unidirectional, and sequence to sequence language modeling. At the time of writing, no public checkpoint was available to us. We compare our work with their results in Table 5. To overcome the issue that the encoder-decoder attention is not pre-trained, Khandelwal et al. (2019) pre-trained a single transformer language model that encodes the source and generates the target. This setup matches our GPT setup. Lample and Conneau (2019) pre-train their model using casual language modeling (like GPT), masked language modeling (like BERT) and a third new objective called translation language modeling to improve cross-lingual pre-training.

Leveraging public checkpoints. BERT has been used for various NLP tasks, such as question answering on the SQuAD 2.0 dataset (Rajpurkar et al., 2018). It also achieved new state-of-the-art results on the General Language Understanding Evaluation (GLUE) benchmark (Williams et al., 2017) and grounded commonsense inference (SWAG) (Zellers et al., 2018). All of these tasks are a form of classification or regression. Liu (2019) fine-tuned BERT for Extractive Summarization.

An analysis of different layers of the BERT model was performed by (Tenney et al., 2019). They found that the classical NLP pipeline appears in the expected sequence. In the context of our experiments in Section 4.5, this would mean that the DiscoFuse task profits the most from pre-trained information about POS, constituents, dependencies and semantic roles. A similar study by (Jawahar et al., 2019) found that BERT captures phrase-level information in the lower layers and linguistic information in intermediate layers, with surface features at the bottom, syntactic features in the middle and semantic features at the top.

GPT was also evaluated on natural language inference tasks. In the extended version of GPT-2, the model was evaluated on more general natural language processing tasks, like machine translation, reading comprehension, summarization, and language modeling. GPT-2 achieved new state-

of-the-art results on several language modeling datasets. On the other tasks, GPT-2 outperformed some unsupervised baselines but is still far behind supervised or task-specific approaches.

After we performed the majority of our experiments, XLNet (Yang et al., 2019), an autoregressive pre-training method based on Transformer XL (Dai et al., 2019) was released. XLNet achieved new state-of-the-art results on several NLP task. We leave the experiments with their public checkpoint for future work.

6 Conclusion

We performed an extensive study on leveraging pre-trained checkpoints for text generation. Our findings show, that a pre-trained encoder is an essential part for sequence generation task. Most tasks also profit from sharing the weights between the encoder and the decoder (BERTSHARE setup). Combining BERT and GPT-2 into a single model (BERT2GPT setup) did not work and often underperformed a randomly initialized baseline. This was not the case for MT de→en, where the vocabulary setting is in favor of this particular task. We conjecture that both pre-training methods can be combined beneficially but that the gains are outweighed by having to combine two vocabularies.

References

Roee Aharoni and Yoav Goldberg. 2018. Split and rephrase: Better evaluation and a stronger baseline. *CoRR*, abs/1805.01035.

Jan A Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. 2018. Learning to split and rephrase from wikipedia edit history. In *EMNLP*.

Samuel R. Bowman, Ellie Pavlick, Edouard Grave, Benjamin Van Durme, Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, and Berlin Chen. 2018. Looking for elmo's friends: Sentence-level pretraining beyond language modeling. *CoRR*, abs/1812.10860.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. Kermit: Generative insertion-based modeling for sequences. volume 1906.01604v1.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. *CoRR*, abs/1511.01432.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *CoRR*, abs/1905.03197.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb).

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. 2019. Discofuse: A large-scale dataset for discourse-based sentence fusion. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *staENCoRRMLPte-of-the-art*, abs/1604.00772.

Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*. Morgan, Kaufmann.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Urvashi Khandelwal, Kevin Clark, Dan Jurafsky, and Lukasz Kaiser. 2019. Sample efficient text summarization using a single pre-trained transformer. *CoRR*, abs/1905.08836.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2018. Abstractive summarization of reddit posts with multi-level memory networks. *CoRR*, abs/1811.00783.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Chin Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction at NAACL*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Matthew Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *CoRR*, abs/1903.05987.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2016. Unsupervised pretraining for sequence to sequence learning. *CoRR*, abs/1611.02683.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. *CoRR*, abs/1905.05950.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Ran Wang, Haibo Su, Chunye Wang, Kailin Ji, and Jupeng Ding. 2019. To tune or not to tune? how about the best of both worlds? *arXiv preprint arXiv:1907.05338*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. *CoRR*, abs/1808.05326.