Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (Rafell et al., 2020)

Muhammad Abdul-Mageed
Deep Learning & Natural Language Processing Lab
The University of British Columbia

Agenda

1. Objective
2. T5 Summary
3. NLP Models for Knowledge Transfer

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

Colin Raffel* CRAFFEL@GMAIL.COM
Noam Shazeer* NOAM@GOOGLE.COM
Adam Roberts* ADAROB@GOOGLE.COM
Katherine Lee* KATHERINELEE@GOOGLE.COM
Sharan Narang SHARANNARANG@GOOGLE.COM
Michael Matena MMATENA@GOOGLE.COM
Yanqi Zhou YANQIZ@GOOGLE.COM
Wei Li MWEILI@GOOGLE.COM
Peter J. Liu PETERJLIU@GOOGLE.COM

Google, Mountain View, CA 94043, USA

Editor: Ivan Titov

Abstract

Transfer learning, where a model is first pre-trained on a data-rich task before being fine-tuned on a downstream task, has emerged as a powerful technique in natural language processing (NLP). The effectiveness of transfer learning has given rise to a diversity of approaches, methodology, and practice. In this paper, we explore the landscape of transfer learning techniques for NLP by introducing a unified framework that converts all text-based language problems into a text-to-text format. Our systematic study compares pre-training objectives, architectures, unlabeled data sets, transfer approaches, and other factors on dozens of language understanding tasks. By combining the insights from our exploration with scale and our new "Colossal Clean Crawled Corpus", we achieve state-of-the-art results on many benchmarks covering summarization, question answering, text classification, and more. To facilitate future work on transfer learning for NLP, we release our data set, pre-trained models, and code.[1]

Keywords: transfer learning, natural language processing, multi-task learning, attention-based models, deep learning

T5 Summary

- Explores the landscape of transfer learning techniques for NLP
- Introduces a unified framework that converts all text-based language problems into a text-to-text format.
- Compares pre-training objectives, architectures, unlabeled data sets, transfer approaches, and other factors on dozens of language understanding tasks.
- Introduces “Colossal Clean Crawled Corpus"
- Achieves state-of-the-art results on many benchmarks covering summarization, question answering, text classification, and more.

NLP Models as Knowledge Carriers

- Goal: Develop a model that “understands" text
- Use model on downstream task: low-level (spelling) or high-level (common sense, e.g., a “tuba" is too large to fit in a backbag)
- Knowledge usually provided as part of an auxiliary task
- Word vectors as an example
- More recently, pre-training of language models affords this knowledge
- In computer vision transfer is performed as supervised learning on a dataset such as ImageNet
- In NLP, it’s unsupervised pre-training
- Common Crawl project provides 20 TB of text each month

NLP as Text-to-Text Modeling

- Cast every NLP problem as a “text-to-text" problem

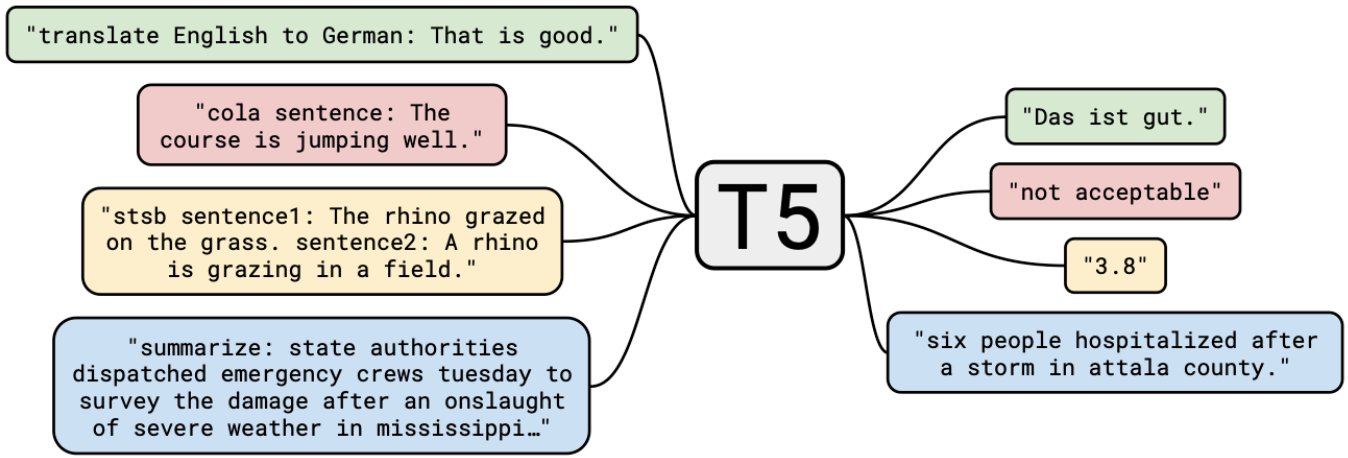

Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. “T5” refers to our model, which we dub the “**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer”.

Common Crawl Issues

- A lot of gibberish or boiler-plate text (e.g., menus, error messages, or duplicate text)
- Contains content unlikely helpful for tasks at hand (offensive language, placeholder text, source code, etc.)

Common Crawl Cleaning I

- 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.

Common Crawl Cleaning II

Additionally, since most of our downstream tasks are focused on English-language text, we used `langdetect`[7] to filter out any pages that were not classified as English with a probability of at least 0.99. Our heuristics are inspired by past work on using Common Crawl as a source of data for NLP: For example, Grave et al. (2018) also filter text using an automatic language detector and discard short lines and Smith et al. (2013); Grave et al. (2018) both perform line-level deduplication. However, we opted to create a new data set because prior data sets use a more limited set of filtering heuristics, are not publicly available, and/or are different in scope (e.g. are limited to News data (Zellers et al., 2019; Liu et al., 2019c), comprise only Creative Commons content (Habernal et al., 2016), or are focused on parallel training data for machine translation (Smith et al., 2013)).

Downstream Tasks (Partial List)

GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019b) each comprise a collection of text classification tasks meant to test general language understanding abilities:

- 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))

Input and Output Format

As an example, to ask the model to translate the sentence "That is good." from English to German, the model would be fed the sequence "translate English to German: That is good." and would be trained to output "Das ist gut." For text classification tasks, the model simply predicts a single word corresponding to the target label. For example, on the MNLI benchmark (Williams et al., 2017) the goal is to predict whether a premise implies ("entailment"), contradicts ("contradiction"), or neither ("neutral") a hypothesis. With our preprocessing, the input sequence becomes "mnli premise: I hate pigeons. hypothesis: My feelings towards pigeons are filled with animosity." with the corresponding target word "entailment". Note that an issue arises if our model outputs text on a text classification task that does not correspond to any of the possible labels (for example if the model outputs "hamburger" when the only possible labels for a task were "entailment", "neutral", or "contradiction"). In this case, we always count the model's output as wrong, though we never observed this behavior in any of our trained models. Note that the choice of text prefix used for a given task is essentially a hyperparameter; we found that changing the exact wording of the prefix had limited impact and so did not perform extensive experiments into different prefix choices. A diagram of our text-to-text framework with a few input/output examples is shown in Figure 1. We provide full examples of preprocessed inputs for every task we studied in Appendix D.

Unsupervised (Denoising) Objective

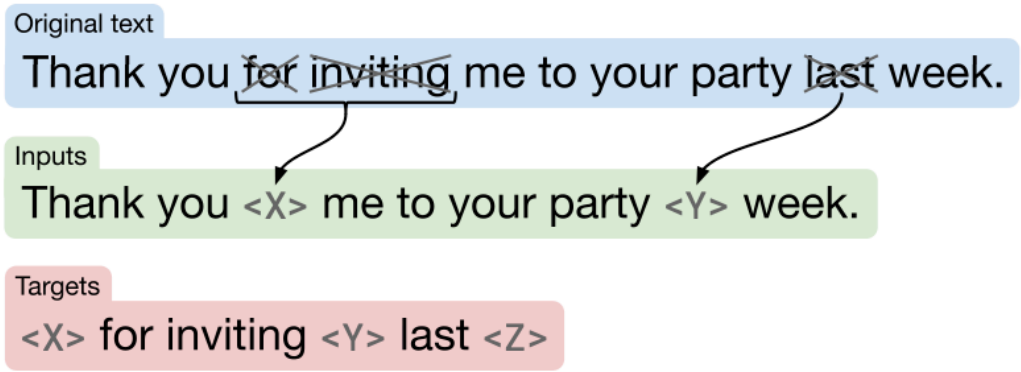

Figure 2: Schematic of the objective we use in our baseline model. In this example, we process the sentence "Thank you for inviting me to your party last week." The words "for", "inviting" and "last" (marked with an ×) are randomly chosen for corruption. Each consecutive span of corrupted tokens is replaced by a sentinel token (shown as **<X>** and **<Y>**) that is unique over the example. Since "for" and "inviting" occur consecutively, they are replaced by a single sentinel **<X>**. The output sequence then consists of the dropped-out spans, delimited by the sentinel tokens used to replace them in the input plus a final sentinel token **<Z>**.

Baseline Model Performance

	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

Table 1: Average and standard deviation of scores achieved by our baseline model and training procedure. For comparison, we also report performance when training on each task from scratch (i.e. without any pre-training) for the same number of steps used to fine-tune the baseline model. All scores in this table (and every table in our paper except Table 14) are reported on the validation sets of each data set.

Mask Settings in Attention Mechanism

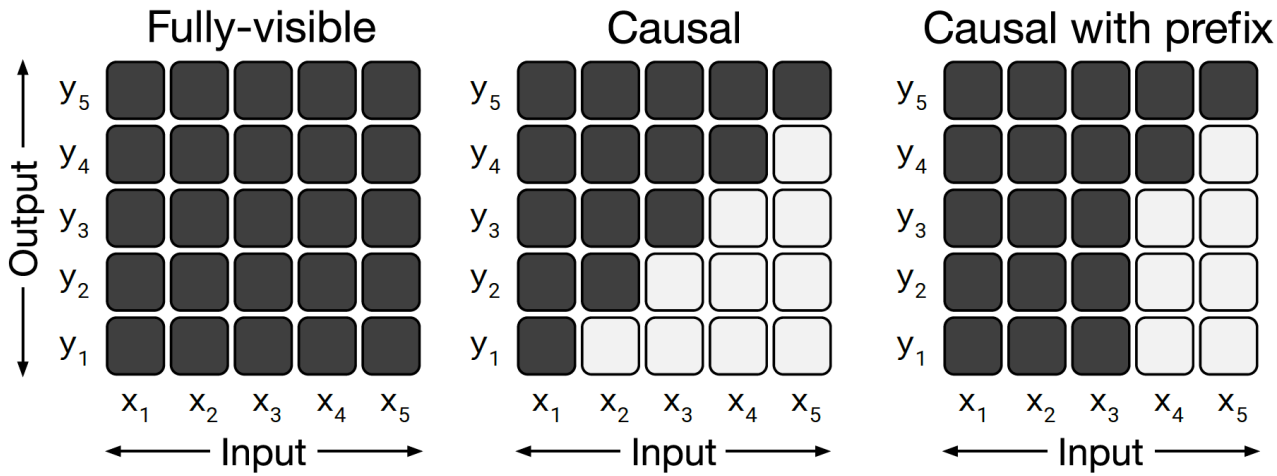

Figure 3: Matrices representing different attention mask patterns. The input and output of the self-attention mechanism are denoted x and y respectively. A dark cell at row i and column j indicates that the self-attention mechanism is allowed to attend to input element j at output timestep i. A light cell indicates that the self-attention mechanism is *not* allowed to attend to the corresponding i and j combination. Left: A fully-visible mask allows the self-attention mechanism to attend to the full input at every output timestep. Middle: A causal mask prevents the ith output element from depending on any input elements from "the future". Right: Causal masking with a prefix allows the self-attention mechanism to use fully-visible masking on a portion of the input sequence.

Transformer Variants

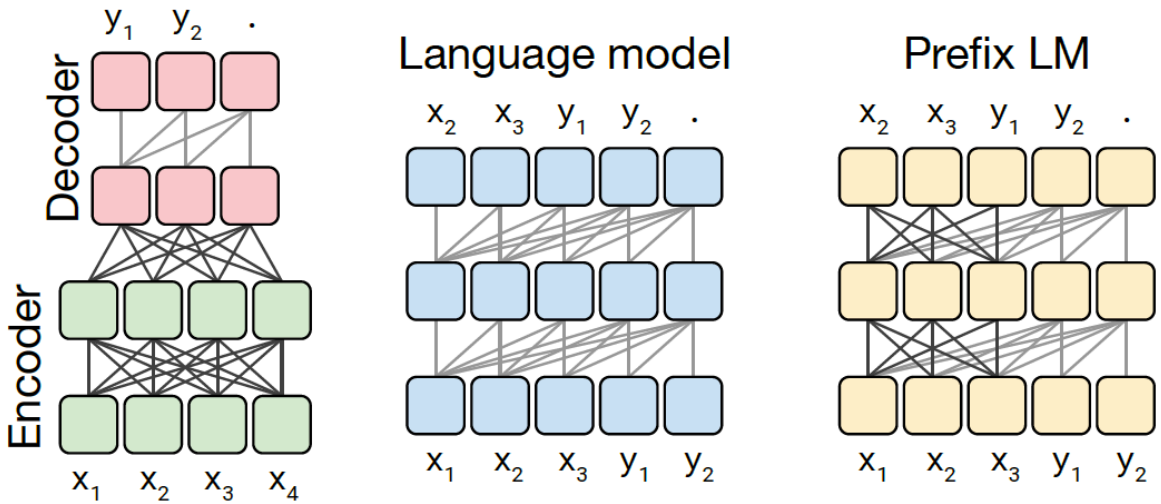

Figure 4: Schematics of the Transformer architecture variants we consider. In this diagram, blocks represent elements of a sequence and lines represent attention visibility. Different colored groups of blocks indicate different Transformer layer stacks. Dark grey lines correspond to fully-visible masking and light grey lines correspond to causal masking. We use "." to denote a special end-of-sequence token that represents the end of a prediction. The input and output sequences are represented as x and y respectively. Left: A standard encoder-decoder architecture uses fully-visible masking in the encoder and the encoder-decoder attention, with causal masking in the decoder. Middle: A language model consists of a single Transformer layer stack and is fed the concatenation of the input and target, using a causal mask throughout. Right: Adding a prefix to a language model corresponds to allowing fully-visible masking over the input.

Tnput-Output Examples

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>

Table 3: Examples of inputs and targets produced by some of the unsupervised objectives we consider applied to the input text "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. <M> denotes a shared mask token and <X>, <Y>, and <Z> 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; we show this via the greyed-out word apple.