Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey

BONAN MIN, Amazon AWS AI Labs, USA
HAYLEY ROSS, Harvard University, USA
ELIOR SULEM, University of Pennsylvania, USA
AMIR POURAN BEN VEYSEH and THIEN HUU NGUYEN, University of Oregon, USA
OSCAR SAINZ and ENEKO AGIRRE, University of the Basque Country (UPV/EHU), Spain
ILANA HEINTZ, Synoptic Engineering, USA
DAN ROTH, University of Pennsylvania, USA

Large, pre-trained language models (PLMs) such as BERT and GPT have drastically changed the Natural Language Processing (NLP) field. For numerous NLP tasks, approaches leveraging PLMs have achieved state-of-the-art performance. The key idea is to learn a generic, latent representation of language from a generic task once, then share it across disparate NLP tasks. Language modeling serves as the generic task, one with abundant self-supervised text available for extensive training. This article presents the key fundamental concepts of PLM architectures and a comprehensive view of the shift to PLM-driven NLP techniques. It surveys work applying the pre-training then fine-tuning, prompting, and text generation approaches. In addition, it discusses PLM limitations and suggested directions for future research.

CCS Concepts: • **Computing methodologies → Natural language processing;**

Additional Key Words and Phrases: Large language models, foundational models, generative AI, neural networks

ACM Reference format:
Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey. *ACM Comput. Surv.* 56, 2, Article 30 (September 2023), 40 pages.
https://doi.org/10.1145/3605943

B. Min, H. Ross, E. Sulem, and A. P. B. Veyseh contributed equally to this research. Bonan Min completed this work at Raytheon BBN Technologies (prior to joining AWS AI).

Authors' addresses: B. Min, Amazon AWS AI Labs, 2795 Augustine Dr, Santa Clara, CA 95054, USA; email: bonanmin@amazon.com; H. Ross, Department of Linguistics, Boylston Hall, 3rd floor, Harvard Yard, Cambridge, MA 02138, USA; email: hayleyross@g.harvard.edu; E. Sulem, Ben-Gurion University of the Negev, Department of Software and Information Systems Engineering, Building 96, Room 207, Marcus Family Campus, P.O.B. 653 Beer-Sheva, 8410501, Israel; email: eliors@seas.upenn.edu; A. P. Ben Veyseh, 1841 Garden Ave, Unit 213, Eugene, Oregon, 97403, USA; email: apouran@cs.uoregon.edu; T. H. Nguyen, University of Oregon, 330, Deschutes Hall, 1477 E. 13th Avenue, Eugene, OR 97403, USA; email: thien@cs.uoregon.edu; O. Sainz, University of the Basque Country (UPV/EHU), Manuel Lardizabal 1, 20008 Donostia, Spain; email: oscar.sainz@ehu.eus; E. Agirre, e.agirre@ehu.eus, University of the Basque Country (UPV/EHU), Manuel Lardizabal 1, 20008 Donostia, Spain; email: e.agirre@ehu.eus; I. Heintz, Synoptic Engineering, 3030 Clarendon Blvd, Arlington, VA, 22201, USA; email: ilana@synopticengineering.com; D. Roth, University of Pennsylvania, 3330 Walnut St., Philadelphia, PA 19104-6309, USA; email: danroth@seas.upenn.edu.

0360-0300/2023/09-ART30 $15.00
https://doi.org/10.1145/3605943

1 INTRODUCTION

Large **pre-trained language models** (**PLMs**), such as the BERT [31] and GPT [144] families of models, have taken **Natural Language Processing** (**NLP**) by storm, achieving state-of-the-art performance on many tasks.

These large PLMs have fueled a paradigm shift in NLP. Take a classification task $p(y|x)$ (classifying textual input x into a label y) as an example: traditional statistical NLP approaches often design handcrafted features to represent x, then apply a machine learning model (e.g., SVM [25], logistic regression) to learn the classification function. Deep learning models learn the latent feature representation via a deep neural network [90] in addition to the classification function. Note that the latent representation needs to be learned afresh for each new NLP task, and that in many cases, the size of the training data limits the quality of the latent feature representation. Given that understanding the structure and content of language is common to all NLP tasks, we might instead learn a generic latent feature representation from some generic task once, then share it across all NLP tasks. **Language modeling** (**LM**), where the model learns to predict a word given its context (often, the next word given the previous words), is one such generic task with abundant, naturally occurring, self-supervised text available for training. We call this "pre-training," expecting that some form of additional training or task modification is necessary to perform the desired NLP task using this pre-trained model as a resource. While leveraging language modeling as a pre-training task was possible to some degree with pre-trained static (context-independent) word embeddings such as word2vec [120], this paradigm shift really took off when PLMs were introduced: for numerous NLP tasks, researchers now leverage existing PLMs via *fine-tuning* for the task of interest, *prompting* the PLMs to perform the desired task, or reformulating the task as a *text generation* problem with application of PLMs to solve it accordingly. Advances in these three PLM-based paradigms have continuously established new state-of-the-art performances.

This article surveys recent works that leverage PLMs for NLP, organized as follows:

- Pre-train then fine-tune (Section 2): perform general-purpose pre-training with a large unlabeled corpus, then perform a small amount of task-specific fine-tuning for the task of interest.
- Prompt-based learning (Section 3): prompt a PLM such that solving an NLP task is reduced to a task similar to the PLM's pre-training task (e.g., predicting a missing word), or a simpler proxy task (e.g., textual entailment). Prompting can more effectively leverage the knowledge encoded in the PLMs, leading to few-shot approaches.
- NLP as text generation (Section 4): reformulate NLP tasks as text generation, to fully leverage knowledge encoded in a generative LMs such as GPT-2 [145] and T5 [147].

Generative PLMs can be also used for text generation tasks. We refer readers to excellent surveys [98, 207] on text generation. This article, unless otherwise specified, focuses on a broad range of NLP tasks that are not generative in nature (e.g., classification, sequence labeling, and structure prediction) including syntactic or semantic parsing, **Information Extraction** (**IE**), **Question Answering** (**QA**), **Textual Entailment** (**TE**), and so on.

The article is organized as follows: Section 2 provides background on the PLMs and describes *pre-train then fine-tune*. Section 3 discusses *prompt-based learning*. Section 4 describes *NLP as text generation*. We discuss limitations and provide directions for future research in Section 5, and discuss the relation of this work to the new paradigm of user alignment (e.g., instruction tuning) in Section 6, before concluding in Section 7.

2 PARADIGM 1: PRE-TRAIN THEN FINE-TUNE

We first provide a primer on PLMs, then describe approaches that use frozen or fine-tuned PLMs for NLP tasks.

2.1 The Introduction of PLMs to NLP

While pre-training on generic tasks and then fine-tuning that model on specific tasks has been applied in machine learning (e.g., computer vision) since at least 2010 [37, 70, 206], the technique did not gain traction in its current form in NLP until later in the decade with the publication of the Transformer architecture [183]. Early steps in this direction were made by using static (context-independent) word embeddings such as word2vec [120]. These approaches only pre-trained the word embeddings (essentially the first layer of the new model) and had to train all the remaining layers from scratch, as opposed to building on an entire pre-trained model. Success of deep neural models in the NLP community required the discovery of an appropriate self-supervised task[1] and drastically larger model sizes for effective performance than those in CV, delaying neural model uptake for several years. Achieving all this together depended on the advent of a new neural architecture, the Transformer [183], which allowed model training to be scaled up in a way not possible with early recurrent neural networks [64]. We explore these aspects further in the discussion below.

The idea of pre-training on a LM task was well established before deep neural modeling became the norm. Collobert and Weston [21] first suggested pre-training a model on a number of tasks to *learn* features instead of handcrafting them (the predominant approach at the time). Their version of LM pre-training, however, differed significantly from the methods we see today. They used language modeling as only one of many tasks in a multitask learning setting, along with other supervised tasks such as **part-of-speech** (**POS**) tagging, **named entity recognition** (**NER**), and **semantic role labeling** (**SRL**). They proposed sharing the weights of their deepest convolutional layer—the word embeddings learned by the model—between the multiple training tasks and fine-tuning the weights of the remaining feed-forward layers for each individual task.

The idea of sharing word embeddings across models took off later with word2vec [120] and GloVe [132]. Static word embeddings generated by these algorithms are trained on a masked language modeling task (especially in the CBOW formulation of word2vec, which predicts a word given its context; see [94]). The "models" used for this task are tiny, however, and only the embedding weights are used in subsequent tasks. This also means that the resulting embeddings are not context dependent. While the widespread adoption of pre-trained word embeddings led to performance gains across the board, models and architectures remained task specific and needed a large amount of task-specific labeled data to succeed.

Pre-training and fine-tuning entire models did not gain popularity in NLP until the advent of ELMo [134] and ULMFiT [66]. Both models are based on **Long Short-Term Memory** architecture (**LSTMs**) [63], and propose fine-tuning the LM layer by layer for downstream application. Both studies also suggested adding classifier layers on top of the LM, which were fine-tuned alongside the LM layers. The combination of these changes with the substantially larger model size and pre-training corpus size compared to previous models marked the first instances of the pre-training then fine-tuning paradigm, yielding competitive or improved performance compared to the then-state-of-the-art across tasks, for both ELMo and ULMFiT. This demonstrated the value of LM pre-training on a large scale.

The pace of this change in approach picked up dramatically in late 2018 when Vaswani et al. [183] introduced the Transformer architecture, which is well suited for language model pre-training. The Transformer's multi-head self-attention mechanism allows every word to attend to either all previous words or every word except the target, allowing the model to efficiently capture long-range dependencies without the expensive recurrent computation in LSTMs. Multiple

[1]In self-supervised learning, the ground truth (e.g., the missing word) comes from the unlabeled text itself. This allows the pre-training to scale up with the near-infinite amount of text available on the web.

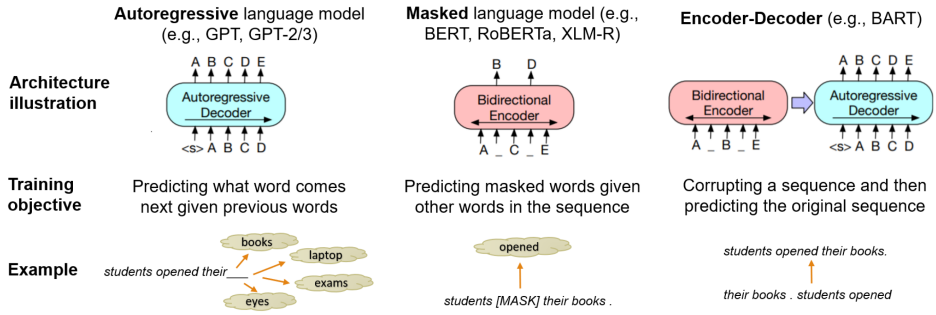

Fig. 1. Three popular configurations (architecture + training objective) of PLMs. Model architecture illustrations are from Lewis et al. [96]. For the encoder-decoder model, only the corruption strategy of document rotation (as in BART) is shown.

layers of multi-head self-attention allow for increasingly more expressive representations, useful for a range of NLP problems. As a result, nearly all popular PLMs, including the GPT family [39, 144, 145], Gopher [38], BERT [31] and family, XLM-R [22], BART [96], T5 [147], and T0 [41], are now based on the Transformer architecture. They also differ in a number of important ways, which we discuss in the following sections. For more details about the Transformer architecture, we refer the reader to the original paper or to the excellent tutorials available.[2,3]

2.2 Modern Pre-trained Language Models

There are three classes of PLM architecture and three classes of training objective. A transformer-based LM may be decoder-only (e.g., GPT, Gopher), encoder-only (e.g., BERT, XLM-R), or encoder-decoder (e.g., BART, T5, T0). Moreover, models may be trained autoregressively (predict the next word given the left-hand context), on **masked language modeling** (**MLM**) (fill in the blank, i.e., the masked word, given context on both sides), or on a range of denoising tasks where the model must undo some corruption of the original sequence, such as sentence permutation, token deletion, or span deletion. In fact, MLM may be viewed as a type of denoising, but it retains a special status due to its popularity. Typically, although not necessarily, decoder-only models are trained with an autoregressive objective, encoder-only models are trained on MLM, and encoder-decoder architectures are trained on a denoising task or on MLM. Models with a decoder always output text; whether this is fluent language depends on the pre-training objective. Autoregressively trained LMs lend themselves to generating the continuation of text, in particular responding to prompts. Encoder-only models can be used to output text via the head used for their pre-training objectives (e.g., by repeatedly predicting a mask token at the end of a sequence), but are typically used to produce embeddings for classification.

Figure 1 shows the difference in model architecture and a typical training objective with an example training input for each. While we will focus on these canonical combinations of architecture and pre-training objective as seen in popular models, recent work attempts to disentangle the effects of the architecture from the objective and explores other combinations such as training a decoder-only model on MLMs [4, 189]. While Wang et al. [189] find that autoregressively trained decoders and encoder-decoder models trained on MLM (two of the classic pairings described above) perform the best, respectively, in their two settings, they also note that it is straightforward to adapt models between objectives.

[2] http://nlp.seas.harvard.edu/2018/04/03/attention.html.
[3] http://jalammar.github.io/illustrated-transformer/.

2.2.1 Decoder-only Language Models and the Autoregressive Task. An autoregressive LM is trained to predict the next word x_i given all previous words x_1, x_2, …, x_{i-1}. The training objective is to maximize the log-likelihood $\sum_i log(P(x_i|x_1, x_2, \ldots, x_{i-1}); \theta_T)$, in which θ_T are the model parameters. In a Transformer decoder, these are in multiple layers of multi-head self-attention modules. Typical models include the GPT family (GPT [144], GPT-2 [145], and GPT-3 [39]), GPT-J [186], and Gopher [38], as well as many models specific to languages other than English.

These models only utilize the autoregressive *decoder* portion of the Transformer architecture, stacking multiple transformer decoder layers with masked self-attention. This allows the model to attend to all previous tokens in the sequence when predicting the next token. Each newer version of GPT and its descendants is trained with increasingly large amounts of text (Table 1).

The GPT paper [144] proposed fine-tuning GPT for specific tasks, providing examples for **natural language inference** (**NLI**), QA (including commonsense reasoning), semantic similarity and paraphrase detection, sentiment analysis, and linguistic acceptability (CoLA, [193]), as well as the GLUE benchmark. In particular, GPT achieves a dramatic improvement on CoLA (scoring 45.4 compared to the previous state of the art of 35.0), showcasing the model's ability to gain a much more sophisticated grasp of language than previous models. Subsequent versions of GPT (GPT-2 and GPT-3, [39, 145]), however, do not opt for the fine-tuning approach and instead leverage GPT's generative design to tackle tasks in a prompt-based manner, as described in Sections 3 and 4. Gopher [38] relies on a mix of language generation and using the probabilities generated by the LM to solve downstream tasks by framing them as multiple-choice tasks.

2.2.2 Encoder-only Language Models and the MLM Task. Whereas autoregressive models are unidirectional, MLMs predict a "masked" word conditioned on all other words in the sequence. When training an MLM, words are chosen at random to be masked using a special token `[MASK]`, or replaced by a random token. This forces the model to collect bidirectional information in making predictions. The training objective is to recover the original tokens at the masked positions: $\sum_i m_i log(P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n); \theta_T)$, in which $m_i \in \{0, 1\}$ indicates whether x_i is masked or not, and θ_T are the parameters in a Transformer encoder. It is a common practice to mask multiple words from a sequence to allow parallel training. Popular examples include BERT [31], RoBERTa [112], and XLM-R [22].

Specifically, MLMs such as BERT use the *encoder* portion of the Transformer architecture. Like autoregressive models, MLMs stack multiple transformer encoder layers to learn increasingly complex and meaningful representations, but they use masked self-attention to attend to all other tokens in the sequence when learning a representation for a particular token, providing more context for any given word.

There is a large family of models derived from BERT, including RoBERTa [112], which trains for longer with a more varied masking pattern; ALBERT [88], which is smaller and faster to train; SpanBERT [76], which masks spans instead of tokens; and XLNet [203] and Transformer-XL [28], which incorporate an autoregressive pre-training approach to better handle long-distance dependencies. See Rogers et al. [151] and Qiu et al. [143] for a full taxonomy of BERT-derived models.

2.2.3 Encoder-Decoder Language Models and the Denoising Task. The encoder-decoder model is a more flexible "text in, text out" model that learns to generate a sequence of token $y_1, \ldots, y_n$ given an input sequence $x_1, \ldots, x_m$. The training objective is to maximize the output's log-likelihood: $log(P(y_1, \ldots, y_n|x_1, \ldots, x_m); \theta_T)$, in which θ_T are the parameters in a full encoder-decoder Transformer model [183].

To generate adequate data for self-supervised pre-training, researchers experiment with different forms of sequence corruption. The input is a corrupted sequence and the output is the reconstructed (i.e., denoised) original sequence. Forms of sequence corruption include document

Table 1. Training Sources, Dataset Size, and Model Parameters for Popular PLMs

Model	Pre-Training Sources	Size of Pre-Training Corpus	# Model Parameters
(1) **English Monolingual Models**			
BERT (Base/Large) [31]	Wiki, books	3.3B tokens (13 GB data)	110M/340M
RoBERTa [112]	Wiki, books, web crawl	161 GB data	340M
XLNet [203]	Wiki, books, web crawl	142 GB data	340M
GPT [144]	Web crawl	800M tokens	117M
GPT-2 [145]	Web crawl	8M documents (40 GB data)	1.5B
GPT-3 [39]	Wiki, books, web crawl	300B tokens	175B
GPT-J [186]	Wiki, books, papers, web crawl	~275B tokens (825 GB data)	6B
Gopher [38]	Books, news, code, web crawl	300B tokens	280B
BART [96]	Wiki, books	3.3B tokens	~370M
T5 [147]	Web crawl	200B tokens (750 GB data)	11B
(2) **Multilingual Models**			
mBERT [31]	Wiki	21.9B tokens	172M
XLM-R(base/large) [22]	Web crawl	295B tokens	270M/550M
mT5 (large/XXL) [147]	Web crawl	6.3T tokens	1.2B/13B

Data sources differ, and are described in the citations listed in each row. Some models report their datasets in number of tokens or documents, others only in total size in GB.

rotation (Figure 1), sentence permutation, text infilling, and token deletion/masking, among others. Representative models include BART [96] and T5 [147]. Text infilling, which is closely related to the span-based MLM used in SpanBERT [76], along with sentence permutation turn out to be the best objectives for BART.

Given the **sequence-to-sequence** (**seq2seq**) nature, it is straightforward to fine-tune the encoder-decoder LM to perform seq2seq tasks such as machine translation, style transfer, and text summarization. The seq2seq formulation is versatile: many tasks can be reformulated as "text in, text out." We describe this unified formulation in Section 4.

2.3 Pre-training Corpora

The size, quality, language(s), and genre of the corpus chosen for pre-training are as important to discuss as model architecture, size, and pre-training task. Table 1 presents the sources and the corpus size used for several popular LMs. There is a clear trend of increasing the size of the pre-training corpus as well as increasing the diversity of the data. For example, ULMFiT [66] is trained on a small, highly pre-processed corpus of ~29,000 Wikipedia articles (103 million words), and is representative of models of that year. A few years later, models such as XLM-R, GPT-3, and T5 leveraged billions of words of crawled web data covering everything from forum posts to academic papers. In Table 1, we classify corpora sources by genre into Wiki content, books, news, academic papers (when explicitly mentioned as a source), code, and broad-coverage web crawl.

The composition of a pre-training corpus is important to understand whether one is training one's own PLM from scratch or adopting an existing PLM for downstream tasks. We note some widely used, publicly available corpora here. BookCorpus [227], used to train BERT, RoBERTa, and XLNet alongside Wikipedia, is a collection of around 7,000 unique books; Bandy and Vincent [9] provide a detailed analysis of its contents and several shortcomings. T5 and Gopher were trained on the Colossal Clean Crawled Corpus (C4). Dodge et al. [32] provide a detailed analysis of C4, citing some unexpected downsides, including a large quantity of machine-translated text, specifically patents. This is one of several published, Common Crawl–derived datasets. The Pile [48] is a notable open source collection of datasets, which includes a cleaned portion of Common Crawl alongside a range of other curated sources including several book corpora, academic papers, mathematics problems, and the Enron emails. The Pile is used for training GPT-J. While the rest of the GPT family and XLM-R are also trained on datasets derived from

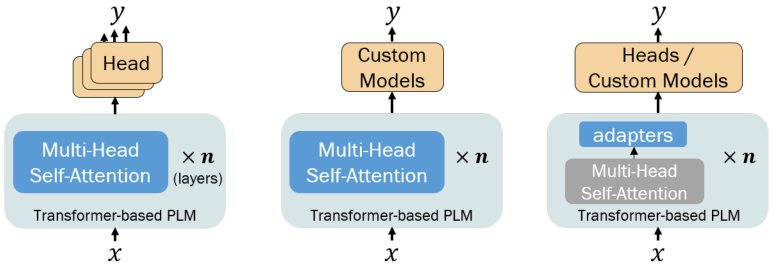

Fig. 2. Typical "pre-train then fine-tune" strategies. We illustrate strategies that fine-tune the full PLM (left), fine-tune the full PLM in a custom model (center), and fine-tune just a small adapter sub-layer. We show the Transformer blocks that will be fine-tuned for the specific tasks in blue, and the frozen blocks in gray. For brevity, we represent the entire Transformer block (stacked in n layers) by its multi-head self-attention and (if applicable) adapter layers. We refer readers to [183] and [137] for more architecture details. "Heads" refers to task-specific prediction functions [40].

CommonCrawl, their datasets are not publicly available, and thus opaque to straightforward analysis.

Raffel et al. [147] observe that the primary gains in performance are typically driven by model size and dataset size ("the bigger, the better"), if the quality of the dataset is held constant. Kaplan et al. [79] and Zhang et al. [214] discuss how model performance scales in relation to model and dataset size. Kaplan et al. [79] is a study of empirical scaling laws for training Transformer-based PLMs. Its main finding is that the test loss scales as a power law with model size, dataset size, and the amount of compute resources available for training. Other researchers highlight the interactions of size, data quality, and genre. Hendrycks et al. [62] show that larger models do not necessarily perform better out of domain. [147] shows that adding data increases performance only if that data is sufficiently "clean": a variant of T5 trained on an unfiltered version of C4 performs substantially worse than when filtering heuristics are applied that strip HTML and boilerplate, enforce a minimum number of sentences, remove duplicates, and so forth. Pursuing a higher level of "quality" produces gains only when improving genre match with the target task. In contrast to Hendrycks et al. [62], [147] find that the quantity of data eventually overcomes quality/genre-driven gains.

All recent models (GPT-2, GPT-3, T5, Gopher, etc.) use heuristics to clean their web-crawled data. Nevertheless, many quality issues persist, as discussed by Bandy and Vincent [9] and Dodge et al. [32]. Further, for multilingual corpora, some low-resource languages may be pervasively mislabeled or contain no usable natural language text [82]. Increasing the corpus size and range of genres covered can also lead to serious issues with bias and factuality, where models repeat biased, unethical (racist, sexist, ...), or incorrect beliefs seen in the training data ([52], among many others). Instances of such behavior are documented for all of the models discussed above and are increasingly discussed at publication time of the model, such as in the Gopher paper [38]. Bias is present even at the dataset cleaning stage; several filters disproportionately filter out content from or about minority individuals [32]. Among many reviews of bias in modern LMs, Bender et al. [12] may be the most outspoken in reviewing ethical risks and environmental considerations and calling for change in the training of use of PLMs. We return to the discussion of ethics in Section 5.

2.4 Fine-Tuning: Applying PLMs to NLP Tasks

Having described the various approaches to creating complex, meaningful representations through pre-training, we turn to the fine-tuning step that allows PLMs to perform accurately on disparate NLP tasks. Figure 2 illustrates typical pre-training then fine-tuning strategies.

2.4.1 Contextual Embeddings. The simplest approach to using large PLMs is to "freeze" the model and use its output as sophisticated, context-sensitive word embeddings for a subsequent architecture, which is trained from scratch for the specific task. While this still involves a forward pass through the PLM over the input text, the LM's weights are *not* fine-tuned, rendering this approach closer to a feature extraction family of approaches in classic statistical NLP. There are four types of scenario for using frozen PLMs.

First, in contexts with insufficient labeled data or compute power, "frozen" contextual embeddings are employed. For non-benchmark tasks, the only labeled training datasets may be too small to fine-tune even the top layers of BERT-base, let alone larger models. The computational cost of fine-tuning the entire PLM may be prohibitive for some applications or developers, leading to use of the more efficient frozen PLM solution. Other data-efficient and time-efficient approaches to fine-tuning are discussed in Section 2.4.4.

Second, highly complex or difficult NLP tasks use frozen PLMs to reduce training complexity. Examples are constituency parsing [215], semantic graph parsing using **Universal Conceptual Cognitive Annotation** (**UCCA**) [1, 73] and **Abstract Meaning Representation** (**AMR**) [8, 123, 212, 224], Aspect-Based Sentiment Analysis [101], and **Machine Translation** (**MT**) [226]. For instance, Zhang et al. [215] uses frozen BERT embeddings to seed an innovative approach to **Conditional Random Field** (**CRF**) modeling [85] that replaces the inside-outside algorithm with backpropagation, using a two-step process to first bracket and then label the parses, and a batched version of the CKY algorithm. For complex tasks like these, there may only be enough data or compute power available to train the secondary model (Zhang et al. [212] cited limitations in compute power). While the use of frozen PLM parameters is currently in vogue for these tasks, perhaps due to researcher preference for simplicity as well as computational requirements, we may see a shift to full-model fine-tuning for tasks with sufficient training data.

Third, unsupervised tasks such as word sense disambiguation [56] and word sense induction [3] are not associated with a supervised dataset for fine-tuning. Instead, frozen BERT embeddings enable a variety of strategies such as nearest-neighbor matching, affine transformations, **gated linear units** (**GLUs**, [29]), or clustering to perform these tasks.

Finally, some studies probe frozen LMs to uncover what knowledge it has learned during pre-training. Typically, a very small classifier is appended to the LM and trained briefly on a supervised task. If the classifier succeeds, we presume the knowledge is present in the pre-trained LM, as the classifier itself is too small to have learned the task from the limited supervised training data. This and related methods have spawned a fast-growing field of probing LMs (see Rogers et al. [151] for a thorough discussion of probing BERT).

2.4.2 Fine-tuning the PLM. This approach fine-tunes some or all the layers of the PLM and then adds one or two simple output layers known as prediction heads [40]. Typically, these are feed-forward classification layers. The output layers and the PLM are trained together in an end-to-end setup, with the bulk of the computation applied to fine-tuning the LM to produce the desired representation of the input. The task of the output layers is merely to condense the information provided by the embeddings of each token into the number of desired classes. The word embeddings used in the output layer may come from the top layer of the LM, or from a concatenation or a weighted average of the top n (often $n = 4$) layers [134]. Figure 2 (left) shows an illustration of this approach.

Fine-tuning in this manner is most suitable for sequence classification tasks (e.g., sentiment analysis, NLI, semantic similarity), sequence tagging tasks such as NER, and span extraction tasks (e.g., QA) in which the newly trained layers learn the start and end span of an answer. For sequence classification tasks, Devlin et al. [31] suggests fine-tuning BERT's representation of the special `[CLS]` token, and following with a single feed-forward layer that classifies it as one of the task labels. For

token-level or span-level classification tasks, the representations of each token, or alternatively just the representation of the first sub-token of each token or span (as in [31]), may be passed to the classifier. This fine-tuning approach is used to apply BERT to the GLUE tasks, QA, NER, and common-sense inference.

In this setting, care is needed to choose a learning rate that works for both the weights of the feed-forward layer(s) and for the PLM. Since the PLM is already largely trained, a low learning rate is appropriate (between 1e-3 [147] and 1e-5 [112]), with a lower learning rate for smaller datasets. In constrast, the randomly initialized, feed-forward layer weights require significant training. It is common practice to freeze the PLM layers temporarily while initially training the feed-forward layers, then unfreeze the PLM gradually for additional fine-tuning [66, 203]. If the majority of the labor is attributed to the `[CLS]` token, as in all the examples in Devlin et al. [31], there are fewer benefits to training the feed-forward layer alone. The next choice is how many layers of the PLM to fine-tune. While the examples in the BERT paper [31] fine-tune the entire model, this is not feasible for NLP tasks with small datasets or in situations where compute power is a limitation. Often, tuning just the top few layers of the PLM is sufficient; [81] find that when all the layers are fine-tuned, only the top layers change substantially in response to the task. A range of "BERTology" papers [20, 151, 176] show that the lower layers of BERT handle lexical and syntactic information such as part of speech, while the upper layers handle more semantic and increasingly complex information such as semantic roles and coreference, and more broadly become specific to the task [81].

2.4.3 Fine-tuning the PLM in Customized Models. Some tasks require significant additional architecture on top of a PLM, as illustrated in Figure 2 (center). With sufficient training data and computational power, researchers may choose to train both a substantial task-specific architecture and also fine-tune the PLM. This is the preferred choice for structure prediction tasks, in particular parsing tasks and occasionally sequence tagging tasks. Examples of sequence tagging models using this approach include BERT-CRF for NER [171, 174]. UDapter is an example of this approach applied to dependency parsing [182].

A related and highly successful approach is to fine-tune the entire PLM with a small number of feed-forward layers, then layer on an algorithmic approach that provides a substantial amount of task-specific heavy lifting. For example, the layered algorithm might transform the task from a classification problem into a desired target formulation, often a structured form such as a tree or a set of clusters. For coreference resolution, Joshi et al. [77, 78] adds the substantial e2e-coref algorithm [91], which transforms ratings of pairs of spans produced by the LM into valid mention clusters. Two more structural parsing examples are temporal dependency parsing [153] and modal dependency parsing [204]. These studies approach tree building algorithmically by first performing a classification problem to identify suitable dependency pairs, then ranking them to construct a valid tree.

2.4.4 Efficient Fine-tuning Approaches. A wide range of approaches, in addition to limiting fine-tuning to the top layers, seek to fine-tune only a small number of model weights. These can be classified into (a) fine-tuning only a small network that is tightly coupled with the PLM, and (b) selecting a small subset of the PLM's weights to fine-tune or keep.

The most prominent approach of the first type are adapter modules [10, 65, 137, 138], as illustrated in Figure 2 (right). Adapters add a small set of newly initialized weights at every layer of the transformer. Houlsby et al. [65] show that a two-layer feed-forward network with a bottleneck works well. The placement and configuration of the adapters within the Transformer blocks varies in the literature [10, 65, 138, 172]. During fine-tuning, all weights in the PLM remain frozen except for the few weights in the adapters. One set of adapters is fine-tuned per task of interest. This approach is more efficient in training (typically $< 5\%$ of all PLM weights), and allows efficient weight

sharing. Notably, the weights of adapters independently trained for different tasks can be successfully combined to solve a new task [138]. This practice also prevents catastrophic forgetting of old capabilities when fine-tuning on a new task or language. AdapterHub [137] and Trankit [124] are examples of frameworks promoting an adapter ecosystem; an example of using adapters for Universal Dependency Parsing is Üstün et al. [182].

Similar methods include side-tuning [211], which adapts a PLM by training a lightweight "side" network that is fused with the (unchanged) PLM using a simple additive process, and diff-pruning [55], which freezes the LM parameters and then adds and trains a sparse, task-specific difference vector to the original LM parameters, just 0.5% the size of the original model's parameters.

Moving to the second type of approach, BitFit [210] limits fine-tuning to the bias weights in each layer (or a subset of the bias weights, around 0.1% of the total parameters) of pre-trained BERT models, plus a task-specific classification layer. This is competitive with, and for some tasks better than, fine-tuning all of BERT. Similarly, Radiya-Dixit and Wang [146] show that fine-tuning only the "most sensitive" layers suffices, i.e., those most distant in parameter space from the rest of the model. In parallel, they sparsify the model substantially by setting 1–4% of pre-trained parameters to zero. This retains performance, as also demonstrated by work like DistilBERT [158] and other pruning studies ([140] inter alia), which show that many parameters in a large PLM are redundant. In fact, Zhao et al. [217] propose masking, i.e., setting weights to zero, as an alternative to fine-tuning the model weights. This approach freezes all the weights of the PLM, selects the weights that are relevant for a given task, and masks (discards) the rest. They train one mask per downstream task, with every layer masked except the embedding layer. While in principle this trains as many parameters as the original model, the mask is both binary and sparse and thus much simpler to learn and store. Zhao et al. show that masking yields comparable performance to fine-tuning on a range of tasks from POS tagging to reading comprehension.

One explanation for the effectiveness of these approaches is the *Lottery Ticket Hypothesis* [16, 105, 140], which suggests that a subnetwork within the model does most of the heavy lifting, and the remaining weights can all be pruned away. Prasanna et al. [140] find that *any* appropriately structured and sized subnetwork within BERT can be trained to perform well, though only some subnetworks perform as well as the whole model.

3 PARADIGM 2: PROMPT-BASED LEARNING

Prompting is the practice of adding natural language text or continuous vectors to the input or output to encourage PLMs to perform specific tasks. There are several advantages to using prompts. Prompting, especially in-context learning [39], may not require updates to the PLM's parameters, reducing computational requirements as compared to fine-tuning approaches. Prompts also encourage a better alignment of the new task formulation with the pre-training objective (e.g., masked text prediction), leading to a better use of the knowledge captured in the pre-training phase of the PLM. The closer match also enables strong zero-shot and few-shot performance [110], especially for tasks with small training datasets, as a good prompt can be worth hundreds of labeled data points [89]. Prompts enable probing of the PLMs, often in an unsupervised way, to assess the knowledge acquired by the PLM for specific tasks of interest (e.g., [136]).

We classify the existing prompting approaches into three main categories. Figure 3 shows illustrations for each of the three approaches.

We first introduce *Learning from Instructions and Demonstrations*, which uses task descriptions and examples to guide a PLM to perform the task. Very large causal models like GPT-3, without fine-tuning, are most commonly associated with this approach. Second, we describe *Template-based learning*, where labeled examples are recast into "natural" text using templates. These templates typically include slots to be filled with instance information or model outputs. Here we see Masked

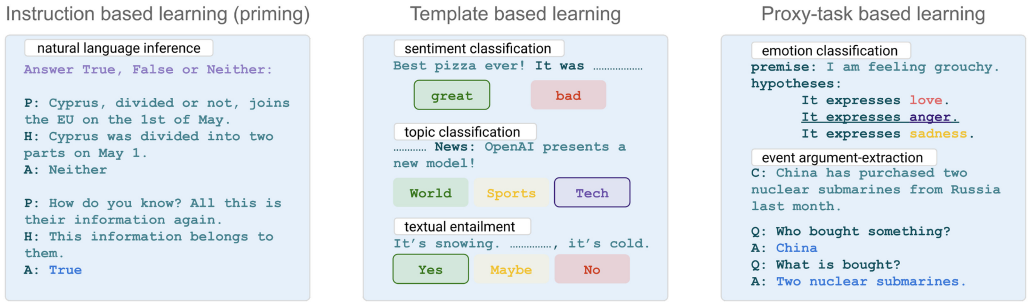

Fig. 3. The three main prompt-based approaches. On the instruction-based learning (left box) the instructions are marked in purple, the in-context examples in cyan, and the answer in blue. On the template -based learning (middle box), the text to classify is marked in light cyan and the prompt in dark cyan; the label verbalizations are shown in small boxes. On the proxy-task-based learning (right box), prompts are marked in dark cyan, the context is in light cyan, and the answers generated by the model are in blue.

or Denoising PLMs of relatively smaller size that are fine-tuned on the target task. Third, we discuss *Proxy-task-based learning*, where prompts recast target task examples into proxy-task examples. In this case the PLMs are adapted and fine-tuned to the proxy tasks (QA or TE) before application to the target task in the proxy-task format. The input of the proxy tasks is composed of a prompt (a question or QA and a premise for TE) together with the input of the original task.

3.1 Learning from Instructions and Demonstrations

Raffel et al. [147] is among the first attempts to use instructions such as "translate X to Y:" to simultaneously teach the model varied tasks in a text-to-text manner. However, this approach required a large amount of labeled data.

With the emergence of large generative PLMs [145], the first signs that LMs are multi-task learners emerged. For instance, GPT-2 understands that if the instruction "TL;DR" ("too long; didn't read") is given, then it should generate a summary of the context following the instruction. [39] shows that even larger generative PLMs are indeed good at few-shot learning. For example, GPT-3 can perform few-shot tasks via priming (in-context learning): given instructions and a few input/output pairs, GPT-3 produces the desired outputs for new inputs. No gradient updates are performed (see Figure 3, left box).

Regarding caveats of the approach, the very large size of GPT-3 is crucial to prompting's success in few-shot tasks, limiting its applicability. The typical context window of a few hundred tokens for GPT-3 and other models limits the scale (length) of the prompts and responses. We learn from [133] that the few-shot performance of PLMs is very sensitive to the choice of prompts, limiting the approach's robustness.

Schick and Schütze [166] and Reynolds and McDonell [150] introduce new tasks based on descriptions. For example, the text pair generation task [166] generates a continuation sentence given an input sentence and a description of the desired inter-sentence relationship (Table 2 (1)).[4] Schick and Schütze [166] use a generative PLM (GPT2-XL) to generate the continuation, replacing the __ token. Impressively, this approach even handles mathematical reasoning (Table 2 (2)): Reynolds and McDonell [150] show that inserting a natural language prompt ("Let's solve this problem by splitting it into steps.") enables GPT-3 to generate a procedure that solves the problem.

[4]In a variation, both sentences are generated, given only the description [166].

Table 2. Example Prompt Designs for Learning from Instructions

Input	Output
(1) Text pair generation [166]	
Task: Write two sentences that mean the same thing. Sentence 1: "A man is playing a flute." Sentence 2: __	"He's playing a flute."
(2) Mathematical reasoning [150]	
$f(x) = x * x$. What is $f(f(3))$? Let's solve this problem by splitting it into steps. __	$f(f(3)) = f(3 * 3) = 3 * 3 * 3 = 27$. We can see that $f(3) = 3 * 3 = 9$, so $f(f(3)) = 27$.

Wei et al. [195] showed that teaching a very large PLM to follow instructions with supervised data improves its zero- and few-shot abilities. They carried out a large-scale multi-task experiment over more than 60 datasets grouped into 12 different tasks. A PLM trained via natural language instructions outperformed a standard PLM on the test task. To study cross-task generalization, Mishra et al. [121] crowdsourced a dataset of instructions and instances for 61 NLP tasks, using crowdsourcing steps that are natural and intuitive for human annotators. They fine-tuned BART [96] in a similar fashion, giving instructions to the PLM that matched the step-by-step crowdsourcing instructions, decomposed into self-contained, separate tasks. This led to improved performance on unseen tasks, in contrast to an earlier work [36] that reported negative performance when using the crowdsourcing instructions as-is.

Scaling limitations may affect the broad applicability of this approach: Wei et al. [195] show that instruction tuning achieves significant improvements on held-out tasks in the zero-shot setting when using very large PLMs (e.g., with 68B or 137B parameters), but hurts performance when applied to PLMs with 10B parameters or less. In a similar setting, [41] showed that it is possible for a model with 11B parameters to benefit from instruction tuning, and identified three key differences compared to Wei et al. [195]. (1) They use an encoder-decoder model trained first with the MLM objective, then as a standard LM, and finally fine-tuned on a multitask objective, rather than a decoder-only autoregressive LM. (2) They argue that their prompts are qualitatively more diverse in terms of length and creativity. (3) They hold out multiple tasks at once, rather than only one at a time.

We note that the descriptions in instruction learning can be very detailed. For example, the crowdsourcing instructions in Mishra et al. [121] contain the task definition, things to avoid, emphasis and caution (i.e., required properties for the output), and positive and negative examples.

3.2 Template-based Learning

A more widely used approach, template-based learning, reformulates NLP tasks into tasks that are closer to language models' pre-training tasks via template-based prompts. This better leverages the knowledge captured in the pre-training tasks, leading to a significant reduction in the number of task-specific training examples required to achieve a similar performance to previous approaches [89], or even eliminating the need for training data. To achieve this goal, template-based learning reformulates various NLP tasks into LM tasks via carefully designed templates with open slots. In this way, solving the tasks is reduced to filling the slots with words or phrases using PLMs, and then projecting these outputs into the task-specific labels.

We consider *template-based* learning to be a subset of *prompt-based* learning, in that a less-detailed template is designed to exploit knowledge in the PLM, usually aligning the target task to the pre-training task. This is contrary to prior surveys such as **?**] where prompt-based learning is defined narrowly to be roughly equivalent to template-based learning.

Table 3. Example Prompt Designs for Template-based Methods

Input	Output
(1) Topic/sentiment classification [164]	
Best pizza ever!. It was __.	*great → Positive*
(2) Textual entailment [164]	
Mia likes pie? __, Mia hates pie.	*No → Contradiction*
(3) Event argument extraction [18]	
... U.S. troops killed 17 people in clashes earlier in the week. someone killed someone with something in some place at some time.	U.S. troops killed 17 people with something in Mosul at earlier in the week.
(4) Probing for relations/facts [136]	
Dante was born in __.	*Florence*
(5) Probing for commonsense [180]	
The trophy doesn't fit in the suitcase because it is too big	*it → trophy*: 0.9 *it → suitcase*: 0.2
(6) Probing for reasoning [175]	
The size of an airplane is __ than the size of a house . A. larger B. smaller	*larger*

great → Positive means that the answer *great* will be converted to label *Positive*. For (3), each underlined word (e.g., someone) will be replaced with the underlined phrase on the output side. "*it → trophy*: 0.9" means by replacing the underlined pronoun *it* with *trophy*, the modified sentence has a likelihood score of 0.9 according to the PLM.

Template-based learning differs from instruction learning (Section 3.1) in that templates do not explicitly describe the task and thus are often less detailed. Templates are typically designed to align the target NLP task with the PLM's pre-training task to explore the latent representation for improved sample efficiency.

3.2.1 Template Design. *Cloze-style prompts* convert inputs into a format for PLM prediction of missing word(s). Table 3 (1) shows a straightforward example of this approach, as applied in the sentiment detection domain.

For classification tasks, each predicted word or phrase is converted into a class label of interest. For example, we can design a cloze-style prompt for a TE task in which the goal is to predict the *entail/contradict* relation between a pair of input sentences. **Pattern-Exploiting Training (PET)** [164] (Table 3 (2)) converts a pair of inputs $\langle X_1, X_2 \rangle$ into the template "X_1? __, X_2" and asks an MLM to predict the missing word (the first word of the second sentence). The prediction (here *yes* or *no*) is directly mapped to one of the TE class labels. This template design reformulates the text entailment problem into the same masked LM problem used to pre-train the PLM. Therefore, it is popular among classification tasks that may be reformulated as predicting a masked word or short phrase (e.g., topic classification, TE, and knowledge probing). Chen et al. [18] reformulate the event argument extraction challenge as a cloze-style problem (Table 3 (3)). They predict fillers for the underlined positions, then apply greedy decoding to fill in the __ position incrementally. Petroni et al. [136] similarly use the cloze-style task for relation/fact probing (Table 3 (4)).

A *multiple-choice style prompt* proves useful when probing for commonsense knowledge. This kind of template provides a selection of hypotheses for the PLM, which selects its preferred answer. For example, in Table 3 (5), Trinh and Le's [180] model selects *trophy* instead of *suitcase* to replace it in the original sentence. Table 3 (6) shows work by Talmor et al. [175], expressing similar reasoning through an hypothesis-driven approach.

Prefix prompts [57, 92, 104] are another common type of template. Prefixes are task-specific vectors prepended to the input. They do not correspond to actual words but consist of free parameters. Prefix prompts are usually the best choice for tasks that require generating text or predicting a next word or phrase, because the prefix-based prompt design is consistent with the left-to-right nature of the autoregressive model [110].

Prompts can be further augmented via adding demonstrations (*demonstration learning*) [51]. A few labeled examples are appended to the template to make it more informative, improving the PLMs' responses.

3.2.2 Template Construction. Templates can be either manually crafted or automatically generated. We survey methods for generating, combining, and manipulating template-based prompts.

Manually Crafted Templates. Most early work in prompt-based learning explored manually crafted templates. For example, Petroni et al. [136] use manual cloze templates to probe the knowledge of the model. Schick and Schütze [162], Schick et al. [160] and Schick and Schütze [164] handcraft cloze templates for text classification in a few-shot setting. Brown et al. [39] leverage manually designed prefix prompts for QA, translation, and probing tasks for commonsense reasoning. The quality of the prompts impacts performance. Indeed, Zhao et al. [219] showed that different prompts can cause accuracy to vary from near chance to near state of the art.

Automatically Generated Discrete Templates. Discrete templates usually refer to natural language phrases. To search for templates appropriate for a set of inputs and outputs, Jiang et al. [75] describe a mining-based approach called MINE that aims to find either the middle words or dependency paths between the inputs and outputs. [75, 208] paraphrase an existing template prompt using back and forth MT, then select the best prompt among the new paraphrases with guidance from a thesaurus. Prompt paraphrasing is also explored by Haviv et al. [60] who used a neural prompt rewriter that optimizes the accuracy of systems using the prompt. In that case, a different paraphrase is generated for each input. A third approach uses gradient-based search to find short sequences that can serve as prompts [170, 185]. Gao et al. [51] and Ben-David et al. [11] further generate prompts using generation PLMs such as T5. In the latter study, the authors proposed a domain adaptation algorithm to train T5 to generate unique, domain-relevant features that can be concatenated with the input to form a template for downstream tasks.

Automatically Generated Continuous Templates. Continuous prompts, which perform prompting directly in the embedding space of the model, allow us to abstract away from natural language prompts and from the parameters of the LM [110]. These continuous prompts often require tuning on task-specific data. Li and Liang [104] propose prefix tuning, which prepends a sequence of continuous, task-specific vectors to the input while keeping the LM parameters frozen. This allows them to fine-tune just 0.1% of the total model parameters. A similar method is used by Lester et al. [92], who differ from Li and Liang [104] by adding special tokens to form a template and tuning the embeddings of these tokens directly, without introducing additional tunable parameters within each network layer. Continuous prefix tuning is also used by Tsimpoukelli et al. [181] in the context of multimodal learning (language and vision), but in that case the prefix is sample dependent. Tuning can be initialized with discrete prompts [57, 142, 221], or by inserting tunable embeddings into a hard prompt template as in Liu et al. [111] and Han et al. [59], who propose **prompt tuning with rules** (**PTR**). This uses manually crafted sub-templates to compose a complete template using logic rules (see Section 3.2.5 for its application to relation extraction).

Logan IV et al. [113] showed that fine-tuning PLMs in the few-shot setting can prevent prompt engineering. Prompts that contain neither task-specific templates nor training examples, and even *null prompts* that are simple concatenations of the inputs and the [MASK] token, still achieve competitive accuracy on NLU t asks.

Multi-prompt Learning. A number of approaches use prompt ensembling, augmentation, and decomposition/composition for a more flexible task design. The use of multiple prompts at inference time has been dubbed *prompt ensembling*. The prompts can be combined using a

uniform average [75, 164, 208] or a weighted average [75, 142, 164, 165]. Another way to combine the prompts is majority voting [57, 92]. Knowledge distillation [2], where knowledge present in an ensemble of models is distilled into a single model, has been borrowed to the context of prompt combination by [163–165] and by Gao et al. [51], who train a separate model for each template-answer pair before ensembling them to annotate an unlabeled dataset. Then, the authors train a new model to distill the knowledge from the annotated dataset. In the case of generation tasks, Schick and Schütze [163] trained a separate model for each prompt. The model outputs were scored by averaging their generation probability across all models.

Prompts can be decomposed or composed to more effectively solve an NLP task. Decomposition involves finding sub-problems for which prompts can be generated separately. An example in the NER domain is Cui et al. [26], who create and separately predict different prompts for each candidate span.

Augmentation methods such as *demonstration learning* [51] create more descriptive prompts, as in a multiple-choice problem. Lu et al. [114] showed that both the choice of examples in the prompts and the order of the prompts can considerably affect the results. Automated *example sampling* (Gao et al. [51], Liu et al. [109]) uses sentence embeddings to find examples semantically close to the input. Mishra et al. [121] used both positive and negative examples, teaching the PLM types of items to avoid in performing new tasks with only instructions. As for *sample ordering*, Kumar and Talukdar [83] searched for the best permutation of prompts and also learned a segmentation token to separate the prompts. They showed the utility of this method for few-shot learning on sentiment classification.

3.2.3 Answer Generation. There are two main types of answers to prompts: those that map to a classification label (e.g., [26, 205]), and those intended as the final answer (e.g., [74, 136, 145]). For classification tasks, typically addressed with cloze-style prompts, the developers identify a subset of words and phrases from which the PLM may choose, and that choice is easily mapped to the class of interest. For instance, in a sentiment detection task, the PLM may answer a prompt with "good," "great," or "excellent," all of which are mapped to a "positive" sentiment label. The second type of answer, free text, prevails for text generation tasks. Examples of both types are shown in Table 3.

In either case, the definition of the answer space may be optimized to produce ideal prompt responses. Jiang et al. [75] used *paraphrasing* to extend the search space with back-translation (translating to another language, then back to the original). Another approach, explored by Schick and Schütze [164], Schick et al. [160], Shin et al. [170], and Gao et al. [51], is *prune-then-search*, a two-step method where the answer space is pruned, for example by only selecting a subset of words according to their zero-shot accuracy on the training data [51] and then an answer is searched for in the pruned space. An approach called *label decomposition* optimizes the search space by modeling the label names for comparison to the answer tokens; for instance, in Chen et al. [17] the decomposed relation labels (their individual tokens) represent the answer space. Hambardzumyan et al. [57] add a virtual token for each class label and optimize its embedding together with the token embeddings of the prompts, using gradient descent. This *gradient descent optimization* approach allows direct optimization of the answers instead of using a discrete search.

3.2.4 Task-Specific Tuning. While prompts can be directly used in a zero-shot, unsupervised setting, prompts have also been used in fully supervised or few-shot settings where either all or part of the specific-task training data is available. Two main approaches currently prevail for tuning a PLM with prompts.

The first approach uses a fixed template-style prompt to perform tuning of the PLM. Here, a fixed template is applied to every training and test example, as in the PET-TC [164], PET-Gen

[163], and LM-BFF [51] models. Le Scao and Rush [89] quantified the benefit of using prompts in classification tasks by fine-tuning in equal conditions across many tasks and data sizes. They showed that prompting consistently improves the results across tasks over just fine-tuning, that it is most robust to the choice of pattern, and that it can be learned without an informative verbalizer (a function that maps each label to a single vocabulary token). Logan IV et al. [113] showed that only tuning 0.1% of the parameters in the prompt-based few-shot setting can achieve comparable or better accuracy than standard fine-tuning. For this purpose, they explored different ways to perform memory-efficient fine-tuning, including (i) Adapters [65], which are neural network layers inserted between the feed-forward portion of the Transformer architecture (see Section 2.4.4); (ii) BitFit [210], which updates only the bias terms inside the Transformer; (iii) PLM head tuning, which updates the embeddings in the MLM output layer that are associated with the tokens of the verbalizer; and (iv) Calibration [219], which learns an affine transformation on top of the logits associated with the verbalizer tokens. They found that BitFit achieves the best results.

The second approach is joint tuning of the prompt and the PLM. Here, prompt-relevant parameters are fine-tuned together with the all or some of the parameters of the PLM, as in PADA [11] and P-Tuning [111]. In PADA, the prompts are properties of source domains, generated based on their relatedness to the input example (from a new domain). P-Tuning uses trainable continuous prompt embeddings when applying GPT models on NLU tasks. Fine-tuning both the model and the prompt-relevant parameters makes this approach very expressive. On the other hand, it requires storage of all parameters, which makes it less applicable to small datasets [110].

Task-specific training can be used earlier during the construction and validation of the prompts. As pointed out by Perez et al. [133], previous PLM-based few-shot learning approaches used many held-out examples to tune various aspects of learning, such as hyperparameters, training objectives, and natural language templates ("prompts"). Perez et al. [133] propose instead to evaluate the few-shot ability of PLMs in a *true few-shot learning* setting, where such held-out examples are unavailable.

3.2.5 Applications of Template-based Methods. Template-based prompting methods are currently applied to a growing list of NLP tasks. We provide a survey of how recent studies have addressed a varied set of NLP applications.

Text Classification. In Puri and Catanzaro [141], natural language descriptions of classification tasks were given as input. Then, the model was trained to generate the correct answer in natural language via an LM objective, aiming to generalize to new classification tasks without task-specific tuning.

IE. Cui et al. [26] considered the NER task as an LM ranking problem in a sequence-to-sequence framework where the source sequence corresponds to the original sentence and the target sequence corresponds to the template prompt, filled by candidate spans. For the relation extraction task, Han et al. [59] proposed a model called PTR, which applies logic rules to construct prompts with several sub-prompts, each corresponding to an NER span of interest. Chen et al. [17], instead of using rules, constructed the prompts by leveraging learnable virtual template words and virtual answer words. Their representation is synergistically optimized with knowledge constraints. For the event extraction task in a cross-lingual setting, Fincke et al. [46] proposed using the event type and an integer representing the argument type as prefixes.

Knowledge Probing. Factual probing has been explored by Petroni et al. [136] and Jiang et al. [74] to quantify the amount of factual knowledge already present in the PLMs, providing the LAMA and X-FACTR datasets, respectively. Other works that investigated model knowledge with discrete template search include Petroni et al. [135], Jiang et al. [75], Haviv et al. [60], Shin et al. [170], and

Perez et al. [133]. Continuous template learning was used in Qin and Eisner [142], Liu et al. [111], and Zhong et al. [221]. Prompt ensemble learning was applied to knowledge probing by Jiang et al. [75] and Qin and Eisner [142].

In addition to factual knowledge, additional types of knowledge that have been probed using the cloze test include commonsense [180], relational knowledge [136], reasoning [175], and understanding rare words [161]. For commonsense reasoning, Winograd Schemas [93] require the model to identify the antecedent of an ambiguous pronoun within context, or involve completing a sentence given multiple choices. For commonsense knowledge mining, Feldman et al. [44] construct a candidate piece of knowledge as a sentence, then use a language model to approximate the likelihood of the text as a proxy for its truthfulness.

Prompts can also be used to explore the linguistic knowledge of PLMs, focusing on different phenomena such as analogies [39], negation [43], or semantic similarity [42]. Linguistic evaluation of LMs [7, 53, 54, 106, 107, 118, 179] usually considers minimal pairs of grammatical and non-grammatical sentences addressing a specific phenomenon that differs in a single place in the sentence. To succeed, a model must score the grammatical sentence higher than its ungrammatical counterpart. A main resource in this context is **BLiMP** (**Benchmark of Linguistic Minimal Pairs**) [192], which provides minimal pairs for various grammatical phenomena. The use of this benchmark was adapted for language acquisition research [69]: the authors probe a RoBERTa-based model pre-trained on transcriptions of child-directed speech [117] to complete the benchmark task. The preference score may be calculated *holistically*, summing the cross-entropy errors at each position in the sentence [69, 209]. Alternatively, an *MLM-based* approach computes the score by summing the log losses of multiple masking attempts at varying indices in the sentence [157].

Other Tasks. The PET procedure [164] was also applied to the TE task. QA is addressed in Khashabi et al. [80] with appropriate prompts from the context and questions, formulating several QA tasks into a unified text generation problem with encoder-decoder pre-trained models such as T5.

Prompts have also been applied to the evaluation of text generation. Yuan et al. [208] used prompts in the BARTSCORE-PROMPT variant of the BARTSCORE measure, which treats the evaluation of various text generation tasks as a generation problem. In BARTSCORE-PROMPT, prompts are either appended to the source text or prepended to the target text and are shown to be useful. For example, adding the phrase "such as" to the translated text when using pre-trained models significantly improves the correlation with human evaluation on German-English MT evaluation.

Schick et al. [167] showed that PLMs are able to recognize the toxicity of the text they produce (self-diagnosis). They propose an algorithm that permits the language model to produce less problematic text (self-debiasing) by providing a textual description of the undesired behavior.

Shin et al. [169] explore the use of PLMs as few-shot semantic parsers. The authors use GPT-3 to convert text into a controlled, canonical text that satisfies a grammar. This outcome is automatically mapped to the target structured meaning representation.

3.3 Learning from Proxy Tasks

Templates and prompts play a role again in an indirect approach to NLP tasks called "proxy tasks." Examples for the use of this approach are emotion classification or event and argument extraction, both shown in Figure 3 (right box) with prompt-based proxy tasks. See Table 4 for additional examples of proxy tasks and prompt design.

The key distinction between learning from proxy tasks and previous methods is the use of supervised **Natural Language Understanding** (**NLU**) tasks as a proxy the target task. Indeed, taking

Table 4. Examples of Task Design and Example Prompts for Four Different Applications of Prompt-based Proxy Tasks

Application	Work	Task design	Prompt design
Relation Extraction	Li et al. [103]	Use question-answering to identify the most appropriate entity span, given an incomplete text and an indication of the class type.	Input: The armory is north of the music center. Prompt: Find a facility near E_1? E_1, physical, facility.
	Sainz et al. [155]	Use textual entailment to determine the likelihood of a candidate relation (such as PlaceOfDeath(X,Y) given an input sentence.	Input: Gary's car crash occurred in Houston; Prompt: Gary died in Houston.
Event Extraction	Du and Cardie [33]	Use a series of ordered questions, each leveraging the output of the previous answer, to find event triggers and appropriate arguments.	(1) Input: Donna purchased a new laptop; Prompt: What is the trigger? purchased (2) Prompt: What was purchased? laptop
Topic and Sentiment Classification	Yin et al. [205]	Use textual entailment to determine whether a topic name T is suitable for a text.	Input: Dinosaurs and humans never coexisted. Prompt: This text is about T.
	Puri and Catanzaro [141]	Use QA to probe for a topic or sentiment name from among a closed set of responses.	Input: Dinosaurs and humans never coexisted. Prompt: How is the text best described? T_1, T_2, or T_3
Coreference Resolution	Wu et al. [198]	Use question-answering to find a coreferent mention of a marked mention from within the same text.	Input: I arrived at the party with my tux on, and introduced myself as George. I told them that <mention> I </mention> was hired to do some Christmas music; Prompt: Who does it I refer to?

advantage of large NLU datasets for extra supervision results in better zero- and few-shot performance in the target task with relatively small PLMs [188], commonly $\text{RoBERTa}_{\text{large}}$ at 345M parameters. Knowledge-rich classification tasks benefit from PLM proxy tasks because in reformulating the class label as a prompt, they take advantage of the meaning of class labels instead of treating them as indices. In this section, we describe the main proxy-task-based learning approaches using QA (Section 3.3.1) and TE (Section 3.3.2).

3.3.1 QA as Proxy Task. The choice of using QA as a proxy task is motivated by the relative ease of answering simple questions, as compared to performing expert annotation for complex linguistic phenomena. In a strong move away from traditional IE, recent studies replace modeling of explicit entity, relation, and event classes with natural language questions that get at the exact item of interest. Questions can be used to probe for the required information in the text.

In IE tasks, question prompts typically address identification and classification jointly, by constructing the question to identify a particular type. For example, the question "Who bought something?" will produce an answer specific to the *Buyer* argument role in an event of type Exchange-Ownership (see Figure 3, right box). Li et al. [102] formulates **NER** as a QA problem. For example, the prompt "which person is mentioned in the text?" will identify a mention classified as a PERSON. The proposed BERT-based system performs detection of multiple spans through the use of separate binary classifiers identifying start and end tokens. The authors incorporate synonyms and examples into the queries. Wu et al. [198] formulated **coreference resolution** as a span prediction task via QA, where a query is generated for each candidate mention using its surrounding context, and a span prediction module uses the query to extract the coreference spans in the document.

Levy et al. [95] first formulated relation extraction as a QA task. This approach has been pursued in the context of PLMs by Li et al. [103] and Zhao et al. [218]. Han et al. [59] addresses relation extraction with sub-prompts for **entity recognition** and **relation classification**, composing them

into a complete prompt using logic rules. Both types of questions are used to probe a QA system in a supervised setting to perform the two sub-tasks. Task decomposition is also used in the work of Zhou et al. [225] for event extraction where natural questions for **argument identification** ("What plays the role?") and **argument classification** ("What is the role?") mutually improve each other.

Chen et al. [18] reformulated **event extraction** as a cloze task with QA model based on BERT and the SQuAD 2.0 dataset [148]. QA is used directly, preserving the QA format, in Du and Cardie [33], Feng et al. [45], Li et al. [97], Zhou et al. [225], and Liu et al. [108] for argument extraction, including the argument identification and classification sub-tasks. In these cases the event extraction training data is converted to the QA format, where the questions are derived from the ontology. Liu et al. [108] also experimented in a zero-shot setting where no task-specific data is used for training, only using prompts for probing. The zero-shot setting for the full event extraction pipeline has been explored in Lyu et al. [116] where QA-based prompts are used for argument extraction and prompts based on Textual Entailment [27] are used for trigger classification (see Section 3.3.1 below). Several ablation experiments analyzed the different components of the system such as the choice of PLM, the choice of QA dataset, and the way to generate the questions (fixed vs. contextualized). It was shown in particular that RoBERTA trained on QAMR [119] achieved the best results for argument extraction. Identification-only sub-tasks such as **trigger identification** [33], are addressed by more general questions, e.g., "What is the trigger?". In contrast, Zhou et al. [225] uses separate questions to address the identification and classification of arguments.

Du et al. [34] addressed **slot filling**, which aims to extract task-specific slot fillers (e.g., a flight date) from user utterances by formulating it as a QA task. In particular, they addressed the zero-shot slot-filling problem, where the model needs to predict spans and their values, given utterances from new, unsupervised domains. Extracting slot-filler spans from utterances with QA improved the performance, compared to a direct encoding of the slot descriptions.

Gao et al. [50] formulated the **dialogue state tracking** as a QA problem. This task aims to estimate the current belief state of a dialogue given all the preceding conversation. The proposed system uses a simple attention-based neural network to point to the slot values within the conversation. This direction was pursued by Gao et al. [49] who also included a multiple-choice setting, where several candidate values for each slot in the question are given. The latter setting was also investigated by Zhou and Small [223] who further improved the results. Namazifar et al. [122] used this approach to address language understanding problems in the dialogue context.

QA Task Design. Questions are typically generated via handcrafted templates derived from task-specific ontologies. Some of the works introduce contextualization, integrating relevant words from the text into the question. For example, in argument extraction, the question can include the trigger extracted from the text (e.g., [108, 116]) or another argument that was previously identified [97] (see the Event Extraction row in Table 4). Neural-based question generation models can also improve the quality of the question, as in Liu et al. [108], where monolingual unsupervised MT [87] is used to generate the part of the question that does not depend on the template, translating a descriptive statement into a question-style expression.

Other aspects of QA-style proxy tasks are the ability to use multiple questions, and to formulate questions in any style. In addition to sequential questions for determining event arguments, multiple formulations of the same question may be used in a weighted voting scheme to generate an ensemble answer (Zhao et al. [218]). The input to the QA system need not include natural questions. It may instead consist of pseudo-questions such as keywords, synonyms, position index of labels, or a single word/type from the ontology or annotation guidelines (e.g., [33, 102]).

PLMs fine-tuned on the SQuAD 2.0 dataset [148] or on QAMR [61] are particularly useful to initialize QA-style prompt-based learning methods. With the advent of web-scale QA datasets [68],

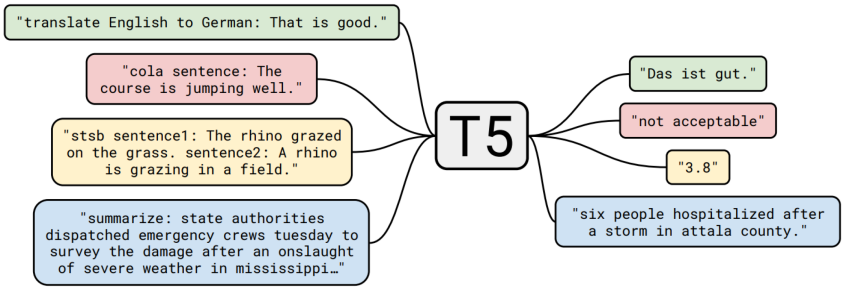

Fig. 4. An illustration of T5 [147] text-to-text text generation approach for MT, linguistic acceptability, text semantic similarity, and summarizing tasks. Figure source: Raffel et al. [147].

QA-infused PLMs may provide significantly richer representation, enabling a wider range of applications.

3.3.2 Textual Entailment as Proxy Task. TE is a popular proxy for classification tasks [205], as these models have shown a striking ability to perform few-shot learning. Wang et al. [188] hypothesizes that this phenomenon might be because the entailment task is a true language understanding task; a model that performs entailment well is likely to succeed on similarly framed tasks. An example of TE as a proxy for **emotion classification** is shown in Figure 3, while an example of its use for **topic detection** is shown in Table 4.

For entailment prompting, developers define a template that describes the task, and create a natural language version ("verbalization") of each potential label. Multiple hypotheses for entailment are produced by inserting the potential labels into the template. Inference is performed by selecting the most probable candidate hypothesis given the input. Recent works also make use of multiple verbalizations for each label to boost the performance [155, 156]. Sainz et al. [155] proposed an approach to guiding the "art" that is prompt crafting more towards a "science": the authors fine-tune a model on TE data and use its probability of a prompt given the template, applied on the guideline example(s), to measure the quality of manually designed prompts. Sainz et al. [155] reformulated **relation extraction** as a TE task, leveraging PLMs. Roughly equivalent to TE is Yes/No QA [19] where a model is asked about the veracity of some fact given a passage. It was also used as a proxy task for text classification by Zhong et al. [220].

PLMs needs to be fine-tuned to solve the TE task. They are commonly fine-tuned on MNLI [197], but other datasets such as SNLI [14], FEVER [177], ANLI [126], or XNLI [23] are also used. In addition, data from different tasks can be used when framed properly [220].

4 PARADIGM 3: NLP AS TEXT GENERATION

The success of generative PLMs[5] such as GPT, BART, and T5 have recently sparked interest in leveraging generative PLMs to solve various non-generative NLP tasks. These tasks include, but are not limited to, traditional discriminative tasks such as classification and structure prediction. These discriminative tasks are reformulated as text generation problems so that they can be directly solved with a "text-to-text" PLM. For example, Figure 4 illustrates the "text-to-text" approach as described in Raffel et al. [147]. The generated text usually includes the desired labels or other auxiliary information, enabling accurate reconstruction of the expected class labels (i.e., to avoid ambiguities in mapping).

[5]We use the term generative PLMs to refer to any PLMs that can generate text. Typical generative PLMs include autoregressive (e.g., GPT) that can generate the continuation of text given a prefix, and sequence-to-sequence PLMs that can take some text as input and generate text as output. In this section, we use the term PLM to refer to a generative PLM.

Table 5. A Summary of Methods Reformulating NLP Task as a Generation Task Solved by PLMs

Output Type	Work	Task	Example	
			Input	Output
Label-augmented Text	Paolini et al. [130]	Joint Entity and Relation Extraction	Tolkien's epic novel The Lord of the Rings was published in 1954-1955.	[Tolkien | person]'s epic novel [The Lord of the Rings | book | author = Tolkien] was published in 1954-1955
		Relation Classification	Born in Bologna, Orlandi was a student of the famous Italian [soprano] and voice teacher [Carmen Melis] in Milan. The relationship between [Carmen Melis] and [soprano] is	relationship between [Carmen Melis] and [soprano] = voice type
		Semantic Role Labeling	The luxury auto maker last year [sold] 1,214 cars in the U.S.	[The luxury auto maker | subject] [last year | temporal] sold [1,214 cars | object] [in the U.S. | location]
		Event Extraction	Two soldiers were attacked and injured yesterday	Two soldiers were [attacked | attack] and [injured | injury] yesterday
		Coreference Resolution	Barack Obama nominated Hillary Rodham Clinton as his secretary of state on Monday.	[Barack Obama] nominated [Hillary Rodham Clinton] as [his | Barack Obama] [secretary of state | Hillary Rodham Clinton] on Monday
		Dialogue State Tracking	[user] : I am looking for a cheap place to stay [agent] : How long? [user] : Two	[belief] hotel price range cheap, hotel type hotel, duration two [belief]
	Athiwaratkun et al. [5]	Slot Filling	Add Kent James to the Disney soundtrack	((AddToPlaylist)) Add [Kent James | artist] to the [Disney | playlist]
		NER	He is John Wethy from NBC News	He is [John Wethy | person] from [NBC News | org]
	Zhang et al. [213]	Aspect Opinion Pair Extraction	Salads were fantastic, our server was also very helpful.	[Salads | fantastic] were fantastic, our [server | helpful] was also very helpful.
		Aspect Sentiment Triplet Extraction	The Unibody construction is solid, sleek and beautiful.	The [Unibody construction | positive | solid, sleek, beautiful] is solid, sleek and beautiful.
		Target Aspect Sentiment Detection	The pizza was cold.	The [pizza | food quality | negative] was cold.
Generating Word Indices	Yan et al. [202]	NER	have muscle pain and fatigue	2 3 7 2 5 6
	Yan et al. [201]	Aspect Term Extraction Opinion Term Extraction Aspect-level Sentiment Classification Aspect-oriented Opinion Extraction	The wine list is interesting and has good values, but the service is dreadful	1, 2, 12, 12 4, 4, 7, 8, 14, 14 1, 2 , Positive 1, 2, 4, 4, 7, 8
	Rongali et al. [152]	Slot Filling	play the song don't stop believin by journey	PlaySongIntent SongName(@pt r3 @pt r4 @pt r5) SongName ArtistName(@pt r7)ArtistName
Generating Answers	Wang et al. [187]	Closed-book QA	What is Southern California often abbreviated as?	Southern California, often abbreviated SoCal, is . . . ANSWER SoCal
	Hsu et al. [67]	Answer Selection	How a water pump works?	A water pump is a device that moves fluids by mechanical action.
Filling Templates	Du et al. [35]	Event Extraction	[CLS] Attack, Bombing, Arson, . . . [SEP_T] (Document tokens): Several attacks were carried out in La Paz . . . [SEP]	[CLS] Attack -T1 REEs- [SEP_T] Bombing -T2 REEs- [SEP_T]
	Li et al. [100]	Event Argument Extraction	Elliott testified that on April 15, McVeigh came into the body -tgr- reserved -tgr- the truck, to be picked up at 4pm two days later shop and	Elliott bought, sold or traded truck to McVeigh in exchange for $280.32 for the benefit of -arg- at body shop place
Structure-linearized Texts	Ren et al. [149]	Joint Entity and Relation Extraction	He was captured in Baghdad late Monday night	"He" Type PER [SEP] "Baghdad" Type GPE PHYS "He"
	Lu et al. [115]	Event Extraction	The man returned to Los Angeles from Mexico	((Transport returned (Artifact The man) (Destination Los Angeles) (Origin Mexico))
Ranking Input-output Pairs	Nogueira dos Santos et al. [128]	Answer Selection	<bos>Ice formations in the Titlis glacier cave <boq>How are glacier cave formed <eoq>	0.5
	Nogueira et al. [127]	Document Retrieval	How are glacier cave formed [Q] A glacier cave is a cave formed within the ice of a glacier [D]	True
	De Cao et al. [30]	Entity Retrieval	Superman saved [START] Metropolis [END]	Metropolis (comics) | Metropolis (1927 film)
	Cui et al. [26]	NER	ACL will be held in Bangkok	Bangkok is a location

Although both *NLP as text generation* and *prompt-based learning* leverage the text generation capability of PLMs, they are significantly different. *NLP as text generation* reformulates NLP problems directly as a "text-to-text" problem aiming at generating the right output in a linearized, string form that encodes the expected classification labels or structures (see Table 5 for examples). *Prompt-based learning*, on the other hand, aims at using prompts to align the target NLP task better with the LM pre-training tasks such as MLM to better exploit the latent representation of PLMs for improved few-shot performance.

Some NLP tasks are inherently text generation tasks. In these cases, a straightforward strategy is to fine-tune a generative PLM using task-specific training data to perform the tasks of interest. Examples include MT [24], text summarization [96], text style transfer [86], and so forth. We refer readers to Section 2 for more detailed discussion of this "pre-train then fine-tune" approach. In this section, we focus on tasks that are not traditionally text generation tasks.

Reformulating NLP Tasks as Text Generation Problems. Pre-trained from large corpora, PLMs demonstrate an extraordinary ability to generate text. PLMs also capture rich knowledge useful for many NLP tasks and show strong performance on learning new patterns via fine-tuning. These factors lead to the hypothesis that many NLP tasks can be reformulated as text generation problems. In particular, given an NLP task with an input text x, this approach first attempts to design an

output sequence y that includes information about the desired labels for x (e.g., markers). Then, a PLM directly generates y, conditioning on the input x, modeling $P(y|x)$. To train the PLMs, the original training data of the NLP task is first converted into pairs (x, y) following the designed format. The PLMs are usually fine-tuned with such pairs using the standard maximum likelihood loss.

There are a few advantages of this approach. First, in this formulation, a unified seq2seq framework can solve different NLP tasks via encoder-decoder architectures, thus facilitating multi-task learning and transfer learning across tasks of different nature [147]. Second, the direct generation of labels in output sequences allows the PLMs to exploit the semantics of the labels to improve the performance and data efficiency, a benefit that cannot be achieved in discriminative models [130]. Finally, when adapting to structure prediction problems, PLM-based models can naturally capture the inter-dependencies between prediction steps or tasks to further improve the performance [5].

The careful design and format of y is critical for performance: the desired labels/outputs x must be retrieved unambiguously from y. In addition to the label information, evidence useful for providing context can also be incorporated into the formulation of y to aid the generation process. Existing works tend to customize such output sequences for specific NLP tasks to better capture the nature of the tasks. Therefore, in this section, we group prior works according to their strategies in designing successful output sequences. Table 5 provides a brief summary.

4.1 Generating Label-Augmented Texts

In this strategy, the output sequence y copies the input text x and augments it with additional markers that can be decoded into the desired label annotations for x. The repetition of the words from x aims to provide explicit context to reduce ambiguity for the generation process [130]. This strategy is often applied to structure prediction tasks that aim to jointly extract the text spans of interest and their relations or dependencies.

Athiwaratkun et al. [5] explores the idea of label-augmented text generation for sequence labeling problems, e.g., slot filling (identifying spans that define the left or right "slot" of a relationship) and NER. Given an input x, y is formed by marking the token sequence for the slots or entity types of interest, for instance with square brackets. The corresponding labels are introduced within the brackets, separated from the token by a bar "|". The encoder-decoder PLM T5 is used to generate label-augmented texts. Paolini et al. [130] extends this idea to other structure prediction tasks, including joint entity and relation extraction, relation classification, SRL, event extraction, coreference resolution, and dialogue state tracking. To encode a relation between two text spans in the input text, the second text span might be annotated with both the relation label and an indicator of the first text span.

For example, for the joint entity and relation extraction task, we transform x into the label-augmented output y, where (1) square brackets indicate token spans for entity mentions; (2) `person` and `book` are entity-type labels; and (3) `author=Tolkien` indicates the author relation between `Tolkien` and `The Lord of the Rings`:

```
x = Tolkien's novel The Lord of the Rings was published in ...
y = [Tolkien|person]'s novel [The Lord of the Rings|book|author=Tolkien] was published in ...
```

In order to transform the generated label-augmented texts into desired annotations, Paolini et al. [130] uses dynamic programming to match the generated output sequence and the input text, searching for the closest entity mention that exactly matches the predicted tail entity and discarding invalid entity/relation types. Similarly, Zhang et al. [213] utilize label-augmented text generation for different variations of **aspect-based sentiment analysis** (**ABSA**), including aspect opinion pair extraction, unified ABSA, aspect sentiment triplet extraction, and target

aspect sentiment detection. Zhang et al. [213] also propose a normalization prediction mechanism: if a generated token does not belong to the original sentence or the label set, the closest word from the input sentence using Levenshtein distance is used instead.

Due to the unified text-to-text formulation, label-augmented text generation enables multi-task learning. Paolini et al. [130] and Athiwaratkun et al. [5] show that learning multiple tasks with a single model can improve the performance on the individual tasks. Furthermore, label-augmented text generation also shows impressive performance in few-shot learning settings [130], improving the data efficiency.

4.2 Generating Word Indices

For many text understanding problems (e.g., span tagging problems such as NER), the generative PLM must not generate non-input tokens other than markers or labels, as shown in the example in Section 4.1. Restricting the PLMs to consider only words in the input text as candidates at decoding (text generation) time enforces this constraint.

An alternative approach is to directly generate *indices* of the words of interest in the input text. Given the input x, the output sequence y provides a sequence of index numbers referring to the positions of words in x. Label indices encode class labels within y. A few examples are included in Table 5 in the "Generating Word Indices" rows.

Yan et al. [202] explores an index generation idea for NER that can naturally handle different settings, e.g., flat, nested, and discontinuous NER. Given the input sequence $x = [x_1, x_2, \ldots, x_n]$, the output sequence y is formed via the indices: $y = [s_{11}, e_{11}, \ldots, s_{1_{k_1}}, e_{1_{k_1}}, t_1, \ldots, s_{i1}, e_{i1}, \ldots, s_{i_{k_i}}, e_{i_{k_i}}, t_i]$ where s and e indicates the start and end indexes of a span. The spans for the i-th name in x are represented by the tuple $[s_{i1}, e_{i1}, \ldots, s_{i_{k_i}}, e_{i_{k_i}}, t_i]$ where t_i is the index of the entity type and k_i is the number of text spans for the i-th name (a name can have multiple spans due to the consideration of discontinuous names). As such, s_{i_j} and e_{i_j} should be between 1 and n while the entity types can be indexed from $n + 1$ (i.e., $t_i > n$). To compute the hidden vectors at decoding time, the representations for the span indices can be obtained from the representations of the corresponding words in the input sentence x (i.e., via pointer networks [184]). BART is used as the base model for the index generation for NER.

Similarly, Yan et al. [201] generates indices of the spans of interest for variations of the ABSA task, including aspect term extraction, opinion term extraction, aspect-level sentiment classification, and aspect-oriented opinion extraction. Casting a problem into an index generation task is also proposed for semantic parsing (i.e., filling slots) [152]. The output sequence in this work starts with the intent, followed by slot names and the index sequences of the words in the input for the slots. At decoding time, each step produces a distribution over the word indices in the input sentence (as a pointer network) and the vocabulary for slots and intents in the datasets.

4.3 Generating Answers

This strategy is designed mainly for the QA task. The basic idea is to fine-tune PLMs to generate answers for the QA problems of interest. Wang et al. [187] use BART for closed-book QA that aims to directly provide answers for input questions. They show that BART struggles on a version of SQuAD for closed-book QA where the test and training data do not have much overlap. It also shows that BART cannot remember knowledge from the fine-tuning data if there are many training passages for fine-tuning. Suggestions to address these issues include decoupling the knowledge memorization and QA fine-tuning, and forcing the model to recall relevant knowledge in the answer generation step.

Hsu et al. [67] addresses the problem of answer selection, in which the system must choose the correct answer from a provided candidate set (it is also provided the question). Instead of training

an answer *selector* [58], Hsu et al. [67] uses answer *generation* through fine-tuning PLMs such as T5 and BART, which consume the input question and the top answer candidates, then generate an answer for the question. To prepare training data for fine-tuning, the output answers might come from human annotators or be directly inherited from the provided correct answer (i.e., the correct answer will be removed from the input for the generative models and may be replaced by another answer candidate).

4.4 Filling Templates

For many extraction tasks, the outputs are spans organized into one or several templates. For example, event extraction tasks require a system to extract templates in the form of *who did what to whom, where, and when.*

A template defines the appropriate relationship and order for the spans and labels for generation, forming the output sequence y. Du et al. [35] explores the template filling idea for an IE task: given a document, a model must identify event templates/types (via trigger words) and entity mention fillers for the argument roles. A sequence-to-sequence model for template filling takes the possible event types concatenated with words in the input document x as the input, and outputs a sequence of tuples. Each tuple corresponds to a detected event template, starting with an event type and followed by the text span fillers for the roles in the input document. Roles with no fillers are associated with *null*. [213] also examines a similar approach of tuple generation for ABSA.

The template filling method can also introduce additional information into the templates to aid the label generation process, such as natural descriptions or definitions of the labels. Li et al. [100] pursue a general template filling approach for document-level event argument extraction: given an event trigger in an input document, find entity mentions to fill in the roles for the event. A conditional generative model (e.g., BART) is employed for argument extraction where the input (the condition) to the model is created by combining an unfilled template and the document context. The template is essentially a sentence describing the event type augmented with placeholders for argument role fillers. The output sequence y is a filled template where placeholders are replaced by concrete arguments (entity mentions). To avoid entity-type mismatch for arguments, the templates in the inputs are also appended with sentences to indicate entity types for arguments (e.g., arg_1 is a person) that can be used to re-rank the output sequences to follow the type constraints. Below is an example input x in which a template over a list of event arguments $arg_1, \ldots, arg_6$ and the document text DOC_TEXT are concatenated, and output y, in which the underlined text spans are fillers from DOC_TEXT) [100]:

x = ⟨s⟩ ⟨arg_1⟩ bought, sold, or traded ⟨arg_3⟩ to ⟨arg_2⟩ in exchange for ⟨arg_4⟩ for the benefit of ⟨arg_5⟩ at ⟨arg_6⟩ place. ⟨s⟩ ⟨/s⟩ DOC_TEXT ⟨/s⟩
y = Elliott bought, sold or traded truck to McVeigh in exchange for 280.32 for the benefit of ⟨arg⟩ at body shop place.

4.5 Generating Structure-Linearized Texts

Structure prediction problems in NLP typically require multiple prediction outputs for an input text x that are interconnected to form a single structure that represents the input. To cast structure prediction tasks as text generation problems, one approach involves linearizing the output structure to serve as the output sequence y. For example, taking x as input, TEXT2EVENT [115] directly generates the event structures y:

Graph traversal algorithms are often used to accomplish the linearization in this approach. Ren et al. [149] study structure linearization for joint entity and relation extraction [99]. The main

x = The man returned to Los Angeles from Mexico following his capture Tuesday by bounty hunters.
y = ((Transport returned (Artifact The man) (Destination Los Angeles) (Origin Mexico))
(Arrest-Jail capture (Person The man) (Time Tuesday) (Agent bounty hunters))

idea is to construct an information graph for each input sentence to capture entity mentions, their entity types, and relations. Depth- or breadth-first traversal can be used for graph linearization for y. To solve the sequence-to-sequence problem for pairs of $\langle x, y \rangle$, Ren et al. [149] linearize the information graph to an alternating sequence of nodes and edge types, and directly generate such sequences via a hybrid span decoder that decodes both the spans and the types recurrently. For joint extraction of event triggers and arguments, a structure-linearization and text generation approach comes from Lu et al. [115]. The authors first build a labeled tree to capture the event types and argument roles in the sentence (i.e., event schema), with trigger and argument text spans as leaves. The labeled tree is transformed into the output sequence y by depth-first traversal where T5 is used to perform the conditional generation of y from x. To improve the model, a trie-based constrained decoding procedure [15, 30] is introduced to ensure the generation of valid event structures. A trie (prefix-tree) determines possible candidates for the next generation step given the previously generated tokens to guarantee valid output sequences. Lu et al. [115] also report the effectiveness of the generation-based model for extraction of new event types.

4.6 Ranking Input-Output Pairs

Some NLP tasks require choosing the best response from among many: answer selection in multiple choice-style QA, information retrieval, and certain kinds of entity retrieval all provide a set of candidate answers to a posed query from which the system selects the best one. Typically, a system will rank the candidates in relation to the input query, a task at which PLMs excel. The idea has its roots in the classical literature on probabilistic models for information retrieval that rank documents using language models [84, 139]. Given an input query, a candidate document is scored in two steps: (i) training a language model on the candidate document, and (ii) computing the likelihood of generating the input query from that language model, which serves as the candidate's ranking score.

We now see the use of PLMs to perform generation-based ranking for selection. Nogueira dos Santos et al. [128] apply the idea for answer selection by fine-tuning generative PLMs over ⟨answer, question⟩ pairs, thus learning to generate questions given correct answer passages. The simplest approach is to fine-tune the models over only the positive pairs. Nogueira dos Santos et al. [128] also explore fine-tuning with negative pairs using an unlikelihood objective or ranking-based objective (e.g., the hinge loss). At inference time, the ranking score for an input passage is obtained via the likelihood of the fine-tuned PLM over the input question conditioning on that passage.

Nogueira et al. [127] approach the document relevance ranking problem in a similar way. The paper concatenates the input query and each candidate document and feeds them as an input/condition for a fine-tuned T5 model. To fine-tune T5, the model is asked to generate "True" or "False" as the output sequence, indicating the document's relevance to the query. The probability of generating "True" is used as the ranking score for the candidate.

De Cao et al. [30] address the entity retrieval problem: given a set of Wikipedia articles representing entities, return the entity that is most relevant to a textual input source x. Each entity is represented by its textual representation (e.g., the title of its Wikipedia article), which will be used as the output sequence y for the generative models. BART is fine-tuned to rank the entities using the generation likelihood $P(y|x)$. Cui et al. [26] explore generation-based ranking for NER, especially in few-shot and cross-domain few-shot settings. Given an input sentence and a text span, a template is formed by concatenating the words in the span and an expression of type "is a *entity_type* entity." The original sentence and the template serve as an input-output pair in

sequence-to-sequence models. BART scores this pair using the probability of the template output produced by the BART decoder. For each span, the entity type corresponding to the template with highest score is selected. Original NER training data comprises the gold standard templates that fine-tune BART in this task.

In addition to QA, other generative tasks have been shown to benefit from PLMs. For instance, semantic parsing, generating a structure representing the semantics of the sentence, is explored by Shin et al. [169]. By reformulating the output of PLMs, the generated natural language can be used to recover the semantic structure of the input text. They use GPT-3 in the experiments.

5 DISCUSSION

Mix of paradigms or PLMs. The three paradigms presented here are by no means mutually exclusive. Instead, it is not rare to see approaches that use two or three paradigms together: fine-tuning techniques are often used as part of prompt-based methods; NLP-as-text-generation approaches often use carefully crafted templates (prompts); and prompt-based learning often leverages the text generation capabilities of PLMs to generate words, phrases, or sentences.

A representative example is Khashabi et al. [80], which combined three paradigms: appropriate prompts from the context and questions help to formulate several QA tasks into a unified text generation problem with seq2seq-based pre-trained models such as T5, with model fine-tuning to improve performance in several QA tasks.

As independently trained models, PLMs are also by no means mutually exclusive. For example, ACE [190] shows that combining multiple PLMs (e.g., ELMo, BERT, mBERT, XLM-R) yields further improvements over using a single PLM for a range of NLP tasks. Investigation of the complementarity of different PLMs is a future research direction.

From another perspective, the design of the training for MLMs has been driven by the results on the fine-tuning paradigm, but it is not clear whether an exploration of different training objectives could lead to PLMs that are more effective when used with prompting or generation to solve NLP tasks.

Environmental impact. The popularity of PLMs has dramatically increased the amount of computation used in NLP, leading to a large environmental impact.

Strubell et al. [173] is among the early attempts to quantify the financial and environmental costs of training large PLMs. They point out that PLMs often require massive computational resources that consume substantial energy and lead to a large carbon footprint. They recommend that published works begin to document the training process more explicitly and that the community prioritize the development of efficient models and hardware.

Schwartz et al. [168] argue for Green AI, suggesting that we should consider efficiency, measured by the number of floating-point operations used to generate a result, as a main evaluation criterion, together with accuracy. Green AI also aims to reduce the financial cost of the computation. In line with this approach, Izsak et al. [72] propose software optimization and design choices for pre-training BERT in 24 hours using a single low-end deep learning server.

In contrast, Patterson et al. [131] argues that the carbon footprint of ML training is a few orders of magnitude less than what was estimated in studies such as Strubell et al. [173] due to incomplete information. They recommend best practices to reduce the energy requirement for training: selecting a more efficient architecture, using processes optimized for ML training, computing in the cloud, and choosing the location with the cleanest energy source. Using their estimate and assuming the whole field follows the best practices, they predict that the carbon footprint of ML training will shrink over this decade.

The role of linguistic information. A frequent debate is whether a symbolic annotation covering syntax or semantics should be integrated to improve the performance of a PLM-based system, or whether this information is already present in the model. In terms of syntax, Xu et al. [200] utilize automatically produced syntax in both the pre-training and fine-tuning stages, and show improved performance on several benchmark datasets. Nguyen et al. [125] and Sachan et al. [154] inject syntax only in the fine-tuning stage. Regarding semantics, Zhang et al. [216] incorporate SRL predictions into the pre-training procedure of BERT, improving the performance on TE and QA tasks. Wu et al. [199] integrate semantic information into the task-specific fine-tuning stage, focusing on the DELPHIN dependencies formalism or "DM" [71]. Experimenting on RoBERTa, they obtained improvements on GLUE. Syntax and semantics can also be jointly integrated, as in Zhou et al. [222], where multi-task learning was used to combine BERT pre-training, semantic, and syntactic parsing tasks, improving the performance on the GLUE benchmark. While these studies show some success in leveraging syntax or semantics, there is no definite answer on whether the practice is necessary.

On the art and science of prompts. The success of prompts in zero- and few-shot learning has been attributed to the prompts serving as instructions that allow the PLM to learn with fewer examples, much the way humans would [39, 121, 164]. In fact, the excellent results may instead be attributable to the mere exploitation of patterns in the training data of PLMs, and not to PLMs' perceived ability to interpret and follow meaningful instructions. Webson and Pavlick [194] show, for instance, that irrelevant templates match the performance of meaningful ones in few-shot entailment experiments, adding that some of the templates discovered by automatic generation of discrete prompts are also unnatural [170]. In this sense, the results of continuous prompts also show that PLMs do not need meaningful instructions for improving few-shot performance.

Another question is on the amount of labeled data needed for prompt-based learning. While Le Scao and Rush [89] present experiments to quantify the impact of prompts, there have been few rigorous experiments studying how many labeled examples are required to achieve various levels of performance for a range of NLP tasks, and using each of the three paradigms outlined in this survey. Such studies will provide a better understanding of the pros and cons of each formulation, including cost-benefit analyses weighing the impact of more labeled data, helping developers design NLP systems that achieve the desired goal while minimizing human labeling effort.

Limitations and biases of PLMs. It is important to understand the limitation of PLMs and what potential societal impacts they may have. Bender and Koller [13] argues that PLMs, trained only on form, cannot in principle learn meaning. Brown et al. [39] probes GPT-3 for biases, and revealed that there are significant biases in different occupation and descriptive words strongly associated with each gender group, sentiment biases toward certain ethnic groups, as well as different words describing different religions.

Bender et al. [12] pointed out that PLMs trained on large, uncurated, and static datasets encode biased views that are harmful to marginalized populations. PLMs can also produce fluent but factually incorrect, misleading, or harmful content. Furthermore, large LMs may be prompted to reveal **personally identifiable information** (**PII**) from their training data. These outcomes can lead to great harm, intentionally or unintentionally.

Weidinger et al. [196] provides a more comprehensive and structural view of the risks associated with PLMs, that includes fairness, toxicity, private data leaks, false and misleading information, environmental and economic risks, as well as the risks raised in conversational agents and malicious applications.

To mitigate bias and other risks, Bender et al. [12] recommend careful planning, documentation of the training data, as well as value-sensitive design [47] as a mitigation strategy in the PLM

development process to identify the values to be expressed and supported (or a lack of support may result in harm).

More generally, Bender and Koller [13] argues for more research in understanding what PLMs learn that leads to success on NLP tasks that require knowledge of meaning.

Theoretical and empirical analysis. The theoretical understanding of the paradigms presented in this survey is preliminary. Apart from the issues mentioned above, there is a lack of understanding of what actually makes these paradigms so successful, and whether their success can be generalized across models and languages. For instance, prompts may be PLM dependent, or they may be transferable across models as indicated in [133]. There is very little work on studying the generalization of prompting and generation across languages, in the way that transfer learning has been applied to learning in one language and testing in another [22].

6 RELATIONS TO USER ALIGNMENT RESEARCH

Since the advent of ChatGPT,[6] the world has seen an explosion of interest in generative PLMs (e.g., LLaMA [178], Bard,[7] Jurrassic-2,[8] Claude[9]) that can follow instructions and provide detailed responses. We briefly discuss two main lines of enabling technologies for such user alignment below.

The first approach is instruction tuning, which fine-tunes PLMs on datasets involving natural language instructions. FLAN [**?**] shows that by training the model using instruction datasets, the model will learn to follow instructions to perform tasks, and can generalize to unseen tasks. Such instruction tuning improves zero-shot performance on unseen tasks significantly, with FLAN even outperforming few-shot GPT-3 on some tasks. A key part of this process is formulating existing datasets as instruction datasets, typically with similar templates to the prompting techniques described in Section 3. T0 [159] also fine-tunes on instruction datasets, with a particular focus on evaluating zero-shot performance on held-out tasks and robustness to prompt wording, and comes with a large collection of prompts for diverse datasets curated via PromptSource [6]. A similar collection of prompts is Super-NaturalInstructions [191], a benchmark of 1,616 diverse NLP tasks and their expert-written instructions. Datasets with such a large and diverse collection of tasks allow training instruction-following models and enable rigorous evaluation of cross-task generalization.

The second approach is **Reinforcement Learning from Human Feedback** (**RLHF**). InstructGPT [129], a sibling and a predecessor to ChatGPT, is trained to follow instructions via RLHF. The basic idea is to align PLMs with user intent by fine-tuning them with human feedback in two steps. The first step is supervised fine-tuning where the PLM is fine-tuned using prompts (instructions) and desired completions. This is similar to instruction tuning, but the prompts and completions are provided by human labelers instead of automatically generating such data from existing NLP datasets via prompt templates. In the second steps, the model is further fine-tuned via reinforcement learning using a reward model trained from rankings of model outputs. Such rankings are again provided by human labelers. These two steps train the PLM with human demonstrations of the desired model behavior and human preferences on model output, respectively, such that the model will generate outputs that are more aligned with the user's expectation. The InstructGPT paper [129] shows that its outputs are preferred to outputs from the 175B GPT-3 by humans (despite InstructGPT being much smaller). It also improves the truthfulness of the output and reduces toxicity.

[6] https://openai.com/blog/chatgpt.
[7] https://bard.google.com/.
[8] https://www.ai21.com/blog/introducing-j2.
[9] https://www.anthropic.com/index/introducing-claude.

User alignment research aims at aligning generative PLMs' output with human intents. Given the cross-task generalization observed in prior work, it has great potential to improve NLP task performances, especially in the zero-shot setting.

7 CONCLUSION

In this article, we present a survey of the three trending paradigms that use pre-trained language models for NLP. We describe each of them in depth, and summarize prior works whose applications have shown promise. We also discuss limitations and suggest directions for future research. We hope this survey has provided readers with key fundamental concepts and a comprehensive view of the training, adaptation, and use of pre-trained language models.

ACKNOWLEDGMENTS

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600006 under the IARPA BETTER program and by Contracts FA8750-19-2-0201 and FA8750-19-2-1004 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, the Department of Defense or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. We would like to thank Paul Cummer for his insightful comments on this work.

REFERENCES

[1] Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. 2021. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. https://arxiv.org/abs/2012.09816

[3] Asaf Amrami and Yoav Goldberg. 2019. Towards better substitution-based word sense induction. https://arxiv.org/abs/1905.12598

[4] Mikel Artetxe, Jingfei Du, Naman Goyal, Luke Zettlemoyer, and Ves Stoyanov. 2022. On the Role of Bidirectionality in Language Model Pre-Training. https://arxiv.org/abs/2205.11726

[5] Ben Athiwaratkun, Cicero Nogueira dos Santos, Jason Krone, and Bing Xiang. 2020. Augmented natural language for generative sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*.

[6] Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M. Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts. https://arxiv.org/abs/2202.01279

[7] Geoff Bacon and Terry Regier. 2019. Does BERT agree? Evaluating knowledge of structure dependence through agreement relations. https://arxiv.org/abs/1908.09892

[8] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

[9] Jack Bandy and Nicholas Vincent. 2021. Addressing "Documentation Debt" in Machine Learning Research: A Retrospective Datasheet for BookCorpus. https://arxiv.org/abs/2105.05241

[10] Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*.

[11] Eyal Ben-David, Nadav Oved, and Roi Reichart. 2021. PADA: A prompt-based autoregressive approach for adaptation to unseen domains. https://arxiv.org/abs/2102.12206
[12] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT'21)*. Association for Computing Machinery, New York.
[13] Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.
[14] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.
[15] Pinzhen Chen, Nikolay Bogoychev, Kenneth Heafield, and Faheem Kirefu. 2020. Parallel sentence mining by constrained decoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL'20)*.
[16] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained BERT networks. *Advances in Neural Information Processing Systems* 33 (2020), 15834–15846.
[17] Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021. KnowPrompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. https://arxiv.org/abs/2104.07650
[18] Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020. Reading the manual: Event extraction as definition comprehension. In *Proceedings of the 4th Workshop on Structured Prediction for NLP*. Association for Computational Linguistics, Online.
[19] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota.
[20] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.
[21] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. Association for Computing Machinery, New York, NY.
[22] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. https://arxiv.org/abs/1911.02116
[23] Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.
[24] Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2021. Recipes for adapting pre-trained monolingual and multilingual models to machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online.
[25] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297.
[26] Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Proceedings of the Findings of the Association for Computational Linguistics (ACL-IJCNLP'21)*.
[27] Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies* 6, 4 (2013), 1–220.
[28] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1285
[29] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 933–941.
[30] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*.
[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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: Human Language Technologies*.
[32] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Online, 1286–1305.

[33] Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. Association for Computational Linguistics, Online.

[34] Xinya Du, Luheng He, Qi Li, Dian Yu, Panupong Pasupat, and Yuan Zhang. 2021. QA-driven zero-shot slot filling with weak supervision pretraining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Online.

[35] Xinya Du, Alexander Rush, and Claire Cardie. 2021. Template filling with generative transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'21)*.

[36] Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? https://arxiv.org/abs/2010.11982

[37] 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, 19 (2010).

[38] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling Language Models: Methods, Analysis and Insights from Training Gopher. https://doi.org/10.48550/ARXIV.2112.11446

[39] Tom Brown et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901.

[40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

[41] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. arXiv:2110.08207.

[42] Yu Sun et al. 2021. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. https://arxiv.org/abs/2107.02137

[43] Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics* 8 (2020).

[44] Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics. 1173–1178. https://arxiv.org/abs/1909.00505

[45] Rui Feng, Jie Yuan, and Chao Zhang. 2020. Probing and fine-tuning reading comprehension models for few-shot event extraction. arXiv:2010.11325.

[46] Steven Fincke, Shantanu Agarwal, Scott Miller, and Elizabeth Boschee. 2021. Language model priming for cross-lingual event extraction. https://arxiv.org/abs/2109.12383

[47] Batya Friedman and David G. Hendry. 2019. *Value Sensitive Design: Shaping Technology With Moral Imagination*. MITt Press.

[48] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800GB dataset of diverse text for language modeling. arXiv:2101.00027.

[49] Shuyang Gao, Sanchit Agarwal, Di Jin, Tagyoung Chung, and Dilek Hakkani-Tur. 2020. From machine reading comprehension to dialogue state tracking: Bridging the gap. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*.

[50] Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*.

[51] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online.

[52] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online.

[53] Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. https://arxiv.org/abs/1901.05287

[54] Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

[55] Demi Guo, Alexander M. Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. arXiv: 2012.07463.

[56] Christian Hadiwinoto, Hwee Tou Ng, and Wee Chung Gan. 2019. Improved word sense disambiguation using pre-trained contextualized word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*. Association for Computational Linguistics.

[57] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

[58] Rujun Han, Luca Soldaini, and Alessandro Moschitti. 2021. Modeling context in answer sentence selection systems on a latency budget. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL'21)*.

[59] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. PTR: Prompt tuning with rules for text classification. https://arxiv.org/abs/2105.11259

[60] Adi Haviv, Jonathan Berant, and Amir Globerson. 2021. BERTese: Learning to speak to BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online, 3618–3623.

[61] Hangfeng He, Qiang Ning, and Dan Roth. 2020. QuASE: Question-answer driven sentence encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online.

[62] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the ACL*.

[63] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[64] Sara Hooker. 2021. The hardware lottery. *Communications of the ACM* 64, 12 (2021), 58–65.

[65] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. PMLR.

[66] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 328–339.

[67] Chao-Chun Hsu, Eric Lind, Luca Soldaini, and Alessandro Moschitti. 2021. Answer generation for retrieval-based question answering systems. In *Findings of the Association for Computational Linguistics (ACL-IJCNLP'21)*.

[68] Patrick Huber, Armen Aghajanyan, Barlas Oğuz, Dmytro Okhonko, Wen tau Yih, Sonal Gupta, and Xilun Chen. 2021. CCQA: A new web-scale question answering dataset for model pre-training. https://arxiv.org/abs/2110.07731

[69] Philip Huebner, Elior Sulem, Cynthia Fisher, and Dan Roth. 2021. BabyBERTa: Learning more grammar with small-scale child-directed language. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL'21)*.

[70] Minyoung Huh, Pulkit Agrawal, and Alexei A. Efros. 2016. What makes ImageNet good for transfer learning? arXiv: 1608.08614.

[71] Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the 6th Linguistic Annotation Workshop*. Association for Computational Linguistics, 2–11.

[72] Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train BERT with an academic budget. https://arxiv.org/abs/2104.07705

[73] Wei Jiang, Zhenghua Li, Yu Zhang, and Min Zhang. 2019. HLT@SUDA at SemEval-2019 Task 1: UCCA graph parsing as constituent tree parsing. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

[74] Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020. X-FACTR: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*.

[75] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? On the calibration of language models for question answering. *Transactions of the Association of Computational Linguistics* 8 (2021), 423–438.

[76] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics* 8 (Jan. 2020), 64–77.

[77] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics* 8 (2020), 64–77.

[78] Mandar Joshi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2019. BERT for coreference resolution: Baselines and analysis. arXiv:1908.09091.

[79] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. arXiv:2001.08361.

[80] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

[81] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*.

[82] Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, et al. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics* 10 (2022), 50–72.

[83] Sawan Kumar and Partha Talukdar. 2021. Reordering examples helps during priming-based few-shot learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online.

[84] John Lafferty, and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 111–119.

[85] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. 282–289.

[86] Huiyuan Lai, Antonio Toral, and Malvina Nissim. 2021. Thank you BART! Rewarding pre-trained models improves formality style transfer. https://arxiv.org/abs/2105.06947

[87] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based and neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

[88] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. arXiv:1909.11942.

[89] Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth?. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online.

[90] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[91] Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

[92] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of EMNLP*.

[93] Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd schema challenge. In *Proceedings of the 13th International Conference on the Principles of Knowledge Representation and Reasoning*.

[94] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. *Advances in Neural Information Processing Systems*, Vol. 27. Curran Associates, Inc.

[95] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL'17)*. Association for Computational Linguistics

[96] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online.

[97] Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online.

[98] Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Pretrained language models for text generation: A survey. https://arxiv.org/abs/2105.10311

[99] Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. Association for Computational Linguistics.

[100] Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online.

[101] Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. Exploiting BERT for end-to-end aspect-based sentiment analysis. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT'19)*. Association for Computational Linguistics, 34–41.

[102] Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online.

[103] Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

[104] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. (2021). https://arxiv.org/abs/2101.00190

[105] Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 6524–6538. https://doi.org/10.18653/v1/2021.acl-long.510

[106] Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization?. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online.

[107] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4 (2016), 521–535.

[108] Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. Association for Computational Linguistics, Online.

[109] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for GPT-3?. https://arxiv.org/abs/2101.06804

[110] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv: 2107.13586.

[111] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. https://arxiv.org/abs/2103.10385

[112] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692.

[113] Robert L. Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. https://arxiv.org/abs/2106.13353

[114] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. https://arxiv.org/abs/2104.08786

[115] Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL'21)*.

[116] Qing Lyu, Hongming Zhang, Elior Sulem, and Dan Roth. 2021. Zero-shot event extraction via transfer learning: Challenges and insights. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*.

[117] Brian MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk. Transcription Format and Programs*. Vol. 1. Psychology Press.

[118] R. Thomas McCoy, Robert Frank, and Tal Linzen. 2020. Does syntax need to grow on trees? Sources of hierarchical inductive bias in sequence-to-sequence networks. *Transactions of the Association for Computational Linguistics* 8 (2020).

[119] Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. 2018. Crowdsourcing question-answer meaning representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 560–568.

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

[121] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. https://arxiv.org/abs/2104.08773

[122] Mahdi Namazifar, Alexandros Papangelis, Gokhan Tur, and Dilek Hakkani-Tür. 2020. Language model is all you need: Natural language understanding as question answering. https://arxiv.org/abs/2011.03023

[123] Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

[124] Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. 2021. Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Online.

[125] Xuan-Phi Nguyen, Shafiq Joty, Steven C. H. Hoi, and Richard Socher. 2020. Tree-structured attention with hierarchical accumulation. https://arxiv.org/abs/2002.08046

[126] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[127] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics (EMNLP)*.

[128] Cicero Nogueira dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [CLS] through ranking by generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*.

[129] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. https://arxiv.org/abs/2203.02155

[130] Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero dos Santos Nogueira, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*.

[131] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. 2022. The carbon footprint of machine learning training will plateau, then shrink. *Computer* 55, 7 (2022).

[132] 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'14)*. Association for Computational Linguistics. 1532–1543.

[133] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. https://arxiv.org/abs/2105.11447

[134] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

[135] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Proceedings of AKBC*.

[136] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases?. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*.

[137] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

[138] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*.

[139] Jay M. Ponte and W. Bruce Croft. 2017. A language modeling approach to information retrieval. *ACM SIGIR Forum* 51, 2 (2017), 202–208.

[140] Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT plays the lottery, all tickets are winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*.

[141] Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. https://arxiv.org/abs/1912.10165

[142] Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online.

[143] XiPeng Qiu, TianXiang Sun, YiGe Xu, YunFan Shao, Ning Dai, and XuanJing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63, 10 (Oct. 2020), 1872–1897. https://doi.org/10.1007/s11431-020-1647-3

[144] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog* (2018), 12.

[145] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019), 9.

[146] Evani Radiya-Dixit and Xin Wang. 2020. How fine can fine-tuning be? Learning efficient language models. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 108)*, Silvia Chiappa and Roberto Calandra (Eds.). 2435–2443.

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

[148] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Association for Computational Linguistics. 784–789.

[149] Liliang Ren, Chenkai Sun, Heng Ji, and Julia Hockenmaier. 2021. HySPA: Hybrid span generation for scalable text-to-graph extraction. In *Findings of the Association for Computational Linguistics (ACL-IJCNLP'21)*.

[150] Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (CHI EA'21)*. Association for Computing Machinery, New York, NY.

[151] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics* 8 (Jan. 2021), 842–866.

[152] Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! A sequence to sequence architecture for task-oriented semantic parsing. In *Proceedings of the International World Wide Web Conference (WWW'20)*.

[153] Hayley Ross, Jonathon Cai, and Bonan Min. 2020. Exploring contextualized neural language models for temporal dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*.

[154] Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. 2021. Do syntax trees help pre-trained transformers extract information?. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.

[155] Oscar Sainz, Oier Lopez de Lacalle, Gorka Labaka, Ander Barrena, and Eneko Agirre. 2021. Label verbalization and entailment for effective zero and few-shot relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online.

[156] Oscar Sainz and German Rigau. 2021. Ask2Transformers: Zero-shot domain labelling with pretrained language models. In *Proceedings of the 11th Global Wordnet Conference*. Global Wordnet Association, University of South Africa (UNISA), 44–52.

[157] Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online.

[158] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. https://arxiv.org/abs/1910.01108

[159] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali

Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. https://arxiv.org/abs/2110.08207

[160] Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*.

[161] Timo Schick and Hinrich Schütze. 2019. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. https://arxiv.org/abs/1904.06707

[162] Timo Schick and Hinrich Schütze. 2020. BERTRAM: Improved word embeddings have big impact on contextualized model performance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[163] Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. (2020). https://arxiv.org/abs/2012.11926

[164] Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online.

[165] Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online.

[166] Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. https://arxiv.org/abs/2104.07540

[167] Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. *Transactions of the Asscociation for Computational Linguistics* 9 (2021).

[168] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Communications of the ACM* 63, 12 (Dec. 2020), 54–63.

[169] Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers.

[170] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics. 4222–4235.

[171] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2020. Portuguese named entity recognition using BERT-CRF. https://arxiv.org/abs/1909.10649

[172] Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 5986–5995.

[173] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

[174] Ehsan Taher, Seyed Abbas Hoseini, and Mehrnoush Shamsfard. 2019. Beheshti-NER: Persian named entity recognition using BERT. In *Proceedings of the 1st International Workshop on NLP Solutions for Under Resourced Languages (NSURL'19) co-located with ICNLSP 2019 - Short Papers*. Association for Computational Linguistics, Trento, Italy, 37–42.

[175] Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. oLMpics—On what language model pre-training captures. https://arxiv.org/abs/1912.13283

[176] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

[177] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

[178] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. https://arxiv.org/abs/2302.13971

[179] Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

[180] Trieu H. Trinh and Quoc V. Le. 2019. A simple method for commonsense reasoning. https://arxiv.org/abs/1806.02847

[181] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. https://arxiv.org/abs/2106.13884

[182] Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency Parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*.

[183] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.

[184] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in Neural Information Processing Systems*, Vol. 28.

[185] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*.

[186] Ben Wang. 2021. Mesh-transformer-JAX: Model-parallel implementation of transformer language model with JAX. Retrieved May 2021 from https://github.com/kingoflolz/mesh-transformer-jax.

[187] Cunxiang Wang, Pai Liu, and Yue Zhang. 2021. Can generative pre-trained language models serve as knowledge bases for closed-book QA?. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online.

[188] Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. https://arxiv.org/abs/2104.14690

[189] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pPretraining objective work bBest for zero-shot generalization? https://arxiv.org/abs/2204.05832

[190] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Automated concatenation of embeddings for structured prediction. https://arxiv.org/abs/2010.05006

[191] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 5085–5109. https://aclanthology.org/2022.emnlp-main.340.

[192] Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: A benchmark of linguistic minimal pairs for English. In *Proceedings of the Society for Computation in Linguistics 2020*.

[193] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. CoLA: The corpus of linguistic acceptability (with added annotations). http://nyu-mll.github.io/cola.

[194] Albert Webson and Ellie Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? https://arxiv.org/abs/2109.01247

[195] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. https://arxiv.org/abs/2109.01652

[196] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. arXiv:2112.04359).

[197] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

[198] Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online.

[199] Zhaofeng Wu, Hao Peng, and Noah A. Smith. 2021. Infusing finetuning with semantic dependencies. *Transactions of the Association of Computational Linguistics* 9 (2021), 226–242.

[200] Zenan Xu, Daya Guo, Duyu Tang, Qinliang Su, Linjun Shou, Ming Gong, Wanjun Zhong, Xiaojun Quan, Daxin Jiang, and Nan Duan. 2021. Syntax-enhanced pre-trained model. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online.

[201] Hang Yan, Junqi Dai, Tuo Ji, Xipeng Qiu, and Zheng Zhang. 2021. A unified generative framework for aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

[202] Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL'21)*.

[203] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[204] Jiarui Yao, Haoling Qiu, Jin Zhao, Bonan Min, and Nianwen Xue. 2021. Factuality assessment as modal dependency parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online.

[205] Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*.

[206] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. *Advances in Neural Information Processing Systems*, Vol. 27. Curran Associates, Inc.

[207] Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2021. A survey of knowledge-enhanced text generation. https://arxiv.org/abs/2010.04389

[208] Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. BARTScore: Evaluating generated text as text generation. https://arxiv.org/abs/2106.11520

[209] Karolina Zaczynska, Nils Feldhus, Robert Schwarzenberg, Aleksandra Gabryszak, and Sebastian Möller. 2020. Evaluating German transformer language models with syntactic agreement tests, In *Proceedings of the 5th Swiss Text Analytics Conference and the 16th Conference on Natural Language Processing (SwissText/KONVENS'20). CoRR* abs/2007.03765.

[210] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. https://arxiv.org/abs/2106.10199

[211] Jeffrey O. Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 2020. Side-Tuning: A baseline for network adaptation via additive side networks. https://arxiv.org/abs/1912.13503

[212] Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR Parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

[213] Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. 2021. Towards generative aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

[214] Yian Zhang, Alex Warstadt, Xiaocheng Li, and Samuel R. Bowman. 2021. When do you need billions of words of pretraining data?. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online.

[215] Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural CRF constituency parsing. In *Proceedings of the 29nth International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2020/560

[216] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-aware BERT for language understanding. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*.

[217] Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*.

[218] Tianyang Zhao, Zhao Yan, Yunbo Cao, and Zhoujun Li. 2020. Asking effective and diverse questions: A machine reading comprehension based framework for joint entity-relation extraction. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI'20)*, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization, 3948–3954. Main track.

[219] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. https://arxiv.org/abs/2102.09690

[220] Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics (EMNLP)*.

[221] Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online.

[222] Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020. LIMIT-BERT: Linguistics informed multi-task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online.

[223] Li Zhou and Kevin Small. 2020. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. https://arxiv.org/abs/1911.06192

[224] Qiji Zhou, Yue Zhang, Donghong Ji, and Hao Tang. 2020. AMR parsing with latent structural information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online.

[225] Yang Zhou, Yubo Chen, Jun Zhao, Yin Wu, Jiexin Xu, and Jinlong Li. 2021. What the role is vs. What plays the role: Semi-supervised event argument extraction via dual question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 14638–14646.

[226] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into neural machine translation. https://arxiv.org/abs/2002.06823

[227] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*. 19–27.

Received 9 January 2022; revised 7 October 2022; accepted 8 June 2023