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

Bonan Min*[1], Hayley Ross*[2], Elior Sulem*[3], Amir Pouran Ben Veyseh*[4],
Thien Huu Nguyen[4], Oscar Sainz[5], Eneko Agirre[5], Ilana Heinz[1], and Dan Roth[3]

[1]Raytheon BBN Technologies
{bonan.min, ilana.Heintz}@raytheon.com
[2]Harvard University
hayleyross@g.harvard.edu
[3]University of Pennsylvania
{eliors, danroth}@seas.upenn.edu
[4]University of Oregon
{apouran, thien}@cs.uoregon.edu
[5]University of the Basque Country (UPV/EHU)
{oscar.sainz, e.agirre}@ehu.eus

* indicate equal contribution

Abstract

Large, pre-trained transformer-based language models such as BERT have drastically changed the Natural Language Processing (NLP) field. We present a survey of recent work that uses these large language models to solve NLP tasks via pre-training then fine-tuning, prompting, or text generation approaches. We also present approaches that use pre-trained language models to generate data for training augmentation or other purposes. We conclude with discussions on limitations and suggested directions for future research.

1 Introduction

In recent years, large pre-trained transformer-based language models (PLMs), such as the BERT (Devlin et al., 2019) and GPT (Radford et al., 2018) 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 hand-crafted features to represent x, and then apply a machine learning model (e.g. SVM (Cortes and Vapnik, 1995), logistic regression) to learn the classification function. Deep learning models learn the latent feature representation via a deep neural network (LeCun et al., 2015) 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 the nuances of language are common to all NLP tasks, one could posit that we could learn a generic latent feature representations from some generic task once, and then share it across all NLP tasks. Language modeling, where the model needs to learn how to predict the next word given previous words, is such a generic task with abundant naturally occurring text to pre-train such a model (hence the name pre-trained language models). In fact, the latest, ongoing paradigm shift begins when PLMs are 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 paper surveys recent works that leverage PLMs for NLP. We organize these works into the following three paradigms:

- Pre-train then fine-tune (§ 2): perform general-purpose pre-training with a large unlabeled

corpus, and then perform a small amount of task-specific fine-tuning for the task of interest.

- Prompt-based learning (§ 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 usually more effectively leverage the knowledge encoded in the PLMs, leading to few-shot approaches.

- NLP as text generation (§ 4): Reformulate NLP tasks as text generation, to fully leverage knowledge encoded in a generative language model such as GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2020).

Generative PLMs can be also used for text generation tasks. We refer readers to the excellent surveys on text generation such as Li et al. (2021b) and Yu et al. (2021b). This paper, unless otherwise specified, focuses on tasks that are not generative in nature (e.g. classification, sequence labeling and structure prediction) that still cover a broad range of NLP tasks including syntactic or semantic parsing of text, Information Extraction (IE), Question Answering (QA), Textual Entailment (TE), sentiment analysis, and so on.

In addition to the three paradigms, there is another, complementary method: to indirectly use any of the PLM paradigms above to improve results of target NLP tasks:

- Data generation (§ 5): run PLMs to automatically generate data for NLP tasks. The generated data can be silver labeled data, where typically the generative PLM is fine-tuned for the task, or some auxiliary data, such as counterexamples, clarifications, contexts, or other. In the first case, the silver labeled data can be added to existing labeled data. In the second case, the auxiliary data supports the target task in some way.

The paper is organized as follows: Section 2 provides background on the PLMs and describes the first paradigm, *pre-train then fine-tune*. Section 3 discusses the second paradigm, *prompt-based learning*. Section 4 summarizes works in the third paradigm, *NLP as text generation*. In Section 5, we describe approaches that generate data

via PLMs for a broad range of NLP tasks. We discuss limitations and provide directions for future research in Section 6 and conclude in Section 7.

2 Paradigm 1: Pre-Train then Fine-Tune

While work in traditional statistical NLP focused on training task-specific models on labeled datasets, this paradigm shifts to training one large model on a shared, "fundamental" pre-training task and then adapting ("fine-tuning") it to a variety of tasks in a second step. The pre-training task is almost invariably a type of language modeling task[1] that can leverage a massive quantity of unlabelled data to learn representations that benefit a range of NLP tasks (Rogers et al., 2020).

In this section, we first provide a primer on pre-trained large language models (PLMs), then describe approaches that use frozen or fine-tuned PLMs for NLP tasks.

2.1 The Beginnings of the Paradigm Shift

While pre-training in machine learning and, in particular, computer vision has been studied since at least 2010 (Erhan et al., 2010; Yosinski et al., 2014; Huh et al., 2016), the technique did not gain traction in NLP until later in the decade, with the publication of Vaswani et al. (2017). The delay in uptake is partly due to the later arrival of deep neural models to NLP compared to computer vision, partly due to the difficulty of choosing a self-supervised task[2] suitable for pre-training, and above all, due to the need for drastically larger model sizes and corpora in order to be effective for NLP tasks. We explore these aspects further in the discussion below.

The idea of pre-training on a language modeling task is quite old. Collobert and Weston (2008) first suggested pre-training a model on a number of tasks to *learn* features instead of hand-crafting them (the predominant approach at the time). Their version of language model 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)

---

[1] The exact formulation varies from the classic unidirectional language modeling (next word prediction) to cloze-style fill-in-the-blank, uncorrupting spans, and other variants (see Section 2.3).

[2] 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.
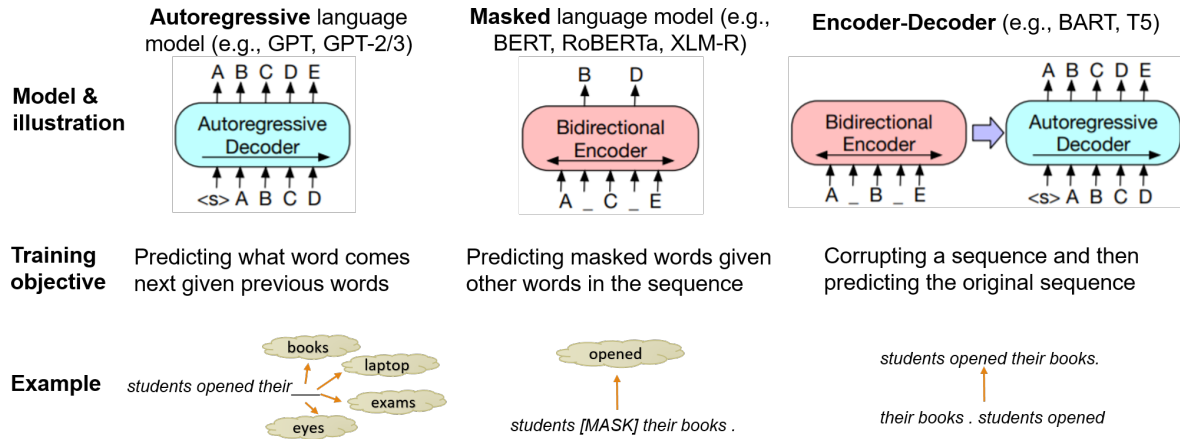
Figure 1: Three types of pre-trained language models. Model architecture illustrations are from Lewis et al. (2020). For the encoder-decoder model, the corruption strategy of document rotation is shown. Alternatives include sentence permutation, text infilling, token deletion/masking, etc.

and semantic role labeling (SRL). Collobert and Weston 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 two remaining two feed-forward layers for each individual task.

Pre-training and fine-tuning did not gain popularity in NLP until the advent of ELMo (Peters et al., 2018) and ULMFiT (Howard and Ruder, 2018). Both models are based on Long Short-Term Memory architecture (LSTMs) (Hochreiter and Schmidhuber, 1997), but differ in significant ways. ULMFiT pre-trains a three-layer LSTM on a standard language modeling objective, predicting the next token in a sequence. ELMo uses layers of bidirectional LSTMs that combine two language model tasks in forward and backward directions to capture context from both sides. Both proposed fine-tuning the language model layer by layer for downstream application. Both studies also suggested adding additional classifier layers on top of the language model, which were fine-tuned alongside the language model layers. These changes, combined with the substantially larger model size and pre-training corpus size compared to previous models, allowed the pre-training then fine-tuning paradigm to succeed. Both ELMo and ULMFiT showed competitive or improved performance compared to the then-state-of-the-art for a number of tasks, demonstrating the value of language model pre-training on a large scale.

The pace of this paradigm shift picked up dramatically in late 2018 when Vaswani et al. (2017) introduced the Transformer architecture that can be used for language model pre-training. The Transformer's multi-head self-attention mechanism allows every word to attend to 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 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 language models, including GPT, BERT, BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), 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[3,4].

2.2 Modern Pre-Trained Language Models

There are three classes of pre-trained language models: autoregressive language models (e.g. GPT), masked language models (e.g. BERT), and encoder-decoder models (e.g. BART, T5). Figure 1 shows the difference in model architecture and training objectives with an example training input for each.

2.2.1 Autoregressive Language Models

An autoregressive language model is trained to predict the next word x_i given all previ-

Model	Pre-Training Sources	Size of Pre-Training Corpus	# Model parameters
(1) **English Monolingual Models**			
BERT(BASE)(Devlin et al., 2019)	Wiki, books	3.3B tokens (13GB data)	110M
BERT(LARGE)(DEVLIN ET AL., 2019)	Wiki, books	3.3B tokens (13GB data)	340M
ROBERTA(Liu et al., 2019)	Wiki, books, web crawl	161GB data	340M
XLNET (Yang et al., 2019)	Wiki, books, web crawl	142GB data	340M
GPT(Radford et al., 2018)	Web crawl	800M tokens	117M
GPT-2(Radford et al., 2019)	Web crawl	8M documents (40GB data)	1.5B
GPT-3(Brown et al., 2020)	Wiki, books, web crawl	500B tokens	175B
BART (Lewis et al., 2020)	Wiki, books	3.3B tokens	∼370M
T5 (Raffel et al., 2020)	Web crawl	200B tokens (750GB data)	11B
(2) **Multilingual Models**			
MBERT(Devlin et al., 2019)	Wiki	21.9B tokens	172M
XLM-R(BASE) (Conneau et al., 2020)	Web crawl	295B tokens	270M
XLM-R(LARGE) (Conneau et al., 2020)	Web crawl	295B tokens	550M
MT5 (LARGE) (Raffel et al., 2020)	Web crawl	6.3T tokens	1.2B
MT5 (XXL) (Raffel et al., 2020)	Web crawl	6.3T tokens	13B

Table 1: Training sources, dataset size, and model parameters for popular PLMs. Data sources differ, and are described in the citations listed in each row.

ous words x_1, x_2, ..., and x_{i-1}. The training objective is to maximize the log-likelihood $\sum_i log(P(x_i|x_1, x_2, ..., x_{i-1}); \theta_T)$, 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 GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020)[5].

GPT only utilizes 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 is trained with increasingly large amounts of text (Table 1).

The GPT paper (Radford et al., 2018) proposed fine-tuning GPT for specific tasks, providing examples for natural language inference, QA (including commonsense reasoning), semantic similarity and paraphrase detection, sentiment analysis, and linguistic acceptability (CoLA, Warstadt et al., 2019), 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, Radford et al., 2019; Brown

et al., 2020), however, do not opt for the fine-tuning approach and instead leverage GPT's generative design to tackle tasks in a prompt-based manner or via outright language generation, as described in Sections 3 and 4.

2.2.2 Masked Language Models

Whereas autoregressive models are unidirectional, masked language models (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: $\sum_i m_i log(P(x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_n); \theta_T)$, 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. Note that in BERT and similar models, it is a common practice to mask multiple words from a sequence to allow parallel training. Popular examples of MLMs include BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and XLM-R (Conneau et al., 2020).

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 it uses masked self-attention to attend to all other tokens in the sequence in both directions when learning a representation for a particular token. The non-

---

[5]Open-source re-implementations of GPT are also available, such as GPT-Neo (Black et al., 2021) and GPT-J (Wang, 2021), trained on an 800GB open-source dataset (Gao et al., 2020a), with model sizes similar to GPT-2 (2.7B and 6B parameters respectively).

autoregressive nature allows the computation to be parallelized, so it is often more efficient at inference time. Dynamic unfolding of all positions in relation to the masked word provides efficiency at training time.

There is a large family of models derived from BERT, including RoBERTa (Liu et al., 2019), which improves BERT's pre-training, ALBERT (Lan et al., 2020), which is smaller and faster to train, and XLNet (Yang et al., 2019) and Transformer-XL (Dai et al., 2019), which incorporate an autoregressive pre-training approach to better handle long-distance dependencies. There are also a range of derived models trained on specific domains (Table 6 in Appendix A). See Qiu et al. (2020) for a full taxonomy of BERT-derived models.

2.2.3 Encoder-Decoder Language Models

The encoder-decoder model is a more flexible "text in, text out" model that learns to generate a sequence of token $y_1, ..., y_n$ given an input sequence $x_1, ..., x_m$. Given a pair of sequences, the training objective is to maximize the log-likelihood of $log(P(y_1, ..., y_n|x_1, ..., x_m); \theta_T)$, in which θ_T are the parameters in a full encoder-decoder Transformer model (Vaswani et al., 2017).

To generate adequate data for self-supervised pre-training, researchers experiment with different forms of sequence corruption. The input is a token sequence modified in some particular way, and the output is the reconstructed, original sequence. Forms of sequence corruption include document rotation, shown in Figure 1, sentence permutation, text infilling, token deletion/masking, and others. Representative models include BART (Lewis et al., 2020) and T5 (Raffel et al., 2020).

Given the sequence-to-sequence (seq2seq) nature, it is straightforward to fine-tune the encoder-decoder language model to perform seq2seq tasks such as Machine Translation, style transfer, and text summarization. The seq2seq formulation is also versatile: many tasks can be reformulated as "text in, text out". We describe those approaches in details in Section 4.

2.3 Pre-Training Corpora

The pre-training corpus is a primary distinguishing factor between language models. Both the size and the quality (source data characteristics) are important considerations. Table 1 presents the sources and the corpus size used for several popular language models. 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 (Howard and Ruder, 2018) 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 (Conneau et al., 2020) and GPT-3 (Brown et al., 2020) leveraged billions of words of crawled web data (diverse in nature). Raffel et al. (2020) 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. They find that quality can play a larger role if there is a genre match to the task, but a larger dataset provides more advantages, eventually overcoming any gain from quality. For a detailed discussion of model performance scaling by model size, dataset size, and other factors, see Kaplan et al. (2020). Despite the advantages of the larger dataset, Raffel et al. (2020) also demonstrate the importance of cleaning large crawled datasets. They show that a model trained on such an unfiltered dataset performs substantially worse than if filtering heuristics are applied. Similarly, GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020) use heuristics to improve the quality of the training data. However, Hendrycks et al. (2020) noted that larger models do not necessarily perform better out of domain. Lin et al. (2021) also observe that larger language models (trained on these very diverse sources) are more likely to incorrectly answer questions that some humans would answer incorrectly due to false beliefs or misconceptions, thus mimicking the inaccuracies in their training data.

The domain of intended downstream applications is an important consideration for pre-training source data selection. Table 6 (Appendix A) provides a list of domain-specific pre-trained language models that achieved significantly better performance in the intended domain than general-purpose language models. These models are either trained from scratch or trained with domain-specific text using a general-purpose model as the initialization.

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
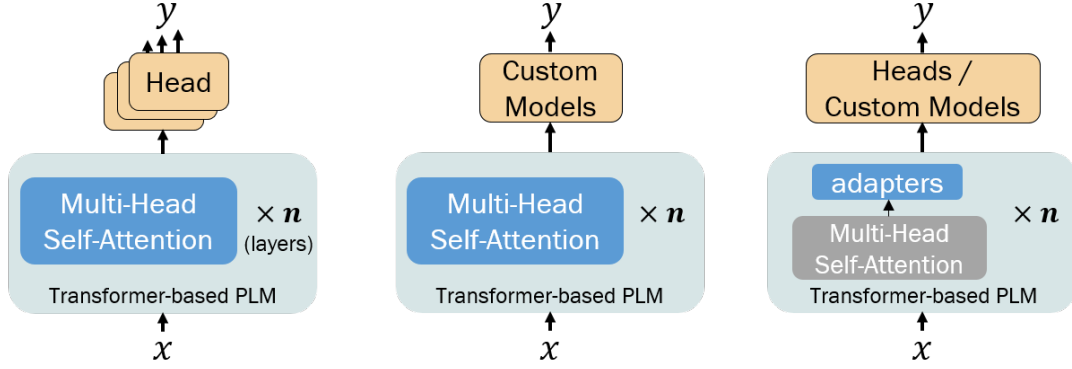
Figure 2: Typical "pre-train then fine-tune" strategies. We illustrate strategies that fine-tune the full PLM (left), fine-tune the full PLM in a custom model (center), and fine-tune just a small adapter sub-layer per each Transformer layer (right). We show the Transformer blocks that will be fine-tuned for the specific tasks in blue, and the frozen blocks (keep the pre-trained weights unchanged) in grey. 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 interested readers to Vaswani et al., 2017 and Pfeiffer et al., 2020a for more architecture details. "Heads" refers to task-specific prediction functions (Wolf et al., 2020).

allows PLMs to perform accurately on disparate NLP tasks. Figure 2 illustrates typical pre-training then fine-tuning strategies. We describe each of them below. A more comprehensive list of prior work using different pre-training then fine-tuning strategies are in Table 8 (Appendix B).

2.4.1 Contextual Embeddings

The simplest approach to using large pre-trained language models 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. In other words, while this still involves a forward pass through the pre-trained language model over the input text, the language model's weights are *not* fine-tuned, rendering this approach closer to a feature extraction family of approaches in classic statistical NLP. There are three types of scenarios for using frozen PLMs.

In contexts with insufficient labeled data or compute power, "frozen" contextual embeddings are employed. For non-benchmark tasks, the only labeled training datasets are 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.

Highly complex or difficult NLP tasks often

make use of the frozen PLM technique to help reduce training complexity. Examples are constituency parsing (Zhang et al., 2020c), semantic graph parsing using UCCA[6] (Jiang et al., 2019) and AMR[7] (Zhang et al., 2019b; Naseem et al., 2019; Zhou et al., 2020b), Aspect-Based Sentiment Analysis (Li et al., 2019b) and Machine Translation (Zhu et al., 2020). For instance, Zhang et al. (2020c) uses frozen BERT embeddings to seed an innovative approach to Conditional Random Field (CRF) modeling (Lafferty et al., 2001) that replaces the inside-outside algorithm with backpropagation, using a two-step process to first bracket 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. (2019b) 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.

Unsupervised tasks such as word sense disambiguation (Hadiwinoto et al., 2019) and word sense induction (Amrami and Goldberg, 2019) are not associated with a supervised dataset for fine-tuning. Instead, frozen BERT embeddings are fed through a variety of strategies such as nearest-neighbour

---

[6]Universal Conceptual Cognitive Annotation (Abend and Rappoport, 2013)

[7]Abstract Meaning Representation (Banarescu et al., 2013)

matching, affine transformations, gated linear units (GLU, Dauphin et al., 2017) or clustering algorithms to perform these tasks.

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, Wolf et al., 2020). Typically, these are feed-forward layers for classification. The output layers and the PLM are trained together in an end-to-end setup, but the bulk of the computation is applied to fine-tuning the language model 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 may come from the top layer, or from a concatenation or a weighted average of the top n (often $n = 4$) layers (Peters et al., 2018). Figure 2 (left) shows an illustration of this approach.

This approach 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. (2019) 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 Devlin et al., 2019), may be passed to the classifier. This fine-tuning approach is use to apply BERT to all 11 tasks in GLUE, as well as QA (SQuAD), NER (CoNLL 2003), and common-sense inference (SWAG). For many additional examples of this highly popular approach, see Table 8 (Appendix B).

In this setting, care is needed to choose an appropriate 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 should be used (between 1e-3 (Raffel et al., 2020) and 1e-5 (Liu et al., 2019)), with a lower learning rate for smaller datasets. However, the randomly initialized feed-forward layer weights still require significant training. As such, it is a common practice to freeze the language model lay-

ers temporarily while initially training the feed-forward layers, then unfreeze the language model gradually for additional fine-tuning (Howard and Ruder, 2018; Yang et al., 2019). The degree to which this should be done depends on the size of feed-forward layers, and whether a token such as BERT's [CLS] is being used. If the majority of the labour is being done by [CLS], as in all the examples in Devlin et al. (2019), there are fewer benefits to training the feed-forward layer alone. Again, this is a function of the availability of supervised training data.

The next choice is how many layers of the PLM to fine-tune. While the examples in the BERT paper 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 language model is sufficient; for example, Ross et al. (2020) only fine-tune the top layer of BERT on their small supervised dataset of 2000 sentences. A range of papers in the growing field of "BERTology" (Tenney et al., 2019, Clark et al., 2019b, Rogers et al., 2020) show that the lower layers of BERT contain word-specific and syntactic information such as part of speech, while the upper layers contain more semantic and increasingly complex information such as semantic roles and coreference information.

2.4.3 Fine-tuning the PLM in Customized Models

Some tasks require significant additional architecture on top of a language model, 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 language model. 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 (Souza et al., 2020b; Taher et al., 2019), though notably Devlin et al. (2019) show that the Conditional Random Field (CRF) layer is not necessarily needed for NER with BERT. Examples of parsing models using this approach include UDapter for dependency parsing (Üstün et al., 2020).

Any sequence-to-sequence task that uses a pre-trained language model as its encoder may employ this approach. An interesting example is Zhu et al. (2020)'s formulation of machine translation.

However, Zhu et al. did not find any significant improvement over using BERT-based frozen word embeddings.

A related and highly successful approach is to fine-tune the entire language model 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, it might transform the task from a classification problem (as understood by the language model) into the desired target formulation, often a structured form such as a tree or a set of clusters. For coreference resolution, Joshi et al. (2019, 2020) adds a substantial algorithm, in their case e2e-coref (Lee et al., 2018) which transforms ratings of pairs of spans into valid mention clusters. Specifically, for each candidate mention span, the algorithm computes a distribution over possible antecedent spans from the mention score (whether it is likely to be a mention) and the compatibility score of the two spans, which itself involves a feed-forward network to compute. Two more structural parsing examples in this vein are temporal dependency parsing (Ross et al., 2020) and modal dependency parsing (Yao et al., 2021). 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 two types: (a) fine-tuning a separate, small network that is tightly coupled with the PLM (but does not change it), and (b) selecting only a small number of the PLM's weights to fine-tune or keep.

The most prominent approach of the first type are adapter modules (Houlsby et al., 2019; Bapna and Firat, 2019; Pfeiffer et al., 2020b,a), as illustrated in Figure 2 (right). Adapters add a small set of newly initialized weights at every layer of the transformer. Houlsby et al. (2019) 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 (Houlsby et al., 2019; Bapna and Firat, 2019; Stickland and Murray, 2019; Pfeiffer et al., 2020b). 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, both in terms of using the same frozen PLM for each task, and in allowing the weights of adapter modules to be distributed and also re-used. Notably, the weights of adapters independently trained for different tasks can be successfully combined to solve a new task (Pfeiffer et al., 2020b). Finally, catastrophic forgetting of old capabilities when fine-tuning on a new task or language is prevented. AdapterHub (Pfeiffer et al., 2020a) and Trankit (Nguyen et al., 2021) are examples of frameworks promoting an adapter ecosystem; an example of using adapters for Universal Dependency Parsing is Üstün et al. (2020).

A similar method is side-tuning (Zhang et al., 2020b), which adapts a pre-trained network by training a lightweight "side" network that is fused with the (unchanged) pre-trained network using a simple additive process. Also closely related is diff-pruning (Guo et al., 2021), which adds a sparse, task-specific difference vector to the original (frozen) parameters. These difference vectors are regularized to be sparse, which further decreases the number of weights that need to be stored (around 0.5% of the original model's parameters).

Moving to the second type of approach, Bit-Fit (Zaken et al., 2021) proposes to limit fine-tuning to the bias terms (or a subset of the bias terms, around 0.1% of the total parameters) of pre-trained BERT models, plus a task-specific classification layer. This is shown to be competitive with, and for some tasks better than, fine-tuning all of BERT. BitFit builds on the intuition that fine-tuning exposes existing capabilities, rather than teaching the model new ones.

Similarly, Radiya-Dixit and Wang (2020) show that it suffices to fine-tune only the "most sensitive" layers, i.e. those which are 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 (Sanh et al., 2020) and other pruning studies (Prasanna et al., 2020 inter alia) which show that many parameters in a large PLM are redundant.

In fact, Zhao et al. (2020a) propose masking, i.e. setting weights to zero, as a sole 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. The initial sparsity of the mask is an important hyperparameter in this approach, as is deciding which layers to mask, since the different layers encode various degrees of syntactic and semantic knowledge (Tenney et al., 2019). Zhao et al. show that masking "top-down" (mostly the top layers, which are more task-specific and encode more semantic and long-distance information) is more effective than masking "bottom-up" (which would mask mostly the layers dealing with elementary word meaning and syntax). In particular, performance on CoLA increases as more layers are masked top-down. The authors further show that masking yields entirely comparable performance to fine-tuning on a range of tasks from POS tagging to reading comprehension.

3 Paradigm 2: Prompt-based Learning

We use prompting to refer to the practice of adding natural language text, often short phrases, to the input or output to encourage pre-trained models to perform specific tasks (Yuan et al., 2021). There are several advantages to using prompts. Prompting, especially in-context learning (e.g. Brown et al., 2020), may not require updates to the PLM's parameters, reducing computational requirements as compared to fine-tuning approaches, or in addition to those described in 2.4.4. Prompts also encourage a better alignment of the new task formulation with the pre-training objective, leading to better use of knowledge captured in pre-training. The closer match also enables a few-shot approach (Liu et al., 2021b), especially for tasks with small training datasets; a good prompt can be worth hundreds of labeled data points (Le Scao and Rush, 2021). Finally, prompts allow probing of the PLMs, often in an unsupervised way, in order to assess the knowledge acquired by the PLM for specific tasks of interest (e.g. Petroni et al., 2019).

We discuss 3 types of prompt-based learning approaches below: Learning from instructions and demonstrations, template-based learning, and learning from proxy tasks. Figure 3 shows illustrations for each of the three approaches.

Input	Output
(1) Text pair generation (Schick and Schütze, 2021)	
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 (Reynolds and McDonell, 2021)	
$f(x) = x * x$. 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 $f(f(3)) = 27$.

Table 2: Example prompt designs for learning from instructions.

3.1 Learning from Instructions and Demonstrations

First attempts made use of instructions such as "translate X to Y:" (Raffel et al., 2020) 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 (Radford et al., 2019), the first signs that language models 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. More recently, and with even larger generative PLMs (GPT-3), Brown et al. (2020) showed that those models are indeed very good at few-shot learning. Brown et al. showed that GPT-3 can perform few-shot tasks via priming (in-context learning): given instructions and a few input/output pairs, GPT-3 is able to produce the desired outputs for new inputs. No gradient updates are performed (see Figure 3, left box). Caveats include the requirement of a very large LM to work well, and an inability to scale to more than a few examples, because the context window of most LMs is limited to a few hundred tokens. We refer readers to the GPT-3 paper (Brown et al., 2020) for many additional examples of learning from instructions and/or demonstrations.

Schick and Schütze (2021) and Reynolds and McDonell (2021) introduce new tasks based on descriptions. For example, the text pair generation task Schick and Schütze (2021) consists in generating a continuation sentence (Sentence 2) given an input sentence (Sentence 1) and a description of the relations between the sentences (Table 2(1)).[8]

---

[8] A variation of this task consists in generating both Sentence 1 and Sentence 2 given the description (Schick and Schütze, 2021).

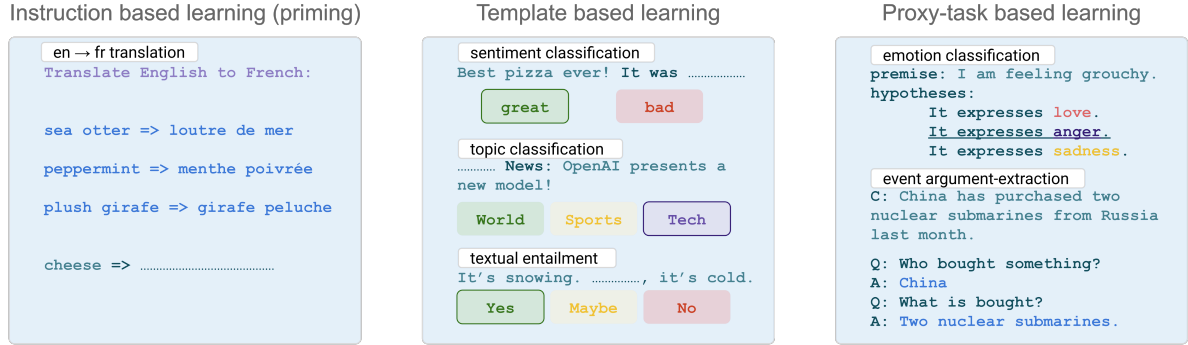| Instruction based learning (priming) | Template based learning | Proxy-task based learning |

Figure 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 blue and the prompt in cyan. On the prompt based learning (middle box), the text to classify is marked on light cyan and the prompt on dark cyan; the label verbalizations are shown in small boxes. On the proxy task based learning (right box), prompts are marked with dark cyan, the context is on light cyan and the answers generated by the model are in blue.

To address this task, Schick and Schütze (2021) use a generative PLM (GPT2-XL) that generates Sentence 2, replacing the _ token. Impressively, even mathematical reasoning can be handled (Table 2(2)): Reynolds and McDonell (2021) show that by inserting a natural language prompt ("Let's solve . . . steps.") after the math problem statement, GPT-3 can generate a procedure that solves the math problem.

Recently, Wei et al. (2021) showed that teaching a very large PLM to follow instructions with supervised data improves the zero and few-shot abilities of these PLMs. They carried out a large scale multi-task experiment over more than 60 datasets grouped into 12 task different tasks, and showed that a PLM trained via natural language instructions on other tasks outperforms a standard language model on the test task. Mishra et al. (2021) fine-tuned BART (Lewis et al., 2020) to perform a similar task using instructions and few-shot examples for a variety of crowd-sourced NLP tasks. The crowdsourcing process of each task consists of several steps that are natural and intuitive for human annotators. The instructions to the PLM match the step-by-step crowdsourcing instructions, decomposed into self-contained, separate tasks, leading to improved performance on unseen tasks, in contrast to an earlier work (Efrat and Levy, 2020) that reported negative performance when using the crowdsourcing instructions as-is.

Scaling limitations may affect the broad applicability of this approach: Wei et al. (2021) 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, Sanh et al. (2021) 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. (2021). (1) They use a 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. (2021) 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(Le Scao and Rush, 2021), or even eliminating the need for training data. To achieve this goal, template-based learning reformulates various NLP tasks into language modeling tasks via carefully designed templates with open slots. In this

Input	Output
(1) Topic/sentiment classification (Schick and Schütze, 2021a)	
Best pizza ever!. It was __.	*great → Positive*
(2) Textual entailment (Schick and Schütze, 2021a)	
Mia likes pie? __, Mia hates pie.	*No → Contradiction*
(3) Event argument extraction (Chen et al., 2020c)	
Americans sought to bring calm to Mosul, where U.S. troops killed 17 people in clashes earlier in the week. <u>someone</u> killed <u>someone</u> with something in some place at <u>some time</u>.	U.S. troops killed 17 people with something in Mosul at earlier in the week.
(4) Probing for relations/facts (Petroni et al., 2019)	
Dante was born in __.	*Florence*
(5) Probing for commonsense (Trinh and Le, 2019)	
The trophy doesn't fit in the suitcase because <u>it</u> is too big	*it → trophy*:0.9 *it → suitcase*:0.2
(6) Probing for reasoning (Talmor et al., 2020)	
The size of an airplane is __ than the size of a house . A. larger B. smaller	*larger*

Table 3: Example prompt designs for template-based methods. *great → Positive* means that the answer *great* will be converted to label *Positive*. For Chen et al. (2020c), each of the underlined words (e.g. <u>someone</u>) will be replaced with the underlined phrase on the output side by a PLM. "*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.

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.

Template-based learning differs from instruction learning (Section 3.1) in that templates are less detailed and do not explicitly describe the task.

3.2.1 Template Design

Using a *cloze-style prompt* design, inputs to a PLM are converted into a format such that solving the NLP task only requires the PLM to predict missing word(s). Table 3 (1) shows the most 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 textual entailment task in which the goal is to predict the *entail/contradict* relation between a pair of input sentences $\langle X_1, X_2 \rangle$. Pattern-Exploiting Training (PET) (Schick and Schütze, 2021a) (Table 3 (2)) converts a pair of inputs $\langle X_1, X_2 \rangle$ into "$X_1?\_\_, X_2$" and asks a masked language model to predict the missing word. The prediction (here *yes*

or *no*) is directly mapped to one of the textual entailment class labels. This template design allows PET to reformulate the text entailment problem into the same masked language modeling problem that was 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, textual entailment, and knowledge probing). Chen et al. (2020c) reformulates the event argument extraction challenge as a cloze-style problem (Table 3 (3)), predicting fillers for the underlined positions, and then apply greedy decoding to fill in the __ position incrementally. Petroni et al. (2019) similarly use the cloze-style task for relation/fact probing (Table 3 (4))

A *multiple-choice style prompt* proves useful for 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 (2019)'s model selects *trophy* instead of *suitcase* to replace <u>it</u> in the original sentence. Table 3 (6) shows work by Talmor et al. (2020), expressing similar reasoning through a hypothesis-driven approach.

Prefix prompts (Li and Liang, 2021; Hambardzumyan et al., 2021; Lester et al., 2021) 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 autoregresive model (Liu et al., 2021b).

Prompts can be further augmented via adding demonstrations (*demonstration learning*) (Gao et al., 2021a). In that case, a few labeled examples are appended to the template to make it more informative, allowing the PLMs to better predicting the answer.

3.2.2 Template Construction

Templates can be either manually crafted or automatically generated. We here survey the different methods for generating them as well as ways to combine and manipulate the template-based prompts (multi-prompt learning).

Manually-crafted Templates. Most early work in prompt-based learning uses some form of manually crafted templates. For example, manual cloze

templates are used in Petroni et al. (2019) to probe the knowledge of the model, as well as in Schick and Schütze (2020), Schick et al. (2020) and Schick and Schütze (2021a) for text classification in a few-shot setting. Manually designed prefix prompts are leveraged in Brown et al. (2020) for QA, translation, and probing tasks for commonsense reasoning. The quality of the prompts impacts performance. Indeed, Zhao et al. (2021) showed that different prompts can cause accuracy to vary from near chance to near state-of-the-art.

Automatically-generated Discrete Templates. Discrete templates, which usually correspond to natural language phrases, are described in a discrete space. To search for such templates given a set of inputs and outputs, Jiang et al. (2021) proposed a mining-based approach called MINE that aims to find either the middle words or dependency paths between the inputs and outputs. A second approach (Jiang et al., 2021; Yuan et al., 2021) consists of paraphrasing an existing template prompt using back and forth machine translation, and then selecting the best prompt among the new paraphrases with guidance from a thesaurus. Prompt paraphrasing is also used by Haviv et al. (2021) 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 prompt (Wallace et al., 2019; Shin et al., 2020). Gao et al. (2021a) and Ben-David et al. (2021) further generate prompts using standard generation models such as T5 (Raffel et al., 2020). In the latter, the authors proposed a domain adaptation algorithm that trains 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 (i.e. the prompts do not correspond to actual words) and from the parameters of the LM (Liu et al., 2021b). These continuous prompts often require tuning on task-specific data. Li and Liang (2021) 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. (2021), who differ from Li and Liang (2021) 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. (2021) 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 as in Zhong et al. (2021b), Qin and Eisner (2021) and Hambardzumyan et al. (2021). It can also be done by inserting some tunable embeddings into a hard prompt template as in Liu et al. (2021c) and Han et al. (2021b), 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).

It is worth noting that Logan IV et al. (2021) showed that fine-tuning PLMs in the few-shot setting can avoid prompt engineering, and that one can use 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 and still achieve competitive accuracy on NLU tasks.

Multi-Prompt Learning A number of approaches use prompt ensembling, augmentation, and decomposition/composition for a more flexible task design. We describe them below.

First, multiple prompts can be used for an input (dubbed *prompt ensembling*) at inference time. The prompts can be combined using a uniform average (Jiang et al., 2021; Schick and Schütze, 2021a; Yuan et al., 2021) or a weighted average (Jiang et al., 2021; Qin and Eisner, 2021; Schick and Schütze, 2021a,b). Another way to combine the prompts is majority voting to combine the results of the different prompts as in Lester et al. (2021) and Hambardzumyan et al. (2021). Knowledge distillation (Allen-Zhu and Li, 2021), where the idea is that the knowledge present in an ensemble of models can be distilled into a single model, has been borrowed to the context of prompt combination by Schick and Schütze (2021a,b); Schick and Schütze (2020) and Gao et al. (2021a) where for each template-answer pair a separate model is trained, 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 (2020) trained a separate model for each prompt. Then the model outputs were scored by averaging their generation probability across all models.

Second, 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. For example, Cui et al. (2021) proposed an approach for named entity recognition, where the different prompts for each candidate span were created and predicted separately.

Third, augmentation methods such as *demonstration learning* (Gao et al., 2021a) create more descriptive prompts, as in a multiple-choice problem. Lu et al. (2021a) showed that both the choice of examples in the prompts and the order of the prompts can considerably affect the results. To select the examples from which the PLM must choose the correct response (*example sampling*), Gao et al. (2021a) and Liu et al. (2021a) used sentence embeddings to find examples semantically close to the input. Mishra et al. (2021) used both positive and negative examples, teaching the PLM types of items to avoid in performing new tasks with only instructions. As for the order of the selected examples (*sample ordering*), Kumar and Talukdar (2021) searched for the best permutation of prompts and also learned a segmentation token to separate between the prompts. They showed the usefulness of this method for few-shot learning on the task of 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. Yin et al., 2019; Cui et al., 2021), and those intended as the final answer (e.g. Petroni et al., 2019; Jiang et al., 2020; Radford et al., 2019). 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. (2021) 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 (2021a), Schick et al. (2020), Shin et al. (2020) and Gao et al. (2021a), 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 (Gao et al., 2021a) and then an answer is searched 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. (2021d) the decomposed relation labels (their individual tokens) represent the answer space. Lastly, Hambardzumyan et al. (2021) 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 usually applied to every training and test example as in the PET-TC (Schick and Schütze, 2021a), PET-Gen (Schick and Schütze, 2020) and LM-BFF (Gao et al., 2021a) models. Le Scao and Rush (2021) 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. (2021) 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 (Houlsby et al., 2019), which are neural

13

network layers inserted between the feed-forward portion of the Transformer architecture (see Section 2.4.4); (ii) BitFit (Zaken et al., 2021), where only the bias terms inside the Transformer are updated; (iii) PLM head tuning, where the embeddings in the MLM output layer that are associated with the tokens of the verbalizer are updated; and (iv) Calibration (Zhao et al., 2021), where an affine transformation on top of the logits associated with the verbalizer tokens is learned. They found that the best results are achieved using BitFit.

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 (Ben-David et al., 2021), where the prompts are properties of source domains, generated based on their relatedness to the input example (from a new domain), and P-Tuning (Liu et al., 2021c), which makes use of trainable continuous prompt embeddings when applying GPT models on NLU tasks. Finetuning both the model and the prompt-relevant parameters makes this approach very expressive. On the other hand, it requires the storage of all the parameters, with makes it less applicable to small datasets (Liu et al., 2021b).

It is worth noting that task-specific training can also be used earlier during the construction and validation of the prompts. Indeed, as pointed out by Perez et al. (2021), 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. (2021) 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 (2019), natural language descriptions of classification tasks were given as input. Then, the model was trained to generate the correct answer in natural language via a language modeling objective, aiming to generalize to new classification tasks without task-specific tuning.

Information Extraction (IE). Cui et al. (2021) considered the NER task as a language model 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. (2021b) proposed a model called Prompt Tuning with Rules (PTR), which applies logic rules to construct prompts with several sub-prompts. Chen et al. (2021d), 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. (2021) proposed using the event type and an integer representing the argument type as prefixes.

Knowledge Probing. Factual probing has been explored in particular by Petroni et al. (2019) and Jiang et al. (2020) 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. (2020), Jiang et al. (2021), Haviv et al. (2021), Shin et al. (2020) and Perez et al. (2021). Continuous template learning was used in Qin and Eisner (2021), Liu et al. (2021c) and Zhong et al. (2021b). Prompt ensemble learning was applied to knowledge probing by Jiang et al. (2021) and Qin and Eisner (2021).

In addition to factual knowledge, additional types of knowledge that have been probed using the cloze test include commonsense (Trinh and Le, 2019), relational knowledge (Petroni et al., 2019), reasoning (Talmor et al., 2020) and understanding rare words (Schick and Schütze, 2019). For commonsense reasoning, Winograd Schemas (Levesque et al., 2012) 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. (2019) 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 (Brown et al., 2020), negation (Ettinger, 2020) or semantic similarity (Sun

et al., 2021). Linguistic evaluation of language models (Linzen et al.; Gulordava et al., 2018; Goldberg, 2019; Tran et al., 2018; Bacon and Regier, 2019; McCoy et al., 2020; Linzen, 2020) 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, Warstadt et al., 2020a) which provides minimal pairs for various grammatical phenomena. Recently, the use of this benchmark was adapted for language acquisition research (Huebner et al., 2021): the authors probe a RoBERTa-based model pre-trained on transcriptions of child-directed speech (MacWhinney, 2000) to complete the benchmark task. The preference score can be calculated either *holistically*, summing the cross-entropy errors at each position in the sentence (Zaczynska et al., 2020; Huebner et al., 2021), or in an *MLM-based* way, where each candidate sentence is masked by a language model multiple times with the mask changing position. The score is computed by summing the log-losses at the different masked positions (Salazar et al., 2020).

Other tasks. The PET procedure (Schick and Schütze, 2021a) was also applied to the Textual Entailment task. QA is addressed in Khashabi et al. (2020) 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 used for the evaluation of text generation. Yuan et al. (2021) used prompts in the BARTSCORE-PROMPT variant of the BARTSCORE measure they propose that 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 machine translation evaluation.

Schick et al. (2021) showed that PLMs are able to recognize the toxicity of the text they produce (self-diagnosis). Then they proposed an algorithm that permits the language model to produce less

problematic text (self-debiasing), using a textual description of the undesired behavior.

Shin et al. (2021) explore the use of PLMs as few-shot semantic parsers. The authors use GPT-3 to convert text into a canonical text (in a controlled sub-language) satisfying a grammar, that is then 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 instead of self-supervised language modeling for the target task. Indeed, taking advantage of large NLU datasets for extra supervision results in better zero and few-shot performance in the target task with relatively small PLMs (Wang et al., 2021b), commonly RoBERTa$_{large}$ at 345M parameters. Knowledge-rich classification tasks in particular benefit from PLM proxy tasks, because the latter can reformulate the class label as a prompt, taking 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 Textual Entailment (Section 3.3.2).

3.3.1 Question Answering as Proxy Task

In a strong move away from traditional information extraction, 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.

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 information extraction 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

Application	Work	Task design	Prompt design
Relation Extraction	Li et al. (2019a)	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. (2021)	Use textual entailment to determine the likelihood of a candidate relation (such as Place-OfDeath(X,Y) given an input sentence.	Input: Gary's car crash occurred in Houston; Prompt: Gary died in Houston
Event Extraction	Du and Cardie (2020)	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. (2019)	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 (2019)	Use question answering 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. (2020b)	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?

Table 4: Examples of task design and example prompts for four different applications of prompt-based proxy tasks.

event of type Exchange-Ownership (see Figure 3, right box).

Li et al. (2020c) formulates **Named Entity Recognition (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. (2020b) 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. (2017) first formulated relation extraction as a QA task. This approach has been pursued in the context of PLMs by Li et al. (2019a) and Zhao et al. (2020b). Han et al. (2021b) 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. (2021) 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. (2020c) reformulated **event extraction** as a cloze task with QA model based on BERT and the SQuAD 2.0 dataset (Rajpurkar et al., 2018). Question answering is used directly, preserving the QA format, in Du and Cardie (2020), Feng et al. (2020a), Li et al. (2020a), Zhou et al. (2021) and Liu et al. (2020a) 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. (2020a) 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. (2021) where QA-based prompts are used for argument extraction and prompts based on Textual Entailment (Dagan et al., 2013) 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 (Michael et al., 2018) achieved the best results for argument extraction.

Identification-only sub-tasks such as **trigger identification** (Du and Cardie, 2020), are addressed by more general questions, e.g. "What is the trigger?". In contrast, Zhou et al. (2021) uses separate questions to address the identification and classification of arguments.

Du et al. (2021a) addressed **slot-filling**, which aims to extract task-specific slot fillers (for example, 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 a QA model improved the performance, compared to a direct encoding of the slot descriptions.

Lastly, Gao et al. (2019) formulated the **dialogue state tracking** task that aims to the estimate of the current belief state of a dialog given all the preceding conversation, as a QA problem. 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. (2020b) 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 (2020) who further improved the results. Namazifar et al. (2020) used this approach to address language understanding problems in the dialogue context, experimenting on ATIS (Airline Travel Information Systems, Hemphill et al., 1990) and on the Restaurants-8k dataset (Coope et al., 2020).

QA Task Design. Questions are typically generated via hand-crafted templates derived from the 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 Liu et al., 2020a; Lyu et al., 2021) or another argument that was previously identified (Li et al., 2020a) (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. (2020a), where monolingual unsupervised machine translation (Lample et al., 2018) 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. (2020b). The input to the QA system need not necessarily 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. Li et al., 2020c; Du and Cardie, 2020).

PLMs fine-tuned on the SQuAD 2.0 dataset (Rajpurkar et al., 2018) or on QAMR are particularly useful to initialize QA-style prompt-based learning methods.[9] With the advent of web-scale QA datasets (Huber et al., 2021), QA-infused PLMs may provide significantly richer representation, enabling a wider range of applications.

3.3.2 Textual Entailment as Proxy Task

Textual Entailment is a popular proxy for classification tasks (Yin et al., 2019), as these models have shown a striking ability to perform few-shot learning. Wang et al. (2021b) 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 textual entailment 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. The inference is performed by selecting the most probable candidate hypothesis given the input. Some recent works also make use of multiple verbalizations for each label to boost the system performance (Sainz and Rigau, 2021; Sainz et al., 2021).

Sainz et al. (2021) also proposed an approach to guiding the "art" that is prompt crafting more towards a "science": the authors fine-tune a model on Textual Entailment data and use the model's probability of a prompt given the template, applied

---

[9]Fine-tuning on a PLM on QAMR corresponds to the p-QuASE representation presented in He et al. (2020).

on the guideline example(s), to measure the quality of manually designed prompts.

Obamuyide and Vlachos (2018) reformulated **relation extraction** as a textual entailment task. This approach has been pursued in the context of PLMs by Sainz et al. (2021).

Roughly equivalent to textual entailment is Yes/No Question Answering (Clark et al., 2019a) where a model is asked about the veracity of some fact given a passage. It has also been used as a proxy task for text classification by Zhong et al. (2021a).

PLMs needs to be fine-tuned to solve the textual entailment task. They are commonly fine-tuned on MNLI (Williams et al., 2018), but other datasets such as SNLI (Bowman et al., 2015), FEVER (Thorne et al., 2018), ANLI (Nie et al., 2020) or XNLI (Conneau et al., 2018) are also used. In addition, data from different tasks can be used when framed properly (Zhong et al., 2021a).

4 Paradigm 3: NLP as Text Generation

The success of generative Transformer-based PLMs[10] such as GPT, BART, and T5 has 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. For example, Figure 4 illustrates this "text-to-text" approach as described in Raffel et al. (2020). Instead of using traditional discriminative models for NLP tasks, these tasks are reformulated as text generation problems so that they can be directly solved with generative PLMs. The generated output sequences usually include the desired labels or other auxiliary information for the given task, enabling accurate reconstruction of the expected class labels (i.e. to avoid ambiguities in mapping) and facilitating the generation/decoding process (i.e. to provide sufficient context for predictions).

It is worth noting that some NLP tasks are already text generation tasks. Therefore, a straightforward strategy for those tasks is to fine-tune a generative PLM using task-specific training data to perform the specific tasks of interest. Examples include Machine Translation (Cooper Stickland et al., 2021), text summarization (Lewis et al., 2020), text style transfer (Lai et al., 2021), etc. We refer read-

---
[10]In this section and next, we use the term PLM to refer to a generative PLM.

ers 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 that could be used 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 $P(y|x)$. In this formulation, the desired labels/outputs for the task on x need to be retrieved unambiguously from y, requiring y to be generated in a valid format by the design of the reformulated task. 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. 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 text-to-text/seq2seq framework can be used to solve different NLP tasks via encoder-decoder architectures, thus facilitating multi-task learning and transfer learning across tasks of different natures (Raffel et al., 2020). 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 (Paolini et al., 2021). Finally, when adapting to structure prediction problems, PLM-based models can naturally capture the inter-dependencies between prediction steps/tasks in the modeling process to further improve the performance (Athiwaratkun et al., 2020).

As such, the formation of the output sequence y for an input x is critical for the performance of the PLM-based methods. Existing works tend to customize such output sequences for specific NLP tasks to better capture the nature of the tasks. Therefore, in the rest of this section, we group
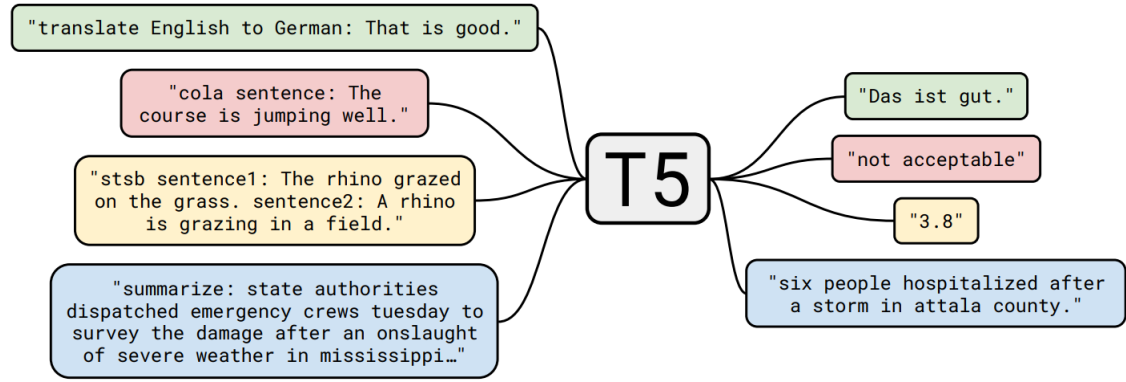
Figure 4: An illustration of T5 (Raffel et al., 2020) text-to-text text generation approach for Machine Translation, linguistic acceptability, text semantic similarity and summarizing tasks. Figure source: Raffel et al. (2020).

prior works according to their strategies in designing the output sequences to solve NLP tasks with generative models, and discuss their representative techniques in each subsection. 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 desired label annotations for x for a given NLP task. The repetition of the words from the input text aims to provide explicit context to reduce ambiguity for the generation process (Paolini et al., 2021). This strategy is often applied to structure prediction tasks that aim to jointly extract the text spans of interest and their relations or dependencies in an input text.

Athiwaratkun et al. (2020) 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 Named Entity Recognition (NER). Given an input sentence, the output sequence is formed by marking the token sequences for the slots or entity types of interest, for instance with square brackets or another identifier. The corresponding labels are then introduced immediately after the token sequences, within the brackets, separated by the token by a bar "|". The encoder-decoder PLM T5 is used to generate label-augmented texts. Paolini et al. (2021) extends this idea to other structure prediction tasks, including joint entity and relation extraction, relation classification, semantic role labeling (SRL), event extraction, coreference resolution, and dialogue state tracking. To encode a relation between two text spans 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, the input sentence x can be transformed into the label-augmented output sequence y, where (1) the square brackets indicate token spans for entity mentions; (2) `person` and `book` are the corresponding entity type labels; and (3) `author=Tolkien` indicates the author relation between `Tolkien` and `The Lord of the Rings`:

```
x = Tolkien's epic novel The
Lord of the Rings was
published in 1954-1955.

y = [Tolkien|person]'s epic
novel [The Lord of the
Rings|book|author=Tolkien]
was published in 1954-1955.
```

In order to transform the generated label-augmented texts into desired annotations, Paolini et al. (2021) 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. (2021a) 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. (2021a), also propose a normalization prediction mechanism: if a generated token does not belong to the original sentence or set of expected labels, the closest word from the input sentence using the Levenshtein distance is used instead.

Due to the unified text-to-text formulation, label-

Output Type	Work	Task	Example	
			Input	Output
Label-augmented Text	Paolini et al. (2021)	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] yester-day
		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. (2020)	Slot Filling	Add Kent James to the Disney soundtrack	((AddToPlaylist)) Add [Kent James	artist] to the [Disney	playlist]
		Named Entity Recognition	He is John Wethy from NBC News	He is [John Wethy	person] from [NBC News	org]
	Zhang et al. (2021a)	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. (2021b)	Named Entity Recognition	have muscle pain and fatigue	2 3 7 2 5 6
	Yan et al. (2021a)	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. (2020)	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. (2021a)	Closed-book QA	What is Southern California often abbreviated as?	Southern California, often abbreviated SoCal, is . . . ANSWER SoCal
	Hsu et al. (2021)	Answer Selection	How a water pump works?	A water pump is a device that moves fluids by mechanical action.
Filling Templates	Du et al. (2021b)	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. (2021c)	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. (2021)	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. (2021b)	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. (2020)	Answer Selection	<bos>Ice formations in the Titlis glacier cave <boq>How are glacier cave formed <eoq>	0.5
	Nogueira et al. (2020)	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. (2021)	Entity Retrieval	Superman saved [START] Metropolis [END]	Metropolis (comics)	Metropolis (1927 film)
	Cui et al. (2021)	Named Entity Recognition	ACL will be held in Bangkok	Bangkok is a location

Table 5: A summary of methods reformulating NLP task as a generation task solved by PLMs.

augmented text generation allows multi-task learning where a single generative model can be trained to simultaneously perform multiple tasks of different natures. Paolini et al. (2021) and Athiwaratkun et al. (2020) show that learning from 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 (Paolini et al., 2021), 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 words that are not in the input text, 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. (2021b) 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 (Vinyals et al., 2015)). BART is used as the base model for the index generation for NER.

Similarly, Yan et al. (2021a) generates indices

of the spans of interest for variations of the aspect-based sentiment analysis (ABSA) task, including aspect term extraction, opinion term extraction, aspect-level sentiment classification and aspect-oriented opinion extraction. Finally, casting a problem into an index generation task is also proposed for semantic parsing (i.e. filling slots) (Rongali et al., 2020). 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 indexes 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. (2021a) 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 question and answer 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 those 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. (2021) applies answer generation to 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* (Han et al., 2021a), Hsu et al. (2021) 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 maybe replaced by another answer candidate).

4.4 Filling templates

For many extraction tasks, the output 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. (2021b) 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 (following an order). The roles with no fillers are associated with *null*. (Zhang et al., 2021a) also examines a similar approach of tuple generation for ABSA.

The template filling methods can also introduce additional information into the templates to aid the label generation process, such as natural descriptions or definitions of the labels. In particular, Li et al. (2021c) 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 $arg_1, ..., arg_6$ and the document text DOC_TEXT are concatenated, and output y, in which the underlined text spans are fillers from DOC_TEXT) from Li et al. (2021c):

```
x = ⟨s⟩ ⟨arg₁⟩ bought, sold,
or traded ⟨arg₃⟩ to ⟨arg₂⟩ in
exchange for ⟨arg₄⟩ for the
benefit of ⟨arg₅⟩ at ⟨arg₆⟩
```

```
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 (Lu et al., 2021b) directly generates the event structures y:

```
  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))
```

Graph traversal algorithms are often used to accomplish the linearization in this approach. Ren et al. (2021) study structure-linearization for joint entity and relation extraction (Li et al., 2014; Miwa and Bansal, 2016). The main idea is to construct an information graph for each input sentence to capture entity mentions, their entity types, and relations. Depth or breath 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. (2021) linearize the information graph to an alternating sequence of nodes and edge types (given depth/breath first traversal), and directly generate such sequences via a hybrid span decoder that decodes both the spans and the types recurrently. For event extraction with joint extraction of event triggers and arguments (Li et al., 2014; Nguyen et al., 2016), a structure-linearization and text generation approach comes from Lu et al. (2021b). 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 (Chen et al., 2020a; De Cao et al., 2021) 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. (2021b) also report the effectiveness of the generation-based model for extending models to extract 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-sytle 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 can excel. The idea has its roots in the classical literature on probabilistic models for information retrieval that rank documents using language models (Ponte and Croft, 1998; Lafferty and Zhai, 2001). 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. (2020) apply the idea for answer selection by fine-tuning generative models (GPT-2 or BART) 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. (2020) 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. (2020) 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. (2021) 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. (2021) 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 is then employed to score this pair (using the probability of the template output produced by the decoder of BART). For each span, the entity type corresponding to the template with highest score is selected. Original NER training data is used to create gold standard templates to fine-tune BART.

In addition to question answering, 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 in a recent work by Shin et al. (2021). Authors show that 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 Data Generation via PLM

In addition to using PLMs to perform NLP tasks directly, PLMs can be used to generate data that can be used to enhance the performance of NLP systems in two ways. Note that these data generation approaches are complementary to the three paradigms of PLM-for-NLP discussed in previous sections.

First, data generated by PLMs can be combined with original training data to improve NLP models where training data is too sparse. Typically, this is applied to create new labeled data to increase diversity, enrich the models, and otherwise alleviate common limitations of hand-labeled data.

The studies presented below discuss, for various downstream NLP tasks: approaches for fine-tuning PLMs to ensure they capture the key characteristics of the task when performing data generation; appropriate reformulation of the original training data for PLM fine-tuning and generation; and filtering the new data for noise introduced by the generation process.

Second, we discuss the use of auxiliary data generated by PLMs to shed light on interesting aspects of NLP models. This approach plays a role in machine learning explainability by providing generations such as counterexamples, clarifying questions, context for answers, inference rules, and other insight-rich sequences.

5.1 Augmenting NLP Models with Automatically Generated Data

Traditional approaches to data augmentation, including generation via semi-supervised learning on large unlabeled data sets and synthesis with back-translation or synonymous word replacement (Feng et al., 2021; Chen et al., 2021a) were shown to be effective for increasing NLP models' accuracy and/or coverage. Newer studies show that PLMs can be also used as an effective method for data augmentation (Zhang et al., 2020a; Yang et al., 2020; Peng et al., 2020; Kumar et al., 2020; Anaby-Tavor et al., 2020), requiring no significant change to the model architecture. The fluency of PLM text generations stand in contrast to the outcomes of traditional approaches that may produce less natural samples. As discussed in previous sections, the massive amount of linguistic knowledge accumulated by the PLM allows for adaptation to many domains and tasks, including those with very limited labeled data. The vast knowledge may also produce a greater variety of new examples, further improving the NLP models trained on them.

We organize the discussion of data augmentation methods according to the NLP tasks they support.

5.1.1 Information Extraction (IE)

Prior works explored synthetic data generation with PLMs (Madaan et al., 2020; Bosselut et al., 2019) for a variety of IE tasks.

Veyseh et al. (2021a) and Veyseh et al. (2021b) use GPT-2 to produce synthetic labeled data for event detection. Sentences in existing training datasets are augmented with markers to indicate positions of event trigger words. The resulting labeled sentences are used to fine-tune GPT-2 us-

ing the standard autoregressive next word prediction (NWP) objective. Veyseh et al. (2021a) shows that the fine-tuned GPT-2 model can generate label-augmented data for different domains (e.g. newswire, cybersecurity); however, the generated data might include some noise, for instance, incorrect grammar, meaningless sentences, or incorrect annotations. To minimize the impact of the noisy generated examples and maximize the benefits of the generated data, Veyseh et al. (2021a) and Veyseh et al. (2021b) present a student-teacher network framework: the teacher network is trained on the original labeled data to obtain anchor knowledge, while the student is trained over the combination of original and synthetic data, with constraints introduced to enforce consistency with the teacher's learned anchor knowledge. The framework leads to significant performance improvement over different datasets for event detection.

Guo and Roth (2021) employ GPT-2 to generate synthetic labeled data for cross-lingual NER following the annotation projection approach: training data in a source language is translated and projected into a target language to train models. To project annotation, a training sentence in the source language is first translated into the target language using word-to-word translation (via a dictionary). GPT-2 is then fine-tuned to generate complete sentences from the important words in target languages. A hard-constrained generation mechanism is also encoded into the decoding process of GPT-2 to ensure the appearance of the named entities in the original source sentence in the automatically generated sentences.

Synthetic data generation with GPT-2 is also explored for relation extraction in Papanikolaou and Pierleoni (2020). This paper fine-tunes GPT-2 over labeled examples of the same relation type, where each sentence in the training data is marked with the two entity mentions in the corresponding relation. The fine-tuned model for each relation type is then leveraged to produce new training instances for that relation.

5.1.2   Question Answering (QA)

Given an input paragraph C and a sampled extractive short answer A in C, Alberti et al. (2019) attempts to generate a question Q using a sequence-to-sequence Transformer (with BERT as its encoder). The triple, consisting of the input paragraph, the generated question, and the sampled answer (C, Q, A), can be used as a new training instance for QA models. To mitigate the noise in the generated data, Alberti et al. (2019) present a round trip consistency approach where a second generative model is trained to take the input passage C and generated question Q from the prior step to produce an answer A'. The tuple (C, Q, A) is only retained as new training data if $A' == A$.

Following a similar principle, Shakeri et al. (2020) explore synthetic data generation for cross-domain QA where models trained on a source domain (typically SQuAD) are evaluated on datasets from a different target domain. The paper aims to generate QA pairs in the target domain and combine them with the source-domain training data to train improved QA models. The data generation model is also trained on the source domain dataset SQuAD using BART and GPT-2. Starting with a passage as the context, the generative models directly generate QA pairs. Generated QA pairs are filtered by the likelihood scores of the generative models to reduce noise.

The data generation idea is extended to multi-hop QA that requires combining disjoint pieces of evidence to answer a question. In particular, Pan et al. (2021b) aim to generate human-like multi-hop question–answer pairs to train QA models. The model consists of three components: operators, reasoning graphs, and question filtration. Operators are atomic operations that are implemented by rules or off-the-shelf pretrained models to retrieve, generate, or fuse relevant information from input contexts. Approaches to fusing relevant information from across contexts include: fine-tuning a T5 model on SQuAD to generate single-hop questions; generating descriptions of table entities with GPT-TabGen (Chen et al., 2020b); and combining single-hop questions with sentences about the same entities to produce multi-hop questions via filling in masked tokens of designed templates. Reasoning graphs then define different types of reasoning chains for multi-hop QA using the operators as building blocks. Training QA pairs are generated by executing the reasoning graphs, which generate output texts. Finally, question filtration removes irrelevant and unnatural QA pairs to produce the final generated training set for multi-hop QA. The filtration is done by choosing the samples ranked as most fluent by GPT-2, and paraphrasing each generated question using BART.

5.1.3   Sentiment Analysis (SA)

Yu et al. (2021a) applies data augmentation for

aspect-based SA in the unsupervised domain adaptation setting, aiming to transform labeled datasets in a source domain to a new target domain. The main approach involves two steps. In the first step of domain generalization, domain-specific words and phrases in the labeled source data and unlabeled target data are identified and masked in the inputs. Opinion words for the source domain and target-specific terms and opinion words are retrieved via sentiment lexicon and bootstrapping methods using relations in dependency trees. The target-specific terms in the unlabeled data will be masked to fine-tune BERT. In the second step of *domain specification*, the source-specific terms in the source data are masked (thus producing domain-independent texts) and sent into the fine-tuned BERT to produce labeled sentences in the target domain. Here, some constraints based on dictionaries are necessary to ensure that the infilled words are terms or opinion words with the same sentiment polarity. The generated data can be used independently or combined with original source training data to train a SA model for the target domain.

Li et al. (2020b) use PLMs to generate synthetic data for aspect term extraction (cast as a sequence labeling problem). To fine-tune PLMs with the sequence-to-sequence framework for this purpose, the input includes a masked sentence from a training dataset and the corresponding label sequence while the output are the masked tokens in the input. The fine-tuned PLMs are then exploited to generate new possibilities for the masked tokens that can be injected into the masked input, using the original label sequence to obtain synthetic labeled data to train models.

5.1.4 Fact Verification

Fact verification aims to predict whether a given claim is supported, denied, or unresolved based on the given evidence. Automatically generated texts can be used to generate claim-evidence pairs for each label category. To this end, Pan et al. (2021a) employ a two-step approach to generate synthetic data for fact verification. In the first step of question generation, given the evidence and an answer, a BART model, fine-tuned on the SQuAD dataset using the similar input-output format, generates a question for that answer. Next, a question-to-claim model is employed to take the question and answer as inputs and generate a claim (also using a BART model fine-tuned on SQuAD). To produce ⟨claim, evidence⟩ pairs with the "support" relation, an en-

tity is selected in the original evidence in the first step of the process. To produce a "refute" claim, the work replaces the original answer with another entity in the generation process. Finally, to create a "not-enough-evidence" claim, the paper expands the original evidence to include other paragraphs in the same document and produce claims for some entity in the extended paragraph. Experiments show competitive results when the augmented data is combined with few or even no human-labeled examples for model training.

5.1.5 Document Classification

A typical approach to generating synthetic data for text classification is to build a conditional generative model for each class by fine-tuning with labeled data from that class. While these models can be fine-tuned with the next word prediction objective with generative PLMs such as GPT-2, Liu et al. (2020b) use reinforcement learning to train generative models to augment text classification labeled data. The rewards for training are based on the similarity between the generated tokens and a salient lexicon of the target class computed via top frequency-based salient words, and the divergence between the conditional and unconditional models. Liu et al. (2020b) demonstrate the effectiveness of using the automatically generated data in multiple text classification problems and datasets, including sentiment analysis and offense detection.

5.2 Generating Auxiliary Data to Improve Different Aspects of NLP Models

The following sections, again arranged by task, discuss ways of using PLM-generated text to aid in auxiliary tasks, helping developers or users understand model strengths and weaknesses or decision-making characteristics.

5.2.1 Explaining Models' Decisions

Despite the impressive performance of deep learning models for various NLP tasks, a remaining challenge to widespread adoption is the lack of explanations for the models' decisions. This hinders the development and debugging process, as well as user trust. This is especially true for application domains such as healthcare, security, and online education. As such, a considerable number of approaches have been proposed for explaining deep learning models' behavior, including model-intrinsic (Ribeiro et al., 2016; Lundberg and Lee, 2017; Chen et al., 2018) and model-agnostic ap-

proaches (Park et al., 2018; Kim et al., 2018; Ling et al., 2017). While model-intrinsic explanations expose internal model state (e.g. feature importance or attention scores), in model-agnostic (post-hoc) methods, explanations are generated via the model predictions without inspecting the internal state. Generative models are often applied for post-hoc explanations, aiming to obtain either counterexamples (Kim et al., 2016; Wachter et al., 2018; Wu et al., 2021a) or natural language texts (Camburu et al., 2018; Kumar and Talukdar, 2020; Chen et al., 2021c) for explaining purposes.

Generating *counterexamples* can shed light on the decision boundaries of the models (i.e. explaining when a model changes its decision), thus improving intepretability. To this end, the generated counterexamples should be close to the decision boundaries so that small modifications result in changing the model predictions. Traditionally, heuristic rules applied to the original inputs create likely counterexamples (Wachter et al., 2018; Ribeiro et al., 2018; Iyyer et al., 2018; Li et al., 2021a). PLMs have been leveraged to generate more diverse examples for better evaluation (Madaan et al., 2021b; Wu et al., 2021a; Ross et al., 2021). In particular, Wu et al. (2021a) proposes a method based on GPT-2 to generate counterfactuals that are close to the original sentences and entail specific relationships with the original, facilitating label induction (e.g. negation, insertion, shuffle). Concretely, an input sentence is concatenated with a relation label (e.g. negation) and a template consisting of the special tokens [BLANK] to form the prompt for GPT-2 model. For instance, for the sentence "*It is great for kids*" and the relation label "`negate`", the following prompt is constructed: "`It is great for kids. [negation] It is [BLANK] great for [BLANK]. [SEP]`". Next, the GPT-2 model generates answers for the [BLANK] in the template (e.g. "`not [ANSWER] children`", separated by the special token [ANSWER]). To fine-tune the GPT-2 model, non-parallel datasets (e.g. CommonGen, Natural Questions and SQuAD) are automatically processed to find the relations between pairs of sentences and to construct the templates for each relation based on the obtained pairs. It is worth noting that the sentences generated by GPT-2 might have the same label as the original input sentence. In addition, Wu et al. (2021a) show that the generated counterexamples can be helpful to improve the performance of the downstream models, e.g. for natural language inference, duplicate question detection, and sentiment analysis.

Other research is informing the task of *natural language explanation generation*, where the goal is to expose the rationale behind the model decisions in automatically generated natural language text. Any approach must critically require that the generated response is faithful to the model behavior. To this end, Kumar and Talukdar (2020) propose to first generate the explanations, and then employ the explanations to obtain the final model predictions. They use natural language inference as the task requiring explanations. Label-specific GPT-2 models are fine-tuned over concatenations of corresponding premises, hypotheses, and human-provided explanations, so that at inference, the model generates an explanation based on premise and hypothesis. Next, the explanations together with the premise and the hypothesis are consumed by an explanation processor model (e.g. RoBERTa) to select the most likely label. This process obtains a more faithful explanation for the label choice, compared to traditional prediction-first approaches (Camburu et al., 2018). However, this approach does not provide explanations that reference non-selected labels. To address the question of why other labels are not chosen, Chen et al. (2021c) exploit counterexamples, deriving them from original samples with heuristic rules. The original samples and counterexamples are provided to GPT-2 to generate an explanation for the question "*Why A not B*".

5.2.2 Knowledge Extraction

Generative PLMs are pre-trained on massive text corpora containing a large amount of information about entities and commonsense knowledge. As such, PLMs might directly be used to elicit knowledge required for downstream applications such as information extraction, sentiment analysis and question answering. To this end, it is important to properly prompt these models so their outputs contain the required information. Section 3.2 describes the prompt design for knowledge extraction/probing tasks, and in particular, the "Knowledge Probing" subsection describes applications in details. Here we focus on the text generation aspect of knowledge extraction approaches.

Prior works can be categorized into two subcategories. The first category involves prompting PLMs with partial knowledge via a prompt and

asking the models to complete the prompt. Specifically, pre-defined templates can be designed and filled with partial knowledge (e.g. the two entities involved in a relation) and the generative PLMs can predict the missing words in the templates (e.g. the relation type between the two entities.) The templates can be fixed (Goswami et al., 2020) or they can be dynamically constructed by a pre-trained model (Shin et al., 2020) (further details are in Section 3.2). The second category instead proposes to prompt the PLMs with full knowledge and ask the models to generate a natural language text to describe that knowledge. This task is known as Data-to-Text (Kukich, 1983), and the goal is to obtain a textual description of existing knowledge bases. The generated textual descriptions can be used by downstream applications such as knowledge probing (Petroni et al., 2019) or QA (Agarwal et al., 2021), among others. Agarwal et al. (2021) introduce a model based on T5 to convert Wikidata knowledge graphs (with triples of relations between two entities) into textual data. The proposed approach consists of three stages. First, create a large but noisy training dataset using distant supervision for relation extraction by aligning knowledge base (KB) triples to Wikipedia texts. Next, fine-tune T5 in stages, starting with the distantly supervised dataset for better coverage, then moving on to a small clean dataset for less hallucination. The model learns to generate descriptive sentences from KB triples. Last, build a filter for the generated texts based on semantic quality with respect to the KB triples by scoring the concatenation of input and output with BERT.

5.2.3 Question Generation

While PLMs can be directly used for generating answers for questions, they might be also helpful to support existing QA systems. Specifically, PLMs can be employed to provide clarification for downstream QA systems. The clarification can be realized in terms of question clarification when the question is ambiguous or it can be fulfilled by providing more context. For instance, in Gao et al. (2021b) and Min et al. (2020), multi-step question generation approaches are proposed for ambiguous QA in which the BART model is prompted with an ambiguous question and the top similar passages retrieved in a document to generate candidate answers. If multiple answers are generated, another BART model is employed to generate a disambiguation question for each answer. The newly generated

questions are later used to extract other candidate answers. Finally, the generated answer-question pairs are ranked to select the top one for the ambiguous QA problem. Min et al. (2020) show that the process of generating auxiliary disambiguation questions could further help the models to encode the interactions between the original input question and the candidate answers.

In another line of work, Mao et al. (2021) seek to generate clarification texts for input questions to improve the retrieval quality in open-domain QA (answering factoid questions without a pre-specified domain). The most common approach for this problem involves a retriever-reader architecture (Chen et al., 2017), which first retrieves a small subset of documents in the pool using the input question as the query and then analyzes the retrieved documents to extract (or generate) an answer. To generate augmented texts for the input question in the first retrieval component, Mao et al. (2021) fine-tune BART to consume the input question and attempt to produce the answer and the sentence or title of the paragraph containing the answer. This method demonstrates superior performance for both retrieval and end-to-end QA performance.

In addition to clarification information, PLMs can also be used to paraphrase questions to support QA models. Mass et al. (2020) explore the problem of FAQ retrieval, retrieving the top QA pair given a user query. Based on the returned QA pairs (q, a) from a retrieval system, this work proposes an unsupervised method to re-rank the pairs to improve the performance. One of the ranking scores is a matching score between the question p in the pair (q, a) with respect to the user question. A triple network is trained over the tuples (p, q, q'), where q is a paraphrase of the question p while q' is randomly selected questions from other QA pairs. To this end, Mass et al. (2020) fine-tune GPT-2 over the concatenations of the corresponding answers and questions in the FAQ. The fine-tuned GPT-2 is then prompted with the answer a to produce a paraphrase q' for q in the ranking network.

5.2.4 Inference Rule Generation

For some applications, it is important to understand the process by which the final predictions of the models are obtained. These intermediate inference rules provide are another form of model explanation and provide insights for improving model performance.

Paul and Frank (2021) exploit GPT-2 to perform narrative story completion: given a few sentences of a story, the goal is to complete the story using sentences that logically follow the narrative in the given incomplete story. In an incremental generation method, each step seeks to generate a contextualized inference rule conditioned on the current incomplete story. To accomplish this, GPT-2 is fine-tuned on human annotation of story line inferences. Next, given the current story and generated inference rule, a new sentence for the story is generated (using another fine-tuned GPT-2 model). By interspersing the inference rules, the storyline generations should create a coherent story that follows logical connections and causal relationships between events.

Madaan et al. (2021a) employ T5 to generate inference graphs for defeasible inference (Rudinger et al., 2020). In this mode of reasoning, given a premise, a hypothesis may be weakened or overturned in light of new evidence. As training inference graphs for this problem requires a large amount of human-annotated inference graphs, they propose to exploit reasoning graphs in related tasks to fine-tune T5. In particular, this work leverages the influence graphs in the WIQA dataset that includes a set of procedural passages, each accompanied by a human-curated influence graph. The influence graphs are linearized to fit into the seq2seq framework for fine-tuning T5 and producing inference graphs for defeasible inference afterward. It has been shown that the generated inference graphs can improve human accuracy on defeasible inference (which is originally challenging for humans).

6 Discussion

Mix of paradigms or PLMs. The three paradigms presented in this paper 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. (2020), 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 (Wang et al., 2021c) 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.

How much unlabeled data is needed? While PLMs are usually trained on billions of words, some works have investigated what can be learned with less pre-training data. Zhang et al. (2021b), experimenting on RoBERTa models trained on 1M, 10M, 100M and 1B words (Warstadt et al., 2020b, MiniBERTas), showed that 10M to 100M words are sufficient to acquire many syntactic and semantic features. Huebner et al. (2021) presented BabyBERTa, a RoBERTa-based model trained on language acquisition data that acquires grammatical knowledge comparable to that of pre-trained RoBERTa-base – and does so with approximately 15x fewer parameters and 6,000x fewer words. On the other hand, Zhang et al. (2021b), using the pre-train then fine-tune paradigm for NLU tasks, found that millions of words are not sufficient for key NLU skills, which instead may require billions of words and continue improvements with additional pre-training data.

How much labeled data is still needed? While Le Scao and Rush (2021) present experiments to quantify the impact of prompts, there has been little work in designing rigorous experiments to study how many labeled examples are required by PLMs 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.

Can we reduce the amount and cost of computation? The development of deep learning in general and the use of PLMs in particular have dramatically increased the amount of computation used in NLP, leading to a high environmental footprint. Schwartz et al. (2020) 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. (2021) propose software optimization and design choices for pre-training BERT in 24 hours using a single low-end deep learning server.

Do PLMs excel at semantic understanding or memorization? Another interesting avenue to explore is separating extraction or text understanding from memorization. To what extent can PLMs memorize facts and extract an answer from a passage provided (understanding a text), for knowledge-intensive tasks such as Questions Answering (QA) and Information Retrieval (IR)? This is motivated by the observation by Wang et al. (2021a) that PLMs are terrible at remembering training facts with high precision and that it is also challenging for them to answer closed-book questions even if relevant knowledge is retained.

Is explicit linguistic information needed? A related 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. Below we list some successes in leveraging syntax or semantics, though there is no definite answer yet. In terms of syntax, Xu et al. (2021) utilize automatically produced syntax in both the pre-training and fine-tuning stages, and show improved performance on several benchmark datasets. Nguyen et al. (2020b) and Sachan et al. (2021) inject syntax only in the fine-tuning stage. Regarding semantics, Zhang et al. (2020d) incorporate Semantic Role Labeling predictions into the pre-training procedure of BERT, improving the performance on textual entailment and QA tasks. Wu et al. (2021b) integrate semantic information into the task-specific fine-tuning stage, focusing on the DELPHIN dependencies formalism or "DM" (Ivanova et al., 2012). Experimenting on RoBERTa, they obtained improvements on the GLUE benchmark. Syntax and semantics can also be jointly integrated, as in Zhou et al. (2020a), where multi-task learning was used to combine BERT pre-training with both semantic and syntactic parsing tasks, improving the performance on the GLUE benchmark.

Can we integrate implicit semantic information using QA? Instead of enriching PLMs with symbolic annotations, a possible alternative for a supervision signal is QA data, as it is easier to answer questions relative to a sentence than to annotate linguistic phenomena in it (Roth, 2017; He et al., 2020). In the s-QuASE PLM presented in He et al. (2020), further pre-training of BERT on QA datasets is done while restricting the interaction between the question and context inputs. s-QuASE is particularly useful in single-sentence tasks such as Semantic Role Labeling and NER. A similar direction was pursued by Jia et al. (2021) who leveraged question generation and knowledge distillation to build a QA-based pre-training objective.

Do PLMs need meaningful 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 (Mishra et al., 2021; Schick and Schütze, 2021a; Brown et al., 2020). 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 (2021) 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 (Shin et al., 2020). In this sense, the results of continuous prompts also show that PLMs do not need meaningful instructions for improving few-shot performance.

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 (Perez et al., 2021). 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 (Conneau et al., 2020).

7 Conclusion

In this paper, 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. In addition, we describe the use of pre-trained language models to automatically generate data that is used to improve performance in NLP tasks. We hope this survey will provide readers with key fundamental concepts and a comprehensive view of the paradigm shift.

Acknowledgments

References

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)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*.

Rodrigo Agerri, Iñaki San Vicente, Jon Ander Campos, Ander Barrena, Xabier Saralegi, Aitor Soroa, and Eneko Agirre. 2020. Give your text representation models some love: the case for Basque. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4781–4788, Marseille, France. European Language Resources Association.

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Zeyuan Allen-Zhu and Yuanzhi Li. 2021. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Asaf Amrami and Yoav Goldberg. 2019. Towards better substitution-based word sense induction. *arXiv preprint arXiv:1905.12598*.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

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)*.

Geoff Bacon and Terry Regier. 2019. Does bert agree? evaluating knowledge of structure dependence through agreement relations. *arXiv preprint arXiv:1908.09892*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

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*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

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)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. 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)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Eyal Ben-David, Nadav Oved, and Roi Reichart. 2021. Pada: A prompt-based autoregressive approach for adaptation to unseen domains. *arXiv preprint arXiv:2102.12206*.

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.

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*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel Broscheit. 2019. Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 677–685, Hong Kong, China. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*.

José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. 2020. Spanish pre-trained BERT model and evaluation data. In *PML4DC Workshop at ICLR*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's next language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Wanxiang Che, Longxu Dou, Yang Xu, Yuxuan Wang, Yijia Liu, and Ting Liu. 2019. HIT-SCIR at MRP 2019: A unified pipeline for meaning representation parsing via efficient training and effective encoding. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 76–85, Hong Kong. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2021a. An empirical survey of data augmentation for limited data learning in nlp. *arXiv preprint arXiv:2106.07499*.

Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. 2018. Learning to explain: An information-theoretic perspective on model interpretation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter,

Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021b. Evaluating Large Language Models Trained on Code. *arXiv:2107.03374 [cs]*. ArXiv: 2107.03374.

Pinzhen Chen, Nikolay Bogoychev, Kenneth Heafield, and Faheem Kirefu. 2020a. Parallel sentence mining by constrained decoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Qianglong Chen, Feng Ji, Xiangji Zeng, Feng-Lin Li, Ji Zhang, Haiqing Chen, and Yin Zhang. 2021c. KACE: Generating knowledge aware contrastive explanations for natural language inference. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL)*.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020b. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021d. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. *arXiv preprint arXiv:2104.07650*.

Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020c. Reading the manual: Event extraction as definition comprehension. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 74–83, Online. Association for Computational Linguistics.

Avihay Chriqui and Inbal Yahav. 2021. Hebert & hebemo: a hebrew bert model and a tool for polarity analysis and emotion recognition. *arXiv preprint arXiv:2102.01909*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. 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)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019b. 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*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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, pages 160–167, New York, NY, USA. Association for Computing Machinery.

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. *arXiv preprint arXiv:1911.02116*.

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*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Samuel Coope, Tyler Farghly, Daniela Gerz, Ivan Vulić, and Matthew Henderson. 2020. Span-ConveRT: Few-shot span extraction for dialog with pretrained conversational representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 107–121, Online. Association for Computational Linguistics.

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*, pages 3440–3453, Online. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics (ACL-IJCNLP)*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668, Online. Association for Computational Linguistics.

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):1–220.

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*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

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)*.

Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. RobBERT: a Dutch RoBERTa-based Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3255–3265, Online. Association for Computational Linguistics.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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)*, pages 671–683, Online. Association for Computational Linguistics.

Xinya Du, Luheng He, Qi Li, Dian Yu, Panupong Pasupat, and Yuan Zhang. 2021a. 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)*, pages 654–664, Online. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021b. 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)*.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*.

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):625–660.

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*, 8:34–48.

Mehrdad Farahani, Mohammad Gharachorloo, Marzieh Farahani, and Mohammad Manthouri. 2021. Parsbert: Transformer-based model for persian language understanding. *Neural Processing Letters*.

Joshua Feldman, Joe Davison, and Alexander M. Rush. 2019. Commonsense knowledge mining from pretrained models. *arXiv preprint arXiv:1909.00505*.

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

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics (ACL-IJCNLP)*.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020b. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Steven Fincke, Shantanu Agarwal, Scott Miller, and Elizabeth Boschee. 2021. Language model priming for cross-lingual event extraction. *arXiv preprint arXiv:2109.12383*.

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

Shuyang Gao, Sanchit Agarwal, Di Jin, Tagyoung Chung, and Dilek Hakkani-Tur. 2020b. From machine reading comprehension to dialogue state tracking: Bridging the gap. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 79–89, Online. Association for Computational Linguistics.

33

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*, pages 264–273, Stockholm, Sweden. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. 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)*, pages 3816–3830, Online. Association for Computational Linguistics.

Yifan Gao, Henghui Zhu, Patrick Ng, Cicero Nogueira dos Santos, Zhiguo Wang, Feng Nan, Dejiao Zhang, Ramesh Nallapati, Andrew O Arnold, and Bing Xiang. 2021b. Answering ambiguous questions through generative evidence fusion and round-trip prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Ankur Goswami, Akshata Bhat, Hadar Ohana, and Theodoros Rekatsinas. 2020. Unsupervised relation extraction from language models using constrained cloze completion. In *Findings of the Association for Computational Linguistics (EMNLP)*.

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)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2021. Parameter-Efficient Transfer Learning with Diff Pruning. *arXiv:2012.07463 [cs]*. ArXiv: 2012.07463.

Ruohao Guo and Dan Roth. 2021. Constrained labeled data generation for low-resource named entity recognition. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*.

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)*, pages 5297–5306, Hong Kong, China. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.

Rujun Han, Luca Soldaini, and Alessandro Moschitti. 2021a. 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)*.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021b. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*.

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*, pages 3618–3623, Online. Association for Computational Linguistics.

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*, pages 8743–8758, Online. Association for Computational Linguistics.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia. Association for Computational Linguistics.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

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 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

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*, pages 2790–2799. PMLR.

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)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

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)*.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2020. ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission. *arXiv:1904.05342 [cs]*. ArXiv: 1904.05342.

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. *arXiv preprint arXiv:2110.07731*.

Philip Huebner, Elior Sulem, Cynthia Fisher, and Dan Roth. 2021. Babyberta: Learning more grammar with small-scale child-directed language. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

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

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2020. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. *arXiv preprint arXiv:2010.05953*.

Tohoku University Inui Laboratory. 2021. Pretrained Japanese BERT models. GitHub.

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 Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train bert with an academic budget. *arXiv preprint arXiv:2104.07705*.

Robin Jia, Mike Lewis, and Luke Zettlemoyer. 2021. Question answering infused pre-training of general-purpose contextualized representations. *arXiv preprint arXiv:2106.08190*.

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*, pages 11–15, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020. X-FACTR: Multilingual factual knowledge retrieval from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5943–5959, Online. Association for Computational Linguistics.

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*, 8:423–438.

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*, 8:64–77.

Mandar Joshi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2019. Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

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 [cs, stat]*. ArXiv: 2001.08361.

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*, pages 1896–1907, Online. Association for Computational Linguistics.

Been Kim, Oluwasanmi Koyejo, and Rajiv Khanna. 2016. Examples are not enough, learn to criticize!

criticism for interpretability. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS)*.

Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John F. Canny, and Zeynep Akata. 2018. Textual explanations for self-driving vehicles. In *In Proceedings of the European conference on computer vision (ECCV)*.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics (ACL)*.

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*, pages 4507–4518, Online. Association for Computational Linguistics.

Sawan Kumar and Partha P. Talukdar. 2020. NILE : Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, (ACL)*.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *arXiv*.

Yuri Kuratov and Mikhail Arkhipov. 2019. Adaptation of deep bidirectional multilingual transformers for russian language. *arXiv preprint arXiv:1905.07213*.

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*.

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*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Huiyuan Lai, Antonio Toral, and Malvina Nissim. 2021. Thank you bart! rewarding pre-trained models improves formality style transfer. *arXiv preprint arXiv:2105.06947*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

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 [cs]*.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, and Didier Schwab. 2020. FlauBERT: Unsupervised language model pre-training for French. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2479–2490, Marseille, France. European Language Resources Association.

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*, pages 2627–2636, Online. Association for Computational Linguistics.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

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)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

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

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

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 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

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, pages 7871–7880, Online. Association for Computational Linguistics.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021a. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020a. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, Online. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021b. Pretrained language models for text generation: A survey. *arXiv preprint arXiv:2105.10311*.

Kun Li, Chengbo Chen, Xiaojun Quan, Qing Ling, and Yan Song. 2020b. Conditional augmentation for aspect term extraction via masked sequence-to-sequence generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

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)*. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021c. 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*, pages 894–908, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020c. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019a. Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350, Florence, Italy. Association for Computational Linguistics.

Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019b. Exploiting BERT for End-to-End Aspect-based Sentiment Analysis. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 34–41, Hong Kong, China. Association for Computational Linguistics.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. TruthfulQA: Measuring How Models Mimic Human Falsehoods. *arXiv:2109.07958 [cs]*. ArXiv: 2109.07958.

Jeffrey Ling, Nicholas FitzGerald, Zifei Shan, Livio Baldini Soares, Thibault Févry, David Weiss, and Tom Kwiatkowski. 2020. Learning cross-context entity representations from text. *arXiv preprint arXiv:2001.03765*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

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*, pages 5210–5217, Online. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020a. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pretrain, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv:2107.13586 [cs]*. ArXiv: 2107.13586.

Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. 2020b. Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021c. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020c. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

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 [cs]*.

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. *arXiv preprint arXiv:2106.13353*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021a. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021b. 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)*.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NIPS)*.

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)*, pages 322–332, Online. Association for Computational Linguistics.

Shuming Ma, Jian Yang, Haoyang Huang, Zewen Chi, Li Dong, Dongdong Zhang, Hany Hassan Awadalla, Alexandre Muzio, Akiko Eriguchi, Saksham Singhal, Xia Song, Arul Menezes, and Furu Wei. 2020. Xlm-t: Scaling up multilingual machine translation with pretrained cross-lingual transformer encoders. *arXiv preprint arXiv:2012.15547*.

Brian MacWhinney. 2000. *The CHILDES Project: Tools for analyzing talk. transcription format and programs*, volume 1. Psychology Press.

Aman Madaan, Dheeraj Rajagopal, Niket Tandon, Yiming Yang, and Eduard Hovy. 2021a. Could you give me a hint? generating inference graphs for defeasible reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*.

Aman Madaan, Dheeraj Rajagopal, Yiming Yang, Abhilasha Ravichander, Eduard Hovy, and Shrimai Prabhumoye. 2020. EIGEN: event influence generation using pre-trained language models. In *arXiv*.

Nishtha Madaan, Inkit Padhi, Naveen Panwar, and Diptikalyan Saha. 2021b. Generate your counterfactuals: Towards controlled counterfactual generation for text. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, (AAAI)*.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. 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)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL)*.

Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. 2021. Muss: Multilingual unsupervised sentence simplification by mining paraphrases. *arXiv preprint arXiv:2005.00352*.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Yosi Mass, Boaz Carmeli, Haggai Roitman, and David Konopnicki. 2020. Unsupervised FAQ retrieval with question generation and BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

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*, 8:125–140.

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)*, pages 560–568, New Orleans,

Louisiana. Association for Computational Linguistics.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics.

Mahdi Namazifar, Alexandros Papangelis, Gokhan Tur, and Dilek Hakkani-Tür. 2020. Language model is all you need: Natural language understanding as question answering. *arXiv preprint arXiv:2011.03023*.

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*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020a. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.

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*, pages 80–90, Online. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xuan-Phi Nguyen, Shafiq Joty, Steven C. H. Hoi, and Richard Socher. 2020b. Tree-structured attention with hierarchical accumulation. *arXiv preprint arXiv:2002.08046*.

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*, pages 4885–4901, Online. Association for Computational Linguistics.

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)*.

Abiola Obamuyide and Andreas Vlachos. 2018. Zeroshot relation classification as textual entailment. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 72–78, Brussels, Belgium. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vipul Raheja, and Oleksandr Skurzhanskyi. 2021. Text Simplification by Tagging. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 11–25, Online. Association for Computational Linguistics.

Liangming Pan, Wenhu Chen, Wenhan Xiong, Min-Yen Kan, and William Yang Wang. 2021a. Zeroshot fact verification by claim generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL)*.

Liangming Pan, Wenhu Chen, Wenhan Xiong, Min-Yen Kan1, and William Yang Wang. 2021b. Unsupervised multi-hop question answering by question generation:. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*.

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)*.

Yannis Papanikolaou and Andrea Pierleoni. 2020. Dare: Data augmented relation extraction with gpt-2. *arXiv preprint arXiv:2004.13845*.

Loreto Parisi, Simone Francia, and Paolo Magnani. 2020. UmBERTo: an Italian language model trained with whole word masking. GitHub.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2018. Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Debjit Paul and Anette Frank. 2021. Coins: Dynamically generating contextualized inference rules for narrative story completion. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Baolin Peng, Chenguang Zhu, Michael Zeng, and Jianfeng Gao. 2020. Data augmentation for spoken language understanding via pretrained models.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *arXiv preprint arXiv:2105.11447*.

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)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

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 *AKBC*.

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)*.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. 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)*, pages 7654–7673, Online. Association for Computational Linguistics.

Marco Polignano, Pierpaolo Basile, Marco Degemmis, Giovanni Semeraro, and Valerio Basile. 2019. Alberto: Italian bert language understanding model for nlp challenging tasks based on tweets. In *CLiC-it*.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jakob Prange, Nathan Schneider, and Vivek Srikumar. 2021. Supertagging the long tail with tree-structured decoding of complex categories. *Transactions of the Association for Computational Linguistics*, 9(0):243–260.

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)*, pages 3208–3229, Online. Association for Computational Linguistics.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *arXiv preprint arXiv:1912.10165*.

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*, pages 5203–5212, Online. Association for Computational Linguistics.

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):1872–1897.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. *OpenAI blog*, page 12.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Evani Radiya-Dixit and Xin Wang. 2020. How fine can fine-tuning be? learning efficient language models. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2435–2443. PMLR.

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. In *Journal of Machine Learning Research*.

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

Giulio Ravasio and Leonardo Di Perna. 2020. GilBERTo: An Italian pretrained language model based on roberta. *https://github.com/idb-ita/GilBERTo*. GitHub.

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)*.

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*, New York, NY, USA. Association for Computing Machinery.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the Demonstrations Session of 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, (ACL)*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

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)*.

Alexis Ross, Ana Marasovic, and Matthew E. Peters. 2021. Explaining NLP models via minimal contrastive editing (mice). In *Findings of the Association for Computational Linguistics: (ACL/IJCNLP)*.

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)*, pages 8548–8553, Online. Association for Computational Linguistics.

Dan Roth. 2017. Incidental supervision: Moving beyond supervised learning. In *Proc. of the Conferennce on Artificial Intelligence (AAAI)*.

Rachel Rudinger, Vered Shwartz, Jena D. Hwang, Chandra Bhagavatula, Maxwell Forbes, Ronan Le Bras, Noah A. Smith, and Yejin Choi. 2020. Thinking like a skeptic: Defeasible inference in natural language. In *Findings of the Association for Computational Linguistics (EMNLP)*.

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*, pages 2647–2661, Online. Association for Computational Linguistics.

Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059, Barcelona (online). International Committee for Computational Linguistics.

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 (EMNLP)*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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*, pages 2699–2712, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

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, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.0820*.

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)*.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. *arXiv preprint arXiv:1811.00146*.

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*, pages 5569–5578, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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*, pages 3996–4007, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021a. 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*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. 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*, pages 2339–2352, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2019. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. *arXiv preprint arXiv:1904.06707*.

Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *arXiv preprint arXiv:2012.11926*.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. *arXiv preprint arXiv:2104.07540*.

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *arXiv preprint arXiv:2103.00453*. To appear in TACL.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. 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)*, pages 1599–1613, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Stefan Schweter. 2020. Berturk - bert models for turkish.

Amit Seker, Elron Bandel, Dan Bareket, Idan Brusilovsky, Refael Shaked Greenfeld, and Reut Tsarfaty. 2021. Alephbert:a hebrew large pretrained language model to start-off your hebrew nlp application with. *arXiv preprint arXiv:2104.04052*.

Siamak Shakeri, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. End-to-end synthetic data generation for domain adaptation of question answering systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

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.

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)*, pages 4222–4235, Online. Association for Computational Linguistics.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2020a. BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS*.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2020b. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*.

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*, pages 5986–5995. PMLR. ISSN: 2640-3498.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification? *arXiv preprint arXiv:1905.05583*.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Xuan Ouyang, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang.

2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

Ehsan Taher, Seyed Abbas Hoseini, and Mehrnoush Shamsfard. 2019. Beheshti-NER: Persian named entity recognition using BERT. In *Proceedings of The First International Workshop on NLP Solutions for Under Resourced Languages (NSURL 2019) colocated with ICNLSP 2019 - Short Papers*, pages 37–42, Trento, Italy. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. olmpics – on what language model pre-training captures. *arXiv preprint arXiv:1912.13283*.

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*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

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)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

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*, pages 4731–4736, Brussels, Belgium. Association for Computational Linguistics.

Trieu H. Trinh and Quoc V. Le. 2019. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. *arXiv preprint arXiv:2106.13884*.

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)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Amir Pouran Ben Veyseh, Viet Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2021a. Unleash GPT-2 power for event detection. 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)*, pages 6271–6282.

Amir Pouran Ben Veyseh, Minh Van Nguye, Bonan Min, and Thien Huu Nguyen. 2021b. Augmenting open-domain event detection with synthetic data from gpt-2. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in Neural Information Processing Systems*, 28:2692–2700.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2018. Counterfactual explanations without opening the black box: Automated decisions and the gdpr.

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)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. *https://github.com/kingoflolz/mesh-transformer-jax*.

Cunxiang Wang, Pai Liu, and Yue Zhang. 2021a. 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)*, pages 3241–3251, Online. Association for Computational Linguistics.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021b. Entailment as few-shot learner. *arXiv preprint arXiv:2104.14690*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021c. Automated concatenation of embeddings for structured prediction. *arXiv preprint arXiv:2010.05006*.

43

Xinyu Wang and Kewei Tu. 2020. Second-order neural dependency parsing with message passing and end-to-end training. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 93–99, Suzhou, China. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020a. BLiMP: A benchmark of linguistic minimal pairs for English. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 409–410, New York, New York. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. CoLA: The Corpus of Linguistic Acceptability (with added annotations). *http://nyu-mll.github.io/cola*.

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel R. Bowman. 2020b. Learning which features matter: RoBERTa acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, Online. Association for Computational Linguistics.

Albert Webson and Ellie Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? *arXiv preprint arXiv:2109.01247*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Fine-tuned Language Models Are Zero-Shot Learners. *arXiv:2109.01652 [cs]*. ArXiv: 2109.01652.

Rongxiang Weng, Heng Yu, Shujian Huang, Shanbo Cheng, and Weihua Luo. 2020. Acquiring knowledge from pre-trained model to neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9266–9273.

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)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020a. Scalable zero-shot entity linking with dense entity retrieval. *arXiv preprint arXiv:1911.03814*.

Shanchan Wu and Yifan He. 2019. Enriching pre-trained language model with entity information for relation classification. *arXiv preprint arXiv:1905.0828*.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S. Weld. 2021a. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. *arXiv preprint arXiv:2101.00288*.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020b. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, Online. Association for Computational Linguistics.

Zhaofeng Wu, Hao Peng, and Noah A. Smith. 2021b. Infusing finetuning with semantic dependencies. *Transactions of the Association of Computational Linguistics*, 9:226–242.

Dongqin Xu, Junhui Li, Muhua Zhu, Min Zhang, and Guodong Zhou. 2020. Improving AMR parsing with sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2501–2511, Online. Association for Computational Linguistics.

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)*, pages 5412–5422, Online. Association for Computational Linguistics.

Hang Yan, Junqi Dai, Tuo Ji, Xipeng Qiu, and Zheng Zhang. 2021a. 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)*.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021b. 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)*.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. *arXiv preprint arXiv:2010.14568*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. G-daug: Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics (EMNLP)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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)*, pages 1540–1550, Online. Association for Computational Linguistics.

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)*, pages 3914–3923, Hong Kong, China. Association for Computational Linguistics.

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

Jianfei Yu, Chenggong Gong, and Rui Xia. 2021a. Cross-domain review generation for aspect-based sentiment analysis. In *Findings of the Association for Computational Linguistics (ACL-IJCNLP)*.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2021b. A survey of knowledge-enhanced text generation. *arXiv preprint arXiv:2010.04389*.

Xiaodong Yu, Wenpeng Yin, and Dan Roth. 2020. Paired representation learning for event and entity coreference. *arXiv preprint arXiv:2010.12808*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *arXiv preprint arXiv:2106.11520*.

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 2020, Zurich, Switzerland, June 23-25, 2020*, volume abs/2007.03765, Zurich, Switzerland. CEUR Workshop Proceedings.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Danqing Zhang, Tao Li, Haiyang Zhang, and Bing Yin. 2020a. On data augmentation for extreme multi-label classification. *arXiv preprint arXiv:2009.10778*.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019a. Pretraining-based natural language generation for text summarization. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 789–797, Hong Kong, China. Association for Computational Linguistics.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 2020b. Side-tuning: A baseline for network adaptation via additive side networks. *arXiv preprint arXiv:1912.13503*.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. 2021a. 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 (ACL)*.

Yian Zhang, Alex Warstadt, Xiaocheng Li, and Samuel R. Bowman. 2021b. 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)*, pages 1112–1125, Online. Association for Computational Linguistics.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020c. Fast and accurate neural crf constituency parsing. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020d. Semantics-aware BERT for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020a. 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)*, pages 2226–2241, Online. Association for Computational Linguistics.

Tianyang Zhao, Zhao Yan, Yunbo Cao, and Zhoujun Li. 2020b. Asking effective and diverse questions: A machine reading comprehension based framework for joint entity-relation extraction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3948–3954. International Joint Conferences on Artificial Intelligence Organization. Main track.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

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

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021b. 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*, pages 5017–5033, Online. Association for Computational Linguistics.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020a. LIMIT-BERT : Linguistics informed multi-task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4450–4461, Online. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

Li Zhou and Kevin Small. 2020. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*.

Qiji Zhou, Yue Zhang, Donghong Ji, and Hao Tang. 2020b. AMR parsing with latent structural information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4306–4319, Online. Association for Computational Linguistics.

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*, volume 35, pages 14638–14646.

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. *arXiv preprint arXiv:2002.06823*.

A PLMs for specialized domains or languages

Table 6 shows PLMs for special domains. Table 7 presents PLMs pre-trained on different languages.

B Pre-train then fine-tune approaches

Table 8 shows a summary of prior work organized by the strategies in the first paradigm, "pretrain then fine-tune". It is worth noting that "Contextual embeddings" does not involve fine-tuning, but we included here because it is architecturally similar to other strategies, except that the PLM's weights are frozen, that is they are not fine-tuned for the specific tasks.

Model	Domain	Training Sources
SCIBERT (Beltagy et al., 2019)	Science	Scientific articles in computer science and biomedicine
BIOBERT (Lee et al., 2019)	Biomedical	Biomedical publications (abstracts and full-text articles)
CLINICALBERT (Huang et al., 2020), Alsentzer et al. (2019)	Clinical	Clinical notes
LEGALBERT (Chalkidis et al., 2020)	Legal	Legal documents (e.g. contracts)
CODEBERT (Feng et al., 2020b), CODEX (Chen et al., 2021b)	Source code	GitHub repositories
BERTweet (Nguyen et al., 2020a), Al-BERTo (for Italian; Polignano et al., 2019)	Twitter	Tweets
BabyBERTa (Huebner et al., 2021)	Child-directed speech	Child-directed speech transcriptions

Table 6: PLMs pre-trained on specific domains.

Language	Model
Arabic	Arabic-BERT (Safaya et al., 2020)
Basque	BERTeus (Agerri et al., 2020)
Chinese	MacBERT (Cui et al., 2020)
Dutch	BERTje (de Vries et al., 2019), RobBERT (Delobelle et al., 2020)
Farsi	ParsBERT (Farahani et al., 2021)
Finnish	FinBERT (Virtanen et al., 2019)
French	CamemBERT (Martin et al., 2020), FlauBERT (Le et al., 2020)
German	GBERT and GELECTRA (Chan et al., 2020)
Hebrew	HeBERT (Chriqui and Yahav, 2021), AlphaBERT (Seker et al., 2021)
Italian	GilBERTo (Ravasio and Perna, 2020), UmBERTo (Parisi et al., 2020)
Japanese	Japanese BERT (Inui Laboratory, 2021)
Portuguese	BERTimbau (Souza et al., 2020a)
Russian	RuBERT (Kuratov and Arkhipov, 2019)
Spanish	BETO (Cañete et al., 2020)
Turkish	BERTurk (Schweter, 2020)

Table 7: PLMs pre-trained on different languages.

Task	Work	PLM
(1) **Contextual Embeddings**		
Word Sense disambiguation/induction	Hadiwinoto et al. (2019), Amrami and Goldberg (2019)	BERT
Coreference resolution	(Lee et al., 2018)	ElMO
Constituency parsing	(Zhang et al., 2020c)	BERT
	Yang and Deng (2020)	XLNet, BERT
	Zhou and Zhao (2019)	ELMo, BERT
Constituency parsing, Dependency parsing	(Mrini et al., 2020)	XLNet, BERT
Dependency parsing	Wang and Tu (2020)	BERT
	Schuster et al. (2019)	ElMo
Semantic role labeling	(He et al., 2018)	ElMO
AMR parsing	Cai and Lam (2020), Xu et al. (2020)	BERT
UCCA parsing	(Jiang et al., 2019)	BERT, mBERT
Commonsense reasoning	ATOMIC (Sap et al., 2019)	ELMo
Machine Translation	Zhu et al. (2020)	BERT
Text summarization	(Zhang et al., 2019a)	BERT
(2) **Fine-tuning the PLM**		
Text classification	(Yang et al., 2019)	XLNet
	(Sun et al., 2020)	BERT
	(Peters et al., 2018)	ElMO
Semantic textual similarity	(Yang et al., 2019)	XLNet
NER	ELMo (Peters et al., 2018)	ELMo
NER, QA, Textual Entailment (TE)	Devlin et al. (2019)	BERT
TE	(Liu et al., 2019)	RoBERTa
Entity Linking	Broscheit (2019), Ling et al. (2020), (Wu et al., 2020a)	BERT
Relation extraction	(Baldini Soares et al., 2019; Wu and He, 2019; Shi and Lin, 2019)	BERT
Intent Detection and Slot Filling	(Chen et al., 2019)	BERT
	XLNet (Yang et al., 2019)	XLNet
Text generation	(Kale and Rastogi, 2020)	T5
Coreference resolution	(Joshi et al., 2019)	BERT
	(Yu et al., 2020)	RoBERTa
Text simplification	(Martin et al., 2021)	BART/mBART
	(Raffel et al., 2020)	T5
Dialogue	(Hosseini-Asl et al., 2020)	GPT-2
Semantic role labeling	(Shi and Lin, 2019)	BERT
Text summarization	(Liu and Lapata, 2019)	BERT
	(Lewis et al., 2020)	BART
Commonsense reasoning	COMET (Bosselut et al., 2019)	GPT
	ATOMIC2020 (Hwang et al., 2020)	GPT2
Machine Translation	(Liu et al., 2020c)	mBART
	(Lample and Conneau, 2019)	XLM
(3) **Fine-tuning Customized Models**		
NER, POS tagging, dependency parsing, aspect extraction	ACE (Wang et al., 2021c)	ELMo, (m)BERT, XLM-R
Semantic parsing	(Che et al., 2019)	BERT
Temporal relation extraction	(Ross et al., 2020)	BERT
Text simplification	(Omelianchuk et al., 2021)	RoBERTa
Text simplification, summarization	(Malmi et al., 2019)	BERT
Coreference resolution	CorefQA (Wu et al., 2020b)	SpanBERT
Machine Translation	(Weng et al., 2020)	BERT, GPT
	(Ma et al., 2020)	XLM-R
CCG parsing	Tree-structured supertagger (Prange et al., 2021)	RoBERTa-base
(4) **Efficient Fine-tuning Approaches**		
	BitFit (Zaken et al., 2021)	
	Adapter-Transformer (Pfeiffer et al., 2020a)	
POS tagging, dependency parsing	Trankit (Nguyen et al., 2021)	XLM-R

Table 8: A summary of prior work organized by the strategies in the first paradigm "pretrain then fine-tune".