• Invited Review •

Pre-trained Models for Natural Language Processing: A Survey

Xipeng Qiu*, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai & Xuanjing Huang

School of Computer Science, Fudan University, Shanghai 200433*, China;*
Shanghai Key Laboratory of Intelligent Information Processing, Shanghai 200433*, China*

Recently, the emergence of pre-trained models (PTMs)* has brought natural language processing (NLP) to a new era. In this survey, we provide a comprehensive review of PTMs for NLP. We first briefly introduce language representation learning and its research progress. Then we systematically categorize existing PTMs based on a taxonomy from four different perspectives. Next, we describe how to adapt the knowledge of PTMs to downstream tasks. Finally, we outline some potential directions of PTMs for future research. This survey is purposed to be a hands-on guide for understanding, using, and developing PTMs for various NLP tasks.

Deep Learning, Neural Network, Natural Language Processing, Pre-trained Model, Distributed Representation, Word Embedding, Self-Supervised Learning, Language Modelling

1 Introduction

With the development of deep learning, various neural networks have been widely used to solve Natural Language Processing (NLP) tasks, such as convolutional neural networks (CNNs) [1–3], recurrent neural networks (RNNs) [4, 5], graph-based neural networks (GNNs) [6–8] and attention mechanisms [9, 10]. One of the advantages of these neural models is their ability to alleviate the *feature engineering* problem. Non-neural NLP methods usually heavily rely on the discrete handcrafted features, while neural methods usually use low-dimensional and dense vectors (aka. *distributed representation*) to implicitly represent the syntactic or semantic features of the language. These representations are learned in specific NLP tasks. Therefore, neural methods make it easy for people to develop various NLP systems.

Despite the success of neural models for NLP tasks, the performance improvement may be less significant compared to the Computer Vision (CV) field. The main reason is that current datasets for most supervised NLP tasks are rather small (except machine translation). Deep neural networks usually have a large number of parameters, which make them overfit on these small training data and do not generalize well in practice. Therefore, the early neural models for many NLP tasks were relatively shallow and usually consisted of only 1∼3 neural layers.

Recently, substantial work has shown that pre-trained models (PTMs), on the large corpus can learn universal language representations, which are beneficial for downstream NLP tasks and can avoid training a new model from scratch. With the development of computational power, the emergence of the deep models (i.e., Transformer [10]), and the constant enhancement of training skills, the architecture of PTMs has been advanced from shallow to deep. The *first-generation PTMs* aim to learn good word embeddings. Since these models themselves are no longer needed by downstream tasks, they

* Corresponding author (email: xpqiu@fudan.edu.cn)
*PTMs are also known as pre-trained language models (PLMs). In this survey, we use PTMs for NLP instead of PLMs to avoid confusion with the narrow concept of probabilistic (or statistical) language models.

are usually very shallow for computational efficiencies, such as Skip-Gram [11] and GloVe [12]. Although these pre-trained embeddings can capture semantic meanings of words, they are context-free and fail to capture higher-level concepts in context, such as polysemous disambiguation, syntactic structures, semantic roles, anaphora. The *second-generation PTMs* focus on learning contextual word embeddings, such as CoVe [13], ELMo [14], OpenAI GPT [15] and BERT [16]. These learned encoders are still needed to represent words in context by downstream tasks. Besides, various pre-training tasks are also proposed to learn PTMs for different purposes.

The contributions of this survey can be summarized as follows:

1. *Comprehensive review.* We provide a comprehensive review of PTMs for NLP, including background knowledge, model architecture, pre-training tasks, various extensions, adaption approaches, and applications.

2. *New taxonomy.* We propose a taxonomy of PTMs for NLP, which categorizes existing PTMs from four different perspectives: 1) representation type, 2) model architecture; 3) type of pre-training task; 4) extensions for specific types of scenarios.

3. *Abundant resources.* We collect abundant resources on PTMs, including open-source implementations of PTMs, visualization tools, corpora, and paper lists.

4. *Future directions.* We discuss and analyze the limitations of existing PTMs. Also, we suggest possible future research directions.

The rest of the survey is organized as follows. Section 2 outlines the background concepts and commonly used notations of PTMs. Section 3 gives a brief overview of PTMs and clarifies the categorization of PTMs. Section 4 provides extensions of PTMs. Section 5 discusses how to transfer the knowledge of PTMs to downstream tasks. Section 6 gives the related resources on PTMs. Section 7 presents a collection of applications across various NLP tasks. Section 8 discusses the current challenges and suggests future directions. Section 9 summarizes the paper.

2 Background

2.1 Language Representation Learning

As suggested by Bengio et al. [17], a good representation should express general-purpose priors that are not task-specific but would be likely to be useful for a learning machine to solve AI-tasks. When it comes to language, a good representation should capture the implicit linguistic rules and common sense knowledge hiding in text data, such as lexical meanings, syntactic structures, semantic roles, and even pragmatics.

The core idea of distributed representation is to describe the meaning of a piece of text by low-dimensional real-valued vectors. And each dimension of the vector has no corresponding sense, while the whole represents a concrete concept. Figure 1 illustrates the generic neural architecture for NLP. There are two kinds of word embeddings: non-contextual and contextual embeddings. The difference between them is whether the embedding for a word dynamically changes according to the context it appears in.

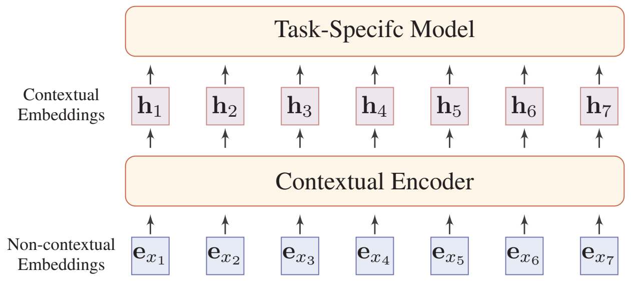

Task-Specifc Model
Contextual Embeddings
$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$ $\mathbf{h}_4$ $\mathbf{h}_5$ $\mathbf{h}_6$ $\mathbf{h}_7$
Contextual Encoder
Non-contextual Embeddings
$\mathbf{e}_{x_1}$ $\mathbf{e}_{x_2}$ $\mathbf{e}_{x_3}$ $\mathbf{e}_{x_4}$ $\mathbf{e}_{x_5}$ $\mathbf{e}_{x_6}$ $\mathbf{e}_{x_7}$

Figure 1: Generic Neural Architecture for NLP

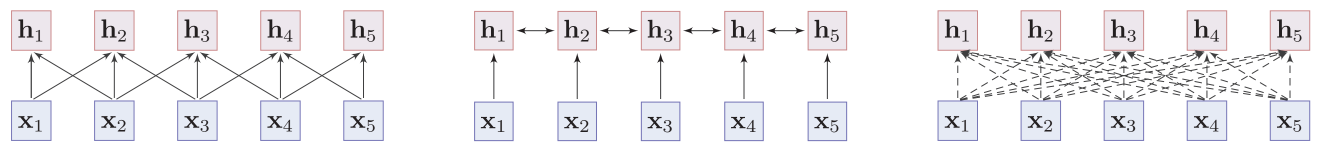
Non-contextual Embeddings The first step of representing language is to map discrete language symbols into a distributed embedding space. Formally, for each word (or sub-word) x in a vocabulary $\mathcal{V}$, we map it to a vector $\mathbf{e}_x \in \mathbb{R}^{D_e}$ with a lookup table $\mathbf{E} \in \mathbb{R}^{D_e \times |\mathcal{V}|}$, where D_e is a hyper-parameter indicating the dimension of token embeddings. These embeddings are trained on task data along with other model parameters.

There are two main limitations to this kind of embeddings. The first issue is that the embeddings are static. The embedding for a word does is always the same regardless of its context. Therefore, these *non-contextual embeddings* fail to model polysemous words. The second issue is the out-of-vocabulary problem. To tackle this problem, character-level word representations or sub-word representations are widely used in many NLP tasks, such as CharCNN [18], FastText [19] and Byte-Pair Encoding (BPE) [20].

Contextual Embeddings To address the issue of polysemous and the context-dependent nature of words, we need distinguish the semantics of words in different contexts. Given a text $x_1, x_2, \cdots, x_T$ where each token $x_t \in \mathcal{V}$ is a word or sub-word, the contextual representation of x_t depends on the whole text.

$$[\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T] = f_{\text{enc}}(x_1, x_2, \cdots, x_T), \tag{1}$$

where $f_{\text{enc}}(\cdot)$ is neural encoder, which is described in Section 2.2, $\mathbf{h}_t$ is called *contextual embedding* or *dynamical em-*


$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$ $\mathbf{h}_4$ $\mathbf{h}_5$
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\mathbf{x}_5$

(a) Convolutional Model (b) Recurrent Model (c) Fully-Connected Self-Attention Model

Figure 2: Neural Contextual Encoders

bedding of token x_t because of the contextual information included in.

2.2 Neural Contextual Encoders

Most of the neural contextual encoders can be classified into two categories: sequence models and non-sequence models. Figure 2 illustrates three representative architectures.

2.2.1 Sequence Models

Sequence models usually capture local context of a word in sequential order.

Convolutional Models Convolutional models take the embeddings of words in the input sentence and capture the meaning of a word by aggregating the local information from its neighbors by convolution operations [2].

Recurrent Models Recurrent models capture the contextual representations of words with short memory, such as LSTMs [21] and GRUs [22]. In practice, bi-directional LSTMs or GRUs are used to collect information from both sides of a word, but its performance is often affected by the long-term dependency problem.

2.2.2 Non-Sequence Models

Non-sequence models learn the contextual representation with a pre-defined tree or graph structure between words, such as the syntactic structure or semantic relation. Some popular non-sequence models include Recursive NN [6], TreeLSTM [7, 23], and GCN [24].

Although the linguistic-aware graph structure can provide useful inductive bias, how to build a good graph structure is also a challenging problem. Besides, the structure depends heavily on expert knowledge or external NLP tools, such as the dependency parser.

Fully-Connected Self-Attention Model In practice, a more straightforward way is to use a fully-connected graph to model the relation of every two words and let the model learn the structure by itself. Usually, the connection weights are dynamically computed by the self-attention mechanism, which implicitly indicates the connection between words. A successful instance of fully-connected self-attention model is the Transformer [10, 25], which also needs other supplement modules, such as positional embeddings, layer normalization, residual connections and position-wise feed-forward network (FFN) layers.

2.2.3 Analysis

Sequence models learn the contextual representation of the word with locality bias and are hard to capture the long-range interactions between words. Nevertheless, sequence models are usually easy to train and get good results for various NLP tasks.

In contrast, as an instantiated fully-connected self-attention model, the Transformer can directly model the dependency between every two words in a sequence, which is more powerful and suitable to model long range dependency of language. However, due to its heavy structure and less model bias, the Transformer usually requires a large training corpus and is easy to overfit on small or modestly-sized datasets [15, 26].

Currently, the Transformer has become the mainstream architecture of PTMs due to its powerful capacity.

2.3 Why Pre-training?

With the development of deep learning, the number of model parameters has increased rapidly. The much larger dataset is needed to fully train model parameters and prevent overfitting. However, building large-scale labeled datasets is a great challenge for most NLP tasks due to the extremely expensive annotation costs, especially for syntax and semantically related tasks.

In contrast, large-scale unlabeled corpora are relatively easy to construct. To leverage the huge unlabeled text data, we can first learn a good representation from them and then use these representations for other tasks. Recent studies have demonstrated significant performance gains on many NLP tasks with the help of the representation extracted from the PTMs on the large unannotated corpora.

The advantages of pre-training can be summarized as follows:

1. Pre-training on the huge text corpus can learn universal

language representations and help with the downstream tasks.

2. Pre-training provides a better model initialization, which usually leads to a better generalization performance and speeds up convergence on the target task.

3. Pre-training can be regarded as a kind of regularization to avoid overfitting on small data [27].

2.4 A Brief History of PTMs for NLP

Pre-training has always been an effective strategy to learn the parameters of deep neural networks, which are then fine-tuned on downstream tasks. As early as 2006, the breakthrough of deep learning came with greedy layer-wise unsupervised pre-training followed by supervised fine-tuning [28]. In CV, it has been in practice to pre-train models on the huge ImageNet corpus, and then fine-tune further on smaller data for different tasks. This is much better than a random initialization because the model learns general image features, which can then be used in various vision tasks.

In NLP, PTMs on large corpus have also been proved to be beneficial for the downstream NLP tasks, from the shallow word embedding to deep neural models.

2.4.1 First-Generation PTMs: Pre-trained Word Embeddings

Representing words as dense vectors has a long history [29]. The "modern" word embedding is introduced in pioneer work of neural network language model (NNLM) [30]. Collobert et al. [31] showed that the pre-trained word embedding on the unlabelled data could significantly improve many NLP tasks. To address the computational complexity, they learned word embeddings with *pairwise ranking* task instead of language modeling. Their work is the first attempt to obtain generic word embeddings useful for other tasks from unlabeled data. Mikolov et al. [11] showed that there is no need for deep neural networks to build good word embeddings. They propose two shallow architectures: Continuous Bag-of-Words (CBOW) and Skip-Gram (SG) models. Despite their simplicity, they can still learn high-quality word embeddings to capture the latent syntactic and semantic similarities among words. Word2vec is one of the most popular implementations of these models and makes the pre-trained word embeddings accessible for different tasks in NLP. Besides, GloVe [12] is also a widely-used model for obtaining pre-trained word embeddings, which are computed by global word-word co-occurrence statistics from a large corpus.

Although pre-trained word embeddings have been shown effective in NLP tasks, they are context-independent and mostly trained by shallow models. When used on a downstream task, the rest of the whole model still needs to be learned from scratch.

During the same time period, many researchers also try to learn embeddings of paragraph, sentence or document, such as paragraph vector [32], Skip-thought vectors [33], Context2Vec [34]. Different from their modern successors, these sentence embedding models try to encode input sentences into a fixed-dimensional vector representation, rather than the contextual representation for each token.

2.4.2 Second-Generation PTMs: Pre-trained Contextual Encoders

Since most NLP tasks are beyond word-level, it is natural to pre-train the neural encoders on sentence-level or higher. The output vectors of neural encoders are also called *contextual word embeddings* since they represent the word semantics depending on its context.

Dai and Le [35] proposed the first successful instance of PTM for NLP. They initialized LSTMs with a language model (LM) or a sequence autoencoder, and found the pre-training can improve the training and generalization of LSTMs in many text classification tasks. Liu et al. [5] pre-trained a shared LSTM encoder with LM and fine-tuned it under the multi-task learning (MTL) framework. They found the pre-training and fine-tuning can further improve the performance of MTL for several text classification tasks. Ramachandran et al. [36] found the Seq2Seq models can be significantly improved by unsupervised pre-training. The weights of both encoder and decoder are initialized with pre-trained weights of two language models and then fine-tuned with labeled data. Besides pre-training the contextual encoder with LM, McCann et al. [13] pre-trained a deep LSTM encoder from an attentional sequence-to-sequence model with machine translation (MT). The context vectors (CoVe) output by the pre-trained encoder can improve the performance of a wide variety of common NLP tasks.

Since these precursor PTMs, the modern PTMs are usually trained with larger scale corpora, more powerful or deeper architectures (e.g., Transformer), and new pre-training tasks.

Peters et al. [14] pre-trained 2-layer LSTM encoder with a bidirectional language model (BiLM), consisting of a forward LM and a backward LM. The contextual representations output by the pre-trained BiLM, ELMo (Embeddings from Language Models), are shown to bring large improvements on a broad range of NLP tasks. Akbik et al. [37] captured word meaning with contextual string embeddings pre-trained with character-level LM. However, these two PTMs are usually used as a feature extractor to produce the contextual word embeddings, which are fed into the main model for downstream tasks. Their parameters are fixed, and the rest parameters of the main model are still trained from scratch. ULMFiT (Universal Language Model Fine-tuning) [38] attempted to fine-tune pre-trained LM for text classification (TC) and achieved state-of-the-art results on six widely-used TC datasets. ULMFiT consists of 3 phases: 1) pre-training LM on general-domain data; 2) fine-tuning LM on target data; 3) fine-tuning on the target task. ULMFiT also investigates some effective fine-tuning strategies, including discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing.

More recently, the very deep PTMs have shown their powerful ability in learning universal language representations: e.g., OpenAI GPT (Generative Pre-training) [15] and BERT (Bidirectional Encoder Representation from Transformer) [16]. Besides LM, an increasing number of self-supervised tasks (see Section 3.1) is proposed to make the PTMs capturing more knowledge form large scale text corpora.

Since ULMFiT and BERT, fine-tuning has become the mainstream approach to adapt PTMs for the downstream tasks.

3 Overview of PTMs

The major differences between PTMs are the usages of contextual encoders, pre-training tasks, and purposes. We have briefly introduced the architectures of contextual encoders in Section 2.2. In this section, we focus on the description of pre-training tasks and give a taxonomy of PTMs.

3.1 Pre-training Tasks

The pre-training tasks are crucial for learning the universal representation of language. Usually, these pre-training tasks should be challenging and have substantial training data. In this section, we summarize the pre-training tasks into three categories: supervised learning, unsupervised learning, and self-supervised learning.

1. *Supervised learning* (SL) is to learn a function that maps an input to an output based on training data consisting of input-output pairs.

2. *Unsupervised learning* (UL) is to find some intrinsic knowledge from unlabeled data, such as clusters, densities, latent representations.

3. *Self-Supervised learning* (SSL) is a blend of supervised learning and unsupervised learning[1]. The learning paradigm of SSL is entirely the same as supervised learning, but the labels of training data are generated automatically. The key idea of SSL is to predict any part of the input from other parts in some form. For example, the masked language model (MLM) is a self-supervised task that attempts to predict the masked words in a sentence given the rest words.

In CV, many PTMs are trained on large supervised training sets like ImageNet. However, in NLP, the datasets of most supervised tasks are not large enough to train a good PTM. The only exception is machine translation (MT). A large-scale MT dataset, WMT 2017, consists of more than 7 million sentence pairs. Besides, MT is one of the most challenging tasks in NLP, and an encoder pre-trained on MT can benefit a variety of downstream NLP tasks. As a successful PTM, CoVe [13] is an encoder pre-trained on MT task and improves a wide variety of common NLP tasks: sentiment analysis (SST, IMDb), question classification (TREC), entailment (SNLI), and question answering (SQuAD).

In this section, we introduce some widely-used pre-training tasks in existing PTMs. We can regard these tasks as self-supervised learning. Table 1 also summarizes their loss functions.

3.1.1 Language Modeling (LM)

The most common unsupervised task in NLP is probabilistic language modeling (LM), which is a classic probabilistic density estimation problem. Although LM is a general concept, in practice, LM often refers in particular to auto-regressive LM or unidirectional LM.

Given a text sequence $\mathbf{x}_{1:T} = [x_1, x_2, \cdots, x_T]$, its joint probability $p(x_{1:T})$ can be decomposed as

$$p(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(x_t|\mathbf{x}_{0:t-1}), \tag{2}$$

where x_0 is special token indicating the begin of sequence.

The conditional probability $p(x_t|\mathbf{x}_{0:t-1})$ can be modeled by a probability distribution over the vocabulary given linguistic context $\mathbf{x}_{0:t-1}$. The context $\mathbf{x}_{0:t-1}$ is modeled by neural encoder $f_{\text{enc}}(\cdot)$, and the conditional probability is

$$p(x_t|\mathbf{x}_{0:t-1}) = g_{\text{LM}}\Big(f_{\text{enc}}(\mathbf{x}_{0:t-1})\Big), \tag{3}$$

where $g_{\text{LM}}(\cdot)$ is prediction layer.

Given a huge corpus, we can train the entire network with maximum likelihood estimation (MLE).

A drawback of unidirectional LM is that the representation of each token encodes only the leftward context tokens and itself. However, better contextual representations of text should encode contextual information from both directions.

1) Indeed, it is hard to clearly distinguish the unsupervised learning and self-supervised learning. For clarification, we refer “unsupervised learning” to the learning without human-annotated supervised labels. The purpose of “self-supervised learning” is to learn the general knowledge from data rather than standard unsupervised objectives, such as density estimation.

Table 1: Loss Functions of Pre-training Tasks

Task	Loss Function	Description
LM	$\mathcal{L}_{\text{LM}} = -\sum_{t=1}^{T} \log p(x_t \mid \mathbf{x}_{<t})$	$\mathbf{x}_{<t} = x_1, x_2, \cdots, x_{t-1}$.
MLM	$\mathcal{L}_{\text{MLM}} = -\sum_{\hat{x} \in m(\mathbf{x})} \log p\Big(\hat{x} \mid \mathbf{x}_{\setminus m(\mathbf{x})}\Big)$	$m(\mathbf{x})$ and $\mathbf{x}_{\setminus m(\mathbf{x})}$ denote the masked words from $\mathbf{x}$ and the rest words respectively.
Seq2Seq MLM	$\mathcal{L}_{\text{S2SMLM}} = -\sum_{t=i}^{j} \log p\Big(x_t \mid \mathbf{x}_{\setminus \mathbf{x}_{i:j}}, \mathbf{x}_{i:t-1}\Big)$	$\mathbf{x}_{i:j}$ denotes an masked n-gram span from i to j in $\mathbf{x}$.
PLM	$\mathcal{L}_{\text{PLM}} = -\sum_{t=1}^{T} \log p(z_t \mid \mathbf{z}_{<t})$	$\mathbf{z} = perm(\mathbf{x})$ is a permutation of $\mathbf{x}$ with random order.
DAE	$\mathcal{L}_{\text{DAE}} = -\sum_{t=1}^{T} \log p(x_t \mid \hat{\mathbf{x}}, \mathbf{x}_{<t})$	$\hat{\mathbf{x}}$ is randomly perturbed text from $\mathbf{x}$.
DIM	$\mathcal{L}_{\text{DIM}} = s(\hat{\mathbf{x}}_{i:j}, \mathbf{x}_{i:j}) - \log \sum_{\tilde{\mathbf{x}}_{i:j} \in \mathcal{N}} s(\hat{\mathbf{x}}_{i:j}, \tilde{\mathbf{x}}_{i:j})$	$\mathbf{x}_{i:j}$ denotes an n-gram span from i to j in $\mathbf{x}$, $\hat{\mathbf{x}}_{i:j}$ denotes a sentence masked at position i to j, and $\tilde{\mathbf{x}}_{i:j}$ denotes a randomly-sampled negative n-gram from corpus.
NSP/SOP	$\mathcal{L}_{\text{NSP/SOP}} = -\log p(t \mid \mathbf{x}, \mathbf{y})$	$t = 1$ if $\mathbf{x}$ and $\mathbf{y}$ are continuous segments from corpus.
RTD	$\mathcal{L}_{\text{RTD}} = -\sum_{t=1}^{T} \log p(y_t \mid \hat{\mathbf{x}})$	$y_t = \mathbf{1}(\hat{x}_t = x_t)$, $\hat{\mathbf{x}}$ is corrupted from $\mathbf{x}$.

[1] $\mathbf{x} = [x_1, x_2, \cdots, x_T]$ denotes a sequence.

An improved solution is bidirectional LM (BiLM), which consists of two unidirectional LMs: a forward left-to-right LM and a backward right-to-left LM. For BiLM, Baevski et al. [39] proposed a two-tower model that the forward tower operates the left-to-right LM and the backward tower operates the right-to-left LM.

3.1.2 Masked Language Modeling (MLM)

Masked language modeling (MLM) is first proposed by Taylor [40] in the literature, who referred to this as a Cloze task. Devlin et al. [16] adapted this task as a novel pre-training task to overcome the drawback of the standard unidirectional LM. Loosely speaking, MLM first masks out some tokens from the input sentences and then trains the model to predict the masked tokens by the rest of the tokens. However, this pre-training method will create a mismatch between the pre-training phase and the fine-tuning phase because the mask token does not appear during the fine-tuning phase. Empirically, to deal with this issue, Devlin et al. [16] used a special `[MASK]` token 80% of the time, a random token 10% of the time and the original token 10% of the time to perform masking.

Sequence-to-Sequence MLM (Seq2Seq MLM) MLM is usually solved as classification problem. We feed the masked sequences to a neural encoder whose output vectors are further fed into a softmax classifier to predict the masked token. Alternatively, we can use encoder-decoder (aka. sequence-to-sequence) architecture for MLM, in which the encoder is fed a masked sequence, and the decoder sequentially produces the masked tokens in auto-regression fashion. We refer to this kind of MLM as sequence-to-sequence MLM (Seq2Seq MLM), which is used in MASS [41] and T5 [42]. Seq2Seq MLM can benefit the Seq2Seq-style downstream tasks, such as question answering, summarization, and machine translation.

Enhanced Masked Language Modeling (E-MLM) Concurrently, there are multiple research proposing different enhanced versions of MLM to further improve on BERT. Instead of static masking, RoBERTa [43] improves BERT by dynamic masking.

UniLM [44, 45] extends the task of mask prediction on three types of language modeling tasks: unidirectional, bidirectional, and sequence-to-sequence prediction. XLM [46] performs MLM on a concatenation of parallel bilingual sentence pairs, called *Translation Language Modeling* (TLM). SpanBERT [47] replaces MLM with *Random Contiguous Words Masking* and *Span Boundary Objective* (SBO) to integrate structure information into pre-training, which requires the system to predict masked spans based on span boundaries. Besides, StructBERT [48] introduces the *Span Order Recovery* task to further incorporate language structures.

Another way to enrich MLM is to incorporate external knowledge (see Section 4.1).

3.1.3 Permuted Language Modeling (PLM)

Despite the wide use of the MLM task in pre-training, Yang et al. [49] claimed that some special tokens used in the pre-training of MLM, like `[MASK]`, are absent when the model is applied on downstream tasks, leading to a gap between pre-training and fine-tuning. To overcome this issue, Permuted Language Modeling (PLM) [49] is a pre-training objective to replace MLM. In short, PLM is a language modeling task on a random permutation of input sequences. A permutation is randomly sampled from all possible permutations. Then some of the tokens in the permuted sequence are chosen as the target, and the model is trained to predict these targets, depending on the rest of the tokens and the natural positions of targets. Note that this permutation does not affect the natural positions of sequences and only defines the order of token predictions. In practice, only the last few tokens in the permuted sequences are predicted, due to the slow convergence. And a special two-stream self-attention is introduced for target-aware representations.

3.1.4 Denoising Autoencoder (DAE)

Denoising autoencoder (DAE) takes a partially corrupted input and aims to recover the original undistorted input. Specific to language, a sequence-to-sequence model, such as the standard Transformer, is used to reconstruct the original text. There are several ways to corrupt text [50]:

(1) *Token Masking:* Randomly sampling tokens from the input and replacing them with `[MASK]` elements.

(2) *Token Deletion:* Randomly deleting tokens from the input. Different from token masking, the model needs to decide the positions of missing inputs.

(3) *Text Infilling:* Like SpanBERT, a number of text spans are sampled and replaced with a single `[MASK]` token. Each span length is drawn from a Poisson distribution ($\lambda = 3$). The model needs to predict how many tokens are missing from a span.

(4) *Sentence Permutation:* Dividing a document into sentences based on full stops and shuffling these sentences in random order.

(5) *Document Rotation:* Selecting a token uniformly at random and rotating the document so that it begins with that token. The model needs to identify the real start position of the document.

3.1.5 Contrastive Learning (CTL)

Contrastive learning [51] assumes some observed pairs of text that are more semantically similar than randomly sampled text. A score function $s(x, y)$ for text pair (x, y) is learned to minimize the objective function:

$$\mathcal{L}_{\text{CTL}} = \mathbb{E}_{x,y^+,y^-}\Big[-\log \frac{\exp\big(s(x, y^+)\big)}{\exp\big(s(x, y^+)\big) + \exp\big(s(x, y^-)\big)}\Big], \tag{4}$$

where (x, y^+) are a similar pair and y^- is presumably dissimilar to x. y^+ and y^- are typically called positive and negative sample. The score function $s(x, y)$ is often computed by a learnable neural encoder in two ways: $s(x, y) = f^{\mathrm{T}}_{\text{enc}(x)} f_{\text{enc}(y)}$ or $s(x, y) = f_{\text{enc}}(x \oplus y)$.

The idea behind CTL is "learning by comparison". Compared to LM, CTL usually has less computational complexity and therefore is desirable alternative training criteria for PTMs.

Collobert et al. [31] proposed *pairwise ranking* task to distinguish real and fake phrases. The model needs to predict a higher score for a legal phrase than an incorrect phrase obtained by replacing its central word with a random word. Mnih and Kavukcuoglu [52] trained word embeddings efficiently with Noise-Contrastive Estimation (NCE) [53], which trains a binary classifier to distinguish real and fake samples. The idea of NCE is also used in the well-known word2vec embedding [11].

We briefly describe some recently proposed CTL tasks in the following paragraphs.

Deep InfoMax (DIM) Deep InfoMax (DIM) [54] is originally proposed for images, which improves the quality of the representation by maximizing the mutual information between an image representation and local regions of the image.

Kong et al. [55] applied DIM to language representation learning. The global representation of a sequence x is defined to be the hidden state of the first token (assumed to be a special start of sentence symbol) output by contextual encoder $f_{\text{enc}}(\mathbf{x})$. The objective of DIM is to assign a higher score for $f_{\text{enc}}(\mathbf{x}_{i:j})^{\mathrm{T}} f_{\text{enc}}(\hat{\mathbf{x}}_{i:j})$ than $f_{\text{enc}}(\tilde{\mathbf{x}}_{i:j})^{\mathrm{T}} f_{\text{enc}}(\hat{\mathbf{x}}_{i:j})$, where $\mathbf{x}_{i:j}$ denotes an n-gram[2)] span from i to j in $\mathbf{x}$, $\hat{\mathbf{x}}_{i:j}$ denotes a sentence masked at position i to j, and $\tilde{\mathbf{x}}_{i:j}$ denotes a randomly-sampled negative n-gram from corpus.

Replaced Token Detection (RTD) Replaced Token Detection (RTD) is the same as NCE but predicts whether a token is replaced given its surrounding context.

CBOW with negative sampling (CBOW-NS) [11] can be viewed as a simple version of RTD, in which the negative samples are randomly sampled from vocabulary with simple proposal distribution.

ELECTRA [56] improves RTD by utilizing a generator to replacing some tokens of a sequence. A generator G and a discriminator D are trained following a two-stage procedure: (1) Train only the generator with MLM task for n_1 steps; (2) Initialize the weights of the discriminator with the weights of the generator. Then train the discriminator with a discriminative task for n_2 steps, keeping G frozen. Here the discriminative task indicates justifying whether the input token has been replaced by G or not. The generator is thrown after pre-training, and only the discriminator will be fine-tuned on downstream tasks.

2) n is drawn from a Gaussian distribution $\mathcal{N}(5, 1)$ clipped at 1 (minimum length) and 10 (maximum length).

RTD is also an alternative solution for the mismatch problem. The network sees `[MASK]` during pre-training but not when being fine-tuned in downstream tasks.

Similarly, WKLM [57] replaces words on the entity-level instead of token-level. Concretely, WKLM replaces entity mentions with names of other entities of the same type and train the models to distinguish whether the entity has been replaced.

Next Sentence Prediction (NSP) Punctuations are the natural separators of text data. So, it is reasonable to construct pre-training methods by utilizing them. Next Sentence Prediction (NSP) [16] is just a great example of this. As its name suggests, NSP trains the model to distinguish whether two input sentences are continuous segments from the training corpus. Specifically, when choosing the sentences pair for each pre-training example, 50% of the time, the second sentence is the actual next sentence of the first one, and 50% of the time, it is a random sentence from the corpus. By doing so, it is capable to teach the model to understand the relationship between two input sentences and thus benefit downstream tasks that are sensitive to this information, such as Question Answering and Natural Language Inference.

However, the necessity of the NSP task has been questioned by subsequent work [47, 49, 43, 63]. Yang et al. [49] found the impact of the NSP task unreliable, while Joshi et al. [47] found that single-sentence training without the NSP loss is superior to sentence-pair training with the NSP loss. Moreover, Liu et al. [43] conducted a further analysis for the NSP task, which shows that when training with blocks of text from a single document, removing the NSP loss matches or slightly improves performance on downstream tasks.

Sentence Order Prediction (SOP) To better model inter-sentence coherence, ALBERT [63] replaces the NSP loss with a sentence order prediction (SOP) loss. As conjectured in Lan et al. [63], NSP conflates topic prediction and coherence prediction in a single task. Thus, the model is allowed to make predictions merely rely on the easier task, topic prediction. Different from NSP, SOP uses two consecutive segments from the same document as positive examples, and the same two consecutive segments but with their order swapped as negative examples. As a result, ALBERT consistently outperforms BERT on various downstream tasks.

StructBERT [48] and BERTje [88] also take SOP as their self-supervised learning task.

3.1.6 Others

Apart from the above tasks, there are many other auxiliary pre-training tasks designated to incorporate factual knowledge (see Section 4.1), improve cross-lingual tasks (see Section 4.2), multi-modal applications (see Section 4.3), or other specific tasks (see Section 4.4).

3.2 Taxonomy of PTMs

To clarify the relations of existing PTMs for NLP, we build the taxonomy of PTMs, which categorizes existing PTMs from four different perspectives:

1. *Representation Type*: According to the representation used for downstream tasks, we can divide PTMs into non-contextual and contextual models.

2. *Architectures*: The backbone network used by PTMs, including LSTM, Transformer encoder, Transformer decoder, and the full Transformer architecture. "Transformer" means the standard encoder-decoder architecture. "Transformer encoder" and "Transformer decoder" mean the encoder and decoder part of the standard Transformer architecture, respectively. Their difference is that the decoder part uses masked self-attention with a triangular matrix to prevent tokens from attending their future (right) positions.

3. *Pre-Training Task Types*: The type of pre-training tasks used by PTMs. We have discussed them in Section 3.1.

4. *Extensions*: PTMs designed for various scenarios, including knowledge-enriched PTMs, multilingual or language-specific PTMs, multi-model PTMs, domain-specific PTMs and compressed PTMs. We will particularly introduce these extensions in Section 4.

Figure 3 shows the taxonomy as well as some corresponding representative PTMs. Besides, Table 2 distinguishes some representative PTMs in more detail.

3.3 Model Analysis

Due to the great success of PTMs, it is important to understand what kinds of knowledge are captured by them, and how to induce knowledge from them. There is a wide range of literature analyzing linguistic knowledge and world knowledge stored in pre-trained non-contextual and contextual embeddings.

3.3.1 Non-Contextual Embeddings

Static word embeddings are first probed for kinds of knowledge. Mikolov et al. [117] found that word representations learned by neural network language models are able to capture linguistic regularities in language, and the relationship between words can be characterized by a relation-specific vector offset. Further analogy experiments [11] demonstrated that word vectors produced by skip-gram model

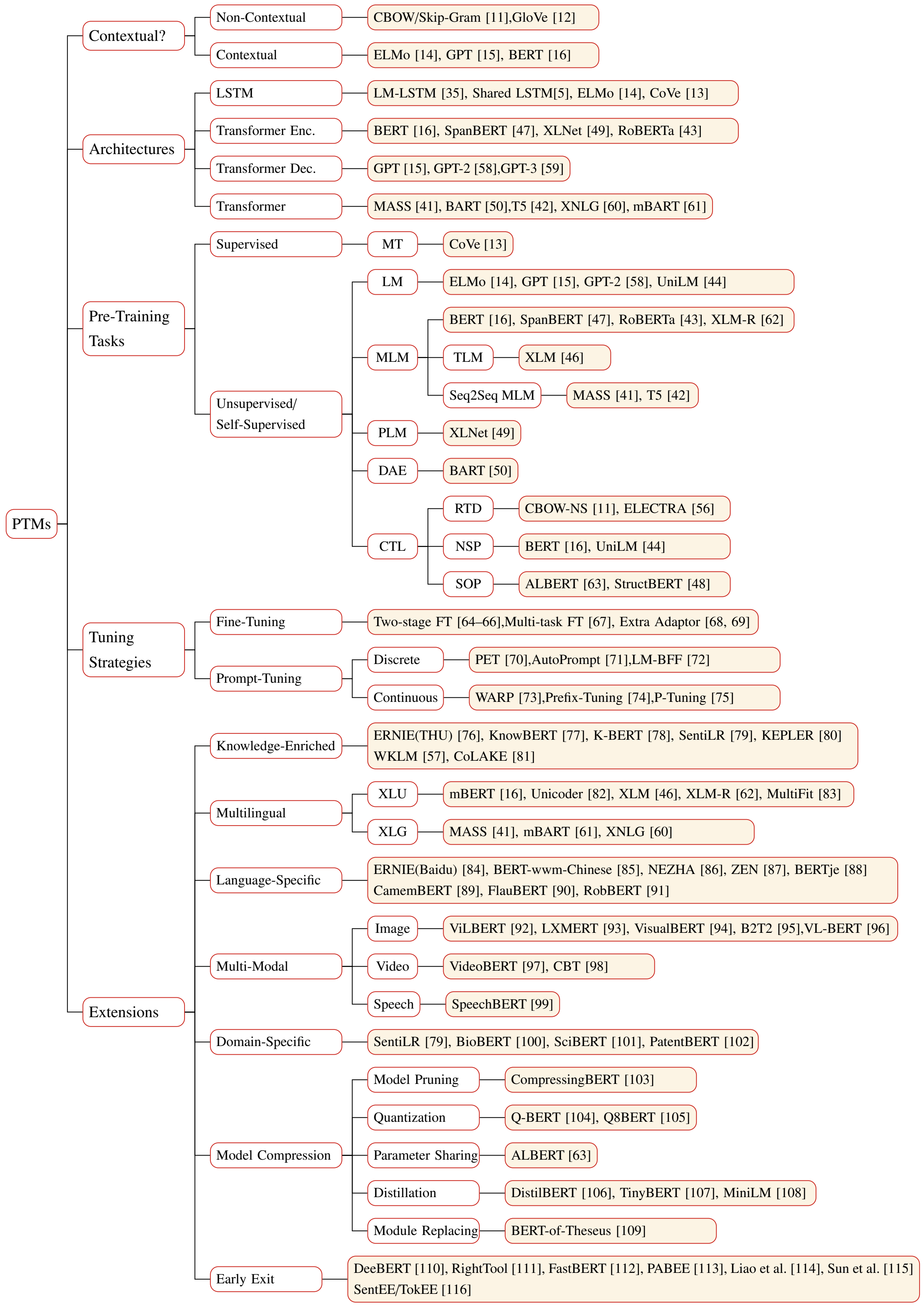

PTMs
Contextual?
Non-Contextual
CBOW/Skip-Gram [11],GloVe [12]
Contextual
ELMo [14], GPT [15], BERT [16]
Architectures
LSTM
LM-LSTM [35], Shared LSTM[5], ELMo [14], CoVe [13]
Transformer Enc.
BERT [16], SpanBERT [47], XLNet [49], RoBERTa [43]
Transformer Dec.
GPT [15], GPT-2 [58],GPT-3 [59]
Transformer
MASS [41], BART [50],T5 [42], XNLG [60], mBART [61]
Pre-Training Tasks
Supervised
MT
CoVe [13]
Unsupervised/ Self-Supervised
LM
ELMo [14], GPT [15], GPT-2 [58], UniLM [44]
MLM
BERT [16], SpanBERT [47], RoBERTa [43], XLM-R [62]
TLM
XLM [46]
Seq2Seq MLM
MASS [41], T5 [42]
PLM
XLNet [49]
DAE
BART [50]
CTL
RTD
CBOW-NS [11], ELECTRA [56]
NSP
BERT [16], UniLM [44]
SOP
ALBERT [63], StructBERT [48]
Tuning Strategies
Fine-Tuning
Two-stage FT [64–66],Multi-task FT [67], Extra Adaptor [68, 69]
Prompt-Tuning
Discrete
PET [70],AutoPrompt [71],LM-BFF [72]
Continuous
WARP [73],Prefix-Tuning [74],P-Tuning [75]
Extensions
Knowledge-Enriched
ERNIE(THU) [76], KnowBERT [77], K-BERT [78], SentiLR [79], KEPLER [80] WKLM [57], CoLAKE [81]
Multilingual
XLU
mBERT [16], Unicoder [82], XLM [46], XLM-R [62], MultiFit [83]
XLG
MASS [41], mBART [61], XNLG [60]
Language-Specific
ERNIE(Baidu) [84], BERT-wwm-Chinese [85], NEZHA [86], ZEN [87], BERTje [88] CamemBERT [89], FlauBERT [90], RobBERT [91]
Multi-Modal
Image
ViLBERT [92], LXMERT [93], VisualBERT [94], B2T2 [95],VL-BERT [96]
Video
VideoBERT [97], CBT [98]
Speech
SpeechBERT [99]
Domain-Specific
SentiLR [79], BioBERT [100], SciBERT [101], PatentBERT [102]
Model Compression
Model Pruning
CompressingBERT [103]
Quantization
Q-BERT [104], Q8BERT [105]
Parameter Sharing
ALBERT [63]
Distillation
DistilBERT [106], TinyBERT [107], MiniLM [108]
Module Replacing
BERT-of-Theseus [109]
Early Exit
DeeBERT [110], RightTool [111], FastBERT [112], PABEE [113], Liao et al. [114], Sun et al. [115] SentEE/TokEE [116]

Figure 3: Taxonomy of PTMs with Representative Examples

Table 2: List of Representative PTMs

PTMs	Architecture[†]	Input	Pre-Training Task	Corpus	Params	GLUE[‡]	FT?[♯]
ELMo [14]	LSTM	Text	BiLM	WikiText-103			No
GPT [15]	Transformer Dec.	Text	LM	BookCorpus	117M	72.8	Yes
GPT-2 [58]	Transformer Dec.	Text	LM	WebText	117M ~ 1542M		No
BERT [16]	Transformer Enc.	Text	MLM & NSP	WikiEn+BookCorpus	110M ~ 340M	81.9[*]	Yes
InfoWord [55]	Transformer Enc.	Text	DIM+MLM	WikiEn+BookCorpus	=BERT	81.1[*]	Yes
RoBERTa [43]	Transformer Enc.	Text	MLM	BookCorpus+CC-News+OpenWebText+ STORIES	355M	88.5	Yes
XLNet [49]	Two-Stream Transformer Enc.	Text	PLM	WikiEn+ BookCorpus+Giga5 +ClueWeb+Common Crawl	≈BERT	90.5[§]	Yes
ELECTRA [56]	Transformer Enc.	Text	RTD+MLM	same to XLNet	335M	88.6	Yes
UniLM [44]	Transformer Enc.	Text	MLM[◇]+ NSP	WikiEn+BookCorpus	340M	80.8	Yes
MASS [41]	Transformer	Text	Seq2Seq MLM	*Task-dependent			Yes
BART [50]	Transformer	Text	DAE	same to RoBERTa	110% of BERT	88.4[*]	Yes
T5 [42]	Transformer	Text	Seq2Seq MLM	Colossal Clean Crawled Corpus (C4)	220M ~ 11B	89.7[*]	Yes
ERNIE(THU) [76]	Transformer Enc.	Text+Entities	MLM+NSP+dEA	WikiEn + Wikidata	114M	79.6	Yes
KnowBERT [77]	Transformer Enc.	Text	MLM+NSP+EL	WikiEn + WordNet/Wiki	253M ~ 523M		Yes
K-BERT [78]	Transformer Enc.	Text+Triples	MLM+NSP	WikiZh + WebtextZh + CN-DBpedia + HowNet + MedicalKG	=BERT		Yes
KEPLER [80]	Transformer Enc.	Text	MLM+KE	WikiEn + Wikidata/WordNet			Yes
WKLM [57]	Transformer Enc.	Text	MLM+ERD	WikiEn + Wikidata	=BERT		Yes
CoLAKE [81]	Transformer Enc.	Text+Triples	MLM	WikiEn + Wikidata	=RoBERTa	86.3	Yes

[†] "Transformer Enc." and "Transformer Dec." mean the encoder and decoder part of the standard Transformer architecture respectively. Their difference is that the decoder part uses masked self-attention with triangular matrix to prevent tokens from attending their future (right) positions. "Transformer" means the standard encoder-decoder architecture.
[‡] the averaged score on 9 tasks of GLUE benchmark (see Section 7.1).
[*] without WNLI task.
[§] indicates ensemble result.
[♯] means whether is model usually used in fine-tuning fashion.
[◇] The MLM of UniLM is built on three versions of LMs: Unidirectional LM, Bidirectional LM, and Sequence-to-Sequence LM.

can capture both syntactic and semantic word relationships, such as vec("China") − vec("Beijing") ≈ vec("Japan") − vec("Tokyo"). Besides, they find compositionality property of word vectors, for example, vec("Germany") + vec("capital") is close to vec("Berlin"). Inspired by these work, Rubinstein et al. [118] found that distributional word representations are good at predicting taxonomic properties (e.g., dog is an animal) but fail to learn attributive properties (e.g., swan is white). Similarly, Gupta et al. [119] showed that word2vec embeddings implicitly encode referential attributes of entities. The distributed word vectors, along with a simple supervised model, can learn to predict numeric and binary attributes of entities with a reasonable degree of accuracy.

3.3.2 Contextual Embeddings

A large number of studies have probed and induced different types of knowledge in contextual embeddings. In general, there are two types of knowledge: linguistic knowledge and world knowledge.

Linguistic Knowledge A wide range of probing tasks are designed to investigate the linguistic knowledge in PTMs. Tenney et al. [120], Liu et al. [121] found that BERT performs well on many syntactic tasks such as part-of-speech tagging and constituent labeling. However, BERT is not good enough at semantic and fine-grained syntactic tasks, compared with simple syntactic tasks.

Besides, Tenney et al. [122] analyzed the roles of BERT's layers in different tasks and found that BERT solves tasks in a similar order to that in NLP pipelines. Furthermore, knowledge of subject-verb agreement [123] and semantic roles [124] are also confirmed to exist in BERT. Besides, Hewitt and Manning [125], Jawahar et al. [126], Kim et al. [127] proposed several methods to extract dependency trees and constituency trees from BERT, which proved the BERT's ability to encode syntax structure. Reif et al. [128] explored the geometry of internal representations in BERT and find some evidence: 1) linguistic features seem to be represented in separate semantic and syntactic subspaces; 2) attention matrices contain grammatical representations; 3) BERT distinguishes word senses at a very fine level.

World Knowledge Besides linguistic knowledge, PTMs may also store world knowledge presented in the training data. A straightforward method of probing world knowledge is to query BERT with "fill-in-the-blank" cloze statements, for example, "Dante was born in `[MASK]`". Petroni et al. [129] constructed LAMA (Language Model Analysis) task by manually creating single-token cloze statements (queries) from several knowledge sources. Their experiments show that BERT contains world knowledge competitive with traditional information extraction methods. Since the simplicity of query generation procedure in LAMA, Jiang et al. [130] argued that LAMA just measures a lower bound for what language models know and propose more advanced methods to generate more efficient queries. Despite the surprising findings of LAMA, it has also been questioned by subsequent work [131, 132]. Similarly, several studies induce relational knowledge [133] and commonsense knowledge [134] from BERT for downstream tasks.

4 Extensions of PTMs

4.1 Knowledge-Enriched PTMs

PTMs usually learn universal language representation from general-purpose large-scale text corpora but lack domain-specific knowledge. Incorporating domain knowledge from external knowledge bases into PTM has been shown to be effective. The external knowledge ranges from linguistic [135, 79, 77, 136], semantic [137], commonsense [138], factual [76–78, 57, 80], to domain-specific knowledge [139, 78].

On the one hand, external knowledge can be injected during pre-training. Early studies [140–143] focused on learning knowledge graph embeddings and word embedding jointly. Since BERT, some auxiliary pre-training tasks are designed to incorporate external knowledge into deep PTMs. LIBERT [135] (linguistically-informed BERT) incorporates linguistic knowledge via an additional linguistic constraint task. Ke et al. [79] integrated sentiment polarity of each word to extend the MLM to Label-Aware MLM (LA-MLM). As a result, their proposed model, SentiLR, achieves state-of-the-art performance on several sentence- and aspect-level sentiment classification tasks. Levine et al. [137] proposed SenseBERT, which is pre-trained to predict not only the masked tokens but also their supersenses in WordNet. ERNIE(THU) [76] integrates entity embeddings pre-trained on a knowledge graph with corresponding entity mentions in the text to enhance the text representation. Similarly, KnowBERT [77] trains BERT jointly with an entity linking model to incorporate entity representation in an end-to-end fashion. Wang et al. [80] proposed KEPLER, which jointly optimizes knowledge embedding and language modeling objectives. These work inject structure information of knowledge graph via entity embedding. In contrast, K-BERT [78] explicitly injects related triples extracted from KG into the sentence to obtain an extended tree-form input for BERT. CoLAKE [81] integrates knowledge context and language context into a unified graph, which is then pre-trained with MLM to obtain contextualized representation for both knowledge and language. Moreover, Xiong et al. [57] adopted entity replacement identification to encourage the model to be more aware of factual knowledge. However, most of these methods update the parameters of PTMs when injecting knowledge, which may suffer from catastrophic forgetting when injecting multiple kinds of knowledge. To address this, K-Adapter [136] injects multiple kinds of knowledge by training different adapters independently for different pre-training tasks, which allows continual knowledge infusion.

On the other hand, one can incorporate external knowledge into pre-trained models without retraining them from scratch. As an example, K-BERT [78] allows injecting factual knowledge during fine-tuning on downstream tasks. Guan et al. [138] employed commonsense knowledge bases, ConceptNet and ATOMIC, to enhance GPT-2 for story generation. Yang et al. [144] proposed a knowledge-text fusion model to acquire related linguistic and factual knowledge for machine reading comprehension.

Besides, Logan IV et al. [145] and Hayashi et al. [146] extended language model to *knowledge graph language model* (KGLM) and *latent relation language model* (LRLM) respectively, both of which allow prediction conditioned on knowledge graph. These novel KG-conditioned language models show potential for pre-training.

4.2 Multilingual and Language-Specific PTMs

4.2.1 Multilingual PTMs

Learning multilingual text representations shared across languages plays an important role in many cross-lingual NLP tasks.

Cross-Lingual Language Understanding (XLU) Most of the early works focus on learning multilingual word embedding [147–149], which represents text from multiple languages in a single semantic space. However, these methods usually need (weak) alignment between languages.

Multilingual BERT[3] (mBERT) is pre-trained by MLM with the shared vocabulary and weights on Wikipedia text from the top 104 languages. Each training sample is a monolingual document, and there are no cross-lingual objectives specifically designed nor any cross-lingual data. Even so, mBERT performs cross-lingual generalization surprisingly well [150]. K et al. [151] showed that the lexical overlap between languages plays a negligible role in cross-lingual success.

XLM [46] improves mBERT by incorporating a cross-lingual task, translation language modeling (TLM), which performs MLM on a concatenation of parallel bilingual sentence pairs. Unicoder [82] further propose three new cross-lingual pre-training tasks, including cross-lingual word recovery, cross-lingual paraphrase classification and cross-lingual

3) https://github.com/google-research/bert/blob/master/multilingual.md

masked language model (XMLM).

XLM-RoBERTa (XLM-R) [62] is a scaled multilingual encoder pre-trained on a significantly increased amount of training data, 2.5TB clean CommonCrawl data in 100 different languages. The pre-training task of XLM-RoBERTa is monolingual MLM only. XLM-R achieves state-of-the-arts results on multiple cross-lingual benchmarks, including XNLI, MLQA, and NER.

Cross-Lingual Language Generation (XLG) Multilingual generation is a kind of tasks to generate text with different languages from the input language, such as machine translation and cross-lingual abstractive summarization.

Different from the PTMs for multilingual classification, the PTMs for multilingual generation usually needs to pre-train both the encoder and decoder jointly, rather than only focusing on the encoder.

MASS [41] pre-trains a Seq2Seq model with monolingual Seq2Seq MLM on multiple languages and achieves significant improvement for unsupervised NMT. XNLG [60] performs two-stage pre-training for cross-lingual natural language generation. The first stage pre-trains the encoder with monolingual MLM and Cross-Lingual MLM (XMLM) tasks. The second stage pre-trains the decoder by using monolingual DAE and Cross-Lingual Auto-Encoding (XAE) tasks while keeping the encoder fixed. Experiments show the benefit of XNLG on cross-lingual question generation and cross-lingual abstractive summarization. mBART [61], a multilingual extension of BART [50], pre-trains the encoder and decoder jointly with Seq2Seq denoising auto-encoder (DAE) task on large-scale monolingual corpora across 25 languages. Experiments demonstrate that mBART produces significant performance gains across a wide variety of machine translation (MT) tasks.

4.2.2 Language-Specific PTMs

Although multilingual PTMs perform well on many languages, recent work showed that PTMs trained on a single language significantly outperform the multilingual results [89, 90, 152].

For Chinese, which does not have explicit word boundaries, modeling larger granularity [85, 87, 86] and multi-granularity [84, 153] word representations have shown great success. Kuratov and Arkhipov [154] used transfer learning techniques to adapt a multilingual PTM to a monolingual PTM for Russian language. In addition, some monolingual PTMs have been released for different languages, such as CamemBERT [89] and FlauBERT [90] for French, FinBERT [152] for Finnish, BERTje [88] and RobBERT [91] for Dutch, AraBERT [155] for Arabic language.

4.3 Multi-Modal PTMs

Observing the success of PTMs across many NLP tasks, some research has focused on obtaining a cross-modal version of PTMs. A great majority of these models are designed for a general visual and linguistic feature encoding. And these models are pre-trained on some huge corpus of cross-modal data, such as videos with spoken words or images with captions, incorporating extended pre-training tasks to fully utilize the multi-modal feature. Typically, tasks like *visual-based MLM*, *masked visual-feature modeling* and *visual-linguistic matching* are widely used in multi-modal pre-training, such as VideoBERT [97], VisualBERT [94], ViLBERT [92].

4.3.1 Video-Text PTMs

VideoBERT [97] and CBT [98] are joint video and text models. To obtain sequences of visual and linguistic tokens used for pre-training, the videos are pre-processed by CNN-based encoders and off-the-shelf speech recognition techniques, respectively. And a single Transformer encoder is trained on the processed data to learn the vision-language representations for downstream tasks like video caption. Furthermore, UniViLM [156] proposes to bring in generation tasks to further pre-train the decoder using in downstream tasks.

4.3.2 Image-Text PTMs

Besides methods for video-language pre-training, several works introduce PTMs on image-text pairs, aiming to fit downstream tasks like visual question answering(VQA) and visual commonsense reasoning(VCR). Several proposed models adopt two separate encoders for image and text representation independently, such as ViLBERT [92] and LXMERT [93]. While other methods like VisualBERT [94], B2T2 [95], VL-BERT [96], Unicoder-VL [157] and UNITER [158] propose single-stream unified Transformer. Though these model architectures are different, similar pre-training tasks, such as MLM and image-text matching, are introduced in these approaches. And to better exploit visual elements, images are converted into sequences of regions by applying RoI or bounding box retrieval techniques before encoded by pre-trained Transformers.

4.3.3 Audio-Text PTMs

Moreover, several methods have explored the chance of PTMs on audio-text pairs, such as SpeechBERT [99]. This work tries to build an end-to-end Speech Question Answering (SQA) model by encoding audio and text with a single Transformer encoder, which is pre-trained with MLM on speech and text corpus and fine-tuned on Question Answering.

4.4 Domain-Specific and Task-Specific PTMs

Most publicly available PTMs are trained on general domain corpora such as Wikipedia, which limits their applications to specific domains or tasks. Recently, some studies have proposed PTMs trained on specialty corpora, such as BioBERT [100] for biomedical text, SciBERT [101] for scientific text, ClinicalBERT [159, 160] for clinical text.

In addition to pre-training a domain-specific PTM, some work attempts to adapt available pre-trained models to target applications, such as biomedical entity normalization [161], patent classification [102], progress notes classification and keyword extraction [162].

Some task-oriented pre-training tasks were also proposed, such as sentiment *Label-Aware MLM* in SentiLR [79] for sentiment analysis, *Gap Sentence Generation* (GSG) [163] for text summarization, and *Noisy Words Detection* for disfluency detection [164].

4.5 Model Compression

Since PTMs usually consist of at least hundreds of millions of parameters, they are difficult to be deployed on the on-line service in real-life applications and on resource-restricted devices. Model compression [165] is a potential approach to reduce the model size and increase computation efficiency.

There are five ways to compress PTMs [166]: (1) *model pruning*, which removes less important parameters, (2) *weight quantization* [167], which uses fewer bits to represent the parameters, (3) *parameter sharing* across similar model units, (4) *knowledge distillation* [168], which trains a smaller student model that learns from intermediate outputs from the original model and (5) *module replacing*, which replaces the modules of original PTMs with more compact substitutes.

Table 3 gives a comparison of some representative compressed PTMs.

4.5.1 Model Pruning

Model pruning refers to removing part of neural network (e.g., weights, neurons, layers, channels, attention heads), thereby achieving the effects of reducing the model size and speeding up inference time.

Gordon et al. [103] explored the timing of pruning (e.g., pruning during pre-training, after downstream fine-tuning) and the pruning regimes. Michel et al. [174] and Voita et al. [175] tried to prune the entire self-attention heads in the transformer block.

4.5.2 Quantization

Quantization refers to the compression of higher precision parameters to lower precision. Works from Shen et al. [104] and Zafrir et al. [105] solely focus on this area. Note that quantization often requires compatible hardware.

4.5.3 Parameter Sharing

Another well-known approach to reduce the number of parameters is parameter sharing, which is widely used in CNNs, RNNs, and Transformer [176]. ALBERT [63] uses *cross-layer parameter sharing* and *factorized embedding parameterization* to reduce the parameters of PTMs. Although the number of parameters is greatly reduced, the training and inference time of ALBERT are even longer than the standard BERT.

Generally, parameter sharing does not improve the computational efficiency at inference phase.

4.5.4 Knowledge Distillation

Knowledge distillation (KD) [168] is a compression technique in which a small model called *student model* is trained to reproduce the behaviors of a large model called *teacher model*. Here the teacher model can be an ensemble of many models and usually well pre-trained. Different to model compression, distillation techniques learn a small student model from a fixed teacher model through some optimization objectives, while compression techniques aiming at searching a sparser architecture.

Generally, distillation mechanisms can be divided into three types: (1) distillation from soft target probabilities, (2) distillation from other knowledge, and (3) distillation to other structures:

(1) *Distillation from soft target probabilities*. Bucilua et al. [165] showed that making the student approximate the teacher model can transfer knowledge from teacher to student. A common method is approximating the logits of the teacher model. DistilBERT [106] trained the student model with a distillation loss over the soft target probabilities of the teacher as:

$$\mathcal{L}_{\text{KD-CE}} = \sum_i t_i \cdot \log(s_i), \tag{5}$$

where t_i and s_i are the probabilities estimated by the teacher model and the student, respectively.

Distillation from soft target probabilities can also be used in task-specific models, such as information retrieval [177], and sequence labeling [178].

(2) *Distillation from other knowledge*. Distillation from soft target probabilities regards the teacher model as a black box and only focus on its outputs. Moreover, decomposing

Table 3: Comparison of Compressed PTMs

Method	Type	#Layer	Loss Function*	Speed Up	Params	Source PTM	GLUE‡
BERT$_\text{BASE}$ [16]	Baseline	12	$\mathcal{L}_\text{MLM} + \mathcal{L}_\text{NSP}$		110M		79.6
BERT$_\text{LARGE}$ [16]		24	$\mathcal{L}_\text{MLM} + \mathcal{L}_\text{NSP}$		340M		81.9
Q-BERT [104]	Quantization	12	HAWQ + GWQ	-		BERT$_\text{BASE}$	$\approx$ 99% BERT$^\diamond$
Q8BERT [105]		12	DQ + QAT	-		BERT$_\text{BASE}$	$\approx$ 99% BERT
ALBERT§ [63]	Param. Sharing	12	$\mathcal{L}_\text{MLM} + \mathcal{L}_\text{SOP}$	$\times$5.6 ~ 0.3	12 ~ 235M		89.4 (ensemble)
DistilBERT [106]	Distillation	6	$\mathcal{L}_\text{KD-CE}$+Cos$_\text{KD}$+ $\mathcal{L}_\text{MLM}$	$\times$1.63	66M	BERT$_\text{BASE}$	77.0 (dev)
TinyBERT$^{\S\,\dagger}$ [107]		4	MSE$_\text{embed}$+MSE$_\text{attn}$+ MSE$_\text{hidn}$+$\mathcal{L}_\text{KD-CE}$	$\times$9.4	14.5M	BERT$_\text{BASE}$	76.5
BERT-PKD [169]		3 ~ 6	$\mathcal{L}_\text{KD-CE}$+PT$_\text{KD}$+ $\mathcal{L}_\text{Task}$	$\times$3.73 ~ 1.64	45.7 ~ 67 M	BERT$_\text{BASE}$	76.0 ~ 80.6$^\sharp$
PD [170]		6	$\mathcal{L}_\text{KD-CE}$+$\mathcal{L}_\text{Task}$+ $\mathcal{L}_\text{MLM}$	$\times$2.0	67.5M	BERT$_\text{BASE}$	81.2$^\sharp$
MobileBERT§[171]		24	FMT+AT+PKT+ $\mathcal{L}_\text{KD-CE}$+$\mathcal{L}_\text{MLM}$	$\times$4.0	25.3M	BERT$_\text{LARGE}$	79.7
MiniLM [108]		6	AT+AR	$\times$1.99	66M	BERT$_\text{BASE}$	81.0$^\flat$
DualTrain$^{\S\,\dagger}$[172]		12	Dual Projection+$\mathcal{L}_\text{MLM}$	-	1.8 ~ 19.2M	BERT$_\text{BASE}$	75.8 ~ 81.9$^\natural$
BERT-of-Theseus [109]	Module Replacing	6	$\mathcal{L}_\text{Task}$	$\times$1.94	66M	BERT$_\text{BASE}$	78.6

[1] The desing of this table is borrowed from [109, 173].

‡ The averaged score on 8 tasks (without WNLI) of GLUE benchmark (see Section 7.1). Here MNLI-m and MNLI-mm are regarded as two different tasks. 'dev' indicates the result is on dev set. 'ensemble' indicates the result is from the ensemble model.

* '$\mathcal{L}_\text{MLM}$ ', '$\mathcal{L}_\text{NSP}$', and '$\mathcal{L}_\text{SOP}$' indicate pre-training objective (see Section 3.1 and Table 1).'$\mathcal{L}_\text{Task}$' means task-specific loss.
'HAWQ', 'GWQ', 'DQ', and 'QAT' indicate Hessian AWare Quantization, Group-wise Quantization, Quantization-Aware Training, and Dynamically Quantized, respectively. 'KD' means knowledge distillation. 'FMT', 'AT', and 'PKT' mean Feature Map Transfer, Attention Transfer, and Progressive Knowledge Transfer, respectively. 'AR' means Self-Attention value relation.

§ The dimensionality of the hidden or embedding layers is reduced.

† Use a smaller vocabulary.

$^\flat$ Generally, the F1 score is usually used as the main metric of the QQP task. But MiniLM reports the accuracy, which is incomparable to other works.

$^\diamond$ Result on MNLI and SST-2 only.

$^\sharp$ Result on the other tasks except for STS-B and CoLA.

$^\natural$ Result on MRPC, MNLI, and SST-2 only.

the teacher model and distilling more knowledge can bring improvement to the student model.

TinyBERT [107] performs layer-to-layer distillation with embedding outputs, hidden states, and self-attention distributions. MobileBERT [171] also perform layer-to-layer distillation with soft target probabilities, hidden states, and self-attention distributions. MiniLM [108] distill self-attention distributions and self-attention value relation from teacher model.

Besides, other models distill knowledge through many approaches. Sun et al. [169] introduced a "*patient*" teacher-student mechanism, Liu et al. [179] exploited KD to improve a pre-trained multi-task deep neural network.

(3) *Distillation to other structures.* Generally, the structure of the student model is the same as the teacher model, except for a smaller layer size and a smaller hidden size. However, not only decreasing parameters but also simplifying model structures from Transformer to RNN [180] or CNN [181] can reduce the computational complexity.

4.5.5 Module Replacing

Module replacing is an interesting and simple way to reduce the model size, which replaces the large modules of original PTMs with more compact substitutes. Xu et al. [109] proposed Theseus Compression motivated by a famous thought experiment called "Ship of Theseus", which progressively substitutes modules from the source model with modules of fewer parameters. Different from KD, Theseus Compression only requires one task-specific loss function. The compressed model, BERT-of-Theseus, is 1.94× faster while retaining more than 98% performance of the source model.

4.5.6 Early Exit

Another efficient way to reduce the inference time is early exit, which allows the model to exit early at an off-ramp instead of passing through the entire model. The number of layers to be executed is conditioned on the input.

The idea of early exit is first applied in computer vision, such as BranchyNet [182] and Shallow-Deep Network [183]. With the emergence of deep pre-trained language models, early exit is recently adopted to speedup Transformer-based models. As a prior work, Universal Transformer [176] uses the Adaptive Computation Time (ACT) mechanism [184] to achieve input-adaptive computation. Elbayad et al. [185] proposed Depth-adaptive transformer for machine translation, which learns to predict how many decoding layers are required for a particular sequence or token. Instead of learning how much computation is required, Liu et al. [186] proposed two estimation approaches based on Mutual Information (MI) and Reconstruction Loss respectively to directly allocate the appropriate computation to each sample.

More recently, DeeBERT [110], RightTool [111], FastBERT [112], ELBERT [187], PABEE [113] are proposed to reduce the computation of transformer encoder for natural language understanding tasks. Their methods usually contain two steps: (a) Training the injected off-ramps (aka internal classifiers), and (b) Designing an exiting strategy to decide whether or not to exit.

Typically, the training objective is a weighted sum of the cross-entropy losses at all off-ramps, i.e.

$$\mathcal{L}_{\text{early-exit}} = \sum_{i=1}^{M} w_i \cdot \mathcal{L}_i, \tag{6}$$

where M is the number of off-ramps. FastBERT [112] adopted the self-distillation loss that trains each off-ramp with the soft target generated by the final classifier. Liao et al. [114] improved the objective by considering both the past and the future information. In particular, the off-ramps are trained to aggregate the hidden states of the past layers, and also approximate the hidden states of the future layers. Moreover, Sun et al. [115] developed a novel training objective from the perspective of ensemble learning and mutual information, by which the off-ramps are trained as an ensemble. Their proposed objective not only optimizes the accuracy of each off-ramp but also the diversity of the off-ramps.

During inference, an exiting strategy is required to decide whether to exit early or continue to the next layer. DeeBERT [110], FastBERT [112], Liao et al. [114] adopt the entropy of the prediction distribution as the exiting criterion. Similarly, RightTool [111] use the maximum softmax score to decide whether to exit. PABEE developed a patience-based strategy that allows a sample to exit when the prediction is unchanged for successive layers. Further, Sun et al. [115] adopt a voting-based strategy to let all of the past off-ramps take a vote to decide whether or not to exit. Besides, Li et al. [116] proposed a window-based uncertainty as the exiting criterion to achieve token-level early exit (TokEE) for sequence labeling tasks.

5 Adapting PTMs to Downstream Tasks

Although PTMs capture the general language knowledge from a large corpus, how effectively adapting their knowledge to the downstream task is still a key problem.

5.1 Transfer Learning

Transfer learning [188] is to adapt the knowledge from a source task (or domain) to a target task (or domain). Figure 4 gives an illustration of transfer learning.

There are many types of transfer learning in NLP, such as domain adaptation, cross-lingual learning, multi-task learning. Adapting PTMs to downstream tasks is *sequential transfer learning* task, in which tasks are learned sequentially and the target task has labeled data.

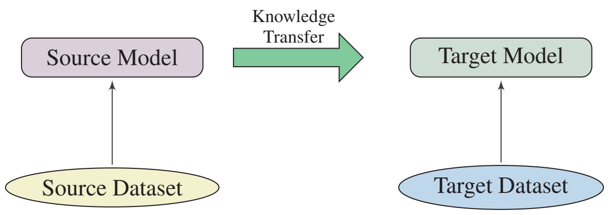


Figure 4: Transfer Learning

5.2 How to Transfer?

To transfer the knowledge of a PTM to the downstream NLP tasks, we need to consider the following issues:

5.2.1 *Choosing appropriate pre-training task, model architecture and corpus*

Different PTMs usually have different effects on the same downstream task, since these PTMs are trained with various pre-training tasks, model architecture, and corpora.

(1) Currently, the language model is the most popular pre-training task and can more efficiently solve a wide range of NLP problems [58]. However, different pre-training tasks have their own bias and give different effects for different tasks. For example, the NSP task [16] makes PTM understand the relationship between two sentences. Thus, the PTM can benefit downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI).

(2) The architecture of PTM is also important for the downstream task. For example, although BERT helps with most natural language understanding tasks, it is hard to generate language.

(3) The data distribution of the downstream task should be approximate to PTMs. Currently, there are a large number of off-the-shelf PTMs, which can just as conveniently be used for various domain-specific or language-specific downstream tasks.

Therefore, given a target task, it is always a good solution to choose the PTMs trained with appropriate pre-training task, architecture, and corpus.

5.2.2 *Choosing appropriate layers*

Given a pre-trained deep model, different layers should capture different kinds of information, such as POS tagging, parsing, long-term dependencies, semantic roles, coreference. For RNN-based models, Belinkov et al. [189] and Melamud et al. [34] showed that representations learned from different layers in a multi-layer LSTM encoder benefit different tasks (e.g., predicting POS tags and understanding word sense). For transformer-based PTMs, Tenney et al. [122] found BERT represents the steps of the traditional NLP pipeline: basic syntactic information appears earlier in the network, while high-level semantic information appears at higher layers.

Let $\mathbf{H}^{(l)}(1 \leqslant l \leqslant L)$ denotes the l-th layer representation of the pre-trained model with L layers, and $g(\cdot)$ denote the task-specific model for the target task.

There are three ways to select the representation:

a) *Embedding Only.* One approach is to choose only the pre-trained static embeddings, while the rest of the model still needs to be trained from scratch for a new target task.

They fail to capture higher-level information that might be even more useful. Word embeddings are only useful in capturing semantic meanings of words, but we also need to understand higher-level concepts like word sense.

b) *Top Layer.* The most simple and effective way is to feed the representation at the top layer into the task-specific model $g(\mathbf{H}^{(L)})$.

c) *All Layers.* A more flexible way is to automatic choose the best layer in a soft version, like ELMo [14]:

$$\mathbf{r}_t = \gamma \sum_{l=1}^{L} \alpha_l \mathbf{h}_t^{(l)}, \tag{7}$$

where α_l is the softmax-normalized weight for layer l and γ is a scalar to scale the vectors output by pre-trained model. The mixup representation is fed into the task-specific model $g(\mathbf{r}_t)$.

5.2.3 *To tune or not to tune?*

Currently, there are two common ways of model transfer: feature extraction (where the pre-trained parameters are frozen), and fine-tuning (where the pre-trained parameters are unfrozen and fine-tuned).

In feature extraction way, the pre-trained models are regarded as off-the-shelf feature extractors. Moreover, it is important to expose the internal layers as they typically encode the most transferable representations [190].

Although both these two ways can significantly benefit most of NLP tasks, feature extraction way requires more complex task-specific architecture. Therefore, the fine-tuning way is usually more general and convenient for many different downstream tasks than feature extraction way.

Table 4 gives some common combinations of adapting PTMs.

Table 4: Some common combinations of adapting PTMs.

Where	FT/FE?[†]	PTMs
Embedding Only	FT/FE	Word2vec [11], GloVe [12]
Top Layer	FT	BERT [16], RoBERTa [43]
Top Layer	FE	BERT[§] [191, 192]
All Layers	FE	ELMo [14]

[†] FT and FE mean Fine-tuning and Feature Extraction respectively.
[§] BERT used as feature extractor.

5.3 Fine-Tuning Strategies

With the increase of the depth of PTMs, the representation captured by them makes the downstream task easier. Therefore, the task-specific layer of the whole model is simple. Since ULMFit and BERT, fine-tuning has become the main adaption method of PTMs. However, the process of fine-tuning is often brittle: even with the same hyper-parameter values, distinct random seeds can lead to substantially different results [193].

Besides standard fine-tuning, there are also some useful fine-tuning strategies.

Two-stage fine-tuning An alternative solution is *two-stage transfer*, which introduces an intermediate stage between pre-training and fine-tuning. In the first stage, the PTM is transferred into a model fine-tuned by an intermediate task or corpus. In the second stage, the transferred model is fine-tuned to the target task. Sun et al. [64] showed that the “further pre-training” on the related-domain corpus can further improve the ability of BERT and achieved state-of-the-art performance on eight widely-studied text classification datasets. Phang et al. [194] and Garg et al. [195] introduced the intermediate supervised task related to the target task, which brings a large improvement for BERT, GPT, and ELMo. Li et al. [65] also used a two-stage transfer for the story ending prediction. The proposed TransBERT (transferable BERT) can transfer not only general language knowledge from large-scale unlabeled data but also specific kinds of knowledge from various semantically related supervised tasks.

Multi-task fine-tuning Liu et al. [67] fine-tuned BERT under the multi-task learning framework, which demonstrates that multi-task learning and pre-training are complementary technologies.

Fine-tuning with extra adaptation modules The main drawback of fine-tuning is its parameter inefficiency: every downstream task has its own fine-tuned parameters. Therefore, a better solution is to inject some fine-tunable adaptation modules into PTMs while the original parameters are fixed.

Stickland and Murray [68] equipped a single share BERT model with small additional task-specific adaptation modules, projected attention layers (PALs). The shared BERT with the PALs matches separately fine-tuned models on the GLUE benchmark with roughly 7 times fewer parameters. Similarly, Houlsby et al. [69] modified the architecture of pre-trained BERT by adding adapter modules. Adapter modules yield a compact and extensible model; they add only a few trainable parameters per task, and new tasks can be added without revisiting previous ones. The parameters of the original network remain fixed, yielding a high degree of parameter sharing.

Others Motivated by the success of widely-used ensemble models, Xu et al. [196] improved the fine-tuning of BERT with two effective mechanisms: *self-ensemble* and *self-distillation*, which can improve the performance of BERT on downstream tasks without leveraging external resource or significantly decreasing the training efficiency. They integrated ensemble and distillation within a single training process. The teacher model is an ensemble model by parameter-averaging several student models in previous time steps.

Instead of fine-tuning all the layers simultaneously, *gradual unfreezing* [38] is also an effective method that gradually unfreezes layers of PTMs starting from the top layer. Chronopoulou et al. [197] proposed a simpler unfreezing method, *sequential unfreezing*, which first fine-tunes only the randomly-initialized task-specific layers, and then unfreezes the hidden layers of PTM, and finally unfreezes the embedding layer.

Li and Eisner [198] compressed ELMo embeddings using variational information bottleneck while keeping only the information that helps the target task.

Generally, the above works show that the utility of PTMs can be further stimulated by better fine-tuning strategies.

5.3.1 Prompt-based Tuning

Narrowing the gap between pre-training and fine-tuning can further boost the performance of PTMs on downstream tasks. An alternative approach is reformulating the downstream tasks into a MLM task by designing appropriate prompts. Prompt-based methods have shown great power in few-shot setting [199, 200, 70, 72], zero-shot setting [129, 201], and even fully-supervised setting [74, 75]. Current prompt-based methods can be categorized as two branches according to the prompt is whether discrete or continuous.

Discrete prompts Discrete prompt is a sequence of words to be inserted into the input text, which helps the PTM to better model the downstream task. Sun et al. [202] constructed an auxiliary sentence by transforming aspect-based sentiment analysis (ABSA) task to a sentence pair classification task, but its model parameters still need to be fine-tuned. GPT-3 [59] proposed the in-context learning that concatenates the original input with the task description and a few examples. By this, GPT-3 can achieve competitive performance without tuning the parameters. Besides, Petroni et al. [129] found that with proper manual prompt, BERT can perform well on entity prediction task (LAMA) without training. In addition to LAMA, Schick and Schütze [200, 70] proposed PET that designed discrete prompts for various text classification and entailment tasks. However, the manually designed prompts can be suboptimal, as a result, many methods are developed to automate the generation of prompts. LPAQA [201] uses two methods, i.e., mining-based generation and paraphrasing-based generation, to find the optimal patterns that express particular relations. AutoPrompt [71] finds the optimal prompt with gradient-guided search. LM-BFF [72] employs T5 [42] to automatically generate prompts.

Continuous prompts Instead of finding the optimal concrete prompt, another alternative is to directly optimize the prompt in continuous space, i.e. the prompt vectors are not necessarily word type embeddings of the PTM. The optimized continuous prompt is concatenated with word type embeddings, which is then fed into the PTM. Qin and Eisner [203] and Zhong et al. [204] found that the optimized continuous prompt can outperform concrete prompts (including manual [129], mined (LPAQA [201]), and gradient-searched (AutoPrompt [71]) prompts) on relational tasks. WARP [73] inserts trainable continuous prompt tokens before, between, and after the input sequence while keeping the parameters of the PTM fixed, resulting in considerable performance on GLUE benchmark. Prefix-Tuning [74] inserts continuous prompt as prefix of the input of GPT-2 for table-to-text generation and BART for summarization. Prefix-Tuning, as a parameter-efficient tuning technique, achieved comparable performance in fully-supervised setting and outperformed model fine-tuning in few-shot setting. Further, P-Tuning [75] showed that, with continuous prompt, GPT can also achieve comparable or even better performance to similar-sized BERT on natural language understanding (NLU) tasks. Very recently, Lester et al. [205] showed that prompt tuning becomes more competitive with scale. When the PTM exceeds billions of parameters, the gap between model fine-tuning and prompt tuning can be closed, which makes the prompt-based tuning a very promising method for efficient serving of large-scale PTMs.

6 Resources of PTMs

There are many related resources for PTMs available online. Table 5 provides some popular repositories, including third-party implementations, paper lists, visualization tools, and other related resources of PTMs.

Besides, there are some other good survey papers on PTMs for NLP [211, 212, 173].

7 Applications

In this section, we summarize some applications of PTMs in several classic NLP tasks.

Table 5: Resources of PTMs

Resource	Description	URL
	Open-Source Implementations §	
word2vec	CBOW,Skip-Gram	https://github.com/tmikolov/word2vec
GloVe	Pre-trained word vectors	https://nlp.stanford.edu/projects/glove
FastText	Pre-trained word vectors	https://github.com/facebookresearch/fastText
Transformers	Framework: PyTorch&TF, PTMs: BERT, GPT-2, RoBERTa, XLNet, etc.	https://github.com/huggingface/transformers
Fairseq	Framework: PyTorch, PTMs:English LM, German LM, RoBERTa, etc.	https://github.com/pytorch/fairseq
Flair	Framework: PyTorch, PTMs:BERT, ELMo, GPT, RoBERTa, XLNet, etc.	https://github.com/flairNLP/flair
AllenNLP [206]	Framework: PyTorch, PTMs: ELMo, BERT, GPT-2, etc.	https://github.com/allenai/allennlp
fastNLP	Framework: PyTorch, PTMs: RoBERTa, GPT, etc.	https://github.com/fastnlp/fastNLP
UniLMs	Framework: PyTorch, PTMs: UniLM v1&v2, MiniLM, LayoutLM, etc.	https://github.com/microsoft/unilm
Chinese-BERT [85]	Framework: PyTorch&TF, PTMs: BERT, RoBERTa, etc. (for Chinese)	https://github.com/ymcui/Chinese-BERT-wwm
BERT [16]	Framework: TF, PTMs: BERT, BERT-wwm	https://github.com/google-research/bert
RoBERTa [43]	Framework: PyTorch	https://github.com/pytorch/fairseq/tree/master/examples/roberta
XLNet [49]	Framework: TF	https://github.com/zihangdai/xlnet/
ALBERT [63]	Framework: TF	https://github.com/google-research/ALBERT
T5 [42]	Framework: TF	https://github.com/google-research/text-to-text-transfer-transformer
ERNIE(Baidu) [84, 153]	Framework: PaddlePaddle	https://github.com/PaddlePaddle/ERNIE
CTRL [207]	Conditional Transformer Language Model for Controllable Generation.	https://github.com/salesforce/ctrl
BertViz [208]	Visualization Tool	https://github.com/jessevig/bertviz
exBERT [209]	Visualization Tool	https://github.com/bhoov/exbert
TextBrewer [210]	PyTorch-based toolkit for distillation of NLP models.	https://github.com/airaria/TextBrewer
DeepPavlov	Conversational AI Library. PTMs for the Russian, Polish, Bulgarian, Czech, and informal English.	https://github.com/deepmipt/DeepPavlov
	Corpora	
OpenWebText	Open clone of OpenAI's unreleased WebText dataset.	https://github.com/jcpeterson/openwebtext
Common Crawl	A very large collection of text.	http://commoncrawl.org/
WikiEn	English Wikipedia dumps.	https://dumps.wikimedia.org/enwiki/
	Other Resources	
Paper List		https://github.com/thunlp/PLMpapers
Paper List		https://github.com/tomohideshibata/BERT-related-papers
Paper List		https://github.com/cedrickchee/awesome-bert-nlp
Bert Lang Street	A collection of BERT models with reported performances on different datasets, tasks and languages.	https://bertlang.unibocconi.it/

§ Most papers for PTMs release their links of official version. Here we list some popular third-party and official implementations.

7.1 General Evaluation Benchmark

There is an essential issue for the NLP community that how can we evaluate PTMs in a comparable metric. Thus, large-scale-benchmark is necessary.

The General Language Understanding Evaluation (GLUE) benchmark [213] is a collection of nine natural language understanding tasks, including single-sentence classification tasks (CoLA and SST-2), pairwise text classification tasks (MNLI, RTE, WNLI, QQP, and MRPC), text similarity task (STS-B), and relevant ranking task (QNLI). GLUE benchmark is well-designed for evaluating the robustness as well as generalization of models. GLUE does not provide the labels for the test set but set up an evaluation server.

However, motivated by the fact that the progress in recent years has eroded headroom on the GLUE benchmark dramatically, a new benchmark called SuperGLUE [214] was presented. Compared to GLUE, SuperGLUE has more challenging tasks and more diverse task formats (e.g., coreference resolution and question answering).

State-of-the-art PTMs are listed in the corresponding leaderboard[4) 5)].

7.2 Question Answering

Question answering (QA), or a narrower concept machine reading comprehension (MRC), is an important application in the NLP community. From easy to hard, there are three types of QA tasks: single-round extractive QA (SQuAD) [215], multi-round generative QA (CoQA) [216], and multi-hop QA (HotpotQA) [217].

BERT creatively transforms the extractive QA task to the spans prediction task that predicts the starting span as well as the ending span of the answer [16]. After that, PTM as an encoder for predicting spans has become a competitive baseline. For extractive QA, Zhang et al. [218] proposed a ret-

4) https://gluebenchmark.com/
5) https://super.gluebenchmark.com/

rospective reader architecture and initialize the encoder with PTM (e.g., ALBERT). For multi-round generative QA, Ju et al. [219] proposed a "PTM+Adversarial Training+Rationale Tagging+Knowledge Distillation" model. For multi-hop QA, Tu et al. [220] proposed an interpretable "Select, Answer, and Explain" (SAE) system that PTM acts as the encoder in the selection module.

Generally, encoder parameters in the proposed QA model are initialized through a PTM, and other parameters are randomly initialized. State-of-the-art models are listed in the corresponding leaderboard. [6) 7) 8)]

7.3 Sentiment Analysis

BERT outperforms previous state-of-the-art models by simply fine-tuning on SST-2, which is a widely used dataset for sentiment analysis (SA) [16]. Bataa and Wu [221] utilized BERT with transfer learning techniques and achieve new state-of-the-art in Japanese SA.

Despite their success in simple sentiment classification, directly applying BERT to aspect-based sentiment analysis (ABSA), which is a fine-grained SA task, shows less significant improvement [202]. To better leverage the powerful representation of BERT, Sun et al. [202] constructed an auxiliary sentence by transforming ABSA from a single sentence classification task to a sentence pair classification task. Xu et al. [222] proposed post-training to adapt BERT from its source domain and tasks to the ABSA domain and tasks. Furthermore, Rietzler et al. [223] extended the work of [222] by analyzing the behavior of cross-domain post-training with ABSA performance. Karimi et al. [224] showed that the performance of post-trained BERT could be further improved via adversarial training. Song et al. [225] added an additional pooling module, which can be implemented as either LSTM or attention mechanism, to leverage BERT intermediate layers for ABSA. In addition, Li et al. [226] jointly learned aspect detection and sentiment classification towards end-to-end ABSA. SentiLR [79] acquires part-of-speech tag and prior sentiment polarity from SentiWordNet and adopts *Label-Aware MLM* to utilize the introduced linguistic knowledge to capture the relationship between sentence-level sentiment labels and word-level sentiment shifts. SentiLR achieves state-of-the-art performance on several sentence- and aspect-level sentiment classification tasks.

For sentiment transfer, Wu et al. [227] proposed "Mask and Infill" based on BERT. In the mask step, the model disentangles sentiment from content by masking sentiment tokens. In the infill step, it uses BERT along with a target sentiment embedding to infill the masked positions.

7.4 Named Entity Recognition

Named Entity Recognition (NER) in information extraction and plays an important role in many NLP downstream tasks. In deep learning, most of NER methods are in the sequence-labeling framework. The entity information in a sentence will be transformed into the sequence of labels, and one label corresponds to one word. The model is used to predict the label of each word. Since ELMo and BERT have shown their power in NLP, there is much work about pre-trained models for NER.

Akbik et al. [37] used a pre-trained character-level language model to produce word-level embedding for NER. TagLM [228] and ELMo [14] use a pre-trained language model's last layer output and weighted-sum of each layer output as a part of word embedding. Liu et al. [229] used layer-wise pruning and dense connection to speed up ELMo's inference on NER. Devlin et al. [16] used the first BPE's BERT representation to predict each word's label without CRF. Pires et al. [150] realized zero-shot NER through multilingual BERT. Tsai et al. [178] leveraged knowledge distillation to run a small BERT for NER on a single CPU. Besides, BERT is also used on domain-specific NER, such as biomedicine [230, 100], etc.

7.5 Machine Translation

Machine Translation (MT) is an important task in the NLP community, which has attracted many researchers. Almost all of Neural Machine Translation (NMT) models share the encoder-decoder framework, which first encodes input tokens to hidden representations by the encoder and then decodes output tokens in the target language from the decoder. Ramachandran et al. [36] found the encoder-decoder models can be significantly improved by initializing both encoder and decoder with pre-trained weights of two language models. Edunov et al. [231] used ELMo to set the word embedding layer in the NMT model. This work shows performance improvements on English-Turkish and English-German NMT model by using a pre-trained language model for source word embedding initialization.

Given the superb performance of BERT on other NLP tasks, it is natural to investigate how to incorporate BERT into NMT models. Conneau and Lample [46] tried to initialize the entire encoder and decoder by a multilingual pre-trained BERT model and showed a significant improvement could be achieved on unsupervised MT and English-Romanian supervised MT. Similarly, Clinchant et al. [232] devised a series

6) https://rajpurkar.github.io/SQuAD-explorer/
7) https://stanfordnlp.github.io/coqa/
8) https://hotpotqa.github.io/

of different experiments for examining the best strategy to utilize BERT on the encoder part of NMT models. They achieved some improvement by using BERT as an initialization of the encoder. Also, they found that these models can get better performance on the out-of-domain dataset. Imamura and Sumita [233] proposed a two stages BERT fine-tuning method for NMT. At the first stage, the encoder is initialized by a pre-trained BERT model, and they only train the decoder on the training set. At the second stage, the whole NMT model is jointly fine-tuned on the training set. By experiment, they show this approach can surpass the one stage fine-tuning method, which directly fine-tunes the whole model. Apart from that, Zhu et al. [192] suggested using pre-trained BERT as an extra memory to facilitate NMT models. Concretely, they first encode the input tokens by a pre-trained BERT and use the output of the last layer as extra memory. Then, the NMT model can access the memory via an extra attention module in each layer of both encoder and decoder. And they show a noticeable improvement in supervised, semi-supervised, and unsupervised MT.

Instead of only pre-training the encoder, MASS (Masked Sequence-to-Sequence Pre-Training) [41] utilizes Seq2Seq MLM to pre-train the encoder and decoder jointly. In the experiment, this approach can surpass the BERT-style pre-training proposed by Conneau and Lample [46] both on unsupervised MT and English-Romanian supervised MT. Different from MASS, mBART [61], a multilingual extension of BART [50], pre-trains the encoder and decoder jointly with Seq2Seq denoising auto-encoder (DAE) task on large-scale monolingual corpora across 25 languages. Experiments demonstrated that mBART could significantly improve both supervised and unsupervised machine translation at both the sentence level and document level.

7.6 Summarization

Summarization, aiming at producing a shorter text which preserves the most meaning of a longer text, has attracted the attention of the NLP community in recent years. The task has been improved significantly since the widespread use of PTM. Zhong et al. [191] introduced transferable knowledge (e.g., BERT) for summarization and surpassed previous models. Zhang et al. [234] tries to pre-trained a document-level model that predicts sentences instead of words, and then apply it on downstream tasks such as summarization. More elaborately, Zhang et al. [163] designed a Gap Sentence Generation (GSG) task for pre-training, whose objective involves generating summary-like text from the input. Furthermore, Liu and Lapata [235] proposed BERTSUM. BERTSUM included a novel document-level encoder, and a general framework for both extractive summarization and abstractive summarization. In the encoder frame, BERTSUM extends BERT by inserting multiple `[CLS]` tokens to learn the sentence representations. For extractive summarization, BERTSUM stacks several inter-sentence Transformer layers. For abstractive summarization, BERTSUM proposes a two-staged fine-tuning approach using a new fine-tuning schedule. Zhong et al. [236] proposed a novel summary-level framework MATCHSUM and conceptualized extractive summarization as a semantic text matching problem. They proposed a Siamese-BERT architecture to compute the similarity between the source document and the candidate summary and achieved a state-of-the-art result on CNN/DailyMail (44.41 in ROUGE-1) by only using the base version of BERT.

7.7 Adversarial Attacks and Defenses

The deep neural models are vulnerable to adversarial examples that can mislead a model to produce a specific wrong prediction with imperceptible perturbations from the original input. In CV, adversarial attacks and defenses have been widely studied. However, it is still challenging for text due to the discrete nature of languages. Generating of adversarial samples for text needs to possess such qualities: (1) imperceptible to human judges yet misleading to neural models; (2) fluent in grammar and semantically consistent with original inputs. Jin et al. [237] successfully attacked the fine-tuned BERT on text classification and textual entailment with adversarial examples. Wallace et al. [238] defined universal adversarial triggers that can induce a model to produce a specific-purpose prediction when concatenated to any input. Some triggers can even cause the GPT-2 model to generate racist text. Sun et al. [239] showed BERT is not robust on misspellings.

PTMs also have great potential to generate adversarial samples. Li et al. [240] proposed **BERT-Attack**, a BERT-based high-quality and effective attacker. They turned BERT against another fine-tuned BERT on downstream tasks and successfully misguided the target model to predict incorrectly, outperforming state-of-the-art attack strategies in both success rate and perturb percentage, while the generated adversarial samples are fluent and semantically preserved.

Besides, adversarial defenses for PTMs are also promising, which improve the robustness of PTMs and make them immune against adversarial attack.

Adversarial training aims to improve the generalization by minimizes the maximal risk for label-preserving perturbations in embedding space. Recent work [241, 242] showed that adversarial pre-training or fine-tuning can improve both generalization and robustness of PTMs for NLP.

8 Future Directions

Though PTMs have proven their power for various NLP tasks, challenges still exist due to the complexity of language. In this section, we suggest five future directions of PTMs.

(1) Upper Bound of PTMs Currently, PTMs have not yet reached its upper bound. Most of the current PTMs can be further improved by more training steps and larger corpora.

The state of the art in NLP can be further advanced by increasing the depth of models, such as Megatron-LM [243] (8.3 billion parameters, 72 Transformer layers with a hidden size of 3072 and 32 attention heads) and Turing-NLG[9] (17 billion parameters, 78 Transformer layers with a hidden size of 4256 and 28 attention heads).

The general-purpose PTMs are always our pursuits for learning the intrinsic universal knowledge of languages (even world knowledge). However, such PTMs usually need deeper architecture, larger corpus, and challenging pre-training tasks, which further result in higher training costs. However, training huge models is also a challenging problem, which needs more sophisticated and efficient training techniques such as distributed training, mixed precision, gradient accumulation, etc. Therefore, a more practical direction is to design more efficient model architecture, self-supervised pre-training tasks, optimizers, and training skills using existing hardware and software. ELECTRA [56] is a good solution towards this direction.

(2) Architecture of PTMs The Transformer has been proved to be an effective architecture for pre-training. However, the main limitation of the Transformer is its computation complexity, which is quadratic to the input length. Limited by the memory of GPUs, most of current PTMs cannot deal with the sequence longer than 512 tokens. Breaking this limit needs to improve the architecture of the Transformer. Although many works [25] tried to improve the efficiency of Transformer, there remains much room for improvement.

Besides, searching for more efficient alternative non-Transformer architecture for PTMs is important to capture longer-range contextual information. The design of deep architecture is challenging, and we may seek help from some automatic methods, such as neural architecture search (NAS) [245].

(3) Task-oriented Pre-training and Model Compression In practice, different downstream tasks require the different abilities of PTMs. The discrepancy between PTMs and downstream tasks usually lies in two aspects: model architecture and data distribution. A larger discrepancy may result in that the benefit of PTMs may be insignificant. For example, text generation usually needs a specific task to pre-train both the encoder and decoder, while text matching needs pre-training tasks designed for sentence pairs.

Besides, although larger PTMs can usually lead to better performance, a practical problem is how to leverage these huge PTMs on special scenarios, such as low-capacity devices and low-latency applications. Therefore, we can carefully design the specific model architecture and pre-training tasks for downstream tasks or extract partial task-specific knowledge from existing PTMs.

Instead of training task-oriented PTMs from scratch, we can teach them with existing general-purpose PTMs by using techniques such as model compression (see Section 4.5). Although model compression is widely studied for CNNs in CV [246], compression for PTMs for NLP is just beginning. The fully-connected structure of the Transformer also makes model compression more challenging.

(4) Knowledge Transfer Beyond Fine-tuning Currently, fine-tuning is the dominant method to transfer PTMs' knowledge to downstream tasks, but one deficiency is its parameter inefficiency: every downstream task has its own fine-tuned parameters. An improved solution is to fix the original parameters of PTMs and by adding small fine-tunable adaption modules for specific task [68, 69]. Thus, we can use a shared PTM to serve multiple downstream tasks. Indeed, mining knowledge from PTMs can be more flexible, such as feature extraction, knowledge distillation [210], data augmentation [247, 248], using PTMs as external knowledge [129]. More efficient methods are expected.

(5) Interpretability and Reliability of PTMs Although PTMs reach impressive performance, their deep non-linear architecture makes the procedure of decision-making highly non-transparent.

Recently, explainable artificial intelligence (XAI) [249] has become a hotspot in the general AI community. Unlike CNNs for images, interpreting PTMs is harder due to the complexities of both the Transformer-like architecture and language. Extensive efforts (see Section 3.3) have been made to analyze the linguistic and world knowledge included in PTMs, which help us understand these PMTs with some degree of transparency. However, much work on model analysis depends on the attention mechanism, and the effectiveness of attention for interpretability is still controversial [250, 251].

Besides, PTMs are also vulnerable to adversarial attacks (see Section 7.7). The reliability of PTMs is also becoming an issue of great concern with the extensive use of PTMs in production systems. The studies of adversarial attacks against PTMs help us understand their capabilities by fully exposing

9) https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

their vulnerabilities. Adversarial defenses for PTMs are also promising, which improve the robustness of PTMs and make them immune against adversarial attack.

Overall, as key components in many NLP applications, the interpretability and reliability of PTMs remain to be explored further in many respects, which helps us understand how PTMs work and provides a guide for better usage and further improvement.

9 Conclusion

In this survey, we conduct a comprehensive overview of PTMs for NLP, including background knowledge, model architecture, pre-training tasks, various extensions, adaption approaches, related resources, and applications. Based on current PTMs, we propose a new taxonomy of PTMs from four different perspectives. We also suggest several possible future research directions for PTMs.

Acknowledgements

We thank Zhiyuan Liu, Wanxiang Che, Minlie Huang, Danqing Wang and Luyao Huang for their valuable feedback on this manuscript. This work was supported by the National Natural Science Foundation of China (No. 61751201 and 61672162), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab.

References

[1] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *ACL*, 2014.

[2] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[3] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *ICML*, pages 1243–1252, 2017.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *NeurIPS*, pages 3104–3112, 2014.

[5] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *IJCAI*, 2016.

[6] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. ACL, 2013.

[7] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566, 2015.

[8] Diego Marcheggiani, Joost Bastings, and Ivan Titov. Exploiting semantics in neural machine translation with graph convolutional networks. In *NAACL-HLT*, pages 486–492, 2018.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2014.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *EMNLP*, 2014.

[13] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *NeurIPS*, 2017.

[14] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, 2018.

[15] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/languageunderstandingpaper.pdf.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[17] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35 (8):1798–1828, 2013.

[18] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, 2016.

[19] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL*, 5:135–146, 2017. doi: https://doi.org/10.1162/tacl_a_00051.

[20] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. doi: https://doi.org/10.1162/neco.1997.9.8.1735.

[22] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent

neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[23] Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612, 2015.

[24] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[25] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[26] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. In *NAACL-HLT*, pages 1315–1325, 2019.

[27] Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, 2010. doi: https://dl.acm.org/doi/10.5555/1756006.1756025.

[28] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313 (5786):504–507, 2006. doi: https://doi.org/10.1126/science.1127647.

[29] GE Hinton, JL McClelland, and DE Rumelhart. Distributed representations. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pages 77–109. 1986.

[30] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155, 2003. doi: https://dl.acm.org/doi/10.5555/944919.944966.

[31] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 2011. doi: https://dl.acm.org/doi/10.5555/1953048.2078186.

[32] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[33] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *NeurIPS*, pages 3294–3302, 2015.

[34] Oren Melamud, Jacob Goldberger, and Ido Dagan. Context2Vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*, pages 51–61, 2016.

[35] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *NeurIPS*, pages 3079–3087, 2015.

[36] Prajit Ramachandran, Peter J Liu, and Quoc Le. Unsupervised pretraining for sequence to sequence learning. In *EMNLP*, pages 383–391, 2017.

[37] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING*, pages 1638–1649, 2018.

[38] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *ACL*, pages 328–339, 2018.

[39] Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. Cloze-driven pretraining of self-attention networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *EMNLP-IJCNLP*, pages 5359–5368, 2019.

[40] Wilson L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953. doi: https://doi.org/10.1177/107769905303000401.

[41] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MASS: masked sequence to sequence pre-training for language generation. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936, 2019.

[42] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

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

[44] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *NeurIPS*, pages 13042–13054, 2019.

[45] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, et al. UniLMv2: Pseudo-masked language models for unified language model pre-training. *arXiv preprint arXiv:2002.12804*, 2020.

[46] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *NeurIPS*, pages 7057–7067, 2019.

[47] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2019. doi: https://doi.org/10.1162/tacl_a_00300.

[48] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. StructBERT: Incorporating language structures into pre-training for deep language understanding. In *ICLR*, 2020.

[49] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764, 2019.

[50] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-

sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[51] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *ICML*, pages 5628–5637, 2019.

[52] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *NeurIPS*, pages 2265–2273, 2013.

[53] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, pages 297–304, 2010.

[54] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[55] Lingpeng Kong, Cyprien de Masson d'Autume, Lei Yu, Wang Ling, Zihang Dai, and Dani Yogatama. A mutual information maximization perspective of language representation learning. In *ICLR*, 2019.

[56] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[57] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *ICLR*, 2020.

[58] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[59] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[60] Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. Cross-lingual natural language generation via pre-training. In *AAAI*, 2019.

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

[62] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[63] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[64] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206, 2019.

[65] Zhongyang Li, Xiao Ding, and Ting Liu. Story ending prediction by transferable bert. In *IJCAI*, pages 1800–1806, 2019.

[66] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, pages 8342–8360, 2020.

[67] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *ACL*, 2019.

[68] Asa Cooper Stickland and Iain Murray. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *ICML*, pages 5986–5995, 2019.

[69] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, pages 2790–2799, 2019.

[70] Timo Schick and Hinrich Schütze. 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, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2339–2352. Association for Computational Linguistics, 2021.

[71] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics, 2020.

[72] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

[73] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: word-level adversarial reprogramming. *arXiv preprint arXiv:2101.00121*, 2021.

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

[75] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie

Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

[76] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: enhanced language representation with informative entities. In *ACL*, 2019.

[77] Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP*, 2019.

[78] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: Enabling language representation with knowledge graph. In *AAAI*, 2019.

[79] Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. SentiLR: Linguistic knowledge enhanced language representation for sentiment analysis. *arXiv preprint arXiv:1911.02493*, 2019.

[80] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *arXiv preprint arXiv:1911.06136*, 2019.

[81] Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. Colake: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3660–3670. International Committee on Computational Linguistics, 2020.

[82] Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In *EMNLP-IJCNLP*, pages 2485–2494, 2019.

[83] Julian Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kadras, Sylvain Gugger, and Jeremy Howard. MultiFiT: Efficient multi-lingual language model fine-tuning. In *EMNLP-IJCNLP*, pages 5701–5706, 2019.

[84] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. ERNIE: enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[85] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. Pre-training with whole word masking for chinese BERT. *arXiv preprint arXiv:1906.08101*, 2019.

[86] Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin, Xin Jiang, Xiao Chen, and Qun Liu. NEZHA: Neural contextualized representation for chinese language understanding. *arXiv preprint arXiv:1909.00204*, 2019.

[87] Shizhe Diao, Jiaxin Bai, Yan Song, Tong Zhang, and Yonggang Wang. ZEN: pre-training chinese text encoder enhanced by n-gram representations. *arXiv preprint arXiv:1911.00720*, 2019.

[88] Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. BERTje: A Dutch BERT model. *arXiv preprint arXiv:1912.09582*, 2019.

[89] Louis Martin, Benjamin Müller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a tasty French language model. *arXiv preprint arXiv:1911.03894*, 2019.

[90] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. FlauBERT: Unsupervised language model pre-training for French. *arXiv preprint arXiv:1912.05372*, 2019.

[91] Pieter Delobelle, Thomas Winters, and Bettina Berendt. RobBERT: a Dutch RoBERTa-based language model. *arXiv preprint arXiv:2001.06286*, 2020.

[92] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, pages 13–23, 2019.

[93] Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. In *EMNLP-IJCNLP*, pages 5099–5110, 2019.

[94] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[95] Chris Alberti, Jeffrey Ling, Michael Collins, and David Reitter. Fusion of detected objects in text for visual question answering. In *EMNLP-IJCNLP*, pages 2131–2140, 2019.

[96] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. VL-BERT: Pre-training of generic visual-linguistic representations. In *ICLR*, 2020.

[97] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. VideoBERT: A joint model for video and language representation learning. In *ICCV*, pages 7463–7472. IEEE, 2019.

[98] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*, 2019.

[99] Yung-Sung Chuang, Chi-Liang Liu, and Hung-yi Lee. SpeechBERT: Cross-modal pre-trained language model for end-to-end spoken question answering. *arXiv preprint arXiv:1910.11559*, 2019.

[100] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 2019.

[101] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pre-trained language model for scientific text. In *EMNLP-IJCNLP*,

pages 3613–3618, 2019.

[102] Jieh-Sheng Lee and Jieh Hsiang. PatentBERT: Patent classification with fine-tuning a pre-trained BERT model. *arXiv preprint arXiv:1906.02124*, 2019.

[103] Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing BERT: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

[104] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-BERT: Hessian based ultra low precision quantization of BERT. In *AAAI*, 2020.

[105] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8BERT: Quantized 8bit BERT. *arXiv preprint arXiv:1910.06188*, 2019.

[106] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[107] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[108] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*, 2020.

[109] Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. BERT-of-Theseus: Compressing BERT by progressive module replacing. *arXiv preprint arXiv:2002.02925*, 2020.

[110] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online, July 2020. Association for Computational Linguistics.

[111] Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. The right tool for the job: Matching model and instance complexities. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6640–6651. Association for Computational Linguistics, 2020.

[112] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. FastBERT: a self-distilling BERT with adaptive inference time. In *ACL*, 2020.

[113] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *arXiv preprint arXiv:2006.04152*, 2020.

[114] Kaiyuan Liao, Yi Zhang, Xuancheng Ren, Qi Su, Xu Sun, and Bin He. A global past-future early exit method for accelerating inference of pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2013–2023. Association for Computational Linguistics, 2021.

[115] Tianxiang Sun, Yunhua Zhou, Xiangyang Liu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. Early exiting with ensemble internal classifiers. *arXiv preprint arXiv: 2105.13792*, 2021.

[116] Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. Accelerating bert inference for sequence labeling via early-exit. In *ACL*, 2021.

[117] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[118] Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. How well do distributional models capture different types of semantic knowledge? In *ACL*, pages 726–730, 2015.

[119] Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. Distributional vectors encode referential attributes. In *EMNLP*, pages 12–21, 2015.

[120] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *ICLR*, 2019.

[121] Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In *NAACL-HLT*, pages 1073–1094, 2019.

[122] Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *ACL*, pages 4593–4601, 2019.

[123] Yoav Goldberg. Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287*, 2019.

[124] Allyson Ettinger. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *TACL*, 8:34–48, 2020.

[125] John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *NAACL-HLT*, pages 4129–4138, 2019.

[126] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *ACL*, pages 3651–3657, 2019.

[127] Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang goo Lee. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *ICLR*, 2020.

[128] Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. Visualizing and measuring the geometry of BERT. In *NeurIPS*, pages 8592–8600, 2019.

[129] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. Language models as knowledge bases? In *EMNLP-IJCNLP*, pages 2463–2473, 2019.

[130] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *arXiv preprint arXiv:1911.12543*, 2019.

[131] Nina Pörner, Ulli Waltinger, and Hinrich Schütze. BERT is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised QA. *CoRR*, abs/1911.03681, 2019.

[132] Nora Kassner and Hinrich Schütze. Negated LAMA: birds cannot fly. *arXiv preprint arXiv:1911.03343*, 2019.

[133] Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. Inducing relational knowledge from BERT. In *AAAI*, 2019.

[134] Joe Davison, Joshua Feldman, and Alexander M. Rush. Commonsense knowledge mining from pretrained models. In *EMNLP-IJCNLP*, pages 1173–1178, 2019.

[135] Anne Lauscher, Ivan Vulic, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavas. Informing unsupervised pretraining with external linguistic knowledge. *arXiv preprint arXiv:1909.02339*, 2019.

[136] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*, 2020.

[137] Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. SenseBERT: Driving some sense into BERT. *arXiv preprint arXiv:1908.05646*, 2019.

[138] Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. A knowledge-enhanced pretraining model for commonsense story generation. *arXiv preprint arXiv:2001.05139*, 2020.

[139] Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. Integrating graph contextualized knowledge into pre-trained language models. *arXiv preprint arXiv:1912.00147*, 2019.

[140] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *EMNLP*, pages 1591–1601, 2014.

[141] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. Aligning knowledge and text embeddings by entity descriptions. In *EMNLP*, pages 267–272, 2015.

[142] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *IJCAI*, 2016.

[143] Jiacheng Xu, Xipeng Qiu, Kan Chen, and Xuanjing Huang. Knowledge graph representation with jointly structural and textual encoding. In *IJCAI*, pages 1318–1324, 2017.

[144] An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *ACL*, pages 2346–2357, 2019.

[145] Robert L. Logan IV, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*, 2019.

[146] Hiroaki Hayashi, Zecong Hu, Chenyan Xiong, and Graham Neubig. Latent relation language models. In *AAAI*, 2019.

[147] Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *EACL*, pages 462–471, 2014.

[148] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, 2015.

[149] Karan Singla, Doğan Can, and Shrikanth Narayanan. A multi-task approach to learning multilingual representations. In *ACL*, pages 214–220, 2018.

[150] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *ACL*, pages 4996–5001, 2019.

[151] Karthikeyan K, Zihan Wang, Stephen Mayhew, and Dan Roth. Cross-lingual ability of multilingual BERT: An empirical study. In *ICLR*, 2020.

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

[153] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE 2.0: A continual pre-training framework for language understanding. In *AAAI*, 2019.

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

[155] Wissam Antoun, Fady Baly, and Hazem Hajj. AraBERT: Transformer-based model for Arabic language understanding. *arXiv preprint arXiv:2003.00104*, 2020.

[156] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Xilin Chen, and Ming Zhou. UniViLM: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*, 2020.

[157] Gen Li, Nan Duan, Yuejian Fang, Daxin Jiang, and Ming Zhou. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *AAAI*, 2020.

[158] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019.

[159] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clin-

icalBERT: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.

[160] Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. Publicly available clinical BERT embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

[161] Zongcheng Ji, Qiang Wei, and Hua Xu. BERT-based ranking for biomedical entity normalization. *arXiv preprint arXiv:1908.03548*, 2019.

[162] Matthew Tang, Priyanka Gandhi, Md Ahsanul Kabir, Christopher Zou, Jordyn Blakey, and Xiao Luo. Progress notes classification and keyword extraction using attention-based deep learning models with BERT. *arXiv preprint arXiv:1910.05786*, 2019.

[163] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777*, 2019.

[164] Shaolei Wang, Wanxiang Che, Qi Liu, Pengda Qin, Ting Liu, and William Yang Wang. Multi-task self-supervised learning for disfluency detection. In *AAAI*, 2019.

[165] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *KDD*, pages 535–541, 2006.

[166] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. Compressing large-scale transformer-based models: A case study on BERT. *arXiv preprint arXiv:2002.11985*, 2020.

[167] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *ICCV*, pages 293–302, 2019.

[168] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[169] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In *EMNLP-IJCNLP*, pages 4323–4332, 2019.

[170] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: The impact of student initialization on knowledge distillation. *arXiv preprint arXiv:1908.08962*, 2019.

[171] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT: a compact task-agnostic BERT for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

[172] Sanqiang Zhao, Raghav Gupta, Yang Song, and Denny Zhou. Extreme language model compression with optimal subwords and shared projections. *arXiv preprint arXiv:1909.11687*, 2019.

[173] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *arXiv preprint arXiv:2002.12327*, 2020.

[174] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *NeurIPS*, pages 14014–14024, 2019.

[175] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, pages 5797–5808, 2019.

[176] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *ICLR*, 2019.

[177] Wenhao Lu, Jian Jiao, and Ruofei Zhang. TwinBERT: Distilling knowledge to twin-structured BERT models for efficient retrieval. *arXiv preprint arXiv:2002.06275*, 2020.

[178] Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. Small and practical BERT models for sequence labeling. In *EMNLP-IJCNLP*, pages 3632–3636, 2019.

[179] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*, 2019.

[180] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from BERT into simple neural networks. *arXiv preprint arXiv:1903.12136*, 2019.

[181] Yew Ken Chia, Sam Witteveen, and Martin Andrews. Transformer to CNN: Label-scarce distillation for efficient text classification. *arXiv preprint arXiv:1909.03508*, 2019.

[182] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 2464–2469. IEEE, 2016.

[183] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3301–3310. PMLR, 2019.

[184] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[185] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[186] Yijin Liu, Fandong Meng, Jie Zhou, Yufeng Chen, and Jinan Xu. Faster depth-adaptive transformers. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational*

Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, pages 13424–13432. AAAI Press, 2021.

[187] Keli Xie, Siyuan Lu, Meiqi Wang, and Zhongfeng Wang. Elbert: Fast albert with confidence-window based early exit. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7713–7717. IEEE, 2021.

[188] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. doi: https://doi.org/10.1109/TKDE.2009.191.

[189] Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. What do neural machine translation models learn about morphology? In *ACL*, pages 861–872, 2017.

[190] Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*, pages 7–14, 2019.

[191] Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Searching for effective neural extractive summarization: What works and what's next. In *ACL*, pages 1049–1058, 2019.

[192] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. Incorporating BERT into neural machine translation. In *ICLR*, 2020.

[193] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

[194] Jason Phang, Thibault Févry, and Samuel R Bowman. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018.

[195] Siddhant Garg, Thuy Vu, and Alessandro Moschitti. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In *AAAI*, 2019.

[196] Yige Xu, Xipeng Qiu, Ligao Zhou, and Xuanjing Huang. Improving BERT fine-tuning via self-ensemble and self-distillation. *arXiv preprint arXiv:2002.10345*, 2020.

[197] Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. An embarrassingly simple approach for transfer learning from pretrained language models. In *NAACL-HLT*, pages 2089–2095, 2019.

[198] Xiang Lisa Li and Jason Eisner. Specializing word embeddings (for parsing) by information bottleneck. In *EMNLP-IJCNLP*, pages 2744–2754, 2019.

[199] Teven Le Scao and Alexander M. Rush. 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, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2627–2636. Association for Computational Linguistics, 2021.

[200] Timo Schick and Hinrich Schütze. 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, EACL 2021, Online, April 19 - 23, 2021*, pages 255–269. Association for Computational Linguistics, 2021.

[201] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438, 2020.

[202] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *NAACL-HLT*, 2019.

[203] Guanghui Qin and Jason Eisner. 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, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5203–5212. Association for Computational Linguistics, 2021.

[204] Zexuan Zhong, Dan Friedman, and Danqi Chen. 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, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5017–5033. Association for Computational Linguistics, 2021.

[205] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[206] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

[207] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[208] Jesse Vig. A multiscale visualization of attention in the transformer model. In *ACL*, 2019.

[209] Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. exbert: A visual analysis tool to explore learned representations in transformers models. *arXiv preprint arXiv:1910.05276*, 2019.

[210] Ziqing Yang, Yiming Cui, Zhipeng Chen, Wanxiang Che, Ting Liu, Shijin Wang, and Guoping Hu. Textbrewer: An open-source knowledge distillation toolkit for natural language processing. *arXiv preprint arXiv:2002.12620*, 2020.

[211] Yuxuan Wang, Yutai Hou, Wanxiang Che, and Ting Liu. From static to dynamic word representations: a survey. *International Journal of Machine Learning and Cybernetics*, pages 1–20, 2020. doi: https://doi.org/10.1007/s13042-020-01069-8.

[212] Qi Liu, Matt J Kusner, and Phil Blunsom. A survey on contextual embeddings. *arXiv preprint arXiv:2003.07278*, 2020.

[213] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

[214] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, pages 3261–3275, 2019.

[215] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *EMNLP*, pages 2383–2392, 2016.

[216] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *TACL*, 7:249–266, 2019. doi: https://doi.org/10.1162/tacl_a_00266.

[217] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380, 2018.

[218] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension. *arXiv preprint arXiv:2001.09694*, 2020.

[219] Ying Ju, Fubang Zhao, Shijie Chen, Bowen Zheng, Xuefeng Yang, and Yunfeng Liu. Technical report on conversational question answering. *arXiv preprint arXiv:1909.10772*, 2019.

[220] Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *AAAI*, 2020.

[221] Enkhbold Bataa and Joshua Wu. An investigation of transfer learning-based sentiment analysis in japanese. In *ACL*, 2019.

[222] Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL-HLT*, 2019.

[223] Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. Adapt or get left behind: Domain adaptation through BERT language model finetuning for aspect-target sentiment classification. *arXiv preprint arXiv:1908.11860*, 2019.

[224] Akbar Karimi, Leonardo Rossi, Andrea Prati, and Katharina Full. Adversarial training for aspect-based sentiment analysis with BERT. *arXiv preprint arXiv:2001.11316*, 2020.

[225] Youwei Song, Jiahai Wang, Zhiwei Liang, Zhiyue Liu, and Tao Jiang. Utilizing BERT intermediate layers for aspect based sentiment analysis and natural language inference. *arXiv preprint arXiv:2002.04815*, 2020.

[226] Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. Exploiting BERT for end-to-end aspect-based sentiment analysis. In *W-NUT@EMNLP*, 2019.

[227] Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. "mask and infill" : Applying masked language model to sentiment transfer. In *IJCAI*, 2019.

[228] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In *ACL*, pages 1756–1765, 2017.

[229] Liyuan Liu, Xiang Ren, Jingbo Shang, Xiaotao Gu, Jian Peng, and Jiawei Han. Efficient contextualized representation: Language model pruning for sequence labeling. In *EMNLP*, pages 1215–1225, 2018.

[230] Kai Hakala and Sampo Pyysalo. Biomedical named entity recognition with multilingual BERT. In *BioNLP Open Shared Tasks@EMNLP*, pages 56–61, 2019.

[231] Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *NAACL-HLT*, pages 4052–4059, 2019.

[232] Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. On the use of BERT for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, Hong Kong, 2019.

[233] Kenji Imamura and Eiichiro Sumita. Recycling a pre-trained BERT encoder for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, Hong Kong, November 2019.

[234] Xingxing Zhang, Furu Wei, and Ming Zhou. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *ACL*, pages 5059–5069, 2019.

[235] Yang Liu and Mirella Lapata. Text summarization with pre-trained encoders. In *EMNLP/IJCNLP*, 2019.

[236] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuan-Jing Huang. Extractive summarization as text matching. In *ACL*, 2020.

[237] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? natural language attack on text classification and entailment. In *AAAI*, 2019.

[238] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. In *EMNLP-IJCNLP*, pages 2153–2162, 2019.

[239] Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. Adv-BERT: BERT is not robust on misspellings! generating nature adversarial samples on BERT. *arXiv preprint arXiv:2003.04985*, 2020.

[240] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: Adversarial attack against BERT using BERT. *arXiv preprint arXiv:2004.09984*, 2020.

[241] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. FreeLB: Enhanced adversarial training for natural language understanding. In *ICLR*, 2020.

[242] Xiulei Liu, Hao Cheng, Peng cheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. Adversarial

training for large neural language models. *arXiv preprint arXiv:2004.08994*, 2020.

[243] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[244] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988, 2019.

[245] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

[246] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[247] Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. Conditional BERT contextual augmentation. In *International Conference on Computational Science*, pages 84–95, 2019.

[248] Varun Kumar, Ashutosh Choudhary, and Eunah Cho. Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*, 2020.

[249] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[250] Sarthak Jain and Byron C Wallace. Attention is not explanation. In *NAACL-HLT*, pages 3543–3556, 2019.

[251] Sofia Serrano and Noah A Smith. Is attention interpretable? In *ACL*, pages 2931–2951, 2019.