AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing

Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha

Abstract—Transformer-based pretrained language models (T-PTLMs) have achieved great success in almost every NLP task. The evolution of these models started with GPT and BERT. These models are built on the top of transformers, self-supervised learning and transfer learning. Transformed-based PTLMs learn universal language representations from large volumes of text data using self-supervised learning and transfer this knowledge to downstream tasks. These models provide good background knowledge to downstream tasks which avoids training of downstream models from scratch. In this comprehensive survey paper, we initially give a brief overview of self-supervised learning. Next, we explain various core concepts like pretraining, pretraining methods, pretraining tasks, embeddings and downstream adaptation methods. Next, we present a new taxonomy of T-PTLMs and then give brief overview of various benchmarks including both intrinsic and extrinsic. We present a summary of various useful libraries to work with T-PTLMs. Finally, we highlight some of the future research directions which will further improve these models. We strongly believe that this comprehensive survey paper will serve as a good reference to learn the core concepts as well as to stay updated with the recent happenings in T-PTLMs. The list of T-PTLMs along with links is available at **https://mr-nlp.github.io/posts/2021/05/tptlms-list/**

Index Terms—Self-Supervised Learning, Transformers, Pretrained Language Models, Survey.

CONTENTS

- *K.S.Kalyan is with the Department of Computer Applications, National Institute of Technology Trichy, Trichy, Tamil Nadu, India, 620015. E-mail: kalyan.ks@yahoo.com, Website: https://mr-nlp.github.io*
- *Ajit Rajasekharan is with the Nference.ai as CTO, Cambridge, MA, USA, 02142.*
- *S.Sangeetha is with the Department of Computer Applications, National Institute of Technology Trichy, Trichy, Tamil Nadu, India, 620015..*

Preprint under review - The paper is named (AMMUS - AMMU Smiles) in the memory of one of the close friends of K.S.Kalyan (https://mr-nlp.github.io).

1 Introduction

Transformer-based pretrained language models (T-PTLMs) like GPT-1 [1], BERT [2], XLNet [3], RoBERTa [4], ELECTRA [5], T5 [6], ALBERT [7], BART [8] and PEGAUSUS [9] have achieved tremendous success in NLP because of their ability to learn universal language representations from large volumes of unlabeled text data and then transfer this knowledge to downstream tasks. In the early days, NLP systems are mostly rule-based which are later replaced by machine-learned models. Machine learning models require feature engineering which requires domain expertise and it is a time-consuming process too. The evolution of better computer hardware like GPUs and word embeddings like Word2Vec [10] and Glove [11] increased the use of deep learning models like CNN [12] and RNN [13], [14] for building NLP systems. The main drawback with these deep learning models is the requirement of training the model from scratch except for the word embeddings. Training the model from scratch requires a large number of labeled instances which are expensive to generate. However, we expect the model to perform well using few labeled instances only. Transfer learning [15] allows the reuse of knowledge learned in source tasks to perform well in the target task. Here the target task should be similar to the source task. Based on the idea of transfer learning, researchers in Computer Vision trained large CNN models [16]–[19] using large scale labeled datasets like ImageNet [20], [21]. These models learn image representations which are common across all the tasks. The large pretrained CNN models are adapted to downstream tasks by including few task-specific layers and then fine-tuned on the target datasets [22]. As the pretrained CNN models provide good background knowledge to the downstream models, they enjoyed tremendous success in many CV tasks [18], [23].

Deep learning models like CNN and RNN have difficulties in modelling long term contexts and learn the word representations with locality bias [24]. Moreover, as RNNs process the input sequentially i.e., word by word, the utilization of parallel computer hardware is limited. To overcome these drawbacks in existing deep learning models, Vaswani et al. [25] proposed a deep learning model called Transformers which is completely based on self-attention. Self-attention allows for more parallelization compared to RNNs and can easily model long term contexts as every token attend to all the tokens in the input sequence [25]. Transformers contains a stack of encoder and decoder layers. With the help of a stack of encoder and decoder layers, transformers can learn complex language information. It is a very expensive and time-taking process to generate a large amount of labeled data in the NLP domain. However, it is very easy to get large volumes of unlabeled text data. NLP research community impressed with the success of CNN-based pretrained models in Computer Vision, have developed T-PTLMs by combining the power of transformers and self-supervised learning. Self-supervised learning allows the transformers to learn based on the pseudo supervision provided by one or more pretraining tasks.

GPT and BERT are the first T-PTLMs developed based on transformer decoder and encoder layers respectively. Following GPT and BERT, models like XLNet , RoBERTa, ELECTRA, ALBERT, T5, BART and PEGAUSUS are proposed. Here XLNet, RoBERTa, ELECTRA and ALBERT are improvements over BERT model while T5, BART and PEGAUSUS are encoder-decoder based models. Kaplan et al. [26] showed that the performance of T-PTLMs can be increased just by increasing the size of the model. This observation triggered the development of large-scale T-PTLMs like GPT-3 (175B) [27], PANGU- (200B) [28], GShard (600B) [29] which contain billions of parameters and Switch-Transformers (1.6T) [30] which contains trillions of parameters. Following the success of T-PTLMs in general English domain, T-PTLMs are also developed for other domains like Finance [31], Legal [32], [33], News

[34], Programming [35]–[39], Dialogue [40], Networking [41], Academic [42]–[44] and Biomedical [45]–[48]. T-PTLMs support transfer-learning also as these models can be adapted to downstream tasks by fine-tuning or prompt-tuning on target datasets. In this survey paper, we present a comprehensive review of recent research works related to T-PTLMs. We summarize the highlights of our survey as

- We present a brief overview of SSL, the backbone behind developing T-PTLMs (Section 2).
- We explain various core concepts related to T-PTLMs like pretraining, pretraining methods, pretraining tasks, embeddings and downstream adaptation methods (Section 3).
- We present a new taxonomy to categorize various T-PTLMs. This taxonomy is based on four perspectives namely pretraining corpus, architecture, type of SSL and extensions (Section 4).
- We present a new taxonomy to categorize various downstream adaptation methods and explain each in detail (Section 5).
- We present a brief overview of various benchmarks including both intrinsic and extrinsic which evaluate the progress of T-PTLMs (Section 6).
- We present a brief overview of various libraries starting from Huggingface Transformers to Transformer-interpret which are useful to work T-PTLMs (Section 7).
- We briefly discuss some of the future research directions which will drive the research community to further improve the models (Section 8).

2 Self-Supervised Learning (SSL)

Self-supervised learning, a relatively new learning paradigm has gained attention in the Artificial Intelligence (AI) research community due to its ability to make use of unlabeled data to inject universal knowledge about language, image or speech into pretrained models. Due to its data efficiency and generalization ability, SSL finds applications in various AI fields like Robotics [49], Speech [50], [51], Natural Language Processing [24], [52] and Computer Vision [53], [54].

2.1 Why Self-Supervised Learning?

Supervised learning has played a crucial part in AI progress by allowing the models to learn from human-annotated instances. Models trained using supervised learning over labeled instances perform well on a specific task. However, a model trained using supervised learning requires a large number of labeled instances to achieve good performance. Data collection and labelling is a time-taking and expensive process. Moreover, it is difficult to obtain labeled data in specific domains like Medical and Legal. Further, the model learns only what is available in the training data and suffers from generalization error and spurious correlations. Although supervised learning is a dominant learning paradigm in developing AI models in the last two decades, the bottlenecks in supervised learning have forced the research community to look for alternative learning paradigms like Self-Supervised Learning (SSL). SSL does not require human labeled data and helps the model to gain more generalization ability by learning from large amounts of unlabeled data. We summarize the drawbacks of supervised learning as

- heavy dependence on human labeled instances which are expensive and time-consuming to generate.
- lack of generalization ability and suffers from spurious correlations.
- many domains like Medical and Legal are labeled data starved which limits the application of AI models in these domains.
- inability to learn from large amount of freely available unlabeled data.

2.2 What is Self-Supervised Learning?

Self-Supervised Learning (SSL) is a new learning paradigm which helps the model to learn universal knowledge based on the pseudo supervision provided by pretraining tasks. In SSL, the labels are automatically generated based on data attributes and the definition of pretraining task. Let $X = (x_1, p_1), (x_2, p_2), (x_3, p_3), \ldots, (x_n, p_n)$ represents pseudo labeled instances. The pretraining loss (L_{SSL}) of SSL learning paradigm can be defined as

$$L_{SSL} = \lambda_1 L_{PT-1} + \lambda_2 L_{PT-2} + ... + \lambda_m L_{PT-m} \quad (1)$$

Here $L_{PT-1}()$, $L_{PT-2}()$,...,L_{PT-m} represent the loss functions of 'm' pretraining tasks and $\lambda_1()$, $\lambda_2()$,...,$\lambda_m()$ represents weights. In general, pretraining using SSL paradigm can involve more than one pretraining task. For example, RoBERTa is pretrained using only masked language modelling (MLM) while BERT model is pretrained using two pretraining tasks namely masked language modelling (MLM) and next sentence prediction (NSP). In case of MLM, the loss function used is cross entropy loss and in case of NSP, it is sigmoid loss. By solving the pretraining tasks over vast amount of unlabeled data, the model learns general language representations which can encode both syntax and semantic information. These representations are useful in downstream tasks and helps the model to achieve much better performance using few labeled instances only. We can say that pretraining over vast amount of unlabeled data using SSL helps the model to gain basic common sense or background knowledge without which the model requires more labeled instances to achieve a good performance.

SSL has similarities with other popular learning paradigms like supervised and unsupervised learning. SSL is like unsupervised learning as it does not require human labeled instances. However, it is different from

unsupervised learning because a) SSL requires supervision unlike unsupervised learning and b) the objective of unsupervised learning is to identify the hidden patterns while the objective of SSL is to learn meaningful representations. SSL is like supervised learning as both the learning paradigms require supervision. However, it is different from supervised learning because a) SSL generates labels automatically without any human involvement and b) the goal of supervised learning is provide task specific knowledge while SSL aims to provide the model with universal knowledge. We summarize the goals of SSL as

- learn universal language representations which provides a good background to the downstream model.
- better generalization ability by learning over vast amount of freely available unlabeled text data.

2.3 Types of Self-Supervised Learning

Self-Supervised Learning can be classified into Generative SSL, Contrastive SSL and Adversarial SSL . Generative SSL allows the model to learn by decoding the encoded input. Generative SSL can use autoregressive, autoencoding or hybrid language models. Autoregressive language model predicts the next tokens based on the previous tokens. GPT-1 [1] is the first PTLM that is based on the autoregressive language model. Autoencoding language model predicts the masked tokens based on the unmasked tokens (bidirectional context). For example, masked language modelling (MLM) involves two steps. The first step is to encode the masked tokens using bidirectional context and the second step is to decode (predict) the original tokens based on the encoded masked token representations. Models like BERT [2], RoBERTa [4] and ALBERT [7] are pretrained using MLM. Hybrid language models combine the advantages of autoregressive and autoencoding language models. For example, permutation language modelling (PLM) in XLNet [3] is an example of a hybrid language model.

Contrastive SSL allows the model to learn by comparing. Next sentence prediction (NSP) in BERT and sentence order prediction in ALBERT are examples of contrastive SSL. NSP involves identifying whether the given sentence pair includes consecutive sentences or not, while SOP involves identifying whether the given pair includes swapped sentences or not. Adversarial SSL allows the model to learn by identifying whether the tokens in the input sentence are replaced or shuffled or randomly substituted. Replaced token detection (RTD) in ELECTRA [5], shuffled token detection (STD) [55] and random token substitution (RTS) [56] are examples of Adversarial SSL. For detailed information about SSL and types, please refer to the survey paper on SSL [49].

3 T-PTLM Core Concepts

3.1 Pretraining

Pretraining on large volumes of unlabeled text and then fine-tuning on small task-specific datasets has become a standard approach in modern natural language processing. In Computer Vision, large models [16]–[19] based on CNN are pretrained on large, labeled datasets like ImageNet [20], [21], and then these models are used in similar target tasks by adding few task-specific layers [22]. Here pretraining allows the model to learn common image features which are useful in many tasks. Inspired by the success of pretrained image models, NLP researchers developed models like BERT [2], RoBERTa [4], ELECTRA [5], XLNet [3], and T5 [6] by pretraining them on large volumes of unlabelled text using self-supervised learning. Some of the benefits of pretraining are

- It helps the model to learn universal language representations by leveraging large volumes of unlabeled text.
- Pretrained models can be adapted to downstream tasks by just adding one or two specific layers. Hence it avoids training the downstream model (except task-specific layers) from scratch by providing a good initialization.
- It helps the model to perform better even with small datasets and hence reduces the requirement of a large number of labeled instances.
- Deep learning models due to having a large number of parameters tend to overfit on small datasets. As pretraining provides a good initialization, it avoids overfitting on small datasets, and hence pretraining can be viewed as a form of regularization [57].

3.1.1 *Pretraining Steps*

Pretraining a model involves the following five steps

1. *Prepare the pretraining corpus* – Pretraining corpus is obtained from one or more sources of unlabelled text and then cleaned. BERT [2] model is pretrained on English Wikipedia and BooksCorpus. Further research [3], [4], [6] showed that pretraining the model on a much larger text corpus obtained from multiple sources further improves the performance of the model. Moreover, Lee et al. [58] showed there is a lot of redundancy in pretraining corpus in the form of near-duplicate sentences and long repetitive substrings. Further, Lee et al. [58] showed pretraining the model on deduplicated corpus requires fewer training steps to achieve similar performance.

2. *Generate the vocabulary* – Most of the transformer-based pretrained language models use tokenizers like WordPiece [59], Byte Pair Encoding (BPE) [60], Byte Level BPE (bBPE) [61], and SentencePiece [62] to generate the vocabulary. Usually, vocabulary consists of all the unique characters and commonly used subwords and words. Vocabulary is generated by applying any of the tokenizers on the pretraining corpus. Different T-PTLMs use different tokenizers and generate vocabulary with different sizes. For example, BERT uses WordPiece vocabulary of size around 30K, RoBERTa uses bBPE vocabulary of size around 50K, XLM [63] uses BPE vocabulary of size 95K, mBERT [2] WordPiece vocabulary

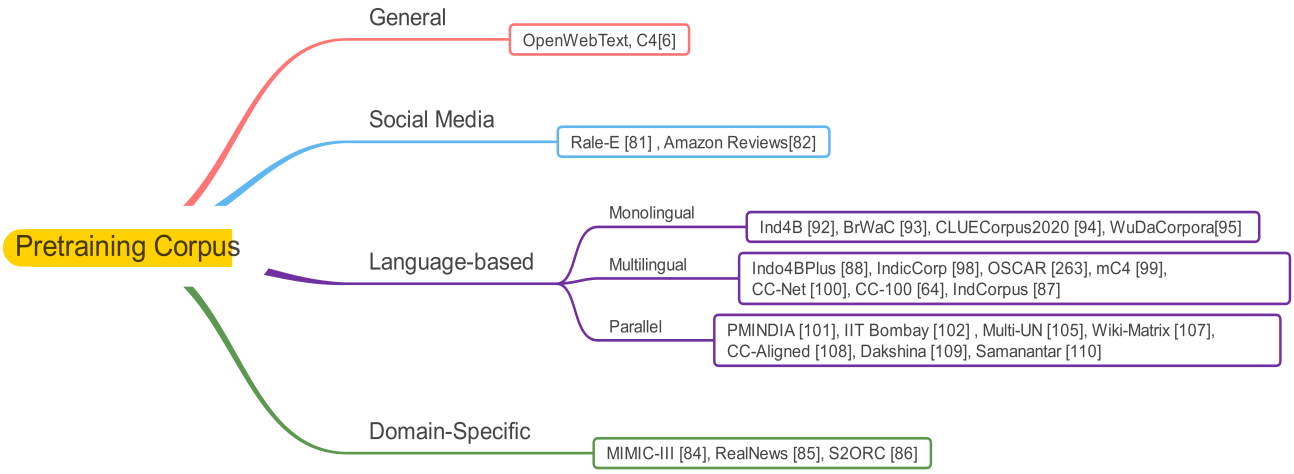

Fig. 1: Pretraining corpus

of size 110K, XLM-R [64], and mBART [65] uses SentencePiece vocabulary of size 250K. The large vocabulary size in multilingual models like XLM, XLM-R, mBERT, and mBART make sense as they have to represent multiple languages. However, the size of the pretrained model increases with an increase in vocabulary size. This step is optional in the case of char-based T-PTLM like CharacterBERT [66] and tokenization-free T-PTLMs like CANINE [67], ByT5 [68], and Charformer [69].

3. *Design the pretraining tasks* - During pretraining, the model learns language representations by minimizing losses based on one or more pretraining tasks. A pretraining task should

- *be challenging enough to allow the model to learn semantics at word, phrase, sentence, or document level*. For example, recent research works [4], [7] questioned the efficiency of NSP task and resulted in new pre-training tasks to learn semantics at sentence level like sentence order prediction [7] and sentence structure prediction [70].
- *provide more training signal so that the model learns more language information with less pretraining corpus*. For example, RTD provides more training signal compared to MLM because RTD is defined over all the input tokens while MLM is defined over a subset of tokens only [5].
- *close to downstream tasks*. For example, span boundary pretraining task in SpanBERT [71] is close to the span extraction task and the gap sentence generation in PEGAUSUS [9] is close to the summarization task. Recent research works resulted in better versions of MLM like Swapped Language Modeling [56] which avoids the use of special mask tokens and hence reduces the discrepancy between pretraining and fine-tuning.

4. *Choose the pretraining method* – Training a new model from scratch using SSL only is highly expensive and consumes a lot of pretraining time. Instead of training from scratch using SSL only, pretraining methods like KIPT [72], [73] which pretrain a model using both SSL and KD can be used. In the case of adapting general models to specific domains, pretraining methods like continual pretraining with new vocabulary [74]–[77] or adapt and distill [78] can be used. To pretrain a domain-specific model with limited domain-specific corpus, simultaneous pretraining which leverages both general and in-domain corpus can be used [79].

5. *Choose the pretraining dynamics* – BERT model is pretrained on sentence pairs with static masking in small batch sizes. Liu et al. [4] showed that carefully designed pretraining choices like dynamic masking, large batch sizes, more pretraining steps, and long input sequences further enhance the performance of the model. Moreover, when using large batch sizes which may cause difficulty in optimization, it is recommended to a) linearly increase the learning rate in the early pretraining steps and b) use different learning rates in different layers which can also help to speed up convergence [80].

3.1.2 Pretraining Corpus

Self-Supervised learning to pretrain T-PTLMs requires large volumes of pretraining data. As shown in Figure, pretraining corpus can be classified into four types (refer Figure 1). The characteristic of the text differs from one type of corpus to another. For example, in the general domain, the text is less noisy and written formally by professionals. In social media, the text is mostly noisy and written colloquially by the general public. Moreover, many specific domains like Biomedical and Finance contain many domain-specific words which are not used in the general domain. In general, the performance of general domain models in domain-specific tasks is limited [45]. So, we have to choose the pretraining corpus depending on the target domain to

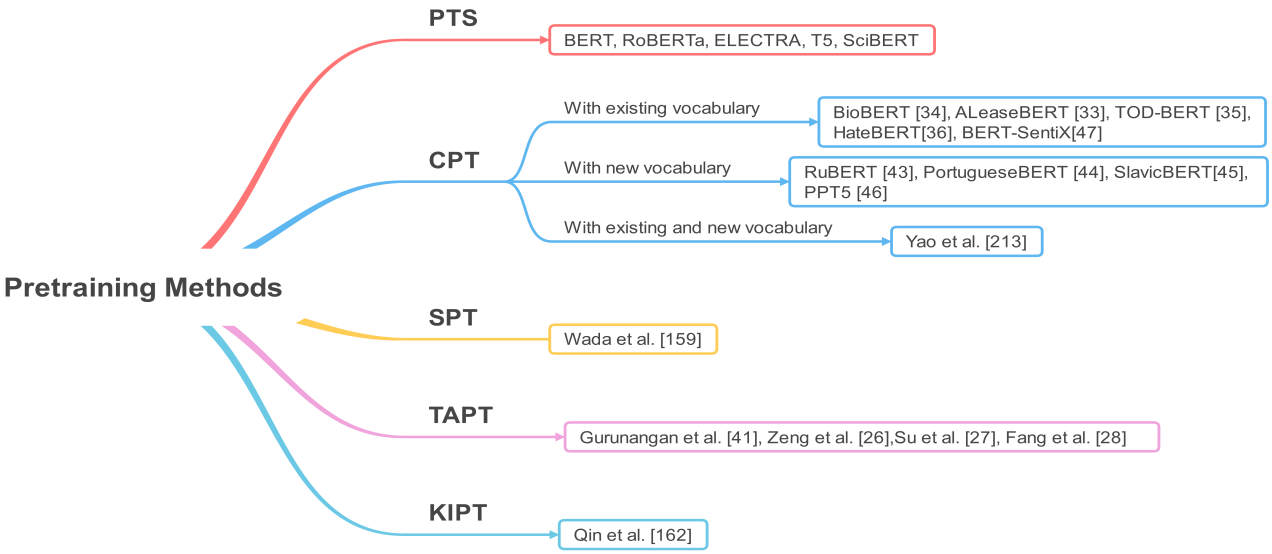

Fig. 2: Pretraining methods

Pretraining Corpus Type		Pretraining Corpus	Description	Models
General	-	OpenWebText	Open source equivalent to the WebText corpus used to pretrain GPT-2 model and it is around 32GB.	RoBERTa [4]
	-	C4 [6]	750GB collection of common crawl text which is deduplicated and filtered to include natural text only.	T5 [6]
Social Media	-	Rale-E [81]	Collection of hateful comments in English which are posted on Reddit, a popular social media platform.	HateBERT [81]
	-	Amazon reviews [82]	Collection of 233M reviews posted by users about various products. The gathered reviews cover around 29 domains.	BERT-SentiX [83]
Domain-Specific	Biomedical	MIMIC-III [84]	Consists of de-identified ICU patient records gathered over more than a decade. It is the largest publicly available corpus of clinical records.	BioBERT [45], BlueBERT [48], ClinicalBERT [46]
	News	RealNews [85]	Common crawl new corpus. It is around 120GB.	Roberta-base-news [34]
	Academic	S2ORC [86]	Large scale collection of more than 80M research papers written in English.	Roberta-base-biomed [34], Roberta-base-cs [34]

TABLE 1. Summary of various pretraining corpora in general, social media and specific domains.

achieve good results. BERT model is pretrained using text from Wikipedia and BookCorpus which amounts to 16GB [2]. Further research works showed that the performance of the model can be increased by using large pretraining datasets [3], [4]. This triggered the development of much larger datasets, especially from the common crawl. For example, C4 data contains around 750GB of text data [6] while CC-100 corpus includes around 2.5TB of text data [64]. Multilingual T-PTLMs like mBERT [2], IndT5 [87], IndoBART [88], and XLM-R [64] are pretrained using only multilingual datasets. Some of the models like XLM [63], XLM-E [89], infoXLM [90], and mT6 [91] are pretrained using both multilingual and parallel datasets. A summary of various pretraining corpora is given in Tables 1 and 2.

Fig. 3: Pretraining from Scratch (PTS)

3.2 Types of Pretraining Methods

Figure 2 shows the classification of pretraining methods into five types.

3.2.1 *Pretraining from Scratch (PTS)*

Models like BERT, RoBERTa , ELECTRA, and T5 are pretrained from scratch on large volumes of unlabeled text (refer Figure 3). Usually, any transformer-based pretrained language model consists of an embedding layer,

Pretraining Corpus Type		Pretraining Corpus	Description	Models
Language-based	Monolingual (Indonesian)	Indo4B [92]	Collection of 23GB of Indonesian text gathered from various public resources including social media platforms. It includes around 3.58 billion words.	IndonesianBERT [92]
	Monolingual (Portuguese)	BrWaC [93]	17.5GB collection of Brazilian Portuguese web text gathered from 3.5 million web pages. It includes around 2.7B words.	PortugueseBERT [75], PTT5 [77]
	Monolingual (Chinese)	CLUECorpus2020 [94]	Collection of 100GB of Chinese common crawl text.	RoBERTa-tiny-clue [94]
	Monolingual (Chinese)	WuDaCorpora [95]	200GB collection of Chinese Web text. It includes around 72B Chinese characters.	Chinese-Transformer-XL [95]
	Multilingual	Indo4Bplus [88]	Includes text from Indo4B corpus for Indonesian and from Wikipedia, CC-100 for Sundanese and Javanese language.	IndoBART [88]
		OSCAR [96]	Large scale collection of common crawl text for around 166 languages.	MuRIL [97]
		IndicCorp [98]	Collection of text from various sources for 12 major Indian languages. It includes around 8.9B words.	IndicBERT [98]
		mC4 [99]	Multi-lingual equivalent of C4. It includes common crawl text for 101 languages.	mT5 [99]
		CC-Net [100]	Large scale collection of common crawl text for more than 100 languages.	mT6 [91]
		CC-100 [64]	Large scale collection (2.5TB) of common crawl text of 100 languages.	XLM-R [64], infoXLM [90], XLM-E [89]
		IndCorpus [87]	Collection of text from Wikipedia and Bible for 11 languages including Indigenous languages. It is around 1.17GB and includes around 5.37M sentences.	IndT5 [87]
	Parallel	PMINDIA [101]	Collection of parallel data gathered from Prime Minister of India website. The corpus includes 56K sentences for each language pair.	MuRIL [97]
		IIT Bombay [102]	Collection of 1.49M English-Hindi parallel sentences.	mT6 [91], XLM [63], infoXLM [90], Unicoder [103], ALM [104], XLM-E [89]
		Multi-UN [105]	Parallel corpus created from UN official documents for six languages.	mT6 [91], XLM [63], infoXLM [90]], Unicoder [103], ALM [104], XNLG [106], XLM-E [89]
		Wiki-Matrix [107]	Parallel data for 85 languages. It includes 135M parallel sentences in 1620 language pairs out of which 34M sentences are aligned with English.	mT6 [91], XLM-E [89]
		CC-Aligned [108]	Parallel corpus of 292 million non-English common crawl document pairs and 100 million English common crawl document pairs.	XLM-E [89]
		Dakshina [109]	Parallel corpus containing 10K sentences for 12 Indian languages. Each sentences consists of sentence in native script sentence and its manually romanized transliteration.	MuRIL [97]
		Samanantar [110]	Includes 49.6M sentences pairs of 12 Indian languages aligned with English. It is the largest publicly available parallel corpus for Indian languages.	-

TABLE 2. Summary of various language-based pretraining corpora.

transformer encoder, or (and) transformer decoder layers. All these layer parameters are randomly initialized and then learned during pretraining by minimizing the losses of one or more pretraining tasks. For example, BERT model is pretrained from scratch using MLM and NSP. Pretraining from scratch is computationally expensive and requires a large number of GPUs or TPUs.

3.2.2 *Continual Pretraining (CPT)*

Models like BioBERT [45], ALeaseBERT [32], TOD-BERT [40], HateBERT [81], infoXLM [90], and XNLG [106] are obtained by initializing from existing pretrained models and then further pretrained. For example, infoXLM is initialized from XLM-R [64] and further pretrained on both monolingual and parallel data, ALeaseBERT is initialized from general ALBERT and further pretrained on lease

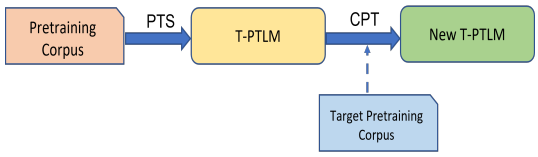

Fig. 4: Continual Pretraining (CPT)

agreements, XLM [63] parameters are used to initialize both encoder and decoder layers in XNLG. Unlike in PTS, in continual pretraining the model parameters are not learned from scratch. Instead, the parameters are initialized with existing language model parameters and then adapted to the target domain by further pretraining (refer Figure 4). Continual pretraining is commonly used to develop T-PTLMs in specific domains like social media [81], [111], [112], biomedical [45], Legal [32], News [34], Computer Networking [41] etc. The main advantage of continual pretraining is that it avoids training a model from scratch and makes use of the existing language model parameters. As CPT starts from existing model parameters, it is less expensive and requires less training time and computational resources compared to PFS.

However, the lack of target domain-specific vocabulary is a drawback in CPT when the target domain consists of many domain-specific words. For example, BioBERT [45] is initialized from general BERT and further pretrained on biomedical text. Though the language model is adapted to the biomedical domain, the vocabulary which is learned over general domain text does not include many of the domain-specific words. As a result, domain-specific words are split into a number of sub-words which hinders model learning and degrades its performance in downstream tasks. Similarly, mBERT accommodates more than 100 languages, the number of tokens in its vocabulary (110K) specific to a language is less. A possible solution for this is continual pretraining with target domain or language-specific vocabulary [75]–[77]. Here, new vocabulary is generated over the target domain or language text. During continual pretraining, the embedding layer is randomly initialized and all other layer parameters are initialized with existing language model parameters. For example, models like RuBERT [74], PortugueseBERT [75], SlavicBERT [76] are initialized from mBERT but further pretrained with language-specific vocabulary. Similarly, PPT5 [77] is initialized from the T5 model but further pretrained with language-specific vocabulary. However, the performance of the model obtained by continual pretraining with new vocabulary is slightly less but on par with the performance of the model trained from scratch. As CPT is computationally less expensive, CPT with new vocabulary can be preferred over PTS in resource-constrained situations. Recently, Yao et al. [78] proposed Adapt and distill approach to adapt general models to a specific domain using vocabulary expansion and knowledge distillation. Different from existing adaptation methods, Adapt and distill approach not only adapt general models to specific domain but also reduces the size of the model.

It is not necessary to use the same set of pretraining tasks used by the existing model for continual pretraining. For example, BERT-SentiX [83] model is initialized from BERT and further pretrained on product reviews using four sentiment-aware pretraining tasks. Similarly, TOD-BERT [40] is initialized from BERT and further pretrained on dialogue corpus text using MLM and response contrastive loss (RCL).

3.2.3 *Simultaneous Pretraining (SPT)*

Domain-specific T-PTLMs can be developed by training from scratch or by continual pretraining. Both these pretraining methods require large volumes of domain-specific unlabelled text to pretrain the model. However, the availability of domain-specific text is limited in many domains. Moreover, domain-specific text in languages other than English is available in small quantities only. For example, in the biomedical domain, MIMIC-III [84] is the largest publicly available (English) medical records dataset. However, it is difficult to obtain such large volumes of medical records in languages like Japanese [79]. PTS or CPT using a small amount of domain-specific text overfits the model. Simultaneous pretraining (SPT) allows the model to pretrain from scratch using a corpus having both general and domain-specific text [79] (refer Figure 5). Here, up sampling of domain-specific text is done to ensure a good number of domain-specific terms in model vocabulary and also to have a balanced pretraining. Wada et al. [79] showed that Japanese clinical BERT pretrained using SPT outperforms Japanese clinical BERT trained from scratch.

Fig. 5: Simultaneous Pretraining (SPT)

3.2.4 *Task Adaptive Pretraining (TAPT)*

Pretraining approaches like PTS, CPT and SPT allow the model to learn universal or domain-specific language representations by training on large volumes of general or domain-specific or combined text. As all these approaches involve training over a large amount of text, these approaches are expensive. Task Adaptive Pretraining (TAPT) allows the model to learn fine-grained task-specific knowledge along with domain-specific knowledge by pretraining on a small amount of task-specific unlabelled text [34] (refer Figure 6). As TAPT requires only a small amount of text, it is less expensive compared to other pretraining methods. Additional task-related sentences can be obtained from large domain corpus using lightweight approaches like VAMPIRE [113] which embeddings all the sentences using a simple bag-of-words language model. Gururangan et al. [34] showed

that TAPT is complementary to other pretraining approaches i.e., PTS / CPT followed by TAPT further improves the performance of the model.

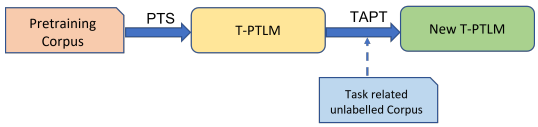

Fig. 6: Task Adaptive Pretraining (TAPT)

3.2.5 *Knowledge Inherited Pretraining (KIPT)*

All the previously discussed pretraining methods like PTS, CPT, SPT, and TAPT solely depend on self-supervised learning to pretrain the models. It is highly expensive and time-consuming to pretrain a large model from scratch using SSL only. In general, humans learn not only learn through self-learning but also learn from other knowledgeable people. Inspired from this, Qin et al. [72] proposed Knowledge Inherited Pretraining (KIPT), a novel pretraining method which pretrains the model using both self-supervised learning and knowledge distillation (refer Figure 7) . KIPT allows reusing the knowledge available in existing pretrained models to pretrain a new model.

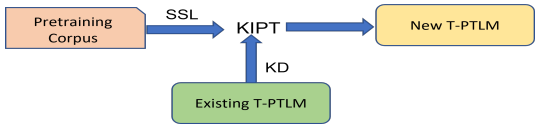

Fig. 7: Knowledge Inherited Pretraining (KIPT)

$$L_{KIPT} = \sigma * L_{SSL} + (1 - \sigma) * L_{KD} \tag{2}$$

where L_{KIPT} represents the overall loss of KIPT, L_{SSL} and L_{KD} represents losses of self-supervised learning and knowledge distillation. KIPT is similar to KD in reusing the knowledge from existing models. However, it is different from KD in two aspects, (a) in KD, generally the student model is compact in size compared to teacher model whereas in KIPT the student model is larger in size compared to teacher model (b) in KD, the student model solely learns from teacher model where as in KIPT the student model encodes the knowledge available in pretraining corpus using self-supervised learning in addition to the knowledge from teacher model. By learning from a knowledgeable teacher model along with self-supervised learning, the model learns more as well as converges faster which makes KIPT more effective and less expensive compared to the pretraining methods which involves only self-supervised learning. Due to the additional knowledge gained from a knowledgeable teacher model, the models trained using KIPT outperforms models trained using self-supervised learning only [72]. Further, Qin et al. [72] showed that KIPT supports both life-long learning and knowledge transfer. CPM-2 [73] is the first large scale pretrained language model pretrained using knowledge inheritance.

3.3 Pretraining Tasks

SSL allows T-PTLMs to learn universal language representations by solving one or more predefined tasks. These tasks are referred to as "pretext" or "pretraining" tasks. Pretraining tasks are self-supervised i.e., these tasks make use of pseudo labeled data. The data attributes and pretraining task definition determine the pseudo labels. A pretraining task should be challenging enough so that it provides more training signals to the model. For example, tasks like MLM involves only 15% of tokens in each training sample for learning while tasks like Replaced Token Detection (RTD) [5], Random Token Substitution (RTS) [56], and Shuffled Token Detection (STD) [55] involves all the tokens in the input sample for model learning. Moreover, a pretraining task should be similar to the downstream task. For example, pretraining tasks like Seq2SeqLM [114] or Denoising Auto Encoder (DAE) [8] are similar to downstream tasks like text summarization, machine translation, etc.

Casual Language Modeling (CLM) - CLM or simply Unidirectional LM predicts the next word based on the context. The unidirectional LM can handle the sequence from left-to-right or right-to-left. In let-to-right LM, the context includes all the words on the left side while in right-to-left LM, the context includes all the words on the right side. GPT-1 [1] is the first transformer-based PTLM to use CLM (left-to-right) as a pretraining task. UniLM [115] uses both left-to-right and right-to-left CLM as pretraining tasks. Let $x = \{x_1, x_2, x_3, \ldots, x_{|x|}\}$ represents a sequence where $|x|$ represents the number of tokens in the sequence. CLM loss is defined as

$$L_{CLM}^{(x)} = -\frac{1}{|x|} \sum_{i=1}^{|x|} logP(x_i/x_{<i}) \tag{3}$$

where $x_{<i} = x_1, x_2, x_3, ..x_{i-1}$.

Masked Language Modeling (MLM) – The main drawback in CLM is the inability to leverage both contexts. Bidirectional contextual information is much better compared to unidirectional context information for encoding token representations. It is not possible to train standard CLM using bidirectional context as it would allow a token to see itself which makes the prediction trivial. MLM is an improved version of CLM to leverage tokens from both contexts. In MLM, we feed the masked token vectors to the softmax layer to get the probability distribution over the vocabulary and then use the cross-entropy loss. BERT is the first model to use MLM as pretraining task [2]. The authors of BERT masked the tokens at a probability of 0.15. Let $x_{\setminus M_x}$ represents the masked version of x and M_x represents the set of masked token positions in x. MLM loss is

defined as

$$L_{MLM}^{(x)} = -\frac{1}{|M_x|}\sum_{i \in M_x} logP(x_i/x_{\setminus M_x}) \quad (4)$$

Replaced Token Detection (RTD) - MLM is better than CLM by leveraging bidirectional contextual information. However, MLM has two drawbacks a) provides less training signal – in MLM, the model learns from only 15% of the tokens and b) model see special mask token only during pretraining which results in a discrepancy between pretraining and fine-tuning stages. RTD overcomes these two issues by a novel approach that involves identifying the replaced tokens [5]. MLM corrupts the sentence by using special mask tokens while RTD corrupts the sentence using the output tokens from the generator model trained using MLM objective. MLM involves predicting the original tokens based on masked token vectors while RTD is a token-level binary classification task that involves classifying every token as replaced or not. ELECTRA model is pretrained in two steps. 1)train the generator model using MLM objective and 2)train the discriminator model initialized from a generator using RTD objective. Let $\hat{x}$ is the corrupted version of x. RTD loss is defined as

$$L_{RTD}^{(x)} = -\frac{1}{|\hat{x}|}\sum_{i=1}^{|\hat{x}|} logP(d/\hat{x}_i) \quad (5)$$

where $d \in \{0, 1\}$ represents whether the token is replaced or not.

Shuffled Token Detection (STD) – STD is a token-level discriminative task that involves identifying the shuffled tokens. Similar to RTD, it is sample efficient and avoids discrepancy between pretraining and fine-tuning stages. In STD, the words are shuffled at a probability of 0.15 (this is based on the masking probability used in BERT and RoBERTa models). Panda et al. [55] showed that continual pretraining RoBERTa using STD improves its performance in many of the GLUE tasks and established that STD allows the model to learn more coherent sentence representations. Let $\hat{x}$ is the corrupted version of x. RTD loss is defined a

$$L_{STD}^{(x)} = -\frac{1}{|\hat{x}|}\sum_{i=1}^{|\hat{x}|} logP(d/\hat{x}_i) \quad (6)$$

where $d \in \{0, 1\}$ represents whether the token is shuffled or not.

Random Token Substitution (RTS) – RTD is sample efficient but requires a separate generator to corrupt the input sequence. Training a separate generator model is computationally expensive. To overcome this drawback, Di et al. [56] proposed RTS which involves identifying the randomly substituted tokens. In RTS, 15% of the tokens are randomly substituted with other tokens from the vocabulary. RTS is sample efficient like RTD but does not require any separate generator model to corrupt the input sequence. Di et al.[56] showed that RoBERTa model trained using RTS matches the performance of RoBERTa model trained using MLM while requiring less training time. RTS loss is defined as

$$L_{RTS}^{(x)} = -\frac{1}{|\hat{x}|}\sum_{i=1}^{|\hat{x}|} logP(d/\hat{x}_i) \quad (7)$$

where $d \in \{0, 1\}$ represents whether the token is randomly substituted or not and $\hat{x}$ is obtained by randomly substituting 15% of tokens in x.

Swapped Language Modeling (SLM) – The MLM pretraining task uses a special mask token to corrupt the input sequence. However, the use of this special token results in a discrepancy between pretraining and fine-tuning stages. SLM overcomes this drawback by corrupting the sequence with random tokens from vocabulary at a probability of 0.15 [56]. SLM is similar to MLM by predicting the corrupting tokens but unlike MLM, SLM replaces the tokens with random tokens. SLM is similar to RTS in using the random tokens for corruption but unlike RTS which is sample efficient by involving every token in the input sequence, SLM is not sample efficient as it involves only 15% of input tokens. SLM loss is defined as

$$L_{SLM}^{(x)} = -\frac{1}{|R_x|}\sum_{i \in R_x} logP(x_i/x_{\setminus R_x}) \quad (8)$$

Where R_x represents set of positions of randomly substituted tokens and $x_{\setminus R_x}$ represents the corrupted version of x.

Translation Language Modeling (TLM) – TLM is an extension of MLM to use parallel data in cross-lingual pretraining. XLM [63] is the first cross-lingual model to use TLM as a pretraining task followed by XNLG [106]. TLM is also referred to as cross-lingual MLM (XMLM). Here the input is a pair of sentences (x,y) where x and y are parallel sentences i.e., x is a translation of y. Similar to MLM, tokens from both sentences are randomly masked. As prediction of masked tokens involves context from both the sentences, TLM helps the model to learn cross-lingual mapping. TLM loss is similar to MLM and is defined as

$$L_{MLM}^{(x,y)} = -\frac{1}{|M_x|}\sum_{i \in M_x} logP(x_i/x_{\setminus M_x}, y_{\setminus M_y}) - \frac{1}{|M_y|}\sum_{i \in M_y} logP(y_i/x_{\setminus M_x}, y_{\setminus M_y}) \quad (9)$$

Where M_x and M_y represents the set of masked positions in the sentences x and y respectively, $x_{\setminus M_y}$ and $y_{\setminus M_y}$ represent the masked version of x and y respectively.

Alternate Language Modeling (ALM): ALM is a pretraining task to train cross-lingual language models. ALM involves predicting the masked tokens in the code-switched sentences generated from parallel sentences [104]. For a given parallel sentence pair (x,y), a code-switched sentence is generated by randomly substituting some phrases of x with their translations from y. ALM

follows the same settings of standard MLM for masking the tokens. By pretraining the model on code-switched sentences, the model learns relationships between languages in a much better way. Yang et al. [104] showed that cross-lingual model pretrained using ALM outperforms XLM which shows that ALM is a better alternative to TLM for pretraining cross-lingual language models. ALM loss is defined as

$$L_{ALM}^{(z(x,y))} = -\frac{1}{|M|}\sum_{i \in M} logP(z_i/Z_{\setminus M}) \tag{10}$$

Where z is the code-switched sentence generated from x and y, $z_{\setminus M}$ represents the masked version of z and M represents the set of masked token positions in $z_{\setminus M}$.

Sentence Boundary Objective (SBO) – SBO pretraining task involves predicting the masked tokens based on the span boundary tokens and position embeddings [71]. SBO is similar to MLM in predicting the masked tokens. However, it is different from MLM in three aspects (a) SBO masks only contiguous span of tokens while MLM masks tokens randomly and (b) in SBO, the prediction of masked tokens involves span boundary tokens and position embeddings while in MLM, the prediction of masked tokens involves only the masked token vectors. Moreover, SBO is much challenging compared to standard MLM as it is difficult to predict the entire span "an American football game" compared to predicting "game" when "an American football" is already known [71]. SBO helps the model to perform better in downstream tasks like entity extraction, coreference resolution, and question answering which involves span-based extraction. SBO loss is defined as

$$L_{SBO}^{(x))} = -\frac{1}{|S|}\sum_{i \in S} logP(x_i/y_i) \tag{11}$$

where $y_i = f(x_{s-1}, x_{e+1}, p_{s-e+1})$ and $f()$ is a two-layered feedforward neural network, S represents the positions of tokens in contiguous span, $|S|$ represents the length of span, s and e represent the start and end positions of span, p represents the position embedding.

Next Sentence Prediction (NSP) – NSP is a sentence-level pretraining task that helps the model to learn relationships between sentences [2]. It is a binary sentence pair classification task that involves identifying consecutive sentences. Here the aggregate representation of the two sentences (x,y) i.e., [CLS] token vector is given to the sigmoid layer to get the probability. For training, the sentence pairs are generated in a way that 50% of instances are consecutive and the rest are not consecutive. Pretraining the model at the sentence level is useful in downstream tasks like question answering, NLI, and STS which involve sentence pair input. NSP loss is defined as

$$L_{NSP}^{(x,y)} = -logP(d/x, y) \tag{12}$$

Where $d \in \{1, 0\}$ represents whether the sentences are consecutive or not.

Sentence Order Prediction (SOP) – NSP allows the model to learn sentence-level semantics and involves both topic and coherence prediction. As topic prediction is easier, the effectiveness of NSP is questioned [4], [7]. SOP is a sentence-level pretraining task based on sentence coherence only. ALBERT [7] is the first pretraining model to use SOP as a pretraining task. It involves identifying whether the given sentences are swapped or not. Following NSP, the training instances are generated in a way that 50% of instances are swapped and the rest or not. SOP loss is defined as

$$L_{SOP}^{(x,y)} = -logP(d/x, y) \tag{13}$$

Where $d \in \{1, 0\}$ represents whether the sentences are swapped or not.

Sequence-to-Sequence LM (Seq2SeqLM) - MLM is approached as a token level classification task over the masked tokens i.e., original words are predicted by feeding the masked token vectors to a softmax layer over the vocabulary. Seq2SeqLM is an extension of standard MLM to pretrain encoder-decoder-based models like T5 [6], mT5 [99] and MASS [114]. In the case of MLM, the context includes all the tokens in the input sequence whereas in Seq2SeqLM, the context includes all the words in the input masked sequence and the left side words in the predicted target sequence. With masked sequence as input to the encoder, the decoder predicts the masked words from left to right sequentially. Seq2SeqLM loss is defined as

$$L_{Seq2SeqLM}^{(x)} = -\frac{1}{l_s}\sum_{s=i}^{j} logP(x_s/\hat{x}, x_{i:s-1}) \tag{14}$$

where $\hat{x}$ is the masked version of x i.e., x with masked n-gram span, l_s represents the length of masked n-gram span.

Denoising Auto Encoder (DAE): DAE helps to model to learn by reconstructing the original text from corrupted text [8]. The text can be corrupted at token (e.g., token deletion and token masking), phrase (e.g., token infilling), sentence (e.g., sentence permutation), or document level (e.g., document rotation). Like Seq2SeqLM, DAE is useful to train encoder-decoder-based models. However, DAE is more sample efficient by providing more training signals for model learning. DAE provides more training signal as it involves reconstructing entire original text while Seq2SeqLM involves reconstructing the masked tokens only. BART [8] uses a bidirectional encoder to encode corrupted input sequence and a left-to-right decoder to recreate the original text. The authors of BART experimented with various corruption strategies and finally trained the model on the sentences corrupted using sentence permutation and text infilling. DAE loss is defined as

$$L_{DAE}^{(x)} = -\frac{1}{|x|}\sum_{i=1}^{|x|} logP(x_i/\hat{x}, x_{<i}) \tag{15}$$

where $\hat{x}$ is the corrupted version of x.

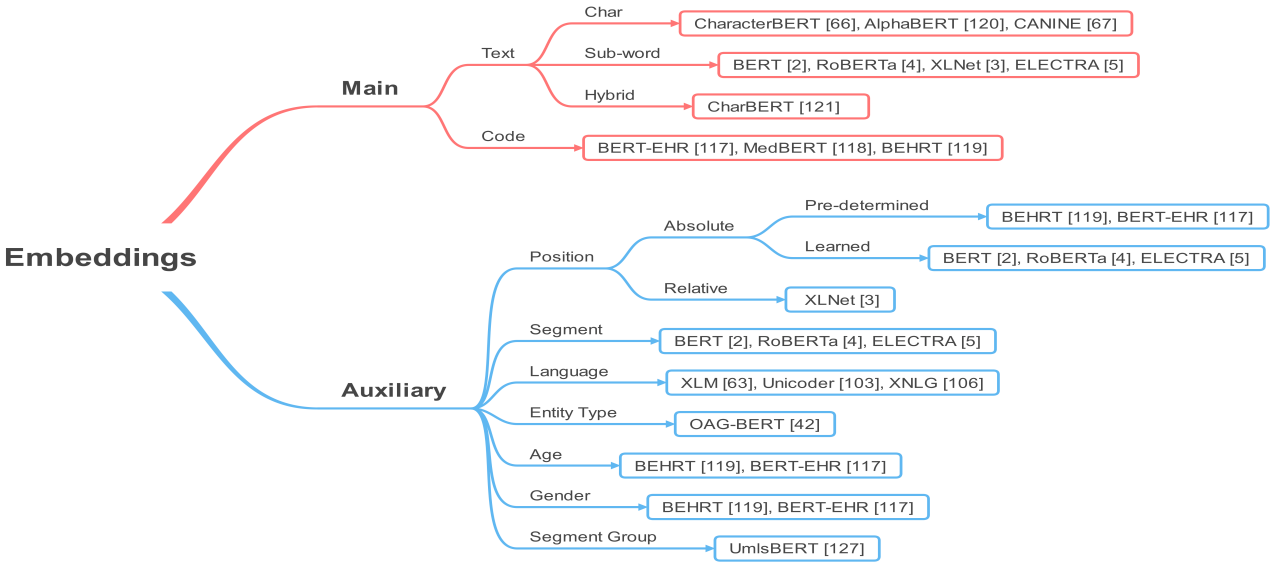

Fig. 8: Embeddings in T-PTLMs

3.4 Embeddings

Deep learning models expect the input in the form of a matrix of numbers and then apply a sequence of matrix operations. As deep learning models including transformers expect numerical input, input text should be mapped to a sequence of dense, low dimensional vectors (commonly called embeddings in natural language processing). In transformer-based pretrained language models, character or sub-word embeddings are preferred over word embeddings. This is because a) small vocabulary size in character and sub-word embeddings compared to word embeddings. The vocabulary of word embeddings consists of all the unique words (or all the words above the cut-off frequency) in the pretraining corpus, whereas vocabulary in character embedding models consists of all the characters and vocabulary in sub-word embedding models consists of all the characters, frequently occurring sub-words and words. The size of vocabulary also determines the overall size of pretrained language model [116]. b) can represent any word and hence overcome the problem of OOV words which is a serious problem with word embeddings c) can encode fine-grained information at character or sub-word levels in word representation. Apart from representing input data using embeddings, it is also necessary additional information like position, language, etc. We classify embeddings into main and auxiliary depending on whether they represent the input data or provide additional information to the model (refer Figure 8).

3.4.1 Main Embeddings

- Main embeddings represent the input data in dense low dimensional vectors. In TPLMs, the input data is mostly a sequence of words. However, in domain-specific models like BERT-EHR [117], MedBERT [118], and BEHRT [119] the input is a sequence of medical codes. All these three models are pretrained on medical text from electronic health records (EHRs).

Text Embeddings: Input for most of the TPLMs is the sequence of words. Input text can be represented using character, sub-word, or combination of character and sub-word embeddings. Models like CharacterBERT [66], AlphaBERT [120] use character embeddings, and models like CharBERT [121] use both character and sub-word embeddings. Models like BERT, RoBERTa, XLNet, T5, and BART use sub-word embeddings but the tokenizer used to generate the vocabulary is different in these models.

Character Embeddings: Character embeddings map each character to a dense low dimensional vector. The vocabulary of character embeddings includes all the characters like letters, symbols, punctuations, and numbers. Once the vocabulary is finalized, the embedding of each character in the vocabulary is randomly initialized and then learned during model pretraining. TPLMs use character embeddings in two ways. The first one is, character level word representation is generated from character embeddings and then a sequence of transformer layers are applied to encode contextual information [66]. For example in CharacterBERT [66], fine-grained word representation is generated from character embeddings using a character encoder based on Char-CNN and highway layer [122]. The second one is based on context string embedding [123]. Here, there is no notion of the explicit word, and input text is modeled as a sequence of characters. In AlphaBERT [120], transformer layers are directly on character embeddings whereas in CharBERT [121] transformer layers are applied after applying BiGRU on character embeddings. Here BiGRU processes the input at the character level and generates

contextualized character embeddings [124].

Sub-Word Embeddings: Unlike character embeddings, the vocabulary of sub-word embeddings consists of characters, frequently occurring sub-words, and words. Here the vocabulary can be generated using any of the tokenizers like WordPiece [59], Byte Pair Encoding (BPE) [60], Byte Level BPE (bBPE) [61], Unigram [125], and SentencePiece [62]. Except Unigram, tokenizers like WordPiece, BPE, bBPE generate vocabulary by starting with base vocabulary having only the characters and iteratively augment the vocabulary until the predefined size is reached. BPE chooses the new symbol pair to be included in the vocabulary based on frequently while WordPiece does it based on language model probability. bBPE is the same as BPE except that it represents each character as a byte. Unigram starts with a large vocabulary and then arrives at a vocabulary of predefined size by iteratively cutting the characters which is exactly opposite to what happens in BPE and WordPiece. Tokenizers like WordPiece and BPE assume space as a word separator in the input text which is not true in all cases. To overcome this, SentencePiece tokenizer treats space as a character and then generates the vocabulary using BPE or Unigram.

The size of the vocabulary must be chosen carefully. Too small vocabulary size results in longer input sequences as more words will be split into many sub-words which hinders model learning and increases pretraining time. Too large vocabulary represents more words using a single token but increases the size overall of the model [126]. However, in the case of multilingual models like mBERT, XLM, and XLM-R, it is necessary to have a large vocabulary to accommodate more languages. Once the vocabulary is generated, each token in the vocabulary is assigned with a randomly initialized embedding and then learned during model pretraining.

Hybrid Embeddings: To leverage the benefits in both character and sub-word embeddings, models like CharBERT [121] uses both character and sub-word embeddings. The model uses dual-channel CNN-based interaction module to model the interaction between character and sub-word embeddings.

Code Embeddings: In the medical domain, each concept is represented using a standard code from ontology. Here the concept can be a disease, drug, symptom, etc. All the information during patient visits to a hospital is represented using medical codes in EHRs. Pretrained language models in the biomedical domain like BERT-EHR [117], MedBERT [118], and BEHRT [119] expect a sequence of medical codes as input. So, in these models vocabulary consists of medical codes from standard clinical ontologies. Embeddings for these medical codes are randomly initialized and learned during model pretraining.

3.4.2 Auxiliary Embeddings

Auxiliary embeddings provide additional information to the model. Each auxiliary embeddings have its purpose. For example, positional embeddings represent the position, while segment embeddings distinguish tokens from different sentences in the input sentence pair, language embeddings in multilingual pretrained models like XLM [63] and Unicoder [103] provide information about the language of the input sentence. Among auxiliary embeddings, position and segment embeddings are commonly used while the other embeddings are used in specific pretrained language models. For example, age, gender, and semantic group embeddings are used in biomedical pretrained language models only [117], [119], [127].

Position Embeddings: In traditional deep learning models like CNN and RNN, it is not necessary to provide any auxiliary embeddings along with text embeddings to represent the position of input tokens. This is because these models implicitly learn the order of input tokens. For example, RNN process input sequence character by character or word by word, and hence it automatically learns the order. In CNN, convolution operations are performed at a fixed size window level instead of character or word level and hence it also learns the order automatically. As transformers do not have convolution or recurrent layers to learn order, position embeddings are provided. Position embeddings can be absolute [2], [4], [5], [117] or relative [3]. In models like BERT, RoBERTa, ELECTRA absolute position embeddings are learned along with other parameters of the model. However, in models like BERT-EHR [117] and BEHRT [119], absolute position embeddings are predetermined to handle an imbalance in patient sequence length.

Segment Embeddings: In the case of sentence pair tasks, the model takes both the input sentences at the same time. So, it is necessary to distinguish tokens of two input sentences using segment embeddings. Position embedding is different for different tokens in the input sentence, but segment embedding is the same for all the tokens in each input sentence.

Language Embeddings: Language embeddings are used in cross-lingual pretrained language models like XLM [63], Unicoder [103], and XNLG [106]. For example, models like XLM are pretrained using a) MLM on monolingual text data in 100 languages and b) TLM using parallel data. Language embeddings are used to explicitly inform the model about the language of the input sentence. In MLM which involves sentences in one language, language embedding will be the same for all the tokens in the input sentence. In the case of TLM which involves a pair of sentences from two different languages, language embedding will be the same for all the tokens in a sentence but different from the language embedding assigned to tokens in other input sentence. However, language embeddings are not used in XLM-R [64] model to allow the model to better deal with code-switching.

Entity Type Embeddings: OAG-BERT [42] is a pretrained academic language model like SciBERT [43]. It is pretrained on academic text corpus. Unlike SciBERT

which is just pretrained on academic text, OAG-BERT is pretrained on academic text as well as augmented with information about various entities in academic text like paper, published venue, author affiliation, research domain, and authors. Information about various entities is provided to the model during pretraining via entity type embeddings.

Age and Gender Embeddings: In medical pretrained models like BEHRT [119] and BERT-EHR [117], the input is a sequence of patient visits where each patient visit is represented as a sequence of medical codes. Apart from model codes, it is useful to provide additional information like age and gender. For example, each patient visits happen at different times. Providing age information to the model allows it to leverage temporal information. Age and gender information is provided to these models explicitly via age and gender embeddings.

Semantic Group Embeddings: UmlsBERT [127] is a knowledge enriched medical language model. It is obtained by continual pretraining ClinicalBERT [46] on UMLS data using novel multi-label MLM pretraining task. UMLS is a collection of over 100 medical ontologies. In UMLS, each medical concept is assigned a unique called Concept Unique Identifier, semantic type, synonyms, related concepts, etc. During continual pretraining, semantic type information is provided explicitly via semantic group embedding so that the language model can a) learn better representation for rare words and b) better model the association between words of the same semantic type.

4 Taxonomy

To understand and keep track of the development of various T-PTLMs, as shown in Figure 9, we classify T-PTLMs from four different perspectives namely Pretraining Corpus (Section 4.1), Model Architecture (Section 4.2), Type of SSL (Section 4.3), and Extensions (Section 4.4).

4.1 Pretraining Corpus-based

4.1.1 General

Models like GPT-1 [1], BERT [2], UnilM [115], XLNet [3], RoBERTa [4], ELECTRA [5], T5 [6], and BART [8] are pretrained on general corpus. For example, GPT-1 is pretrained on Books corpus while BERT and UniLM are pretrained on English Wikipedia and Books corpus. As the amount of text data available in Book corpus or English Wikipedia, text is gathered from multiple sources for pretraining models like XLNet, RoBERTa, ELECTRA, BART and T5.

4.1.2 Social Media-based

T-PTLMs like BERT and RoBERTa are pretrained on formal text. As social media text is highly informal in nature with a lot of noise in the form of irregular grammar, slang words, and non-standard abbreviations, these models have limited performance on social media datasets [81], [111], [129], [130] . Researchers working at the intersection of social media and NLP have developed social media-specific T-PTLMs either by training from scratch [129] or continual pretraining [34], [81], [83], [111], [112], [130] and a summary of these models is presented in Table 3. Except for Bertweet [129], all other social media-based T-PTLMs are developed by continual pretraining. Training from scratch is effective only when the pretraining corpus consists of a large number of tweets. Otherwise, continual pretraining is recommended. For example, BERTweet [129] is pretrained from scratch using 850M tweets. Barbieri et al. [111] showed that RoBERTa model trained from scratch on tweets achieved less performance compared to RoBERTa model adapted to social media by continual pretraining. This is because of using just 60M tweets for pretraining. Different from other social media-based T-PTLMs which are developed using commonly used pretraining tasks like MLM and NSP, BERT-SentiX [83] is obtained by continual pretraining on user reviews using four novel sentiment aware pretraining tasks.

4.1.3 Language-based

Language-based T-PTLMs can be monolingual or multilingual. Monolingual T-PTLMs are pretrained on specific language corpus while multi-lingual T-PTLMs are pretrained multiple language corpus.

Multi-lingual T-PTLMs Inspired by the tremendous success of BERT in English, the authors of BERT developed mBERT by pretraining BERT model from scratch using Wikipedia text from 104 languages [2]. mBERT is the first multilingual T-PTLM. Following mBERT, many multilingual T-PTLMs are proposed. Recent research works showed that the performance of the model can be improved by training on large volumes of text. So, XLM-R [64] is pretrained on CC-100 which consists of a large amount of text particularly for low-resource languages compared to Wikipedia. Inspired from "Scaling laws for neural language models" [26] much larger models like XLM-RXL [131] and XLM-RXXL [131] are pretrained on CC-100 and achieved much better results. mBERT, XLM-R, and its variants are pretrained on only non-parallel data. However, pretraining the model on parallel data along with non-parallel data allows the model to learn cross-lingual representations in a much better way. Models like MuRIL [97], mT6 [91], InfoXLM [90], XLM [63] and Unicoder [103] are pretrained on both parallel and non-parallel data. Multi-lingual NLP research community also developed many generative T-PTLMs like mT5 [99], mT6 [91], mBART [65] and IndoBART [88] based on encoder-decoder architecture. A summary of various multi-lingual T-PTLMs is presented in Table 4.

Monolingual T-PTLMs Multilingual models are pretrained on the corpus from multiple languages and hence they can be used for NLP tasks in more than one language. However, the following drawbacks force the NLP community to develop separate models for each

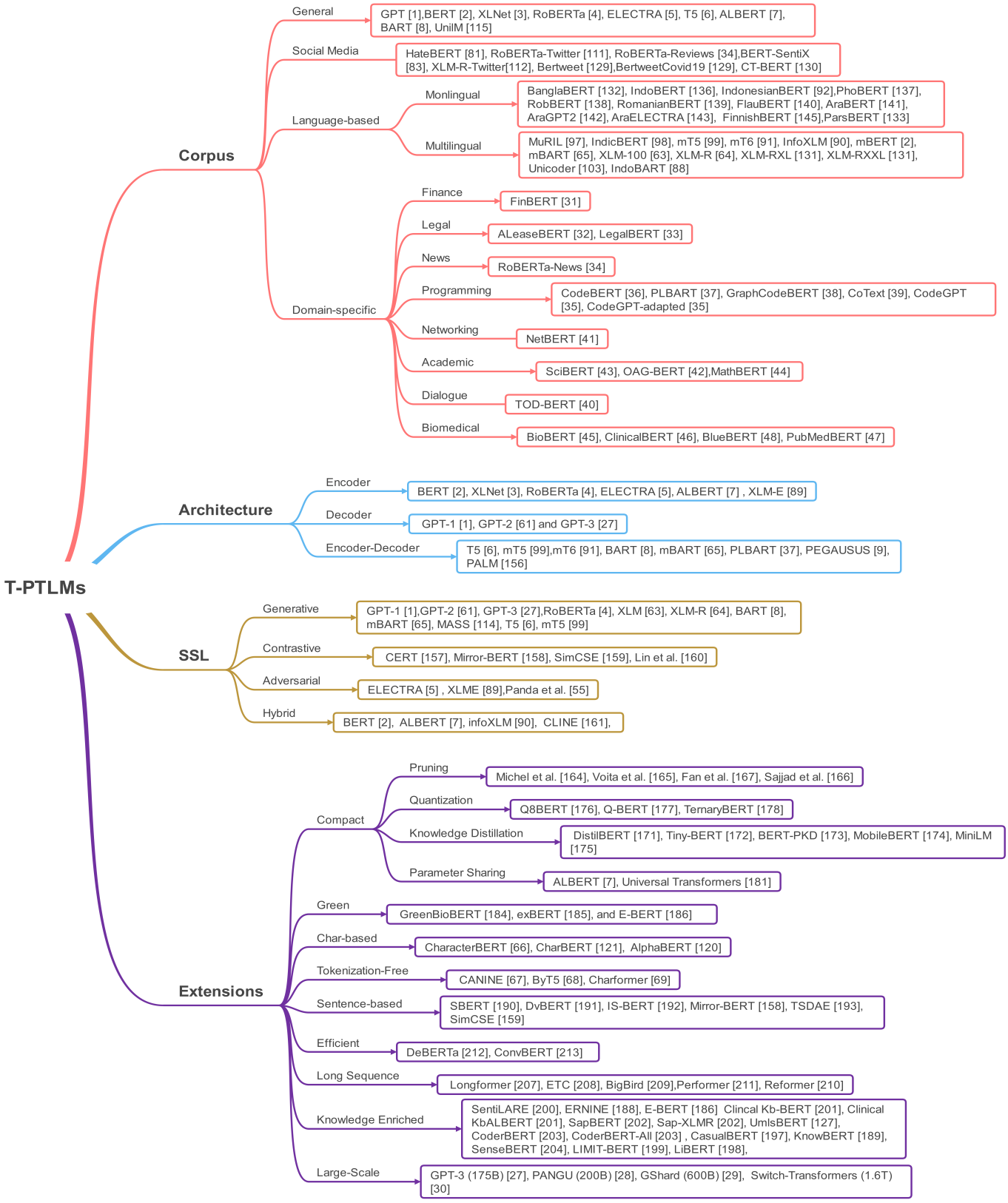

Fig. 9: Taxonomy of T-PTLMs

Name	Pretrained from	Pretraining tasks	Corpus	Evaluation
HateBERT [81]	BERT	MLM	RAL-E (dataset of 1.5M hateful Reddit comments)	Offensive tweets classification
RoBERTa-Twitter [111]	RoBERTa	MLM	Tweets (60M)	Tweet classification
RoBERTa-Reviews [34]	RoBERTa	MLM	Amazon reviews (24.75M) [128]	Review classification
BERT-SentiX [83]	BERT	SWP, WP, EP and RP	Amazon (233M) [82] and Yelp reviews (8M) Reviews	Cross domain sentiment analysis
XLM-R-Twitter [112]	XLM-R	MLM	Tweets in multiple languages (198M)	TweetEval [111] and UMSAB [112]
Bertweet [129]	Scratch	MLM	Tweets (845M English + 5M COVID tweets)	POS, NER and Tweets classification
BertweetCovid19 [129]	Bertweet	MLM	COVID tweets (23M)	Tweet classification
CT-BERT [130]	BERT	MLM, NSP	COVID tweets (160M)	Tweet classification

TABLE 3. Summary of social-media based T-PTLMs.

Name	Architecture	Pretraining tasks	Corpus	Vocabulary	#Lang	#Parameters
MuRIL [97]	Encoder	MLM + TLM	Wikipedia, Common Crawl + Parallel data	WordPiece (197K)	17	236M
IndicBERT [98]	Encoder	MLM	IndicCorp	SentencePiece (200K)	12	33M
mT5 [99]	Encoder-Decoder	Seq2SeqLM	mC4	SentencePiece (250K)	101	300M, 580M, 1.2B, 3.7B and 13B (base, large, xl and xxl)
mT6 [91]	Encoder-Decoder	Seq2SeqLM, MT, TPSC, TSC	CCNet + Parallel data	SentencePiece (250K)	94	300M
InfoXLM [90]	Encoder	MLM, TLM , XLCo	CC-100 + Parallel data	SentencePiece (250K)	94	270M and 559M (base and large)
mBERT [2]	Encoder	MLM , NSP	Wikipedia	WordPiece (110K)	104	172M
mBART [65]	Encoder-Decoder	DAE	CC-25	SentencePiece (250K)	25	680M
XLM-15 [63]	Encoder	MLM, TLM	Wikipedia + Parallel data	BPE (95K)	15	250M
XLM-17 [63]	Encoder	MLM	Wikipedia	BPE (200K)	17	570M
XLM-100 [63]	Encoder	MLM	Wikipedia	BPE (200K)	100	570M
XLM-R [64]	Encoder	MLM	CC-100	SentencePiece (250K)	100	270M ,560M (base and large)
XLM-RXL [131]	Encoder	MLM	CC-100	SentencePiece (250K)	100	3.5B
XLM-RXXL [131]	Encoder	MLM	CC-100	SentencePiece (250K)	100	10.7B
Unicoder [103]	Encoder	MLM, TLM, CLWR, CLPC, CLMLM	Wikipedia + Parallel data	BPE (95K)	15	250M
IndoBART [88]	Encoder-Decoder	DAE	Indo4B-plus	BPE (40K)	3	130M

TABLE 4. Summary of multi-lingual T-PTLMs.

language starting from BanglaBERT [132] to ParsBERT [133].

- **Curse of Multilinguality** [64]: Multilingual models cannot represent all the languages equally. This is because of the underrepresentation of low-resource languages in the pretraining corpus and the limited capacity of the model. Moreover, adding more languages after a certain limit reduces the model performance.
- **Embedding barrier** [132]: The performance of multi-

lingual models in high resource languages that have adequate representation in model vocabulary is on par with their monolingual models [134]. However, in the case of languages without adequate representation in model vocabulary, the difference in the performance of the monolingual and multilingual model is significant. Due to the high imbalance in the pretraining corpus, the representation of low-resource languages in multilingual model vocabulary is very limited which is referred to as the embedding barrier [90]. For example, the representation of the Arabic language [135] in popular multilingual models is 5K out of 110K in mBERT and 14K out of 250K in XLM. This issue is more severe in the case of languages like Bangla that does not share vocabulary or script with any high-resource languages. The percentage of Bangla vocabulary in the multilingual model is less than 1% [132]. With very limited representation in the vocabulary, words in low resource languages are tokenized into many subwords which increases input sequence length, hinders model learning, and makes training expensive.

A summary of the various monolingual model is presented in Tables 5 and 6. Monolingual models are pretrained based on standard model architectures like GPT [142] BERT [74], [75], [92], [133], [135], [136], [139]–[141], [144]–[146], [151]–[154], RoBERTa [137], [138], [147]–[149], [152], [155], ALBERT [92], [150], ELECTRA [132], [143], and T5 [77]. As the availability of corpus from one source is limited in the case of many languages, most of these models are pretrained using corpus gathered from multiple sources. For example, IndoBERT [136] is pretrained on corpus having text from Wikipedia, News domain, and Internet. Except for models like PTT5 [77], RuBERT [74] and PortugueseBERT [75], all other monolingual models are pretrained from scratch. Models like PTT5, RuBERT, and PortugueseBERT are initialized from existing models and further pretrained with new language-specific vocabulary. In these models, only the transformer encoder layer parameters are copied from existing models while embedding layer parameters are randomly initialized. During CPT, embedding layer parameters are updated along with other layers.

4.1.4 Domain-Specific Models

Following the success of T-PTLMs in general domain, T-PTLMs in specific domains like Finance [31], Legal [32], [33], News [34], Programming [35]–[39], Dialogue [40], Networking [41], Academic [42]–[44] and Biomedical [45]–[48] have been developed (refer Table 7 for a brief summary). Models like BERT, RoBERTa, BART, and T5 are pretrained on general domain text. For a model to perform well on domain-specific datasets the model should have enough domain knowledge [31], [45]. These general domain models can not acquire enough domain knowledge just through fine-tuning. As a result, the performance of these models on domain-specific datasets is limited [31], [45]. The initial trend to develop domain-specific models is using continual pretraining i.e., initialize the model with any of the existing general domain models and further pretrain on a domain-specific corpus. For example, BioBERT [45] is the first domain-specific BERT model developed using continual pretraining. Following BioBERT in biomedical domain, models like AleaseBERT [32], RoBERTa-News [34], GraphCodeBERT [38], CoText [39], CodeGPT-adapted [35], NetBERT [41], MathBERT [44], TOD-BERT [40], ClinicalBERT [46] and BluBERT [48] have been developed using continual pretraining.

The main advantage of developing domain-specific models using continual pretraining is that the model converges faster as it is not trained from scratch and hence it is comparatively less expensive. However, as these models use the same vocabulary learned over general domain text, many of the domain-specific words are missing in the vocabulary. For example, the vocabularies of FinBERT and general BERT have 41% of common tokens [31]. As a lot of domain-specific words are missing in the vocabulary, many of the domain-specific words are not represented properly which hinders model learning. The advantage of having domain-specific vocabulary is that even if the word is missing in the vocabulary, the word will be split into meaningful tokens. For example, the word actyeltransferase is split into [“ace”, “ty”, “lt”, “ran”, “sf”, “eras”, “e’] by BERT model whereas the same word is split into meaning tokens [“acetyl”, “transferase”] by SciBERT which has domain-specific vocabulary [47]. Models like PubMedBERT [47], FinBERT [31], LegalBERT [33], CodeBERT [36], PLBART [37], CodeGPT [35], SciBERT [43] and OAG-BERT [42] are pretrained from scratch.

4.2 Architecture

Transformers, a novel self-attention model deep learning proposed by Vaswani et al. [25] consists of stack of both encoder and decoder layers. A T-PTLM can be pretrained using a stack of encoders or decoders or both.

4.2.1 Encoder-based

In general, an encoder-based T-PTLM consists of an embedding layer followed by a stack of encoder layers. For example, the BERT-base model consists of 12 encoder layers while the BERT-large model consists of 24 encoder layers [2]. The output from the last encoder layer is treated as the final contextual representation of the input sequence. In general, encoder-based models like BERT [2], XLNet [3], RoBERTa [4], ELECTRA [5], ALBERT [7] and XLM-E [89] are used in NLU tasks.

4.2.2 Decoder-based

A decoder-based T-PTLM consists of an embedding layer followed by a stack of decoder layers. Here transformer

Name	Language	Pretrained from	Pretraining tasks	Corpus	Vocabulary
BanglaBERT [132]	Bangla	Scratch	RTD	Bangla Web text corpus	WordPiece (32k)
IndoBERT [136]	Indonesian	Scratch	MLM	Indonesian Wikipedia, News and Web corpus	WordPiece (32k)
IndonesianBERT [92]	Indonesian	Scratch	MLM	Indo4B	Sentencepiece (30k)
IndonesianBERT-Lite [92]	Indonesian	Scratch	MLM and SOP	Indo4B	Sentencepiece (30k)
PhoBERT [137]	Vietnamese	Scratch	MLM	Vietnamese Wikipedia and News corpus	BPE (64K)
RobBERT [138]	Dutch	Scratch	MLM	OSCAR corpus	bBPE (40K)
RomanianBERT [139]	Romanian	Scratch	MLM,NSP	OPUS, OSCAR and Wikipedia corpus	BPE (50k)
FlauBERT [140]	French	Scratch	MLM	French text corpus	BPE (50K)
AraBERT [141]	Arabic	Scratch	MLM and NSP	Arabic Wikipedia and News corpus	SentencePiece (64K)
AraGPT2 [142]	Arabic	Scratch	CLM	Arabic Wikipedia, OSCAR and News corpus	bBPE (64K)
AraELECTRA [143]	Arabic	Scratch	RTD	Arabic Wikipedia, OSCAR and News corpus	SentencePiece (64K)
BERTje [144]	Dutch	Scratch	MLM, SOP	Dutch Books, Wikipedia and News corpus	SentencePiece (30K)
FinnishBERT [145]	Finnish	Scratch	MLM, NSP	Finnish News, Online discussion and Common Crawl Corpus	SentencePiece (50K)
ALBERTO [146]	Italian	Scratch	MLM,NSP	Italian tweets corpus	SentencePiece (128K)
PortugueseBERT [75]	Portuguese	mBERT	MLM,NSP	BrWaC [93]	SentencePiece (30K)
RuBERT [74]	Russian	mBERT	MLM,NSP	Russian Wikipedia and News corpus	SentencePiece
BETO [147]	Spanish	Scratch	MLM	Wikipedia and Common Crawl corpus	SentencePiece (32K)
CamemBERT [148]	French	Scratch	MLM	French OSCAR corpus	SentencePiece(32K)
ARBERT [135]	Arabic	Scratch	MLM, NSP	Arabic Wikipedia, Books, Common Crawl, News corpus	WordPiece (100K)
MARBERT [135]	Arabic	Scratch	MLM	Arabic tweets corpus	WordPiece (100K)
WangchanBERTa [149]	Thai	Scratch	MLM	Thai Wikipedia , Social media posts, Books and reviews corpus	SentencePiece (25K)
KoreALBERT [150]	Korean	Scratch	MLM, SOP and WOP	Korean Wikipedia, News, Internet crawl and Books corpus	SentencePiece (32K)

TABLE 5. Summary of monolingual T-PTLMs.

decoder layer consists of only masked multi-head attention and feed-forward network layers. The multi-head attention module which performs encoder-decoder cross attention is removed. In general, decoder-based models like GPT-1 [1], GPT-2 [61] and GPT-3 [27] are used in NLG tasks.

4.2.3 *Encoder-Decoder based*

Encoder-decoder based T-PTLMs are more suitable for sequence-to-sequence modeling tasks like Machine

Name	Language	Pretrained from	Pretraining tasks	Corpus	Vocabulary
PTT5 [77]	Portuguese	T5	Seq2SeqLM	BrWac corpus	SentencePiece (32K)
SweedishBERT [153]	Sweedish	Scratch	MLM ,NSP	Sweedish text corpus (Wikipedia, News, Social media and Legal)	SentencePiece (50K)
SweedishALBERT [153]	Sweedish	Scratch	MLM, SOP	Sweedish text corpus (Wikipedia, News, Social media and Legal)	SentencePiece (50K)
SweedishELECTRA [153]	Sweedish	Scratch	RTD	Sweedish text corpus (Wikipedia, News, Social media and Legal)	SentencePiece (50K)
HerBERT [151]	Polish	Scratch	MLM	Polish text corpus (Wiki, OSCAR, Subtitles and Books)	BPE (50K)
PolishBERT [154]	Polish	Scratch	MLM	Polish text corpus (Common Crawl, Wikipedia, Books and OPUS)	SentencePiece (50K)
RobeCzech [155]	Czech	Scratch	MLM	Wikipedia, Web and News corpus	bBPE (52K)
KLUE-BERT [152]	Korean	Scratch	MLM,NSP	Korean text corpus	BPE(32K)
KLUE-RoBERTa [152]	Korean	Scratch	MLM	Diverse Korean text corpus	BPE(32K)
ParsBERT [133]	Persian	Scratch	MLM,NSP	Persian text corpus	WordPiece (100K)

TABLE 6. Summary of monolingual T-PTLMs.

Translation, Text Summarization, etc. MASS [114] is the first encoder-decoder based T-PTLM model. It is pretrained using Seq2SeqLM, an extension of MLM to encoder-decoder architectures. Following MASS, a number of encoder-decoder models like T5 [6], mT5 [99], mT6 [91], BART [8], mBART [65], PLBART [37], PEGAUSUS [9] and PALM [156] are proposed in the recent times. For example, Models like MASS and BART use bidirectional encoder over corrupted text and lef-to-right auto regressive decoder to reconstruct the original text.

4.3 SSL

SSL is one of the key ingredients in building T-PTLMs. A T-PTLM can be developed by pretraining using Generative, Contrastive or Adversarial, or Hybrid SSL.

4.3.1 Generative SSL

Generative SSL helps the model to learn by predicting tokens. The different scenarios in generative SSL are a) predicting the next token based on current tokens (CLM) b) prediction the masked tokens (MLM and its variants like TLM, Seq2SeqLM) c) reconstructing the original text from the corrupted text (DAE). Some of the popular models developed using Generative SSL are GPT-1 [1], GPT-2 [61], GPT-3 [27] (based on CLM), RoBERTa [4], XLM [63], XLM-R [64] (based on MLM and its variants like TLM), BART [8], mBART [65] (based on DAE), and MASS [114], T5 [6], mT5 [99] (based on Seq2SeqLM).

4.3.2 Contrastive SSL

Contrastive SSL helps the model to learn by comparison. In NLP, there is no T-PTLM which is pretrained using Contrastive SSL only. Contrastive SSL is used in continual pretraining to further improve the model i.e., to learn sentence-level semantics. For example, CERT [157] uses contrastive SSL to improve the BERT model by injecting more sentence-level semantics. CERT outperforms BERT in many GLUE tasks. Similarly, Mirror-BERT [158] and SimCSE [159] use contrastive SSL to allow the BERT model to generate quality sentence embeddings. Lin et al. [160] showed that multi-lingual contrastive pretraining improves the performance of multilingual T-PTLMs in Mickey Probe.

4.3.3 Adversarial SSL

Adversarial SSL helps the model to learn by distinguishing corrupted tokens. Here the corrupted tokens can be replaced or shuffled. Adversarial SSL can be used in training the model from scratch or in continual pretraining. Models like ELECTRA [5] and XLM-E [89] are pretrained using adversarial SSL. ELECTRA is pretrained using replaced token detection (RTD) while XLM-E is pretrained using multi-lingual replaced token detection(MRTD) and translation replaced token detection (TRTD). Panda et al. [55] used adversarial SSL based on shuffled token detection (STD) to further improve RoBERTa model.

Name	Domain	Pretrained from	Pretraining tasks	Corpus	Vocabulary
FinBERT [31]	Finance	Scratch	MLM + NSP	Financial Communication Corpus	WordPiece (31K)
ALeaseBERT [32]	Legal	ALBERT	MLM	Lease Agreements	Same as ALBERT
LegalBERT [33]	Legal	Scratch	MLM + NSP	English Legal Text	Sentencepiece (31K)
RoBERTa-News [34]	News	RoBERTa	MLM	Real News corpus	Same as RoBERTa
CodeBERT [36]	Programming	Scratch	MLM+RTD	CodeSearchNet	WordPiece
PLBART [37]	Programming	Scratch	DAE	Github and Stackoverflow corpus	Sentencepiece (50K)
GraphCodeBERT [38]	Programming	CodeBERT	MLM, EP and NA	CodeSearchNet	Same as CodeBERT
CoText [39]	Programming	T5	Seq2SeqLM	CodeSearchNet and Github code	Same as T5
CodeGPT [35]	Programming	Scratch	CLM	CodeSearchNet	BPE (50K)
CodeGPT-adapted [35]	Programming	GPT-2	CLM	CodeSearchNet	Same as GPT-2
NetBERT [41]	Networking	BERT	MLM	Computer Networking Corpus	Same as BERT
SciBERT [43]	Academic	Scratch	MLM and NSP	Semantic Scholar	WordPiece (30K)
OAG-BERT [42]	Academic	Scratch	MLM	OAG text corpus	WordPiece (44K)
MathBERT [44]	Academic	BERT	MLM,CCP and MSP	Arxiv papers	Same as BERT
TOD-BERT [40]	Dialogue	BERT	MLM and RCL	Dialogue Corpus	Same as BERT
BioBERT [45]	Biomedical	BERT	MLM+NSP	PubMed and PMC	Same as BERT
ClinicalBERT [46]	Biomedical	BERT	MLM+NSP	MIMIC-III	Same as BERT
BlueBERT [48]	Biomedical	BERT	MLM+NSP	PubMed and MIMIC-III	Same as BERT
PubMedBERT [47]	Biomedical	Scratch	MLM+NSP	PubMed and PMC	WordPiece

TABLE 7. Summary of domain-specific T-PTLMs.

4.3.4 *Hybrid SSL*

Some of the T-PTLMs are pretrained using more than one type of SSL. For example, BERT model - generative (MLM) and contrastive SSL (NSP), ALBERT- generative (MLM) and contrastive SSL (SOP), infoXLM – generative (MLM, TLM) and contrastive SSL (XLCo). Here XLCo represents the cross lingual contrastive pretraining task. Models like CLINE [161] are obtained by further pretraining RoBERTa model using generative (MLM), contrastive, and adversarial SSL (RTD).

4.4 Extensions

4.4.1 *Compact T-PTLMs*

PTLMs have achieved huge success in almost every NLP task. Recently researchers observed that the performance of PTLMs can be increased just by increasing the size of the model and training with large volumes of corpus for more training steps [26]. The large size and high latency make the deployment of PLTMs difficult in real-word applications where resources are limited and require fast inference. To reduce the size of T-PTLMs and make them faster, many model compression techniques like pruning, parameter sharing, knowledge distillation, and

quantization are explored in recent times [162].

Pruning: In general, deep learning models like T-PTLMs are over parameterized i.e., some of the model components (weights [163], attention heads [164], [165], or layers [166], [167] can be removed during pretraining or after pretraining without much impact on the model performance and also reducing the model storage space and inference time. Pruning is inspired from the biological observation, "thousands of trillions of synapses in a newborn baby reduces to 500 trillion synapses after ten years" [162]. T-PTLMs are trained with multiple attention heads. Michel et al. [164] and Voita et al. [165] showed that most of the attention heads are redundant and can be removed during inference. Fan et al. [167] showed that encoder layers can be dropped during pretraining which allows dropping layers during inference. In contrast, Sajjad et al. [166] applied layer dropping on the pre-trained models which eliminates training from scratch unlike Fan et al. [167].

Knowledge Distillation: Knowledge Distillation is a model compression method that allows training compact student models using the knowledge from large teacher models. During knowledge distillation, the student learns the generalization ability of the teacher model by reproducing its behavior, and hence the performance of the student model is on par with the teacher model. Knowledge Distillation is introduced by Bucila et al. [168] and later generalized by Ba and Caruna [169] and Hinton et al. [170]. The approach of Ba and Caruna [169] trains the student model using L2 loss between teacher and student model logits while the approach of Hinton et al. [170] uses cross-entropy between softmax logits of teacher and student (soft loss) as well as cross-entropy loss between student prediction and actual label (hard loss). Some of the popular models trained using Knowledge distillation are DistilBERT [171], Tiny-BERT [172], BERT-PKD [173], MobileBERT [174], and MiniLM [175].

Quantization: Quantization compresses a model by using fewer bits to represent weights. In general, T-PTLMs parameters are represented using 32 or 16 bits. Quantized T-PTLMs use 8 bits [176] or even lesser [177]–[179] to present weights. Pruning compresses a model by removing less important weights, while quantization compresses a model by using fewer bits for weights representation. Some of the popular quantized BERT models are Q8BERT [176], Q-BERT [177], and Ternary-BERT [178]. To reduce performance drop in ultra-low (1 or 2) bit models, researchers proposed methods like mixed-bit quantization [177], [179], combining knowledge distillation with quantization [178], and product quantization (PQ) [180]. Mixed-bit quantization is not supported by some hardware while PQ requires extra clustering operations. Overall, quantization compresses the model by using fewer bits but as quantization is hardware specific, we need specialized hardware to use quantized models.

Parameter Sharing: ALBERT [7] a lite version of the BERT model achieves parameter reduction by cross-layer parameter sharing and factorized embedding parameterization. Factorized embedding parameterization splits the large vocabulary matrix into two small matrices which allow growing the hidden vector size without significantly increasing vocabulary matrix parameters. Cross-layer parameter sharing prevents the growth of parameters with the increase in the depth of the model. With these two parameter reduction techniques, the ALBERT model has 18x fewer parameters and can be trained 1.7x faster compared to the BERT-large model. Cross-layer parameter sharing is also explored in Universal Transformers [181].

4.4.2 Character-based T-PTLMs

Most of the T-PTLMs use sub-word embeddings based on tokenizers like BPE, bBPE, WordPiece, Unigram, and SentencePiece. The problem with the use of word embeddings is the requirement of a large vocabulary and OOV problem. Sub-word embeddings are based on the idea that only rare and misspelled words should be represented using sub-words while frequently used words should be represented as it is. Sub-word embeddings overcome the two problems in word embeddings. However sub-word embeddings have two drawbacks a) cannot encode fine-grained character level information in the word representation and b) brittleness to noise i.e., even simple typos can change the representation of a word which hinders the model learning [66], [121].

To overcome these drawbacks, character-based T-PTLMs like CharacterBERT [66], CharBERT [121], and AlphaBERT [120] are proposed. CharacterBERT uses CharCNN+Highway layer to generate word representations from character embeddings and then apply transformer encoder layers. The use of CharCNN+Highway layer is inspired from ELMo [122]. Different from CharacterBERT which uses only character embeddings, CharBERT uses both character and sub-word embeddings. In CharBERT, character-level word embeddings are generated from character embeddings using Bidirectional GRU similar to contextual string embeddings [123]. Then a dual-channel CNN is used in every transformer encoder layer to model the interaction between character and sub-word embedding channels. Due to the inclusion of an extra channel for character embeddings and interaction module in every layer, the size of the model increases by 5M parameters. CharBERT or CharRoBERTa are more robust to noise and perform better than BERT or RoBERTa models. Unlike CharacterBERT and CharBERT, AlphaBERT in Biomedical domain operates directly at the character level. In AlphaBERT, transformer encoder layers are directly applied on character embeddings after adding to position embeddings.

4.4.3 Green T-PTLMs

The standard approach to adapt general models to a specific domain or improve general models with knowledge from Knowledge bases is continual pretraining.

The resulting models after continual pretraining achieve good results. But this process is expensive in terms of hardware and run time and also not environmentally friendly with CO2 emissions [182], [183]. Recently, researchers focused on developing less expensive methods to adapt general models to a specific domain or to inject knowledge from knowledge bases. The models developed using less expensive methods like GreenBioBERT [184], exBERT [185], and E-BERT [186] are referred to as Green models as they are developed in a environmentally friendly way.

GreenBioBERT [184] is developed by extending the vocabulary of the general BERT model using domain-specific word embeddings developed using Word2Vec which are further aligned with WordPiece embeddings. GreenBioBERT achieves comparable performance with BioBERT which is developed by further pretraining for 10 days using eight v100 NVIDIA GPUs. exBERT [185] is developed by extending general BERT with domain-specific WordPiece embeddings and an extension module. During continual pretraining, as the extra WordPiece embeddings and extension module parameters are only updated while keeping other parameters freezed, this process is less expensive. E-BERT is developed by extending BERT model vocabulary with the Wikipedia2Vec entity vectors [187] after aligning. E-BERT [186] which doesn't require any further pretraining outperforms models like ERNINE [188] and KnowBERT [189] (both these models require further pretraining to inject information from knowledge bases) on the LAMA benchmark.

4.4.4 Sentence-based T-PTLMs

The sentence embeddings obtained from T-PLTMs like BERT by applying any of the pooling strategies are not effective [190]. Recently many approaches based on supervised learning [190], [191] or self-supervised learning [158], [159], [192]–[194], [194], [195] are proposed. SBERT [190] is one of the first supervised approaches which extend T-PTLMs like BERT to generate quality sentence embeddings. SBERT fine-tunes BERT model using the Siamese network over NLI and STSb datasets. By fine-tuning over NLI and STSb which are sentence pair classification tasks, the model learns sentence-level semantics and hence generates quality sentence vectors. DvBERT [191] which is based on multi-view learning [196] extends SBERT by adding word-level interaction features across two sentences. As supervised approaches require labeled datasets which limits the application of these models in labeled data scare domains.

Moreover, Zhang et al. [192] showed that the performance of NLI and STSb fine-tuned models is limited when training data is limited or distribution of test differs significantly from training data. To overcome the requirement of labeled datasets, many approaches based on SSL are proposed. IS-BERT [192] uses mutual information maximization strategy to learn quality sentence embeddings and CNN instead of mean pooling on the top of BERT model. Mirror-BERT [158] trains BERT by using a contrastive learning-based objective to push positive sentence pairs to have similar representations. Here positive sentences are obtained by random masking in input space or applying dropout in feature space. TSDAE [193] involves reconstructing the original sentence from the sentence embedding generated by the encoder using a corrupted sentence. SimCSE [159] is a contrastive learning-based framework to further train PTLMs to generate better sentence embeddings using unlabelled or labeled data.

4.4.5 Tokenization-Free T-PLTMs

Most of the existing PTLMs use sub-word or character or both the embeddings. The main drawback with these approaches are

- Sub-word embeddings require a fixed vocabulary that is modest in size in models like BERT and RoBERTa but larger in multilingual models. The vocabulary requires a vocabulary matrix in which each token is mapped with a vector and softmax matrix in the output layer. These two matrix parameters occupy a significant amount of model parameters. For example, these two matrix parameters are about 66% of mT5 model parameters [99]. Moreover, having a fixed vocabulary makes the adaptation of models to other domains inefficient i.e., many domain-specific words are not represented properly which impacts model adaptation as well as model downstream performance [47], [66].
- Both sub-word and character embeddings require an explicit tokenizer that splits the input sequence based on white space or punctuation. This becomes problematic in the case of languages that do not use white space or punctuations as word separators. For example, languages like Chinese and Thai do not use white space as separators and languages like Hawaiian and Twi use punctuations as consonants [67].

Recently there is a rising interest in the research community to overcome the above drawbacks with tokenization-free T-PTLMs [67]–[69]. With tokenization-free models, there is no need for language-specific tokenizers, models are more robust to noise and have no large vocabulary which requires a significant amount of model parameters. CANINE [67] is the first tokenization-free T-PTLM which directly operates on character sequence. The model applies convolution layers on the character sequence to reduce the input sequence length and then applies transformer encoder layer stack. CANINE is pretrained with the same tasks as the BERT model. CANINE with 28% fewer parameters compared to mBERT output performs it by 2.8 points on multilingual QA.

ByT5 [68] is an improved version of T5 model to handle input at byte-level without using any fixed vocabulary. T5 uses the same number of encoder and decoder

layers while the depth of the encoder is 3x compared to the decoder in ByT5. Charformer [69] uses a novel tokenizer that uses gradients to automatically learn sub-words from characters which eliminate the requirement of a fixed vocabulary. Charformer performs on par with models like T5 while outperforming byte-level models like ByT5. Moreover,unlike CANINE, the gradient-based sub-word tokenizer output is interpretable.

4.4.6 Large Scale T-PTLMs

Initially the sizes of T-PTLMs are in the range of 110M to 340M parameters [2], [4], [5]. Kaplan et al. [26] showed that the performance of T-PTLMs is strongly related to the scale rather than the depth or width of the model. The authors showed that the performance of T-PTLMs is largely determined by the scale i.e., the number of parameters, the size of pretraining data, and the amount of pretraining compute. According to Kalplan et al. [26] the performance of the model can be increased by increasing the size of the model or training the model on much large volumes of large data or training the model for more training steps. All these three must be scaled up at the same time to achieve optimal performance. This observation triggered the development of large-scale T-PTLMs like GPT-3 (175B) [27], PANGU(200B) [28], GShard (600B) [29] which contains billions of parameters and Switch-Transformers (1.6T) [30] which contains trillions of parameters.

4.4.7 Knowledge Enriched T-PTLMs

T-PTLMs are developed by pretraining over large volumes of text data. During pretraining, the model learns knowledge available in the pretraining text data by solving one ore more challenging pretraining tasks. Recent works [188], [189], [197]–[204] showed that these models can be further improved by integrating the knowledge available in external knowledge sources. T-PTLMs integrated with knowledge from external sources are referred to as Knowledge Enriched T-PTLMs. Some of the popular external knowledge sources are WordNet, Wikidata in the general domain and UMLS [205] in specific domains like Biomedical. Some of the examples of knowledge enriched T-PTLMs are CasualBERT [197], KnowBERT [189], SenseBERT [204], LIMIT-BERT [199], LiBERT [198], SentiLARE [200], ERNINE [188], E-BERT [186] in the general domain and Clincal Kb-BERT [201], Clinical Kb-ALBERT [201], SapBERT [202], Sap-XLMR [202], UmlsBERT [127], CoderBERT [203], CoderBERT-All [203] in specific domain like Biomedical. For example, CasualBERT injects casual knowledge into BERT model using two novel pretraining tasks on cause-effect pairs. LiBERT is pretrained from scratch using lingual relation classification (LRC) task along with MLM and NSP. LRC task helps to inject linguistic knowledge. LiBERT outperforms BERT model in most of the GLUE tasks. SentiLARE introduces label-aware MLM to inject POS tag and word polarity information into BERT model. All the biomedical domain-specific knowledge-enriched models are obtained by integrating knowledge from the biomedical ontology UMLS.

4.4.8 Long-Sequence T-PTLMs

The self-attention attention module in transformers updates the representation of each input token by attending to all tokens in the input sequence. The quadratic time complexity of the self-attention module limits the application of T-PTLMs to long input sequences. To overcome this drawback, self-attention variants like sparse self-attention and linearized self-attention are proposed to reduce its complexity and hence extend T-PTLMs to long input sequences also [206]. Some of the popular T-PTLMs based on a) sparse self-attention are Longformer [207], ETC [208], BigBird [209] and Reformer [210] and b) linearized self-attention are Performer [211]. Sparse self-attention reduces the complexity by including sparsity bias which reduces the number of query-key pairs that each query attends to. In linearized self-attention, reduced complexity is achieved by disentangling the attention with kernel feature maps and then computing the attention in reverse order.

4.4.9 Efficient T-PTLMs

T-PTLMs require pretraining on large volumes of text data for longer durations which makes pretraining highly expensive. Recently, with better model architectures it is possible to achieve similar or better performances using less pretraining data [212] and less pretraining costs [213]. DeBERTa [212] improves the BERT model using disentangled attention mechanism and enhanced masked decoder. Disentangled attention mechanism represents a word using separate vectors to encode its content and position information and then compute the attention weights based on contents and relative positions. An enhanced masked decoder is used to predict masked tokens instead of softmax layer during pretraining. These two novel changes improve pretraining efficiency and DeBERTa model which is pretraining on 78GB of data outperforms RoBERTa which is pretrained on 160GB of data.

ConvBERT [213] improves the BERT model with mixed attention block consisting of self-attention and span based dynamic convolution modules. The self-attention modules model global dependencies while span-based dynamic convolution modules model local dependencies. The authors of ConvBERT observed that some attention heads are needed to model only local dependencies and hence they replaced these attention heads with span-based dynamic convolution modules. ConvBERT with better model architecture outperforms ELECTRA base using less than ¼ of its pretraining cost.

5 Downstream Adaptation Methods

Once a language model is pretrained, it can be used in downstream tasks. A pretrained language model can

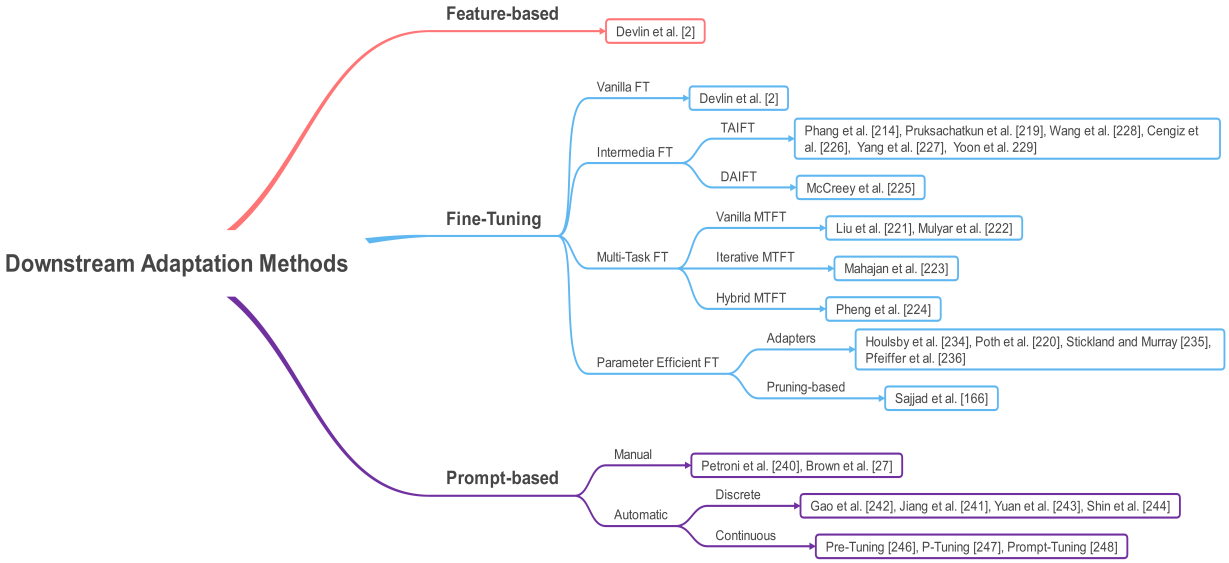

Fig. 10: Downstream adaptation methods

be used in downstream tasks in three ways namely a) feature-based b) fine-tuning and c) prompt-based tuning (refer Figure 10). The feature-based approach involves generating contextual word embeddings from language models and then using them as input features in task-specific downstream models. Fine-tuning involves adapting model weights to downstream tasks by minimizing task-specific loss.

5.1 Feature-based

In traditional deep learning models like CNN or RNN, word embeddings generated using embedding models like Word2Vec [10] or Glove [11] are used as word features. In feature-based approach, BERT [2] based models are used to generate contextual word vectors, and then they are used as input features similar to Word2Vec or Glove embeddings in task-specific downstream models. BERT-based contextual word embeddings are much better as a) they are contextual unlike Word2Vec and Glove embeddings b) overcome the issue of OOV words and c) encode more information in word vectors because of the deep layered model architecture. Here, word vectors can be taken from the last layer (or from multiple layers using any of the pooling strategies) [2]. The advantage with the feature-based approach is that contextualized word vectors can be used any in any of the handcrafted state-of-the-art task specific-architectures. However, feature-based approach involves training the downstream model from scratch (except embeddings) which requires a large number of labeled instances.

5.2 Fine-tuning

Pretraining allows the pretrained language model to gain universal language knowledge. However, the performance of the model in downstream tasks requires task-specific knowledge i.e., for the model to perform well in downstream tasks, its weights should be close to the ideal setting for the target task [214]. Fine-tuning imparts task-specific knowledge to the model by adapting its weights based on task-specific loss [2]. Moreover, fine-tuning enhances the model performance because it clusters the points of different labels away from each other such that there is a large separation between the cluster regions [215]. Fine-tuning updates all the transformer layers including the embedding layer but the higher layers are subjected to more changes compared to the lower layers [215]–[218].

Models like BERT, RoBERTa, and ELECTRA do not follow unified input-output format across tasks i.e., different tasks have different input and output formats. So, it is required to add task-specific layers in these models during fine-tuning. However, in models like T5 [6] which follow the same input-output format across tasks, there is no need to add any extra layers specific to each task. T5 follows a text-to-text format in any task i.e., input for the model is some text and the model has to produce some text as output.

Fine-tuning can be a) Vanilla fine-tuning [2] b) Intermediate fine-tuning [214], [219], [220] c) Parameter efficient fine-tuning and d) Multi-task fine-tuning [221]–[224]. Unlike Vanilla fine-tuning which is prone to overfit the model on small datasets, intermediate fine-tuning or multi-task fine-tuning avoid overfitting the model on small datasets. As fine-tuning involves adjustments to the entire model weights, methods like adapters or pruning-based fine-tuning help to fine-tune the model in a parameter-efficient way.

5.2.1 Vanilla Fine-Tuning

In Vanilla fine-tuning, the model is adapted to downstream tasks based on task-specific loss [2]. The main drawback in vanilla fine-tuning is that PTLM having large parameters is prone to overfit on small task-specific datasets. Moreover, with small datasets, the model weights are not adapted well to the end task which limits its performance. Intermediate fine-tuning or multi-task fine-tuning overcome the issues in vanilla fine-tuning.

5.2.2 Intermediate Fine-Tuning (IFT)

IFT involves fine-tuning the model on an intermediate dataset with a large number of labeled instances. IFT helps the model to gain additional domain or task-specific knowledge which avoids overfitting and enhances its performance on small target datasets [214], [219], [220]. Poth et al. [220] established that intermediate pre-training can yield performance gains in adapter-based setups, similar to what has been previously found for full model finetuning. IFT can be domain adaptive [225] or task adaptive [214], [226]–[230].

Domain adaptive intermediate fine-tuning (DAIFT): DAIFT involves fine-tuning the model on the same domain dataset with a large number of labeled instances i.e., source and target datasets are of the same domain but different tasks. DAIFT on the same domain source dataset imparts more domain knowledge to the model which enhances the model performance on the same domain target task [225]. McCreery et al. [225] fine-tuned models like BERT and XLNet on the medical question-answer pairs dataset to enhance the performance on the Medical question similarity dataset. Here source (medical question-answer pair) dataset and target (medical question similarity) datasets are from the same domain i.e., Medical but from different tasks. The models (BERT and XLNet) are pretrained on a general domain text corpus. DAFT on medical domain dataset injects medical knowledge into BERT and XLNet which are pretrained on a general domain text corpus. McCreery et. al. [225] showed that the improvement is more when the number of training instances in the target task is less.

Task adaptive intermediate fine-tuning (TAIFT): TAIFT involves fine-tuning the model on the same or related task dataset with a large number of labeled instances i.e., source and target datasets are from the same or related task. Here the source and target datasets need not be from the same domain. TAIFT on the same or related task source dataset imparts more task-specific knowledge to the model which enhances the model performance on the target dataset. For example, Cengiz et al. [226] showed that TAIFT on general domain NLI datasets improves the in-domain model performance on the medical NLI dataset. Similarly, Yang et al. [227] and Wang et al. [228] achieved better performance on clinical STS dataset with TAIFT on general STS dataset. Yoon et al. [229] showed that TAIFT on the general domain SQUAD dataset improves the performance on the biomedical question answering dataset. Jeong et al. [230] improved BioBERT model performance in biomedical QA with TAIFT on the general NLI dataset. Phang et al. [214] achieved an improvement of 1.4 in GLUE score for BERT with TAIFT on the general NLI dataset. Further, the authors showed that the improvement is more when target labeled instances are less in number. TAIFT on NLI datasets imparts sentence-level reasoning skills to the model which improves the model performance in other tasks.

However, IFT does not guarantee better performance all the time [214], [219], [231] i.e., IFT sometimes negatively impacts the transferability to downstream tasks. Pruksachatkun et al. [219] performed a large-scale study on the pretrained RoBERTa model with 110 intermediate–target task combinations to investigate when and why IFT is beneficial. The authors showed that intermediate tasks requiring high-level inference and reasoning abilities tend to work best i.e., NLI and QA tasks that involve common sense reasoning are generally useful as intermediate tasks.

5.2.3 Multi-task Fine-Tuning (MTFT)

Multi-task learning (MTL) allows the model to learn knowledge that is useful across tasks. The primary focus of MTL can be improving the performance of target tasks with the help of auxiliary tasks or improving the performance of all the tasks [232]. The advantages of MTL are a) allows the model to gain more knowledge by learning from multiple datasets simultaneously which reduces the requirement of a large number of labeled instances in a specific target task. b) provides a regularization effect by avoiding overfitting to a specific target task [221]. Multi-task fine-tuning can be a) Vanilla MTFT b) Iterative MTFT and c) Hybrid.

Vanilla MTFT : Vanilla MTFT involves fine-tuning the model on multiple datasets simultaneously [221], [233]. For example, Liu et al. [221] improved the performance of BERT model in GLUE tasks using vanilla MTFT. Here, the embedding and transformer layers are shared across the tasks while each task has a task-specific layer. However, it is not guaranteed that Vanilla MTFT always improves the performance of the model across tasks [222]. For example, Mulyar et al. [222] developed MT-ClinicalBERT by fine-tuning ClinicalBERT using multiple datasets related to NER, STS, and RTE tasks. MT-ClinicalBERT achieved on par but less performance compared to task-specific ClinicalBERT models. The possible reason for this is that some of the tasks may negatively transfer knowledge which reduces the performance of the model. The drawback in vanilla MTFT can be avoided using Iterative MTFT which allows selecting the best set of tasks or MTFT followed by vanilla fine-tuning.

Iterative MTFT: Iterative MTFT allows to select the best set of tasks for fine-tuning the model [223]. It is necessary to select the best set of related datasets as vanilla MTFT on all the tasks sometimes may degrade the performance of the model [222]. Iterative MTFT is

similar to traditional feature selection in machine learning. Iterative MTFT helps to select the best set of datasets to fine-tune the model whereas feature selection helps to select the best set of features. Mahanjan et al. [223] applied iterative MTFT to choose the best set of related datasets and achieved SOTA results on the clinical STS dataset.

Hybrid MTFT: Iterative MTFT allows to choose the best set of related datasets, but it is expensive as it involves multiple iterations. Moreover, each iteration involves training the model on multiple datasets and then fine-tuning it on the target dataset. Instead of iteratively applying MTFT, we can fine-tune the model on multiple related datasets and then fine-tune it on the target dataset with a small learning rate [224]. We refer to this as hybrid MTFT as it involves vanilla MTFT followed by vanilla fine-tuning.

5.2.4 Parameter Efficient Fine-Tuning

Fine-tuning allows the model weights to adapt to downstream tasks by minimizing the task-specific loss i.e., fine-tuning starts with copying the entire model weights and making small changes. As fine-tuning involves updating the entire model weights, it is required to train a separate model for each task which is not parameter efficient. Adapters [234] and pruning-based fine-tuning [166] helps to fine-tune the model in a parameter-efficient way.

Adapters [234]– The adapter is a special trainable layer module proposed by Houlsby et al. [234] to fine-tune pretrained language models in a parameter-efficient way. The adapter module consists of two feed-forward layers with a non-linear layer in between and a skip connection. The adapter module projects the input vector into a small vector and then projects back into the original dimension using the two feed-forward layers and non-linear layer. Let x be the original vector dimension and y be the small vector dimension, then the total number parameters in the adapter module are $2xy + x + y$. By setting $x << y$, we can further reduce the number of parameters in the adapter module. The small vector dimension (y) provides a trade-off between performance and parameter efficiency. Adapters are added to each of the sublayers in transformer layer before layer normalization. During fine-tuning, only parameters of adapters, layer normalization in each transformer layer, and task-specific layers are only updated while the rest of the parameters in pretrained model are kept frozen.

Houlsby et al. [234] showed that adapter-based fine-tuning is highly parameter efficient and they can achieve the performance of a fully fine-tuned model using adapter-based fine-tuning which involves only 3% of task-specific parameters. Moreover, Poth et al. [220] showed that intermediate fine-tuning using adapters improve model performance in the target task. Stickland and Murray [235] proposed an approach to train adapters in a multi-task setting. However, this approach suffers from issues like a) requirement of simultaneous access to multiple datasets and b) difficulty in balancing various tasks as the model may overfit on low resource tasks and underfit on high resource tasks. Pfeiffer et al. [236] proposed AdapterFusion a novel two-stage method based on adapters that overcome the issues in sequential learning and multi-task learning to leverage knowledge from multiple tasks. They showed that AdapterFusion outperforms full fine-tuning as well the adapter-based model trained in single and multi-task setups.

Pruning-based fine-tuning - Recent studies [164]–[166], [237], [238] show that deep pre-trained language models have redundancy. Pruning methods are based on the idea that not all the parameters are important in the pre-trained models and some of them can be removed without much impact on the model performance. For example, research studies [164], [165], [237] show that some of the attention heads can be pruned. Sajjad et al. [166] proposed different strategies to drop encoder layers in pretrained language models. The authors showed that the size of the pre-trained models can be reduced by dropping encoder layers after pre-training and the resulting pruned BERT model can be fine-tuned to get competitive performance. The experimental results show that it is possible to prune BERT, RoBERTa, and XLNet models by up to 40% while maintaining up to 98% of their original performance.

5.3 Prompt-based Tuning

In general, most of the P-TLMs are pretrained using language modeling objectives and then adapted to downstream tasks using fine-tuning which involves task-specific objectives. The discrepancy in objectives during pretraining and fine-tuning impacts the downstream performance of the model. The downstream performance of models can be improved especially in few-short and zero-shot settings by prompt-based tuning which formulates the tuning process as a slot filling which is close to the language modeling objective. Here the prompt can be close or pre-fix shape and it can be generated manually or automatically [239]. Close style prompts are suitable for models pretrained masked language modeling objective while pre-fix style prompts are suitable for models pretrained using casual modeling objective.

Prompt-based tuning initially is based on manually created prompts. For example, LAMA [240] probe is conducting manually created close-style prompts while GPT-3 [27] model is tuned using manually created pre-fix style prompts. As manually creating prompts is a time-taking process and these prompts can be sub-optimal also [241]. To over these drawbacks prompts are created automatically. Automatically generated prompts can be discrete or continuous. A discrete prompt is simply a string i.e., a sequence of words included in the input text to guide the T-PTLM to better model the downstream task. Some of the popular methods to generate discrete prompts are Prompt mining [241], Prompt generation [242], Prompt paraphrasing [241], [243], and Gradient-based search [244], [245].

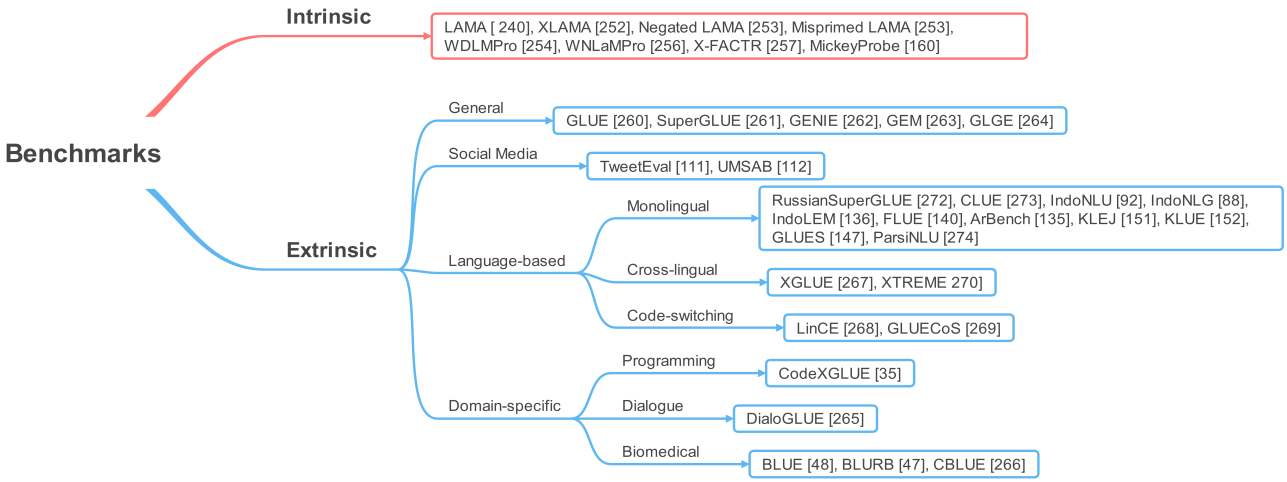

Fig. 11: Benchmarks to evaluate the progress in T-PTLMs

Prompt mining [241] involves collecting a large number of sentences having the subject x and object y and then generate the new prompts using the middle words or the dependency paths. Prompt generation [242] involves generating prompts using T-PTLMs like T5. Prompt paraphrasing generates new prompts from seed prompts using methods like back translation [241] or equivalent phrases from thesaurus [243]. Gradient-based search generates trigger tokens which can be combined with input sequence to create a prompt [244], [245].

Continuous prompts perform prompting in the embedding space of T-PTLM i.e., add a sequence of task-specific vectors to the input sequence. Here the prompt vectors need not be embeddings of natural language words. Unlike discrete prompts where the template parameters are determined by T-PTLM parameters, in continuous prompts, the templates have their parameters independent of T-PTLM parameters. Some of the popular continuous prompt generating approaches are Pretuning [246], P-Tuning [247], and Prompt-tuning [248]. Further Lester et al. [248] showed that prompt ensembling outperforms traditional model ensembling. Here prompt ensembling means generating multiple prompts for the same task.

6 Evaluation

A T-PTLM gains knowledge encoded in pretraining corpus during pretraining. Here the knowledge refers to syntactic, semantic, factual, and common-sense knowledge. The effectiveness of a T-PTLM can be evaluated in two ways namely intrinsic and extrinsic (refer Figure 11). Intrinsic evaluation probes the knowledge encode in T-PLTM while extrinsic evaluation evaluates how effective the T-PTLMs are in real-world downstream tasks. Intrinsic evaluation sheds light on the knowledge gained by T-PTLM during pretraining which helps us to design better pretraining tasks so that the model learns more knowledge during the pretraining stage itself.

6.1 Intrinsic Evaluation

Intrinsic evaluation involves probing the model knowledge using probes like LAMA [240], XLAMA [252], X-FACTR [257], MickeyProbe [160] , Negated LAMA [253], Misprimed LAMA [253], WDLMPro [254] or WNLaMPro [256] (refer Table 8). LAMA is one of the first probes introduced to evaluate factual and common-sense knowledge in PLTMs under zero-shot settings. LAMA consists of a corpus of facts where the fact can be a relation triplet or a question-answer pair gathered from SQUAD. Here facts are converted to fill-in-the-blank style questions and the model is evaluated based on the prediction of blank tokens i.e., *argmax* $x \in W$ *P(x/temp)* represents the model vocabulary and $temp$ represents the fill-in-the-blank template. LAMA is based on the hypothesis that a model with a good amount of factual knowledge correctly predicts the blank tokens i.e., i.e., the ground truth tokens are predicted with the highest probability compared to other tokens in the model vocabulary. Negated LAMA and Misprimed LAMA probe shows that the language models are not able to consider the negated or misprimed words in the templates. For example, the model predicts the same token whether the templated is negated or not. Poerner et al. (2020) [186] introduced LAMA-UHN which is a collection of triples from the LAMA probing benchmark which are difficult to guess.

The main drawbacks in the LAMA probe are a) restriction to single token entities only b) limits the prediction of tokens over the model vocabulary which hinders the evaluation of models with different vocabulary c) it probes only English language models and d) many of the triples in LAMA are easy to guess [186]. XLAMA

Name	Probe	Language	Method	Includes	Data Source
LAMA [240]	Factual and Commonsense knowledge	English	UnTQ	Single token entities	TREx [249], GoogleRE, ConceptNet [250] and SQUAD [251]
XLAMA [252]	Factual knowledge	Multilingual (53 languages)	TQ (Typed Query)	Single and Multi token entities	TREx [249] and GoogleRE
Negated LAMA [253]	Impact of negation in probing factual and common-sense knowledge	English	UnTQ	Single token entities	TRex [249], GoogleRE, ConceptNet [250]and SQUAD [251]
Misprimed LAMA [253]	Impact of mis primes in probing factual and common-sense knowledge	English	UnTQ	Single token entities	TRex [249], GoogleRE, ConceptNet [250]and SQUAD [251]
WDLMPro [254]	Word Understanding	English	Ranking	Single and Multi-token entities	WordNet [255]
WNLaMPro [256]	Word Understanding	English	UnTQ	Single and Multi-token entities	WordNet [255]
X-FACTR [257]	Factual knowledge	Multilingual (26 languages)	UnTQ	Single and Multi-token entities	TREx [249]
MickeyProbe [160]	Common-sense knowledge	Multilingual (11 languages)	Sentence ranking	-	OMCS Corpus [258]

TABLE 8. Summary of various intrinsic benchmarks.

extends the LAMA probe to multiple languages (53 languages) and includes multi-token entities also. Moreover, in LAMA the model has to predict over the entire model vocabulary while in XLAMA the model has to predict over a fixed set of candidates specific to each relation type i.e., *argmax x∈C P(x/temp)* where C represents a set of candidate entities specific to a relation type. The authors of XLAMA refer to this type of querying as Typed Query (TQ) and the querying in LAMA as UnTyped Query (UnTQ). Similar to XLAMA, X-FACTR is also a multi-lingual probe for 23 languages. Moreover, the authors of X-FACTR developed several decoding algorithms to predict multi-token entities. MickeyProbe [160] is a zero-shot common-sense probe that uses sentence-level ranking based on Pseudo-Likelikhood [259]. Here the model has to rank a set of declarative sentences having similar words and syntactic features. The performance of multilingual models in retrieving knowledge varies with language i.e., it is high in high resource languages compared to low resource languages [160], [252], [257] . Moreover, the multilingual model exhibits language bias i.e., the language of query affects the model prediction [252]. The model performance in retrieving knowledge can be improved by pretraining on cod-switched data [257] or further pretraining using a multilingual contrastive loss function [160].

Probes like LAMA, XLAMA, and X-FACTR focus on evaluating the relations between entities. Unlike these probes, WDLMPro and WNLaMPro focus on understanding how the pretrained models understand the words. WNLamPro uses fill-in-the-black style templates while WDLMPro evaluates the model by matching a word with its definition. WDLMPro probe is based on the assumption that a model correctly matches a word with its definition only when the model understands the word. WDLMPro consists of synset groups where each group consists of a word, its taxonomic sister words from WordNet along their definitions.

6.2 Extrinsic Evaluation

Extrinsic evaluation helps to assess the performance of a model in downstream tasks. To get the maximum out of a model, the model should perform well across a wide range of tasks rather than just performing well on one or two tasks. A benchmark provides a standard way of evaluating the model's generalization ability across tasks. A benchmark usually consists of a set of datasets, a leader board, and a single metric [260]. The datasets are chosen in a way that they are challenging and represent diverse tasks. A leaderboard is an online repository that helps to compare and rank models. For a model to achieve a good score in a benchmark, it should share knowledge i.e., parameters across tasks with one or two layers specific to each task [260]. A benchmark uses a single metric to evaluate the overall performance of the model across tasks. Without a benchmark, it is difficult to evaluate models in a standard way and track the progress in the development of pretrained language models. A summary of various extrinsic bechmarks are presented in Tables 9 and 10.

GLUE [260] and SuperGLUE [261] benchmarks are the commonly used benchmarks to evaluate the natural language understanding ability of pretrained language models. GLUE benchmark consists of nine tasks which include both single sentence and sentence pair tasks. With rapid progress in model development, the models achieved good performance in the GLUE benchmark resulting in little space for further improvement [261].

Benchmark	Type	Category	Language	Public Leaderboard	Diagnostic Dataset	Details
GLUE [260]	NLU	General	English	✓	✓	Five NLU tasks (TC, SA, STS, PI, and NLI) with nine datasets and one diagnostic dataset.
SuperGLUE [261]	NLU	General	English	✓	✓	Five NLU tasks (QA, WSD, NLI, and Coref) with eight datasets and two diagnostic datasets.
GENIE [262]	NLG	General	English	✓	✗	Four NLG tasks (MT, TS, MRC, and CSR).
GEM [263]	NLG	General	English	✓	✗	Four NLG tasks (Data2text, TS, TSim, and Dialog) with thirteen data sets.
GLGE [264]	NLG	General	English	✓	✗	Eight language generation tasks, including Abstractive TS, Answer-aware Question Generation, Conversational QA, and Personalizing Dialogue.
TweetEval [111]	NLU	Social-media	English	✓	✗	Seven tweets related tasks, and all are framed as multi-class tweet classification.
UMSAB [112]	XLU	Social-media	Cross-lingual	✓	✗	Cross-lingual sentiment analysis for eight different languages and all are framed as tweet classification with three labels (positive, negative, and neutral).
CodeXGLUE [35]	PLU and PLG	Domain-specific	Programming	✓	✗	Ten programming language understanding and generation tasks with fourteen datasets.
DialoGLUE [265]	NLU	Domain-specific (Dialogue)	English	✓	✗	Four NLU tasks (Intent Prediction, Slot-filling, Semantic parsing, and Dialogue state tracking) with seven task-oriented dialogue datasets.
BLUE [48]	NLU	Domain-specific (biomedical)	English	✓	✗	Five tasks (STS, NER, RE, NLI and DC) with ten datasets that cover both biomedical and clinical texts with different dataset sizes and difficulties.
BLURB [47]	NLU	Domain-specific (biomedical)	English	✓	✗	Thirteen biomedical NLP datasets in 6 tasks (NER, PICO, RE, STS, DC, and QA).
CBLUE [266]	NLU	Domain-specific (biomedical)	Chinese	✓	✗	Includes tasks like NER, PI, QA, IR, IC, and ToC.

TABLE 9. Summary of general, social-media and domain-specific Extrinsic Benchmarks. TC - Text Classification, STS- Semantic Text Similarity, TS - Text Summarization, TSim – Text Simplification, ToC – Topic Classification, IC – Intent Classification, IR – Information Retrieval, QA- Question Answering, DC- Document Classification, RE – Relation Extraction, POS - Parts-of-speech tagging, DP - Dependency Parsing, MRC - Machine Reading Comprehension, SA – Sentiment Analysis, PI- Paraphrase Identification, NLI – Natural Language Inference, WPR – Web Page Ranking, QAM – Question Matching, QADSM – Query Ad Matching, MT- Machine Translation, LID – Language Identification, CSR – Common Sense Reasoning, WSD – Word Sense Disambiguation, MCQA – Multiple Choice Question Answering.

To have a more challenging benchmark, the SuperGLUE benchmark is introduced with more challenging tasks like QA, word sense disambiguation (WSD), and coreference resolution while retaining the two difficult tasks from GLUE benchmark.

Inspired by the success of GLUE and SuperGLUE benchmarks in the general English domain, benchmarks like GENIE [262], GEM [263], GLGE [264] have been introduced to evaluate NLG models in the general English domain. To evaluate cross-lingual models, XGLUE [267] and XTREME [270] benchmarks have been introduced. XTREME benchmark includes only XNLU tasks while the XGLUE benchmark includes both XNLU and XNLG tasks. Moreover, the XGLUE benchmark includes diverse datasets related to search, ads, and news scenarios which makes it more challenging and practical. Recently, as there is less room for improvement in the XTREME benchmark with existing achieving improvements by almost 13 points, Ruder et al. [271] extended XTREME to XTREME-R which consists of ten challenging NLU tasks. Moreover, XTREME covers only forty languages while XTREME-R covers 50 languages.

To evaluate social media-based T-PTLMs, we have benchmarks like TweetEval [111] and UMSAB [112]. TweetEval includes datasets from English only while UMSAB includes datasets from eight languages including English. In both the benchmarks, all the tasks are framed as tweet classification. Apart from XGLUE and

Benchmark	Type	Category	Language	Public Leaderboard	Diagnostic Dataset	Details
XGLUE [267]	XLU and XLG	Language-based	Cross-lingual	✓	✗	Eleven tasks in which nine tasks are XNLU (NER, POS, QA, NLI, PI, WPR, QAM, QADSM and TC) and two tasks are XNLG (question and news title generation). This benchmark covers 19 languages.
LinCE [268]	NLU and NLG	Language-based	Code-Switching	✓	✗	Five tasks (MT, LID, NER, POS, and SA) with eighteen datasets covering nine different code-switched language pairs.
GLUECoS [269]	NLU	Language-based	Code-Switching	✓	✗	Eleven datasets covering six tasks (LID, POS, NER, SA, QA and NLI) and two language pairs (English-Hindi and English-Spanish).
XTREME [270]	XLU	Language-based	Cross-lingual	✓	✗	Nine tasks spanning forty typologically diverse languages from 12 language families.
XTREME-R [271]	XLU	Language-based	Cross-lingual	✓	✓	Includes ten challenging NLU tasks for 50 languages.
RussianSuperGLUE [272]	NLU	Language-based	Russian	✓	✗	Nine Russian NLU tasks.
IndicGLUE [98]	NLU	Language-based	Indian languages	✓	✗	Ten tasks covering multiple Indian languages.
CLUE [273]	NLU	Language-based	Chinese	✓	✓	Nine language understanding tasks in Chinese and diagnostic dataset for linguistic analysis.
IndoNLU [92]	NLU	Language-based	Indonesian	✓	✗	Twelve tasks clustered into four categories: (a) single-sentence classification, (b) single-sentence sequence tagging, (c) sentence-pair classification, and (d) sentence-pair sequence labelling.
IndoNLG [88]	NLG	Language-based	Indonesian	✗	✗	Six commonly used NLG tasks: TS, QA, Chitchat, and three different pairs of machine translation (MT) tasks.
IndoLEM [136]	NLU	Language-based	Indonesian	✓	✗	Seven tasks for the Indonesian language spanning Morpho-syntax and Sequence labelling, Semantics, and Discourse with eight datasets.
FLUE [140]	NLU and XLU	Language-based	French	✗	✗	Six tasks (TC, PI, NLI, WSD, DP, and POS). Three out of six tasks (TC, PI, and NLI) are from cross-lingual datasets.
ArBench [135]	NLU	Language-based	Arabic	✓	✗	Six different Arabic language understanding tasks (SA, Social meaning tasks, ToC, Dialect identification, NER, and QA) with 42 datasets.
KLEJ [151]	NLU	Language-based	Polish	✓	✗	Seven tasks (NER, Semantic relatedness, QA, TE, SA, and Cyberbully detection) with 9 datasets.
KLUE [152]	NLU	Language-based	Korean	✓	✗	Eight Korean natural language understanding tasks, including ToC, STS, NLI, NER, RE, DP, MRC, and Dialogue state tracking.
GLUES [147]	NLU	Language-based	Spanish	✗	✗	Includes tasks like NLI, PI, NER, POS, DC, DP and QA.
ParsiNLU [274]	NLU	Language-based	Persian	✓	✗	Includes six NLU tasks like TE, PI, SA, MT, MRC, MCQA.

TABLE 10. Summary of language-based extrinsic benchmarks. TC - Text Classification, STS- Semantic Text Similarity, TS - Text Summarization, TSim – Text Simplification, ToC – Topic Classification, IC – Intent Classification, IR – Information Retrieval, QA- Question Answering, DC- Document Classification, RE – Relation Extraction, POS - Parts-of-speech tagging, DP - Dependency Parsing, MRC - Machine Reading Comprehension, SA – Sentiment Analysis, PI- Paraphrase Identification, NLI – Natural Language Inference, WPR – Web Page Ranking, QAM – Question Matching, QADSM – Query Ad Matching, MT- Machine Translation, LID – Language Identification, CSR – Common Sense Reasoning, WSD – Word Sense Disambiguation, MCQA – Multiple Choice Question Answering.

XTREME which evaluate cross-lingual models, we have separate benchmarks in each language like Russian (RussianSuperGLUE [272]), Indian (IndicGLUE [98]), Chinese (CLUE [273]), Indonesian (IndoNLU [92], IndoNLG [88], IndoLEM [136]), French (FLUE [140]), Arabic (ArLUE [135]), Polish (KLEJ [151]), Korean (KLUE [152]) Spanish (GLUES [147]), and Persian (ParsiNLU [274]) to evaluate monolingual language models. Besides, we have benchmarks like GLUECoS [269] and LinCE [268] for CodeSwitching, BLUE [48], BLURB [47] and Chinese-BLUE [266] in the Biomedical domain, CodeXGLUE [35] in the Code intelligence domain and DialogGLUE [265] to evaluate Dialog models. Further, we have benchmarks like FewCLUE [275], FLEX [276], and FewGLUE [277] to evaluate T-PTLMs under few shot settings.

7 Useful Libraries

We present a summary of popular libraries to work with transformer-based PTLMs. Libraries like Transformers [278] and Fairseq [279] are useful for model training and evaluation. Some of the libraries like SimpleTransformers, HappyTransformer, AdaptNLP which are built on the top of Transformers library make the model training and evaluation easier with just a few lines of code. Libraries like FastSeq [280], DeepSpeed [281], FastT5, OnnxT5 and LightSeq [282] are useful to increase the inference speed of models. Ecco, BertViz [283], and exBERT [284] are visual analysis tools to explore the layers of transformer models while Transformers-interpret and Captum help to explain the model decisions.

8 Discussions and Future Directions

8.1 Better Pretraining Methods

It is highly expensive to pretrain a model especially large-scale models with billions or trillions of parameters using SSL only. Novel pretrained methods like Knowledge Inherited Pretraining (KIPT) involve both SSL and Knowledge Distillation [72]. SSL allows the model to learn the knowledge available in pretraining corpus while KD allows the model to learn the knowledge already encoded in existing pretrained models. Due to the additional knowledge gained by the model during pretraining through KD, a) the model converges faster and hence reduces the pretraining time b) the model performs better in downstream tasks compared to the models pretrained using SSL only [72]. The research community must focus more on developing better pretraining methods like KIPT which allow the model to gain more knowledge as well as reduce the pretraining time.

8.2 Sample Efficient Pretraining Tasks

A pretraining task is sample efficient if it makes maximum out of each train instance i.e., it should be defined over all the tokens in the training instance. Sample efficient pretraining tasks make pretraining more compute efficient [5]. MLM, the most commonly used pretraining task is less sample efficient as it involves only a subset of tokens i.e., masked tokens which amount to 15% of total tokens [2], [5]. Pretraining tasks like RTD [5], RTS [56], and STD [55] can be considered as early attempts to develop sample-efficient pretraining tasks. All these three pretraining tasks are defined over all the tokens in each training instance i.e., they involve identifying whether each token is replaced [5], randomly substituted [56], or shuffled [55] or not. We can expect more sample efficient pretraining tasks which make pretraining more compute efficient.

8.3 Efficient Models

Pretraining T-PLMs is highly expensive due to the large model size and also there is a requirement of large volumes of unlabelled text data. However long pretraining times are not environmentally friendly due to CO2 emission and the availability of large volumes of unlabelled text data is not possible in all the domains like Biomedical. Recently, models like DeBERTa [212] with novel improvements to BERT model achieve better performance than RoBERTa model even though it is pretrained using just 78GB of data which is just half of the data used to pretrain RoBERTa model. Similarly, ConvBERT [213] with a novel mixed attention module outperforms ELECTRA model using just ¼ of its pretraining cost. There is a great need for efficient models like DeBERTa and ConvBERT to reduce the amount of pretraining data as well as the pretraining costs.

8.4 Better Position Encoding Mechanisms

The self-attention mechanism is permutation invariant without position bias. The position bias can be provided using absolute or relative position embeddings. Moreover, absolute position embeddings can be predetermined or learned. However, there are drawbacks to both these approaches [288]. Absolute position embeddings suffer from generalization issues but are easy to implement. Unlike absolute positions, relative position embeddings are robust to sequence length changes but difficult to implement and yield less performance. There is a great need for more novel position encoding mechanisms like CAPE [288] which combines the advantages in both absolute and relative position embeddings.

8.5 Improving existing T-PTLMs

T-PTLMs like BERT and RoBERTa have achieved good results in many of the NLP tasks. Recent research works showed that these models can be further improved by injecting sentence-level semantics through continual pretraining based on adversarial [55] or contrastive pretraining tasks [157], [160] . For example, Panda et al. [55] showed that continual pretraining using shuffled token detection objective improves RoBERTa model performance in GLUE tasks by allowing the model to

Library	Purpose	Description	Framework	Link
Transformers [278]	Training and Inference	State-of-the-art library for transformer based PTLMs.	Pytorch, Tensorflow and Jax	https://github.com/huggingface/transformers
SimpleTransformers	Training and Inference	Built on the top of transformers and lets you to quickly train and evaluate models.	PyTorch	https://github.com/ThilinaRajapakse/simpletransformers
HappyTransformer	Training and Inference	Built on the top of transformers and makes the use of state-of-the-art models easy.	PyTorch	https://github.com/EricFillion/happy-transformer
FairSeq [279]	Training and Inference	Library to train custom models for translation, summarization, language modeling and other text generation tasks.	PyTorch	https://github.com/pytorch/fairseq
AdaptNLP	Training and Inference	Built on the top of Flair and Transformers library and makes the use of state-of-the-art models easy.	PyTorch	https://github.com/Novetta/adaptnlp
SimpleT5	Training and Inference	Built on top of PyTorch-lightning and Transformers that lets you quickly train your T5 models.	PyTorch-Lightning	https://github.com/Shivanandroy/simpleT5
SpacyTransformers	All NLP tasks	spaCy pipelines for pretrained BERT, RoBERTa XLNet, GPT-2 etc.	PyTorch	https://github.com/explosion/spacy-transformers
TextBox	Text Generation	Library for building text generation systems based on models like GPT-2, BART, T5 etc.	PyTorch	https://github.com/RUCAIBox/TextBox
Trankit	Multilingual NLP	Light-Weight Transformer-based Python Toolkit for Multilingual Natural Language Processing and is built on the top of transformers library.	PyTorch	https://github.com/nlp-uoregon/trankit
Haystack	Information Retrieval	Library to build powerful and production-ready pipelines for different search use cases.	PyTorch	https://github.com/deepset-ai/haystack
EasyNMT	Machine Translation	Easy to use, state-of-the-art Neural Machine Translation library for 100+ languages.	PyTorch	https://github.com/UKPLab/EasyNMT
Aitextgen	Text Generation	Library for training and generation using OpenAI's GPT-2 and EleutherAI's GPT Neo/GPT-3 architecture.	PyTorch-Lightning	https://github.com/minimaxir/aitextgen
Dl-Translate	Machine Translation	Deep Learning-based translation library built on Huggingface transformers.	PyTorch	https://github.com/xhlulu/dl-translate
FastSeq [280]	Fast Inference	Efficient implementation of the popular sequence models for text generation, summarization, and translation tasks.	PyTorch	https://github.com/microsoft/fastseq
LightSeq [282]	Fast Inference	High performance training and inference library for sequence processing and generation.	PyTorch, Tensorflow	https://github.com/bytedance/lightseq
TurboTransformers [285]	Fast Inference	A library open source by WeChat AI to get fast inference using transformer models.	PyTorch	https://github.com/Tencent/TurboTransformers
EET	Fast Inference	PyTorch library to make transformer models inference faster.	PyTorch	https://github.com/NetEase-FuXi/EET
DeepSpeed [281]	Distributed Model Training	Deep learning optimization library that makes distributed training easy, efficient, and effective.	PyTorch	https://github.com/microsoft/DeepSpeed
FastT5	Fast Inference	Reduce T5 model size by 3X and increase the inference speed up to 5X.	PyTorch	https://github.com/topics/fastt5
OnnxT5	Fast Inference	Fast Inference of T5 model.	PyTorch	https://github.com/abelriboulot/onnxt5
exBERT [284]	Visualization	Library to explore the learned attention weights and contextual representations.	PyTorch	https://github.com/bhoov/exbert
BertViz [283]	Visualizaation	Library to visualize attention in the Transformer model.	PyTorch	https://github.com/jessevig/bertviz
Transfomers-interpret	Model Interpretation	Library to explain the decision of transformer models.	PyTorch	https://github.com/cdpierse/transformers-interpret
Ecco	Visualization	Library to visualize and explore NLP language models.	PyTorch	https://github.com/jalammar/ecco
Captum	Model Interpretation	PyTorch interpretation library.	PyTorch	https://github.com/pytorch/captum
TextBrewer [286]	Model Compression	Supports Knowledge distillation methods.	PyTorch	https://github.com/airaria/TextBrewer
KD-Lib [287]	Model Compression	Library to develop compact model using model compression techniques like quantization, pruning and knowledge distillation.	PyTorch	https://github.com/SforAiDl/KD_Lib
Parallelformers	Model Parallelization	An Efficient Model Parallelization Toolkit for Deployment (inference).	PyTorch	https://github.com/tunib-ai/parallelformers

TABLE 11. Useful Libraries to work with T-PTLMs

learn more coherent sentence representations. Similarly, continual pretraining using contrastive pretraining objectives improves the performance of T-PTLMs in GLUE tasks [157] and multilingual T-PTLMs in Mickey Probe [160]. Further research is required to extend this to other monolingual and domain-specific T-PTLMs.

8.6 Beyond Vanilla Fine-tuning

Fine-tuning is the most commonly used method to adapt pretrained models to downstream tasks. However, the main drawback with vanilla fine-tuning is that it makes changes to all the layers in the pretrained model, and hence it requires maintaining a separate copy for each task which makes deployment expensive. Methods like Adapters [234] and Pruning-based tuning [166] are proposed to adapt pretrained models to downstream tasks in a parameter-efficient way. For example, adapters are small task-specific layers added to each transformer layer and during downstream task adaptation, only adapter layer parameters are updated while keeping the transformer layer parameter fixed. Moreover, Poth et al. [220] showed that adapters are useful for intermediate fine-tuning also. Recently prompt-based tuning methods (discrete – [27], [242], [244] and continuous – [246], [248]) have attracted the research community with much better parameter efficiency. For example, prompt-based tuning methods like Prefix-tuning [246] require only 0.1% of task-specific parameters while adapter-based fine-tuning involves 3% of task-specific parameters [234].

8.7 Benchmarks

In the last four layers many benchmarks have been introduced to evaluate the progress in pretrained models in general [260]–[264] as well as in specific domains [35], [47], [48], [111], [265], [266]. Apart from English, benchmarks are introduced to evaluate the progress in other monolingual [88], [92], [135], [136], [140], [147], [151], [152], [272]–[274] as well as multilingual models [112], [267], [270]. However, the existing benchmarks are not sufficient to cover all the scenarios. For example, there are no benchmarks to evaluate a) the progress in compact pretrained models b) the robustness of pretrained models c) PTLMs specific to social media as well as specific to other domains like Academic. Recently, leaderboards like Explainboard [289] which not only evaluate the progress using a single metric like existing benchmarks but also dig deeper by analyzing the strengths and weaknesses of models are introduced. This kind of leaderboard should be extended to other domains also. Moreover, benchmarks like FewGLUE [277], FLEX [276], and FewCLUE [275] which evaluate few-shot learning techniques should be extended to other languages and domains also.

8.8 Compact Models

Transformer-based PTLMs achieved state-of-the-art results in almost every NLP task. However, these models are large which requires more amount of storage space. As these models have many layers through which the input has to pass through to get the model prediction, latency is high [162]. As real-world applications are resource-constrained and require less latency, model compression methods like pruning, quantization, knowledge distillation, parameter sharing, and factorization are explored to develop compact models in English for general domain applications [162], [290]. There is a great need to explore these model compression methods to develop compact models for other languages as well as for other domains also.

8.9 Robustness to Noise

Transformer-based PTLMs are brittle to noise which includes both adversarial and natural noise [291], [292] . The main reason behind this is the use of sub-word embeddings. In the case of sub-word embeddings, even a small typo error can change the overall representation of the word by breaking the word into many sub-word tokens which hinders model learning and impact the model predictions [66], [121]. To increase the robustness of PTLMs to noise, models like CharacterBERT [66] use character embeddings only while models like CharBERT [121] use character embeddings along with sub-word embeddings. Both these approaches improved the robustness to noise. Recently, tokenization-free models like CANINE [67], ByT5 [68], and Charformer [69] are proposed which further improve robustness to noise. There is a need for more robust models to increase the use of PTLMs in real-world applications especially in sensitive domains like Medicine.

8.10 Novel Adaptation Methods

The commonly used strategy to adapt general models to specific domains like biomedical or multilingual models to specific languages is continual pretraining [45], [46], [48]. Although this approach achieves good results by adapting the model to a specific domain or language, the lack of domain or language-specific vocabulary hurts the model downstream performance. Recently researchers proposed methods like vocabulary expansion [184], vocabulary expansion and then continual pretraining [185], [293]. These methods overcome the issue of OOV words but increase the size of vocabulary due to the addition of new terms in the vocabulary. Recently, Yao et al. [78] proposed the Adapt and Distill approach to adapt general models to a specific domain using vocabulary expansion and knowledge distillation. Different from existing adaptation methods, this approach not only adapts general models to specific domain but also reduces the size of the model. Further research on this topic will result in more novel adaption methods.

8.11 Privacy Issues

Transformer-based PTLMs achieved impressive results in many of the NLP tasks. However, there are some

unexpected as well as unwanted risks associated with these models. For example, data leakage from these models is of primary concern especially when the model is pretrained over private data. As the model is pretrained over a large amount of text data, it is possible to recover sensitive data including personally identifiable information [294]–[297]. This prevents the public release of models pretrained on private data. Recently, Carlini et al. [295] showed that GPT-2 model generates the entire postal address of a person which is included in training data when prompted with the person's name. Recently frameworks like KART [294] are introduced in the Biomedical domain which performs various attacks to assess data leakage. There is a great need to develop more sophisticated attacks to assess data leakage and also methods to prevent leakage of sensitive data from pretrained models.

8.12 Mitigating Bias

Deep learning-based models are increasingly used in many real-world applications including specific domains like Biomedical [298] and Legal [299]. However, these models are prone to learn and amplify the bias already present in training data. As a result, the decisions from these models are biased i.e., may favor a particular race, gender, or aged people. This behavior is completely undesirable. Some of the recent works focused on identifying and mitigating bias. For example, Minot et al. [182] proposed a data augmentation-based approach to reduce gender bias while Liang et al. [300] proposed A-INLP approach which dynamically identifies bias-sensitive tokens. Further research in this area helps to mitigate bias in pretrained models and help them to make fair decisions.

8.13 Mitigating Fine-Tuning Instabilities

Fine-tuning is the most widely adopted approach to adapt PTLMs to the downstream task. Though fine-tuning achieves good performance, it is unstable i.e., fine-tuning the model with different random seeds results in the large variance of downstream performance. It is believed that catastrophic forgetting and the small size of datasets are possible reasons for fine-tuning instabilities [2], [301], [302]. However, Mosbach et al. [303] showed that fine-tuning instability is not caused by any of these two and further showed that fine-tuning instability can be attributed to a) optimization difficulties which lead to vanishing gradients and b) generalization issues. The possible solutions to mitigate fine-tuning instability are a) intermediate fine-tuning [214] b) mixout [301] c) smaller learning rates in early epochs and fine-tuning the model for more number of epochs [303] and d) use of supervised contrastive loss along with cross-entropy loss [304]. Further work related to this will make fine-tuning more stable.

9 Conclusion

In this survey paper, we present a comprehensive review of recent research works in transformer-based pretrained language models. This paper covers various pretraining methods, pretraining tasks, embeddings, downstream adaptation methods, intrinsic and extrinsic benchmarks, useful libraries to work with T-PTLMs. We also present a new taxonomy to categorize various T-PTLMs. We discuss various future research directions which will direct the research community to further improve T-PTLMs.

Acknowledgments

Kalyan would like to thank his father Katikapalli Subramanyam for giving a) $750 to buy a new laptop, 24-inch monitor and study table. b) $180 for one year subscription of Medium, Overleaf and Edraw MindMaster software. Edraw MindMaster is used to create all the diagrams in the paper.

References

[1] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[3] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in Neural Information Processing Systems*, vol. 32, pp. 5753–5763, 2019.

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[5] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," in *International Conference on Learning Representations*, 2019.

[6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[7] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2019.

[8] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.

[9] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 328–11 339.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[12] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 655–665.

[13] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016, pp. 2873–2879.

[14] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 3485–3495.

[15] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[19] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[22] T. Kaur and T. K. Gandhi, "Automated brain image classification based on vgg-16 and transfer learning," in *2019 International Conference on Information Technology (ICIT)*. IEEE, 2019, pp. 94–98.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[24] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, pp. 1–26, 2020.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[26] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[27] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[28] W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang, K. Wang, X. Zhang *et al.*, "Pangu: Large-scale autoregressive pretrained chinese language models with auto-parallel computation," *arXiv preprint arXiv:2104.12369*, 2021.

[29] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," *arXiv preprint arXiv:2006.16668*, 2020.

[30] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *arXiv preprint arXiv:2101.03961*, 2021.

[31] Y. Yang, M. C. S. Uy, and A. Huang, "Finbert: A pretrained language model for financial communications," *arXiv preprint arXiv:2006.08097*, 2020.

[32] S. Leivaditi, J. Rossi, and E. Kanoulas, "A benchmark for lease contract review," *arXiv preprint arXiv:2010.10386*, 2020.

[33] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "Legal-bert: The muppets straight out of law school," *arXiv preprint arXiv:2010.02559*, 2020.

[34] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8342–8360.

[35] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang *et al.*, "Codexglue: A machine learning benchmark dataset for code understanding and generation," *arXiv preprint arXiv:2102.04664*, 2021.

[36] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang *et al.*, "Codebert: A pre-trained model for programming and natural languages," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 1536–1547.

[37] W. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "Unified pre-training for program understanding and generation," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 2655–2668.

[38] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu *et al.*, "Graphcodebert: Pre-training code representations with data flow," *arXiv preprint arXiv:2009.08366*, 2020.

[39] L. Phan, H. Tran, D. Le, H. Nguyen, J. Anibal, A. Peltekian, and Y. Ye, "Cotext: Multi-task learning with code-text transformer," *arXiv preprint arXiv:2105.08645*, 2021.

[40] C.-S. Wu, S. C. Hoi, R. Socher, and C. Xiong, "Tod-bert: Pre-trained natural language understanding for task-oriented dialogue," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 917–929.

[41] A. Louis, "Netbert: A pre-trained language representation model for computer networking," Ph.D. dissertation, Cisco Systems, 2020.

[42] X. Liu, D. Yin, X. Zhang, K. Su, K. Wu, H. Yang, and J. Tang, "Oag-bert: Pre-train heterogeneous entity-augmented academic language models," *arXiv preprint arXiv:2103.02410*, 2021.

[43] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3606–3611.

[44] S. Peng, K. Yuan, L. Gao, and Z. Tang, "Mathbert: A pre-trained model for mathematical formula understanding," *arXiv preprint arXiv:2105.00377*, 2021.

[45] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[46] E. Alsentzer, J. Murphy, W. Boag, W.-H. Weng, D. Jindi, T. Naumann, and M. McDermott, "Publicly available clinical bert embeddings," in *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, 2019, pp. 72–78.

[47] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *arXiv preprint arXiv:2007.15779*, 2020.

[48] Y. Peng, S. Yan, and Z. Lu, "Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets," in *Proceedings of the 18th BioNLP Workshop and Shared Task*, 2019, pp. 58–65.

[49] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *arXiv preprint arXiv:2006.08218*, vol. 1, no. 2, 2020.

[50] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[51] A. Sivaraman and M. Kim, "Self-supervised learning from contrastive mixtures for personalized speech enhancement," *arXiv preprint arXiv:2011.03426*, 2020.

[52] Q. Liu, M. J. Kusner, and P. Blunsom, "A survey on contextual embeddings," *arXiv preprint arXiv:2003.07278*, 2020.

[53] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *arXiv preprint arXiv:2101.01169*, 2021.

[54] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on visual transformer," *arXiv preprint arXiv:2012.12556*, 2020.

[55] S. Panda, A. Agrawal, J. Ha, and B. Bloch, "Shuffled-token detection for refining pre-trained roberta," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, 2021, pp. 88–93.

[56] L. Di Liello, M. Gabburo, and A. Moschitti, "Efficient pre-training objectives for transformers," *arXiv preprint arXiv:2104.09694*, 2021.

[57] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?" in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 201–208.

[58] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini, "Deduplicating training data makes language models better," *arXiv preprint arXiv:2107.06499*, 2021.

[59] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[60] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1715–1725.

[61] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[62] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 66–71.

[63] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *arXiv preprint arXiv:1901.07291*, 2019.

[64] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, É. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8440–8451.

[65] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer, "Multilingual denoising pre-training for neural machine translation," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 726–742, 2020.

[66] H. El Boukkouri, O. Ferret, T. Lavergne, H. Noji, P. Zweigenbaum, and J. Tsujii, "Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 6903–6915.

[67] J. H. Clark, D. Garrette, I. Turc, and J. Wieting, "Canine: Pre-training an efficient tokenization-free encoder for language representation," *arXiv preprint arXiv:2103.06874*, 2021.

[68] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," *arXiv preprint arXiv:2105.13626*, 2021.

[69] Y. Tay, V. Q. Tran, S. Ruder, J. Gupta, H. W. Chung, D. Bahri, Z. Qin, S. Baumgartner, C. Yu, and D. Metzler, "Charformer: Fast character transformers via gradient-based subword tokenization," *arXiv preprint arXiv:2106.12672*, 2021.

[70] W. Wang, B. Bi, M. Yan, C. Wu, Z. Bao, J. Xia, L. Peng, and L. Si, "Structbert: Incorporating language structures into pre-training for deep language understanding," *arXiv preprint arXiv:1908.04577*, 2019.

[71] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[72] Y. Qin, Y. Lin, J. Yi, J. Zhang, X. Han, Z. Zhang, Y. Su, Z. Liu, P. Li, M. Sun *et al.*, "Knowledge inheritance for pre-trained language models," *arXiv preprint arXiv:2105.13880*, 2021.

[73] Z. Zhang, Y. Gu, X. Han, S. Chen, C. Xiao, Z. Sun, Y. Yao, F. Qi, J. Guan, P. Ke *et al.*, "Cpm-2: Large-scale cost-effective pre-trained language models," *arXiv preprint arXiv:2106.10715*, 2021.

[74] Y. Kuratov and M. Arkhipov, "Adaptation of deep bidirectional multilingual transformers for russian language," *arXiv preprint arXiv:1905.07213*, 2019.

[75] F. Souza, R. Nogueira, and R. Lotufo, "Bertimbau: Pretrained bert models for brazilian portuguese," in *Brazilian Conference on Intelligent Systems*. Springer, 2020, pp. 403–417.

[76] M. Arkhipov, M. Trofimova, Y. Kuratov, and A. Sorokin, "Tuning multilingual transformers for language-specific named entity recognition," in *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, 2019, pp. 89–93.

[77] D. Carmo, M. Piau, I. Campiotti, R. Nogueira, and R. Lotufo, "Ptt5: Pretraining and validating the t5 model on brazilian portuguese data," *arXiv preprint arXiv:2008.09144*, 2020.

[78] Y. Yao, S. Huang, W. Wang, L. Dong, and F. Wei, "Adapt-and-distill: Developing small, fast and effective pretrained language models for domains," *arXiv preprint arXiv:2106.13474*, 2021.

[79] S. Wada, T. Takeda, S. Manabe, S. Konishi, J. Kamohara, and Y. Matsumura, "Pre-training technique to localize medical bert and enhance biomedical bert," *arXiv preprint arXiv:2005.07202*, 2020.

[80] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large batch optimization for deep learning: Training bert in 76 minutes," in *International Conference on Learning Representations*, 2019.

[81] T. Caselli, V. Basile, J. Mitrović, and M. Granitzer, "Hatebert: Retraining bert for abusive language detection in english," *arXiv preprint arXiv:2010.12472*, 2020.

[82] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[83] J. Zhou, J. Tian, R. Wang, Y. Wu, W. Xiao, and L. He, "Sentix: A sentiment-aware pre-trained model for cross-domain sentiment analysis," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 568–579.

[84] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[85] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 9054–9065.

[86] K. Lo, L. L. Wang, M. Neumann, R. Kinney, and D. S. Weld, "S2orc: The semantic scholar open research corpus," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4969–4983.

[87] W.-R. Chen, M. Abdul-Mageed, H. Cavusoglu *et al.*, "Indt5: A text-to-text transformer for 10 indigenous languages," in *Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas*, 2021, pp. 265–271.

[88] S. Cahyawijaya, G. I. Winata, B. Wilie, K. Vincentio, X. Li, A. Kuncoro, S. Ruder, Z. Y. Lim, S. Bahar, M. L. Khodra *et al.*, "Indonlg: Benchmark and resources for evaluating indonesian natural language generation," *arXiv preprint arXiv:2104.08200*, 2021.

[89] Z. Chi, S. Huang, L. Dong, S. Ma, S. Singhal, P. Bajaj, X. Song, and F. Wei, "Xlm-e: Cross-lingual language model pre-training via electra," *arXiv preprint arXiv:2106.16138*, 2021.

[90] Z. Chi, L. Dong, F. Wei, N. Yang, S. Singhal, W. Wang, X. Song, X.-L. Mao, H. Huang, and M. Zhou, "Infoxlm: An information-theoretic framework for cross-lingual language model pre-training," *arXiv preprint arXiv:2007.07834*, 2020.

[91] Z. Chi, L. Dong, S. Ma, S. H. X.-L. Mao, H. Huang, and F. Wei, "mt6: Multilingual pretrained text-to-text transformer with translation pairs," *arXiv preprint arXiv:2104.08692*, 2021.

[92] B. Wilie, K. Vincentio, G. I. Winata, S. Cahyawijaya, X. Li, Z. Y. Lim, S. Soleman, R. Mahendra, P. Fung, S. Bahar *et al.*, "Indonlu: Benchmark and resources for evaluating indonesian natural language understanding," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 2020, pp. 843–857.

[93] J. A. Wagner Filho, R. Wilkens, M. Idiart, and A. Villavicencio, "The brwac corpus: A new open resource for brazilian portuguese," in *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.

[94] L. Xu, X. Zhang, and Q. Dong, "Cluecorpus2020: A large-scale chinese corpus for pre-training language model," *arXiv preprint arXiv:2003.01355*, 2020.

[95] S. Yuan, H. Zhao, Z. Du, M. Ding, X. Liu, Y. Cen, X. Zou, Z. Yang, and J. Tang, "Wudaocorpora: A super large-scale chinese corpora for pre-training language models," *AI Open*, 2021.

[96] P. J. O. Suárez, B. Sagot, and L. Romary, "Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures," in *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache, 2019.

[97] S. Khanuja, D. Bansal, S. Mehtani, S. Khosla, A. Dey, B. Gopalan, D. K. Margam, P. Aggarwal, R. T. Nagipogu, S. Dave *et al.*, "Muril: Multilingual representations for indian languages," *arXiv preprint arXiv:2103.10730*, 2021.

[98] D. Kakwani, A. Kunchukuttan, S. Golla, N. Gokul, A. Bhattacharyya, M. M. Khapra, and P. Kumar, "inlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 4948–4961.

[99] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mt5: A massively multilingual pre-trained text-to-text transformer," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 483–498.

[100] G. Wenzek, M.-A. Lachaux, A. Conneau, V. Chaudhary, F. Guzmán, A. Joulin, and É. Grave, "Ccnet: Extracting high quality monolingual datasets from web crawl data," in *Proceedings of the 12th Language Resources and Evaluation Conference*, 2020, pp. 4003–4012.

[101] B. Haddow and F. Kirefu, "Pmindia–a collection of parallel corpora of languages of india," *arXiv preprint arXiv:2001.09907*, 2020.

[102] A. Kunchukuttan, P. Mehta, and P. Bhattacharyya, "The iit bombay english-hindi parallel corpus," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[103] H. Huang, Y. Liang, N. Duan, M. Gong, L. Shou, D. Jiang, and M. Zhou, "Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2485–2494.

[104] J. Yang, S. Ma, D. Zhang, S. Wu, Z. Li, and M. Zhou, "Alternating language modeling for cross-lingual pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 9386–9393.

[105] M. Ziemski, M. Junczys-Dowmunt, and B. Pouliquen, "The united nations parallel corpus v1. 0," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016, pp. 3530–3534.

[106] Z. Chi, L. Dong, F. Wei, W. Wang, X.-L. Mao, and H. Huang, "Cross-lingual natural language generation via pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7570–7577.

[107] H. Schwenk, V. Chaudhary, S. Sun, H. Gong, and F. Guzmán, "Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 1351–1361.

[108] A. El-Kishky, V. Chaudhary, F. Guzmán, and P. Koehn, "A massive collection of cross-lingual web-document pairs," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 5960–5969.

[109] B. Roark, L. Wolf-Sonkin, C. Kirov, S. J. Mielke, C. Johny, I. Demirsahin, and K. Hall, "Processing south asian languages written in the latin script: the dakshina dataset," in *Proceedings of the 12th Language Resources and Evaluation Conference*, 2020, pp. 2413–2423.

[110] G. Ramesh, S. Doddapaneni, A. Bheemaraj, M. Jobanputra, R. AK, A. Sharma, S. Sahoo, H. Diddee, D. Kakwani, N. Kumar *et al.*, "Samanantar: The largest publicly available parallel corpora collection for 11 indic languages," *arXiv preprint arXiv:2104.05596*, 2021.

[111] F. Barbieri, J. Camacho-Collados, L. E. Anke, and L. Neves, "Tweeteval: Unified benchmark and comparative evaluation for tweet classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 1644–1650.

[112] F. Barbieri, L. E. Anke, and J. Camacho-Collados, "Xlm-t: A multilingual language model toolkit for twitter," *arXiv preprint arXiv:2104.12250*, 2021.

[113] S. Gururangan, T. Dang, D. Card, and N. A. Smith, "Variational pretraining for semi-supervised text classification," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5880–5894.

[114] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mass: Masked sequence to sequence pre-training for language generation," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5926–5936.

[115] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 13 063–13 075.

[116] P. Ganesh, Y. Chen, X. Lou, M. A. Khan, Y. Yang, D. Chen, M. Winslett, H. Sajjad, and P. Nakov, "Compressing large-scale transformer-based models: A case study on bert," *arXiv preprint arXiv:2002.11985*, 2020.

[117] Y. Meng, W. F. Speier, M. K. Ong, and C. Arnold, "Bidirectional representation learning from transformers using multimodal electronic health record data to predict depression," *IEEE Journal of Biomedical and Health Informatics*, 2021.

[118] L. Rasmy, Y. Xiang, Z. Xie, C. Tao, and D. Zhi, "Med-bert: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction," *NPJ digital medicine*, vol. 4, no. 1, pp. 1–13, 2021.

[119] Y. Li, S. Rao, J. R. A. Solares, A. Hassaine, R. Ramakrishnan, D. Canoy, Y. Zhu, K. Rahimi, and G. Salimi-Khorshidi, "Behrt: transformer for electronic health records," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[120] Y.-P. Chen, Y.-Y. Chen, J.-J. Lin, C.-H. Huang, and F. Lai, "Modified bidirectional encoder representations from transformers extractive summarization model for hospital information systems based on character-level tokens (alphabert): development and performance evaluation," *JMIR medical informatics*, vol. 8, no. 4, p. e17787, 2020.

[121] W. Ma, Y. Cui, C. Si, T. Liu, S. Wang, and G. Hu, "Charbert: Character-aware pre-trained language model," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 39–50.

[122] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 2227–2237.

[123] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proceedings of the 27th international conference on computational linguistics*, 2018, pp. 1638–1649.

[124] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[125] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 66–75.

[126] H.-C. Shin, Y. Zhang, E. Bakhturina, R. Puri, M. Patwary, M. Shoeybi, and R. Mani, "Bio-megatron: Larger biomedical domain language model," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4700–4706.

[127] G. Michalopoulos, Y. Wang, H. Kaka, H. Chen, and A. Wong, "Umlsbert: Clinical domain knowledge augmentation of con-

textual embeddings using the unified medical language system metathesaurus," *arXiv preprint arXiv:2010.10391*, 2020.
[128] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, 2016, pp. 507–517.
[129] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "Bertweet: A pre-trained language model for english tweets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 9–14.
[130] M. Müller, M. Salathé, and P. E. Kummervold, "Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter," *arXiv preprint arXiv:2005.07503*, 2020.
[131] N. Goyal, J. Du, M. Ott, G. Anantharaman, and A. Conneau, "Larger-scale transformers for multilingual masked language modeling," *arXiv preprint arXiv:2105.00572*, 2021.
[132] A. Bhattacharjee, T. Hasan, K. Samin, M. S. Rahman, A. Iqbal, and R. Shahriyar, "Banglabert: Combating embedding barrier for low-resource language understanding," *arXiv preprint arXiv:2101.00204*, 2021.
[133] M. Farahani, M. Gharachorloo, M. Farahani, and M. Manthouri, "Parsbert: Transformer-based model for persian language understanding," *arXiv preprint arXiv:2005.12515*, 2020.
[134] P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych, "How good is your tokenizer? on the monolingual performance of multilingual language models," *arXiv preprint arXiv:2012.15613*, 2020.
[135] M. Abdul-Mageed, A. Elmadany, and E. M. B. Nagoudi, "Arbert & marbert: Deep bidirectional transformers for arabic," *arXiv preprint arXiv:2101.01785*, 2020.
[136] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, "Indolem and indobert: A benchmark dataset and pre-trained language model for indonesian nlp," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 757–770.
[137] D. Q. Nguyen and A. T. Nguyen, "Phobert: Pre-trained language models for vietnamese," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 1037–1042.
[138] P. Delobelle, T. Winters, and B. Berendt, "Robbert: a dutch roberta-based language model," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 3255–3265.
[139] S. Dumitrescu, A.-M. Avram, and S. Pyysalo, "The birth of romanian bert," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 4324–4328.
[140] H. Le, L. Vial, J. Frej, V. Segonne, M. Coavoux, B. Lecouteux, A. Allauzen, B. Crabbé, L. Besacier, and D. Schwab, "Flaubert: Unsupervised language model pre-training for french," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, pp. 2479–2490.
[141] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," in *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, p. 9.
[142] ——, "Aragpt2: Pre-trained transformer for arabic language generation," in *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, 2021, pp. 196–207.
[143] ——, "Araelectra: Pre-training text discriminators for arabic language understanding," in *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, 2021, pp. 191–195.
[144] W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim, "Bertje: A dutch bert model," *arXiv preprint arXiv:1912.09582*, 2019.
[145] A. Virtanen, J. Kanerva, R. Ilo, J. Luoma, J. Luotolahti, T. Salakoski, F. Ginter, and S. Pyysalo, "Multilingual is not enough: Bert for finnish," *arXiv preprint arXiv:1912.07076*, 2019.
[146] M. Polignano, P. Basile, M. De Gemmis, G. Semeraro, and V. Basile, "Alberto: Italian bert language understanding model for nlp challenging tasks based on tweets," in *6th Italian Conference on Computational Linguistics, CLiC-it 2019*, vol. 2481. CEUR, 2019, pp. 1–6.
[147] J. Canete, G. Chaperon, R. Fuentes, and J. Pérez, "Spanish pre-trained bert model and evaluation data," *PML4DC at ICLR*, vol. 2020, 2020.
[148] L. Martin, B. Muller, P. J. O. Suárez, Y. Dupont, L. Romary, É. V. De La Clergerie, D. Seddah, and B. Sagot, "Camembert: a tasty french language model," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7203–7219.
[149] L. Lowphansirikul, C. Polpanumas, N. Jantrakulchai, and S. Nutanong, "Wangchanberta: Pretraining transformer-based thai language models," *arXiv preprint arXiv:2101.09635*, 2021.
[150] H. Lee, J. Yoon, B. Hwang, S. Joe, S. Min, and Y. Gwon, "Korealbert: Pretraining a lite bert model for korean language understanding," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 5551–5557.
[151] P. Rybak, R. Mroczkowski, J. Tracz, and I. Gawlik, "Klej: Comprehensive benchmark for polish language understanding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 1191–1201.
[152] S. Park, J. Moon, S. Kim, W. I. Cho, J. Han, J. Park, C. Song, J. Kim, Y. Song, T. Oh *et al.*, "Klue: Korean language understanding evaluation," *arXiv preprint arXiv:2105.09680*, 2021.
[153] M. Malmsten, L. Börjeson, and C. Haffenden, "Playing with words at the national library of sweden–making a swedish bert," *arXiv preprint arXiv:2007.01658*, 2020.
[154] S. Dadas, M. Perełkiewicz, and R. Poświata, "Pre-training polish transformer-based language models at scale," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2020, pp. 301–314.
[155] M. Straka, J. Náplava, J. Straková, and D. Samuel, "Robeczech: Czech roberta, a monolingual contextualized language representation model," *arXiv preprint arXiv:2105.11314*, 2021.
[156] B. Bi, C. Li, C. Wu, M. Yan, W. Wang, S. Huang, F. Huang, and L. Si, "Palm: Pre-training an autoencoding&autoregressive language model for context-conditioned generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 8681–8691.
[157] H. Fang, S. Wang, M. Zhou, J. Ding, and P. Xie, "Cert: Contrastive self-supervised learning for language understanding," *arXiv preprint arXiv:2005.12766*, 2020.
[158] F. Liu, I. Vulić, A. Korhonen, and N. Collier, "Fast, effective and self-supervised: Transforming masked language models into universal lexical and sentence encoders," *arXiv preprint arXiv:2104.08027*, 2021.
[159] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.
[160] B. Y. Lin, S. Lee, X. Qiao, and X. Ren, "Common sense beyond english: Evaluating and improving multilingual language models for commonsense reasoning," *arXiv preprint arXiv:2106.06937*, 2021.
[161] D. Wang, N. Ding, P. Li, and H.-T. Zheng, "Cline: Contrastive learning with semantic negative examples for natural language understanding," *arXiv preprint arXiv:2107.00440*, 2021.
[162] M. Gupta and P. Agrawal, "Compression of deep learning models for text: A survey," *arXiv preprint arXiv:2008.05221*, 2020.
[163] M. Gordon, K. Duh, and N. Andrews, "Compressing bert: Studying the effects of weight pruning on transfer learning," in *Proceedings of the 5th Workshop on Representation Learning for NLP*, 2020, pp. 143–155.
[164] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?" *arXiv preprint arXiv:1905.10650*, 2019.
[165] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5797–5808.
[166] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov, "Poor man's bert: Smaller and faster transformer models," *arXiv preprint arXiv:2004.03844*, 2020.
[167] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," in *International Conference on Learning Representations*, 2019.
[168] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.
[169] L. J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, 2014, pp. 2654–2662.
[170] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[171] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[172] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 4163–4174.

[173] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for bert model compression," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 4323–4332.

[174] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 2158–2170.

[175] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *arXiv preprint arXiv:2002.10957*, 2020.

[176] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8bert: Quantized 8bit bert," *arXiv preprint arXiv:1910.06188*, 2019.

[177] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8815–8821.

[178] W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, and Q. Liu, "Ternarybert: Distillation-aware ultra-low bit bert," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 509–521.

[179] A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos, "Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 811–824.

[180] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, "Training with quantization noise for extreme model compression," *arXiv preprint arXiv:2004.07320*, 2020.

[181] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser, "Universal transformers," *arXiv preprint arXiv:1807.03819*, 2018.

[182] J. R. Minot, N. Cheney, M. Maier, D. C. Elbers, C. M. Danforth, and P. S. Dodds, "Interpretable bias mitigation for textual data: Reducing gender bias in patient notes while maintaining classification performance," *arXiv preprint arXiv:2103.05841*, 2021.

[183] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3645–3650.

[184] N. Poerner, U. Waltinger, and H. Schütze, "Inexpensive domain adaptation of pretrained language models: Case studies on biomedical ner and covid-19 qa," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 1482–1490.

[185] W. Tai, H. Kung, X. L. Dong, M. Comiter, and C.-F. Kuo, "exbert: Extending pre-trained models with domain-specific vocabulary under constrained training resources," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 1433–1439.

[186] N. Poerner, U. Waltinger, and H. Schütze, "E-bert: Efficient-yet-effective entity embeddings for bert," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 803–818.

[187] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, "Joint learning of the embedding of words and entities for named entity disambiguation," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 250–259.

[188] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "Ernie: Enhanced language representation with informative entities," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 1441–1451.

[189] M. E. Peters, M. Neumann, R. Logan, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 43–54.

[190] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[191] X. Cheng, "Dual-view distilled bert for sentence embedding," *arXiv preprint arXiv:2104.08675*, 2021.

[192] Y. Zhang, R. He, Z. Liu, K. H. Lim, and L. Bing, "An unsupervised sentence embedding method by mutual information maximization," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 1601–1610.

[193] K. Wang, N. Reimers, and I. Gurevych, "Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning," *arXiv preprint arXiv:2104.06979*, 2021.

[194] F. Carlsson, A. C. Gyllensten, E. Gogoulou, E. Y. Hellqvist, and M. Sahlgren, "Semantic re-tuning with contrastive tension," in *International Conference on Learning Representations*, 2020.

[195] Y. Yan, R. Li, S. Wang, F. Zhang, W. Wu, and W. Xu, "Consert: A contrastive framework for self-supervised sentence representation transfer," *arXiv preprint arXiv:2105.11741*, 2021.

[196] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *arXiv preprint arXiv:1304.5634*, 2013.

[197] Z. Li, X. Ding, K. Liao, T. Liu, and B. Qin, "Causalbert: Injecting causal knowledge into pre-trained models with minimal supervision," *arXiv preprint arXiv:2107.09852*, 2021.

[198] A. Lauscher, I. Vulić, E. M. Ponti, A. Korhonen, and G. Glavaš, "Specializing unsupervised pretraining models for word-level semantic similarity," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 1371–1383.

[199] J. Zhou, Z. Zhang, H. Zhao, and S. Zhang, "Limit-bert: Linguistics informed multi-task bert," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 4450–4461.

[200] P. Ke, H. Ji, S. Liu, X. Zhu, and M. Huang, "Sentilare: Sentiment-aware language representation learning with linguistic knowledge," *arXiv preprint arXiv:1911.02493*, 2019.

[201] B. Hao, H. Zhu, and I. Paschalidis, "Enhancing clinical bert embedding using a biomedical knowledge base," in *Proceedings of the 28th international conference on computational linguistics*, 2020, pp. 657–661.

[202] F. Liu, E. Shareghi, Z. Meng, M. Basaldella, and N. Collier, "Self-alignment pretraining for biomedical entity representations," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 4228–4238.

[203] Z. Yuan, Z. Zhao, and S. Yu, "Coder: Knowledge infused cross-lingual medical term embedding for term normalization," *arXiv preprint arXiv:2011.02947*, 2020.

[204] Y. Levine, B. Lenz, O. Dagan, O. Ram, D. Padnos, O. Sharir, S. Shalev-Shwartz, A. Shashua, and Y. Shoham, "Sensebert: Driving some sense into bert," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4656–4667.

[205] O. Bodenreider, "The unified medical language system (umls): integrating biomedical terminology," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D267–D270, 2004.

[206] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *arXiv preprint arXiv:2106.04554*, 2021.

[207] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[208] J. Ainslie, S. Ontanon, C. Alberti, V. Cvicek, Z. Fisher, P. Pham, A. Ravula, S. Sanghai, Q. Wang, and L. Yang, "Etc: Encoding long and structured inputs in transformers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 268–284.

[209] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences." in *NeurIPS*, 2020.

[210] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2019.

[211] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser *et al.*, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2020.

[212] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv preprint arXiv:2006.03654*, 2020.

[213] Z.-H. Jiang, W. Yu, D. Zhou, Y. Chen, J. Feng, and S. Yan, "Convbert: Improving bert with span-based dynamic convolution," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[214] J. Phang, T. Févry, and S. R. Bowman, "Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks," *arXiv preprint arXiv:1811.01088*, 2018.

[215] Y. Zhou and V. Srikumar, "A closer look at how fine-tuning changes bert," *arXiv preprint arXiv:2106.14282*, 2021.

[216] A. Merchant, E. Rahimtoroghi, E. Pavlick, and I. Tenney, "What happens to bert embeddings during fine-tuning?" in *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2020, pp. 33–44.

[217] M. Mosbach, A. Khokhlova, M. A. Hedderich, and D. Klakow, "On the interplay between fine-tuning and sentence-level probing for linguistic knowledge in pre-trained transformers," in *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2020, pp. 68–82.

[218] Y. Hao, L. Dong, F. Wei, and K. Xu, "Investigating learning dynamics of bert fine-tuning," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 2020, pp. 87–92.

[219] Y. Pruksachatkun, J. Phang, H. Liu, P. M. Htut, X. Zhang, R. Y. Pang, C. Vania, K. Kann, and S. Bowman, "Intermediate-task transfer learning with pretrained language models: When and why does it work?" in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 5231–5247.

[220] C. Poth, J. Pfeiffer, A. Rücklé, and I. Gurevych, "What to pre-train on? efficient intermediate task selection," *arXiv preprint arXiv:2104.08247*, 2021.

[221] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4487–4496.

[222] A. Mulyar and B. T. McInnes, "Mt-clinical bert: Scaling clinical information extraction with multitask learning," *arXiv preprint arXiv:2004.10220*, 2020.

[223] D. Mahajan, A. Poddar, J. J. Liang, Y.-T. Lin, J. M. Prager, P. Suryanarayanan, P. Raghavan, and C.-H. Tsou, "Identification of semantically similar sentences in clinical notes: Iterative intermediate training using multi-task learning," *JMIR medical informatics*, vol. 8, no. 11, p. e22508, 2020.

[224] Y. Peng, Q. Chen, and Z. Lu, "An empirical study of multi-task learning on bert for biomedical text mining," in *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, 2020, pp. 205–214.

[225] C. McCreery, N. Katariya, A. Kannan, M. Chablani, and X. Amatriain, "Domain-relevant embeddings for medical question similarity," *arXiv preprint arXiv:1910.04192*, 2019.

[226] C. Cengiz, U. Sert, and D. Yuret, "Ku_ai at mediqa 2019: Domain-specific pre-training and transfer learning for medical nli," in *Proceedings of the 18th BioNLP Workshop and Shared Task*, 2019, pp. 427–436.

[227] X. Yang, X. He, H. Zhang, Y. Ma, J. Bian, and Y. Wu, "Measurement of semantic textual similarity in clinical texts: Comparison of transformer-based models," *JMIR Medical Informatics*, vol. 8, no. 11, p. e19735, 2020.

[228] Y. Wang, K. Verspoor, and T. Baldwin, "Learning from unlabelled data for clinical semantic textual similarity," in *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, 2020, pp. 227–233.

[229] W. Yoon, J. Lee, D. Kim, M. Jeong, and J. Kang, "Pre-trained language model for biomedical question answering," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 727–740.

[230] M. Jeong, M. Sung, G. Kim, D. Kim, W. Yoon, J. Yoo, and J. Kang, "Transferability of natural language inference to biomedical question answering," *arXiv preprint arXiv:2007.00217*, 2020.

[231] A. Wang, J. Hula, P. Xia, R. Pappagari, R. T. McCoy, R. Patel, N. Kim, I. Tenney, Y. Huang, K. Yu *et al.*, "Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4465–4476.

[232] J. Worsham and J. Kalita, "Multi-task learning for natural language processing in the 2020s: Where are we going?" *Pattern Recognition Letters*, vol. 136, pp. 120–126, 2020.

[233] M. R. Khan, M. Ziyadi, and M. AbdelHady, "Mt-bioner: Multi-task learning for biomedical named entity recognition using deep bidirectional transformers," *arXiv preprint arXiv:2001.08904*, 2020.

[234] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[235] A. C. Stickland and I. Murray, "Bert and pals: Projected attention layers for efficient adaptation in multi-task learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5986–5995.

[236] J. Pfeiffer, E. Simpson, and I. Gurevych, "Low resource multi-task sequence tagging–revisiting dynamic conditional random fields," *arXiv preprint arXiv:2005.00250*, 2020.

[237] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky, "Revealing the dark secrets of bert," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 4365–4374.

[238] F. Dalvi, H. Sajjad, N. Durrani, and Y. Belinkov, "Analyzing redundancy in pretrained transformer models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4908–4926.

[239] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv preprint arXiv:2107.13586*, 2021.

[240] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, "Language models as knowledge bases?" in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2463–2473.

[241] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, 2020.

[242] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," *arXiv preprint arXiv:2012.15723*, 2020.

[243] W. Yuan, G. Neubig, and P. Liu, "Bartscore: Evaluating generated text as text generation," *arXiv preprint arXiv:2106.11520*, 2021.

[244] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Eliciting knowledge from language models using automatically generated prompts," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4222–4235.

[245] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing nlp," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2153–2162.

[246] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[247] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint arXiv:2103.10385*, 2021.

[248] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[249] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, E. Simperl, and F. Laforest, "T-rex: A large scale alignment of natural language with knowledge base triples," 2019.

[250] R. Speer, C. Havasi *et al.*, "Representing general relational knowledge in conceptnet 5." in *LREC*, vol. 2012, 2012, pp. 3679–86.

[251] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[252] N. Kassner, P. Dufter, and H. Schütze, "Multilingual lama: Investigating knowledge in multilingual pretrained language models," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 3250–3258.

[253] N. Kassner and H. Schütze, "Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7811–7818.

[254] L. K. Senel and H. Schütze, "Does he wink or does he nod? a challenging benchmark for evaluating word understanding of language models," *arXiv preprint arXiv:2102.03596*, 2021.

[255] C. Fellbaum, "Wordnet," in *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 231–243.

[256] T. Schick and H. Schütze, "Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8766–8774.

[257] Z. Jiang, A. Anastasopoulos, J. Araki, H. Ding, and G. Neubig, "X-factr: Multilingual factual knowledge retrieval from pretrained language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 5943–5959.

[258] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu, "Open mind common sense: Knowledge acquisition from the general public," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2002, pp. 1223–1237.

[259] J. Salazar, D. Liang, T. Q. Nguyen, and K. Kirchhoff, "Masked language model scoring," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 2699–2712.

[260] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018, pp. 353–355.

[261] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *arXiv preprint arXiv:1905.00537*, 2019.

[262] D. Khashabi, G. Stanovsky, J. Bragg, N. Lourie, J. Kasai, Y. Choi, N. A. Smith, and D. S. Weld, "Genie: A leaderboard for human-in-the-loop evaluation of text generation," *arXiv preprint arXiv:2101.06561*, 2021.

[263] S. Gehrmann, T. Adewumi, K. Aggarwal, P. S. Ammanamanchi, A. Anuoluwapo, A. Bosselut, K. R. Chandu, M. Clinciu, D. Das, K. D. Dhole *et al.*, "The gem benchmark: Natural language generation, its evaluation and metrics," *arXiv preprint arXiv:2102.01672*, 2021.

[264] D. Liu, Y. Yan, Y. Gong, W. Qi, H. Zhang, J. Jiao, W. Chen, J. Fu, L. Shou, M. Gong *et al.*, "Glge: A new general language generation evaluation benchmark," *arXiv preprint arXiv:2011.11928*, 2020.

[265] S. Mehri, M. Eric, and D. Hakkani-Tur, "Dialoglue: A natural language understanding benchmark for task-oriented dialogue," *arXiv preprint arXiv:2009.13570*, 2020.

[266] N. Zhang, Q. Jia, K. Yin, L. Dong, F. Gao, and N. Hua, "Conceptualized representation learning for chinese biomedical text mining," *arXiv preprint arXiv:2008.10813*, 2020.

[267] Y. Liang, N. Duan, Y. Gong, N. Wu, F. Guo, W. Qi, M. Gong, L. Shou, D. Jiang, G. Cao *et al.*, "Xglue: A new benchmark datasetfor cross-lingual pre-training, understanding and generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6008–6018.

[268] G. Aguilar, S. Kar, and T. Solorio, "Lince: A centralized benchmark for linguistic code-switching evaluation," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, pp. 1803–1813.

[269] S. Khanuja, S. Dandapat, A. Srinivasan, S. Sitaram, and M. Choudhury, "Gluecos: An evaluation benchmark for code-switched nlp," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 3575–3585.

[270] J. Hu, S. Ruder, A. Siddhant, G. Neubig, O. Firat, and M. Johnson, "Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4411–4421.

[271] S. Ruder, N. Constant, J. Botha, A. Siddhant, O. Firat, J. Fu, P. Liu, J. Hu, G. Neubig, and M. Johnson, "Xtreme-r: Towards more challenging and nuanced multilingual evaluation," *arXiv preprint arXiv:2104.07412*, 2021.

[272] T. Shavrina, A. Fenogenova, E. Anton, D. Shevelev, E. Artemova, V. Malykh, V. Mikhailov, M. Tikhonova, A. Chertok, and A. Evlampiev, "Russiansuperglue: A russian language understanding evaluation benchmark," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4717–4726.

[273] L. Xu, H. Hu, X. Zhang, L. Li, C. Cao, Y. Li, Y. Xu, K. Sun, D. Yu, C. Yu *et al.*, "Clue: A chinese language understanding evaluation benchmark," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 4762–4772.

[274] D. Khashabi, A. Cohan, S. Shakeri, P. Hosseini, P. Pezeshkpour, M. Alikhani, M. Aminnaseri, M. Bitaab, F. Brahman, S. Ghazarian *et al.*, "Parsinlu: a suite of language understanding challenges for persian," *arXiv preprint arXiv:2012.06154*, 2020.

[275] L. Xu, X. Lu, C. Yuan, X. Zhang, H. Yuan, H. Xu, G. Wei, X. Pan, and H. Hu, "Fewclue: A chinese few-shot learning evaluation benchmark," *arXiv preprint arXiv:2107.07498*, 2021.

[276] J. Bragg, A. Cohan, K. Lo, and I. Beltagy, "Flex: Unifying evaluation for few-shot nlp," *arXiv preprint arXiv:2107.07170*, 2021.

[277] T. Schick and H. 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*, 2021, pp. 2339–2352.

[278] T. Wolf, J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.

[279] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "Fairseq: A fast, extensible toolkit for sequence modeling," *NAACL HLT 2019*, p. 48, 2019.

[280] Y. Yan, F. Hu, J. Chen, N. Bhendawade, T. Ye, Y. Gong, N. Duan, D. Cui, B. Chi, and R. Zhang, "Fastseq: Make sequence generation faster," *arXiv preprint arXiv:2106.04718*, 2021.

[281] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3505–3506.

[282] X. Wang, Y. Xiong, Y. Wei, M. Wang, and L. Li, "Lightseq: A high performance inference library for transformers," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, 2021, pp. 113–120.

[283] J. Vig, "Bertviz: A tool for visualizing multihead self-attention in the bert model," in *ICLR Workshop: Debugging Machine Learning Models*, 2019.

[284] B. Hoover, H. Strobelt, and S. Gehrmann, "exbert: A visual analysis tool to explore learned representations in transformer models," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020, pp. 187–196.

[285] J. Fang, Y. Yu, C. Zhao, and J. Zhou, "Turbotransformers: an efficient gpu serving system for transformer models," in *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2021, pp. 389–402.

[286] Z. Yang, Y. Cui, Z. Chen, W. Che, T. Liu, S. Wang, and G. Hu, "Textbrewer: An open-source knowledge distillation toolkit for natural language processing," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020, pp. 9–16.

[287] H. Shah, A. Khare, N. Shah, and K. Siddiqui, "Kd-lib: A pytorch library for knowledge distillation, pruning and quantization," *arXiv preprint arXiv:2011.14691*, 2020.

[288] T. Likhomanenko, Q. Xu, R. Collobert, G. Synnaeve, and A. Rogozhnikov, "Cape: Encoding relative positions with continuous augmented positional embeddings," *arXiv preprint arXiv:2106.03143*, 2021.

[289] P. Liu, J. Fu, Y. Xiao, W. Yuan, S. Chang, J. Dai, Y. Liu, Z. Ye, and G. Neubig, "Explainaboard: An explainable leaderboard for nlp," *arXiv preprint arXiv:2104.06387*, 2021.

[290] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *arXiv preprint arXiv:2106.08962*, 2021.

[291] L. Sun, K. Hashimoto, W. Yin, A. Asai, J. Li, P. Yu, and C. Xiong, "Adv-bert: Bert is not robust on misspellings! generating nature

adversarial samples on bert," *arXiv preprint arXiv:2003.04985*, 2020.
[292] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5582–5591.
[293] Z. Wang, S. Mayhew, D. Roth *et al.*, "Extending multilingual bert to low-resource languages," *arXiv preprint arXiv:2004.13640*, 2020.
[294] Y. Nakamura, S. Hanaoka, Y. Nomura, N. Hayashi, O. Abe, S. Yada, S. Wakamiya, and E. Aramaki, "Kart: Privacy leakage framework of language models pre-trained with clinical records," *arXiv preprint arXiv:2101.00036*, 2020.
[295] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," *arXiv preprint arXiv:2012.07805*, 2020.
[296] V. Misra, "Black box attacks on transformer language models," in *ICLR 2019 Debugging Machine Learning Models Workshop*, 2019.
[297] S. Hisamoto, M. Post, and K. Duh, "Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 49–63, 2020.
[298] J. Wang, G. Zhang, W. Wang, K. Zhang, and Y. Sheng, "Cloud-based intelligent self-diagnosis and department recommendation service using chinese medical bert," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–12, 2021.
[299] D. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," *arXiv preprint arXiv:1908.10063*, 2019.
[300] P. P. Liang, C. Wu, L.-P. Morency, and R. Salakhutdinov, "Towards understanding and mitigating social biases in language models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6565–6576.
[301] C. Lee, K. Cho, and W. Kang, "Mixout: Effective regularization to finetune large-scale pretrained language models," *arXiv preprint arXiv:1909.11299*, 2019.
[302] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *arXiv preprint arXiv:2002.06305*, 2020.
[303] M. Mosbach, M. Andriushchenko, and D. Klakow, "On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines," in *International Conference on Learning Representations*, 2020.
[304] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," *arXiv preprint arXiv:2011.01403*, 2020.