

---

# AN OVERVIEW ON LANGUAGE MODELS: RECENT DEVELOPMENTS AND OUTLOOK

---

Chengwei Wei<sup>1</sup>, Yun-Cheng Wang<sup>1</sup>, Bin Wang<sup>2</sup>, and C.-C. Jay Kuo<sup>1</sup>

<sup>1</sup>University of Southern California, Los Angeles, California, USA

<sup>2</sup>National University of Singapore, Singapore  
chengwei@usc.edu

## ABSTRACT

Language modeling studies the probability distributions over strings of texts. It is one of the most fundamental tasks in natural language processing (NLP). It has been widely used in text generation, speech recognition, machine translation, etc. Conventional language models (CLMs) aim to predict the probability of linguistic sequences in a causal manner. In contrast, pre-trained language models (PLMs) cover broader concepts and can be used in both causal sequential modeling and fine-tuning for downstream applications. PLMs have their own training paradigms (usually self-supervised) and serve as foundation models in modern NLP systems. This overview paper provides an introduction to both CLMs and PLMs from five aspects, i.e., linguistic units, structures, training methods, evaluation methods, and applications. Furthermore, we discuss the relationship between CLMs and PLMs and shed light on the future directions of language modeling in the pre-trained era.

**Keywords** Language model, Natural language processing, Pre-trained language model, Conventional language model.

## 1 Introduction

Language modeling studies the probability distributions over a sequence of words. It is one of the most fundamental tasks and long-standing research topics in natural language processing (NLP). The developed language models (LMs) find applications in many computational linguistic problems such as text generation, machine translation, speech recognition, natural language generation, question-and-answer systems, etc.

There are two major approaches to language modeling: 1) the statistical approach based on a relatively small corpus set, and 2) the data-driven approach based on a significantly larger corpus set. Conventional language models (CLMs) predict the probability of linguistic sequences in a causal manner. They can be learned by both language modeling approaches. The data-driven approach has become mainstream nowadays. It exploits a large number of corpora to train neural-network models, leading to pre-trained language models (PLMs). PLMs are then fine-tuned with task-specific datasets and objectives for downstream applications. In this paper, we provide an overview of CLMs and PLMs and study them from five perspectives: 1) linguistic units, 2) structures, 3) training methods, 4) evaluation methods, and 5) applications. In the end, we point out several future research directions.

The goal of CLMs is to model the probability distributions over sequences of linguistic units:

$$P(u_1, u_2, \dots, u_t), \quad (1)$$

where  $u_i$  can be either a character, a word, a phrase, or other linguistic units. CLMs attempt to predict the next linguistic unit in a text sequence given its preceding contexts:

$$P(u_t | u_{<t}) \quad (2)$$

CLMs are also called auto-regressive language models since the units are predicted in a causal way. Estimating the probability of a text sequence as shown in Eq. (1) directly encounters the data sparsity problem. CLMs often estimate

the joint probability of the text sequence by decomposing a text sequence into smaller units. For example, CLMs leverage the chain rule and the conditional probability to estimate the joint probability in the form of

$$P(u_1, u_2, \dots, u_t) = P(u_1)P(u_2|u_1)P(u_3|u_1, u_2) \dots P(u_t|u_1, \dots, u_{t-1}). \quad (3)$$

CLMs are often trained from scratch with a training corpus and, then, predict the probability of text sequences with respective applications. Representative models include N-grams LMs [1, 2, 3], exponential LMs [4, 5, 6] and earlier neural LMs [7, 8]. CLMs give a high probability to natural text sequences occurring frequently in the real world. As a result, they play a fundamental role in text generation, speech recognition [9, 10, 11], and machine translation [12, 13, 14] until the emergence of PLMs. Nowadays, high-performance PLMs serve as the backbone of many NLP systems. They are not limited to the causal predictive functionality of CLMs and provide more different types of LMs.

The differences between CLMs and PLMs can be summarized below.

- **Training Methodology.** With the development of deep learning, PLMs with neural network structures are pre-trained by collections of massive unlabeled corpora to learn generic knowledge which is then transferred to downstream tasks by task-specific fine-tuning.
- **Causality Constraint.** PLMs do not necessarily follow CLMs in predicting linguistic units as shown in Eq. (2). For example, bidirectional LMs [15, 16] use both preceding and succeeding contexts to predict the missing linguistic units via probability estimation:

$$P(u_t|u_{<t}, u_{>t}). \quad (4)$$

Bidirectional LMs do not follow the causality constraint and the chain rule in Eq. (3), to access the probability of a text sequence, which makes it inherently different from CLMs.

- **Token Representation.** Apart from the differences in the training paradigm and probability modeling, PLMs adopt a different representation for basic units called tokens. PLMs represent tokens by embedding them in a high-dimensional continuous space such as word embeddings [17, 18] and sentence embeddings [19, 20, 21]. The new representations offer a flexible and powerful tool that enables PLMs to handle a wide range of tasks.

This overview paper serves two objectives. On one hand, instead of only focusing on recently developed PLMs [22, 23, 24], we aim to provide a comprehensive overview of the basic concepts of LMs, the transition from CLMs to PLMs, LM’s recent developments and applications to beginners in the field. On the other hand, we would like to shed light on future research directions and offer our outlook to experienced engineers and researchers in the NLP field. For example, we cover large LMs (LLMs) in the survey as there are growing interests in LLMs due to the new services provided by ChatGPT. Furthermore, we include efficient LMs as an emerging topic since there are increasing concerns about large model sizes and high training costs of LLMs.

The rest of the paper is organized as below. We introduce several types of LMs that go beyond CLMs in Sec. 2, and provide an overview of common ways to decompose text sequences into smaller linguistic units in Sec. 3. Sec. 4 introduces different model structures. We discuss the training procedures of LMs in Sec. 5. Common evaluation methods including, both intrinsic and extrinsic ones, are introduced in Sec. 6. The application of LMs to text generation is discussed in Sec. 7. We comment on the redundancy problem of LMs and analyze techniques for efficient LMs in Sec. 8. Promising future research directions are pointed out in Sec. 9. Concluding remarks are given in Sec. 10

## 2 Types of Language Models

CLMs commonly refer to auto-regressive models that predict the next linguistic units given the preceding context as shown in Eq. (2). LMs can access the probability of a text sequence using the chain rule. The goal of CLMs is to decode the probability of text sequences in a causal manner. In this section, we introduce more LMs that go beyond CLMs.

### 2.1 Structural LM

Instead of predicting linguistic units in a sequential or reversed sequential order, structural LMs [25, 26, 27, 28, 29] predict linguistic units based on pre-defined linguistic structures such as dependency or constituent parse trees. Structural LMs utilize the linguistic structure to bring linguistically relevant context closer to the linguistic unit to be predicted. For example, given a parse tree structure, a structural LM can define the ancestor context  $A(u_t)$  of  $u_t$  as the sequence from the root node to the parent of  $u_t$ . For example, the ancestor sequence of word ‘strong’ is {‘binoculars’, ‘saw’, ROOT} in Fig. 1. Then, the structural LM uses the ancestor context in the tree to predict the next linguistic unit as

$$P(u_t|A(u_t)), \quad (5)$$

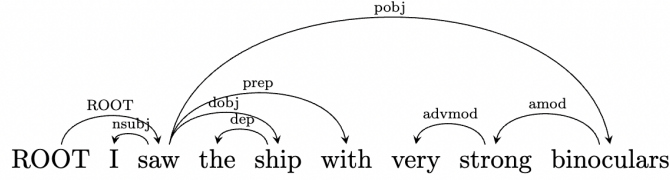


Figure 1: The example of a dependency parse tree example [28].

**The input text:**

Language modeling is very important in NLP

**Permutations:**

modeling Language very is NLP important in

Language very modeling is in NLP important

NLP modeling is important very in Language

.....

Figure 2: The use of different permutations in a natural sentence.

where  $A(u_t)$  is the ancestor context of linguistic unit  $u_t$ . Similar to CLMs, structural LMs are designed to model the probability of text sequences. Differently, structure LMs decode the sequence probability in the order of their synthetic structures. It has been successfully applied to sentence completion [27, 28] and speech recognition [25, 26].

## 2.2 Bidirectional LM

Instead of using the causal contexts to make predictions, bidirectional LMs utilize contexts from both directions as shown in Eq. (4). The masked LM is one representative bidirectional LM. It masks out linguistic units in a text sequence and, then, encodes their preceding and succeeding contexts to predict the masked linguistic units. Formally, the prediction can be defined as the estimation of the following conditional probability

$$P(u_m|\bar{S}), \quad (6)$$

where  $u_m$  is the masked linguistic unit and  $\bar{S}$  is the corrupted text sequence by replacing a certain number of linguistic units with [MASK] symbols. The goal of bidirectional LMs is to learn the inner dependency between linguistic units in an unsupervised manner. The trained model can inherit semantics meanings from large-scale unlabeled corpora. Different from CLMs that aim to model the generation probability of text sequences, pre-trained bidirectional LMs are used as the backbone that transfers the learned knowledge through further fine-tuning in various downstream applications.

## 2.3 Permutation LM

CLMs and masked LMs have their own advantages and disadvantages. A masked LM needs to create artificial tokens such as [mask], which never occur in downstream tasks while CLMs only condition on preceding context. The permutation LM [30] is a recently proposed LM that takes advantage of CLMs and masked LMs. Given an input sequence of linguistic units, permutation LMs randomize the order of input linguistic units and construct different permutations of the input sequence. Fig. 2 shows an example of different permutations given an input text sequence. Let  $\mathbb{Z}$  be the set of all possible permutations. Permutation LMs predict the next linguistic unit,  $u_t$ , in one permutation,  $Z$ , of the sequence based on

$$P(u_t|u_{<t}^Z), Z \in \mathbb{Z}. \quad (7)$$

### 3 Linguistic Units

To estimate the probability of text sequences, LMs partition text sequences into small linguistic units such as characters, words, phrases, or sentences. This process is called tokenization. Different languages and models may have different appropriate tokenization methods. Here, we focus on English and use it as an example. In this section, we examine typical tokenization methods used in language modeling according to unit sizes.

#### 3.1 Characters

LMs can model text sequences probability based on characters [31, 32, 33, 34, 35]. As compared with other linguistics units, using characters has a much smaller vocabulary size, leading to a smaller discrete space and model size. On the other hand, it is challenging to predict the next character. Usually, it requires a long historical context. This makes the performance of character-level LMs poorer than that of word-level LMs. In addition, the input and output lengths have to be longer to model the character distribution accurately. This results in higher computational costs, especially for auto-regressive decoding. Several LM methods use the combination of words and characters to alleviate the issue [36, 37, 38].

#### 3.2 Words and Subwords

The most natural tokenization for English is to decompose a text sequence into words by white spaces. Many LMs apply word tokenization. However, there are several issues of naive word tokenization. The first one is the Out-Of-Vocabulary (OOV) problem. Because an LM has a pre-defined vocabulary size that cannot be arbitrarily large. Less frequent words and words with character-level errors may not be stored in the pre-defined vocabulary. Thus, they cannot be retrieved from the dictionary. Although one can extend the vocabulary size to alleviate this problem, it will increase the model size and still cannot handle all possible words.

LMs beyond the word level still have the OOV problem while a single character is not semantically meaningful by themselves. Recently, researchers are in favor of decomposing words into subwords if they do not appear in the dictionary. This offers a flexible and effective solution to the OOV problem [39, 40]. Several subword segmentation algorithms are developed to boost the performance of LMs. They strike a balance between the good performance of word-level models and the flexibility of character-level models. Two subword segmentation approaches, statistics-based and linguistics-based, are presented below.

##### 3.2.1 Statistics-based Subword Tokenizers

The statistics-based subword tokenizers generate subword vocabulary purely based on the corpus. The associated methods are derived from a compression point of view. They work by replacing the commonly appeared character sequences with a new symbol (word) that does not exist in the current vocabulary. Then, fewer bytes are needed for information transmission.

**Byte Pair Encoding (BPE).** BPE [41] is a simple data compression technique that replaces the most common pair of bytes in a sequence by a single unused byte recursively. It was adopted by [40] to solve the word segmentation problem. That is, frequent characters or character sequences are merged to generate subwords. BPE is also used by several advanced PLMs such as GPT-2 [42] and RoBERTa [16] with the following algorithm, called the BPE merge operation.

1. Prepare a training corpus and define the size of the subword vocabulary.
2. Split all words into characters.
3. Generate a new subword by merging a pair of characters or subwords with the highest frequency.
4. Repeat step 3 until the desired vocabulary size is reached.

An illustration of the BPE merge operation conducted on a small dictionary is given in Fig. 3.

**WordPiece.** [43] WordPiece is another data-driven subword algorithm. The difference between WordPiece and BPE is that WordPiece merges the pair of  $A$  and  $B$  if they have the highest score  $P(AB)/P(A)P(B)$  (rather than the highest frequency  $P(AB)$ ) at each iterative step. For example, WordPiece merges the pair of “u” and “g” in Fig. 3 only if they have the highest value,  $P('ug')/P('u')P('g')$ , as compared with other pairs. WordPiece is used as the tokenization method in BERT [15], DistilBERT [44], and Electra [45].

There are other statistics-based subword tokenizers such as **Unigram** [46]. Different subword tokenizers and their performance comparison are studied in [47].

<b>Words and their frequency in the training corpus:</b> ("hug": 10), ("pug": 5), ("pun": 12), ("bun": 4)
<b>Split all words to characters:</b> ("h" "u" "g": 10), ("p" "u" "g": 5), ("p" "u" "n": 12), ("b" "u" "n": 4) <b>Current vocabulary:</b> {"b", "g", "h", "n", "p", "u"}
"p" followed by "u" occur 17 times and are the most frequent. <b>Merge "p" and "u":</b> ("h" "u" "g": 10), ("pu" "g": 5), ("pu" "n": 12), ("b" "u" "n": 4) <b>Current vocabulary:</b> {"b", "g", "h", "n", "p", "u", "pu"}
<b>Keep merging until reaching the desired vocabulary</b>

Figure 3: Illustration of the BPE merge operation conducted on the dictionary {"hug", "pug", "pun", "bun"}. The vocabulary is initialized with all characters. Then, a new subword is created by merging the most frequent pair.

### 3.2.2 Linguistics-based Subword Tokenizers

Linguistics-based subword tokenizers exploit the linguistic knowledge and decompose words into smaller grammatical units, such as morphemes or syllables. Such subword tokenizers are widely used in machine translation and speech recognition among different languages [48, 49, 50, 51, 52, 53, 54]. For example, in machine translation, words formed by compounding, affixation, or inflection, can be conveniently translated by translating the morphemes, respectively. However, linguistics-based subword tokenizers are not as popular as statistics-based ones due to the complexity and the rule-based nature of language decomposition.

### 3.3 Phrases

The semantic meaning of a single word can be ambiguous because of various contexts and set collocations. Since the linguistic dictionary does not go beyond the word-level, the inter-word dependency is ignored. Phrase-level LMs replace common and cohesive word sequences by phrases [55, 56, 57, 58]. Phrase-level LMs are suitable for some applications. For example, it is observed in [57] that short words with fewer syllables in automatic speech recognition (ASR) are more frequently misrecognized than longer ones. Since phrases provide longer phone sequences than their constituents, they are more robust to recognition errors for ASR.

### 3.4 Sentences

LMs with smaller linguistic units (e.g., characters, words, subwords, and phrases) rely on conditional probabilities to estimate the probability of text sequences as given in Eq. (3). Sentence-level LMs [59, 60, 61, 62, 63] avoid the use of the chain rule. They generate sentence features and, then, model the sentence probability directly. This is because modeling the sentence probability directly is more convenient than that in Eq. (3) in encoding the sentence-level information. It is also easier to encode the inter-sentence information such as the effects of preceding utterances in a dialog flow.

## 4 Model Structures

In this section, we conduct a survey on several common structures to model the probability distributions of text sequences. They are N-gram, maximum entropy, and neural network models. PLMs typically use continuous representations in probability modeling built upon recurrent neural networks (RNNs) or transformers.

### 4.1 N-gram Models

An N-gram consists of N consecutive linguistic units from a text sequence. N-gram LMs [1, 2, 3] assume that the probability of a word depends only on its preceding N-1 linguistic units and it is independent of other contexts. This is known as the Markov assumption. Thus, instead of using all historical contexts, N-gram LMs only use the previous N-1 linguistic units to predict the current one; namely,

$$P(w_t|w_{<t}) = P(w_t|w_{t-N+1:t-1}). \quad (8)$$

N-gram LMs calculate the conditional probability by counting the occurrence time of N-grams given a training corpus as

$$P(w_t|w_{t-N+1:t-1}) = \frac{C(w_{t-N+1:t})}{C(w_{t-N+1:t-1})}. \quad (9)$$

N-gram LMs simplify the word probability calculation based on previous N-1 words, but they encounter two sparsity issues. First, if an N-gram,  $(w_{t-N+1:t})$ , never occurs in the training corpus, the probability for the next word being  $w_t$  is zero. Second, if the (N-1)-gram,  $(w_{t-N+1:t-1})$ , in the denominator never occurs, we cannot calculate the probability of any word. These sparsity issues can be alleviated by smoothing techniques. A simple smoothing method [64, 65], called additive smoothing, is to add a small value to the count for every N-gram so as to avoid zero in the numerator and the denominator in Eq. (9). However, this simple smoothing is still deficient because it assigns the same probability for N-grams that never occur in the training corpus.

There are more advanced smoothing techniques such as back-off and interpolation [66, 67, 68, 69, 70] that achieve better probability estimation. In back-off, lower-order N-grams are used for probability estimation if higher-order N-grams do not occur. For example, if  $C(w_{t-3:t-1}) = 0$ , we back off to compute  $P(w_t|w_{t-2:t-1})$ . In interpolation, different N-grams are considered for conditional probability computation. Mathematically, the N-gram probability is estimated by

$$P(w_t|w_{t-N+1:t-1}) = \lambda_N P(w_t|w_{t-N+1:t-1}) + \lambda_{N-1} P(w_t|w_{t-N:t-1}) + \lambda_{N-2} P(w_t|w_{t-N-1:t-1}) + \dots + \lambda_1 P(w_t), \quad (10)$$

where  $\lambda_i$  is the weight for each n-gram and  $\sum_{i=1}^N \lambda_i = 1$ .

## 4.2 Maximum Entropy Models

Maximum Entropy models (also called the exponential models) [4, 5, 6] estimate the probability of text sequences using feature functions in the form of

$$P(w|h) = \frac{\exp(a^T f(w, h))}{\sum_{w'} \exp(a^T f(w', h))}, \quad (11)$$

where  $f(w, h)$  is the feature function that generates the feature of word  $w$  and its historical context  $h$ ,  $\sum_{w'} \exp(a^T f(w', h))$  is a normalization factor, and  $a$  is a parameter vector derived by the Generalized Iterative Scaling algorithm [71]. The features are usually generated from the N-grams.

## 4.3 Feed-forward Neural Network (FNN) Models

The discrete nature of the N-gram model is its performance bottleneck even with advanced smoothing techniques. Neural LMs embrace the continuous embedding space (distributed representation of words) to overcome the data sparsity problem. Feed-forward Neural Network (FNN) LMs [7, 72, 73, 74] is one of the earlier neural network models.

An FNN LM takes historical contexts as the input, and outputs the probability distribution of words. As shown in Fig. 4, each word in the preceding context is represented as a vector through a projection layer (i.e., an embedding matrix). These word vectors are sent to the hidden layer with  $H$  hidden units followed by non-linear activation. Then, a softmax function is used to obtain the posterior probabilities for word candidates,  $P(w_j = i|h_j)$ , which are the probabilities of words given a specific history predicted by the language model.

An FNN LM uses a fixed window to collect fixed-length contexts. It is essentially a neural version of N-gram LMs. The FNN LM have several advantages over the N-gram LM by projecting words into continuous space. First, it can handle unseen N-grams by representing each word as an N-gram with a dense vector space. Second, it is storage-efficient since it does not need to count and store the transition probability of conventional N-gram models.

## 4.4 Recurrent Neural Network (RNN) Models

It is clearly insufficient to use the historical context in a fixed-length to predict the next word. In contrast to the limited historical context used in the N-gram and FNN LMs, Recurrent Neural Network (RNN) LMs [8, 75, 76, 77, 78] can exploit arbitrarily long histories to predict the next word.

The structure of a vanilla RNN LM is shown in Fig. 5. A word  $x(i)$  in position  $i$  is first converted into a one-hot representation  $\hat{x}(i)$ . Then, the recurrent hidden state,  $h(i+1)$ , is computed using the previous hidden state,  $h(i)$ , and the one-hot representation,  $\hat{x}(i)$ , of word  $x(i)$  as

$$h(i+1) = f(W\hat{x}(i) + Uh(i)), \quad (12)$$

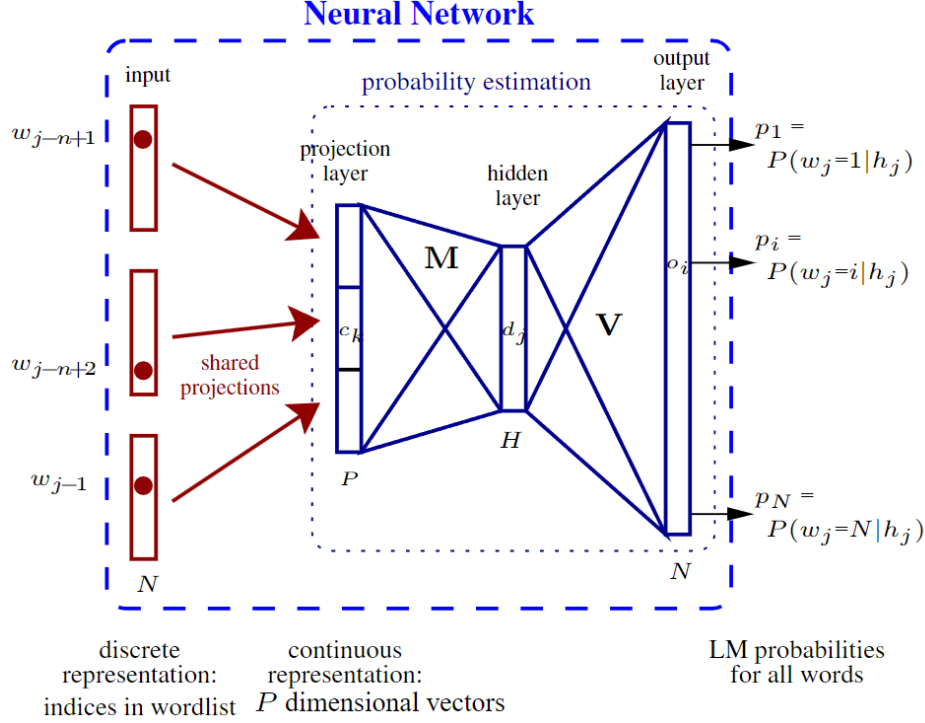


Figure 4: The structure of FFN LMs, where  $h_j$  denotes the preceding contexts  $w_{j-n+1}, \dots, w_{j-1}$  in a fixed-window, and  $P$ ,  $H$ , and  $N$  are the dimensions of the projection, the hidden layer, and the output layer, respectively [72].

where  $f(\cdot)$  is a non-linear activation function,  $W$  is the weight matrix of the connections from the input layer to the hidden layer, and  $U$  is the connection between the previous and current hidden layers, respectively. By iteratively computing the hidden states, RNN LMs can encode the historical context of varying length. Finally, the output layer gives the conditional probability of words  $y(t) = g(Vh(t))$ , where  $V$  is the weight matrix connecting the hidden layer and the output layer and  $g(\cdot)$  is the softmax activation function.

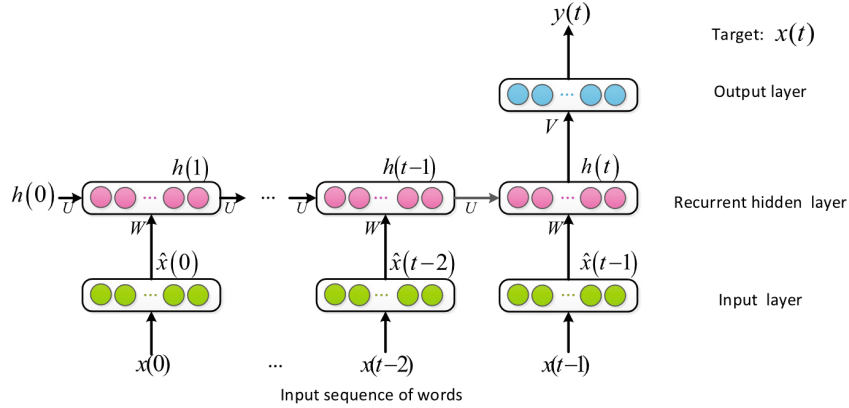


Figure 5: The structure of RNN LMs [79].

In theory, RNN LMs do not need the Markov assumption. They can use all preceding history to predict the next word. However, the inherent gradient vanishing problem of RNN hampers the learning of the model [80]. Since the gradient may become very small over a long distance, model weights are actually updated by the nearby context in practice. Generally, RNN LMs cannot learn the dependency between the current word and its far-away historical context. Although an attention mechanism can be introduced to RNNs to alleviate this problem [81, 79]. The inherent sequential nature of RNNs makes them less powerful than transformer-based LMs with a self-attention mechanism.

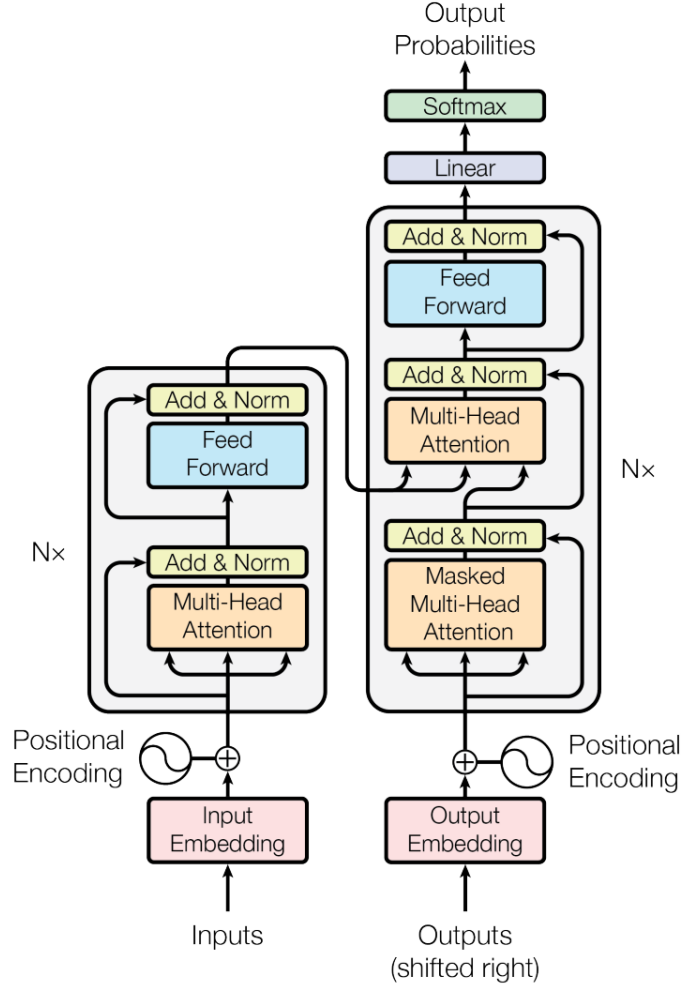


Figure 6: The structure of a transformer [82].

## 4.5 Transformers

The transformer architecture [82] can capture long-term dependencies and important sequence components by exploiting a self-attention mechanism. Unlike the recurrent structure of RNNs, a transformer is easy to parallelize in both training and inference. Its structure is shown in Fig. 6. It consists of an encoder and a decoder. Before being sent to the encoder, the input textual sequence is first converted to an embedding through an embedding layer plus positional embedding. Multi-head attention, which is an ensemble of multiple self-attention mechanisms, enables the transformer to capture more robust and diverse attention between tokens. The other parts in the transformer encoder include feed-forward layers, residual connections, and normalization layers. The difference between the transformer encoder and decoder is that the transformer decoder has an additional masked multi-head attention layer. The masking ensures the decoder can only access preceding tokens of the current one, which makes the decoder auto-regressive.

Based on different purposes, transformers have encoder-only, decoder-only, and encoder-decoder three variants as shown in Table 1 and Fig. 7. Encoder-only models can access all positions given an input and utilize bi-directional contexts to predict words. They are suitable for tasks requiring understanding full sentences, such as text classification. Transformer decoder-only models can only use previous words to predict the current word (namely, auto-regressive models). They are good at text generation tasks such as story generation. Transformer encoder-decoder models can access all words in the encoding phase, and words before the current word in the decoding phase. They are suitable for sequence-to-sequence tasks such as translation and summarization.



Table 1: Transformer-based PLMs.

Encoder-only models (Bidirectional)	BERT [15] RoBERTa [16] ELECTRA [45]
Decoder-only models (Unidirectional)	PaLM [83] GPT-1, 2 and 3 [84, 42, 85] Transformer XL [86]
Encoder-Decoder models (Sequence to sequence)	BART [87] T5 [88]

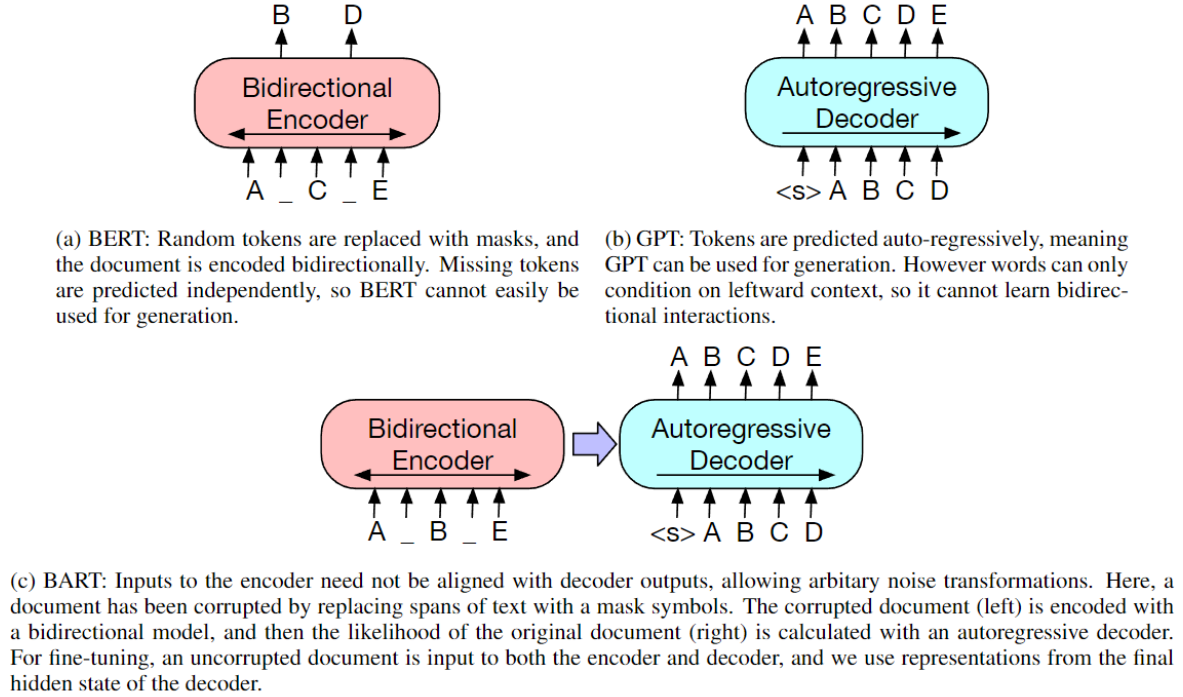


Figure 7: Illustration of different transformer models, where BERT is the encoder-only model, GPT is the decoder-only model, and BART is the encoder-decoder model [87].

## 5 Pre-trained Language Models

Pre-trained language models (PLMs) are dominating in the NLP field nowadays. With the development of deep learning, the training and usage of PLMs have changed a lot as compared with conventional statistical LMs. Before being applied to real-world tasks, PLMs are first pre-trained on massive collections of corpora so that they learn universal representations that carry both syntactic and semantic knowledge. After pre-training, PLMs are fine-tuned for downstream tasks so that the acquired knowledge can be transferred to different tasks. In the following, we first explain the pre-training objectives in Sec. 5.1 and then talk about how to adapt PLMs to various tasks of interest through fine-tuning in Sec. 5.2. It is also worthwhile to point out several good survey papers on PLMs, e.g., [22, 23, 24].

### 5.1 Pre-training

The most commonly used pre-training task is “missing word prediction”. There are other pre-training tasks for different purposes, e.g., next-sentence prediction, which allows an LM to learn sentence relationships.

**Word Prediction.** Auto-aggressive language LMs are trained to predict the next word using previous words. While bidirectional LMs mask a subset of words in a sample and learn to predict such masked words using the rest of the context. For the latter, the most popular objective is the masked language model (MLM) objective as proposed in BERT [15]. The MLM objective is the cross-entropy loss in predicting masked tokens. It randomly masks out 15% of

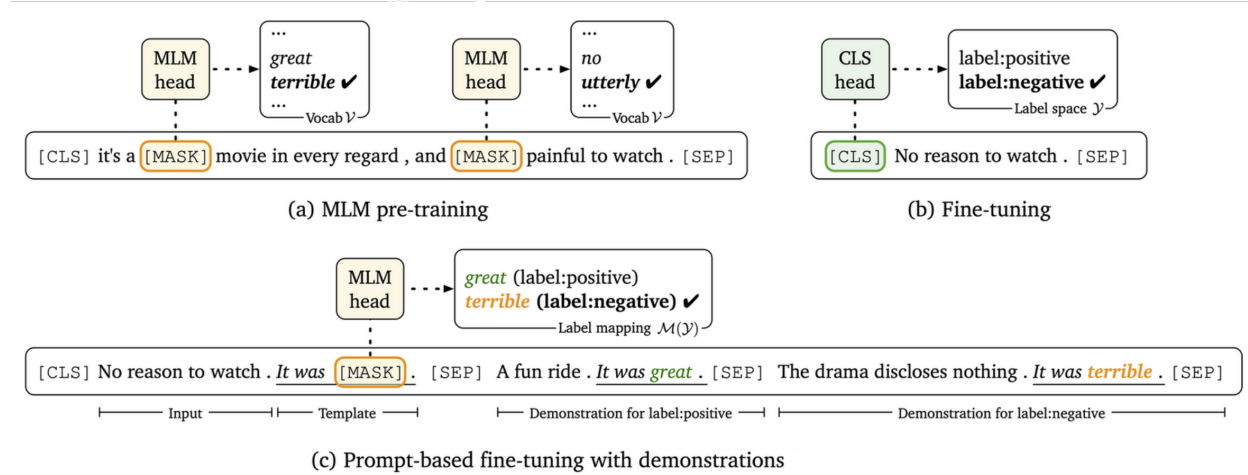


Figure 8: An illustration of (a) LM pre-training, (b) standard fine-tuning, and (c) prompt-based fine-tuning (or prompt-tuning) [89].

the input tokens and then predicts the masked tokens. The number of masked tokens is set to 15% based on experimental verification. If the masking rate is too small, the model only learns from a limited number of masked tokens. On the other hand, if it is too large, there is not enough context to do reasonable predictions and models cannot learn well.

**Other Pre-training Tasks.** There are other pre-training tasks to make LMs learn better linguistic knowledge such as sentence relationships. For example, next sentence prediction is used as the pre-training task in BERT [15]. Next sentence prediction is formalized as a binary prediction task that decides whether two sentences are two consecutive sentences or not. In this way, a PLM can be used in downstream tasks that require the understanding of the relationship between two sentences, such as Question Answering (QA) and Natural Language Inference (NLI). Other pre-training objectives are adopted by BART [87]. They include token deletion, text infilling, sentence permutation, and document rotation to corrupt the original sequence for reconstruction. Shuffled tokens are used in T5 [88] to increase the robustness of the learned representation.

## 5.2 Fine-Tuning and Prompt-Tuning

PLMs learn non-task-specific language knowledge in the pre-training stage. Fine-tuning performs task-specific adaptations of the model so that they can be applied to different downstream tasks. The model parameters are updated in the fine-tuning stage. One can choose to update the entire model, freeze the language model and update the task-specific heads, or only update certain layers of an LM. PLMs are pre-trained by one or several pre-training objectives and, then, applied to different downstream tasks. The gap between pre-training tasks and downstream task-specific fine-tuning can be substantial. Prompt-tuning is used to discover the potential of PLMs by mimicking the pre-training objectives in the fine-tuning or inference stage. As PLMs get more powerful, they can handle various downstream tasks by seeing a few examples without any gradient updates or fine-tuning [85]. This is achieved by prompt-based fine-tuning (or prompt-tuning in short).

A prompt is a human-designed text that is concatenated with the original text as the new model input. Fig. 8 shows an example of the pre-training task, fine-tuning and prompt-tuning of MLMs. In the pre-training, the MLMs are trained to predict masked words. Assuming that the downstream task is the sentiment analysis of the movie review. In standard fine-tuning, we train a new head on the top of a PLM and predict the sentiment labels. The original input appended with a designed prompt, say, 'It was', is sent to the PLM. The PLM has to assign probabilities to designed answers, which can be 'great' or 'terrible'. If the probability of 'great' is higher, then the label of the input will be positive and vice versa. In this way, prompt-tuning converts a distinct downstream task to the word prediction task to narrow the gap between the pre-training and fine-tuning stages.

## 6 Model Evaluation

There are two LM evaluation types: intrinsic evaluation and extrinsic evaluation. The intrinsic evaluation examines the internal properties of an LM while the extrinsic evaluation studies its performance in downstream tasks.

## 6.1 Intrinsic Evaluation

**Auto-regressive LM.** LMs estimate the probability of text sequences. A good LM assigns higher probabilities to natural text sequences and lower ones to unreal or random text sequences. The perplexity is a common evaluation metric for this purpose. Given a testing text sequence, the perplexity, denoted by  $PPL$ , is defined as the inverse probability of the sequence normalized by the number of words. Mathematically, we have

$$PPL(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}, \quad (13)$$

where  $W = w_1 w_2 \dots w_N$  is a testing text sequence. The perplexity can be rewritten in form of

$$PPL(W) = \sqrt[N]{\prod_{i=1}^n \frac{1}{P(w_i | w_1 \dots w_{i-1})}}. \quad (14)$$

A good LM should maximize the text set probability. It is equivalent to minimizing the perplexity. The lower the perplexity, the better the LM.

**Bidirectional Language Model.** To calculate the inverse probability in Eq. (13), the auto-regressive LMs can use a sequence of conditional probabilities. However, this approach does not work for bidirectional LMs (or masked LMs). Several intrinsic evaluation metrics have been proposed for bidirectional LMs. The pseudo-log-likelihood score (PLL) [90] is defined as

$$PLL(W) = \sum_{t=1}^{|W|} \log P(w_t | W_{\setminus t}), \quad (15)$$

where  $\log P(w_t | W_{\setminus t})$  is the conditional probability of token  $w$  in sentence  $W$  with all remaining tokens. Instead of maximizing the joint probability of the entire text sequence, a good bidirectional LM should maximize the probability of each token in the text sequence given other tokens. Based on PLLs, the pseudo-Perplexity (PPPL) for corpora  $C$  is defined as [91]

$$PPL(C) = \exp\left(-\frac{1}{N} \sum_{W \in C} PLL(W)\right). \quad (16)$$

Both PLL and PPPL provide effective means to measure the naturalness of sentences for a bidirectional LM. For example, it was shown in [91] that PLL and PPPL correlate well with the performance of an LM on downstream tasks, such as automatic speech recognition and machine translation.

## 6.2 Extrinsic Evaluation

Any downstream task of LMs can be used for extrinsic evaluation. There are several common downstream tasks selected as extrinsic evaluation benchmarks. Two popular ones are GLUE (General Language Understanding Evaluation) [92] and SuperGLUE [93]. GLU is an evaluation benchmark for natural language understanding. It contains single-sentence tasks, similarity and paraphrase tasks, and inference tasks. SuperGLUE is an enhanced version of GLUE. It includes a new set of more challenging language understanding tasks, more diverse task formats, improved resources, and a public leaderboard.

## 6.3 Relation between Intrinsic and Extrinsic Evaluations

If an LM achieves a lower perplexity, does that mean it can also perform well on downstream tasks? In other words, is there any correlation between pre-training tasks (based on word prediction) and the downstream tasks? There are many empirical studies on this question but few theoretical studies.

**Empirical Studies.** Researchers design experiments to understand what kind of knowledge is learned by an LM from the pre-training tasks. Examples include [94, 95, 96, 97, 98, 99]. They use part-of-speech tagging, constituent labeling, and dependency labeling to measure the degree of syntactic knowledge learning, and named entity labeling, semantic role labeling, and semantic proto-role for testing semantic knowledge. Empirical studies show that pre-training tasks help LMs learn the linguistic knowledge such as the grammar [97] and the semantic role [99]. However, these experimental results can only be used as evidence supporting that the word prediction tasks benefit downstream tasks. They cannot explain the underlying mechanism.

**Theoretical Studies.** Some researchers attempt to build the connection between LM's perplexities and its performance on downstream tasks mathematically. The text classification tasks were studied in [100]. They first hypothesized and

verified that text classification tasks can be reformulated as sentence completion tasks. Since the LM pre-training task is essentially a sentence completion task, it does help the text classification downstream task. Then, they quantified the connection mathematically and showed that the features from LMs that achieve  $\epsilon$ -optimal in log-perplexity can linearly solve text classification tasks with  $O(\sqrt{\epsilon})$  error. An underlying generative model was utilized in [101] to show the relationship between the pre-training tasks and the downstream tasks. Current theoretical studies are limited in the sense that only a specific downstream task (say, the text classification task) is considered and the proof holds under certain conditions.

## 7 Applications in Text Generation

One of the most important applications of LMs is text generation, which aims to generate sequences of words based on the input data. There are many text generation tasks because of different purposes and inputs. For example, the automatic speech recognition (ASR) task demands that the input be a speech sequence while the output be the corresponding text sequence. The machine translation task generates the translated text sequence based on the input text sequence and the target language. Story Generation is a topic-to-text generation task. In this section, we introduce common techniques used in text generation and then explain how LMs can be applied in each of the representative tasks.

## 7.1 Decoding Methods

Decoding decides the next output linguistic unit to generate text. A good decoding method should generate coherent continuation given a context. As LMs get more sophisticated, decoding methods have played an increasingly important role. As shown in Fig. 9, deficient decoding methods lead to bad generated texts even with a powerful LM. There are two main decoding methods for text generation.

**Context:** In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

### Beam Search, $b=32$ :

[illegible]

### Pure Sampling:

They were cattle called **Bolivian Cavalleros**; they live in a remote desert **uninterrupted by town**, and they speak **huge beautiful, paradisiacal Bolivian linguistic thing**. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

Figure 9: Comparison of texts generated by the powerful GPT-2 large language model (LLM) using Beam search (left) and pure sampling decoding (right). Beam search yields degenerate repetition (in blue) while pure sampling results in incoherent gibberish (in red) [102].

**Maximization-based decoding.** This is the most commonly used decoding objective. Assuming that the model assigns a higher probability to a higher quality text which is closer to the ground truth written by humans, the maximization-based decoding strategy searches for tokens with the highest probability as the generated text. Greedy search [103, 104] chooses the token with the highest probability as the next token in a greedy manner. Beam search [105, 106, 107] keeps a certain number of most likely tokens at each time step and selects the generated token sequences with the overall highest probability eventually. It avoids missing reasonable tokens that do not have the highest probability. Trainable decoding algorithms have been proposed recently. Trainable greedy decoding [108] is a neural-based solution that works as part of a neural machine translation decoder. It utilizes reinforcement learning to find a translation that maximizes a decoding objective.

**Sampling-based decoding.** It chooses the next token from a set of sampled tokens. Because maximization-based decoding depends highly on the underlying model probabilities and suffers from producing degenerate repetition, sampling-based decoding increases the diversity of generated texts by random sampling. However, the simple pure sampling may choose a token with low probability (from an unreliable tail distribution) as the next generated token. As a result, the generated text could be unrelated to the prefix, leading to incoherent gibberish. Top-k sampling [109] and Nucleus sampling [102] have recently been proposed to address this problem. Both sample from truncated LM distributions (i.e., sampling from the most probable tokens). Diverse Beam search [105] is a trainable sampling-based

(stochastic) decoding algorithm based on the Beam search. It uses reinforcement learning to determine the beam diversity parameters for different inputs or tasks.

## 7.2 Dialogue Systems

A dialogue system aims at simulating human responses when conversing with human users. Recent dialogue systems such as ChatGPT<sup>1</sup> and LaMDA [110] have attracted a lot of attention in the generative AI field because of their superior performance as interactive chatbot systems. Dialogue systems can be categorized into task-oriented systems and open-domain systems. The former is designed for specific tasks such as customer service for online shopping. The latter is also known as chatbots [111]. Most modern dialogue systems are fine-tuned versions of generative LMs. Taking ChatGPT as an example, ChatGPT is built based on a generative LM, GPT-3 [85], with over 188 billion parameters. It is further fine-tuned by supervised learning and reinforcement learning on labeled data.

LMs play an important role in dialogue systems, especially in their natural language understanding (NLU) and natural language generation (NLG) components [112, 113]. NLU is responsible for understanding and recognizing users' intent. Nowadays, PLM encoders provide informative representations for NLU while the associated PLM decoders are responsible for generating an appropriate response. The latter involves constructing the response text, selecting appropriate words, and determining the correct phrasing and tone. The effectiveness of representations of PLMs was examined in [114] for dialogue tasks. The evaluation PLM targets included BERT [15] and GPT2 [42]. The few-shot capability of PLMs in dialogue tasks such as NLU and NLG was evaluated in [115]. Overall, LMs in dialogue systems play a key role in understanding users' input and generating appropriate and natural responses.

## 7.3 Automatic Speech Recognition

Automatic speech recognition (ASR) is a speech-to-text generation task that aims to transform raw audio input into the corresponding text sequence. The LM plays an essential role in an ASR system. First, it helps solve acoustically ambiguous utterances. Second, it can lower the computational cost by constraining the search space in a set of words of higher probability. ASR systems contain an acoustic model and a language model, which are related by

$$P(\text{word}|\text{sound}) \propto P(\text{sound}|\text{word})P(\text{word}). \quad (17)$$

The acoustic model is conditioned on phones  $P(\text{sound}|\text{word})$  while the LM gives the word distribution denoted by  $P(\text{word})$ . LMs help search the word hypotheses during recognition. Different types of LMs have been explored in ASR such as N-gram [116, 117], FFNN [118], RNN [119, 120] and Transformer [121]

## 7.4 Machine Translation

Machine translation is a text-to-text generation task where the text in the source language is translated into that of the target language. LMs adopted by machine translation are conditioned on both the source sentence and the previous partial translation. Recently, transformer-based models has achieved great successes in machine translation [82, 122, 85]

## 7.5 Detection of Generated texts

As the performance of LMs gets closer to or even outperforms humans, the misuse of LMs, such as fake news and fake product reviews generation, has become a serious problem. The ability to detect machine-generated texts is important. There are two types of detection problems: 1) human written vs. machine generated, and 2) inercacious vs. veracious. Most datasets, e.g., [123, 124, 125], are collected for the first type. Problems of the second type are much harder than those of the first type [126] since one needs to connect generated text to the fact, which requires a high-level knowledge reasoning capability.

Two common approaches to detecting machine-generated text are reviewed below. One is to exploit the probability distribution of LMs [127, 128]. If the probability distribution of a text sequence is closer to that of human-written texts as compared with known machine-generated texts, the text sequence is classified as human-written. The other is to train classifiers with supervised learning [129, 124]. It converts the distribution to a supervised binary classification task. For more details on the detection of machine-generated texts, readers are referred to two survey papers [130, 131].

---

<sup>1</sup><https://openai.com/blog/chatgpt/>

## 8 Efficient Models

As recent PLMs get more powerful, their model size, training cost, and demand for training data increase tremendously. They need high computational resources and energy consumption, limiting their real-world applications. Table 2 shows the model size, training data, cost, and time of recently developed LMs. This issue is a concern to many people and the construction of efficient LMs has received attention.

Model	Year	Model size	Training data	Training cost	Training time
BERT-Large	2018	340M	2.5B words	\$7,000	64 TPU chips 4 days
XLNet-Lagre	2019	340M	32.9B words	\$245,000	512 TPU v3 chips 5.5 days
GPT-2	2019	1.5B	8 million web pages	\$60,000	
Megatron-LM	2019	8.3B	174 GB deduplicated text		512 GPUs 53 minutes
T5	2019	11B		Above \$1.3 million for a single run	
Turing-NLG	2020	17.2B			
GPT-3	2020	175B	45TB of text data	\$12 million	
Megatron-Turing NLG	2021	530B	270B		2K A100 GPUs 3 months

Table 2: Comparison of model sizes, training data, cost, and time of several large LMs, where blank cells indicate that the data are not available.

### 8.1 Data Usage

**Pre-training Data Size.** A critical question for PLM training is how much data is needed. The effect of the pre-training data size on the RoBERTa model was studied in [132]. The learning curves of four model performance measures as a function of the pre-training dataset size are shown in Fig. 10. When the data size ranges between 100M and 1B words, three learning curves gradually level off and it implies that LMs encode most syntactic and semantic features. However, a much larger quantity of data is needed for LMs to acquire enough common-sense knowledge and other skills to achieve better performance on downstream NLU tasks.

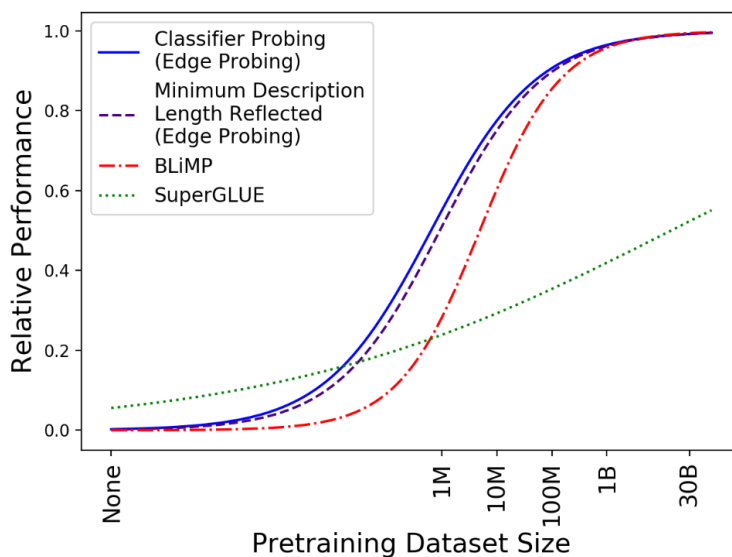


Figure 10: The performance curves as functions of the pre-training dataset size, where the classifier probing measures the quality of the syntactic and semantic features, the minimum description length probing quantifies the accessibility of these features, the BLiMP curve measures the model’s knowledge of various syntactic phenomena, and the superGLUE measures the capability of handling NLU tasks [132].

**Efficient Pre-Training.** Several methods have been proposed to use the pre-training data more efficiently. In the pre-training of masked LMs, a certain percentage of tokens are masked and need to be inferred by context. This approach incurs a substantial amount of computational cost because the network only learns from a certain percentage of tokens which are masked. To enhance training efficiency, the work in [45] uses “replaced token detection” (rather than “masked token prediction”) as the pre-training task. As shown in Fig. 11, a generator is trained to perform the masked LM and predicts the masked tokens. Then, the main model works as a discriminator, called ELECTRA, which learns to decide the original or replaced tokens. In this way, pre-training tasks are conducted on all tokens instead of a small subset of masked tokens. Learning from all input positions causes ELECTRA to train much faster than BERT which adopts masked word prediction. Besides, ELECTRA achieves higher accuracy on downstream tasks when it is fully trained. Later, a new pre-training task using an energy-based model, which is closely related to ELECTRA, is proposed in [133].

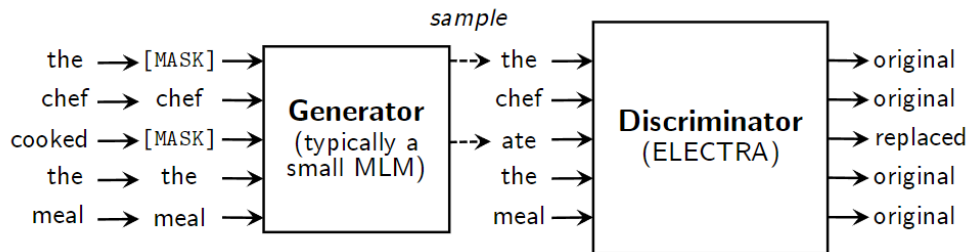


Figure 11: The structure of ELECTRA (Efficiently Learning an Encoder that Classifier Token Replacements Accurately) [45]

**Bridging Pre-training and Downstream Tasks.** A typical pre-training task is word prediction, which often has a large gap with downstream tasks. To mitigate the gap between pre-training and downstream tasks, prompt tuning has been studied in [84, 42, 89, 134, 135]. As illustrated in Fig. 8, the head is trained to predict the masked words in masked LMs. For the downstream sentiment analysis task, the head is trained to predict the positive or the negative label in traditional fine-tuning. A template (e.g., ‘It was’) and its expected text responses (e.g., ‘great’ and ‘terrible’) are used in prompt tuning. In this way, pre-training and prompt tuning share the same “word prediction” objective.

## 8.2 Model Size

Besides improving training efficiency, efficient LMs focus on the design of models of smaller sizes. Many methods are investigated to reduce the model size so that the model can be implemented on mobile or edge devices with limited computing resources. Model compression is a widely studied topic. Compression methods first train a large LM and then compress it into a target size. Examples include model pruning [136, 137, 138], knowledge distillation [139, 44, 140], low rank matrix approximation [78, 141, 142], and parameter sharing [143, 144, 145].

## 9 Future Research Directions

In this section, we describe several promising future research directions in language modeling.

### 9.1 Integration of LMs and KGs

Knowledge Graph (KG) provides a structured representation of human knowledge [146, 147]. It has been widely used in many NLP applications, such as question answering [148] and text summarization [149], because of its capability to represent relationships between entities. There is a growing interest in evaluating the knowledge learned in PLMs [150], where the relationship between different semantic units is captured in the embedding space and the self-attention layers. Several ideas are proposed in [151] to leverage KGs for LM training. As a result, the knowledge learned in the models can be greatly improved. Thus, it is worth careful investigation of integrating KGs with LMs and understanding how they interact with each other.

It appears that KG can serve as an information database to be queried by LMs. LMs are powerful in natural language understanding and generation while KGs can organize and store the knowledge information extracted from the training corpus. In other words, we may decompose knowledge sources into semantic and syntactic two components, which can be handled by KGs and LMs, respectively. In the training phase, a KG is constructed based on the information extracted from the training corpus, and an LM can be trained simultaneously. In the inference phase, an LM can serve



as an interface between humans and the knowledge database represented in form of KGs. There are advantages to assigning semantic and syntactic processing tasks to KGs and LMs, respectively. For example, the decoupling facilitates incremental learning, allows a smaller model size, and improves interpretability. They will be further elaborated on below.

## 9.2 Incremental Learning

Incremental learning aims to incorporate new information without re-training existing models entirely. The problem of catastrophic forgetting associated with neural network models was pointed out in [152]. That is, the information that has already been learned by a model can be gradually forgotten when training with new information. This problem is particularly critical to large LMs since the new information keeps arriving. A solution to catastrophic forgetting was proposed in [153]. It attempts to remember prior important tasks by slowing down learning on weights that are more relevant to them. However, it is difficult to define important tasks in LMs. Re-training a large LM with both old and new data is too expensive.

The importance of developing a satisfactory solution to incremental learning for LMs cannot be over-emphasized. Incremental learning is challenging for neural networks. Yet, it is easy for KGs to add new data to (or remove old data from) an existing database by adding or removing factual triples [154]. Clearly, the current information in the KGs will not be overwritten by newly collected data. The information in the database is updated incrementally. To this end, the integration of KGs and LMs provides an excellent solution that meets the need for incremental learning.

## 9.3 Lightweight Models

As mentioned in Section 8, PLMs get more powerful at the expense of huge computational resources and energy consumption. The cost issue has to be faced seriously in the development of large LMs (LLMs). Besides, LLMs are unfriendly to our environment due to their high carbon footprint. Green Learning (GL) targets learning solutions with low carbon footprint. The design of lightweight models of smaller sizes and lower computational complexity without sacrificing performance has received more attention in recent years [155, 156, 157, 158]. The design of green LMs is an important topic worth serious investigation.

Current PLMs are data-driven models that use neural architectures to learn generic language knowledge from a large amount of data. Efforts have been made in the development of lightweight LMs. Model compression is one of the popular approaches to obtaining a small LM. Examples include knowledge distillation or pruning [159]. However, this methodology appears to be a detour since it trains large models and then shrinks their sizes by compression. Instead, we may incorporate the linguistic information and the domain knowledge to offer a more direct way to reduce the model size and the amount of training data.

## 9.4 Universal versus Domain-Specific Models

A universal LM is developed to handle tasks in the general domain. For example, ChatGPT is a universal dialogue LM pre-trained on multilingual and general domain corpora. It can converse on open-domain topics in multiple languages. In contrast, domain-specific LMs [160, 161, 162, 163] are designed to deal with domain-specific tasks, e.g., biomedicine, economics, musicology, etc.

A universal LM demands a huge model size, a large number of training examples, and a tremendous amount of computational resources. Based on the scaling law of neural language models [164], the inference performance scales as a power-law with the model size, the dataset size, and the amount of computing used for training. So far, the largest PLM contains 540-billion parameters [83]. Despite the superior performance and the flexibility to adapt to multiple tasks, we may wonder whether a huge universal LM is cost-effective.

For domain-specific LMs, the amount of training data in need is significantly lower. It was believed that the general domain PLMs benefit the training of domain-specific LMs. However, it is reported in [161] that domain-specific LMs, which were pre-trained from scratch on in-domain data, can provide a solid foundation for biomedical NLP. In other words, training a domain-specific LM may not need a huge amount of general corpora and labeled data. Domain-specific LMs to be deployed on task-specific scenarios with less training and inference efforts expect to receive more attention in the future.

## 9.5 Interpretable Models

Although deep-learning-based LMs are dominating in the NLP field, they are inherently black-box methods without mathematical transparency. Its interpretability is of concern. Efforts have been made to explain the black-box LMs. As



mentioned in 6.3, empirical studies are conducted to understand what PLMs have learned through experimental design. However, the progress in this direction may offer insights but not a satisfactory and clean answer. Providing theoretical explanations or establishing explainable LMs is still a challenging and open issue. A direction to interpretability is to design an interpretable learning model from scratch. For example, we may incorporate KGs with LMs. KGs are proven to be powerful in many reasoning tasks such as information retrieval [165] and recommendation systems [166]. KGs can provide a logical path for each prediction so that predictions offered by LMs can be more explainable.

## 9.6 Machine Generated Text Detection

The most common application of LMs is text generation. As generative LM's performance gets closer to or even outperforms humans, these LMs can be used for malicious purposes such as academic dishonesty, spamming, targeted bot attacks, and fake news/reviews generation. How to determine whether a text is generated by LMs or written by humans is a big challenge nowadays. A high-performance machine-generated text classifier can only serve as a reference in real-world applications, since it has false positives (i.e., human-written texts classified as machine-generated) and false negatives (i.e., machine-generated texts classified as human-written). In addition, people may be even more interested in detecting veracious and unveracious texts. They care more about whether the text is true or not. Detecting disinformation could be more difficult than detecting machine/human-generated text without assessing the factuality. Additionally, the factuality may change as time goes by. It is critical to our society in developing effective tools to identify malicious usages of generative LMs.

## 10 Conclusion

A comprehensive overview of CLMs and their successors, PLMs, was presented in this paper and a wide range of topics was covered. First, different levels of linguistic units were introduced and how linguistic unit prediction is used to train language models was examined. Second, tokenization methods adopted by language models were discussed. Third, language model structures and the training paradigm of PLMs were reviewed. Fourth, we studied the evaluation and applications of language models. Especially, several applications in the context of text generation were detailed. Finally, several future research directions were pointed out. The need for explainable, reliable, domain-specific, and lightweight language models was emphasized.

## References

- [1] Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480, 1992.
- [2] Marcello Federico. Bayesian estimation methods for n-gram language model adaptation. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 1, pages 240–243. IEEE, 1996.
- [3] Thomas R Niesler and Philip C Woodland. A variable-length category-based n-gram language model. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 164–167. IEEE, 1996.
- [4] Stephen A Della Pietra, Vincent J Della Pietra, Robert L Mercer, and Salim Roukos. Adaptive language modeling using minimum discriminant estimation. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.
- [5] Adam Berger, Stephen A Della Pietra, and Vincent J Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.
- [6] Roni Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. 1996.
- [7] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.
- [8] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.
- [9] Frederick Jelinek, Lalit Bahl, and Robert Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, 21(3):250–256, 1975.
- [10] Frederick Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, 1976.
- [11] Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, (2):179–190, 1983.

- [12] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederick Jelinek, John Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.
- [13] Franz Josef Och, Nicola Ueffing, and Hermann Ney. An efficient a\* search algorithm for statistical machine translation. In *Proceedings of the ACL 2001 Workshop on Data-Driven Methods in Machine Translation*, 2001.
- [14] Kenji Yamada and Kevin Knight. A decoder for syntax-based statistical mt. In *Proceedings of the 40th Annual meeting of the Association for Computational Linguistics*, pages 303–310, 2002.
- [15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [17] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke 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)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
- [18] Chengwei Wei, Bin Wang, and C.-C. Jay Kuo. Synwmd: Syntax-aware word mover’s distance for sentence similarity evaluation. *arXiv preprint arXiv:2206.10029*, 2022.
- [19] Bin Wang and C.-C. Jay Kuo. SBERT-WK: A sentence embedding method by dissecting bert-based word models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2146–2157, 2020.
- [20] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SIMCSE: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.
- [21] Bin Wang and Haizhou Li. Relational sentence embedding for flexible semantic matching. *arXiv preprint arXiv:2212.08802*, 2022.
- [22] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.
- [23] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.
- [24] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021.
- [25] Ciprian Chelba and Frederick Jelinek. Exploiting syntactic structure for language modeling. *arXiv preprint cs/9811022*, 1998.
- [26] Ciprian Chelba and Frederick Jelinek. Structured language modeling. *Computer Speech & Language*, 14(4):283–332, 2000.
- [27] Joseph Gubbins and Andreas Vlachos. Dependency language models for sentence completion. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1405–1410, 2013.
- [28] Piotr Mirowski and Andreas Vlachos. Dependency recurrent neural language models for sentence completion. *arXiv preprint arXiv:1507.01193*, 2015.
- [29] Chengwei Wei, Bin Wang, and C.-C. Jay Kuo. Task-specific dependency-based word embedding methods. *Pattern Recognition Letters*, 2022.
- [30] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.
- [31] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *ICML*, 2011.
- [32] Kyuhyeon Hwang and Wonyong Sung. Character-level language modeling with hierarchical recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5720–5724. IEEE, 2017.
- [33] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

- [34] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3159–3166, 2019.
- [35] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.
- [36] Moonyoung Kang, Tim Ng, and Long Nguyen. Mandarin word-character hybrid-input neural network language model. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.
- [37] Yasumasa Miyamoto and Kyunghyun Cho. Gated word-character recurrent language model. *arXiv preprint arXiv:1606.01700*, 2016.
- [38] Lyan Verwimp, Joris Pelemans, Patrick Wambacq, et al. Character-word lstm language models. *arXiv preprint arXiv:1704.02813*, 2017.
- [39] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. Subword language modeling with neural networks. *preprint (http://www.fit.vutbr.cz/imikolov/rnnlm/char.pdf)*, 8(67), 2012.
- [40] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.
- [41] Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.
- [42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [43] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE, 2012.
- [44] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
- [45] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.
- [46] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.
- [47] Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*, 2020.
- [48] Daniel Kiecza, Tanja Schultz, and Alex Waibel. Data-driven determination of appropriate dictionary units for korean lvsr. In *Proceedings of ICASSP*, pages 323–327, 1999.
- [49] Mathias Creutz and Krista Lagus. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology Helsinki, 2005.
- [50] Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pytkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(1):1–29, 2007.
- [51] Tomaž Rotovnik, Mirjam Sepesy Maučec, and Zdravko Kačič. Large vocabulary continuous speech recognition of an inflected language using stems and endings. *Speech communication*, 49(6):437–452, 2007.
- [52] Ruhi Sarikaya, Mohamed Afify, and Yuqing Gao. Joint morphological-lexical language modeling (jmlm) for arabic. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP’07*, volume 4, pages IV–181. IEEE, 2007.
- [53] Haşim Sak, Murat Saraçlar, and Tunga Güngör. Morphology-based and sub-word language modeling for turkish speech recognition. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5402–5405. IEEE, 2010.
- [54] Mijit Ablimit, Graham Neubig, Masato Mimura, Shinsuke Mori, Tatsuya Kawahara, and Askar Hamdulla. Uyghur morpheme-based language models and asr. In *IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS*, pages 581–584. IEEE, 2010.
- [55] Bernhard Suhm and Alex Waibel. Towards better language models for spontaneous speech. 1994.

- [56] Klaus Ries, Finn Dag Buo, and Alex Waibel. Class phrase models for language modeling. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 1, pages 398–401. IEEE, 1996.
- [57] George Saon and Mukund Padmanabhan. Data-driven approach to designing compound words for continuous speech recognition. *IEEE transactions on Speech and audio processing*, 9(4):327–332, 2001.
- [58] Michael Levit, Sarangarajan Parthasarathy, Shuangyu Chang, Andreas Stolcke, and Benoit Dumoulin. Word-phrase-entity language models: Getting more mileage out of n-grams. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.
- [59] Ronald Rosenfeld. A whole sentence maximum entropy language model. In *1997 IEEE workshop on automatic speech recognition and understanding proceedings*, pages 230–237. IEEE, 1997.
- [60] Stanley F Chen and Ronald Rosenfeld. Efficient sampling and feature selection in whole sentence maximum entropy language models. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 1, pages 549–552. IEEE, 1999.
- [61] Ronald Rosenfeld, Stanley F Chen, and Xiaojin Zhu. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computer Speech & Language*, 15(1):55–73, 2001.
- [62] Daphne Ippolito, David Grangier, Douglas Eck, and Chris Callison-Burch. Toward better storylines with sentence-level language models. *arXiv preprint arXiv:2005.05255*, 2020.
- [63] Haejun Lee, Drew A Hudson, Kangwook Lee, and Christopher D Manning. Slm: Learning a discourse language representation with sentence unshuffling. *arXiv preprint arXiv:2010.16249*, 2020.
- [64] George James Lidstone. Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8(182-192):13, 1920.
- [65] William Ernest Johnson. Probability: The deductive and inductive problems. *Mind*, 41(164):409–423, 1932.
- [66] Frederick Jelinek. Interpolated estimation of markov source parameters from sparse data. In *Proc. Workshop on Pattern Recognition in Practice, 1980*, 1980.
- [67] Slava Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401, 1987.
- [68] Kenneth W Church and William A Gale. A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams. *Computer Speech & Language*, 5(1):19–54, 1991.
- [69] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *1995 international conference on acoustics, speech, and signal processing*, volume 1, pages 181–184. IEEE, 1995.
- [70] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.
- [71] John N Darroch and Douglas Ratcliff. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, pages 1470–1480, 1972.
- [72] Holger Schwenk and Jean-Luc Gauvain. Training neural network language models on very large corpora. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 201–208, 2005.
- [73] Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.
- [74] Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28, 2012.
- [75] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531. IEEE, 2011.
- [76] Stefan Kombrink, Tomas Mikolov, Martin Karafiát, and Lukáš Burget. Recurrent neural network based language modeling in meeting recognition. In *Interspeech*, volume 11, pages 2877–2880, 2011.
- [77] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.
- [78] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

- [79] Hongli Deng, Lei Zhang, and Lituan Wang. Global context-dependent recurrent neural network language model with sparse feature learning. *Neural Computing and Applications*, 31(2):999–1011, 2019.
- [80] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.
- [81] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [82] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [83] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- [84] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.
- [85] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [86] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.
- [87] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.
- [88] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.
- [89] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.
- [90] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.
- [91] Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. Masked language model scoring. *arXiv preprint arXiv:1910.14659*, 2019.
- [92] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.
- [93] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.
- [94] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.
- [95] Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. *arXiv preprint arXiv:1808.08079*, 2018.
- [96] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.
- [97] Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang-goo Lee. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. *arXiv preprint arXiv:2002.00737*, 2020.
- [98] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019.
- [99] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

- [100] Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. A mathematical exploration of why language models help solve downstream tasks. *arXiv preprint arXiv:2010.03648*, 2020.
- [101] Colin Wei, Sang Michael Xie, and Tengyu Ma. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. *Advances in Neural Information Processing Systems*, 34:16158–16170, 2021.
- [102] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.
- [103] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*, 2017.
- [104] Zhen Xu, Bingquan Liu, Baoxun Wang, Cheng-Jie Sun, Xiaolong Wang, Zhuoran Wang, and Chao Qi. Neural response generation via gan with an approximate embedding layer. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 617–626, 2017.
- [105] Jiwei Li, Will Monroe, and Dan Jurafsky. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*, 2016.
- [106] Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [107] Iliia Kulikov, Alexander H Miller, Kyunghyun Cho, and Jason Weston. Importance of search and evaluation strategies in neural dialogue modeling. *arXiv preprint arXiv:1811.00907*, 2018.
- [108] Jiatao Gu, Kyunghyun Cho, and Victor OK Li. Trainable greedy decoding for neural machine translation. *arXiv preprint arXiv:1702.02429*, 2017.
- [109] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.
- [110] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.
- [111] Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, and Erik Cambria. Recent advances in deep learning based dialogue systems: A systematic survey. *Artificial intelligence review*, pages 1–101, 2022.
- [112] Bin Wang, Chen Zhang, Chengwei Wei, and Haizhou Li. A focused study on sequence length for dialogue summarization. *arXiv preprint arXiv:2209.11910*, 2022.
- [113] Bin Wang, Chen Zhang, Yan Zhang, Yiming Chen, and Haizhou Li. Analyzing and evaluating faithfulness in dialogue summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4897–4908, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.
- [114] Chien-Sheng Wu and Caiming Xiong. Probing task-oriented dialogue representation from language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5036–5051, Online, November 2020. Association for Computational Linguistics.
- [115] Andrea Madotto, Zihan Liu, Zhaojiang Lin, and Pascale Fung. Language models as few-shot learner for task-oriented dialogue systems. *arXiv preprint arXiv:2008.06239*, 2020.
- [116] Fred Jelinek, B Meriello, S Roukos, M Strauss, et al. Self-organized language modeling for speech recognition. In *Readings in speech recognition*. Citeseer, 1990.
- [117] Manhung Siu and Mari Ostendorf. Variable n-grams and extensions for conversational speech language modeling. *IEEE Transactions on Speech and Audio Processing*, 8(1):63–75, 2000.
- [118] Ebru Arisoy, Stanley F Chen, Bhuvana Ramabhadran, and Abhinav Sethy. Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):184–192, 2013.
- [119] Ebru Arisoy, Abhinav Sethy, Bhuvana Ramabhadran, and Stanley Chen. Bidirectional recurrent neural network language models for automatic speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5421–5425. IEEE, 2015.
- [120] Zhiheng Huang, Geoffrey Zweig, and Benoit Dumoulin. Cache based recurrent neural network language model inference for first pass speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6354–6358. IEEE, 2014.

- [121] Joonbo Shin, Yoonhyung Lee, and Kyomin Jung. Effective sentence scoring method using bert for speech recognition. In *Asian Conference on Machine Learning*, pages 1081–1093. PMLR, 2019.
- [122] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.
- [123] Max Weiss. Deepfake bot submissions to federal public comment websites cannot be distinguished from human submissions. *Technology Science*, 2019.
- [124] Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. Authorship attribution for neural text generation. In *Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
- [125] Tiziano Fagni, Fabrizio Falchi, Margherita Gambini, Antonio Martella, and Maurizio Tesconi. Tweepfake: About detecting deepfake tweets. *Plos one*, 16(5):e0251415, 2021.
- [126] James Thorne and Andreas Vlachos. Automated fact checking: Task formulations, methods and future directions. *arXiv preprint arXiv:1806.07687*, 2018.
- [127] Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. Automatic detection of generated text is easiest when humans are fooled. *arXiv preprint arXiv:1911.00650*, 2019.
- [128] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*, 2019.
- [129] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. *Advances in neural information processing systems*, 32, 2019.
- [130] Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks VS Lakshmanan. Automatic detection of machine generated text: A critical survey. *arXiv preprint arXiv:2011.01314*, 2020.
- [131] Harald Stiff and Fredrik Johansson. Detecting computer-generated disinformation. *International Journal of Data Science and Analytics*, 13(4):363–383, 2022.
- [132] Yian Zhang, Alex Warstadt, Haau-Sing Li, and Samuel R Bowman. When do you need billions of words of pretraining data? *arXiv preprint arXiv:2011.04946*, 2020.
- [133] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Pre-training transformers as energy-based cloze models. *arXiv preprint arXiv:2012.08561*, 2020.
- [134] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020.
- [135] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.
- [136] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.
- [137] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.
- [138] Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.
- [139] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.
- [140] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*, 2019.
- [141] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32, 2019.
- [142] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [143] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.
- [144] Raj Dabre and Atsushi Fujita. Recurrent stacking of layers for compact neural machine translation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6292–6299, 2019.
- [145] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280, 2020.

- [146] Yun-Cheng Wang, Xiou Ge, Bin Wang, and C-C Jay Kuo. Kgboost: A classification-based knowledge base completion method with negative sampling. *Pattern Recognition Letters*, 157:104–111, 2022.
- [147] Xiou Ge, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. Compounde: Knowledge graph embedding with translation, rotation and scaling compound operations. *arXiv preprint arXiv:2207.05324*, 2022.
- [148] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 105–113, 2019.
- [149] Luyang Huang, Lingfei Wu, and Lu Wang. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. *arXiv preprint arXiv:2005.01159*, 2020.
- [150] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.
- [151] Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. *arXiv preprint arXiv:2010.12688*, 2020.
- [152] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.
- [153] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- [154] Bin Wang, Guangtao Wang, Jing Huang, Jiaxuan You, Jure Leskovec, and C-C Jay Kuo. Inductive learning on commonsense knowledge graph completion. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.
- [155] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.
- [156] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. A survey on green deep learning. *arXiv preprint arXiv:2111.05193*, 2021.
- [157] C.-C. Jay Kuo and Azad M Madni. Green learning: Introduction, examples and outlook. *Journal of Visual Communication and Image Representation*, page 103685, 2022.
- [158] Yun-Cheng Wang, Xiou Ge, Bin Wang, and C-C Jay Kuo. GreenKGC: A lightweight knowledge graph completion method. *arXiv preprint arXiv:2208.09137*, 2022.
- [159] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on Machine Learning*, pages 5958–5968. PMLR, 2020.
- [160] Denghui Zhang, Zixuan Yuan, Yanchi Liu, Fuzhen Zhuang, Haifeng Chen, and Hui Xiong. E-bert: A phrase and product knowledge enhanced language model for e-commerce. *arXiv preprint arXiv:2009.02835*, 2020.
- [161] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23, 2021.
- [162] Kesong Liu, Jianhui Jiang, and Feifei Lyu. A domain knowledge enhanced pre-trained language model for vertical search: Case study on medicinal products. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1014–1023, 2022.
- [163] Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23(6), 2022.
- [164] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [165] Laura Dietz, Chenyan Xiong, Jeff Dalton, and Edgar Meij. The second workshop on knowledge graphs and semantics for text retrieval, analysis, and understanding (kg4ir). In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1423–1426, 2018.
- [166] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. Knowledge graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1434–1443, 2022.