Novel Methods for Natural Language Modeling and Pretraining

by

He Bai

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Jimmy Huang Professor, School of Information Technology, York University
Supervisor(s):	Ming Li Professor, Dept. of Computer Science, University of Waterloo
Internal Member:	Jimmy Lin Professor, Dept. of Computer Science, University of Waterloo
	Charles Clarke Professor, Dept. of Computer Science, University of Waterloo
Internal-External Member:	Helen Chen Professor of Practice, School of Public Health Sciences, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contribution

I would like to thank all my co-authors who made this thesis possible. And here, I details the contributions of each co-author for all work presented in my thesis.

- Chapter 2. **He Bai**, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, Ming Li. **Segatron: Segment-Aware Transformer for Language Modeling and Understanding.** ***AAAI 2021*** *(full paper).*

  This work has been accepted by AAAI 2021 as a long paper in the main conference. I am the first author who proposed the main idea, conducted most of the experiments, and this work was collaborated with researchers at RSVP and Peking University. Ming and Jimmy gave advice on experiments and evaluations. Luchen gave advice on visualization. Peng and Yuqing contributed some results of SQUAD dataset. Computational resources were supported by Wen. All co-authors reviewed the paper and gave feedback for multiple rounds.

- Chapter 3. **He Bai**, Tong Wang, Alessandro Sordoni, Peng Shi. **Better Language Model with Hypernym Class Prediction.** ***ACL 2022*** *(full paper).*

  This work has been accepted by ACL 2022 as a long paper in the main conference. I am the first author, conducted all the experiments, and most of this work was done during my internship at Microsoft Research. The idea was formulated and developed from the weekly discussion with Tong. Alessandro gave advice on rare words comparisons, evaluations, and visualization. All co-authors reviewed the paper and gave feedbacks.

- Chapter 4. **He Bai**, Renjie Zheng, Junkun Chen, Xintong Li, Mingbo Ma, Liang Huang. **A^{3}T: Alignment-Aware Acoustic and Text Pretraining for Speech Synthesis and Editing.** ***ICML 2022*** *(full paper).*

  This work has been accepted by ICML 2022 as a spotlight long paper in the main conference. I am the first author, conducted all the experiments, and this work was collaborated with researchers at Baidu Research USA and Oregon State University. The initial idea was formulated from my discussions with Mingbo, Renjie, and Liang. I proposed the alignment-aware structure. Renjie gave advice on datasets, experiments, and evaluations. Renjie also conducted the AmazonTurk human evaluations. Liang gave advices on visualization and ablation study. Xintong gave advice on speech editing baselines. Junkun helped the codes of speech masking. All co-authors reviewed the paper and gave feedbacks in multiple rounds.

Abstract

This thesis is about modeling language sequences to achieve lower perplexity, better generation, and benefit downstream language tasks; specifically, this thesis addresses the importance of natural language features including the segmentation feature, lexical feature, and alignment feature. We present three new techniques that improve language sequence modeling with different language features.

Segment-Aware Language Modeling is a novel model architecture leveraging the text segementation feature for text sequence modeling. It encodes richer positional information for language modeling, by replacing the original token position encoding with a combined position encoding of paragraph, sentence, and token. By applying our approach to Transformer-XL, we train a new language model, Segatron-XL, that achieves a 6.6-7.8% relative reduction in perplexity. Additionally, BERT pretrained with our method – SegaBERT – outperforms BERT on general language understanding, sentence representation learning, and machine reading comprehension tasks. Furthermore, our SegaBERT-large model outperforms RoBERTa-large on zero-shot STS tasks. These experimental results demonstrate that our proposed Segatron works on both language models with relative position embeddings and pretrained language models with absolute position embeddings.

Hypernym-Instructed Language Modeling is a novel training method leveraging the lexical feature for rare word modeling. It maps words that have a common WordNet hypernym to the same class and trains large neural LMs by gradually annealing from predicting the class to token prediction during training. Class-based prediction leads to an implicit context aggregation for similar words and thus can improve generalization for rare words. Empirically, this curriculum learning strategy consistently reduces perplexity over various large, highly-performant state-of-the-art Transformer-based models on two datasets, **WikiText-103** and **arXiv**. Our analysis shows that the performance improvement is achieved without sacrificing performance on rare words.

Alignment-Aware Acoustic and Text Modeling is a novel pretraining method leveraging both the segmentation and alignment features for text-speech sequence modeling. It reconstructs masked acoustic signals with text input and acoustic-text alignment during training. In this way, the pretrained model can generate high quality of reconstructed spectrogram, which can be applied to the speech editing and new speaker TTS directly. Experiments show A^3T outperforms SOTA models on speech editing and improves multi-speaker speech synthesis without the external speaker verification model.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Ming Li, for his guidance, encouragement, and unwavering support throughout my PhD journey. His insightful feedback and constructive criticism have helped shape my research and academic growth. His patience, availability, and unwavering support have provided me with the confidence and motivation I needed to navigate the challenges of graduate school. I am also deeply grateful to my supervisor for helping me to develop the necessary research skills and cultivate the academic mindset that are essential for success in academia. In addition to the outstanding mentorship, I am grateful to my supervisor for his unwavering support and encouragement during difficult times. His belief in me and my abilities has been a constant source of inspiration and motivation.

I would like to express my sincere gratitude to my thesis committee, Jimmy Lin, Charles Clarke, Helen Chen, and Jimmy Huang, for their invaluable input and feedback on my research. Their diverse expertise and insightful commentary have helped me to refine my ideas and produce a more rigorous and comprehensive thesis. Their perspectives and suggestions have broadened my understanding of my research field and helped me to identify new avenues for exploration. Their constructive criticism and thorough evaluations have been instrumental in helping me to produce a high-quality thesis.

I am deeply grateful for the invaluable support and guidance provided by my industry advisors, including Navdeep Jaitly at Apple, Alessandro Sordoni and Tong Wang at Microsoft, Liang Huang, Renjie Zheng, and Mingbo Ma at Baidu Research, and Luchen Tan at RSVP.AI. Their mentorship during my research internships was truly exceptional, making them fruitful and enjoyable experiences. I feel fortunate to have had the opportunity to learn from and work alongside such talented and knowledgeable individuals, and their contributions have undoubtedly enriched the quality of my research.

I would like to thank my colleagues and friends in my program who have made my time as a PhD student both productive and enjoyable, including Yuqing Xie, Ruixue Zhang, Ajay Singh, Mojtaba Vàlipour, Christopher West, Gautam Pathak, Minghan Li, Haoye Lu, Yimu Wang, Shufan Zhang, and Jianlin Li. Their support, encouragement, and camaraderie have helped me to overcome the challenges of graduate school and to pursue my academic goals with renewed energy and enthusiasm. In addition to my graduate school, I am also grateful to my friends and loved ones who have provided me with a supportive and nurturing environment outside of the lab. Their companionship, kindness, and encouragement have provided me with the balance and perspective I needed to stay motivated and focused throughout my academic career.

I would like to express my deepest appreciation and love to my partner. The unwavering support, encouragement, and understanding have been an essential source of strength and motivation throughout my PhD journey, and I am grateful for the depth of our connection. Throughout the ups and downs of our lives, my partner has been a constant source of support and comfort, and we have shared in each other's joys and sorrows. Our love has been a source of strength and inspiration, and I am grateful for the mutual support and understanding that we share.

Lastly, I would like to express my heartfelt gratitude to my family for their unconditional love, unwavering support, and constant encouragement. Unlike many traditional Chinese families, my family has always encouraged me to pursue my dreams and to follow my heart. As the first generation in my family to attend university, I am grateful for their understanding and encouragement. I am proud to have been raised by such loving and supportive parents.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Language modeling (LM) is an unsupervised sequence modeling task that assigns probabilities to sequences of words [53], widely used in speech recognition, statistical machine translation, and optical character recognition. Over the past decades, LM has transitioned from n-gram model to neural models [12, 84, 80] and more recently to the Transformer architecture [129].

Today's Transformer-based language models pretrained with a massive amount of text data have shown great success in representation learning and transfer learning, and achieved the state-of-the-art results in various natural language processing tasks [93, 30, 106, 18, 23]. The pretraining refers to train a model with unsupervised task first – language modeling task in most cases – then apply the model to downstream tasks with or without downstream data finetuning.

Most recently, Large Language Models (LLMs) have shown great improvements for various NLP tasks in zero-shot and few-shot evaluation without any finetuning, i.e. GPT-3 [18], PaLM [23], Chinchilla [42], Gopher [102], CodeX [21], ChatGPT, etc. By pretraining the auto-regressive language model with a huge amount of data, LLMs can do both the classification tasks and generation tasks without any finetuning by transforming the downstream task examples into natural language sequence with prompting/templates.

Besides, the idea of text pretraining has also been applied in the speech processing domain for acoustic model pretraining, benefiting speech-related downstream tasks, such as speech recognition, speech classification, and speech translation [6, 20, 68, 150, 43]. In this case, the input is a sequence of speech features, such as spectrogram. The next/masked word prediction task is then replaced by the next/masked speech feature prediction. Considering there is no vocabulary for speech features, the prediction task is formulated as

a regression problem which is more challenging for model to generate high-quality speech features than a model to select a word from the vocabulary. But the similarity between the text sequence and the spectrogram sequence makes the transferring of the text pretraining method to speech pretraining possible.

1.1 Language Modeling

There are two common paradigms for pretrained LMs: causal language modeling (CLM) and masked language modeling (MLM), corresponding to GPT style [101, 18] and BERT style [30, 72], respectively. CLM is the traditional language modeling task that predicts the next word given the previous words in a sequence, while MLM is to predict masked words given the surrounding context. Both CLM and MLM are widely used in the pretraining of Transformer-based LMs.

1.1.1 Causal Language Model

The CLM models the the probability of the word sequence $x_1...x_n$ with the chain rule of probability:

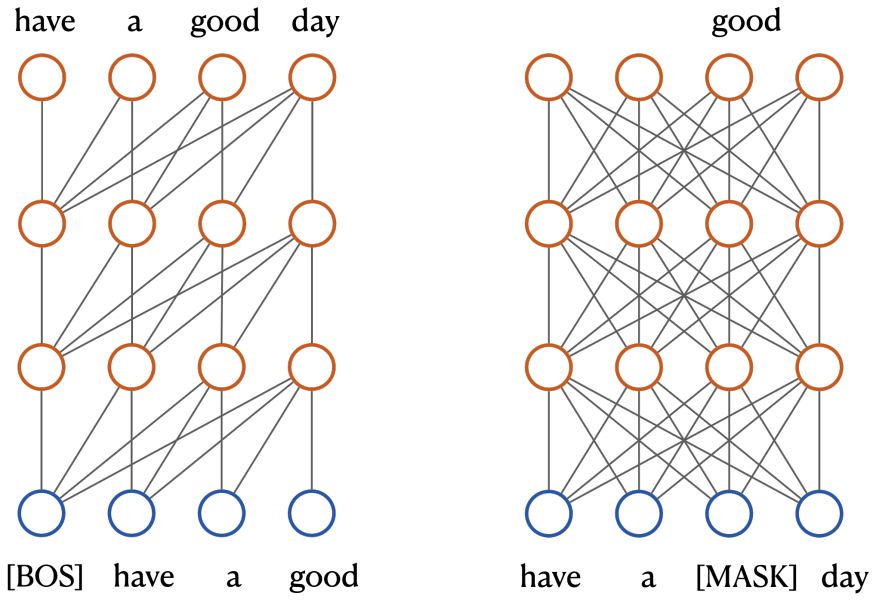
$$p(x_1...x_n) = \prod_{t=1}^{n} p(x_t|x_{1:t-1}) \tag{1.1}$$

where x_t is the t_{th} word and $x_{1:t-1}$ represents the context words before the word x_t. This equation shows CLM treats language modeling as a next word prediction task whose inputs are $x_{1:t-1}$.

The datasets for CLM training and evaluation are usually large and lengthy, for example, Wikipedia [79], webpages [106], scholarly papers [64], and books [105]. The evaluation metric of CLM is the perplexity (PPL). The test PPL of a CLM is the inverse probability of the test set, normalized by the number of words [53]:

$$\begin{aligned} \text{PPL}(x_1...x_n) &= p(x_1...x_n)^{-\frac{1}{n}} \\ &= (\prod_{t=1}^{n} \frac{1}{p(x_t|x_{1:t-1})})^{-\frac{1}{n}} \end{aligned} \tag{1.2}$$

Early CLM is based on n-gram model, which approximates the probability of a word given all the previous words by using only the conditional probability of the previous $n-1$



(a) CLM (b) MLM

Figure 1.1: Illustration of the CLM and MLM.

words. The assumption that the probability of a word depends only on the previous $n-1$ word is called a Markov assumption. It is not able to capture the long-term dependencies in the language. To address this issue, the recurrent neural network (RNN) is introduced to model the probability of a word given the previous words [148]. The RNN is a sequential neural network that can model the probability of a word given the previous words by using the hidden state of the previous words. However, RNN models suffer from the vanishing gradient problem [41], and the long-term dependencies are still not well captured.

More recently, Transformer [129] dominates the CLM domain, with more parallelization and less training times than RNN models [80]. The Transformer is a self-attention-based encoder-decoder architecture that can model the probability of a word by using the self-attention of the previous words. However, the vanilla Transformers are not able to model the probability of a word when the previous words are longer than the maximum sequence length. To capture the long-range context in language modeling, [28] proposes Transformer-XL by extending the vanilla Transformers with a memory segment, which can encode more context tokens to predict the next token and achieve significant improvements. [105] extends Transformer-XL with a compressed memory segment to further

encode long-time context memory. It should be noted that self-attention is computationally expensive, which limits the maximum sequence length that can be encoded. To encode even longer sequences, [11] proposes Longformer, which uses sliding window attention to encode the context tokens. Other works explore different sparse Transformers to encode much longer sequences for LM [114, 147]. ERNIE-Doc [31] proposes a document-level language pretraining model based on Recurrence Transformers, which has a much longer effective context length. There are also some works using memory or cache to retrieve the supporting evidence for the next token prediction task, which can reduce the perplexity significantly [54, 151].

Datasets of CLM are varies in size. Penn Treebank (PTB) [78, 82] is a small dataset with about 1M tokens. Compared to PTB, WikiText-2 [79] is over 2 times larger and WikiText-103 [79] is over 110 times larger. WikiText-103 is a popular word-level LM benchmark with long-term dependency and is widely used in the CLM domain. There are also some character-level datasets, such as enwik8 [77] and text8 [77], which are unprocessed and processed versions of Wikipedia text with about 100M characters. Although Wikipedia is the most popular domain of language modeling, some datasets are extracted from books [11], ArXiv [64], and common webpages [106].

1.1.2 Masked Language Model

Different from the CLM, MLM only predicts the probabilities of the masked tokens in the input sequence. An MLM example is shown in Fig. 1.1.

A key difference between CLM and MLM is that the MLM only predict the masked words, while the CLM predicts the next word given the previous words. Hence, the MLM needs masking schema to decide which tokens to mask for predicting. The masking scheme is usually random: the masked tokens are randomly selected from the input sequence, and the masked token is replaced with a special token, e.g., `[MASK]` in BERT [30]. In BERT's pretraining, the input sequence length is 512 and the masking rate is 15%, which means 15%*512 tokens of the input sequence are randomly selected and masked. The training task is to predict the masked tokens given the unmasked parts. Although many works following BERT to mask 15% tokens, [136] finds masking 40% tokens is better. There are also works that use more sophisticated masking schemes, such as the span masking [51, 106], which masks spans of tokens and the span lengths are randomly selected from a range.

Transformer encoder is the most popular architecture for MLM. BERT is the first work that uses Transformer encoder to pretrain MLM. BERT uses a bidirectional Transformer encoder to encode the input sequence, and the masked tokens are predicted by the last layer

of the encoder. There are also other variants of MLM, where an additional Transformer decoder is used to predict the masked tokens. For example, T5 [106] uses a bidirectional Transformer encoder and a unidirectional Transformer decoder to encode the input sequence and predict the masked tokens with a span masking schema. BART [66] also uses encoder-decoder model but trains the model with a denoising objective.

MLM is not a standard NLP task that has standard datasets and evaluation metrics, but an unsupervised pretraining task. The pretraining data is usually a large corpus of text, such as Wikipedia, BookCorpus [152], and common webpages. Pretrained MLM is usually finetuned before the evaluation. The finetuning is to train the pretrained model with the training data of the downstream task. The evaluation downstream tasks can be many NLP tasks, such as text classification [149, 76], question answering [109, 52, 61], and information extraction [123]. There are some challenging benchmarks to evaluate the performance of pretrained MLM, such as GLUE [131] and SuperGLUE [130] for natural language understanding evaluation, DiscoEval [45] for discourse evaluation, STS [19] for sentence representation evaluation, etc.

1.2 Natural Language Pretraining

Given the unsupervised nature of language modeling, pretraining a model with a large amount of text with either CLM or MLM becomes a common practice in NLP. Also, the idea of pretrained LM is not limited to text, but also has been applied to other modalities, such as image [100, 47], speech [6, 10, 43], and video [74]. Among these modalities, speech shares the most similarities with text, where many text-pretraining methods have been applied to the speech domain successfully.

1.2.1 Pretraining for Text

Large neural LMs trained with a massive amount of text have shown great success on many NLP tasks, benefiting from the dynamic contextual representations learned from language modeling and other self-supervised pretraining tasks.

Contextual Representations

Before the LM pretraining era, the word vectors such as GloVe [92] and word2vec [81] are widely used in NLP tasks. **ELMo** [94] first proposes to train a language model to get

the contextual word representations that outperform the static word vectors. ELMo uses a bidirectional LSTM [41] to encode the input sequence and concatenates the hidden states of the two directions as the contextual word representations. The usage of ELMo is to replace the static word vectors with the contextual word representations in the downstream task models. **GPT** [99] is similar to ELMo, but uses Transformer instead of LSTM. GPT uses the last layer of the Transformer encoder as the contextual word representation.

BERT-style

Instead of providing the contextual word representations for other NLP models, **BERT** [30] proposes the idea of finetuning that trains itself with the downstream task data. Also, different from ELMo and GPT trained with CLM task, BERT is trained with MLM task and an auxiliary task named next sentence prediction (NSP). The performance of BERT outperforms GPT and ELMo on many NLP tasks. MLM and finetuning become the most popular techniques in NLP at that time.

Considering there is no ablation study of BERT pretraining, **RoBERTa** [72] was proposed. RoBERTa is a robustly optimized BERT and finds: a large batch size and more training data are beneficial to BERT pretraining. Besides, NSP is not necessary for BERT pretraining. RoBERTa also finds that masking tokens of long documents is better than BERT's masking of sentence pairs. RoBERTa did not grow the model parameters but outperformed BERT significantly. **SpanBERT** [51] investigates different schemas of random masking and finds that the span masking is better than the token masking. **ALBERT** [63] proposes to share parameters across layers of BERT and replaced NSP with sentence order prediction (SOP). According to their experiments, SOP is more challenging than NSP. MLM and other downstream tasks can benefit more from replacing NSP with SOP. **ELECTRA** [25] proposes to train an additional discriminator to distinguish whether a token is the output of MLM or not. They show the proposed objectives are more efficient and perform comparably to RoBERTa with less than 1/4 compute. **XLNeT**[142] transforms BERT's MLM task into an autoregressive formulation and outperforms BERT on 20 tasks.

In addition to the vanilla encoder-based MLM, there are also some MLM variants with encoder-decoder architecture. **BART** [66] is a seq2seq model. The pretraining task is to generate the original text sequence given shuffled/masked input. The noises of the input include: randomly shuffling the order of the original sentences, and spans of text are replaced with a single mask token. BART works well for comprehension tasks (matches the performance of RoBERTa), but it is particularly effective when finetuned for text generation tasks. **T5** [106] is also a seq2seq model, but its objective is to generate the masked spans of the input sequence with its decoder. Compare to BERT, the training

data of T5 is about 37 times over and the largest T5 model is about 11B parameters. It achieves state-of-the-art results on many benchmarks covering summarization, question answering, text classification, and more. In addition to monolingual pretrained LMs, there are many multilingual LMs focusing on the multilingual and cross-lingual NLP tasks, such as **XLM** [26], **MBERT** [30], **MT5** [139], **MBART** [71], etc.

GPT style

Although GPT and ELMo are CLMs with the potential to generate text, their generation quality is limited by the small amount of training data and model parameters. **GPT-2** [101] is a larger CLM trained with much more data: 1.5 billion parameters and 8 million webpages data. Training such a big CLM with so much data is costly and the key speculation to support their exploration of such a large LM is that a CLM with sufficient capacity is equivalent to a multitask learner when the training data is large and diverse enough. They assume many tasks can be learned from the webpage text data, for example, translation, mathematical problem solving, coding, and so on. And the CLM can learn all these tasks by predicting the next word for each webpage. GPT-2 is the first work that shows LM can perform the downstream task in a zero-shot setting – without any parameter or architecture modification.

Following GPT-2, **GPT-3** [18] proposes to train a larger CLM and a stronger multitask learner. GPT-3 is a 175 billion parameter CLM trained with 300 billion tokens. Experiments show that GPT-3 can perform the downstream task in zero-shot and few-shot settings: given several demonstrations of a downstream task, GPT-3 can outperform the finetuned model. The k-shot evaluation refers to predicting the label given k examples as the context and there is no gradient and parameter updating, which is different from the traditional k-shot learning in machine learning. There are also many other large language models (LLMs) with an impressive performance on zero-shot and few-shot learning.

CodeX [21] is a GPT-3 variants finetuned on code for a total of 100 billion tokens. This work shows it is possible to train LLMs to produce functionally correct code bodies from natural language docstrings. **Gopher** [103] is a 280B model and evaluated with 150+ tasks, which outperforms GPT-3 175B model significantly. **Chinchilla** [42] is a 67B model trained with more data and more training steps, which achieves similar results as GPT-3 175B model. This work shows that LLMs are under-trained and can be further improved by training with more data and more training steps. **PaLM** [23] is a 540B model and evaluated with 200+ tasks. PaLM uses mix-of-expert [121, 33] to improve the performance of LLMs. By activating different expert parameters during the inference, PaLM gets better results without increasing the inference cost. BigScience group [117, 132] train an

open-sourced 176B model **BLOOM** [116] with multilingual dataset. The development of BLOOM involves over 1000 researchers from 70+ countries and 250+ institutions. Compared with GPT3, BLOOM emphasizes the importance of the multilingual pretraining data. Most recently, **ChatGPT**, a GPT-3 variant finetuned on human feedback with reinforcement learning, can generate impressive detailed and human-like text. However, it is unpublished without the details of the implementation.

1.2.2 Pretraining for Speech

Given the success of text LM pretraining, the speech community also investigates the pretraining methods for speech representation learning. However, different from the words of text, the audio waveform is continuous and without a fixed vocabulary for prediction. Also, speech is a long sequence without segment boundaries. To represent speech into the shorter sequences, spectral representation and time-domain localization are two common ways.

Time Domain

Contrastive Predictive Coding (**CPC**) [89] is a time-domain pretraining method for speech representation learning. By localizing the speech into short segments with a convolution network, CPC first encodes the windows of the waveform into speech representations in hidden space and then predicts the future in latent space by using the powerful autoregressive model and contrastive loss [39]. The **Wav2vec** [119] model extends the CPC method and replaces the RNN module with a CNN module for autoregressive encoding. Besides, the contrastive loss of wav2vec is the sum of N independent binary classification loss while CPC chooses a positive class from N classes. **Wav2vec 2.0** [6] encodes windows of the waveform into the speech representations in vector space, masks the speech input in vector space to simulate the MLM task, and trains the model by predicting the masked units via a contrastive task to simulate the text vocabulary. It is a combination of contrastive learning and masking.

In addition to learning from contrastive objectives, BERT-style pretraining has been applied to speech. **Discrete BERT** approach [5] uses the wav2vec model to extract speech representations and uses quantization to convert the continuous speech representations into the discrete tokens. Then, the discrete tokens are fed into the BERT-style model to learn the MLM objective. **HuBERT** [43] proposes to use a clustering method (k-means) to assign an MFCC cluster center to each frame to provide the frame-level targets. Once

pretrained, a second iteration of training is performed where the clustering is updated from the MFCC features to the HuBERT features. With the two iterations, HuBERT can match and outperform the performance of the previous state-of-the-art ASR system in low-resource settings. In addition to transforming the continuous audio into the discrete features for MLM training, [49] masks chunks of consecutive frames for waveform reconstruction. Similar to BERT, XLNet has been applied to speech (**Speech XLNet** [126]), to address the discrepancy between the pretraining and finetuning. There has also been growing interest in modeling text and speech jointly. For example, **SLAM** [10] unifies speech and text pretraining within a single model, by learning the MLM and wav2vec objectives with speech-text recognition data.

Frequency Domain

In addition to the waveform, features in the frequency domain (spectrogram) are commonly used to represent speech.

Mockingjay [69] and **MAM** [20] extend BERT's MLM pretraining to the speech frequency domain, masking consecutive spectrograms for speech pretraining, where the learning objective is to reconstruct the masked spectrograms. Similarly, **Audio ALBERT** [22] and **Speech T5** [3] extends ALBERT [63] and T5 [106] pretraining to speech frequency domain. [150] propose a Fused Acoustic and Text Masked Language Model (**FAT-MLM**) which jointly learns a unified representation for both the acoustic and text input from various types of corpora including parallel data for speech recognition and machine translation, and even pure speech and text data. These pretraining methods have improved many speech-related tasks, e.g., speech translation and speech recognition.

Although these models learn to reconstruct the spectrograms, the quality of their reconstructed spectrogram is far from the requirement of speech synthesis tasks. These pretrained models are all used in speech understanding tasks, where the quality of the reconstructed spectrogram is not very important.

1.3 Issues of Improving LM with More Data and More Parameters

Today's LM improvements usually benefit from more training data and more model parameters, especially for the large language model pretraining. However, such improvements are limited by the training cost and the availability of training data.

1) More model parameters and training data will increase the training cost. The training cost of pretraining should be addressed. For example, BLOOM [116], an open-sourced GPT-3 [18] model, costs more than 1 million GPU hours and 433K kwH of electricity [73] for pretraining. Since the development and research of large language models are still in their infancy, the training cost of large language models will be even higher. So it is essential to investigate methods to train a better language model without increasing the training cost.

2) On the other hand, the training data is smaller than English text for speech and text in other low-resource languages. There is a massive amount of English text to train the large language models. But in addition to English, there are many other languages with less training data. Only about 20 of 7000 languages have text corpora of hundreds of millions of words. Also, different from the text, the speech data is far less, especially for high-quality speech data. Given the similarity between text sequence and speech sequence, language modeling has inspired not only the text pretraining but also the speech pretraining, i.e., text BERT [30] and speech BERT [43], text T5 [106] and speech T5 [3], etc. However, the speech data is far less than the text data, which means some training methods could not be applied to the speech data. For example, text GPT [99, 101, 18, 8] show great power in text generation, but there is no similar speech model for speech generation. The size of high-quality speech data limits the development of speech-language modeling and pretraining.

Hence, it is essential to investigate “smart" methods to train a better language model without increasing the training data.

1.4 Thesis Overview

To address the challenges above, this thesis investigates how to improve the language model without increasing the training data, training cost, and the size of the model parameters. In this thesis, we find that the **language features** of natural language sequences can help LM achieve the above objectives via **novel language modeling techniques**, including novel model architectures and novel training methods.

The unique features of the natural language include the segment features and the linguistic features. The segment features refer to the segmentation of language. For the text data, the punctuations and paragraph breakers separate a long document into different text segments, which are the basic segmentation units of text. For the speech data, the phoneme is another segmentation unit. For the text and speech data, segmentation plays an essential role in human understanding of natural language but needs to be further explored in

the language modeling community. On the other hand, the linguistic feature refers to the linguistic information of language, for example, lexical category, syntactic structure, and semantic meaning. Previous works mainly focus on incorporating these linguistic features into the pretraining to benefit some downstream tasks, for example, corefBERT [144] for coreference resolution and senseBERT [65] for word sense disambiguation. These works show that linguistic features can help pretraining and improve linguistic-related tasks. However, whether these linguistic features can reduce the perplexity of a LM is under-explored.

In this thesis, the novel methods we present target leveraging the features of language – linguistic feature and segment feature – to help natural language model achieve lower perplexity and better generation. In detail, we present the **Segment-Aware Language Modeling**, which is a novel model architecture leveraging text segmentation feature for text sequence modeling; the **Hypernym-Instructed Language Modeling**, which is a novel training method leveraging lexical feature for rare words modeling; the **Alignment-Aware Acoustic and Text Modeling**, which is a novel pretraining method leveraging segmentation and alignment features for text-speech sequence modeling. All these features can be extracted in unsupervised ways, which is a desirable attribute to avoid the expensive feature extraction for a large training corpus.

Segment-Aware Language Modeling. In Chapter 2, we introduce a novel model architecture leveraging the text segmentation features for text sequence modeling. Although Transformer has dominated the language modeling world, the Transformer network was initially proposed in the sequence-to-sequence (seq2seq) architecture for machine translation, whose input is usually a sentence. However, the input of language modeling is usually a long document, with sentence and paragraph segments. Although Transformer can handle the long sequence, the internal segment information is implicit in the Transformer because of its position encoding method. The Transformer's position encoding simply assigns a unique index to each token, which means the token index in a sentence, sentence index in a paragraph, and paragraph index in a document are all implicit. In this thesis, we argue that the segmentation feature is important to language modeling, and propose a novel model architecture, Segment-Aware Transformer (Segatron), to explicitly model the segment information to model language better. We first introduce the segment-aware mechanism to Transformer-XL, which is a popular Transformer-based CLM with memory extension and relative position encoding. We find that our method can further improve the Transformer-XL base model and large model, achieving 17.1 perplexities on the **WikiText-103** dataset. We further investigate the pretraining MLM task with Segatron. Experimental results show that BERT pretrained with Segatron (SegaBERT) can outperform BERT with vanilla Transformer on various NLP tasks, and outperforms

RoBERTa on zero-shot sentence representation learning.

Hypernym-Instructed Language Modeling. In Chapter 3, we present a novel training method leveraging the lexical feature for rare word modeling. Linguistic knowledge is a unique feature of natural language, which can be used to help language modeling. But the performance of today's neural LMs is often improved at the cost of increased computational resources, instead of leveraging linguistic knowledge. For example, to capture long-term dependencies, various extensions of Transformer-based LMs have been proposed [28, 105]. These modifications bring about significant improvements in held-out perplexity, but training cost also increases significantly due to large GPU memory consumption and more computations at each training step. Many works show linguistic knowledge can help pretrained LM to achieve better performance on linguistic tasks, but little work shows that linguistic knowledge can help LM achieve lower perplexity. In parallel, alternative training strategies have also been proposed [40, 153, 29, 98], to achieve lower perplexity with the same computational resources. Among these works, curriculum learning (CL) is a promising one but under-explored in the context of the neural language model. The key idea of CL is to train a model with easy data first and hard data later. [13] first proposed CL, and examined its effectiveness with n-gram language model task. For the n-gram language model, the easy input is a span of $n-1$ high-frequency words, and the hard input is a span with $n-1$ low-frequency words. But for the neural language model, the input is much longer than $n-1$ words, and it is hard to distinguish what input is easy and what input is hard. We propose a new LM training strategy with WordNet's super-subordinate relation and curriculum learning. Mapping words to their hypernyms gives rise to a natural gradation of difficulty in the prediction task. Empirically, the proposed method consistently yields a 0.6-1.9% relative reduction in perplexity over baselines on the Wikipedia data and 1.3-3.1% on the scholarly paper data. Importantly, both rare and frequent tokens can be modeled better with our proposed method while other optimization methods may sacrifice the performance of rare tokens.

Alignment-Aware Acoustic and Text Modeling. In Chapter 4, we present a novel pretraining method leveraging bith the segment and alignment features for text-speech sequence modeling. Pretrained LM can generate text with a given prompt without any finetuning, which is a powerful tool for text generation. But for the speech pretraining, there is no pretrained model that can do speech synthesis without finetuning. The previous methods in the time domain, such as wav2vec 2.0 [6] and SLAM [10], are good at recognizing and extracting discrete information from speech and successfully improving automatic speech recognition, but they are unable to generate continuous acoustic signals for speech synthesis. Another line of pretraining work in the frequency domain, such as MAM [20], and FAT-MLM [150] show that reconstructing masked spectrogram with con-

tinuous units can improve the speech-to-text translation. However, the quality of their proposed speech reconstruction is far from the requirement of speech synthesis tasks. To address the problem that the pretrained speech model cannot generate high-quality speech, we extend our Segatron model to the speech-text pretraining. We propose our framework, Alignment-Aware Acoustic-Text pretraining (A^3T). In this case, our segment embeddings help the model to learn the alignment between the acoustic and phoneme input during the multi-modal pretraining, and significantly improve the quality of the reconstructed acoustic features. Our A^3T can generate speech without any finetuning. Experiments show A^3T outperforms SOTA models on speech editing and improves multi-speaker speech synthesis without the external speaker verification model.

1.5 Contributions

Overall, this thesis emphasizes the importance of language features for language modeling, which can improve language models without extra training data or training costs. The features discussed in this thesis include the segment features, lexical features, and alignment features. This thesis proposes different methods for leveraging these language features to improve language modeling in different settings.

The contributions of this thesis are summarized as follows:

This thesis contributes a novel model architecture leveraging text segmentation feature for text sequence modeling.

- The proposed model Segatron outperforms the vanilla Transformer for language modeling with lower perplexity;
- SegaBERT pretrained with Segatron outperforms BERT on various NLP tasks;
- Experimental results show the segment feature can improve both the LM and pretrained LM.

This thesis contributes a novel training method leveraging the lexical feature for rare words modeling.

- The proposed hypernym-instructed language modeling can reduce perplexities without increasing the training cost;

- This is the first work shows how the perplexity of large Transformer LMs can be improved by leveraging WordNet's hypernymy relation.
- Both rare and frequent words can be modeled better with the proposed method while other optimization methods may sacrifice the performance of rare words.

This thesis contributes a novel cross-modal pretraining method leveraging both the segment and alignment features for text-speech sequence modeling.

- This is the first pretraining method for speech synthesis that can generate high-quality speech without any finetuning;
- A^3T outperforms SOTA models on speech editing and improves multi-speaker speech synthesis without the external speaker verification model.

Chapter 2

Segment-Aware Language Modeling

2.1 Introduction

Language modeling (LM) is a traditional sequence modeling task which requires learning long-distance dependencies for next token prediction based on the previous context. Recently, large neural LMs trained on a massive amount of text data have shown great potential for representation learning and transfer learning, and also achieved state-of-the-art results in various natural language processing tasks.

To the best of our knowledge, state-of-the-art language models [28, 4, 105] and pre-trained language models [99, 30, 143, 63] all use a multi-layer Transformer [129]. The Transformer network was initially used in the sequence-to-sequence (seq2seq) architecture for machine translation, whose input is usually a sentence. Hence, it is intuitive to distinguish each token with its position index in the input sequence. However, the input length can grow to 1024 or more tokens and come from different sentences and paragraphs for language modeling. Although vanilla position encoding can help the Transformer be aware of the token position by assigning a unique index to each token, the token index in a sentence, sentence index in a paragraph, and paragraph index in a document are all implicit. Such segmentation information is essential for language modeling, as tokens in different segments of context hold different significance for next token prediction. If the Transformer model can be aware of the segment position of each token of the context, we argue that the Transformer model will model language more efficiently and successfully, and will generate better context representations. It should be noticed that, although punctuations and paragraph breakers can provide boundary information to some extent, the

boundary is not as straightforward as segment position, especially for the self-attention's dot-product operation in Transformer.

Hence, we argue that the segmentation feature of language is essential for language modeling, and we propose a novel segment-aware Transformer (Segatron), which encodes paragraph index in a document, sentence index in a paragraph, and token index in a sentence all together for the input sequence. We first verify the proposed method with relative position encoding on the language modeling task. By applying the segment-aware mechanism to Transformer-XL [28], our base model trained with the **WikiText-103** dataset [79] outperforms Transformer-XL base by 1.5 points in terms of perplexity. Our large model achieves a perplexity of 17.1, the same score as Compressive Transformer [105], which is a more complicated model with longer input context and additional training objectives. We also pre-train masked language models with Transformer (BERT-base$^-$) and Segatron (SegaBERT-base$^-$) with English Wikipedia for 500K training steps. According to experimental results, SegaBERT outperforms BERT on both general language understanding (GLUE) and machine reading comprehension tasks. We further pre-trained a large model SegaBERT-large with the same data used in BERT. Experimental results show that SegaBERT-large not only outperforms BERT-large on all the above tasks, but also outperforms RoBERTa-large on zero-shot Semantic Textual Similarity tasks, where we use less data and no more than 10% computational resources of RoBERTa. These results demonstrate the value of segment encodings in Transformers.

2.2 Related Work

Language modeling is a traditional natural language processing task which requires capturing long-distance dependencies for predicting the next token based on the context.

Most of the recent advances in language modeling are based on the Transformer [129] decoder architecture. [2] demonstrated that self-attention can perform very well on character-level language modeling. [4] proposed adaptive word input representations for the Transformer to assign more capacity to frequent words and reduce the capacity for less frequent words. [28] proposed Transformer-XL to equip the Transformer with relative position encoding and cached memory for longer context modeling. [105] extended the Transformer-XL memory segment to fine-grained compressed memory, which further increases the length of the context and obtains a perplexity of 17.1 on WikiText-103.

Although these works prove that longer context can be helpful for the language modeling task, how to get better context representations with richer positional information has not been investigated.

On the other hand, large neural LMs trained with a massive amount of text have shown great success on many NLP tasks, benefiting from the dynamic contextual representations learned from language modeling and other self-supervised pretraining tasks. GPT-2 [101] and BERT [30] are two representative models trained with the auto-regressive language modeling task and the masked language modeling task, respectively. In addition, BERT is also trained with an auxiliary task named next sentence prediction (NSP). ALBERT [63] then proposed to share parameters across layers of BERT and replaced NSP with sentence order prediction (SOP). According to their experiments, SOP is more challenging than NSP, and MLM together with other downstream tasks can benefit more from replacing NSP with SOP. Concurrently to ALBERT, [133] proposed two auxiliary objectives to provide additional structural information for BERT.

All these powerful pretrained models encode input tokens with token position encoding, which was first proposed by [129] to indicate the position index of the input tokens in the context of machine translation and constituency parsing. After that, Transformer has been extensively applied in machine translation and other sequence generation tasks [67, 70, 113]. However, the input length of language modeling tasks are much longer than these tasks, and simply assigning 0–512 token position embeddings is not enough for LMs to learn the linguistic relationships among these tokens. [8] show that incorporating segmentation information with paragraph separating tokens can improve the LM generator (GPT-2) in the context of story generation. However, compared with punctuation and paragraph breaker, segment position indexes are more straightforward for dot-product self-attention based Transformers. In this chapter, we try to encode segmentation information into the Transformer with the segment-aware position encoding approach.

2.3 Model

In this section, we show how to apply our proposed segment-aware Transformer to language modeling. More specifically, we first introduce our Segatron-XL (Segment-aware Transformer-XL) with non-learnable relative position encoding for auto-regressive language modeling. Then we introduce our pretrained Segatron (SegaBERT) with learnable absolute position encoding for masked language modeling (MLM).

2.3.1 Segatron-XL

We first introduce our method in the context of auto-regressive language modeling, by replacing the vanilla Transformer index in Transformer-XL [28] with Segatron. Transformer-

XL is a memory augmented Transformer with relative position encoding:

$$\begin{aligned}\mathbf{A}_{i,j}^{rel} &= \mathbf{E}_{x_i}^T \mathbf{W}_q^T \mathbf{W}_{k,E} \mathbf{E}_{x_j} + \mathbf{E}_{x_i}^T \mathbf{W}_q^T \mathbf{W}_{k,R} \mathbf{R}_{i-j} \\ &+ u^T \mathbf{W}_{k,E} \mathbf{E}_{x_j} + v^T \mathbf{W}_{k,R} \mathbf{R}_{i-j}\end{aligned} \tag{2.1}$$

where $\mathbf{A}_{i,j}^{rel}$ is the self-attention score between query i and key j. $\mathbf{E}_{x_i}$ and $\mathbf{E}_{x_j}$ are the input representations of query i and key j, respectively. $\mathbf{R}_{i-j}$ is the relative position embedding. $\mathbf{W}_{k,E}$ and $\mathbf{W}_{k,R}$ are transformation matrices for input representation and position embedding, respectively. $\mathbf{u}$ and $\mathbf{v}$ are learnable variables. The position embeddings are non-learnable and defined as:

$$\mathbf{R}_{i-j,k} = \begin{cases} sin(\frac{i-j}{10000^{2k/dim}}) & k < \frac{1}{2}dim \\ cos(\frac{i-j}{10000^{2k/dim}}) & k \geq \frac{1}{2}dim \end{cases} \tag{2.2}$$

where dim is the dimension size of $\mathbf{R_{i-j}}$, and k is the dimension index.

Our proposed method introduces paragraph and sentence segmentation to the relative position encoding. The new position embeddings $\mathbf{R_{I,J}}$ are defined as:

$$\mathbf{R}_{\mathbf{I},\mathbf{J},k} = \begin{cases} \mathbf{R^t}_{t_i - t_j, k} & k < \frac{1}{3}dim \\ \mathbf{R^s}_{s_i - s_j, k - \frac{1}{3}dim} & \frac{2}{3}dim > k \geq \frac{1}{3}dim \\ \mathbf{R^p}_{p_i - p_j, k - \frac{2}{3}dim} & k \geq \frac{2}{3}dim \end{cases} \tag{2.3}$$

where $\mathbf{I} = \{t_i, s_i, p_i\}$, $\mathbf{J} = \{t_j, s_j, p_j\}$. t, s, and p are token position index, sentence position index, and paragraph position index, respectively. $\mathbf{R^t}$, $\mathbf{R^s}$, and $\mathbf{R^p}$ are the relative position embeddings of token, sentence, and paragraph. These embeddings are defined in Eq. 2.2 and the dimensions of each are equal to 1/3 of $\mathbf{R_{I,J}}$. The input representation of our model is shown in Figure 2.1(a).

To equip the recurrence memory mechanism of Transformer-XL with the segment-aware relative position encoding, the paragraph position, the sentence position, and the token position indexes of the previous segment should also be cached together with the hidden states. Then, the relative position can be calculated by subtracting the cached position indexes from the current position indexes.

2.3.2 Pretrained Segatron

In this section, we introduce how to pretrain a language model with our proposed model Segatron in the context of BERT pretraining.

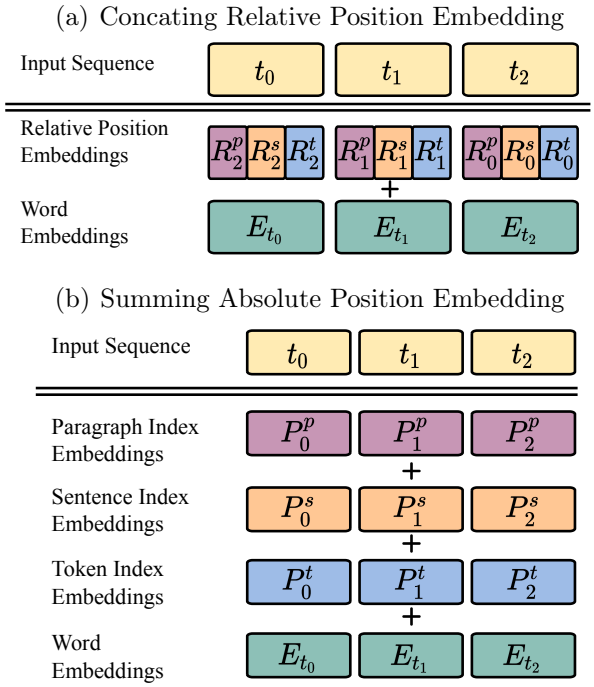


Figure 2.1: Input representation of Segatron-XL and SegaBERT.

First, pretraining a masked language model in the setting of BERT is a practical choice, as BERT is a popular baseline model and requires less computational resources compared with more recent large models. For example, BERT-large only needs about 10% of the resources of RoBERTa-large [72]. Hence, in this work, we first pretrain two base size models: SegaBERT-base$^-$ and BERT-base$^-$ with only English Wikipedia data for 500K training steps, to compare BERT pretrained with Transformer and Segatron fairly. We then pretrain a large size model SegaBERT-large with Wikibooks dataset and 1M training steps, same as BERT-large.

Input Representation. Input $\mathbf{X}$ of SegaBERT is a sequence of tokens, which can be one or more sentences or paragraphs. The representation x_t for token t is computed by summing the corresponding token embedding $\mathbf{E}_t$, token index embedding $\mathbf{P}^t_t$, sentence index embedding $\mathbf{P}^s_t$, and paragraph index embedding $\mathbf{P}^p_t$, as shown in Figure 2.1(b). Two special tokens `[CLS]` and `[SEP]` are added to the text sequence before the first token and after the last token, and their paragraph/sentence indexes are the same as their adjacent tokens. Following BERT, the text is tokenized into subwords with WordPiece and the maximum sequence length is 512.

Model	#Param.	PPL
LSTM+Neural cache [36]	-	40.8
Hebbian+Cache [104]	-	29.9
Transformer-XL base, M=150 [28]	151M	24.0
Transformer-XL base, M=150 (ours)	151M	24.4
Segatron-XL base, M=150	151M	**22.5**
Adaptive Input [4]	247M	18.7
Transformer-XL large, M=384 [28]	257M	18.3
Compressive Transformer, M=1024 [105]	257M	17.1
Segatron-XL large, M=384	257M	**17.1**

Table 2.1: Comparison with Transformer-XL and competitive baseline results on WikiText-103.

Training Objective. Following BERT, we use the masked LM as our training objective. However, next sentence prediction (NSP) is not used in our model, as our input contains more than two sentences.

Data preparation. For the pretraining corpus we use English Wikipedia and Bookcorpus [152]. For each document, we firstly split each into N_p paragraphs, and all the sub-tokens in the i-th paragraph are assigned the same Paragraph Index Embedding $\mathbf{P}_i^p$. The paragraph index starts from 0 for each document. Similarly, each paragraph is further segmented into N_s sentences with NLTK [15], and all the sub-tokens in the i-th sentence are assigned the same Sentence Index Embedding $\mathbf{P}_i^s$. The sentence index starts from 0 for each paragraph. Within each sentence, all the sub-tokens are indexed from 0; the i-th sub-token will have its Token Index Embedding $\mathbf{P}_i^t$.

When building a training example, we randomly (length weighted) sample a document from the corpus and randomly select a sentence in that document as the start sentence. Then, the following sentences are added to that example until the example meets the maximum length limitation (512) or runs out of the selected document. If any position index in that example exceeds the maximum index, all such position indexes will be subtracted by one until they meet the maximum requirements. The maximum position index of paragraph, sentence, and token are 50, 100, and 256, respectively.

Training Setup. [72] have shown that BERT pretrained with document input (more than two sentences) without NSP performs better than the original BERT on some tasks. Hence, we not only pretrain a SegaBERT-large, but also pretrain two base models with

the same setting for fair comparison. Similar to BERT, the base model is 12 layers, 768 hidden size, and 12 self-attention heads. The large model is 24 layers, 1024 hidden size, and 24 self-attention heads. For optimization, we use Adam with learning rate 1e-4, β_1=0.9, β_2=0.999, with learning rate warm-up over the first 1% of the total steps and with linear decay of the learning rate.

2.4 Experiments

In this section, we first conduct auto-regressive language modeling experiments with our proposed Segatron and also conduct an ablation study with this task. Then, we show the results of pretrained SegaBERT on general language understanding tasks, semantic textual similarity tasks, and machine reading comprehension tasks.

2.4.1 Autoregressive Language Modeling

Dataset

WikiText-103 is a large word-level dataset with long-distance dependencies for language modeling. This dataset preserves both punctuations and paragraph line breakers, which are essential for our segmentation pre-processing. There are 103M tokens, 28K articles for training. The average length is 3.6K tokens per article.

Model Configuration

Following Transformer-XL, we train a base size model and a large size model. The base model is a 16 layer Transformer with a hidden size of 410 and 10 self-attention heads. This model is trained for 200K steps with a batch size of 64. The large model is an 18 layer Transformer with a hidden size of 1024 and 16 attention heads. This model is trained with 350K steps with a batch size of 128. The sequence length and memory length during training and testing all equal 150 for the base model and 384 for the large model. The main differences between our implementation and Transformer-XL are: we use mixed-precision mode; our input/memory lengths between training and testing are the same; the large model training steps of Transformer-XL are 4M according to their implementation.

Figure 2.2: Valid perplexities during the training processes of language modeling.

Model	PPL
Transformer-XL base	24.35
+ paragraph position encoding	24.07
+ sentence position encoding	22.51
Segatron-XL base	22.47

Table 2.2: Ablation over the position encodings using Transformer-XL base architecture.

Main Results

Our results are shown in Table 2.1. As we can see from this table, the improvement with the segment-aware mechanism is quite impressive: the perplexity decreases 1.5 points for the Transformer-XL base and decreases 1.2 for Transformer-XL large. We also observe that our large model achieves 18.3 PPL with only 172K training steps. We finally obtain a perplexity of 17.1 with our large model – comparable to prior state-of-the-art results of Compressive Transformer [105], which is based on Transformer-XL but trained with longer input length and memory length (512) and a more complicated memory cache mechanism.

It is worth noting that we do not list methods with additional training data or dynamic evaluation [59] which continues training the model on the test set. We also note that there is a contemporaneous work RoutingTransformer [114], which modifies the self-attention to

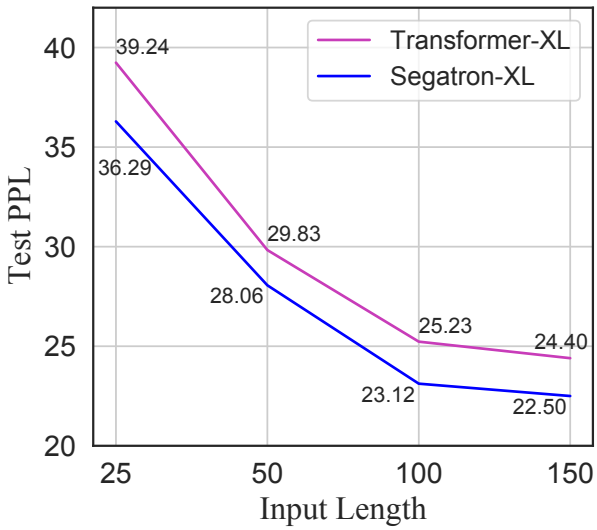


Figure 2.3: Test perplexities of Segatron-XL and Transformer-XL trained with different input lengths.

local and sparse attention with a clustering method. However, their implementations are not available. We believe our method is orthogonal to their work and can be introduced to their model.

Analysis

We plot the valid perplexity of Segatron-XL base and Transformer-XL base during training in Figure 2.2. From this figure, we can see that the segment-aware model outperforms the base model all the time, and the gap between them becomes larger as training progresses. Segatron-XL at 10K steps approximately matches the performance of Transformer-XL at 20K steps. We then test the effectiveness of Segatron over different input lengths (25, 50, 100, and 150 input tokens) by comparing Transformer-XL and Segatron-XL base models. As we can see from Figure 2.3, the improvements are consistent and significant. There is no evidence showing our method prefers shorter or longer input.

Ablation Study

We finally conduct an ablation study with Segatron-XL base, to investigate the contributions of the sentence position encoding and the paragraph position encoding, respectively.

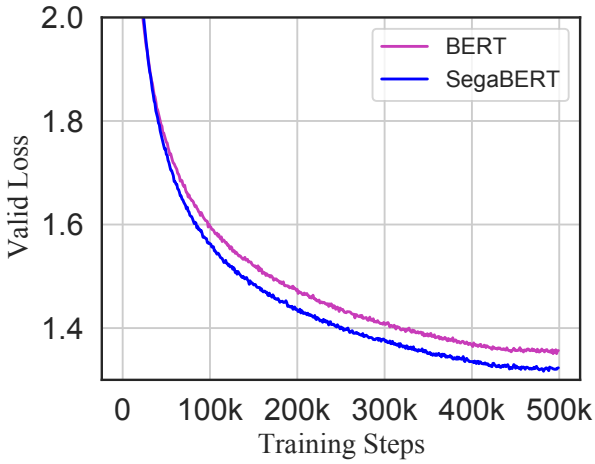


Figure 2.4: Valid losses during the pretraining.

Experimental results are shown in Table 2.2. From this table, we find that the PPL of Transformer-XL decreases from 24.35 to 24.07/22.51 after adding paragraph/sentence position encoding, and further decreases to 22.47 by encoding paragraph and sentence positions simultaneously. The results show that both the paragraph position and sentence position can help the Transformer to model language. Sentence position encoding contributes more than paragraph position encoding in our experiments.

2.4.2 Pretrained Masked Language Model

We first plot the valid losses of BERT-base$^-$ and SegaBERT-base$^-$ during pretraining in Figure 2.4. The overall trends between Figure 2.2 and Figure 2.4 are similar, which demonstrates that our proposed segment-aware method works on both auto-regressive language modeling and masked language modeling. We will detail our experiments with our pretrained models in the following sections.

General Language Understanding Evaluation

The General Language Understanding Evaluation (GLUE) benchmark [131] is a collection of resources for evaluating natural language understanding systems. Following [30], we evaluate our model over these tasks: linguistic acceptability CoLA [135], sentiment SST-2 [125], paraphrase MRPC [32], textual similarity STS-B [19], question paraphrase QQP,

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	AVG
BERT-base$^-$	83.2	90.4	86.5	68.3	91.3	**92.6**	55.0	88.9	82.0
SegaBERT-base$^-$	83.8	91.5	87.0	71.8	92.1	92.4	54.7	89.0	82.8
BERT-large (best of 3)	87.3	93.0	**91.4**	74.0	94.0	88.7	63.7	90.2	85.3
SegaBERT-large	**87.6**	**93.6**	89.1	**78.3**	**94.7**	92.3	**65.3**	**90.3**	**86.4**

Table 2.3: Fair comparison on GLUE dev.

textual entailment RTE [14] and MNLI [137], and question entailment QNLI [131]. We finetune every single task only on its in-domain data without two-stage transfer learning.

On the GLUE benchmark, we conduct the finetuning experiments in the following manner: For single-sentence classification tasks, such as sentiment classification (SST-2), the sentence will be assigned Paragraph Index 0 and Sentence Index 0. For sentence pair classification tasks, such as question-answer entailment (QNLI), the first sentence will be assigned Paragraph Index 0 and Sentence Index 0 and the second sentence will be assigned Paragraph Index 1 and Sentence Index 0.

We conduct grid search with the GLUE dev set for small data tasks: CoLA, MRPC, RTE, SST-2, and STS-B. Our grid search space is as follows:

- Batch size: 16, 24, 32;
- Learning rate: 2e-5, 3e-5, 5e-5;
- Number of epochs: 3-10.

For QQP, MNLI, and QNLI, we use the default hyper-parameters: 3e-5 learning rate, 256 batch size, and 3 epochs. The other hyper-parameters are the same as in the Hugging-Face Transformers library.[1]

We compare BERT and SegaBERT in a fair setting to decouple the effects of document-level inputs and the removal of NSP. In Table 2.3, two base models are pretrained by us and the only difference is the position encoding. The two base models are pretrained in the same setting. For large models comparison, we choose the best of 3 BERT-large models: the original BERT, whole word masking BERT, and BERT without NSP task. Results of BERT-large (best of 3) are from [143]. We can see that our SegaBERT-base$^-$ outperforms BERT-base$^-$ on most tasks. We also notice that SegaBERT-base$^-$ is lower

[1] https://github.com/huggingface/transformers

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	AVG
BERT-base$^-$	82.9	90.1	70.8	65.4	91.2	88.9	43.5	83.9	77.1
SegaBERT-base$^-$	83.5	90.8	71.4	68.1	91.5	89.3	50.7	84.6	78.7
BERT-large	86.7	92.7	72.1	70.1	**94.9**	89.3	60.5	86.5	81.6
SegaBERT-large	**87.9**	**94.0**	**72.5**	**71.6**	94.8	**89.7**	**62.6**	**88.6**	**82.7**

Table 2.4: Results on GLUE test set.

than BERT-base$^-$ by over 2.5 points on CoLA. However, this gap decreases to 0.1 on the test set, which is shown in Table 2.4. Results of BERT-large are from [30]. This is because the size of CoLA is quite small and not as robust as other datasets. Improvements can also be observed easily when comparing SegaBERT-large with the best score of 3 BERT-large models.

These results demonstrate SegaBERT's effectiveness in general natural language understanding. The improvements on these sentence and sentence pair classification tasks show that our segment-aware pretrained model is better than vanilla Transformer on sentence-level tasks.

Sentence Representation Learning Evaluation

Model	STS-12	STS-13	STS-14	STS-15	STS-16	STS-B	SICK-R	AVG
S-BERT-large	72.27	78.46	74.90	80.99	76.25	79.23	73.75	76.55
S-BERT-large*	72.39	78.06	**75.26**	81.79	76.35	78.64	73.85	76.62
S-RoBERTa-large	**74.53**	77.00	73.18	81.85	76.82	79.10	**74.29**	76.68
S-SegaBERT-large	74.49	**78.64**	74.88	**83.28**	**77.10**	**79.42**	73.77	**77.37**

Table 2.5: Zero-shot spearman's rank correlation $\rho \times 100$ between the negative distance of sentence embeddings and the gold labels.

Since our SegaBERT has shown great potential on sentence-level tasks, in this section, we further investigate whether SegaBERT can generate better sentence representations. Following Sentence-BERT [110], we finetune SegaBERT in a siamese structure on the combination of SNLI [16] and MNLI datasets. The finetuned model is named S-SegaBERT. We then evaluate the zero-shot performance of S-SegaBERT and other baselines on Semantic Textual Similarity (STS) tasks using the Spearman's rank correlation between the cosine similarity of the sentence embeddings and the gold labels.

System Model	SQUAD1.1		SQUAD2.0	
	EM	F1	EM	F1
BERT-base	80.8	88.5	72.3	75.6
BERT-base$^-$	81.9	89.4	75.4	78.2
SegaBERT-base$^-$	83.2	90.2	76.3	79.2
BERT-large	84.1	90.9	78.7	81.9
BERT-large wwm	86.7	92.8	80.6	83.4
SegaBERT-large	86.0	92.6	81.8	85.2

Table 2.6: Evaluation results on SQUAD v1.1 and v2.

In Table 2.5, the results of S-BERT-large and S-RoBERTa-large are from [110]. STS-B and SICK-R refers to STS benchmark and SICK relatedness dataset, respectively. Results of BERT-large and RoBERTa-large are from [110]. The results of S-BERT-large* are re-implemented by us, which is similar to Sentence-BERT's results. We can see that our SegaBERT achieves the highest average scores on STS tasks, even outperforms RoBERTa, which uses much more training data, larger batch size, and dynamic masking. These results conform with our improvements on GLUE benchmarks, which indicate that a language model pretrained with Segatron can learn better sentence representations (single sentence encoding) than the original Transformer.

Reading Comprehension Evaluation

We finally test our pretrained model on machine reading comprehension tasks. For these tasks, the question is assigned Paragraph Index 0 and Sentence Index 0. For a context with n paragraphs, Paragraph Index 1 to $n+1$ are assigned to them accordingly. Within each paragraph, the sentences are indexed from 0.

We first finetune our SegaBERT model with SQUAD v1.1 [109] for 4 epochs with 128 batch size and 3e-5 learning rate. The finetuning setting of SQUAD v2.0 [108] is the same as SQUAD v1.1. Results are shown in Table 2.6. Results of BERT-base and BERT-large are from [30]. Results of BERT-large wwm on SQUAD v1.1 are from BERT's github repository. There are no official results of BERT-large wwm on SQUAD v2 and here we report our finetuning results. As we can see from Table 2.6, our pretrained SegaBERT-base$^-$ outperforms our pretrained BERT-base$^-$ on both dataset: 1.3 EM and 0.8 F1 improvements on SQUAD v1.1; 0.9 EM and 1.0 F1 improvements on SQUAD v2. It should be noticed that our pretrained BERT-base$^-$ outperforms the original BERT-base model, although ours is

Model	Acc-Dev	Acc-Test
BERT-large	72.7	72.0
SegaBERT-large	74.5	73.8

Table 2.7: Accuracy on dev and test sets of RACE.

pretrained with fewer data and steps. This confirms [72]'s finding that BERT pretrained with document-level input can contribute to performance improvements on SQUAD. For large models, as we cannot afford to train a new BERT-large model in the same setting as BERT-base$^-$, we compare our model with BERT-large wwm (with whole word masking), which is a stronger baseline model. We can see that SegaBERT large is slightly lower than BERT-large wwm on SQUAD v1.1 but outperforms it on SQUAD v2 over 1.2 EM scores and 1.8 F1 scores.

We further test our models with RACE [62], which is a large-scale reading comprehension dataset with more than 28,000 passages. RACE has significantly longer contexts than SQUAD. Our results are shown in Table 2.7. Results of BERT-large are from [90]. The overall trend is similar to SQUAD.

Visualization

We further visualize the self-attention scores of BERT-base$^-$ and SegaBERT-base$^-$ in different layers. Figure 2.5 is an example document of BERT's input. Figure 2.6 shows the average attention scores across different attention heads. By comparing Figure 2.6(b) with Figure 2.6(a), we find that SegaBERT can capture context according to the segmentation, for example, tokens tend to attend more to tokens in its paragraph than tokens in the other paragraphs. A similar trend can be observed at the sentence level but is more prominent in the shallow layers On the other hand, the BERT model seems to pay more attention to its neighbors: the attention weights of the elements around the main diagonal are larger than other positions in Figure 2.6(a), and a band-like contour around the main diagonal can be observed in this figure.

From Figure 2.6(f) and Figure 2.6(e), we can see the attention structure in the final layer is different from the shallow layers, and SegaBERT pays more attention to its context than BERT. We also notice that a fractal-like structure can be observed in the first 10 layers of SegaBERT, while the last two layers of SegaBERT have a striped structure.

These attention behaviors show that: in the shallow layers, our model is segment-aware while BERT is neighborhood-aware; in the top layers, both of these two models focus on

Japanese destroyer Hatsukaze

The K̈agerō-̈class destroyers were outwardly almost identical to the preceding light cruiser-sized , with improvements made by Japanese naval architects to improve stability and to take advantage of Japan's lead in torpedo technology. They were designed to accompany the Japanese main striking force and in both day and night attacks against the United States Navy as it advanced across the Pacific Ocean, according to Japanese naval strategic projections. Despite being one of the most powerful classes of destroyers in the world at the time of their completion, only one survived the Pacific War.

Ḧatsukaze, built at the Kawasaki Shipbuilding Corporation, was laid down on 3 December 1937, launched on 24 January 1939 and commissioned on 15 February 1940.

At the time of the attack on Pearl Harbor, Ḧatsukaze was assigned to Destroyer Division 16 (Desdiv 16), and a member of Destroyer Squadron 2 (Desron 2) of the IJN 2nd Fleet, and had deployed from Palau, as part of the escort for the aircraft carrier in the invasion of the southern Philippines and minelayer .

In early 1942, Ḧatsukazeṗarticipated in the invasion of the Netherlands East Indies, escorting the invasion forces for Menado, Kendari and Ambon in January, and the invasion forces for Makassar, Timor and eastern Java in February. On 27-28 February, Ḧatsukazeänd Desron 2 participated in the Battle of the Java Sea, taking part in a torpedo attack on the Allied fleet. During the month of March, Desron 2 was engaged in anti-submarine operations in the Java Sea. At the end of the month, the squadron escorted the Christmas Island invasion force, then returned to Makassar. At the end of April, Ḧatsukazes̈ailed to Kure Naval Arsenal for maintenance, docking on 3 May.

On 21 May 1942, Ḧatsukazeänd Desron 2 steamed from Kure to Saipan, where they rendezvoused with a troop convoy and sailed toward Midway Island. Due to the defeat of the Carrier Striking Force and loss of four fleet carriers in the Battle of Midway, the invasion was called off and the convoy withdrew without seeing combat. Desdiv 16 was ordered back to Kure.

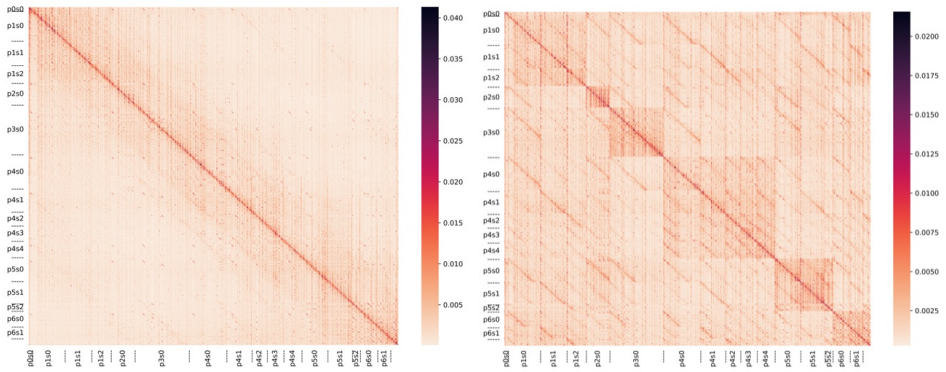
On 14 July, Ḧatsukazeänd Desdiv 16 were reassigned to Desron 10, Third Fleet. On 16 August, Desron 10 departed Kure, escorting a fleet towards Truk. On 24 August, Desron 10 escorted Admiral Nagumo's Striking Force in the Battle of the Eastern Solomons. During September and October, the squadron escorted the fleet patrolling out of Truk north of the Solomon Islands. On 26 October, in the Battle of the Santa Cruz Islands, the squadron escorted the Striking Force, then escorted the damaged carriers and into Truk on 28 October. On 4 November, Desron 10 escorted from Truk to Kure, then engaged in training in the Inland Sea, and then escorted Z̈uikakuf̈rom Truk to the Shortland Islands in January 1943.

On 10 January, while providing cover for a supply-drum transport run to Guadalcanal, Ḧatsukazeässisted in sinking the American PT boats P̈T-43änd P̈T-112.S̈he suffered heavy damage when struck by a torpedo (possibly launched by P̈T-112)ïn the port side; her best speed was 18 knots as she withdrew to Truk, for emergency repairs. Then she sailed to Kure in April for more extensive repairs. In September, Ḧatsukazeänd Desron 10 escorted the battleship from Kure to Truk. In late September and again in late October, Desron 10 escorted the main fleet from Truk to Eniwetok and back again, in response to American carrier airstrikes in the Central Pacific region. Between these two missions, Ḧatsukazes̈ortied briefly from Truk in early October 1943 to assist the fleet oiler Ḧazakaya,ẅhich had been torpedoed by an American submarine.

On 2 November 1943, while attacking an Allied task force off Bougainville in the Battle of Empress Augusta Bay, Ḧatsukazec̈ollided with the cruiser . The collision sheared off her bow, leaving her dead in the water. Ḧatsukazeänd the light cruiser were sunk (at position) by Allied destroyer gunfire. Of those on board, 164 were killed, including its commanding officer, Lieutenant Commander Buichi Ashida.

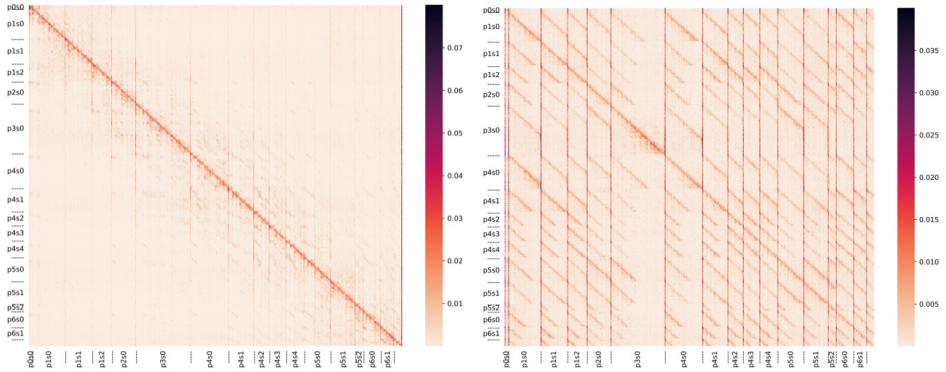
Ḧatsukazeẅas removed from the navy list on 5 January 1944."

Figure 2.5: An example article for visualization



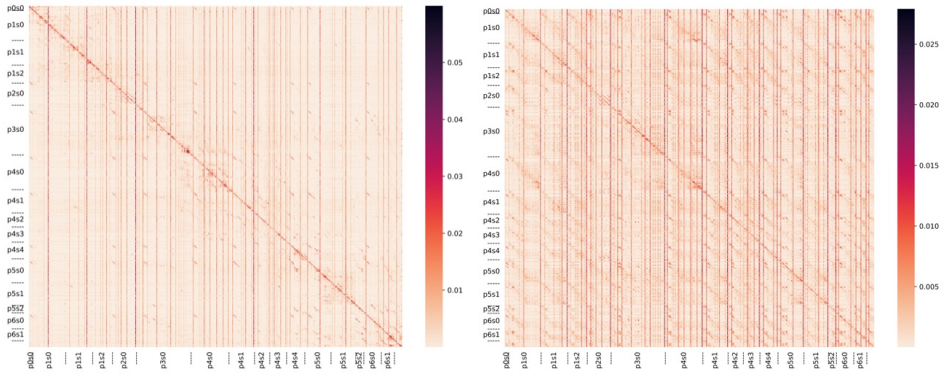
(a) BERT-Layer 1

(b) SegaBERT-Layer 1



(c) BERT-Layer 6

(d) SegaBERT-Layer 6



(e) BERT-Layer 12

(f) SegaBERT-Layer 12

Figure 2.6: Self-attention heat maps of the first, the sixth, and the last layer of SegaBERT and BERT when encoding the first 512 tokens of a Wikipedia article.

some tokens across the article rather than local neighbors, but our model can capture more contextual tokens.

2.5 Summary

In this chapter, we propose a novel segment-aware Transformer that can encode richer positional information for language modeling. The motivation behind the proposed approach is to leverage the segmentation feature of natural language for language modeling. By applying our approach to Transformer-XL, we train a new language model, Segatron-XL, that achieves 17.1 test perplexity on WikiText-103. Additionally, we pretrainin BERT with our SegaBERT approach and show that our model outperforms BERT on general language understanding, sentence representation learning, and machine reading comprehension tasks. Furthermore, our SegaBERT-large model outperforms RoBERTa-large on zero-shot STS tasks. These experimental results demonstrate that our proposed method works on both language models with relative position embeddings and pretrained language models with absolute position embeddings. The experimental results demonstrate the effectiveness of the segment feature and our proposed method.

Chapter 3

Hypernym-Instructed Language Modeling

3.1 Introduction

Over the course of the past decades, language modeling (LM) has transitioned from n-gram to neural models [12, 84, 30, 18]. Performance improvement of today's neural LMs is often achieved at the cost of increased computational resources. For example, to capture long-term dependencies, various extensions of Transformer-based LMs have been proposed [28, 105]. These modifications bring about significant improvements on held-out perplexity, but training cost also increases significantly due to large GPU memory consumption and more computations at each training step.

In parallel, alternative training strategies have also been proposed [40, 153, 29]. In this work, we explore the effectiveness of class-based language models (CLMs, [17]) in the context of neural LMs. CLMs group individual words into coarser-grained classes and has proven effective in alleviating context sparsity in n-gram LMs [27]. It has been also used to improve computational efficiency in neural LMs [85, 35]. More recently, [65] pretrain masked LMs [30] by predicting WordNet supersense labels. However, the work focuses on word-sense disambiguation tasks and doesn't provide clear evidence of gains in terms of perplexity. Although linguistic knowledge is a unique feature of language and should help language modeling, it is still unclear how to leverage linguistic features to reduce the perplexity.

In this chapter, we revisit CLM and assign words to classes by leveraging hypernym relations from the WordNet [83]. Our proposal, dubbed Hypernym Class Prediction (HCP)

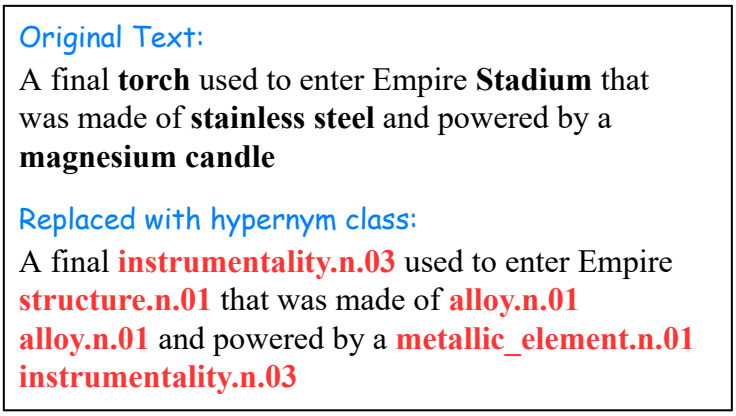


Figure 3.1: An example of word prediction training text and hypernym class prediction training text.

is simple and effective: for each batch, we substitute a subset of the tokens with their WordNet hypernyms (see Figure 3.1). Then, we train an auto-regressive LM on the resulting sentences using a mixed vocabulary composed of hypernyms and tokens. Crucially, we anneal the substitution rate during training, i.e., we gently switch from hypernym prediction to token prediction, following a curriculum learning approach. Note that this approach does not require WordNet information at inference time nor increases training time.

Our approach is motivated by two hypotheses. Firstly, mapping words to their hypernyms gives rise to a natural gradation of difficulty in the prediction task. Prior work has shown that LM benefits from training on instances of increasing difficulty [13, 98]. We thus postulate that, when coupled with the right curriculum, HCP can improve LM training and perplexity. Secondly, we hypothesize that HCP can improve rare word generalization through implicit context sharing. Neural models still struggle to learn reliable representations for rare words [118]. With CLM-based models, data sparsity for rare words can be abated, e.g., when the representation of their contexts are potentially drawn closer to those of their more frequent siblings by way of label (hypernym) sharing.

Empirically, the proposed method consistently yields about 0.6–1.9% relative reduction in perplexity over baselines on the **WikiText-103** dataset [79], and 1.3–3.1% on the **arXiv** dataset [64]. These improvements are observed with respect to memory-augmented [28] and segment-aware [7] LMs. Importantly, the proposed method improves performance for both rare and frequent words. We also observe that this is in contrast with performance improvements in regular LMs, which seem to be achieved at the cost of worsened performance on rare words.

To the best of our knowledge, this is the first work that shows how perplexity of Transformer LMs can be improved by leveraging hypernymy relationships. We provide an extensive ablation study highlighting crucial elements of HCP. Amongst those, we found particularly important to adopt a curriculum learning approach, rather than multi-objective learning or adaptive-softmax, and excluding frequent words from the hypernym prediction task. We highlight the simplicity and effectiveness of the proposed method as our main contribution, and hope this study would facilitate further exploration in this line of research.

3.2 Related Work

Transformer-based models are now popular language models. [28] propose Transformer-XL by extending the vanilla Transformer with a memory segment, which can encode more context tokens to predict the next token. [105] extend Transformer-XL with a compressed memory segment to further encode long-time context memory. Other works explore different sparse Transformers to encode much longer sequences for LM [11, 114]. Despite their effectiveness, neural models still struggle to learn reliable representations for rare words. Some approaches have been proposed to tackle this challenge by way of morphology [75], lexical similarity [56], context similarity [118, 55] and tokenization [60].

In addition to the model modifications, other work investigated curriculum learning to train LMs. [13] first find that curriculum learning could benefit LM training by training with high-frequency tokens first and low-frequency tokens later. [138] find that curricula works well when the training data is noisy or the training data is too large to iterate multiple epochs. [98] find that training Transformer-based LMs with short sequences first could improve convergence speed and perplexity.

Related work aimed at integrating WordNet information into pretrained language models. [65] propose SenseBERT by adding the word sense (WordNet supersense) prediction as an additional task during BERT [30] pretraining. SenseBERT outperforms BERT on both word supersense disambiguation [107] task and word in context [95] task. Recently, [96] use WordNet hypernymy chains as input to a RoBERTa [72] model to predict the plausibility of input events. In this work, our focus is to improve performance of auto-regressive LMs. We show that a multi-task strategy harms performance in this setting, and give a successful recipe to consistently boost LM performance with class-based predictions.

3.3 Method

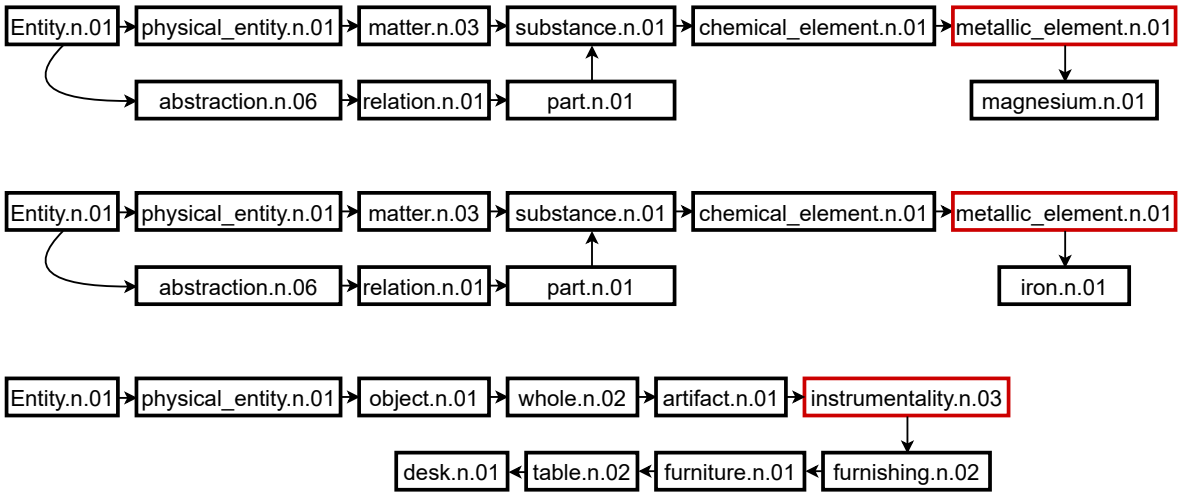


Figure 3.2: Hypernym-path example.

```
def token2class(token2freq, d, f):
    # token2freq is a dictionary whose key is the token and value is the tokens
        occurrences
    # d is the depth, f is the occurrence threshold
    rtn = {}
    for token, freq in token2freq.items():
        if freq > f:
            continue
        for synset in wordnet.synsets(token):
            for path in synset.hypernym_paths():
                if len(path)>=d and "noun" in path[d-1]:
                    rtn[token] = path[d-1]
                    break
            if token in rtn:
                break
    return rtn
```

Code 1: Pseudocode for token to class mapping.

Coupling class-based LM (CLM) and curriculum learning, HCP is to gradually anneal class prediction to token prediction during LM training. In this section, we first describe how we instantiate word classes by leveraging hypernym relation from the WordNet. We

then present how to incorporate the proposed Hypernym Class Prediction task into LM training via curriculum learning.

3.3.1 Hypernymy as Word Classes

WordNet [83] is a lexical database that groups words into sets of cognitive synonyms known as synsets, which are in turn organized into a directed graph by various lexical relations including the hypernymy (*is-a*) relation. In Figure 3.2, we show the hypernym-paths of synsets "magnesium.n.01", "iron.n.01", and "desk.n.01", corresponding to the word *magnesium*, *iron*, and *desk* respectively. As shown in Figure 3.2, each vertex is a synset, labeled by the text within the box, and each edge points from the hypernym (supertype) to the hyponym (subtype). Note that a word form (spelling) may be associated with multiple synsets – each corresponding to a different sense of the word, which are sorted by the frequency of the sense estimated from a sense-annotated corpus. For example, *iron* has 6 synsets, among which "iron.n.01" is the most common one.

Hence, if two words share the same hypernym at a certain level in their hypernym-paths (to the root in WordNet), we could say they are similar at that level. Here we use "Depth" to quantify the hypernym-path level. In Figure 3.2, for example, at Depth 6, *iron* and *magnesium* are mapped to the same group named "metallic_element.n.01", while *desk* is mapped to "instrumentality.n.03". At Depth 2, all these three words share the same (indirect) hypernym "physical_entity.n.01".

In this work, we map each token in our training set into its hypernym class if this token (1) has a noun synset in the WordNet, (2) with a hypernym-path longer than a given depth d, and (3) has frequency below a given threshold f in the training corpus. We only consider nouns because it is not only the most common class in the WordNet but also a difficult class for LMs to learn [64]. For tokens with multiple synsets, we iterate over the synsets in the order of sense frequency and break the loop once found. We select the most frequent synset no less than the required depth. The mapping pseudocode is illustrated in Code 1, which is a data pre-processing algorithm conducted only once before the training and takes no more than 5 minutes in our implementation.

3.3.2 Hypernym Class Prediction

We first partition the vocabulary into $\mathbf{V_x}$ and $\mathbf{V_{\neg x}}$ based on whether or not a token has a hypernym in the WordNet, and $\mathbf{V_h}$ denotes the set of all hypernyms. The original task

in a Transformer-based LM is then to predict the token w_j's probability with the output $\mathbf{x}$ from the last layer:

$$P(y = w_j|\mathbf{x}) = \frac{\exp(\mathbf{x}^\top \mathbf{v}_{w_j})}{\sum_{w_k \in \mathbf{V_x} \cup \mathbf{V}_{\neg \mathbf{x}}} \exp(\mathbf{x}^\top \mathbf{v}_{w_k})} \tag{3.1}$$

where w_k is the k_{th} word in the original vocabulary and $\mathbf{v}_{w_k}$ is its embedding. Here we assume the output layer weights are tied with the input embeddings. We call any training step predicted with Eq. 3.1 a token prediction step.

To do the Hypernym Class Prediction step, we replace all tokens in $\mathbf{V_x}$ in a batch of training data with their corresponding hypernym classes in $\mathbf{V_h}$. After the replacement, only hypernym classes in $\mathbf{V_h}$ and tokens in $\mathbf{V}_{\neg \mathbf{x}}$ can be found in that batch. Then, the LM probability prediction becomes:

$$P(y = w_j|\mathbf{x}) = \frac{\exp(\mathbf{x}^\top \mathbf{v}_{w_j})}{\sum_{w_k \in \mathbf{V_h} \cup \mathbf{V}_{\neg \mathbf{x}}} \exp(\mathbf{x}^\top \mathbf{v}_{w_k})} \tag{3.2}$$

where w_j could be either a token or a hypernym class. We called this batch step is a Hypernym Class Prediction (HCP) step.

Note that Eq. 3.2 is different from the multi-objective learning target, where the hypernym class would be predicted separately:

$$P(y = w_j|\mathbf{x}) = \frac{\exp(\mathbf{x}^\top \mathbf{v}_{w_j})}{\sum_{w_k \in \mathbf{V_h}} \exp(\mathbf{x}^\top \mathbf{v}_{w_k})} \tag{3.3}$$

where w_j is a hypernym class. We will elaborate on this difference in the experiment results part.

3.3.3 Training Method

We train a LM by switching from HCP to token prediction. For the example in Figure 3.2, our target is to teach a model to distinguish whether the next token belongs to the metallic element class or instrumentality class during the earlier stage in training, and to predict the exact word from magnesium, iron, and desk later.

Inspired by [13], we choose curriculum learning to achieve this. Curriculum learning usually defines a score function and a pacing function, where the score function maps from a training example to a difficulty score, while the pacing function determines the amount of the easiest/hardest examples that will be added into each epoch. We use a simple scoring function which treats HCP as an easier task than token prediction. Therefore, there is no

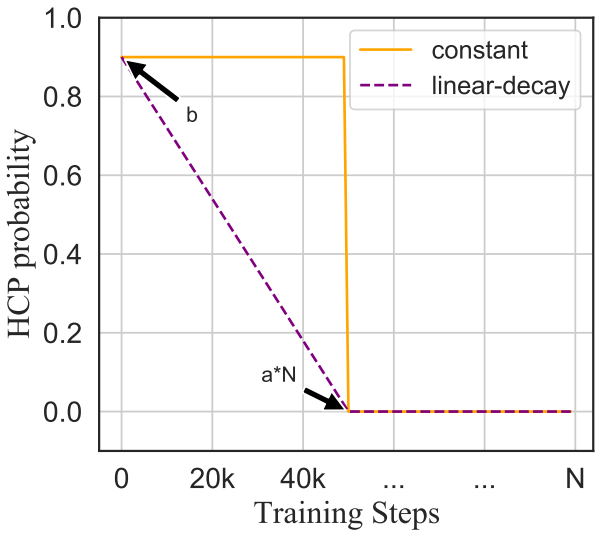


Figure 3.3: Probabilities of HCP step over training process with different pacing functions.

need to sort all training examples. The pacing function determines whether the current training step is a HCP step, i.e. whether tokens will be substituted with their hypernyms.

Our pacing function can be defined as:

$$P(y = c|t) = \begin{cases} b & t < a * N \\ 0 & t \geq a * N \end{cases} \tag{3.4}$$

or

$$P(y = c|t) = \begin{cases} b - b * \frac{t}{a*N} & t < a * N \\ 0 & t \geq a * N \end{cases} \tag{3.5}$$

where $P(y = c|t)$ is the probability that the current step t is a hypernym class prediction step. N is the total training steps. a and b are hyper-parameters. So, Eq. 3.4 is a constant pacing function in the first $a * N$ steps, while Eq. 3.5 is a linear decay function. We plot these two functions in Figure 3.3. According to our experimental results Tab. 3.5, these two functions are both effective in improving the language model.

Model	#Param.	Valid PPL	Test PPL
LSTM+Neural cache [36]	-	-	40.8
Transformer small	91M	34.5	36.5
+ HCP		34.1	**35.9**
Transformer base	151M	29.2	30.7
+ HCP		29.1	30.2
Transformer-XL base, M=150 [28]	151M	-	24.0
Segatron-XL base [7], M=150	151M	-	22.5
+ HCP		21.9	**22.1**
Transformer Large	257M	24.0	25.8 (80k steps)
+ HCP		23.7	25.3 (80k steps)
Adaptive Input [4]	247M	-	18.7 (286k steps)
Transformer-XL large, M=384 [28]	257M	-	18.3 (400k steps)
Compressive Transformer, M=1024 [105]	257M	16.0	17.1 (400k steps)
Segatron-XL large, M=384 [7]	257M	-	17.1 (350k steps)
+ HCP		16.1	**17.0** (350k steps)

Table 3.1: Results on **WikiText-103** dataset with different models.

3.4 Experiments

We conduct experiments on two datasets. **WikiText-103** [79] is a large word-level dataset with long-distance dependencies for language modeling. There are 103M tokens and 28K articles (3.6K tokens per article on average). The original vocabulary size is 271121, among which we find 3383 hypernym classes for 71567 tokens with $d = 6$ and $f = 6000$ (Section 3.3.1). **arXiv** [64] is collected from publicly available arXiv abstracts[1] with an average of 172 words per abstract and partitioned into training (1986–Sept 2017), evaluation (Aug–Dec 2017), and test (2018–2019). Following [64], we use the BPE [120] tokenization for this dataset. The final vocabulary size is 48935, which is about 1/6 of **WikiText-103**'s vocabulary, due to the different tokenization methods. In this case, we did not change the hypernym-depth d, but shrunk the frequency threshold from 6000 to 1000, as the tokenization method doesn't associate with the super-subordinate relationship but is associated with the word frequency. Finally, we find 1148 hypernym classes for 5969 tokens among the vocabulary with $d = 6$ and $f = 1000$.

[1] https://arxiv.org/help/oa/index

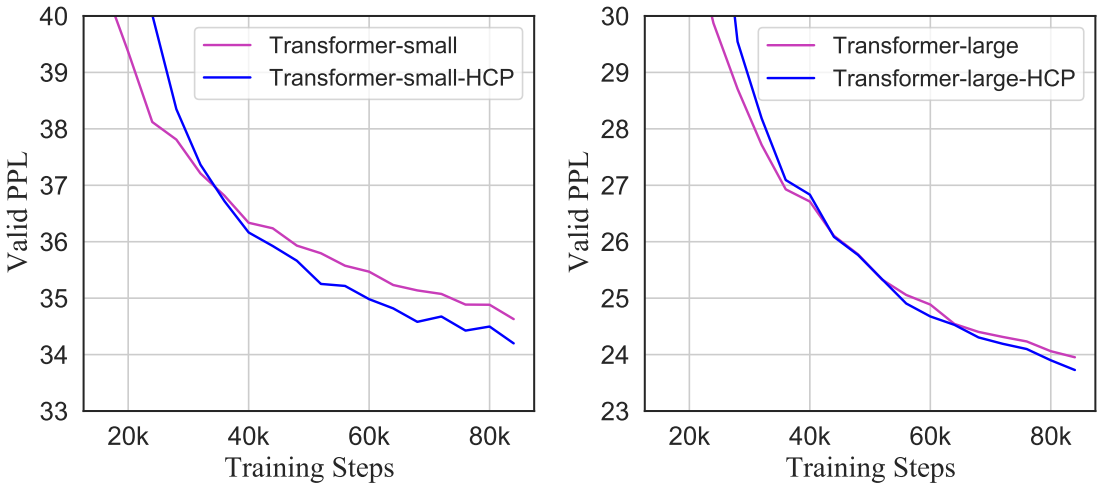


Figure 3.4: Valid perplexity curves during the training of small and large models.

Several variants of the Transformer model have been used for our experiments:

- small model: 12 layers, 10 heads, hidden size 300, batch size 256, training steps 100k;
- base model: 16 layers, 10 heads, hidden size 410, batch size 64, training steps 200k;
- large model: 18 layers, 16 heads, hidden size 1024 batch size 128.

The input lengths are 150 for the base model and 384 for the large model. The memory length is equal to the input length for both training and testing. The hyper-parameters used for the arXiv dataset are as same as the **WikiText-103**, except the arXiv base model's input length is 384. The number of training steps varies greatly for the large model in previous work, so we experiment on both the lower (80k) higher (350k) ends.

3.4.1 Main results

Our main results are shown in Table 3.1. We can see that all architectures could benefit from HCP: Transformer-small improved 0.6 ppl, Transformer-base improved 0.5, Segatron-XL base improved 0.4, Transformer-large improved 0.5, and Segatron-XL large improved 0.1. We also plot the validation perplexities of small and large models trained with and without HCP in Figure 3.4. In the beginning, the perplexity of the HCP models is higher due to the mixed training steps from the two tasks, but we can see that HCP perplexity goes

Model	#Param.	Valid PPL	Test PPL
Segatron-XL base	59M	22.39	24.21
+ HCP		21.79	**23.46**
Transformer-XL large [64]	287M	-	23.07
Segatron-XL large	283M	21.28	22.99 (80k steps)
+ HCP	283M	20.93	**22.60** (80k steps)

Table 3.2: Results on **arXiv** dataset with different models.

down faster than the baseline method. And after fully switching to token prediction, HCP outperforms the baseline method quickly and the gap between these two methods remains stable. These results suggest that HCP is indeed effective in improving LM training.

For experiments on the **arXiv** dataset, we first compare the Segatron-XL base model trained with and without HCP. The results are shown in Table 3.2. The improvements over the validation set and test set are 0.6 and 0.75 respectively. For the large model, we use the same model architecture and hyper-parameters as the **WikiText-103** large model but change the vocabulary to BPE sub-tokens. The final perplexity outperforms its counterparts about 0.4 and outperforms a larger model trained with 1024 input sequence length over 0.47, while our model length is 384.

3.4.2 Generalization on Rare Tokens

In addition to the overall perplexity comparison, we conduct comparisons with frequency-stratified validation subsets, to show the perplexity of tokens that has been replaced with the hypernym classes during training. Results are shown in Figure 3.5. We can see that, after the first 12k hypernym class prediction steps, there is a large gap between our HCP model and the baseline model as the HCP model only learn to predict the hypernym class instead of the token itself. After that, in the next 12k steps, HCP's PPL decreases faster, achieves similar PPL at 24k steps, and finally outperforms the baseline method in all frequency groups. The results show that our proposed training method can benefit the learning of the replaced tokens in various frequencies. Strikingly, we observe that, for the baseline, more training steps lead to a *degradation* of performance for rare tokens, a behavior that deserves investigation in future work. We further conduct pairwise model comparisons with tokens that have been replaced during HCP training on the **WikiText-103** test set. Given two models, we compare the prediction probabilities for each occurrence of a target token, and register a "win" for the model with a higher probability. We then

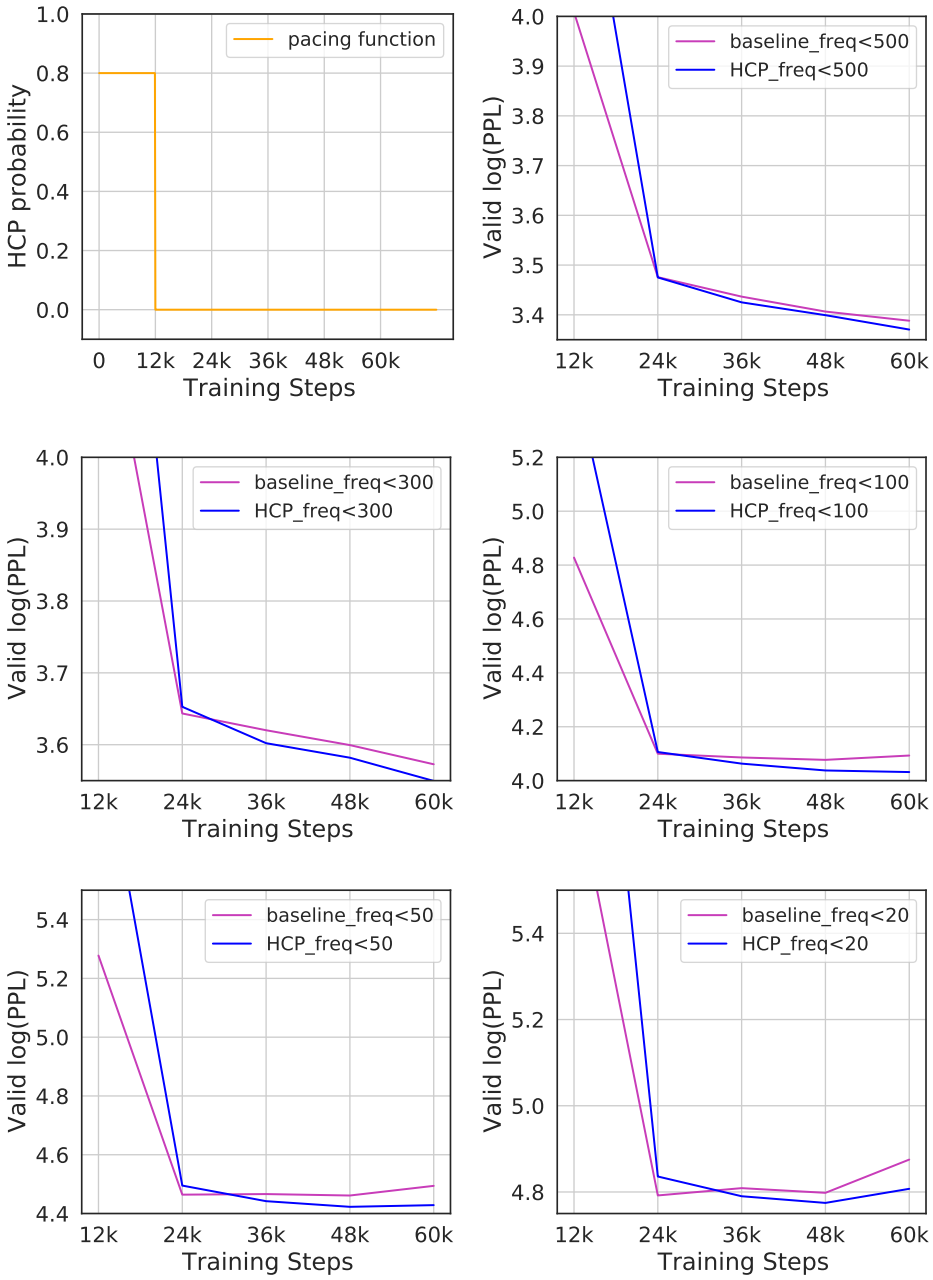
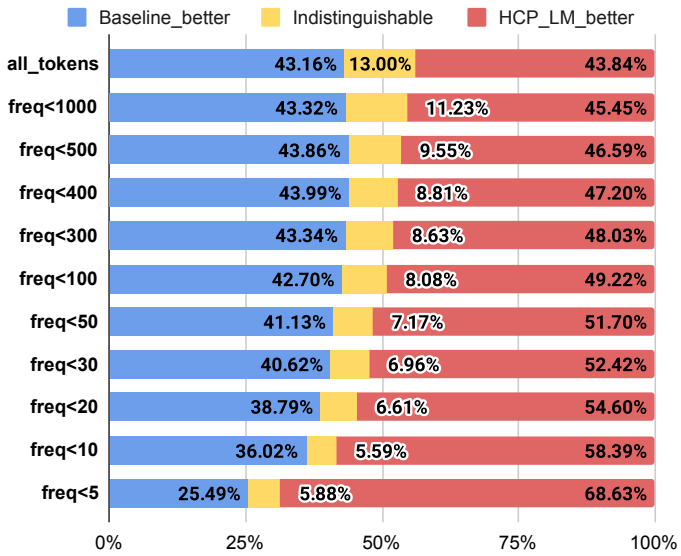
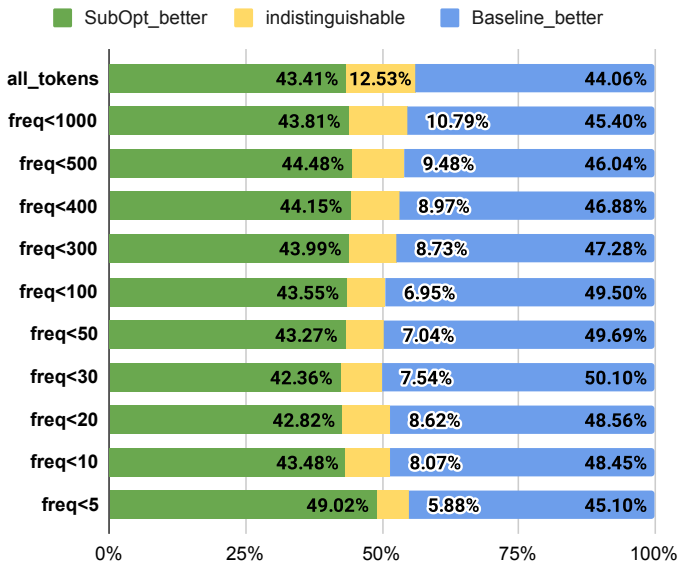


Figure 3.5: Frequency-stratified validation *log*(perplexity) of baseline model (Transformer-small) and HCP model (Transformer-small-HCP) with **WikiText-103**.



(a) Baseline model and HCP model



(b) Baseline model and sub-optimal model

Figure 3.6: Pairwise comparison results.

calculate the percentage of winnings (as well as ties) for each model by tallying over all occurrences of the token. The results are then stratified by token frequency and plotted in Figure 3.6. The better model is placed on the right in both sub-figures. In this figure, the baseline model and HCP model are trained without and with hypernym class prediction respectively. The sub-optimal model is trained without HCP and trained with different hyper-parameters, whose perplexity is increased by 0.9 compared with the baseline model.

From Figure 3.6(a), we see that HCP outperforms the baseline model on all frequency strata. Interestingly, *the performance gap widens as frequency decreases*, indicating that HCP is beneficial in modeling rare tokens. In Figure 3.6(b), we compare the baseline model against an under-optimized model of identical architecture but slightly different hyper-parameters.[2] Here, the (optimal) baseline outperforms the sub-optimal model on all but the least frequent stratum, suggesting the possibility that perplexity reduction (resulting from hyperparameter tuning in this case) might be achieved by improving frequent word prediction *at the expense of rare words*. This is inline with observations made recently in vision tasks [115].

3.4.3 Ablation study

We conduct ablation studies with **WikiText-103** dataset and Transformer small model to investigate how to map words to hypernym classes, how to select curriculum learning pacing functions and to show why we use curriculum training.

Hypernym-path Depth

The hypernym classes are chosen from the hypernym-paths in WordNet. Considering that a hypernym-path consists of multiple hypernyms, it is not straightforward to tell which layer is the best. But the best depth d should be some layer in the middle. Because a small depth might map multiple distant words into the same class, while a large depth will result in too many classes which are hard for a model to learn. The extreme examples could be $d = 1$ and $d = \infty$, corresponding to mapping all candidate words into the class "Entity.n.01" and mapping each word into itself respectively. In Table 3.3, we show evaluation results among different depth selections. The average depth is 8.03. #Classes denotes the total number of hypernym classes. We find that depth 6th is the best choice, with the lowest

[2]The sub-optimal model has batch size 128 instead of the optimal 64, and the perplexity gap between these two models is observed to be slightly larger than that between HCP and the baseline (0.9 vs 0.5).

Depth	Valid PPL	#Classes
Baseline	34.5	0
$d = 4$	34.54	145
$d = 5$	34.29	1169
$d = 6$	**34.05**	3383
$d = 7$	34.37	6604
$d = 8$	34.25	9063

Table 3.3: Clustering words into classes with different layer's hypernym parents.

valid perplexity. The results also confirm our assumption that the best one would be some middle layer.

Filter Frequency

In addition to the hypernym-path depth, we also investigate how to select frequency threshold f. As we mentioned above, our target is to map similar words into the same class, where predicting a hypernym class might be easier than predicting multiple different words. After the mapping process, low-frequency words can be clustered into hypernym classes with higher frequency. Table 3.4 shows the results of different f. #Rep. denotes the number of tokens in the vocabulary that will mapped. We can see that $f = 6000$ achieves the best results while $f = \infty$ (without filter) is the worst. We hypothesize this might be due to two reasons. First, for some high-frequency common words, the model can learn them well already, while mapping them into hypernym classes may be superfluous or even harmful. Second, including frequent words skews the marginal distribution over hypernym classes, causing hypernym prediction to be more class-imbalanced, which in turn might lead to collapsed representation in the resulting LM [34]. This hypothesis deserves further investigation. It should be noted that although the difference of #Rep.Tokens looks minor, the difference in the token's appearance is significant. For example, $f = \infty$ maps only 776 additional tokens compared with $f = 8000$, but each token's appearance is more than 8000, which explains the different perplexities in Table 3.4.

Pacing Function

Table 3.5 shows the results of models trained with various curriculum pacing functions. We also report the validation perplexities of the tokens that have ever been replaced with hyper-

FilterFreq.	Valid PPL	#Rep.
Baseline	34.5	0
$f = 3000$	34.14	70859
$f = 5000$	34.50	71735
$f = 6000$	**34.05**	71971
$f = 7000$	34.32	72153
$f = 8000$	34.35	72291
$f = \infty$	40.10	73067

Table 3.4: Ignoring words whose frequency more than a threshold f during hypernym class clustering.

nym class (Rep.PPL) during training and tokens without hypernym class (NonRep.PPL).

For the constant pacing function, we fix $b = 1$ and change the value of a, In this case, the models are always training with HCP in the first $a * 100k$ steps and then switch to the token prediction training, which is a pretraining pacing function. We can see that all models outperform the baseline model over the validation perplexity. Rep.PPL improves from 348 to 339. The perplexity of NonRep.PPL between baseline model and HCP models are similar, except the model trained with $a = 4$, which indicates the pretraining should not take up too many steps.

For the linear pacing function, we choose some specific a and b to achieve the same HCP steps as the constant functions above. For simplicity, we also set $a = b$. We show results with different pacing functions in Table 3.5, where NonRep.PPL denotes non-replaced tokens' perplexity, and Rep.PPL denotes replaced tokens' perplexity. In Table 3.5, we can see that the overall perplexity of the linear functions is similar to the corresponding constant functions, where the NonRep. PPL is slightly decreased while the Rep.PPL is slightly increased. We conduct a grid search over different pacing functions with Transformer small model and **WikiText-103**, and finally, use the constant function with $a = 0.12$ and $b = 0.8$ for all base models and large models.

Curriculum hyper-parameters could be transferred to the **arXiv** dataset successfully. However, we tune the frequency threshold f on each dataset, because different tokenization methods change the frequency distribution. All HCP models in Table 3.2 are using $d = 6$, $f = 1000$, and the constant pacing function with $a = 0.12$ and $b = 0.8$.

Constant Func.	HCP steps	Valid PPL	NonRep.PPL	Rep.PPL
a=0 b=0	0	34.5	22.07	348.87
a=0.1 b=1	10k	34.18	22.08	339.30
a=0.2 b=1	20k	**34.15**	22.07	339.34
a=0.3 b=1	30k	34.26	22.07	338.14
a=0.4 b=1	40k	34.39	22.26	338.31
Linear Func.				
a=0.45 b=0.45	10k	34.14	22.04	340.55
a=0.64 b=0.64	20k	**34.05**	21.96	341.33
a=0.78 b=0.78	30k	34.26	22.05	346.77
a=0.90 b=0.90	40k	34.56	22.12	354.40

Table 3.5: Training N steps hypernym class prediction among 100k training steps with different pacing functions.

Other Training Objectives

We also experimented with two other methods to incorporate hypernym information into LM training. Although neither method has yielded any empirical gain, we nonetheless report these methods and offer possible explanations for their failure.

Multi-objective Training Multi-objective (or multi-task) training consists in a weighted sum of token and hypernym prediction losses. We set the weight of the hypernym prediction loss to 0.2. The prediction of a token is calculated with Eq. 3.1. The prediction of a hypernym class is calculated with Eq. 3.3, where $\mathbf{x}$ can be the output vector from any layer in the Transformer LM. Table 3.6 lists the results using the last layer and the 8th layer. Using the last layer significantly undermines the original token prediction results. Using the 8th layer is better but the final perplexity is still no better than the baseline model. Simply forcing the language model to predict the hypernym class for each token is harmful to LM performance. We also tried to replace Eq. 3.3 with Eq. 3.2, by mixing $\mathbf{V_h}$ and $\mathbf{V_{\neg w}}$ together when predicting the hypernym classes (mix vocab). This significantly improves multi-objective training. Learning to predict the hypernym class from a mixed vocabulary $\mathbf{V_h} \cup \mathbf{V_{\neg w}}$ is better than only hypernym classes $\mathbf{V_h}$.

	Valid PPL	Test PPL	NonRep.PPL	Rep.PPL
Baseline	34.50	36.46	22.07	348.87
Adaptive Softmax	36.32	38.16	22.48	435.93
Multi-obj				
last layer	46.06	48.49	27.81	627.23
8th layer	43.42	45.37	26.13	597.66
8th layer + mix vocab	35.97	38.02	22.98	365.27
Hypernym Class Prediction	**34.05**	**35.87**	**21.96**	**341.33**

Table 3.6: Results obtained by alternative strategies.

Adaptive Softmax Another method is the adaptive-softmax [35], where the model first predict the hypernym probability among $\mathbf{V_h} \cup \mathbf{V}_{\neg \mathbf{w}}$ and then predict the token probability among the tokens with the same hypernym class. In Table 3.6, we can see that the adaptive-softmax is no better than the multi-objective trained model. By looking into the poor perplexity of Rep.PPL, we find this method cannot improve the prediction of tokens in $\mathbf{V_w}$. We believe this is due to the noise of hypernym class mapping, where we choose the first synset path as the token's hypernym synset without considering the context. Such noise will affect the adaptive-softmax prediction but is not an issue for curriculum training as the final training stage is fully trained with the original text.

3.5 Summary

In this chapter, we propose a novel training method which leverages lexical feature for language modeling. The proposed training strategy is based on curriculum learning and WordNet's super-subordinate relation. Although WordNet is an external resources, it's not clear how to get lower perplexity using WordNet before this work. This is the first work shows how the perplexity of large Transformer LMs can be improved by using WordNet. Also, consistent perplexity reduction can be observed over various models and datasets, where the improvements are obtained without increasing the training cost. Finally, both the rare and frequent tokens can be modeling better with our proposed method while other optimization method may sacrifice the performance on rare tokens.

We'd like to address the limitations of this work: in addition to the super-subordinate relation and lexical feature, there may be other more effective methods to map words to classes by using other linguistic knowledge; our experiments are only conducted with

English, and experiments with other languages should be verified in the future; it is not sure whether the proposed method can help LM pretraining and downstream tasks. We hope to investigate these directions in the future.

Chapter 4

Alignment-Aware Acoustic and Text Modeling

4.1 Introduction

Self-supervised pretraining methods have shown great power for NLP tasks, and has also been applied to the speech processing domain for speech representation learning. It has attracted much attention in the speech community due to its strong performance to many speech-related downstream tasks, such as speech recognition, speech classification, and speech translation [6, 20, 68, 150, 43]

However, all these efforts can only support *speech understanding* tasks which take speech as input, but for the inverse direction, *speech synthesis*, which synthesis speech as output, the potential of representation learning is yet to be realized. For example, one line of work, such as wav2vec 2.0 [6], HuBERT [43] and SLAM [10], learn discrete quantized speech units as latent representations. In this way, these models are good at recognizing and extracting discrete information from speech and successfully improves automatic speech recognition (ASR), but they are unable to generate continuous acoustic signals for speech synthesis. On the other hand, another line of work, such as MAM [20] and FAT-MLM [150], show that reconstructing masked spectrogram with continuous units can improve speech-to-text translation. However, the quality of their proposed speech reconstruction is far from the requirement of speech synthesis tasks (see Fig. 4.6(f)).

To address this problem, we propose our framework, Alignment-Aware Acoustic-Text Pretraining (A^3T), where we introduce cross-modal alignment embeddings which make

Model	Data		Reconstructed Masked Speech	Tasks	
	$\langle \mathbf{s}, \mathbf{x} \rangle$	$\langle \mathbf{s} \rangle$		*Speech Editing*	*Text-to-Speech*
wav2vec 2.0		✓	discrete units		
HuBERT		✓			
SLAM	✓	✓			
MAM		✓	low-quality spectrogram		
FAT-MLM	✓	✓			
A^{3}T	✓	✓	high-quality spectrogram	✓	✓

Table 4.1: Comparisons of A^{3}T with other existing speech pretraining models. Here $\mathbf{s}$ stands for speech input, while $\mathbf{x}$ stands for text, and $\langle \mathbf{s}, \mathbf{x} \rangle$ denotes parallel speech-text data.

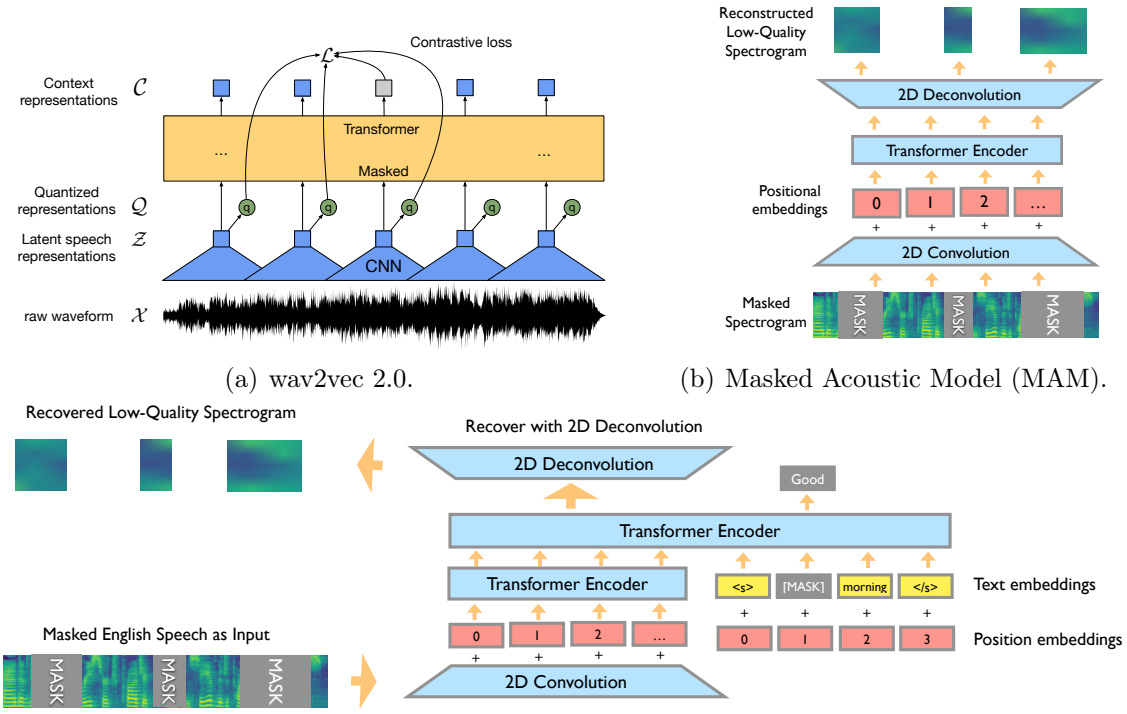
the model easier to learn the alignment between the acoustic and phoneme input during multi-modal pretraining, and significantly improve the quality of the reconstructed acoustic signals. By leveraging the phoneme segmentation feature and speech-text alignment feature, the proposed A^{3}T model is improved significantly without increasing any training cost. Moreover, we borrow several useful ideas from recent text-to-speech (TTS) literature, including Conformer [37, 38] and Post-Net [122], to further improve the quality of our reconstructed spectrograms.

The proposed model can be adopted as a speech-editing system, a task that modifies an existing speech, by reconstructing the desired acoustic signals given original contextual speech and modified text. Furthermore, the model can be adopted as a multi-speaker TTS system with our proposed prompt-based decoding method, to synthesis unseen speaker's speech without the external speaker verification model (speaker embeddings). Our experiments show that our A^{3}T with prompt-based decoding can outperform the TTS model equipped with both the speaker embedding [48] and the global style token (GST) [134].

We make the following contributions[1]:

- We propose the Alignment-Aware Acoustic-Text Pretraining (A^{3}T), which can reconstruct masked spectrograms with high quality.

[1]See our Demo at:
https://educated-toothpaste-462.notion.site/Demo-fdacf73d17904fad8901548504aece9d



(a) wav2vec 2.0.

(b) Masked Acoustic Model (MAM).

(c) FAT-MLM.

Figure 4.1: Previous work for speech representation learning.

- We show that the proposed A^{3}T model has the ability to do speech editing and outperforms the current SOTA.
- We propose the prompt-based decoding method. We show that our A^{3}T model has the ability to do speech synthesis for unseen speaker and outperforms the speaker-embedding-based multi-speaker TTS system.

4.2 Related Work

4.2.1 Speech Synthesis and Editing

Recently, neural TTS systems become capable of generating audios with high naturalness [128, 122, 112, 91, 111, 57]. SOTA neural TTS systems generally consist of two

stages: the *text-to-spectrogram* stage which generates an intermediate acoustic representation (linear- or mel-spectrogram) from the text, and the *spectrogram-to-wave* stage (vocoder) which converts the aforementioned acoustic representation into actual wave signals [88, 97].[2]

In the multi-speaker and unseen-speaker settings, the existing TTS models need to be trained with an additional input feature: speaker embedding [48], which is extracted from an external speaker verification model trained with tens of thousands of speakers' audio. And during the inference for an unseen speaker, the embedding will be extracted from one of this speaker's other audio examples. However, the embedding from the speaker verification model is not optimized directly to capture speaker characteristics relevant to synthesis, and cannot provide enough information for the TTS model to generate audio similar to the example.

The input of speech editing includes the original speech, the original and modified text. [50] propose to insert a regenerated audio clip back into the original recording. However, due to the absence of speech contextual information, the boundaries of the modified region would be not smooth. [86] propose to retrieve the modified speech segments from other utterances of the same speaker and correct the prosody with a context-aware TD-PSOLA corrector [87]. However, the edited content may not be found in the speech data of the same speaker. Most recently, [127] use neural TTS model to generate better-modified speech. This method is only compatible with auto-regressive decoding models and highly relies on the speaker embeddings, which limits its efficiency and transferability to new speakers.

4.2.2 Speech Pretraining

To improve the Text-to-Speech model from larger-scale pure speech data, one idea is to do pretraining on speech data. All existing speech pretraining work learn either discrete units, which can only support speech understanding tasks, or spectorgram, but with very low quality.

Reconstructing Discrete Units

Wav2vec 2.0 proposed by [6] is the most popular speech pretrain model recently. It masks the speech input in the latent space and pretrains the model by predicting discrete units via a contrastive task defined over a quantization of the latent representations, as shown in Fig. 4.1(a). Similar to wav2vec 2.0, HuBERT [43] and SLAM [10] also learn discrete

[2]We focus on the *text-to-spectrogram* stage and use an off-the-shelf vocoder Parallel WaveGAN [141].

speech units from contextualized representations to represent the latent representations. Thus these models can achieve good performance in speech recognition tasks, but they are unable to generate continuous acoustic signals for speech synthesis.

Reconstructing Low-Quality Spectrogram

Recently, [20] propose to learn a speech encoder in a self-supervised fashion on the speech side, which can utilize speech data without transcription. Fig. 4.1(b) demonstrate the architecture of this model, termed Masked Acoustic Modeling (MAM). MAM replaces a span of speech spectrogram with mask tokens, and learns to recover the masked spectrogram during training. On the other hand, [150] propose a Fused Acoustic and Text Masked Language Model (FAT-MLM) which jointly learns a unified representation for both acoustic and text input from various types of corpora including parallel data for speech recognition and machine translation, and even pure speech and text data, as shown in Fig. 4.1(c).

Both MAM and FAT-MLM reconstruct spectrograms, however, the quality of their spectrogram output is far from the requirement of speech synthesis tasks (see Fig. 4.6(f)), since these pretrained models are all used in speech understanding task (speech-to-text translation), where the quality of the reconstructed spectrogram is not very important.

4.3 Model

Although existing speech pretraining models show a strong representation learning ability and significantly improve upon many down-stream tasks in *speech understanding*, all these efforts can not support *speech synthesis* tasks. To address this problem, we propose the Alignment-Aware Acoustic-Text Pretraining (A^3T) which learns to generate high-quality spectrogram given speech context and text.

4.3.1 A^3T

A^3T takes speech and transcription tuples as input, denotes as $D_{\mathbf{s},\mathbf{x}} = \{\langle \mathbf{s}, \mathbf{x} \rangle^{(n)}\}_{n=1}^{|D|}$, where $\mathbf{s} = (s_1, ..., s_{|s|})$ is a sequence of acoustic features $s_i \in \mathbb{R}^{d_s}$ which can be the spectrogram or mel-spectrogram of the speech audio, and each s_i represents the frame-level speech feature, and $\mathbf{x} = (x_1, ..., x_{|\mathbf{x}|})$ is the sequence of corresponding transcription.

As shown in Fig. 4.2, we first randomly mask several spans of $\mathbf{s}$ by a random masking function over the input $\mathbf{s}$: $\hat{\mathbf{s}} \sim \text{Mask}_{\text{span}}(\mathbf{s}, \lambda)$, where $\text{Mask}_{\text{span}}(\cdot)$ replaces several random

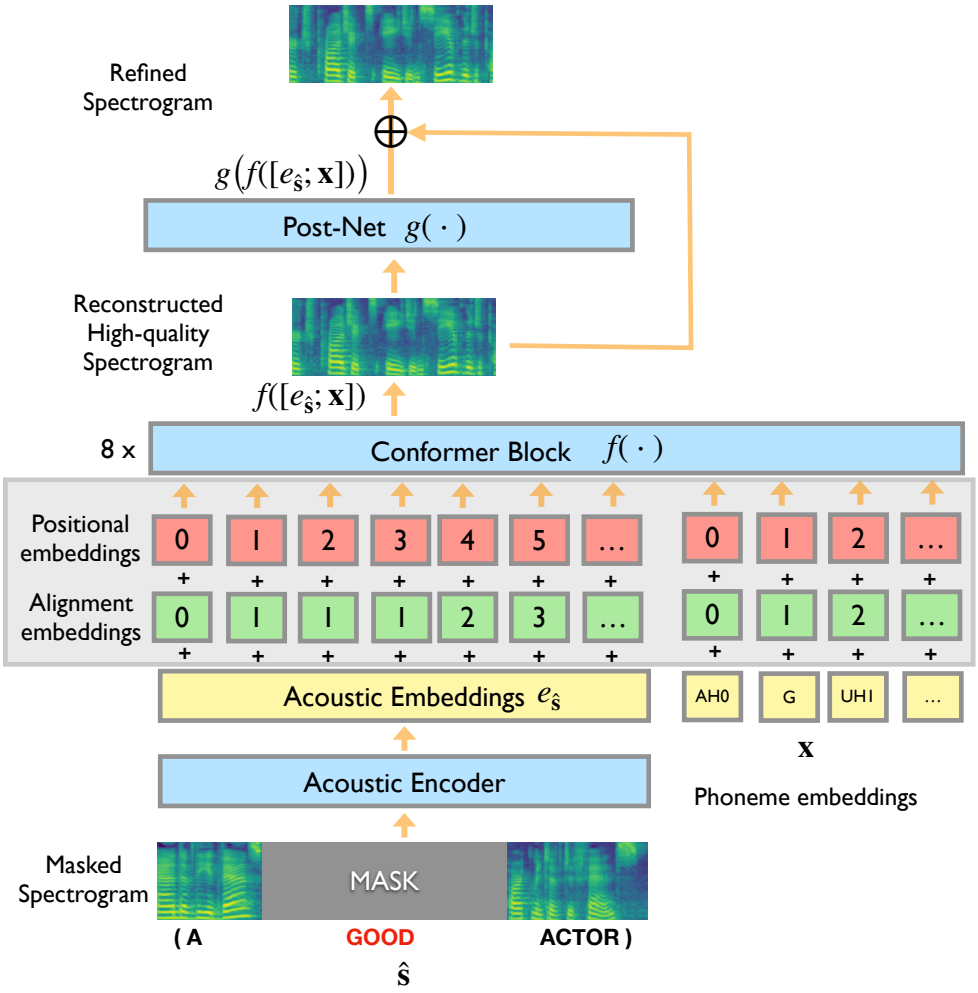
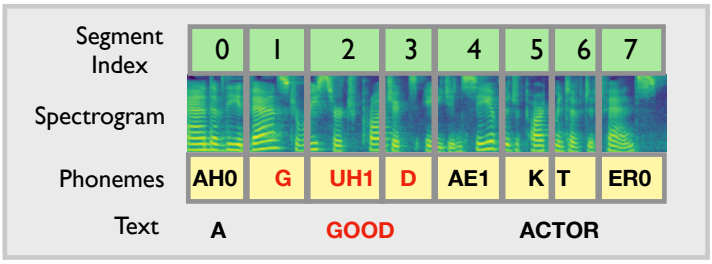


Figure 4.2: Model architecture of A^{3}T.

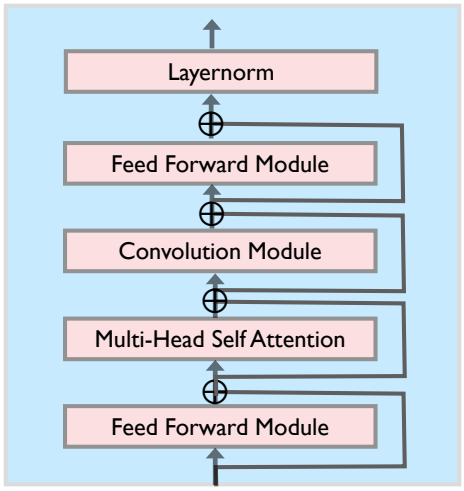
spans of $\mathbf{s}$ by the probability of λ with the same number of a random initialized masking vector $\epsilon_{\mathbf{s}} \in \mathbb{R}^{d_{\mathbf{s}}}$. Then we encode $\hat{\mathbf{s}}$ with a acoustic encoder for acoustic embeddings $\boldsymbol{e}_{\hat{\mathbf{s}}}$. In this work, we use a nonlinear feed-forward layer as the acoustic encoder.

4.3.2 Cross-modal Alignment Embedding

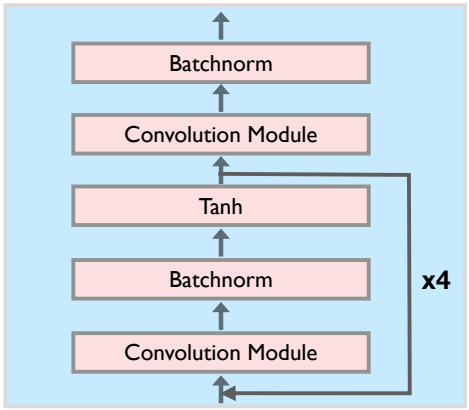
To strengthen the interaction between the speech and text input, we introduce cross-modal alignment embedding as one input of encoder, where we sum the ith acoustic embedding



(a) Forced Alignment Preprocessing.



(b) Conformer Block.



(c) Post-Net.

Figure 4.3: Details of the Embedding block, Conformer block and Post-Net block.

$\boldsymbol{e}_{s_i}$ or text embedding $\mathbf{x}_i$ with its positional embedding $\boldsymbol{e}_{\text{pos}_i}$ and alignment embedding $\boldsymbol{e}_{\text{aln}_i}$ all together: $\boldsymbol{e}_{s_i} + \boldsymbol{e}_{\text{pos}_i} + \boldsymbol{e}_{\text{aln}_i}$, where previous work have proved the embedding sum operation is simple and effective [30, 7]. After that, the phoneme embedding and its acoustic embeddings will share the same alignment embedding. We use a forced aligner [146] to pre-process the dataset to get the alignment information, which is shown in Fig. 4.3(a). Forced alignment refers to the process by which transcriptions are aligned to audio recordings to automatically generate phone-level segmentation. This process is similar to speech recognition but rather than being given a set of possible phoneme candidates by LM, the exact phonemes are given to the force aligner, and then the aligner identifies which speech segment belongs to particular phonemes in the transcription.

4.3.3 Conformer

Given the recent success of Convolution-augmented Transformer (Conformer) on various speech tasks [37, 38], we adopt Conformer as the backbone of our encoder and decoder. Compared with Transformer, Conformer introduces a convolution module and an additional feedforward module, which is shown in Fig. 4.3(b). In our experiments, we find Conformer is better than Transformer for acoustic-text pretraining.

4.3.4 Post-Net and Loss Function

We follow Tacotron 2 [122] to use Post-Net to refine the generated spectrogram. The predicted spectorgram is passed through a 5-layer convolution Post-Net to be refined as shown in Fig. 4.3(c).

The training objective of multi-modal A^{3}T includes a speech reconstruction loss $\ell_{\mathbf{s}}(D_{\mathbf{s},\mathbf{x}})$ which takes a spectrogram $\mathbf{s}$ and a text sequence $\mathbf{x}$ as input. We have the following training objective to reconstruct the original speech signal with the surrounding context information:[3]

$$\begin{aligned}\ell_{\mathbf{s}}(D_{\mathbf{s},\mathbf{x}}) = \sum_{\langle \mathbf{s},\mathbf{x}\rangle \in D_{\mathbf{s},\mathbf{x}}} & \| \underbrace{f([e_{\hat{\mathbf{s}}};\mathbf{x}]) + g\big(f([e_{\hat{\mathbf{s}}};\mathbf{x}])\big)}_{\text{refined spectrogram}} - \mathbf{s}\|_1 \\ & + \| \underbrace{f([e_{\hat{\mathbf{s}}};\mathbf{x}])}_{\text{reconstructed spectrogram}} - \mathbf{s}\|_1\end{aligned} \tag{4.1}$$

where g is a Post-Net which tries to recover a better original signal from encoded representation $f([e_{\hat{\mathbf{s}}};\hat{\mathbf{x}}])$. We use mean absolute error (MAE) for measuring the difference between s and the reconstructed spectrogram.

4.3.5 A^{3}T for Speech Editing

Once A^{3}T finishes the pretraining process, it can be used as a speech editing system directly with an external duration predictor, which is shown in Fig. 4.4.

Given a speech $\mathbf{s}$, its original phonemes $\tilde{\mathbf{x}}$, and the target modified phonemes $\mathbf{x}$, our system first finds the phonemes that need to be modified $\hat{\mathbf{x}}$. To predict the duration $\hat{\mathbf{d}}_i$ of modified phonemes $\mathbf{x}_i \in \hat{\mathbf{x}}$, we use an external duration predictor. Since the duration

[3]Similar with previous work using masked language model objective, this loss only takes the masked input into consideration.

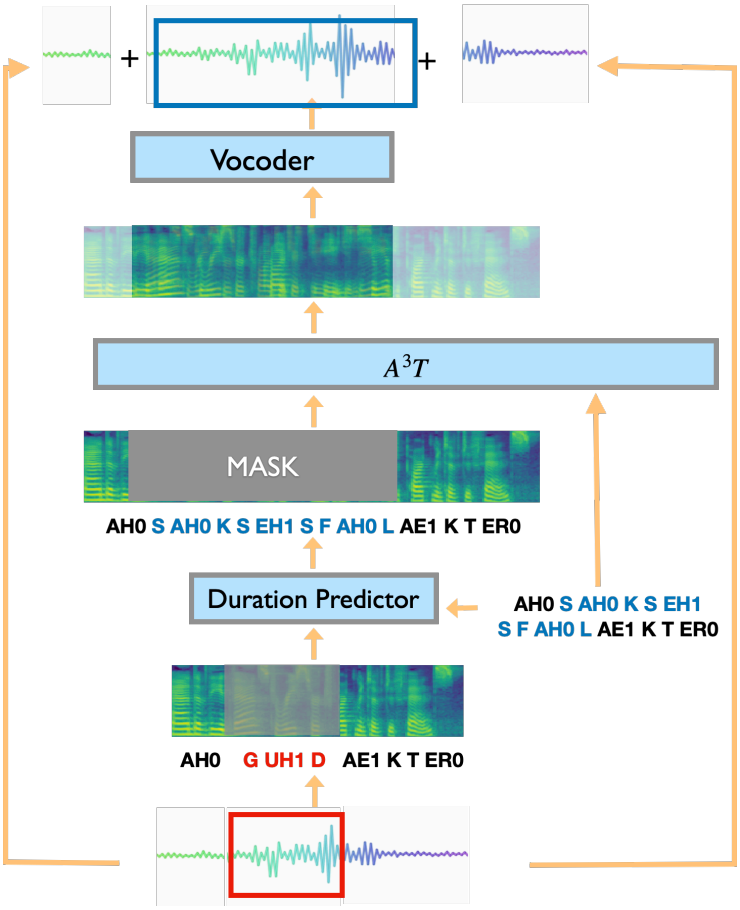


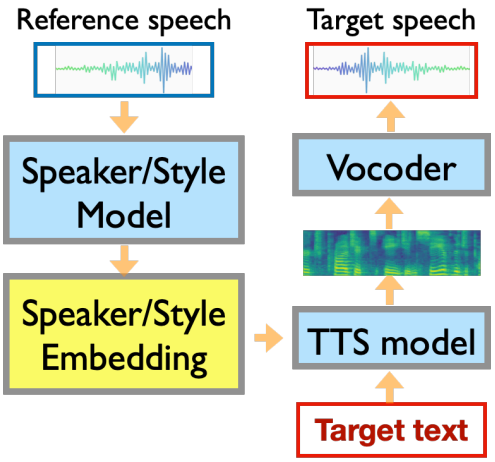
Figure 4.4: Speech editing pipeline.

predictor only takes phoneme sequence as input, we adjust model predicted duration $\mathbf{d}'_i$ according to the origin speech durations as: $\hat{\mathbf{d}}_i = \mathbf{d}'_i \sum_{j=1}^{|\mathbf{x}|} \frac{\tilde{\mathbf{d}}_j}{\mathbf{d}'_j}$, where $\tilde{\mathbf{d}}'_j/\mathbf{d}'_j$ is the ratio between the duration of the original phone $\tilde{\mathbf{x}}_j$ in the given speech and the predicted duration of the original phone $\tilde{\mathbf{x}}_j$ in the original sentence $\tilde{\mathbf{x}}$. We compute this ratio from all phones in the original sentence $\tilde{\mathbf{x}}$ and use it to adjust the predicted duration $\mathbf{d}'_i$ and get the final duration $\hat{\mathbf{d}}_i$ for phone $\hat{\mathbf{x}}_i$.

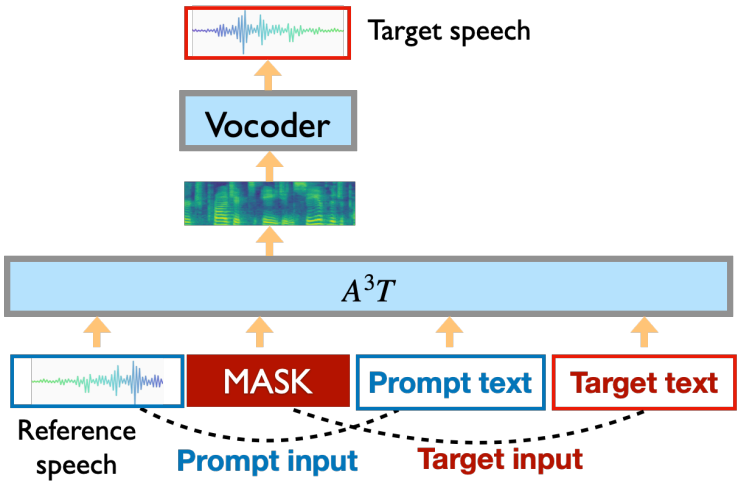
With this predicted duration, we insert $\sum_{i=1}^{|\tilde{\mathbf{x}}|} \hat{\mathbf{d}}_i \cdot \mathtt{sr}/\mathtt{h}$[4] number of `[MASK]` frames into the non-modified spectrogram context. This masked spectrogram $\hat{\mathbf{s}}$ is the input of A^{3}T and $f([e_{\hat{\mathbf{s}}}; \mathbf{x}]) + g\big(f([e_{\hat{\mathbf{s}}}; \mathbf{x}])\big)$ is the prediction of the modified portion spectrogram. Then,

[4] `sr` stands for sample rate and `h` stands for hop size.

we use a vocoder to generate the waveform of this spectrogram and output the final edited speech by replacing the modified part of the original speech.



(a) Speaker/Style embedding-based method.



(b) Prompt-based decoding.

Figure 4.5: Illustrations for one-shot TTS.

4.3.6 A^{3}T for Multi-speaker TTS

In addition to the speech editing, we find our model has the potential for unseen speaker TTS.

Existing popular unseen speaker TTS models [48] are trained with seen speaker embeddings and generalizes to unseen speaker embeddings during the inference. However, such speaker embeddings are extracted from an external speaker verification model which is trained with tens of thousands of speakers.

In this work, we find our model can achieve comparable naturalness to models with speaker embeddings for unseen speaker TTS task; What's more, our generations are more similar to the unseen speaker's reference speech. The illustrations of how to synthesis speech for unseen speakers with our A^{3}T model are shown in Fig. 4.5, which is named prompt-based A^{3}T. The prompt speech and text are wrapped with blue rectangles, and the target speech and text are wrapped with red.

The key idea is to concatenate the prompt and the target together into a new utterance input, where the target speech is consist of n `[MASK]` and n is predicted by a duration predictor. By inputting the concatenated speech and text, A^{3}T model will predict the spectrogram of these masked frames. The role of the reference text and speech in our model is similar to prompts in language model [18], and hence we call it prompt-based decoding/generation. Prompts in language model refer to demonstrations or examples, to perform downstream tasks. Here, the prompts is the speech-text pair, and the downstream task is to mimic the speech example and generate new audio which is similar to the given speaker.

4.4 Experiments

In this section, we introduce our experiments for spectrogram reconstruction pretraining task, speech-editing task, and multi-speaker TTS. The spectrogram reconstruction is our pretraining task, where we conduct ablation study to show the contributions of different components and also the effects of different masking rates. The experiment settings of speech-editing are followed [127], where we deploy two speech-editing systems with two datasets and evaluate the Mel-cepstral distortion (MCD) score and human-annotated mean opinion score (MOS) [24]. The multi-speaker TTS experiments include seen speaker TTS and unseen speaker TTS evaluated with the MOS scores.

4.4.1 Datasets

Following [127], we conduct our speech-editing experiments with a single-speaker TTS dataset LJSpeech [46] and a multi-speaker TTS dataset VCTK [140]. The LJSpeech dataset is a single-speaker dataset with 13K examples in 24 hours. The VCTK dataset is a multi-speaker dataset with 109 speakers and 44K examples in 44 hours. It should be noted that after finishing the pretraining process with LJSpeech or VCTK, our A^{3}T will be used as a speech-editing system without any further finetuning.

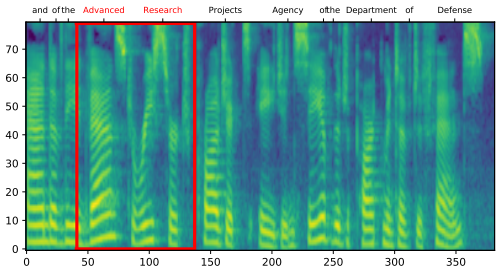
We test multi-speaker TTS task with VCTK dataset. For seen multi-speaker TTS, each speaker's examples would be split into train and test sets. For unseen multi-speaker TTS, the test set contains 10 speakers' examples, and the other 99 speaker's examples are used for training.

4.4.2 Configuration Details

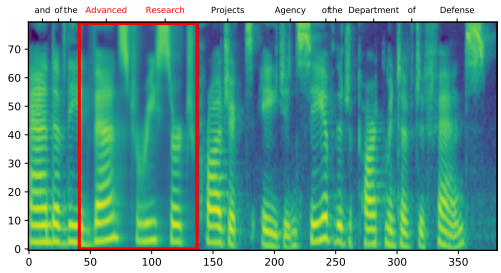
Raw audio files are processed with 50 ms frame size and 12.5 ms frame hop with the Hann window function to extract 80-dimensional log-Mel filterbanks. We use 24K sampling rate for VCTK and 22K for LJSpeech. The forced alignment and G2P are both carried out by HTK [145] to convert English words to phones and align phones with audio segments. For speech-editing systems and prompt-based TTS, we use the publicly available duration predictor from FastSpeech 2 implemented in ESPnet [44]. We use Parallel-WaveGAN [141] vocoder for all the systems.

All A^{3}T models pretrained in our experiments share the same architecture: 4 layers Conformer encoder, 4 layers Conformer decoder, and 5 layers Conv1d Post-Net, with 2 heads multi-head attention in 384-dim. The convolution kernel sizes of the encoder and decoder are 7 and 31, respectively. The shape of alignment embeddings is (500, 384), where we assume the number of phones will not exceed 500 for a single input. The shape of input phone embeddings is (73, 384), and we use a ReLU [1] nonlinear layer to transform 80-dim log-Mel filterbanks features to 384-dim. The total number of parameters is 67.7M.

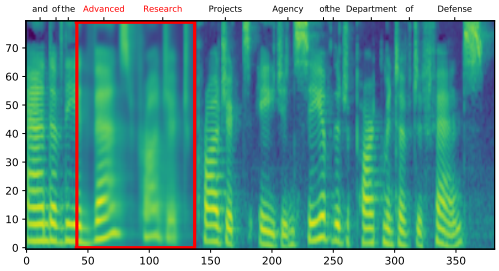
During training, we use Adam optimizer with a 1.0 initial learning rate, 4000 warmup steps, and Noam learning rate scheduler. Instead of setting a fixed batch size, we adjust the batch size according to the length of the input example and set a maximum batch-bin (the total number of input elements) for each model. Following MAM [20], 15% frames will be masked for speech-only input, For speech-text input, we randomly select several phonemes spans (80% phonemes) and mask their corresponding frames. For speech-editing



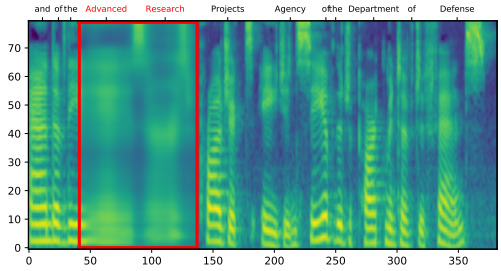
(a) Groundtruth spectrogram from LJSpeech.



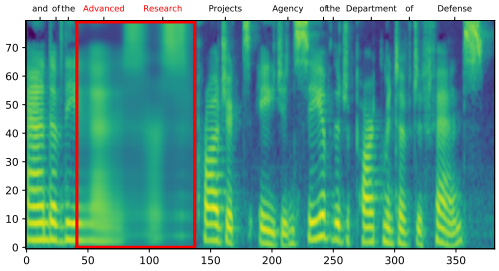
(b) Reconstructed spectrogram by A^3T.



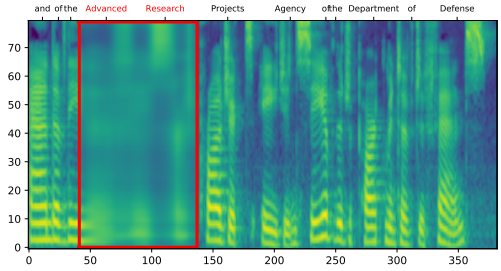
(c) Reconstructed spectrogram based on A^3T in 4.6(b) without segment embedding.



(d) Reconstructed spectrogram based on A^3T in 4.6(c) with Transformer instead of Conformer.

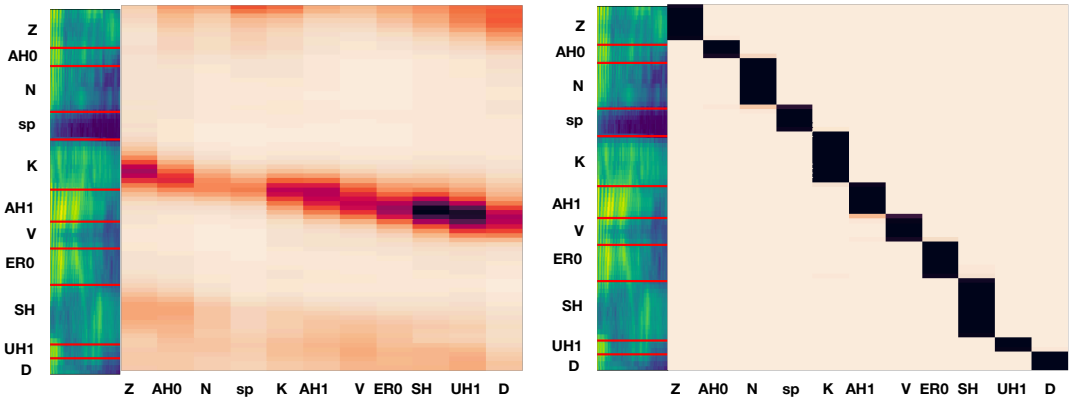


(e) Reconstructed spectrogram based on A^3T in 4.6(d) without Post-Net.



(f) Reconstructed spectrogram based on A^3T in 4.6(e) with L2 loss instead of L1 loss. This model uses the similar architecture of FAT-MLM.

Figure 4.6: An example of ablation study in LJSpeech. Original text is "and of the Advanced Research Projects Agency of the Department of Defense". The portion with red box is "Advanced Research" which is masked in (b,c,d,e,f) subfigures.



(a) Attention map of A^3T w/o alignment embeddings

(b) Attention map of A^3T with alignment embeddings

Figure 4.7: Attention map between speech and text of A^3T with and without alignment embeddings.

experiments, we use 2.4M batch-bin, 1M steps for LJSpeech, and 3M batch-bin, 1.2M steps for VCTK.

4.4.3 Ablation Study with Spectrogram Reconstruction

Example	**Model**	**MCD ↓**
Fig. 4.6(b)	A^3T	8.09
Fig. 4.6(c)	- Alignment Embeddings	10.73
Fig. 4.6(d)	- Conformer	12.43
Fig. 4.6(e)	- Post-Net	12.94
Fig. 4.6(f))	- L1 loss	11.55

Table 4.2: Ablation study for A^3T pretrained with LJSpeech.

We first conduct an ablation study with LJSpeech dataset for our pretraining task: spectrogram reconstruction. This task requires A^3T to predict the masked frames. We sample 30 utterances randomly from the test set, and 1/3 phones in the middle of each sentence are masked. We adopt MCD (Mel-cepstral distortion) to measure the difference between the ground-truth audio and the reconstructed audio, where lower MCD means

MaskRate	Seen MCD ↓	Unseen MCD ↓
20%	10.35	12.22
50%	7.75	9.99
80%	8.72	9.68

Table 4.3: MCD scores of A^{3}T pretrained in different masking rates with VCTK.

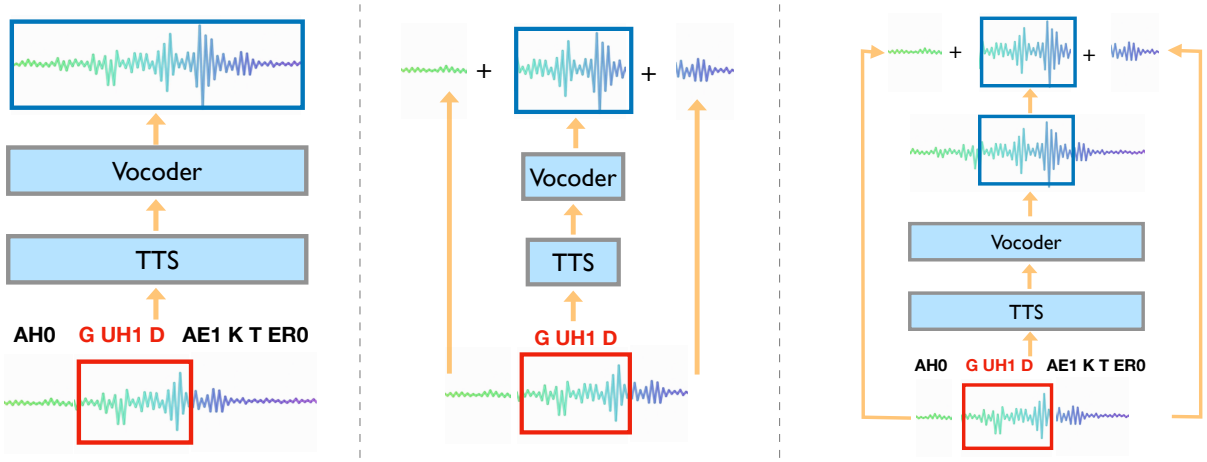
higher similarity. Here we only measure the MCD of the masked/reconstructed region. We incrementally discard the components of A^{3}T: removing the cross-modal alignment embedding, replacing the Conformer with Transformer, removing the Post-Net, and using L2 (MSE) loss instead of L1 (MAE) loss.

Results are shown in Tab. 4.2. We remove modules one by one. For each model, we show a spectrogram example in Fig. 4.6. By comparing Fig. 4.6(b) and Fig. 4.6(c), we can see that many details are lost when A^{3}T trained without the alignment embedding, and the MCD scores rise from 8.09 to 10.73. Similar degrading can be observed after replacing Conformer with Transformer: the MCD scores rise from 10.73 to 12.43 and the spectrogram becomes blurrier (Fig. 4.6(d)). Compared with the alignment embedding and Conformer, Post-Net contributes only 0.49 MCD score, and L2 loss even achieves better MCD score than L1 loss. However, when looking into the spectrograms, we can see that Fig. 4.6(f) is blurrier than Fig. 4.6(e), which conforms to the previous finding [58] that L1 loss is better than L2 loss for speech synthesis. Hence, we choose L1 loss for A^{3}T pretraining. Also, Fig. 4.6(f) indicates the quality that previous pretrained model (MAM/FAT-MAM) could achieve, and the other figures show how our A^{3}T transforms Fig. 4.6(f) to Fig. 4.6(b).

We also conduct a study with VCTK to show the impacts of difference masking rates. Results are shown in Tab. 4.3. We can see that 20% masking rate leads to large MCD scores, while 50% and 80% are better. Also, 50% masking rate outperforms 80% on the seen test cases, but not on the unseen. Considering 80% masking rate has a better generalization on unseen cases, we choose 80% for all the following experiments.

Finally, we plot the attention heat maps of encoder with and without our proposed cross-modal alignment embedding in Fig. 4.7. The attention matrices are collected from the encoder's last layer with a mean-pooling across heads. It should be noted that the original attention matrix is 310*310, which contains both the speech and phones, and for clarity, we plot only 11 phones and their corresponding frames in Fig. 4.7. We can see that our A^{3}T is aware of the speech segmentations and their corresponding phones, while the baseline model fails to capture such alignment information. This observation demonstrates

the effectiveness of our A^{3}T for cross-modal pretraining.



(a) Baseline 1. (b) Baseline 2. (c) Baseline 3.

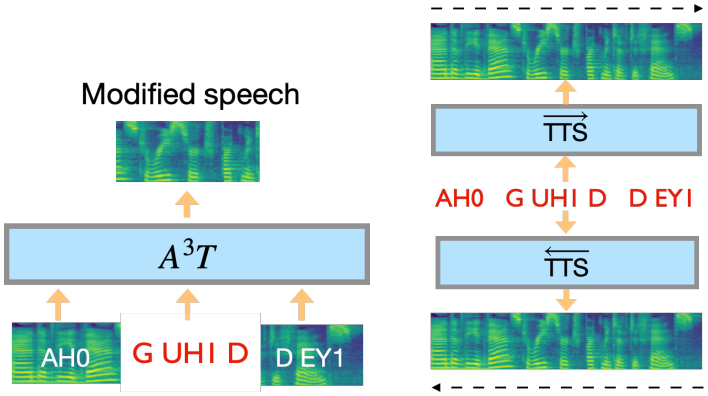
Figure 4.8: Illustrations for speech editing baselines.

4.4.4 Speech Editing

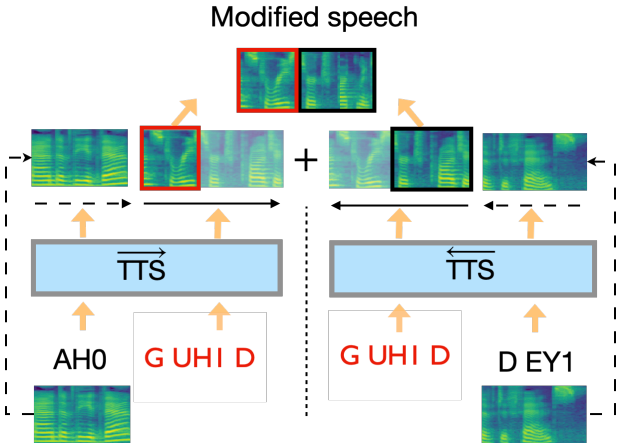
Following [127], we list several baseline systems below:

- **Baseline 1**: This is a TTS system regenerating a complete waveform from the whole sentence to be edited.
- **Baseline 2**: This system generates the modified region with a TTS model and insert the generation back to the original waveform with a forced aligner.
- **Baseline 3**: This system is similar to Baseline 1, but we cut the modified region from the generation and insert it back to the original waveform with a forced aligner.

- **EditSpeech** [127]: This is a speech-editing system which introduces partial inference and bidirectional fusion to sequence-to-sequence neural TTS model. EditSpeech trains two conventional auto-regressive TTS models, one left-to-right and the other right-to-left (Fig. 4.9(b)). For decoding, the left-to-right TTS model force-decodes the prefix speech context and synthesizes the modified region, and the right-to-left TTS model force-decodes the suffix context and generates the modified region reversely. Finally, the two synthesized speeches are fused for final output (Fig. 4.9(c)).



(a) A^3T train/decode (b) EditSpeech train



(c) EditSpeech decode

Figure 4.9: Comparisons between A^3T and EditSpeech. $\rightarrow$: free decoding, $\dashrightarrow$: forced decoding.

Different from EditSpeech, A^3T trains a non-auto-regressive encoder to reconstruct masked acoustic signals, and uses the identical framework for decoding. EditSpeech trains two auto-regressive TTS models: a left-to-right and a right-to-left. For decoding, these two models synthesize two speeches and are fused for the output.

Inspired by [127], we also evaluate our speech editing system with an identical reconstruction task, which is similar to the above ablation experiments but without the ground-truth duration length and can be evaluated with MCD metric. 30 utterances are randomly sampled for each dataset, and a part of speech, which corresponds to 1/3 phonemes in the

middle of each sentence, is masked. The audio of the masked region is replaced with each system's generation. A duration model is used to predict the length of masked speech from phonemes. Results are shown in Tab. 4.4. From this table, we can see that our system achieves the best MCD score. Besides, alignment embedding is the key to reducing MCD, which confirms our observation in Fig. 4.6(c). For TTS-based systems, we find that generating the whole audio and then extracting the modified region is better than generating the modified region only.

We then conduct the human evaluation with Amazon Mechanical Turk for the real speech insertion and replacement tasks using the VCTK dataset. To compare our results with [127], we use the same 15 audio samples and modification operations from their work. For each audio sample, we have 10 English native speakers to evaluate the naturalness of synthesized audios. In Tab.4.5, our A^3T speech editing system outperforms [127]'s and gets the highest MOS (Mean opinion score) scores among all these systems. Audio examples can be found at our demo link.

Model	VCTK MCD ↓	LJSpeech MCD ↓
Baseline 1/3	10.66	10.32
Baseline 2	12.06	10.91
A^3T	**7.76**	**9.26**
w/o Alignment Emb.	11.37	10.30

Table 4.4: MCD evaluation on identity speech reconstruction using VCTK and LJSpeech.

Model	Insert	Replace
Baseline 1	3.02 ± 0.20	2.64 ± 0.16
Baseline 2	2.89 ± 0.17	2.70 ± 0.16
Baseline 3	2.89 ± 0.17	2.44 ± 0.16
EditSpeech[127]	3.50 ± 0.16	3.58 ± 0.16
A^3T	**3.53** ± 0.17	**3.65** ± 0.15
w/o Alignment Emb.	2.48 ± 0.21	1.98 ± 0.17

Table 4.5: The MOS evaluation (↑) on speech editing task on VCTK with 95% confidence intervals.

Model	Seen	Unseen
FastSpeech 2	3.33 ± 0.10	3.78 ± 0.10
+GST [134]	3.42 ± 0.10	3.81 ± 0.11
A^3T	3.61 ± 0.09	3.90 ± 0.10
Groundtruth	3.94 ± 0.08	4.09 ± 0.10

Table 4.6: The MOS evaluation (↑) for speaker similarity on multi-speaker TTS on VCTK with 95% confidence intervals. The FastSpeech2 model is equipped with X-vectors [124].

Model	Seen	Unseen
FastSpeech 2	3.34 ± 0.11	3.85 ± 0.11
+GST [134]	3.27 ± 0.11	3.72 ± 0.11
A^3T	3.63 ± 0.10	3.94 ± 0.11
Groundtruth	4.04 ± 0.08	4.05 ± 0.10

Table 4.7: The MOS evaluation (↑) for speech quality on multi-speaker TTS on VCTK with 95% confidence intervals. The FastSpeech2 model is equipped with X-vectors [124].

4.4.5 Prompt-based Multi-speaker TTS

We also conduct human evaluation for multi-speaker TTS systems with seen speaker and unseen speaker testing cases. The quality of the generations and the speaker similarity between the generation and the reference are evaluated, and the results are shown in Tab. 4.6 and Tab. 4.7. From this table, we can see that the style embedding GST [134] improves the similarity scores but harms the quality scores, while our A^3T model is the most favourable system in both the speaker similarity and the speech quality. Strikingly, we observe that, the average score of the Unseen cases are higher than the Seen, which is counterintuitive. However, when looking into the MOS of the groundtruth, the gap is still there and we believe this is due to the difference of these two test cases set . Audio examples can be found at our demo link.

4.5 Summary

In this chapter, we introduce Alignment-Aware Acoustic-Text Pretraining (A^3T) which can reconstruct masked acoustic signals with high quality. The distinctive feature of our

proposed A^3T model is that it leverages both the segmentation and alignment feature of natural language for text-speech sequence modeling. Our experiments show the importance of these features. This is the first pretraining method for speech synthesis that can generate high-quality speech without any finetuning. Our proposed A^3T model can do speech editing and outperforms the previous SOTA models. Also, A^3T improves multi-speaker speech synthesis without any external speaker verification model.

Although we only focus on the speech synthesis task in this chapter, our proposed A^3T model can be applied to other sequence modeling tasks, such as hand-writing synthesis. In this case, the pretrained model may generate high-quality images which can help to repair damaged handwriting images. We leave this direction for future work.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis is about modeling language sequences to achieve lower perplexity, better generation, and benefit downstream language tasks. Although increasing the training data and model parameters can achieve the above objectives, we argue that such improvements are limited by the training cost and the availability of the training data.

In this thesis, we emphasize the importance of language features, including the segmentation feature, lexical feature, and alignment feature, which are essential for language modeling. This thesis presents three new techniques that improve language sequence modeling with the above language features, including better model architecture with segmentation feature in Chapter 2, better training method with the lexical feature in Chapter 3, and modeling cross-modal (text and speech) sequences with both the segmentation and alignment feature in Chapter 4.

Improving Text Sequence Modeling with Segmentation Feature. The input of language modeling is usually a long document with different sentence and paragraph segments. Although vanilla position encoding can help Transformer be aware of the token position by assigning a unique index to each token, the token index in a sentence, sentence index in a paragraph, and paragraph index in a document are all implicit. With our proposed method in Chapter 2, we can explicitly encode the segment position information into the Transformer, which reduces the LM perplexity, boosts the pretraining, and benefits the downstream tasks. We illustrate the importance of the segmentation feature by comparing the proposed Segatron with the vanilla Transformer in various settings, which also shows

the effectiveness of the proposed method. The proposed Segatron works for both the language models with relative position encoding and the pretrained language models with absolute position encoding.

Improving Rare Words Modeling with Lexical Feature. Performance improvement of today's neural LMs is often achieved at the cost of increased computational resources. Exploring training strategies that can achieve lower perplexity without increasing the computational resources is an important direction. In Chapter 3, we revisit curriculum learning and propose a novel training strategy for the Transformer language model to gradually learn the lexical knowledge by curriculum training, which can achieve lower perplexity without increasing the computational resources. The proposed method is based on WordNet's super-subordinate relation and curriculum learning. Although WordNet is an external resource, it was not clear how to get lower perplexity using WordNet before this work. This is the first work that shows how the perplexity of Transformer LMs can be improved by leveraging hypernym relationships. Also, we provide an extensive ablation study highlighting crucial elements of our method. The results confirm that our method is effective and the lexical feature can reduce the perplexities with curriculum learning.

Facilitating Acoustic-Text Sequence Modeling with Segmentation and Alignment Features. Acoustic pretraining has improved many speech-related tasks, such as speech recognition, speech classification and speech-to-text translation. However, all the above tasks are in the direction of *speech understanding*, but for the inverse direction, *speech synthesis*, the potential of representation learning is yet to be realized, due to the challenging nature of generating high-quality speech. To address this problem, we introduce our framework Alignment-Aware Acoustic-Text Pretraining (A^3T) in Chapter 4, which reconstructs masked acoustic signals with text input and acoustic-text alignment during training. The key idea is to use the acoustic-text alignment feature and phoneme segmentation feature to guide the acoustic reconstruction. In this way, the pretrained model can generate high quality of reconstructed spectrogram, which can be applied to the speech editing and unseen speaker TTS directly. Experiments show A^3T outperforms the state-of-the-art models on speech editing, and improves multi-speaker speech synthesis without the external speaker verification model. The results demonstrate the effectiveness of the proposed method and the necessity of the alignment feature and phoneme segmentation feature for acoustic-text sequence modeling.

5.2 Future work

Exploration of Other Language Features for Language Modeling. In this thesis, we introduce the segment feature, linguistic feature, and alignment feature, and show how these features can help language sequence modeling. However, for each kind of feature, many variants at different levels can be explored. For example, we explore the paragraph segment, sentence segment, and phoneme segment of the segment feature. We show these features can help both text sequence modeling and speech sequence modeling. In addition to these segments, there are many other language segments, i.e. subword segments, phrase segments, semantic role segments, etc. Also, for linguistic features, this thesis only introduces a super-subordinate lexical feature. Other linguistic features such as POS tag, entity, and dependency relation can be incorporated into LM training with our proposed curriculum training method too. And for the alignment feature, we use the phoneme-frame alignment to facilitate the pretraining of speech-text sequence. In addition to speech-text alignment, cross-lingual text alignment features may also be effective to improve cross-lingual pretraining.

Extending Language Sequence Modeling Methods to Other Sequences. This thesis mainly focuses on the exploration of language features for language sequence modeling. There are many other sequences in the world, i.e. biology sequence (genetic and peptide) and image sequence (video and hand-writing), and distinct features can be found for these sequences. We argue that the proposed methods which work for language sequences have the potential to be extended to other sequences. For example, various relationships can be found among different frames of the video image sequence, where related frames can be grouped. These features are similar to the language segment feature, where Segatron and A^3T may be applied to modeling such a sequence. Another example could be the hand-writing image. The sequence modeling of hand-writing image and transcript is similar to the modeling of speech-text, and A^3T can be applied for such pretraining. In this case, the pretrained model may generate high-quality images which can help to repair damaged handwriting documents.

Language Modeling on a Large Scale. GPT-3 has shown great potential in few-shot learning and text generation, trained with 300 billion tokens and 175B parameters. However, training such a large model is costly. In this project, we would like to answer this question: Can multiple experts outperform a super expert? In other words, can ten 17.5B models compete or outperform one 175B model? To answer this question, we will first explore the relationship between model size and training data in the context of GPT pretraining. We would also explore how to schedule the data distribution during the

training of multiple LMs. Finally, how to select one from all generations of multiple LMs will be investigated. This research direction is important to the NLP community, where the possible impacts include: a new direction for large LM pretraining; GPT-3 could become easier to be trained without a large amount of GPUs; GPT-3 could be investigated with GPUs of the university labs, while the current large model can only be investigated with super GPU clusters like Azure, AWS, and Google Cloud.

Multilingual Language Modeling. Multilingual pretrained LMs have achieved SOTA results on low-resource and cross-lingual NLP tasks. These strong PLMs are either pretrained with MLM with monolingual inputs or parallel sequence pair inputs. The former learns MLM for different languages and the latter learns bilingual alignments with MLM task. However, we argue that these pretraining tasks are too easier to learn the alignments among different languages. We will present a more challenging pretraining task: Sentence level code-switching MLM, where we will map sentences in a document into different languages, and conduct MLM on the mapped document. The training would be on top of existing multilingual PLMs, e.g. mT5 and XLM. And the mapping processing will be relying on Wikipedia where a document in different languages can be found. This project can further improve the existing multilingual pretrained models and benefit low-resource language tasks and cross-lingual understanding tasks.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *ArXiv Prepr. ArXiv180308375*, 2018.

[2] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proc. AAAI Conf. Artif. Intell.*, volume 33, pages 3159–3166, 2019.

[3] Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, et al. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. *arXiv preprint arXiv:2110.07205*, 2021.

[4] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *ICLR*, 2019.

[5] Alexei Baevski and Abdelrahman Mohamed. Effectiveness of self-supervised pre-training for asr. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7694–7698. IEEE, 2020.

[6] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. Wav2vec 2.0: A framework for self-supervised learning of speech representations. *NeurIPS*, 2020.

[7] Bai, He, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. Segatron: Segment-aware transformer for language modeling and understanding. *ArXiv Prepr. ArXiv200414996*, 2020.

[8] He Bai, Peng Shi, Jimmy Lin, Luchen Tan, Kun Xiong, Wen Gao, Jie Liu, and Ming Li. Semantics of the unwritten: The effect of end of paragraph and sequence tokens on text generation with GPT2. In *Proc. 59th Annu. Meet. Assoc. Comput. Linguist. 11th Int. Jt. Conf. Nat. Lang. Process. Stud. Res. Workshop*, pages 148–162, Online, August 2021. Association for Computational Linguistics.

[9] He Bai, Tong Wang, Alessandro Sordoni, and Peng Shi. Better language model with hypernym class prediction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1352–1362, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[10] Ankur Bapna, Yu-an Chung, Nan Wu, Anmol Gulati, Ye Jia, Jonathan H Clark, Melvin Johnson, Jason Riesa, Alexis Conneau, and Yu Zhang. SLAM: A unified encoder for speech and language modeling via speech-text joint pre-training. *ArXiv Prepr. ArXiv211010329*, 2021.

[11] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *ArXiv Prepr. ArXiv200405150*, 2020.

[12] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[13] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum Learning. In *Proc. 26th Annu. Int. Conf. Mach. Learn.*, ICML '09, pages 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[14] Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *TAC 2009 Gaithersburg Md. USA Novemb. 16-17 2009*, 2009.

[15] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly, 2009.

[16] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proc. 2015 Conf. Empir. Methods Nat. Lang. Process.*, pages 632–642, Lisbon, Portugal, 2015. Association for Computational Linguistics.

[17] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–480, 1992.

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark,

Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Adv. Neural Inf. Process. Syst.*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[19] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proc. 11th Int. Workshop Semantic Eval. SemEval-2017*, pages 1–14, Vancouver, Canada, 2017. Association for Computational Linguistics.

[20] Junkun Chen, Mingbo Ma, Renjie Zheng, and Liang Huang. MAM: Masked acoustic modeling for end-to-end speech-to-text translation. *ArXiv Prepr. ArXiv201011445*, 2020.

[21] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[22] Po-Han Chi, Pei-Hung Chung, Tsung-Han Wu, Chun-Cheng Hsieh, Yen-Hao Chen, Shang-Wen Li, and Hung-yi Lee. Audio albert: A lite bert for self-supervised learning of audio representation. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 344–350. IEEE, 2021.

[23] 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 Prepr. ArXiv220402311*, 2022.

[24] Min Chu and Hu Peng. Objective measure for estimating mean opinion score of synthesized speech, April 2006.

[25] 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.

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

[27] Ido Dagan, Lillian Lee, and Fernando CN Pereira. Similarity-based models of word cooccurrence probabilities. *Mach. Learn.*, 34(1):43–69, 1999.

[28] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. 57th Annu. Meet. Assoc. Comput. Linguist.*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[29] Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. In *ICLR*, 2020.

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

[31] Siyu Ding, Junyuan Shang, Shuohuan Wang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Ernie-doc: A retrospective long-document modeling transformer. *arXiv preprint arXiv:2012.15688*, 2020.

[32] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proc. Third Int. Workshop Paraphrasing IWP2005*, 2005.

[33] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

[34] Cong Fang, Hangfeng He, Qi Long, and Weijie J. Su. Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. *Proc. Natl. Acad. Sci.*, 118(43), 2021.

[35] Édouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for GPUs. In Doina Precup and Yee Whye Teh, editors, *Proc. 34th Int. Conf. Mach. Learn.*, volume 70 of *Proceedings of Machine Learning Research*, pages 1302–1310. PMLR, August 2017.

[36] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *5th Int. Conf. Learn. Represent. ICLR 2017 Toulon Fr. April 24-26 2017 Conf. Track Proc.* OpenReview.net, 2017.

[37] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *ArXiv Prepr. ArXiv200508100*, 2020.

[38] Pengcheng Guo, Florian Boyer, Xuankai Chang, Tomoki Hayashi, Yosuke Higuchi, Hirofumi Inaguma, Naoyuki Kamo, Chenda Li, Daniel Garcia-Romero, Jiatong Shi, et al. Recent developments on espnet toolkit boosted by conformer. In *ICASSP 2021-2021 IEEE Int. Conf. Acoust. Speech Signal Process. ICASSP*, pages 5874–5878. IEEE, 2021.

[39] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[40] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pretraining, 2020.

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

[42] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *ArXiv Prepr. ArXiv220315556*, 2022.

[43] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *ArXiv Prepr. ArXiv210607447*, 2021.

[44] Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplin, Tomoki Hayashi, and Shinji Watanabe. ESPnet-ST: All-in-one speech translation toolkit. *ArXiv Prepr. ArXiv200410234*, 2020.

[45] Dan Iter, Kelvin Guu, Larry Lansing, and Dan Jurafsky. Pretraining with contrastive sentence objectives improves discourse performance of language models. *arXiv preprint arXiv:2005.10389*, 2020.

[46] Keith Ito and Linda Johnson. The LJ Speech Dataset, 2017.

[47] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.

[48] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Adv. Neural Inf. Process. Syst.*, 31, 2018.

[49] Dongwei Jiang, Wubo Li, Ruixiong Zhang, Miao Cao, Ne Luo, Yang Han, Wei Zou, Kun Han, and Xiangang Li. A further study of unsupervised pretraining for transformer based speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6538–6542. IEEE, 2021.

[50] Zeyu Jin, Gautham J Mysore, Stephen Diverdi, Jingwan Lu, and Adam Finkelstein. Voco: Text-based insertion and replacement in audio narration. *ACM Trans. Graph. TOG*, 36(4):1–13, 2017.

[51] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Trans. Assoc. Comput. Linguist.*, 8:64–77, 2020.

[52] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[53] Dan Jurafsky. *Speech & Language Processing*. Pearson Education India, 2000.

[54] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[55] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*, 2020.

[56] Yerbolat Khassanov, Zhiping Zeng, Van Tung Pham, Haihua Xu, and Eng Siong Chng. Enriching rare word representations in neural language models by embedding matrix augmentation. *Interspeech 2019*, September 2019.

[57] Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5530–5540. PMLR, 18–24 Jul 2021.

[58] Viacheslav Klimkov, Alexis Moinet, Adam Nadolski, and Thomas Drugman. Parameter generation algorithms for text-to-speech synthesis with recurrent neural networks. In *2018 IEEE Spok. Lang. Technol. Workshop SLT*, pages 626–631. IEEE, 2018.

[59] Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. In Jennifer G. Dy and Andreas Krause, editors, *Proc. 35th Int. Conf. Mach. Learn. ICML 2018 Stock. Stockh. Swed. July 10-15 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2771–2780. PMLR, 2018.

[60] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proc. 2018 Conf. Empir. Methods Nat. Lang. Process. Syst. Demonstr.*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[61] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[62] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proc. 2017 Conf. Empir. Methods Nat. Lang. Process.*, pages 785–794, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

[63] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th Int. Conf. Learn. Represent. ICLR 2020 Addis Ababa Ethiop. April 26-30 2020*. OpenReview.net, 2020.

[64] Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Sebastian Ruder, Dani Yogatama, et al. Pitfalls of static language modelling. *ArXiv Prepr. ArXiv210201951*, 2021.

[65] Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. SenseBERT: Driving some sense into BERT. In *Proc. 58th Annu. Meet. Assoc. Comput. Linguist.*, pages 4656–4667, Online, July 2020. Association for Computational Linguistics.

[66] 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.

[67] Junjie Li, Xuepeng Wang, Dawei Yin, and Chengqing Zong. Attribute-aware sequence network for review summarization. In *Proc. 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. EMNLP-IJCNLP*, pages 3000–3010, Hong Kong, China, 2019. Association for Computational Linguistics.

[68] Andy T Liu, Shang-Wen Li, and Hung-yi Lee. Tera: Self-supervised learning of transformer encoder representation for speech. *ArXiv Prepr. ArXiv200706028*, 2020.

[69] Andy T Liu, Shu-wen Yang, Po-Han Chi, Po-chun Hsu, and Hung-yi Lee. Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6419–6423. IEEE, 2020.

[70] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proc. 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. EMNLP-IJCNLP*, pages 3730–3740, Hong Kong, China, 2019. Association for Computational Linguistics.

[71] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.

[72] 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. *Arixv*, abs/1907.11692, 2019.

[73] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *arXiv preprint arXiv:2211.02001*, 2022.

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

[75] Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proc. Seventeenth Conf. Comput. Nat. Lang. Learn.*, pages 104–113, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[76] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[77] Matt Mahoney. Large text compression benchmark, 2011.

[78] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[79] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *ArXiv Prepr. ArXiv160907843*, 2016.

[80] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

[81] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[82] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE, 2012.

[83] George A. Miller. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41, November 1995.

[84] Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proc. 24th Int. Conf. Mach. Learn.*, ICML '07, pages 641–648, New York, NY, USA, 2007. Association for Computing Machinery.

[85] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Int. Workshop Artif. Intell. Stat.*, pages 246–252. PMLR, 2005.

[86] Max Morrison, Lucas Rencker, Zeyu Jin, Nicholas J Bryan, Juan-Pablo Caceres, and Bryan Pardo. Context-aware prosody correction for text-based speech editing. In *ICASSP 2021-2021 IEEE Int. Conf. Acoust. Speech Signal Process. ICASSP*, pages 7038–7042. IEEE, 2021.

[87] Eric Moulines and Francis Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Commun.*, 9(5-6):453–467, 1990.

[88] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel WaveNet: Fast high-fidelity speech synthesis. In *Int. Conf. Mach. Learn.*, pages 3915–3923, 2018.

[89] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[90] Xiaoman Pan, Kai Sun, Dian Yu, Jianshu Chen, Heng Ji, Claire Cardie, and Dong Yu. Improving question answering with external knowledge. In *Proc. 2nd Workshop Mach. Read. Quest. Answering*, pages 27–37, Hong Kong, China, 2019. Association for Computational Linguistics.

[91] Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao. Parallel neural text-to-speech. *ArXiv Prepr. ArXiv190508459*, 2019.

[92] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[93] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *ArXiv Prepr. ArXiv180205365*, 2018.

[94] 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.

[95] Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: The word-in-context dataset for evaluating context-sensitive meaning representations. In *Proc. 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. Vol. 1 Long Short Pap.*, pages 1267–1273, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[96] Ian Porada, Kaheer Suleman, Adam Trischler, and Jackie Chi Kit Cheung. Modeling event plausibility with consistent conceptual abstraction. In *Proc. 2021 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.*, pages 1732–1743, Online, June 2021. Association for Computational Linguistics.

[97] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE Int. Conf. Acoust. Speech Signal Process. ICASSP*, pages 3617–3621. IEEE, 2019.

[98] Ofir Press, Noah A. Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs. In *Proc. 59th Annu. Meet. Assoc. Comput. Linguist. 11th Int. Jt. Conf. Nat. Lang. Process. Vol. 1 Long Pap.*, pages 5493–5505, Online, August 2021. Association for Computational Linguistics.

[99] Alec Radford. Improving language understanding by generative pretraining. OpenAI blog, 2018.

[100] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

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

[102] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[103] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling Language Models: Methods, Analysis & Insights from Training Gopher, January 2022.

[104] Jack W. Rae, Chris Dyer, Peter Dayan, and Timothy P. Lillicrap. Fast parametric learning with activation memorization. In *ICML 2018 Stock. Stockh. Swed. July 10-15 2018*, volume 80, pages 4225–4234, 2018.

[105] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *ICLR*, 2020.

[106] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:1–67, 2020.

[107] Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguist. Vol. 1 Long Pap.*, pages 99–110, Valencia, Spain, April 2017. Association for Computational Linguistics.

[108] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proc. 56th Annu. Meet. Assoc. Comput. Linguist. Vol. 2 Short Pap.*, pages 784–789, Melbourne, Australia, 2018. Association for Computational Linguistics.

[109] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. 2016 Conf. Empir. Methods Nat. Lang. Process.*, pages 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics.

[110] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-Networks. In *Proc. 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. EMNLP-IJCNLP*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics.

[111] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. *ArXiv Prepr. ArXiv200604558*, 2020.

[112] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. FastSpeech: Fast, robust and controllable text to speech. *ArXiv Prepr. ArXiv190509263*, 2019.

[113] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. Recipes for building an open-domain chatbot. *ArXiv Prepr. ArXiv200413637*, 2020.

[114] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguist.*, 9:53–68, 2021.

[115] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *Int. Conf. Mach. Learn.*, pages 8346–8356. PMLR, 2020.

[116] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[117] Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, Saiful Bari, Stella Rose Biderman, Hady ElSahar, Jason Phang, Ofir Press, Colin Raffel, Victor Sanh, Sheng Shen, Lintang A. Sutawika, Jaesung Tae, Zheng Xin Yong, Julien Launay, and Iz Beltagy. What language model to train if you have one million GPU hours? 2022.

[118] Timo Schick and Hinrich Schütze. Rare words: A major problem for contextualized representation and how to fix it by attentive mimicking. In *Proc. Thirty-Fourth AAAI Conf. Artif. Intell.*, 2020.

[119] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[120] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *ArXiv Prepr. ArXiv150807909*, 2015.

[121] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[122] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE Int. Conf. Acoust. Speech Signal Process. ICASSP*, pages 4779–4783. IEEE, 2018.

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

[124] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE Int. Conf. Acoust. Speech Signal Process. ICASSP*, pages 5329–5333. IEEE, 2018.

[125] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. 2013 Conf. Empir. Methods Nat. Lang. Process.*, pages 1631–1642, Seattle, Washington, USA, 2013. Association for Computational Linguistics.

[126] Xingchen Song, Guangsen Wang, Zhiyong Wu, Yiheng Huang, Dan Su, Dong Yu, and Helen Meng. Speech-xlnet: Unsupervised acoustic model pretraining for self-attention networks. *arXiv preprint arXiv:1910.10387*, 2019.

[127] Daxin Tan, Liqun Deng, Yu Ting Yeung, Xin Jiang, Xiao Chen, and Tan Lee. EditSpeech: A text based speech editing system using partial inference and bidirectional fusion. *ArXiv Prepr. ArXiv210701554*, 2021.

[128] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *ArXiv Prepr. ArXiv160903499*, 2016.

[129] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Adv. Neural Inf. Process. Syst. 30 Annu. Conf. Neural Inf. Process. Syst. 2017 Dec. 4-9 2017 Long Beach CA USA*, pages 5998–6008, 2017.

[130] 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. *Adv. Neural Inf. Process. Syst.*, 32, 2019.

[131] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th Int. Conf. Learn. Represent. ICLR 2019 New Orleans USA May 6-9 2019*. OpenReview.net, 2019.

[132] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization?, April 2022.

[133] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. StructBERT: Incorporating language structures into pre-training for deep language understanding. In *8th Int. Conf. Learn. Represent. ICLR 2020 Addis Ababa Ethiop. April 26-30 2020*. OpenReview.net, 2020.

[134] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ-Skerry Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Ye Jia, Fei Ren, and Rif A Saurous. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *Int. Conf. Mach. Learn.*, pages 5180–5189. PMLR, 2018.

[135] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguist.*, 7:625–641, 2019.

[136] Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. Should you mask 15% in masked language modeling? *ArXiv Prepr. ArXiv220208005*, 2022.

[137] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. Vol. 1 Long Pap.*, pages 1112–1122, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

[138] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? In *ICLR*, 2021.

[139] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[140] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92), 2019.

[141] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE Int. Conf. Acoust. Speech Signal Process. ICASSP*, pages 6199–6203. IEEE, 2020.

[142] 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.

[143] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Adv. Neural Inf. Process. Syst. 32 Annu. Conf. Neural Inf. Process. Syst. 2019 NeurIPS 2019 Dec. 8-14 2019 Vanc. BC Can.*, pages 5754–5764, 2019.

[144] Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. Coreferential reasoning learning for language representation. *arXiv preprint arXiv:2004.06870*, 2020.

[145] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The HTK book. *Camb. Univ. Eng. Dep.*, 3(175):12, 2002.

[146] Jiahong Yuan and Mark Liberman. Speaker identification on the SCOTUS corpus. *J. Acoust. Soc. Am.*, 123(5):3878, 2008.

[147] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Adv. Neural Inf. Process. Syst.*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020.

[148] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization, February 2015.

[149] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[150] Renjie Zheng, Junkun Chen, Mingbo Ma, and Liang Huang. Fused acoustic and text encoding for multimodal bilingual pretraining and speech translation. *Proc. 38th Int. Conf. Mach. Learn.*, 2021.

[151] Zexuan Zhong, Tao Lei, and Danqi Chen. Training language models with memory augmentation. *arXiv preprint arXiv:2205.12674*, 2022.

[152] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE Int. Conf. Comput. Vis. ICCV 2015 Santiago Chile Dec. 7-13 2015*, pages 19–27. IEEE Computer Society, 2015.

[153] Zachary M. Ziegler and Alexander M. Rush. Latent normalizing flows for discrete sequences, 2019.