Towards Effective Utilization of Pretrained Language Models

— Knowledge Distillation from BERT

by

Linqing Liu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this 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.

Abstract

In the natural language processing (NLP) literature, neural networks are becoming increasingly deeper and more complex. Recent advancements in neural NLP are large pretrained language models (e.g. BERT), which lead to significant performance gains in various downstream tasks. Such models, however, require intensive computational resource to train and are difficult to deploy in practice due to poor inference-time efficiency. In this thesis, we are trying to solve this problem through knowledge distillation (KD), where a large pretrained model serves as teacher and transfers its knowledge to a small student model. We also want to demonstrate the competitiveness of small, shallow neural networks.

We propose a simple yet effective approach that transfers the knowledge of a large pretrained network (namely, BERT) to a shallow neural architecture (namely, a bidirectional long short-term memory network). To facilitate this process, we propose heuristic data augmentation methods, so that the teacher model can better express its knowledge on the augmented corpus. Experimental results on various natural language understanding tasks show that our distilled model achieves high performance comparable to the ELMo model (a LSTM based pretrained model) in both single-sentence and sentence-pair tasks, while using roughly 60–100 times fewer parameters and 8–15 times less inference time.

Although experiments show that small BiLSTMs are more expressive on natural language tasks than previously thought, we wish to further exploit its capacity through a different KD framework. We propose MKD, a Multi-Task Knowledge Distillation Approach. It distills the student model from different tasks jointly, so that the distilled model learns a more universal language representation by leveraging cross-task data. Furthermore, we evaluate our approach on two different student model architectures, one is bi-attentive LSTM based network, another uses three layer Transformer models. For LSTM based student, our approach keeps the advantage of inference speed while maintaining comparable performance as those specifically designed for Transformer methods. For our Transformer-based student, it does provide a modest gain, and outperforms other KD methods without using external training data.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Jimmy Lin, for his invaluable guidance and inspiring discussions during my study. He has taught me the methodology to carry out the research and to present the research works in a clear and attractive way. Besides being a brilliant mind in the research field, Jimmy is also a super cool person! He deeply influenced me with his dynamism and optimism towards the surrounding world.

I also want to thank the readers of my thesis, Professor Yaoliang Yu and Professor Charles Clark, for reviewing my work. Thanks Yaoliang for inspiring us on our first knowledge distillation work in his Theory of Deep Learning class.

I would like to express my sincere gratitude to my supervisor, Caiming Xiong, at Salesforce Research. He always has a very insightful, high-level view about this field and he understands the nature of the problems very deeply. I have been truly amazed by his insight and broad knowledge during every discussion with him. He led me to think critically and empirically. He's an undoubted world expert in both academia and industry.

I was very lucky to be surrounded by many amazing friends at the University of Waterloo: Michael, Ralph, Xinji, Lili, Qian, Hao, and Yangtian (knowing that I would miss someone). I also want to thank my friends at Salesforce for all the fun times we spent together in sunny California. Special thanks go to Yao, thank you for all the accompany and support from college to graduate school.

Finally, I would like to thank my family for their unwavering love and encouragement throughout all these years. Thank you for always being there for me.

Dedication

This is dedicated to my parents.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the natural language processing (NLP) domain, a wide range of neural networks such as convolutional neural networks (CNNs) [31, 32], recurrent neural networks (RNNs) [76] and attention mechanisms [2] have been employed to solve different downstream tasks. These neural models usually use distributed representations to implicitly capture semantic and syntactic information of the language. Most of them are relatively shallow and only comprise less than three neural layers. These models, such as RNN, are typical for sequence modeling tasks. However, the sequential nature of RNN makes it difficult in learning long-term dependencies [24] and taking advantage of parallel computing devices.

Recently, Devlin et al. [80] propose a new type of deep model Transformer, which is solely based on attention mechanisms and can directly model the dependency between every pair of words in a sequence. It advances the model architectures from shallow to deep. After that, various pretrained language models (PLMs) built with Transformers have been highly successful in effectiveness gains across many NLP tasks. They learn highly effective general language representations from large-scale unlabeled data. A few prominent examples include ELMo [61], BERT [17], RoBERTa [51], and XLNet [88]. However, such models use dozens, if not hundreds, of millions of parameters. Due to large number of parameters, they are undeployable in resource-restricted systems such as mobile devices. They may be inapplicable in real-time systems either, because of low inference-time efficiency. The consensus [68, 75, 29] is that we need to cut down the model size and reduce the computational cost while maintaining comparable quality.

One approach to solve this problem is knowledge distillation [1, 23] - where a larger model serves as a teacher and a small model learns to mimic the teacher as a student. This approach is model agnostic: the choice of student model does not depend on the teacher

model architecture; The teacher model can be easily switched to any powerful PLMs. This advantage makes it easy to further compare both the influence of teacher and student models.

To the best of our knowledge, we are the first to explore distilling knowledge from BERT, one of the most influential PLMs [78]. We transfer task-specific knowledge from BERT to a shallow neural architecture — in particular, a bidirectional long short-term memory network (BiLSTM). Our motivation is twofold: we question whether a simple architecture actually lacks representation power for text modeling, and we wish to study effective approaches to transfer knowledge from BERT to a BiLSTM. To facilitate effective knowledge transfer, however, we often require a large, unlabeled dataset. To this end, we further propose a novel, rule-based textual data augmentation approach for constructing augmented samples. We evaluate our approach on six tasks in sentence classification and sentence matching. Experiments show that our knowledge distillation procedure significantly outperforms training the original simpler network alone. With our approach, a shallow BiLSTM-based model achieves results comparable to ELMo, a LSTM-based pre-trained model, while using around 100 times fewer parameters for single sentence tasks.

Besides our work mentioned above, all other previous methods [75, 29, 91] all focus on task-specific KD, which transfers knowledge from a single-task teacher to its single-task student. Put it another way, the knowledge distillation process needs to be conducted all over again when performing on a new NLP task. The inference speed of the large-scale teacher model remains the bottleneck for various downstream tasks distillation. Our goal is to find a distill-once-fits-many solution.

To achieve this goal, we explore [45] the knowledge distillation method under the setting of multi-task learning (MTL; [7, 3]). We propose to distill the student model from different tasks jointly. The reason is twofold: firstm the distilled model could learn a more universal language representation by leveraging cross-task data. Second, the student model achieves both comparable quality and fast inference speed across multiple tasks. We evaluate our approach on two different student model architectures. One uses three layers Transformers, since most of the KD works [75] use Transformers as their students. Another is LSTM based network with bi-attention mechanism. Since we already examine the representation capacity of a simple, single-layer Bi-LSTM only in [78], we are interested in whether adding more previous effective modules, such as an attention mechanism, will further improve its effectiveness. We evaluate our approach on GLUE benchmark [81]. For LSTM based student, our approach keeps the advantage of inference speed while maintaining comparable performances as those specifically designed for Transformer methods. For our Transformer based student, it does provide a modest gain, and outperforms other KD methods without using external training data.

1.1 Contributions

We summarize our contributions in this thesis as follows:

- The task data is usually limited for teacher to fully express its knowledge for the student to learn. However, text augmentation is not as easy as image synthesis by rotating or adding noises. We propose two heuristic data augmentation methods for NLP: random word masking and POS-guided word replacement.
- We explore distilling the knowledge from BERT (teacher) into a simple BiLSTM-based model (student). The distilled model achieves comparable results with ELMo (a pretrained deep contextualized word representation), while using much fewer parameters and less inference time. Our results suggest that shallow BiLSTMs are more expressive for natural language tasks than previously thought.
- We propose a general framework for multi-task knowledge distillation. The student is jointly distilled across different tasks from a multi-task refined BERT model (teacher model). We evaluate our approach on Transformer-based and LSTM-based student model. Compared with previous KD methods using only data within tasks, our approach achieves better performance. In contrast to other KD methods using large-scale external text corpus, our approach balances the problem of computational resources, inference speed, performance gains and availability of training data.

1.2 Thesis Organization

The thesis is organized as follows: In Chapter 2, we go over related work and background concepts for Pretrained Language Models. In Chapter 3, we describe our first KD work which distills Task-Specific Knowledge from BERT into simple neural networks. In Chapter 4, we describe our Multi-Task Knowledge Distillation framework (MKD). Chapter 5 concludes the thesis by summarizing the main contributions and discussing potential future work.

Chapter 2

Background and Related Work

2.1 Neural Contextual Encoders

Human languages exist as a free form of text. In order to enable computational models understand and process natural language, we need to transform text into low-dimensional real-valued vectors [67, 56, 57]. We can calculate the word vectors based on its neighbors in a corpus. According to whether the word vector changes in different contexts, word embeddings can be classified as non-contextual and contextual embeddings. For non-contextual embeddings, there are multiple pre-trained word vectors, such as word2vec trained on Google News, GloVe [60] trained on Wikipedia/Gigaword/Common Crawl, and fastText[67] trained on Wikipedia/Common Crawl. However, non-contextual embeddings can hardly capture the meaning of polysemous words. For example, the word *bank* have different meanings in phases" money from the bank" and "the bank of the river". The address this issue, we need contextual embedding approaches to distinguish word semantics from its own context. Given a sentence of length T $x_1, x_2..., x_T$, the contextual representation for each token x_t is calculated as:

$$h_1, h_2, ..., h_n = f_{enc}(x_1, x_2..., x_T) \tag{2.1}$$

where f_{enc} is the neural encoder, h_t is the learnt word embeddings. I'll mainly introduce two kinds of neural contextual encoders: recurrent models and fully-connected self-attention model (Transformer). Recurrent models are strictly expressive but hard to capture long-range word dependencies, while Transformer can directly model the dependency between every pair of words and offers an advantage in training speed.

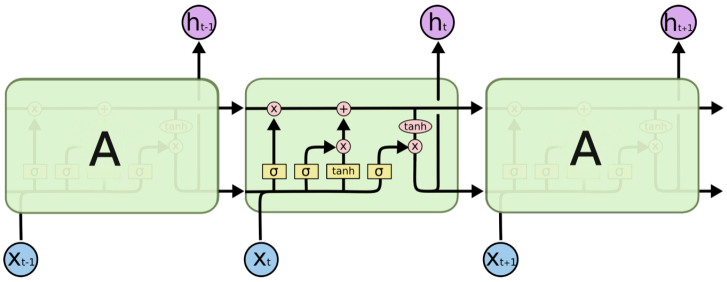

Figure 2.1: The architecture of LSTM[2]. Green boxes represent a chain of repeating modules, each repeating module contains four interacting layers. Each line carries an vector from the output of one node to the input of others. Lines merging denote vector concatenation, while lines forking denotes their vector being copied to different locations.

2.1.1 Recurrent Models

Recurrent models are often applied on sequences whose elements are dependent on each other. The model maintains hidden states (memory) at each time step to capture information about what has been calculated so far. The model uses previous outputs and current token as inputs, perform the same operation for every token in the sequence. In theory it can process arbitrary length of input sequences. In practice, during backpropagation, the gradient at each output depends on not only the calculation of the current step, but also that of all the previous steps. It will cause the vanishing/exploding gradient problems for lone-term dependencies.

Long Short Term Memory Networks (LSTM) [25] is a popular variant of RNN which is capable of learning long-term dependencies. It is designed to get around the vanishing gradient problems through gating mechanism. Figure 2.1 shows the architecture of LSTM. Different from vanilla RNN, the key to LSTM is the cell state, regulated by the structure called gates. Each gate is comprised of a sigmoid layer and a pointwise multiplication operation. There are three kinds of gates in charge of different functions: forget gate decides how much of the information from previous cell state should be kept; input gate decides what new information should be stored in the cell state; output gate filters what part of the cell state we are going to output. By learning the parameters for its gates, the model learns how the memory should behave.

Bidirectional Long Short Term Memory networks (BiLSTM) [70, 21] is an extension of LSTMs. It trains two LSTMs instead of one to model the input sequence. The first on the

[2]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

regular sequence as it is and the second on a reversed copy of the sequence. It models the context of the whole utterance, utilizing more information than a linear interpretation.

Recurrent models have been widely applied in sequence modeling problems such as language modeling and machine translation [76, 2]. However, the sequential nature of recurrent models precludes parallelization within training examples. It's critical when processing longer sequence due to memory constraints.

2.1.2 Fully-Connected Self-Attention Model

Unlike recurrent models which capture word context in a sequential order, fully-connected self-attention model breaks the limitation of recurrence and draw global dependency between input and output sequences depending on attention. Transformer [80] is a most successful and popular example of fully-connected self-attention models.

The Transformer follows the overall encoder-decoder architecture. Each Transformer layer is comprise of two sub-layers: a multi-head self-attention layer and a position wise fully connected feed-forward layer. The residual connection and layer normalization are employed around each of the sub-layers. The key to Transformer is the self-attention mechanism. For each of the input vectors, we first create a Query vector Q, Key vector K and Value vector V of dimension d_k by multiplying the corresponding embedding with three matrices. Then we calculate the self-attention using the three vectors by scoring each word against the whole sequence:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

It determines how much attention should be placed on other parts of the sequence when encoding certain word. The resulting vector is then sent to the feed-forward neural network. Instead of performing the single attention function, Transformer linearly projects queries, keys and values into different vectors with multi-head attention matrices:

$$\begin{aligned} head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \\ MultiHead(Q, K, V) &= Concat(head_1, head_2, ..., head_h)W^O \end{aligned} \tag{2.3}$$

where W^Q, W^K, W^V, W^O are learnable projection matrices. Multi-head attention expands the model's ability to focus on different positions in the sequence, and also help to project the input embeddings to various representation subspace. Self-attention could further help yield more interpretable models by inspecting the attention distributions.

Transformer contains no recurrence. In order for the model to make use of the order of the words in sequence, it adds a positional vector to each input embedding. They use sine and cosine functions since it would allow the model to easily learn to attend relative positions:

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.4}$$

where pos is the position of the token and i is dimension. The intuition is that adding these vectors provide meaningful distances between the embedding vectors when they are calculated with self-attention.

2.2 Pretrained Language Models for NLP

Substantial works have shown that pretrained language models (PLM) are extremely beneficial for downstream natural language processing tasks. They learn universal language representations from large-scale unlabeled corpora. Pre-training provides better model initialization. It not only leads to better generalization performance across all tasks, but also accelerates the convergence speed on the target task. The major differences between PLMs are contextual encoders types and pretraining tasks [63]. In terms of neural encoders, CoVe [54] and ELMo [61] capture contextual word embeddings through LSTM architecture. More recently, deep PLMs, such as OpenAI GPT [64] and BERT [17] are built with Transformer. I'll first introduce BERT in details since I use it as the teacher model, then introduce the following PLMs and pretraining tasks following the birth of BERT.

2.2.1 BERT

BERT stands for Bidirectional Encoder Representations from Transformers. It pretrains deep bidirectional representations by jointly conditioning on both left and right context in all layers. There are two steps in their framework (as shown in Figure 2.2): pre-training and fine-tuning. In order to make it fit into a variety of downstream tasks, they design a uniform input format. The input sequence is always prepended by a special token [CLS] and sentence pairs are separated by another special token [SEP]. The representation for each token is constructed by summing up its corresponding token, segment (belongs to which sequence) and position (denotes order of the sequence) embedding.

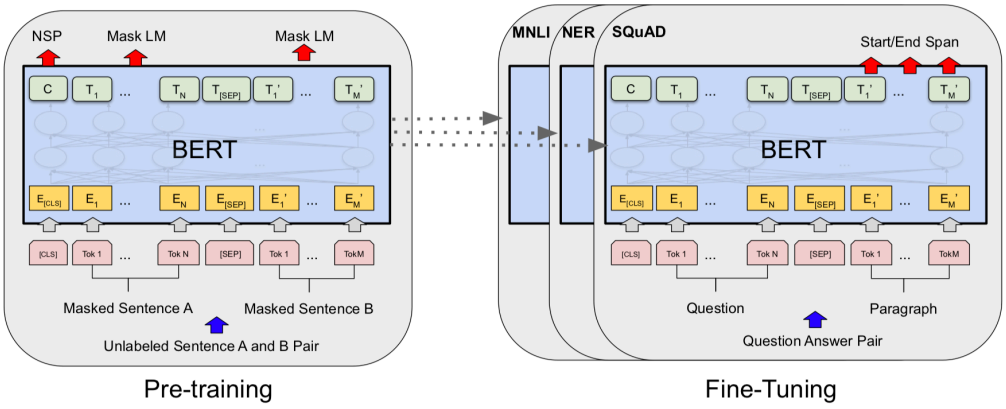

Figure 2.2: Pre-training and fine-tuning procedures for BERT [17]. For both pre-training and finetuning, they have the same layers in blue boxes, but differ in the output layers. [CLS] is added in front of the input examples, [SEP] is a special separator token.

They use two unsupervised tasks to pretrain BERT. The first is Masked Language Model (MLM). They randomly mask 15% of input tokens, and then predict those masked tokens according to both the left and right context. This task trains the deep bidirectional representations. The second is Next Sentence Prediction (NSP). Given sentence A, they select the actual next sentence that follows A in 50% probability, select random sentence from the corpus in another 50% probability. This task trains a model to understand sentence relationships. For the pre-training corpus they use the BookCorpus (800M words) [92] and English Wikipedia (2500M words).

After pretraining, we can fine-tune BERT on different downstream tasks. For each task, they add task-specific layers (usually linear layers) on top of BERT and fine-tune all the parameters end-to-end. At the output, for token-level tasks (e.g. sequence tagging), the token representations are fed into an output layer; for sequence classification task (e.g. sentiment analysis), the [CLS] representation is a proxy for the sequence representation and is fed into an output layer for prediction.

BERT has been widely used in many NLP tasks, such as question answering [87], natural language inference [28] and text summarization [50]. Recent papers have probed the information learnt by different layers of BERT [27, 10]. They show that BERT captures rich linguistic information in a hierarchical way: phrase-level information in lower layers, syntactic features in middle layers and semantic features in higher layers. Other works [62] study relational knowledge already present (without fine-tuning) in BERT. They show that BERT contains relational knowledge competitive with traditional NLP methods that

have some access to oracle knowledge.

2.2.2 Pretraining Tasks and Models

The widely used PTM pretraining tasks can be regarded as self-supervised learning. Some prominent examples and their corresponding models are listed as follows: Masked Language Modeling (MLM) masks out partial tokens in the input sequence and trains the model to predict the masked tokens. MLM is first adopted by BERT, SpanBERT [30] masked out continuous tokens, RoBERTa[51] generates masking pattern dynamically and MASS [74] adds a decoder producing masked tokens sequentially. The use of [MASK] token in MLM could leads to the problem of pretraining and fine-tuning gap. Permuted Language Modeling (PLM) is proposed by [88] in XLNet to solve this problem. Denoising autoencoder (DAE) is another pretraining task which takes a partially corrupted sequence as input and train the model to recover the original undistorted input. BART [42] proposed several text corruption methods. Contrastive learning [69] is also applied in pretraining tasks, the idea is to force the learner to distinguish similar data points from negative samples. ELECTRA applies this idea by learning a discriminator to justify if the input token is replaced or not.

2.2.3 GLUE Benchmark

Most of the pretrained language models (e.g. BERT, RoBERTa) are evaluated on the General Language Understanding Evaluation (GLUE) benchmark [81]. It is a collection of NLU tasks consisting of natural language inference, question answering, sentiment analysis and textual similarity. All the tasks are built on existing datasets and covering diverse dataset sizes and task difficulties (as shown in Table 2.1). GLUE is a public leaderboard [3] to keep track of performances of different submitted models. The groundtruth labels of test datasets for all tasks are not available to users. Users should submit their prediction results on test data to the GLUE server. The tasks can be broadly classified into three categories: 1) Inference tasks which include MNLI, QNLI, RTE and WNLI. Given a premise sentence and a hypothesis sentence, the task is to predict the relationships between the premise and hypothesis, whether it's entailmemnt, neutral or contradiction. 2) Similarity and paraphrase tasks which include QQP, MRPC and STS-B. In general, it predicts if the sentence pairs are semantically similar to each other. 3) Single sentence tasks which consist of two tasks. SST-2 predicts the sentiment of a given sentence. CoLA predicts whether the input sentence is grammatically acceptable.

[3] https://gluebenchmark.com

Dataset	Size	Metrics	Example and Label	
MNLI	Train 393K Test 20K	Matched acc. Mismatched acc.	*The Old One always comforted Ca'daan, except today.* *Ca'daan knew the Old One very well.*	**Neutral** contradiction entailment
QNLI	Train 105K Test. 5.4K	acc.	*How many alumni does Olin Business School have worldwide?* *Olin has a network of more than 16,000 alumni worldwide.*	**entailment** not_entailment
RTE	Train 2.5K Test. 3K	acc.	*Both candidates are making a major push in Iowa.* *Both candidates are delivering a great attack in Iowa.*	entailment **not_entailment**
WNLI	Train 634 Test. 146	acc.	*I put the butterfly wing on the table and it broke.* *The table broke.*	entailment **not_entailment**
QQP	Train 364K Test. 391K	acc. / F1	*How is air traffic controlled?* *How do you become an air traffic controller?*	paraphrase **non_paraphrase**
MRPC	Train 3.7K Test. 1.7K	acc. / F1	*Only Intel Corp. has a lower dividend yield.* *Only Intel's 0.3 percent yield is lower.*	**paraphrase** non-paraphrase
STS-B	Train 7K Test. 1.4K	Pearson / Spearman corr.	A *woman is sitting at a desk.* *A woman is riding a donkey.*	score[0-5] **0.4**
SST-2	Train 67K Test. 1.8K	acc.	*by far the worst movie of the year*	positive **negative**
CoLA	Train 8.5K Test. 1K	Matthews corr.	*The house destroyed John.*	acceptable **unacceptable**

Table 2.1: Statistics and examples for all tasks in GLUE. There is only one regression task STS-B, the others are single sentence or sentence pair classification.

There are two caveats about GLUE benchmark: 1) WNLI is often excluded from evaluation since GLUE webpage notes construction issues with this dataset. The train and dev contains same sentence but have opposite labels, also the test set has different label distribution than the train / dev sets. 2) There are two versions of QNLI datasets. Version 1 is expired on January 30, 2019. Users should submit the results with newer version of QNLI.

2.3 Knowledge Distillation

The goal of knowledge distillation (KD) is to train a shallow neural networks to mimic complex functions learned by the deep neural nets without sacrificing significant model quality [1, 23]. The deep neural nets in this scenario are referred to as **Teacher Model**, since it transfers its knowledge on the target dataset to the shallow neural nets, which called **Student Model**.

Unlike in computer vision domain where deeper and larger models are widely applied ever since AlexNet [38], in NLP models are relatively shallow and small. Thus in early

years there are only limited works focusing on KD in NLP. Kim et al. [33] investigate KD in the context of neural machine translation, Yu et al. [89] distill a deployable neural language model on mobile devices.

With the birth of BERT-series pretrained language models, more research begin to focus on distilling PLMs into small and shallow neural nets. To the best of our knowledge, we are the first work to explore BERT distillation [78]. Besides our work, many other efforts are also along this line: BERT-PKD [75] extracts knowledge not only from the last layer of the teacher, but also from previous layers. TinyBERT [29] introduces a two-stage learning framework which performs transformer distillation at both pretraining and task-specific stages. Zhao et al. [91] train a student model with smaller vocabulary and lower hidden states dimensions. DistilBERT [68] reduces the layers of BERT and uses this small version of BERT as its student model. All these distillation methods are specifically designed for Transformers-based students. However, we hope to take the advantage of the model-agnostic nature of KD, i.e., the choice of student model does not depend on the teacher model architecture; The teacher model can be easily switched to other powerful language models other than BERT.

Besides knowledge distillation, another prominent line of work is devoted to compressing large neural networks to accelerate inference. Early pioneering works include Lecun et al. [41], who propose a local error-based method for pruning unimportant weights. Recently, Han et al. [22] propose a simple compression pipeline, achieving 40 times reduction in model size without hurting accuracy. Unfortunately, these techniques induce irregular weight sparsity, which precludes highly optimized computation routines. Thus, others explore pruning entire filters [43, 52], with some even targeting device-centric metrics, such as floating-point operations [77] and latency [9]. Still other studies examine quantizing neural networks [84]; in the extreme, Courbariaux et al. [13] propose binarized networks with both binary weights and binary activations. Regarding to PLMs, Google recently released 24 BERT miniatures (e.g. BERT-Tiny, BERT-Mini and BERT-Medium) by cutting down hidden state dimensions and Transformer layers[4].

[4]https://github.com/google-research/bert/

Chapter 3

Distilling BERT into Simple Neural Networks

In this chapter, we propose a simple yet effective approach that transfers task-specific knowledge from BERT to a shallow neural architecture — in particular, a bidirectional long short-term memory network (BiLSTM). Recently, deeper and larger models [17, 42, 65, 6] greatly improve the state of the art on many tasks. We are interested in investigating whether a simple architecture actually lacks representation power for text modeling, and wish to study effective approaches to transfer knowledge from BERT to a BiLSTM.

3.1 Model Architecture

3.1.1 Teacher Model

We use the pretrained, fine-tuned BERT [17] (described in 2.2.1) as our teacher model. BERT computes a feature vector $h \in \mathbb{R}^d$ for a given input sentence (pair). We then directly build a classifier upon h for the task. During training, we jointly fine-tune the parameters of BERT and the classifier by maximizing the probability of the correct label, using the cross-entropy loss.

3.1.2 Student Model

The student model is a single-layer BiLSTM with a non-linear classifier. We design separate model architectures for single-sentence tasks (figure 3.1) and sentence-pair tasks (figure 3.2). We restrict the architecture engineering to a minimum to revisit the representation power of BiLSTM itself, avoiding additional components such as attention and layer normalization.

Single-Sentence Tasks Given a sequence of n tokens, let $w = (w_1, w_2..., w_n)$ represent the word embedding vectors in the sequence. We first feed the input word embeddings into the BiLSTM for context modeling. BiLSTMs consist of two LSTMs that run in parallel in opposite directions: one on the input sequence and the other on the reverse of the sequence. The concatenation of the hidden states of the last step in each direction are then fed to a fully connected layer with rectified linear units (ReLUs).

$$\begin{aligned} h_s &= \text{BiLSTM}(w) \\ h'_s &= \text{RELU}(Wh_s + b) \end{aligned} \tag{3.1}$$

where h_s represents the concatenated last step hidden states in both directions. The output h'_s is passed to a fully connected layer and then Softmax layer for classification.

Sentence-Pair Tasks Given a pair of sentences, we share BiLSTM weights in a siamese architecture between the two sentence encoders. Siamese networks [5, 36] are neural networks containing two or more identical subnetwork components. The two input sequences are encoded as h_{s1} and h_{s2} through their respective BiLSTM. We then apply a standard concatenate–compare operation between the two sentence vectors:

$$\begin{aligned} h_s &= [h_{s1}, h_{s2}, h_{s1} \odot h_{s2}, |h_{s1} - h_{s2}|] \\ h'_s &= \text{RELU}(Wh_s + b) \end{aligned} \tag{3.2}$$

where $\odot$ denotes elementwise multiplication. We feed the output h_s to a RELU activated classifier.

3.2 Distillation Objective

The student network learns to mimic a teacher network's behaviour given any data point at the output level. The discrete probability output of a neural network is calculated by the softmax layer:

$$y_i = \text{softmax}(z) = \frac{exp\{w_i^\intercal h\}}{\sum_j exp\{w_j^\intercal h\}} \tag{3.3}$$

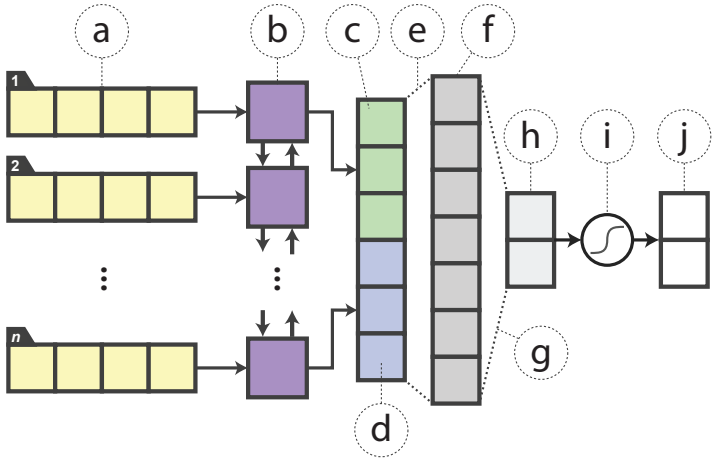

Figure 3.1: The BiLSTM model for single-sentence classification. The labels are (a) input embeddings, (b) BiLSTM, (c, d) backward and forward hidden states, respectively, (e, g) fully-connected layer; (e) with ReLU, (f) hidden representation, (h) logit outputs, (i) softmax activation, and (j) final probabilities.

where w_i denotes the i_{th} row of softmax weight W, and z is equivalent to $w^\intercal h$. z is also called logit, before the softmax activation. In addition to one-hot predicted labels, training on logarithms of predicted probabilities makes learning easier for the student model [1]. The relationship learned by the teacher model across all of the targets are equally emphasized. For example, in paraphrase identification. some sentence pairs are obviously distinct from each other, whereas others appear neutral. If we use only the teacher's predicted one-hot label to train the student, we may lose valuable information about the prediction uncertainty.

The distillation objective is to minimize the mean-squared-error (MSE) loss between the student network's logits against the teacher's logits:

$$\mathcal{L}_{\text{distill}} = ||\boldsymbol{z}^{(B)} - \boldsymbol{z}^{(S)}||_2^2 \tag{3.4}$$

where $z^{(B)}$ and $z^{(S)}$ are the teacher's and student's logits, respectively. It should be emphasized that it is inappropriate to penalize MSE loss with predicted probabilities, because a probability is always in the range of $[0, 1]$, and thus MSE could be insensitive. By contrast, the range of a logit could be the entire real space, and thus MSE loss is much stronger if the gap between the student and the teacher is large. Also, the teacher model's logit provides more information than its predicted one-hot label. Another example in sentiment

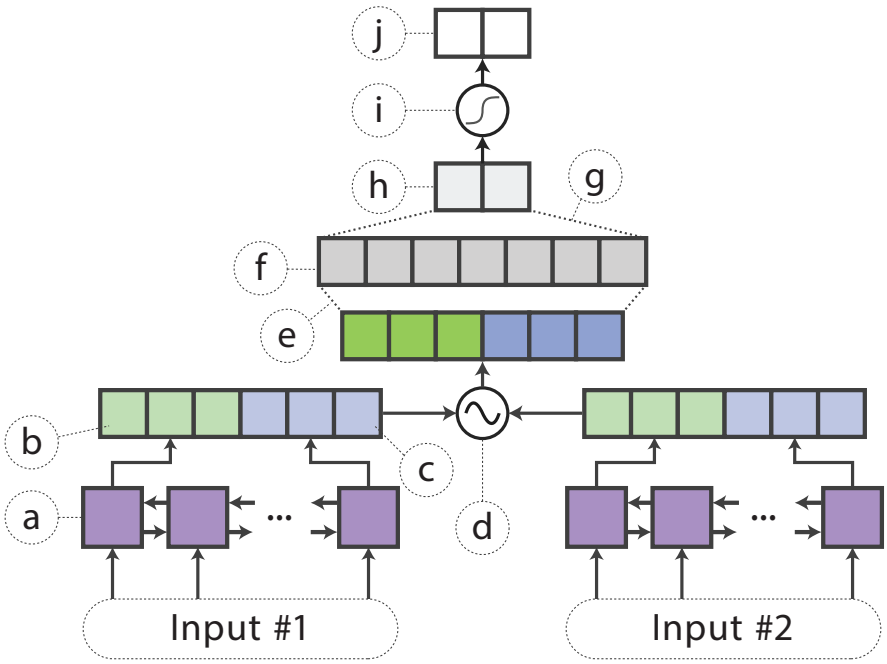

Figure 3.2: The siamese BiLSTM model for sentence matching, with shared encoder weights for both sentences. The labels are (a) BiLSTM, (b, c) final backward and forward hidden states, respectively, (d) concatenate–compare unit, (e, g) fully connected layer; (e) with ReLU, (f) hidden representation, (h) logit outputs, (i) softmax activation, and (j) final probabilities.

analysis, a more positive sentence could yield a larger logit value than a less positive one, although both may belong to the "positive" category.

Hinton et al. [23] proposes an alternative way to distill by penalizing the cross entropy loss with the teacher's and the student's predicted probabilities [23], given by $\mathcal{L}_{\text{distill}}^{(\text{CE})} = -\sum_i y_i^{(B)} \log y_i^{(S)}$. Directly using the teacher's predicted probabilities, however, might also be less effective, as the y is likely to peak in a certain category, strongly resembling a one-hot distribution. Therefore, Hinton et al [23] introduce a temperature T when computing the probability: $\tilde{y}_i = \frac{\exp\{z_i/T\}}{\sum_i \exp\{z_j/T\}}$, and their distilling objective becomes the cross-entropy against the softened probabilities (with $T > 1$). However, in our preliminary experiments, we found that this alternative works slightly worse than matching logits.

It might be intuitive to use the distilling objective in conjunction with a traditional cross-entropy loss against the one-hot ground-truth label as in [1]. In our preliminary experiments, this lead to no further improvement.

3.3 Data Augmentation for Distillation

In the distillation approach, a small dataset may not suffice for the teacher model to fully express its knowledge [1]. Therefore, we augment the training set with a large, unlabeled dataset, with pseudo-labels provided by the teacher, to aid in effective knowledge distillation.

However, unlike in computer vision, data augmentation in NLP is usually more difficult. First, there exist a large number of homologous images in computer vision tasks. For example, CIFAR-10 is a subset of the 80 million tiny images dataset [37]. Second, it is easy to synthesize a near-natural image by rotating, adding noise, and other distortions. But if we manually manipulate a natural language sentence, the sentence may not be fluent, and its effect in NLP data augmentation less clear.

In this work, we propose a set of heuristics for task-agnostic data augmentation: we use the original sentences in the small dataset as blueprints, and then modify them with our heuristics, a process analogous to image distortion. Specifically, we randomly perform the following operations.

Masking. We randomly replace a word in the sentence with [MASK] with probability p_{mask}. The [MASK] corresponds to an unknown token in our models and the masked word token in BERT. Intuitively, this rule helps to clarify the contribution of each word towards

the label, e.g., the teacher network produces less confident logits for "I [MASK] the movie" than for "I loved the movie."

POS-guided word replacement. With probability p_{pos}, we replace a word with another of the same POS tag. In order to preserve the original training distribution, the new word is sampled from the unigram word distribution re-normalized by the part-of-speech (POS) tag. This rule aims to slightly alter the semantics of each example, e.g., "What does fox say?" is different from "Where does fox say?".

The data augmentation procedure is as follows: given a training example $w = (w_1, w_2..., w_n)$, we draw from the uniform distribution $X_i \sim \text{UNIFORM}[0, 1]$ to sample a probability X_i for each word w_i. If $X_i < p_{mask}$, we apply masking to w_i. If $p_{mask} \in [p_{mask}, p_{mask} + p_{pos})$,we apply POS-guided word replacement. These two operations are mutually exclusive: once one rule is applied, the other is disregarded. Both p_{mask} and p_{pos} are set to 0.1. With the remaining probability we do not perform modification on this word. The final synthetic example is appended to the augmented, unlabeled dataset.

For single sentence datasets, we apply this procedure n_{iter} times per example to generate up to n_{iter} samples from a single exam- ple, with any duplicates discarded. For sentence-pair datasets, we cycle through augmenting the first sentence only while holding the second fixed, the second sentence only while holding the first fixed, and both sentences.

3.4 Experimental Setup

3.4.1 Implementation Details

Suppose the number of Transformer layers are denoted as L, the hidden size H, the number of self-attention heads A. [17] introduces two BERT variants: $\text{BERT}_{\text{BASE}}$(L=12, H=768, A=12, Total Parameters=110M) and $\text{BERT}_{\text{LARGE}}$(L=24, H=1024, A=16, Total Parameters=340M). In our experiments, we use $\text{BERT}_{\text{LARGE}}$ as our teacher model. When adapting to a specific task, we take the pretrained model hyperparameters and fine-tune both the BERT parameters and top task-specific parameters. Same as [17], we use the Adam optimizer with learning rates chosen from $\{2, 3, 4, 5\} \times 10^{-5}$ according to the development set loss on each task. We only use the original training data for each task without the data augmentation.

For the student BiLSTM model, we choose the number of BiLSTM hidden units from $\{150, 300\}$, the dimension of the last hidden layer from $\{200, 400\}$, depending on the development set performance on each task. Following [32], we use the multichannel embedding

technique with 300-dimensional word2vec trained on Google News. For optimization, we use AdaDelta [90] with its default learning rate of 1.0 and $\rho = 0.95$. For SST-2, STS-B, and MRPC, we use a batch size of 50; for MNLI and QQP, due to their larger size, we choose 256 for the batch size.

For our dataset augmentation hyperparameters, we fix $p_{\text{mask}} = p_{\text{pos}} = 0.1$ across all datasets. In our preliminary experiments, these values are not sensitive. The size of the augmented dataset is 40 times the size of the original corpus for the smaller datasets STS-B and MRPC, and 10 times for the larger datasets MNLI, QQP and SST-2. The original corpus is also included.

3.4.2 Datasets

The General Language Understanding Evaluation (GLUE) [81] benchmark is a collection of NLU tasks including question answering, sentiment analysis, and textual entailment. We conduct our experiments on five most widely used datasets: SST-2, QQP, STS-B, MRPC and MNLI. Details about each dataset are described in section 2.2.3.

3.4.3 Baseline Models

BERT [17] is a multi-layer, bi-directional Transformer encoder that comes in two variant: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. We use $\text{BERT}_{\text{LARGE}}$ as our teacher model and list $\text{BERT}_{\text{BASE}}$ as a competing method.

OpenAI GPT [64] is, like BERT, a generative pretrained transformer (GPT) encoder fine-tuned on downstream tasks. Unlike BERT, however, GPT is unidirectional and only makes use of previous context at each time step. They first train a transformer encoder on a very large corpus using language modeling as a training signal, and then fine-tune this model on a much smaller dataset of specific tasks.

ELMo [61] is a deep contextualized word representation that models both complex characteristics of word use (e.g., syntax and semantics), and the various of their use under different contexts. ELMo word representations are computed on top of a deep bidirectional language model (biLM). They are pre-trained on a large text corpus. The pre-trained word representations can be easily added to existing models and significantly improve the performance of various downstream tasks. In GLUE paper, [81] provide a BiLSTM-based model baseline trained on top of ELMo and jointly fine-tuned across *all* tasks. This model contains 4096 units in the ELMo BiLSTM and more than 93 million total parameters.

#	Model	SST-2	QQP	STS-B	MRPC	MNLI-m	MNLI-mm
		Acc	F_1/Acc	Acc	Acc	r/ρ	F_1/Acc
1	$BERT_{LARGE}$ [17]	94.9	72.1/89.3	87.6/86.5	89.3/85.4	86.7	85.9
2	$BERT_{BASE}$ [17]	93.5	71.2/89.2	87.1/85.8	88.9/84.8	84.6	83.4
3	OpenAI GPT [64]	91.3	70.3/88.5	82.0/80.0	82.3/75.7	82.1	81.4
4	ELMo (reported by GLUE)	90.4	63.1/84.3	74.2/72.3	84.4/78.0	74.1	74.5
5	Distilled BiLSTM	**91.6**	**68.5/88.4**	**79.6/78.2**	**82.7/75.6**	**72.5**	**72.4**
6	BiLSTM (our implementation)	86.7	63.7/86.2	66.9/64.3	80.9/69.4	68.7	68.3
7	BiLSTM (reported by GLUE)	85.9	61.4/81.7	66.0/62.8	79.4/69.3	70.3	70.8

Table 3.1: Test results on different datasets. All of our test results are obtained from the GLUE benchmark website. We cannot compare with GPT-2, as they have neither released the full model nor provided any results of these datasets in their paper.

3.5 Results and Analyses

In this section, we first present the model quality on different datasets (in Table 3.1), and compare inference efficiency of our distilled BiLSTM with other models (in Table 4.4). Then, we provide detailed model analysis and a case study.

3.5.1 Model Quality

We first show that our implementation of student architecture is fair. Line 7 reports BiLSTM results evaluated on all datasets from the GLUE platform. We train a base BiLSTM model on the original labeled set, without using distillation and report results in Line 6. The scores of Line 6 are comparable with that of Line 7, suggesting that our implementation is fair, and that we're ready to conduct the distillation research. It's also evident that BiLSTM performs much worse than a pretrained models. For exmaple, the gap between the BiLSTM and BERT is around 10 percentage points in all experiments.

We then apply our distillation approach on both the original training set and augmented samples. The results show that: 1)In general, *Distilled BiLSTM* achieves an improvement of 1.8-4.9 points compared to the base BiLSTM without distillation. 2) Compared with ELMo, *distilled BiLSTM* outperforms the best reported ELMo (Line 4) on SST-2, STS-B, and QQP. On STS-B, we achieve higher results than ELMo by 5 points in Pearson's r, a 10-point improvement over original BiLSTM. On MNLI and MRPC, our results are

Model		$\text{BERT}_{\text{LARGE}}$	ELMo	**Distilled BiLSTM**
SST-2	# of Par.	335(349×)	93.6(98×)	0.96(1×)
	Inf. Time	1060(434×)	36.71(15×)	2.44(1×)
QQP	# of Par.	335(211×)	93.6(59×)	1.59(1×)
	Inf. Time	5766(185×)	256(8×)	31.14(1×)

Table 3.2: Model size and inference speed on SST-2 (single sentence task, 67k) and QQP (sentence pairs task, 363k) training set. # of Par. denotes number of millions of parameters, and inference time is in seconds.

slightly worse than ELMo's by 1.6–2.4 points; however, they still show improvement of 1.8–6.2 points against our BiLSTM. 3) Interestingly, the MRPC result is even better than the Open AI GPT model (Line 3) in F_1-measure.

To the best of our knowledge, this work is the first to achieve such high performance with a very lightweight, single-layer BiLSTM; in this sense, the distilled BiLSTM is the state-of-the-art "small" model on the GLUE benchmark.

3.5.2 Inference Efficiency

In this part, we provide a quantitative analysis of inference-time efficiency. We use the open-source PyTorch implementations for BERT[1] and ELMo.[2] The model inference is performed on a single NVIDIA V100 GPU with a batch size of 512. Since the inference time depends only on the network architecture, but not the task itself, we report results on the SST-2 and QQP, which contains 67,350 sentences and 363,850 sentence pairs, respectively.

As shown in Table 3.2, our model Distilled LSTM is 15 and 434 times faster in inference time than ELMo and $\text{BERT}_{\text{LARGE}}$ for single-sentence tasks, while using 98 and 349 times fewer parameters, respectively. Moreover, for sentence-pair tasks, Distilled LSTM is 8 and 185 times faster than ELMo and $\text{BERT}_{\text{LARGE}}$, while using 59 and 211 times fewer parameters, respectively.

3.5.3 Model Analysis

Distilling Approach. We conduct an ablation study to verify the effect of distilling

[1] https://goo.gl/iRPhjP
[2] https://allennlp.org/elmo

Method	SST-2 (Acc)		QQP (F1/Acc)	
	Dev	Test	Dev	Test
No Aug. (label)	85.43	86.2	81.44/86.05	64.6/85.6
No Aug. (logit)	87.50	88.9	83.87/88.26	66.3/87.3
Aug (Mask only)	88.76	89.8	84.61/88.75	67.7/88.0
Aug (Mask + POS)	**90.25**	**91.6**	**84.68/88.94**	**68.5/88.4**

Table 3.3: Ablation study. We compare the masking and POS-guided augmentation techniques, as well as no data augmentation. In the augmentation settings, we control the number of samples to be 10 times the size of original corpus.

objective and data augmentation heuristics. We choose one single-sentence input task SST and one sentence-pair input task QQP as the testbeds.

In Table 3.3, we first compare the logits matching distilliong objectve with the traditional one-hot cross-entropy loss the originial training set (without augmented samples). When distilling with a labeled dataset, the one-hot target is simply the ground-truth label. Results of first two lines show that using matching logits leads to consistent improvement. This confirms our hypothesis in Section 3.2 that logits reflect more information than one-hot labels. And the distilling objective effectively aids the model to transfer knowledge from teacher (BERT) to student (a small BiLSTM).

We then compare the augmentation techniques in the following two lines of the table. Comparing with the above lines, data augmentation is an indispensable component in the distillation approach. Masking technique contributes around one point on the overall performance, POS-guided replacement improves on SST-2 dataset by another point, but not clear improvement on QQP dataset.

We conclude that both the distilling objective and data augmentation technique play a role in the distilling process. The approach is valid and effective in transferring knowledge from teacher to student.

Effect of the Size of Augmented Samples. The left part of figure 3.3 plots the model performance distilled with different size of augmented samples. The x-axis indicates the augmentation dataset is in multiples of the original dataset's size. When feeding more data, the model performance increases gradually. The improvement gradually saturates when the augmented dataset is 10 times as the original corpus.

Effect of BiLSTM Size. As we desire that the student model to be shallow and small, we further investigate how small the student can be. In right part of figure 3.3, the x-axis is

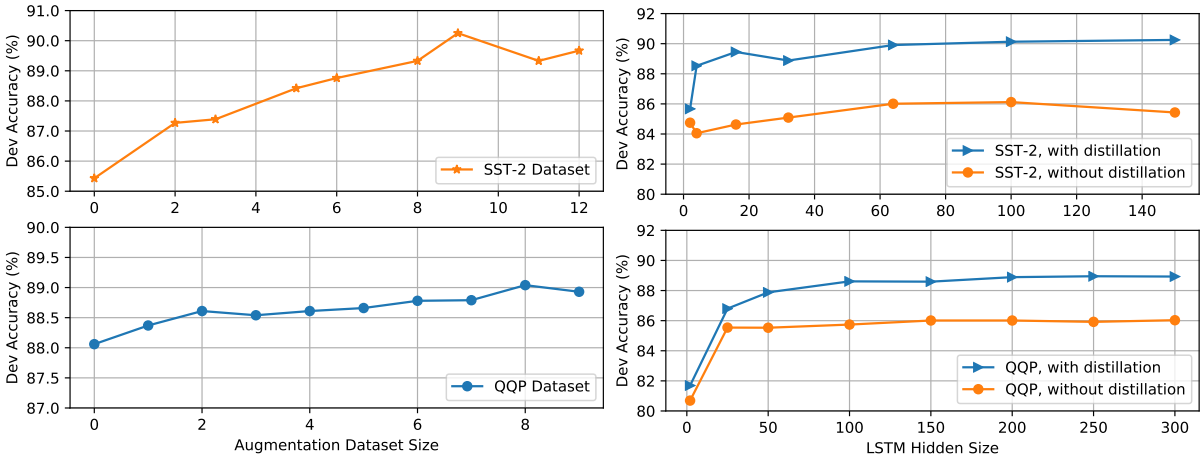

Figure 3.3: **Left:** Development set accuracy compared to the size of the augmented dataset, expressed in multiples of the size of the original corpus. **Right:** Development set accuracy compared to the number of BiLSTM hidden units. In this analysis, the smallest hidden size of the BiLSTM is two.

the varying size of BiLSTM hidden units with minimum two in the experiment. We present both the distilled BiLSTM results and regularly trained BiLSTM without distillation.

Although there exists clear gap between distilling approach (blue line) and BiLSTM trained alone (yellow line), they generally have the same trend that the performance is better with more hidden units. An interesting observation is that SST-2 only requires very few hidden units for high accuracy. However, this is not observed on QQP dataset. The potential reasons could be: SST-2 dataset is much smaller than the QQP dataset, and the SST-2 sentiment analysis task is relatively easy. Therefore, a very small network suffices to achieve high performance if it is trained with BERT as a teacher.

3.5.4 Case Study

We conduct a case study on the sentiment analysis task to examine the robustness of the model. We mask out each word in the sentence and see if model could detect the consequent lack of sentiment of the sentence. From the development set, we select two sample sentences which only have a single sentiment word. For example, in sentence "my reaction in a word : disappointment ." We mask out the word *disappointment* and feed the ablated sentence "my reaction in a word : [MASK] ." into the model. Here we mainly examine three models: the teacher model BERT, the distilled BiLSTM and non-distilled

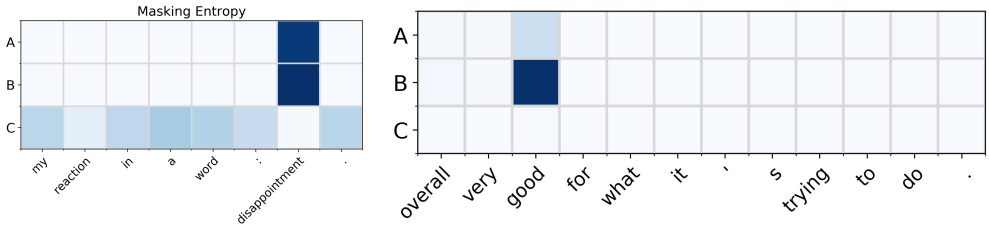

Figure 3.4: Illustration of the importance of the masked word to each model to make sentiment predictions, where the x-axis denotes masked words. Darker colors denote higher values; white is zero. The model labels are as follows: (A) BERT, (B) distilled BiLSTM, and (C) non-distilled BiLSTM.

BiLSTM.

We then compute the associated entropy for the probability output. This metric measures the degree of uncertainty, i.e., the lower the entropy, the more certain the model is. Ideally, model should predict high entropy for the ablated sentences since they lack sentiment key word being neither positive nor negative.

In Figure 3.4, we illustrate the model output of two example sentences. Each word on x-axis indicates the masked-out word in the sentence. The corresponding cell is colored according to the magnitude of the entropy associated with the masked sentence. BERT and the distilled BiLSTM can accurately characterize the lack of sentiment on both examples. When *disappointed* and *good* are masked, the model entropy increases substantially. However, non-distilled BiLSTM fails and inaccurately yielding a high-confidence prediction. These results show that through knowledge distillation, the student distilled BiLSTM successfully learns knowledge from BERT and it's more robust than non-distilled BiLSTM.

Chapter 4

A Multi-Task Knowledge Distillation Approach

In the previous chapter, we explore distilling knowledge from BERT into a small, single-layer BiLSTM. The distilled model achieves comparable results to ELMo, while having much fewer parameters and less inference time. These results suggest that small BiLSTMs are more expressive on natural language tasks than previously thought. However, can we further maximize the expressive capacity of the small student model?

Previous methods focus on task-specific KD, which transfer knowledge from a single-task teacher to its single-task student. We propose to distill the student model from different tasks jointly, so that the student could learn a more universal language representation by leveraging cross-task data. Multiple training objectives from different tasks serve as a form of regularization, discouraging the student model from overfitting to a specific task. The overall framework is illustrated in Figure 4.1. The left figure shows the task-specific KD, the knowledge distillation process needs to be conducted all over again when performing on a new NLP task. In the right figure, the student model is distilled from all tasks together into shared layers. For a specific task it only needs to further fine-tune the shared layers together with its task-specific layer.

Since we already examine the representation capacity of a simple, single-layer BiLSTM only as a student, we are interested in whether adding more previous effective modules, such as attention mechanism, will further improve its effectiveness. So we use the LSTM based network with bi-attention mechanism as the student model. Since most of the KD works [75, 29] use Transformer as their students, we also evaluate the KD approach on a three-layer Transformer. The LSTM based student and Transformer based student exemplify

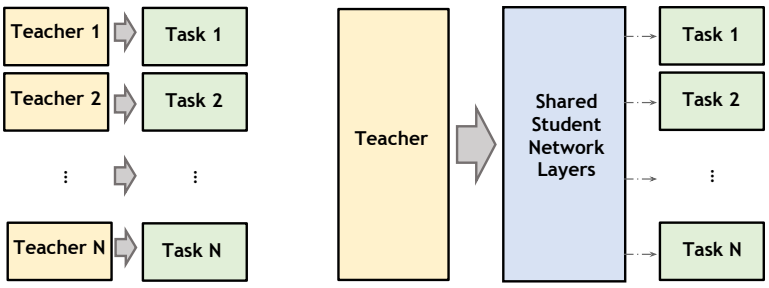

Figure 4.1: The left figure represents task-specific KD. The distillation process needs to be performed for each different task. The right figure represents our proposed multi-task KD. The student model consists of shared layers and task-specific layers.

that our approach is model agnostic, the choice of student model does not depend on the teacher model architectiure.

We conduct experiments on seven datasets across four different tasks. For LSTM based student, our approach keeps the advantage of inference speed while maintaining comparable performances as those specifically designed for Transformer methods. For our Transformer based student, it does provide a modest gain, and outperforms other KD methods without using external training data. We also study several important problems in KD in ablation studies, such as the influence of different tokenization methods and the influence of MTL in KD.

4.1 Model Architecture

In this section, we introduce the teacher model and student model for our distillation approach. For student, we explore two different network architecture: a traditional bidirectional long short-term memorynetwork (BiLSTM) with bi-attention mechanism, and the popular Transformer.

4.1.1 Multi-Task Refined Teacher Model

Multi-task learning learn multiple task jointly so that the knowledge learned in one task can benefit others. It also leverages the regularization of different tasks via alleviating overfitting to a specific task. Language models under this setting can be more effective

in learning universal language representations. Given this consideration, We employ the bidirectional transformer language mode (BERT) as bottom shared text encoding layers, and fine-tune the task-specific top layers for each type of NLU task. There are mainly two stages for the training procedure: pretraining the shared layer and multi-task refining.

Shared layer pretraining. Following Devlin et al. [17], input tokens are first encoded as summation of their corresponding token embeddings, segmentation embeddings (learned embeddings indicating the token belongs to which sentence) and position embeddings (information relative or absolute position of tokens in the sequence). The input embeddings are then mapped into contextual embeddings C through a multi-layer bidirectional transformer encoder. The pretraining of these shared layers use the cloze task and next sentence prediction task. We use the pretrained $\text{BERT}_{\text{LARGE}}$ to initialize these shared layers.

Multi-task refining. The contextual embeddings C output from the shared pre-trained text encoding layers are then fed into the above task-specific layers. Following Liu et al. [49], we classify the natural language understanding tasks on GLUE [81] into four categories: single-sentence classification (CoLA and SST-2), pairwise text classification (RTE, MNLI, WNLI, QQP, and MRPC), pairwise text similarity (STS-B), and relevance ranking (QNLI). Each category corresponds to its own output layer.

Here we take the single-sentence classification task and text similarity task as examples to demonstrate the implementation details. The process for other two categories of tasks are similar to these two, interested readers can see more details in [49]. For single-sentence classification, the contextual embedding of the special token [CLS] can be viewed as the semantic representation of the input sentence X. We use a logistic regression with softmax to predict the probability that X is labeled as class c:

$$P(c|X) = softmax(W_c^\top \cdot x) \tag{4.1}$$

where W_c is the task-specific parameter matrix. For text similarity task. For text similarity task, the input sentence pair (X_1, X_2) is still represented as the contextual embedding of the `[CLS]` token. The similarity score is predicted by the similarity ranking layer:

$$\text{Sim}(X_1, X_2) = W_{sim}^\top x \tag{4.2}$$

where W_{sim} is a task-specific learnable parameter matrix.

In the multi-task refining stage, all the model parameters, including bottom shared layers and task-specific layers, are updated through mini-batch stochastic gradient descent [44]. The training samples of each task is first packed into mini-batches, the collection of all the mini-batches from all tasks is the training data. Running all the mini-batches in each

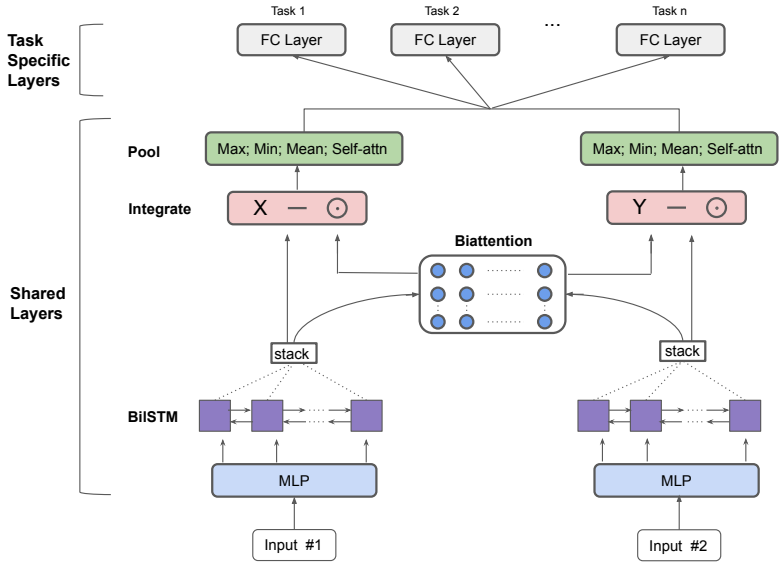

Figure 4.2: Architecture for the bi-attentive student neural network.

epoch approximately optimizes the sum all of all multi-task objectives. In each epoch, the model is updated according to the selected mini-batch and its task-specific objective. we still take the text similarity task as an example, where each pair of sentences is labeled with a real-value similarity score y. We use the mean-squared error loss as objective function:

$$\|y - \text{Sim}(X_1, X_2)\|_2^2 \tag{4.3}$$

For text classification task, we use the cross-entropy loss as the objective:

$$-\sum_c \mathbb{1}(X, c) log(P(c|X)) \tag{4.4}$$

The tasks are not limited to the presented categories, new task can be easily added to the current models by simply adding its own task-specific layer and refine the model together with all the tasks.

4.1.2 LSTM-based Student Model

We are interested in exploring whether a simple architecture such as LSTM, plus additional modules such as attention mechanism, has stronger representation capacity to transfer

knowledge from teacher network. We incorporate bi-attention module since it's widely used between sentence-pair tasks [61, 81]. And the inputs in most our experiments are two sentences. The overall architecture is shown in figure 4.2.

For equation representations, the embedding vectors of input sequences are denoted as w_x and w_y. For single input task, the input sequence is duplicated to form two sequences, w_y is the same as w_x in this case. So I'll assume the input is a pair of sequences in the following descriptions. $\oplus$ represents vectors concatenation.

The input sequence w_x and w_y are first converted into $\hat{w}^x$ and $\hat{w}^y$ through a feedforward network with ReLU activation [58] function. For each token in $\hat{w}^x$ and $\hat{w}^y$, we then use a bi-directional LSTM encoder to compute its hidden states at each step. We stack the hidden states of all steps to form matrices X and Y separately.

$$x = \text{BiLSTM}(\hat{w}^x) \quad y = \text{BiLSTM}(\hat{w}^y) \tag{4.5}$$

$$X = [x_1; x_2; ...; x_n] \quad Y = [y_1; y_2; ...; y_m] \tag{4.6}$$

where n and m are the length of each sequence. Next, we apply the biattention mechanism [86, 71] to compute the attention contexts $A = XY^\top$. The attention weight A_x and A_y is extracted through a column-wise normalization for each sequence:

$$A_x = \text{softmax}(A) \quad A_y = \text{softmax}(A^\top) \tag{4.7}$$

Then we multiply the representation of each token with attention weight to compute the context vectors. The context vector of one sequence condition on the other.

$$C_x = A_x^\top X \quad C_y = A_y^\top Y \tag{4.8}$$

Same as [54], we reinforce the relationship between the original representation and context vector through three different computations: the original representation itself (to ensure conditioning on complete information), the difference from the context vector, and the element-wise product. Another BiLSTM layer is applied on the concatenation of these three computation results.

$$\begin{aligned} X_y &= \text{BiLSTM}([X \oplus X - C_y \oplus X \odot C_y]) \\ Y_x &= \text{BiLSTM}([Y \oplus Y - C_x \oplus Y \odot C_x]) \end{aligned} \tag{4.9}$$

In order to get the final sentence representation, we aggregate the output hidden states along the time step by pooling operation. Besides the regular max, mean pooling. We also

use the self-attentive pooling to extract features. The self-attentive pooling x_{self} and y_{self} are weighted summation of each sequence:

$$\begin{aligned} a_x &= \text{softmax}(X_y v_1 + d_1) \\ a_y &= \text{softmax}(Y_x v_2 + d_2) \end{aligned} \tag{4.10}$$

$$x_{self} = X_y^\top a_x \quad y_{self} = Y_x^\top a_y \tag{4.11}$$

The max, min, mean, and self-attentive pooled representations are then concatenated to get one context representation:

$$\begin{aligned} x_{pool} &= [max(X_y), ; mean(X_y); x_{self}] \\ y_{pool} &= [max(Y_x), ; mean(Y_x); y_{self}] \end{aligned} \tag{4.12}$$

We feed this context representation through a fully-connected layer to get final output.

4.1.3 Transformer-based Student Model

The pre-trained language models [17, 51], which can be employed as teacher, are mostly built with Transformers. Most of the KD works [75, 29] use small-sized or shallow layers Transformers as their students. We have described Transformer in detail in section 2.1.2. It draws global dependencies between input and output entirely relying on an self-attention mechanism. We use three layers of Transformers in student model. Same as BERT [17], [CLS] is added in front of every input example, and [SEP] is added between two input sentences. We apply average-pooling on the [CLS] representations of all layers as the final output.

4.2 Multi-task Distillation

We described the distillation objectives in section 3.2, which minimize the mean-squared error (MSE) between the student network's logits and the teacher's logits. Considering one text classification problem, denoted as task t, a softmax layer will perform the following operations on the i^{th} dimension of z to get the predicted probability for the i^{th} class, and the distillation objective $L^t_{distill}$ is:

$$\text{softmax}(z_i^t) = \frac{\exp\{z_i^t\}}{\sum_j \exp\{z_j^t\}} \tag{4.13}$$

$$L^t_{distill} = \|z^t_T - z^t_S\|^2_2 \tag{4.14}$$

Similar to teacher model, the parameters of student model, introduced in Section 4.1.2 and 4.1.3, are shared across all tasks. To distill the student model from different tasks jointly, each task has its individual layer on top of it. For each task, the hidden representations learnt from the shared layers are first fed to a fully connected layer with rectified linear units (ReLU). Its outputs are then passed to another linear transformation to get logits $z = Wh$. During multi-task distillation, the parameters from both the bottom shared layers and upper task-specific layers are jointly updated.

The training samples are collected from each dataset and packed into task-specific batches. For task t, we denote the current selected batch as b_t. For each epoch, the model runs through all the batches which is equal to attend over all the tasks:

$$L_{distill} = L^1_{distill} + L^2_{distill} + ... + L^t_{distill} \tag{4.15}$$

We first train the teacher model under multi-task setting. The teacher model first uses the pretrained BERT model [17] to initialize its parameters of shared layers. It then follows the multi-task refining procedure described in Section 4.1.1 to update both the bottom shared-layers and upper task-specific layers.

The next step is performing the multi-task distillation. For each batch, the MTL refined teacher model first predicts teacher logits for the training samples. The student model then updates both the bottom shared layer and the upper task-specific layers according to the training signals of teacher logits. The complete procedure is summarized in Algorithm 1.

4.3 Experimental Setup

4.3.1 Datasets

We conduct the experiments on seven most widely used datasets in the General Language Understanding Evaluation (GLUE) benchmark [81]: one sentiment dataset SST-2 [73], two paraphrase identification datasets QQP and MRPC [18], one text similarity dataset STS-B [8], and three natural language inference datasets MNLI [82], QNLI [66] and RTE. For the QNLI dataset, version 1 expired on January 30, 2019; the result is evaluated on QNLI version 2. Details of the datasets have been described in Section 2.2.3.

Algorithm 1 Multi-task Distillation

Initialize the shared layers with PLM parameters, multi-task refine the teacher model
Initialize the student model parameters
Set the max number of epoch: $epoch_{max}$
// Pack the data for T Tasks into batches
for $t \leftarrow 1$ to T **do**
    1. Generate augmented data: $\mathrm{t_{aug}}$
    2. Pack the dataset t and $\mathrm{t_{aug}}$ into batch D_t
end for
// Train the student model
for *epoch* $\leftarrow 1$ to $epoch_{max}$ **do**
    1. Merge all datasets:
       $D = D_1 \cup D_2 \ ... \ \cup D_T$
    2. Shuffle D
    for b_t in D **do**
        3. Predict logits z^T from teacher model
        4. Predict logits z^S from student model
        5. Compute loss $L_{distill}(\theta)$
        6. Update student model:
           $\theta = \theta - \alpha \nabla_\theta L_{distill}$
    end for
end for

4.3.2 Implementation Details

Teacher Model. We use the released MT-DNN model[1] to initialize teacher model. We further refine the model against the multi-task learning objective for one epoch with learning rate set to 5e-4. The performance of our refined MT-DNN is lower than reported results in [49].

Student Model. The LSTM based student model (MKD-LSTM) is initialized randomly. For multi-task distillation, we use the Adam optimizer [34] with learning rates of 5e-4. The batch size is set to 128, and the maximum epoch is 16. We clip the gradient norm within 1 to avoid gradient exploding. The number of BiLSTM hidden units in student model are all set to 256. The output feature size of task-specific linear layers is 512. The Transformer-based student model (MKD-Transformer) consists of three layers of Transformers. Following the settings of BERT-PKD [75], it is initialized with the first three layers parameters from pre-trained BERT-base.

For further performance improvement, we fine-tune the student model for each task after MTL distilled. During fine-tuning, the parameters of both shared layers and upper task-specific layers are updated. The learning rate is chosen from $\{1, 1.5, 5\} \times 10^{-5}$ according to the validation set loss on each task. Other parameters remain the same as above. For both teacher and student models, we use WordPiece embeddings [85] with a 30522 token vocabulary.

Data Augmentation. As stated in Section 3.3, it's desirable to augment original training corpus so that the student can better learn from teacher. There are two methods for text data augmentation: masking and POS-guided word replacement. Preliminary experiments show that the second method does not lead to consistent improvements in quality across most of the tasks. Therefore, for each word in sentence, we perform masking with probability $p_{mask} = 0.1$. We don't perform any other modifications on this word with remaining probabilities.

4.3.3 Methods and Baselines

Table 4.1 shows that results on test data reported from the GLUE test server. Each entry in the table is briefly introduced below. Some entries have been described in previous sections so we don't repeat here. The best numbers we achieved compared all other KD methods are bolded.

[1] https://github.com/namisan/mt-dnn

Model	Size # Param	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE
		Acc	F_1/Acc	r/ρ	F_1/Acc	Acc	Acc	Acc
MTL-BERT (Teacher)	303.9M	94.7	84.7/79.7	84.0/83.3	72.3/89.6	85.9/85.7	90.5	77.7
OpenAI GPT	116.5M	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	-	56.0
ELMo	93.6M	90.4	84.4/78.0	74.2/72.3	63.1/84.3	74.1/74.5	79.8	58.9
Distilled BiLSTM	1.59M	91.6	82.7/75.6	79.6/78.2	68.5/88.4	72.5/72.4	-	-
BERT-PKD	21.3M	87.5	80.7/72.5	-	68.1/87.8	76.7/76.3	84.7	58.2
TinyBERT	5.0M	92.6	86.4/81.2	81.2/79.9	71.3/89.2	82.5/81.8	87.7	62.9
$BERT_{EXTREME}$	19.2M	88.4	84.9/78.5	-	-	78.2/77.7	-	
MKD-LSTM	10.2M	91.0	85.4/79.7	80.9/80.9	70.7/88.6	78.6/78.4	85.4	**67.3**
MKD-Transformer	21.3M	90.1	86.2/79.8	**81.5/81.5**	71.1/**89.4**	79.2/78.5	83.5	67.0

Table 4.1: Results from the GLUE test server. The first group contains large-scale pre-trained language models. The second group lists previous knowledge distillation methods for BERT. Our MKD results based on LSTM and Transformer student model architectures are listed in the last group. The number of parameters doesn't include embedding layer.

MTL-BERT. We use the multi-task refined BERT (described in Section 4.1.1) as our teacher model. We tried to replicate the results of the released MT-DNN [49] model.

Distilled BiLSTM. [78] distill BERT into a simple BiLSTM. They use different models for single and pair sentences tasks.

BERT-PKD. The Patient-KD-Skip approach [75] which student model patiently learns from multiple intermediate layers of the teacher model. We use their student model consisting of three layers of Transformers.

TinyBERT [29] propose a knowledge distillation method specially designed for transformer-based models. It requires a general distillation step which is performed on a large-scale English Wikipedia (2,500 M words) corpus.

$BERT_{EXTREME}$. [91] aims to train a student model with smaller vocabulary and lower hidden state dimensions. Similar to TinyBERT, they use the same training corpus to train BERT to perform KD.

4.4 Results and Analyses

The results of our model are listed as MKD-LSTM and MKD-Transformer in the tables.

4.4.1 Model Quality

Compared with large-scale pre-trained models in the first group, MKD-LSTM achieved better or comparable performance while using much less parameters. It has higher performance than ELMo over all seven datasets: notably 8.4 points for RTE, 8.6 points in Spearman's ρ for STS-B, 7.6 points in F-1 measure for QQP, and 0.6 to 5.6 points higher for other datasets. Compared with OpenAI GPT, MKD-LSTM is 11.3 points higher for RTE and 4 points higher for MRPC.

It's not fair to directly compare MKD-LSTM and Distilled BiLSTM, since the former model additionally incorporates bi-attention module and is distilled jointly from all tasks. It's only reasonable to compare these two models under variate controlling principle. In this sense, more comparisons are presented in ablation studies: 1) BiLSTM and bi-attentive BiLSTM student models trained without distillation; 2) two models distilled from single specific task; 3) two models distilled from multiple tasks.

While using the same Transformer layers and same amount of parameters, MKD-Tranformer significantly outperforms BERT-PKD by a range of $0.4 \sim 9.1$ points. MKD-LSTM leads to significant performance gains than BERT-PKD while using far less parameters.

TinyBERT and $\text{BERT}_{\text{EXTREME}}$, these two approaches both use the large-scale unsupervised text corpus, same as the ones to train the teacher model, to execute their distillation process. However, we only use the data within downstream tasks. There are two caveats for their methods: (1) Due to massive training data, KD still requires intensive computing resources, e.g. $\text{BERT}_{\text{EXTREME}}$ takes 4 days on 32 TPU cores to train their student model. (2) The text corpus to train the teacher model is not always available due to data privacy. Under some conditions we can only access to the pretrained models and their approach are not applicable.

While not resorting to external training data, our model has the best performance across the state-of-the-art KD baselines (i.e., BERT-PKD). It also achieves comparable performance compared to intensively trained KD methods (i.e, $\text{BERT}_{\text{EXTREME}}$) on external large corpora.

4.4.2 Ablation Study

We conduct ablation studies to investigate the contributions of: (1) different training procedures (in Table 4.2); (2) Different training tasks in multi-task distillation (in Table

#	Model	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE
1	Biatt LSTM	85.8	80.4/69.9	12.24/11.33	81.1/86.5	73.0/73.7	80.3	53.1
2	Single Task Distilled Biatt LSTM	89.2	82.5/72.1	20.2/20.0	84.6/88.4	74.7/75.0	82.0	52.0
3	$\text{BiLSTM}_{\text{MTL}}$	87.5	83.2/72.8	71.6/72.6	81.6/87.0	70.2/71.3	75.4	56.3
4	MKD-LSTM *Word-level Tokenizer*	87.3	84.2/75.7	72.2/72.6	71.1/79.3	69.4/70.9	75.1	54.9
5	**MKD-LSTM**	89.3	86.8/81.1	84.5/84.5	85.2/89.0	78.4/79.2	83.0	67.9

Table 4.2: Ablation studies on GLUE dev set of different training procedures. All models are not fine-tuned. Line 1 is our bi-attentive LSTM student model trained without distillation. Line 2 is our bi-attentive LSTM student distilled from single task. Line 3 is the Multi-task distilled BiLSTM. Line 4 is the Multi-task distilled model using word-level tokenizer.

Model	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE
Sentiment Task	✓						
MKD-LSTM	89.9	81.4/70.8	51.2/49.9	84.9/88.3	74.3/74.7	83.2	50.9
PI Tasks		✓		✓			
MKD-LSTM	89.3	85.2/77.2	83.4/83.3	84.9/88.7	73.2/73.9	83.8	59.6
NLI Tasks					✓	✓	✓
MKD-LSTM	90.4	87.9/82.1	84.1/84.1	84.8/88.4	77.1/78.1	84.5	66.8
All Tasks	✓	✓	✓	✓	✓	✓	✓
MKD-LSTM	90.5	86.9/80.2	85.0/84.8	84.8/89.0	77.4/78.3	84.9	68.2

Table 4.3: Ablation experiments on the dev set using different training tasks (marked with ✓) in multi-task distillation. The results are reported with the original corpus, without augmentation data. The model is fine-tuned on each individual task.

4.3). The ablation studies are all conducted on LSTM-based student model since it has the advantage of model size and inference speed compared to Transformers.

Do we need attention in the student model? Yes. Tang et al. [78] distill BERT into a simple BiLSTM network. Results in Table 4.1 demonstrates that our model is better than *Distilled BiLSTM* and achieves an improvement range of $2.2 \sim 6.1$ points across six datasets. To make fair comparison, we also list the results of multi-task distilled BiLSTM in Line 3 in Table 4.2. It's obvious that Line 5, which is the model with bi-attentive mechanism, significantly outperform Line 3. We surmise that the attention module is an integral part of the student model for sequence modeling.

Better vocabulary choices? WordPiece works better than the word-level tokenizers in our experiments. The WordPiece-tokenized vocabulary size is 30522, while the word-level

	Distilled BiLSTM	BERT-PKD	TinyBERT	MKD-LSTM
Inf. Time	1.36	8.41	3.68	2.93

Table 4.4: The inference time (in seconds) for baselines and our model. The total inference time is reported on QNLI training set with a single NVIDIA V100 GPU.

tokenized vocabulary size is much larger, along with more unknown tokens. WordPiece effectively reduces the vocabulary size and improves rare-word handling. The comparison between Line 4 and Line 5 in Table 4.2 demonstrates that the method of tokenization influences all the tasks.

The influence of MTL in KD? The single-task distilled results are represented in Line 2 of Table 4.2. Compared with Line 5, all the tasks benefit from information sharing through multi-task distillation. Especially for STS-B, the only regression task, greatly benefit from the joint learning from other classification tasks.

We also illustrate the influence of different number of tasks for training. In Table 4.3, the training set incorporates tasks of the same type individually. Even for the tasks which are in the training sets, they still perform better in the all tasks training setting. For example, for RTE, the *All Tasks* setting increases 1.4 points than *NLI Tasks* setting. For other training settings which RTE is excluded from training set, *All Tasks* leads to better performance.

4.4.3 Inference Efficiency

To test the model efficiency, we ran the experiments on QNLI training set. We perform the inference on a single NVIDIA V100 GPU with batch size of 128, maximum sequence length of 128. The reported inference time is the total running time of 100 batches.

From Table 4.4, the inference time for our model is 2.93s. We re-implemented Distilled BiLSTM from [78] and their inference time is 1.36s. For fair comparison, we also ran inference procedure using the released BERT-PKD and TinyBERT model on the same machine. Our model significantly outperforms Distilled BiLSTM with same magnitude speed. It also achieves comparable results but is faster in efficiency compared with other distillation models.

Chapter 5

Conclusion and Future Work

In this thesis, we explore effective knowledge distillation methods to distill pretrained language models (e.g. BERT) into small student models. In order for the teacher model to fully express its knowledge, it's desirable to augment the original corpus, which is usually limited in size. In chapter 3, we first propose two strategies for task-agnostic data augmentation: random word masking and replace a word with another of the same part-of-speech (POS) tag. In order to corroborate our hypothesis that shallow neural architecture also has the representation power for text modeling, we distill BERT into a small, single-layer BiLSTM. Experiments show that our distilled BiLSTM achieves considerably better results than directly training the base BiLSTM itself. The results are competitive with ELMo, while using 98 times fewer parameters and 15 times faster inference times for single-sentence tasks, and 59 times fewer parameters and 8 times faster inference times for sentence-pair tasks. Ablation studies represent that both the distilling objectives and the data augmentation techniques contribute their own role in the process.

In chapter 3 we transfer knowledge from a single-task teacher to its single-task student. It has a downside that the KD process needs to be conducted all over again when performing on a new NLP task. The inference speed of the large-scale teacher model still remains the bottleneck for various downstream tasks distillation. In order to get around this problem and further improve the performance of the distilling approach, we propose to distill the student model from different tasks jointly in chapter 4. Also, we evaluate our approach on a LSTM-based student model and a Transformer-based student model. Compared with previous KD methods using only the training corpus within tasks, our approach achieves better performance. In contrast to other KD methods using large-scale external text corpus, our approach balances the problem of computational resources, inference speed, performance gains and availability of training data. Ablation studies show that

attention module is an integral part of the LSTM-based student model for sequence modeling. Training the student with all tasks leads to better performance than task-specific distillation.

In future studies, besides using distilled student model as a proxy for cutting down the model size of pre-trained models, we are also interested in using PLMs effectively in other aspects. One direction is to introduce syntactic structures to guide the learning of lightweight models. Jawahar et al. [27] demonstrates that BERT captures linguistic information in a compositional way that mimics tree-like structures. A recent work [46] demonstrates that incorporating structural information contributes to consistent improvements over strong baselines. We are curious what will happen to combine the syntactic information from PLM with traditional lightweight models. The other direction is how to apply such a model for a specific task. Currently, all tasks whose input are sentence pairs use BERT in the same way. All the information is squeezed into a single [CLS] word representation. Nevertheless, different tasks focus on different features. We would like to formulate a framework on the basis of pre-trained models which could be easily customized to each specific task.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5(1):135–146, 2017.

[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[7] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[8] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, 2017.

[9] Changan Chen, Frederick Tung, Naveen Vedula, and Greg Mori. Constraint-aware deep neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 400–415, 2018.

[10] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, 2019.

[11] Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc V Le. Bam! born-again multi-task networks for natural language understanding. *arXiv preprint arXiv:1907.04829*, 2019.

[12] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[13] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv:1602.02830*, 2016.

[14] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.

[15] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603. IEEE, 2013.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[18] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[19] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[20] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[21] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.

[22] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv:1510.00149*, 2015.

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

[24] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

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

[26] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

[27] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, 2019.

[28] Nanjiang Jiang and Marie-Catherine de Marneffe. Evaluating bert for natural language inference: A case study on the commitmentbank. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6088–6093, 2019.

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

[30] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[31] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences.

[32] Yoon Kim. Convolutional neural networks for sentence classification.

[33] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, 2016.

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[35] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[36] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[37] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] Leslie Lamport. *LATEX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[40] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[41] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[42] 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.

[43] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv:1608.08710*, 2016.

[44] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM, 2014.

[45] Linqing Liu, Huan Wang, Jimmy Lin, Richard Socher, and Caiming Xiong. Attentive student meets multi-task teacher: Improved knowledge distillation for pretrained models. *arXiv preprint arXiv:1911.03588*, 2019.

[46] Linqing Liu, Wei Yang, Jinfeng Rao, Raphael Tang, and Jimmy Lin. Incorporating contextual and syntactic structures improves semantic similarity modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1204–1209, 2019.

[47] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, 2015.

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

[49] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[50] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731, 2019.

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

[52] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[53] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *The Journal of Machine Learning Research*, 17(1):2853–2884, 2016.

[54] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

[55] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61, 2016.

[56] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[57] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[58] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[59] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*, 2015.

[60] Jeffrey Pennington, Richard Socher, and Christopher 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.

[61] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.

[62] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019.

[63] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *arXiv preprint arXiv:2003.08271*, 2020.

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

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

[66] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[67] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

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

[69] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637, 2019.

[70] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[71] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. 2016.

[72] Nikhil Dandekar Shankar Iyer and Kornél Csernai. First Quora dataset release: Question pairs, 2017.

[73] 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 *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[74] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

[75] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

[76] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[77] Raphael Tang, Ashutosh Adhikari, and Jimmy Lin. FLOPs as a direct optimization objective for learning sparse neural networks. *arXiv:1811.03060*, 2018.

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

[79] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[80] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[81] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[82] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, 2018.

[83] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv:1704.05426*, 2017.

[84] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. In *International Conference on Learning Representations*, 2018.

[85] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[86] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[87] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, 2019.

[88] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[89] Seunghak Yu, Nilesh Kulkarni, Haejun Lee, and Jihie Kim. On-device neural language model based word prediction. *Proceedings of COLING 2018, the 28th International Conference on Computational Linguistics: Technical Papers*, page 128, 2018.

[90] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

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

[92] Yukun Zhu, Ryan Kiros, Rich 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 *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.