Alexa Teacher Model: Pretraining and Distilling Multi-Billion-Parameter Encoders for Natural Language Understanding Systems

Jack FitzGerald*
Amazon, Denver, USA

Shankar Ananthakrishnan
Amazon, Cambridge, USA

Konstantine Arkoudas
Amazon, New York, USA

Davide Bernardi
Amazon, Turin, Italy

Abhishek Bhagia
Amazon, Seattle, USA

Claudio Delli Bovi
Amazon, Turin, Italy

Jin Cao
Amazon, New York, USA

Rakesh Chada
Amazon, Seattle, USA

Amit Chauhan
Amazon, Seattle, USA

Luoxin Chen
Amazon, Cambridge, USA

Anurag Dwarakanath
Amazon, Bangalore, India

Satyam Dwivedi
Amazon, Bangalore, India

Turan Gojayev
Amazon, Aachen, Germany

Karthik Gopalakrishnan
Amazon, Santa Clara, USA

Thomas Gueudre
Amazon, Turin, Italy

Dilek Hakkani-Tur
Amazon, Sunnyvale, USA

Wael Hamza
Amazon, New York, USA

Jonathan Hueser
Amazon, Aachen, Germany

Kevin Martin Jose
Amazon, Aachen, Germany

Haidar Khan
Amazon, New York, USA

Beiye Liu
Amazon, Cambridge, USA

Jianhua Lu
Amazon, Cambridge, USA

Alessandro Manzotti
Amazon, Turin, Italy

Pradeep Natarajan
Amazon, Illinois, USA

Karolina Owczarzak
Amazon, Cambridge, USA

Gokmen Oz
Amazon, Cambridge, USA

Enrico Palumbo
Spotify, Turin, Italy

Charith Peris
Amazon, Cambridge, USA

Chandana Satya Prakash
Amazon, Cambridge, USA

Stephen Rawls
Amazon, New York, USA

Andy Rosenbaum
Amazon, Cambridge, USA

Anjali Shenoy
Amazon, Bangalore, India

Saleh Soltan
Amazon, New York, USA

Mukund Harakere Sridhar
Amazon, Cambridge, USA

Liz Tan
Amazon, Cambridge, USA

Fabian Triefenbach
Amazon, Aachen, Germany

Pan Wei
Amazon, Cambridge, USA

Haiyang Yu
Amazon, Cambridge, USA

Shuai Zheng
Amazon, Santa Clara, USA

Gokhan Tur
Amazon, Sunnyvale, USA

Prem Natarajan
Amazon, Los Angeles, USA

*Corresponding Author - jgmf@amazon.com

ABSTRACT

We present results from a large-scale experiment on pretraining encoders with non-embedding parameter counts ranging from 700M to 9.3B, their subsequent distillation into smaller models ranging

from 17M-170M parameters, and their application to the Natural Language Understanding (NLU) component of a virtual assistant system. Though we train using 70% spoken-form data, our teacher models perform comparably to XLM-R and mT5 when evaluated on the written-form Cross-lingual Natural Language Inference (XNLI) corpus. We perform a second stage of pretraining on our teacher models using in-domain data from our system, improving error rates by 3.86% relative for intent classification and 7.01% relative for slot filling. We find that even a 170M-parameter model distilled from our Stage 2 teacher model has 2.88% better intent classification and 7.69% better slot filling error rates when compared to the 2.3B-parameter teacher trained only on public data (Stage 1), emphasizing the importance of in-domain data for pretraining. When evaluated offline using labeled NLU data, our 17M-parameter Stage 2 distilled model outperforms both XLM-R Base (85M params) and DistillBERT (42M params) by 4.23% to 6.14%, respectively. Finally, we present results from a full virtual assistant experimentation platform, where we find that models trained using our pretraining and distillation pipeline outperform models distilled from 85M-parameter teachers by 3.74%-4.91% on an automatic measurement of full-system user dissatisfaction.

CCS CONCEPTS

• **Computing methodologies → Natural language processing**; **Neural networks**; • **Human-centered computing → Personal digital assistants**.

KEYWORDS

natural language understanding, model pretraining, knowledge distillation, transformers, self-attention, distributed training, virtual assistant, voice a.i.

1 INTRODUCTION

A multi-step model training process is now dominant in most Natural Language Processing (NLP) applications, including Natural Language Understanding (NLU) [11]. In the first step of training, usually called pretraining, models are trained on large, self-supervised datasets, and they are tasked to fill in masked words, de-shuffle words or sentences, or predict the next word of a sequence. In the final step, called fine-tuning, the model is adapted to a specific task using a comparatively small labeled dataset. Additional training

steps optionally can occur between the first pretraining step and the final fine-tuning step.

In parallel to this paradigm shift, researchers have discovered a clear correlation between task performance and model size, motivating work on dense models with tens or hundreds of billions of parameters and sparse models with up to trillions of parameters (see Section 4). To be useful for latency-sensitive online applications, large models must be distilled into smaller versions or otherwise compressed. Recent knowledge distillation techniques have resulted in up to 99% [43] to 96.8% [17] of task performance preservation even after model size reductions of 50% to 86%, respectively. Moreover, models distilled from larger models typically outperform models trained from scratch at the target size [39].

In this work we consider language model pretraining and distillation for improving the NLU performance of a large-scale virtual assistant. Our core tasks are intent classification and slot filling. Given the utterance "*can you call mom,*" the NLU model should understand that the user's intent is to make a call, and it should also fill the contact name slot with the "mom" token.

We refer to our models and pipeline as Alexa Teacher Model(s) (AlexaTM) throughout this paper. Our problem space is unique as compared to many research tasks because (1) we possess relatively large labeled datasets, which reduces the effectiveness of pretraining, (2) our models must adhere to strict latency and memory constraints, (3) incoming data is of "spoken form" which differs from the "written form" text used to pretrain most public models, and (4) our system supports more than one language.

Our contributions include:

- The first example (to our knowledge) of billion-parameter encoder pretraining using spoken-form data, as well as comparisons to models trained with written-form data,
- Results from performing Stage 2 pretraining of the teacher models using in-domain data from a large, real-world system,
- Setup and results for knowledge distillation to a student 0.2% as large as its teacher (9.3B to 17M), contrasted, for example, with TinyBERT$_4$, which is 6% the size of its teacher (85M to 5M),
- Standalone results of our teacher and distilled models on both public datasets and datasets from a major NLU system, and
- Full virtual assistant system results comparing our models to baseline models trained by smaller teachers.

2 SETUP

2.1 Pretraining Datasets

Pretraining requires large datasets composed of diverse data spanning many domains, topics, tones, levels of formality, desired languages, and more. We considered three primary pretraining data sources, being the multilingual Colossal Clean Common Crawl (mC4) dataset, which was used to train T5 [31] and mT5 [53], the CC-100 dataset, which was used to train XLM-R [8, 47], and Wikipedia data, which was used to train BERT and mBERT in addition to the BooksCorpus [11]. mC4 and CC-100 are derived from Common Crawl data.

We included 12 languages for pretraining: Arabic, English, French, German, Hindi, Italian, Japanese, Marathi, Portuguese, Spanish,

Tamil, and Telugu. Following [9] we sampled sentences from the training corpus according to a multinomial distribution $\{q_i\}_{i=1...N}$, where:

$$q_i = \frac{p_i^\alpha}{\sum_{j=1}^N p_j^\alpha} \quad \text{with} \quad p_i = \frac{n_i}{\sum_{k=1}^N n_k}, \tag{1}$$

n_i is the number of examples in a given language's dataset, and N is the number of languages considered.

We used $\alpha = 0.5$ to mix data for both tokenizer training and language model pretraining. The effect is to up-sample low-resource languages. This up-sampling was performed offline prior to training. Besides language-based sampling, we also packed sentences into sequences of approximately 700 words. We performed tokenization on the fly during training, and 700 words per example allowed us to keep over 90% of sequences under 1,024 tokens post-tokenization.

In addition to public datasets, we considered a proprietary Stage 2 pretraining dataset composed of unlabeled and anonymized utterance text from our system. As preprocessing, we first reduced the duplication of examples in the dataset by repeating a given utterance only the square root of its actual count. For instance, an example that appeared 100 times in the original dataset was reduced to appear only 10 times. Second, we used the same language-sampling technique as described above for the Stage 1 pretraining dataset. Third, we removed examples with a length of fewer than 5 tokens. Finally, in order to reduce catastrophic forgetting, we then mixed the data with the public dataset used for Stage 1 pretraining following a 1:2 ratio of Stage 1 data to in-house data. Our final Stage 2 pretraining dataset had approximately 50M examples. Figure 1 shows how our datasets were used in our training pipeline.

2.2 Spoken Form Text

In Spoken Language Understanding (SLU) systems [40, 46, 55], which are composed of both Automatic Speech Recognition (ASR) and NLU components, it is common to transform text from its original "written form" into a canonical "spoken form" to facilitate ASR. This may include lower-casing, verbalizing of numbers, etc. For example, a text like *"Can you set an alarm for 7:30AM?"* might be converted to *"can you set an alarm for seven thirty a. m."*.

We observed that differences in the tokenization format can impact downstream performance. For instance, XLM-R Base, when trained on the written form of XNLI, has an English accuracy of 85.2, while on the spoken form of the English test set it drops to 83.4. We find such drops due to mismatched formatting to be more significant in smaller sized models like the ones used in a production system. To mitigate this and better align with our use cases, we train on a mixed tokenization regime. To support both formats, yet bias towards the spoken-form setting, we transformed our pretraining data into spoken form using in-house formatters, and we mixed the spoken-form version (70%) with the original written-form version (30%).

2.3 Tokenizer

We trained a SentencePiece [22] tokenizer using the unigram setting. As shown in [8], tokenizer vocabulary size can have a large impact on model performance. Larger vocabulary sizes generally lead to better task performance at the cost of training convergence speed, inference memory, and latency for masked language modeling.

It is prohibitively expensive to train a full teacher model with numerous tokenizer settings, so we developed two intrinsic tokenizer metrics: (1) split-ratio, and (2) unk-token portion. Split-ratio uses the intuition that more subword splits will result in degraded accuracy. Unk-token portion is defined as the percentage of output tokens that have the unknown token <unk>, which can seriously harm performance.

We increased the vocabulary size until we had a split-ratio and unk-token portion similar to our baseline production models (Section 3.4). To improve coverage for Japanese characters, we explicitly added the full set of the 2,136 of JōYō most common kanji [48], as well as all hiragana and katakana symbols.

We arrived at a vocabulary size of 150k subword tokens, and we used the same 70/30 mix of spoken-form and written-form data that we used for the pretraining corpus.

2.4 Pretraining

We performed pretraining following the general examples of BERT [11], RoBERTa [24], XLM-R [8], and others. Our teacher models are based on RoBERTa, but we modified them to use a pre-layernorm architecture, meaning that the layer normalization occurs immediately prior to the self attention block and the feedforward block in each transformer layer [51].

Training was conducted using the masked language modeling objective, in which 15% of tokens are masked, of which 10% are kept unchanged and 10% are replaced with a random token.

We trained teacher models with up to 9.3B non-embedding parameters, and we used Deepspeed to increase our training throughput [34]. Deepspeed Stage 1 partitions optimizer states across GPUs, and Deepseed Stage 2 further partitions gradients across GPUs. This partitioning can be achieved without any increase in network-based bottlenecks. With mixed precision training, were able to achieve up to 107 TFLOP/sec per GPU for a 9.3B-parameter encoder using AWS p4d.24xlarge instances, which are composed of Nvidia a100 GPUs, using Elastic Fabric Adapters to ensure good network throughput.

We used Deepspeed's version of mixed precision training for our pretraining runs, and we encountered FP16 overflow during certain operations in the model. To mitigate these issues, we (1) used the `baddbmm` operation instead of the `matmul` operation for our query-key multiplication and (2) converted to FP32 prior to calculating the variance as part of the layer normalizations. These changes reduced throughput by up to 20%, but they eliminated our model stability issues. Another way to mitigate the stability issues is to use BFLOAT16 [18], which was not available in Deepspeed at the time of our experiments.

2.5 Stage 2 Pretraining

Stage 2 pretraining was explored with the Muppet system [1], which the authors of the associated paper refer to as pre-finetuning. Though their pre-finetuning is multitask, for our Stage 2 pretraining, we simply continued the pretraining objective using our Stage 2 dataset described in Section 2.1. The goal was to improve our model's specialization and ability to handle virtual assistant utterances, which are typically short and often ungrammatical, while
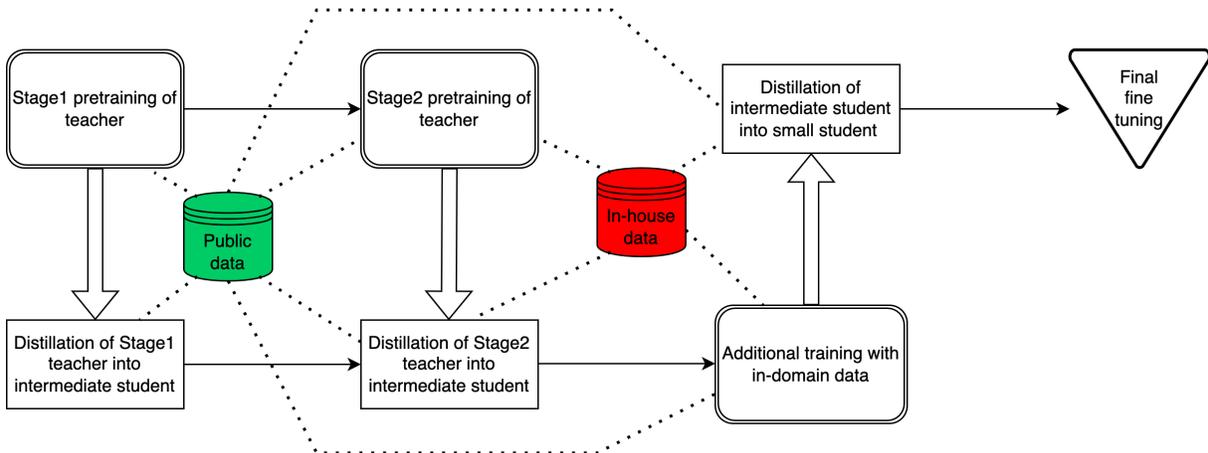
Figure 1: Our model training pipeline. A large teacher is first pretrained using public data as Stage 1. Pretraining continues with Stage 2 in-house data to create a new teacher. We then distill an intermediate student, starting with the Stage 1 teacher and then using the Stage 2 teacher. The intermediate student/teacher is then further trained on in-house unlabeled data before being distilled into the final student. The final student in then fine-tuned on labeled data.

not catastrophically forgetting general language knowledge learned during Stage 1 [5, 14]. Our hyperparameter choices are explained further in Section 3.2.

To examine the effectiveness of domain adaptation with Stage 2 pretraining, we evaluated our models' performance on intent classification and slot filling, using data from three diverse domains. Each domain's dataset contained a mix of 7 languages—English, German, Spanish, French, Italian, Portuguese and Japanese—and each dataset contained 80k to 90k training utterances. Statistics are shown in Table 1.

	Domain 1	Domain 2	Domain 3
Training data size	90k	86k	80k
Validation data size	10k	10k	10k
Test data size	20k	20k	20k
# of intents	16	8	12
# of slots	98	25	56

Table 1: Description of the manually transcribed and labeled datasets used for offline NLU evaluation of Stage 2 versus Stage 1 performance.

We adopted two modes for Stage 2 evaluation. First, we followed a standard fine-tuning scheme, allowing all parameters, including parameters from the pretrained encoder as well as from classification head, to adapt to the task. Another mode we adopted is to freeze all the parameters from the pretrained network and only allow parameters in the classification head to learn. We consider the latter mode as a more difficult task, given that the entire pretrained encoder is frozen and is essentially used as a feature extractor. Thus, this latter mode may be a stronger indicator for the effectiveness of the pretrained encoder at creating generic representations useful for downstream tasks.

2.6 Distillation

Low-latency applications require models of relatively small sizes. However, distilling from large pretrained models into much smaller models directly can hinder the student from fully taking advantage of the teacher's knowledge [7]. Therefore, we distill the pretrained teacher models in two phases with a teacher assistant setup [26, 44]. The distillation workflow is depicted in Figure 1. First, we distill an intermediate sized model from the large teacher model. We then use this distilled model as a teacher for the final student.

When distilling the intermediate model, we followed a similar approach to the pretraining of the teacher. A randomly initialized student model was distilled from the Stage 1 teacher model. Once training converged, we switched the teacher model to the Stage 2 teacher and resumed the distillation process. For both of these stages, the distillation data is the same that was used for teacher pretraining for its respective stage. As for our distillation techniques, we explored different components described in [17]. Our final run for the intermediate student/teacher used the sum of categorical cross-entropy (MLM loss) and soft cross-entropy weighted equally, because we did not observe any gain from utilizing the attention and hidden layer outputs of the teacher.

For our final student, we first pretrained the intermediate model further without teacher involvement on Stage 2 data only. Next, we distilled it into the final, small student. The distillation techniques in this phase were similar to the first distillation phase, with an additional usage of hidden-layer output matching as in [17].

2.7 No-Fine-Tune Validation

In order to monitor the progress of training, one standard approach is to measure perplexity on a held-out validation dataset. One issue with perplexity measurement is that it differs depending on the tokenizer choice. Thus, we developed a separate task called "mask-filling accuracy" to compare models.
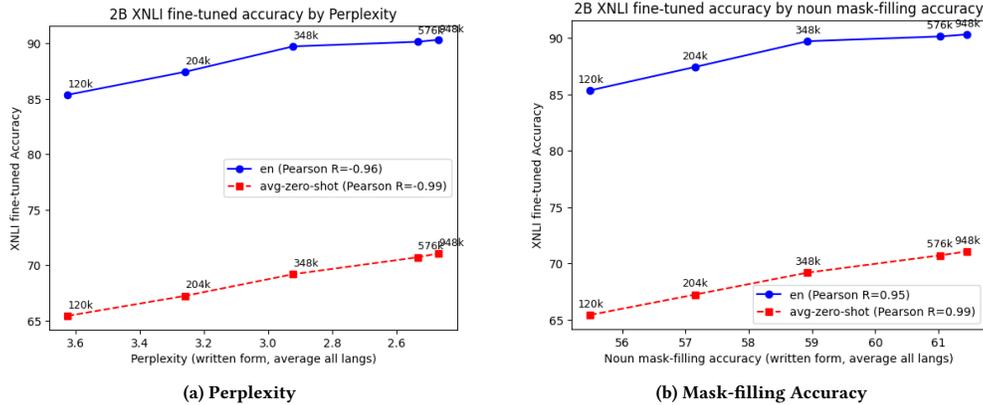
(a) Perplexity

(b) Mask-filling Accuracy

Figure 2: Correlation to XNLI accuracy from (a) perplexity and (b) mask-filling accuracy across model updates using our 2.3B-parameter model. The greater the correlation, the better the metric is for no-fine-tune validation.

Model	en	ar	de	es	fr	hi	avg 0-shot
XLM-R Base (0.27B)	85.8	73.8	78.7	80.7	79.7	72.4	77.1
XLM-R Large (0.6B)	89.1	79.8	83.9	85.1	84.1	76.9	82
XLM-R XL (3.5B)	90.7	81.6	84.6	86.5	85.5	78.5	83.3
XLM-R XXL (10.7B)	91.6	82.5	87	87.3	86.2	79.8	84.6
mT5 Large (1.2B)	89.4	79.8	83.4	84.2	84.1	77.6	81.8
mT5 XL (3.7B)	90.6	82.2	85.8	81.3	85.3	80.4	83
mT5 XXL (13B)	92.3	84.4	87.3	88.3	87.3	82.5	86
AlexaTM 9.3B Stage 1 (9.9B)	91.9	82.2	86.9	87.4	86.8	80.2	84.7
AlexaTM 2.3B Stage 1 (2.68B)	90.3	80	84.7	85.9	85.3	77.3	82.6
AlexaTM 170M from 2.3B Stage 1 (0.33B)	87.3	77.6	81	82.5	81.7	74.6	79.5

Table 2: Results on XNLI for the Stage 1 pretrained 2.3B- and 9.3B-parameter models, as well as the 170M-parameter model distilled from the Stage 1 2.3B-parameter model. The number of parameters including the embeddings is given in parentheses.

We selected texts from a variety of public tasks including XNLI [10], PAWS-X [54], and Multi-lingual Amazon Reviews [20]. We then removed these examples from our training data. For each example, we use the Stanza tagger [29] to identify a noun word, then mask all subword tokens for that word. The model must correctly predict all subword tokens in the noun to count as correct.

We show (Figure 2) that both perplexity and mask-filling accuracy correlate strongly with XNLI performance across model update steps.

3 RESULTS AND ANALYSES

3.1 Stage 1 Pretraining

To measure the effectiveness of our pretraining and distillation setup using Stage 1 data (public data), we used XNLI to benchmark our 2.3B-parameter teacher model, a similarly trained 9.3B-parameter teacher model, and a 170M-parameter model distilled from the 2.3B-parameter model. Following standard practice, we trained and validated on English data only, and we tested on all languages separately. Non-English test results were averaged to determine the average zero-shot accuracy. See Table 2.

We found that the 2.3B-parameter and 9.3B-parameter models are competitive with comparably-sized public models, even though our model training set was 70% spoken-form data, whereas the public models and XNLI use written-form data. English XNLI accuracy drops by 3 points after distillation from 2.3B non-embedding parameters to 170M parameters, as well as by 3.1 points for average zero-shot accuracy.

Next, we examined the perplexity and noun mask-filling accuracy for spoken-form data derived from datasets spanning all of our languages, including XNLI, PAWS-X, and Amazon Reviews, as described in Section 2.7. See Table 3. We expected to see perplexity decrease and noun mask filling accuracy increase with increasing model sizes, which we do observe.

3.2 Stage 2 Pretraining

To examine the effectiveness of Stage 2 pretraining, we used the 2.3B-parameter Stage 1 model as a baseline and compared it to various sizes of the Stage 2 models, including the 2.3B-parameter Stage 2 model, the 170M-parameter Stage 2 model distilled from the 2.3B-parameter Stage 2 model, and the 17M-parameter Stage 2

Model	Perplexity	Noun Mask Fill Acc
XLM-R Large	39.04	49.69
AlexaTM 2.3B Stage 1	11.56	62.74
AlexaTM 9.3B Stage 1	8.80	65.09

Table 3: No-fine-tune perplexity and noun mask-filling accuracy on spoken-form data only, macro-averaged across all languages. See Sections 2.7 and 2.4. Note that XLM-R was trained on written-form data.

model distilled from the 170M-parameter Stage 2 model. For classification heads, we implemented two feed-forward layers of hidden size 256, followed by one softmax layer for intent classification and one for slot filling.

Throughout our experiments, we trained all models (baselines and different size of Stage 2 models) with mini-batch sizes ranging from 16 to 64 using 8 Nvidia Tesla V100 GPUs. We used the Adam optimizer, a maximum learning rate of 2e-5 for the fine-tune mode, and a maximum learning rate of 1e-3 for the frozen mode . We report mean statistics across 3 random seed runs.

We show the domain adaptation results from full fine-tuning and the frozen encoder mode in Tables 4(a) and 4(b). In fine-tune mode and compared with the 2.3B-parameter Stage 1 model, the 2.3B-parameter Stage 2 model reduces intent classification error rate by 3.86% on average, as well as slot filling errors by 7.01% on average. The 170M-parameter Stage 2 model performs surprisingly similarly to its 2.3B-parameter Stage 2 teacher, suggesting that the 2.3B-parameter model may be overparameterized for this task. However, when distilling to 17M parameters from the 170M-parameter model, intent classification error and slot error degrade by 10.95% and 11.51% relative to the 2.3B-parameter Stage 1 teacher.

When freezing the encoder, the Stage 2 models perform even better than the fine-tuning differential with the Stage 1 models, which is logical given the Stage 2 pretraining task. Yet again, only the 17M-parameter model cannot beat the 2.3B-parameter Stage 1 model (except for slot-filling in Domain 2).

Overall, Stage 2, domain-adaptive pretraining shows improved results on intent classification and slot filling tasks when compared with a model trained only on public data.

3.3 NLU Results after Distillation

We compared our distilled models to public models using the full training sets for our system (the same training data used in Section 3.4). As public models, we consider both XLM-R Base, which has 85M non-embedding parameters, and the multilingual DistillBERT [36], which has 42M non-embedding parameters. Results are given in Table 5 using exact match error rate. To count as an exact match for a given example, the model must get the intent and all slots correct.

We see that both of our distilled models outperform both public models on average. Most encouragingly, our 17M-parameter model (improvement of 4.23% versus XLM-R) shows only minimal degradation versus our 170M-parameter model (improvement of 4.82% versus XLM-R).

3.4 Full System Results

To evaluate model performance in the context of a full virtual assistant system, we follow the setup described in Section 2.6 and use an intermediate-sized model as a teacher-assistant to distill the final student models. This intermediate-sized model consisted of 170M non-embedding parameters and was distilled from a 700M-parameter Stage 1 teacher for 160K updates, the Stage 1 2.3B-parameter teacher for 105K more updates, followed by the stage 2 2.3B-parameter model for 300K more updates. See Appendix A for hyperparameter details of the 700M-parameter model.

The 170M-parameter model was then used as a teacher to distill 17M-parameter models that were used online. Before commencing distillation, the 170M-parameter teacher was fine-tuned for 15,625 updates with the same task-specific dataset that was used for the subsequent distillation process. The distillation itself was performed using both logit matching and hidden layer matching, for which we mapped student layers (0, 1, 2, 3) to teacher layers (3, 7, 11, 15), following [17]. We found that optimal performance, for the two locales explored, was achieved by using two different checkpoints from the same 17M-parameter model distillation process—the first which was taken after 80M examples and the second which was taken after 200M examples. We used a combination of 9 languages when distilling to the 17M-parameter models, being English, French, German, Hindi, Italian, Marathi, Spanish, Tamil, and Telugu.

We considered two baseline models, each being a 5M-parameter monolingual encoder distilled from a teacher with a BERT-Base architecture. The training and distillation sets for these baseline models was comprised of Wikipedia dumps using the language in question. The text was converted to spoken form in the same manner as described in Section 2.2. See Table 8 for details on architectures.

We conducted our studies using an experimentation platform for an entire virtual assistant system. We compared our models to the baselines both in parallel, as an A/B test using a different user cohort, as well as in series using the same user cohort. Results are given in Table 6. We examined an automated measurement of user dissatisfaction across the entire virtual assistant system (not just the NLU component), which was based on the user's responses and whether the system correctly executed a task. We also consider tail dissatisfaction, which is the dissatisfaction rate for utterances not within the top 500 most common utterances. Finally, we provide the offline Semantic Error Rate (SemER) [28, 35] results for the models using the same NLU test set as was used in Section 3.3. The SemER metric is used to evaluate the intent and slot-filling performance jointly. Comparing a reference of tokens and their accompanying labels, performance is defined according to the following: (1) Correct slots, where the slot name and slot value is correctly identified, (2) Deletion errors, where the slot name is present in the reference but not in the hypothesis, (3) Insertion errors, where extraneous slot names are included in the hypothesis, (4) Substitution errors, where slot names from the hypothesis are included but with an incorrect slot value. Intent classification errors are substitution errors.

$$SemER = \frac{\text{\# Deletion} + \text{\# Insertion} + \text{\# Substitution}}{\text{\# Correct} + \text{\# Deletion} + \text{\# Substitution}} \quad (2)$$

	(a)			
Full Fine Tuning				
Relative Intent Class Error Reduction Versus 2.3B Stage 1				
	Domain 1	Domain 2	Domain 3	Avg
2.3B Stage 2	-3.41%	-2.38%	-5.79%	-3.86%
170M from 2.3B	-3.16%	-4.13%	-1.36%	-2.88%
17M from 170M	11.49%	10.63%	10.73%	10.95%
Relative Slot Filling Error Reduction Versus 2.3B Stage 1				
	Domain 1	Domain 2	Domain 3	Avg
2.3B Stage 2	-5.40%	-9.95%	-5.68%	-7.01%
170M from 2.3B	-2.52%	-12.03%	-8.53%	-7.69%
17M from 170M	27.07%	2.11%	5.36%	11.51%

	(b)			
Frozen Encoder				
Relative Intent Class Error Improvement Versus 2.3B Stage 1				
	Domain 1	Domain 2	Domain 3	Avg
2.3B Stage 2	-12.60%	-4.59%	-2.23%	-6.47%
170M from 2.3B	-16.07%	-17.23%	-13.95%	-15.75%
17M from 170M	13.99%	7.42%	10.83%	10.74%
Relative Slot Filling Error Improvement Versus 2.3B Stage 1				
	Domain 1	Domain 2	Domain 3	Avg
2.3B Stage 2	-5.51%	-18.71%	-6.72%	-10.31%
170M from 2.3B	-6.15%	-12.03%	-3.70%	-7.29%
17M from 170M	15.41%	-6.30%	3.20%	4.11%

Table 4: (a) Full fine-tuning and (b) frozen-encoder results for the 2.3B-parameter Stage 2 model, the distilled 170M-parameter Stage 2 model, and the 17M-parameter Stage 2 model, evaluated using a natural language understanding dataset (intent classification and slot filling) from our real-world system (see Table 1). A negative value indicates a reduced error rate versus the baseline 2.3B-parameter Stage 1 model.

	Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Avg
Distill-mBERT	5.50%	2.07%	1.61%	-2.30%	1.41%	3.74%	12.34%	3.48%
AlexaTM 170M Stage 2	-2.20%	-8.53%	-7.61%	-5.84%	-7.64%	-2.80%	0.90%	-4.82%
AlexaTM 17M Stage 2	0.50%	-6.12%	-6.19%	-8.02%	-5.64%	-1.51%	-2.63%	-4.23%

Table 5: Exact match results for our AlexaTM distilled models and DistillBERT versus XLM-R. A given example is a successful exact match if the intent and all slots are correct. All models are trained on the same training sets as used for Section 3.4. Results are given across 7 locales (language and region). A negative value indicates an improvement in exact match error rate.

	Exp 1	Exp 2
Base Teacher Non-Embed Params	85M	85M
Base Layers/Hidden Size/FF Size	4/312/1200	4/312/1200
Base Non-Embed Param Count	5M	5M
Base Langs Supported	1	1
Cand Teacher's Teacher Non-Embed Params	2.3B	2.3B
Cand Teacher Non-Embed Params	170M	170M
Cand Layers/Hidden Size/FF Size	4/768/1200	4/768/1200
Cand Non-Embed Params	17M	17M
Cand Langs Supported	9	9
Cand Distill Examples	80M	200M
Test Locale	1	2
Whole System User Dissatisfaction A/B	-3.74%	-4.91%
Whole System User Dissatisfaction Tail A/B	-10.3%	-7.50%
Whole System User Dissatisfaction Seq	-14.9%	-7.2%
Offline SemER	-15.6%	-2.98%

Table 6: Results from a virtual assistant experimentation platform for two experiments (Exp) in two locales comparing our candidate distilled 17M-parameter model (Cand) to baseline models (Base) distilled from an 85M-parameter teacher trained on Wikipedia data. Relative results are given for whole-system user dissatisfaction, an automatic metric, from both a parallel, A/B test with different user cohorts, as well as sequential results with the same users. Tail A/B results based on utterances not within the top 500 are also given. For reference, we also report Semantic Error Rate (SemER) for the NLU component using the same labeled test set as used for Table 5.

We find that models produced using our pretraining and distillation pipeline reduce overall user dissatisfaction by 3.74% to 4.91% and tail utterance dissatisfaction by 7.50% to 10.3% in the A/B test framework. Sequential results are even better, with up to a 14.9% improvement, though they are less trustworthy given other possible changes to the platform over time. Offline SemER improves by 2.98% to 15.6%.

One caveat of our full-system study is the difference in parameter count between the baseline models and the candidate models. To determine the effect of final model size on performance, we fine-tuned a 5M-parameter model akin to the baselines used for experiments 1 and 2. We then distilled and fine-tuned an otherwise-equivalent 17M-parameter model using the same data. Across two different languages we only saw offline SemER improvements from between 0.25% and 0.38% when increasing the model size from 5M to 17M parameters with everything else equal. This suggests that a significant portion of the improvement seen in Table 6 is due to our pretraining and distillation pipeline, not due to the differing final model sizes. Moreover, our candidate encoders were pretrained using 12 languages and distilled with 9 languages, whereas the baseline encoders were trained, distilled, and fine-tuned only with a single language.

4 RELATED WORK

SLU systems, composed of both speech recognition and NLU components, have been explored extensively for decades [6, 25, 41]. Many recent efforts have focused on scaling up the size of pretrained language models to improve downstream tasks, including tasks related to NLU. [19] proposed a power-law scaling relationship between parameter count and performance, and subsequent papers empirically confirmed this relationship for very large models, including [30] which trained models up to 1.5 billion parameters and [3] which trained models up to 175 billion parameters. Various approaches to efficiently train such large models have been explored, including model state partitioning approaches [33] and pipeline parallelism approaches [27, 37]. Recently [38] combined these approaches and trained a 530 billion parameter model. Other lines of work have explored increasing the parameter count by introducing sparsity, using various mixture-of-expert approaches to train models of over 1 trillion parameters [12, 21, 23, 32].

Early work [2, 4, 16] suggested supervising small-sized models by using larger teacher models with the idea being that the mimicking of the teacher behavior can give small models a competitive advantage over the same-sized models trained without a teacher. In recent years, matching internal layer outputs from student and teacher models as an auxiliary task [17, 43, 45] has yielded even higher performance gains.

5 CONCLUSION

We have described a model development pipeline in which transformer-based encoders are first pretrained from scratch using public data (Stage 1), adapted to their system using in-house, unlabeled data (Stage 2), distilled to runtime-ready sizes using a 2-step distillation process, and then fine-tuned. Traditionally, production-focused NLU models are either distilled from models with 85M-300M parameters (Base-sized to Large-sized) and then fine-tuned, or they are trained from scratch on the final labeled dataset. Our AlexaTM pipeline, which starts with models containing 2.3B+ parameters, significantly improves upon this paradigm, including in NLU benchmarks and in user dissatisfaction reduction across an entire virtual assistant system. In particular, we find a large teacher, Stage 2

pretraining, a teacher-assistant distillation process, and in-domain-specific final distillation to be key techniques for improving task performance.

As future work, we would like to more robustly characterize the use of public pretrained conversational models like TOD-BERT [49] and ConveRT [15], evaluate more combinations of teacher and distilled model sizes, benchmark with different public datasets like MultiATIS [42, 52], mTOP [50], or MASSIVE [13], make greater use of dialog and user context, experiment with code-switching, examine varying levels of ASR noise, and more.

REFERENCES

[1] Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettle-moyer, and Sonal Gupta. 2021. Muppet: Massive Multi-task Representations with Pre-Finetuning. , 5799–5811 pages. https://doi.org/10.18653/v1/2021.emnlp-main.468

[2] Jimmy Ba and Rich Caruana. 2014. Do Deep Nets Really Need to be Deep?. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/ea8fcd92d59581717e06eb187f10666d-Paper.pdf

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. 33 (2020), 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[4] Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model Compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Philadelphia, PA, USA) *(KDD '06)*. Association for Computing Machinery, New York, NY, USA, 535–541. https://doi.org/10.1145/1150402.1150464

[5] Jin Cao, Jun Wang, Wael Hamza, Kelly Vanee, and Shang-Wen Li. 2020. Style Attuned Pre-training and Parameter Efficient Fine-tuning for Spoken Language Understanding. (2020).

[6] Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909* (2019).

[7] Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4794–4802.

[8] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747

[9] Alexis CONNEAU and Guillaume Lample. 2019. Cross-lingual Language Model Pretraining. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf

[10] Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating Cross-lingual Sentence Representations.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. , 4171–4186 pages. https://doi.org/10.18653/v1/N19-1423

[12] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. , 39 pages. http://jmlr.org/papers/v23/21-0998.html

[13] Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2022. MASSIVE: A 1M-Example Multilingual Natural Language Understanding Dataset with 51 Typologically-Diverse Languages. https://doi.org/10.48550/ARXIV.2204.08582

[14] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. , 8342–8360 pages. https://doi.org/10.18653/v1/2020.acl-main.740

[15] Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vulić. 2020. ConveRT: Efficient and Accurate Conversational Representations from Transformers. , 2161–2174 pages. https://doi.org/10.18653/v1/2020.findings-emnlp.196

[16] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*. http://arxiv.org/abs/1503.02531

[17] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 4163–4174. https://doi.org/10.18653/v1/2020.findings-emnlp.372

[18] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. 2019. A Study of BFLOAT16 for Deep Learning Training. arXiv:1905.12322 [cs.LG]

[19] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. arXiv:2005.14165 [cs.LG]

[20] Phillip Keung, Y. Lu, György Szarvas, and Noah A. Smith. 2020. The Multilingual Amazon Reviews Corpus.

[21] Young Jin Kim, Ammar Ahmad Awan, Alexandre Muzio, Andres Felipe Cruz Salinas, Liyang Lu, Amr Hendy, Samyam Rajbhandari, Yuxiong He, and Hany Hassan Awadalla. 2021. Scalable and Efficient MoE Training for Multitask Multilingual Models. arXiv:2109.10465 [cs.CL]

[22] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Brussels, Belgium, 66–71. https://doi.org/10.18653/v1/D18-2012

[23] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. BASE Layers: Simplifying Training of Large, Sparse Models.

[24] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 [cs.CL]

[25] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 3 (2014), 530–539.

[26] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved Knowledge Distillation via Teacher Assistant. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (Apr. 2020), 5191–5198. https://doi.org/10.1609/aaai.v34i04.5963

[27] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. 2019. PipeDream: Generalized Pipeline Parallelism for DNN Training. (2019), 1–15. https://doi.org/10.1145/3341301.3359646

[28] Charith Peris, Gokmen Oz, Khadige Abboud, Venkata sai Varada Varada, Prashan Wanigasekara, and Haidar Khan. 2020. Using multiple ASR hypotheses to boost i18n NLU performance. (Dec. 2020), 30–39. https://aclanthology.org/2020.icon-main.5

[29] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. https://nlp.stanford.edu/pubs/qi2020stanza.pdf

[30] A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. , 67 pages. http://jmlr.org/papers/v21/20-074.html

[32] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. arXiv:2201.05596 [cs.LG]

[33] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. ZeRO: Memory Optimization Towards Training A Trillion Parameter Models. *CoRR* abs/1910.02054 (2019). arXiv:1910.02054 http://arxiv.org/abs/1910.02054

[34] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. arXiv:1910.02054 [cs.LG]

[35] Milind Rao, Pranav Dheram, Gautam Tiwari, Anirudh Raju, Jasha Droppo, Ariya Rastrow, and Andreas Stolcke. 2021. DO as I Mean, Not as I Say: Sequence Loss Training for Spoken Language Understanding. , 7473-7477 pages.

[36] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108 [cs.CL]

[37] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *CoRR* abs/1909.08053 (2019). arXiv:1909.08053 http://arxiv.org/abs/1909.08053

[38] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. arXiv:2201.11990 [cs.CL]

[39] Saleh Soltan, Haidar Khan, and Wael Hamza. 2021. Limitations of Knowledge Distillation for Zero-shot Transfer Learning. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*. Association for Computational Linguistics, Virtual, 22–31. https://doi.org/10.18653/v1/2021.sustainlp-1.3

[40] Gokhan Tur and Renato De Mori. 2011. Spoken Language Understanding: Systems for Extracting Semantic Information from Speech.

[41] Gökhan Tür, Jerry H Wright, Allen L Gorin, Giuseppe Riccardi, and Dilek Hakkani-Tür. 2002. Improving spoken language understanding using word confusion networks.. In *Interspeech*.

[42] Shyam Upadhyay, Manaal Faruqui, Gokhan Tür, Hakkani-Tür Dilek, and Larry Heck. 2018. (Almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6034–6038.

[43] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 2140–2151. https://doi.org/10.18653/v1/2021.findings-acl.188

[44] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957 [cs.CL]

[45] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 5776–5788. https://proceedings.neurips.cc/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[46] Ye-Yi Wang, Li Deng, and Alex Acero. 2005. Spoken language understanding. *IEEE Signal Processing Magazine* 22 (2005), 16–31.

[47] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data. In *Proceedings of the 12th Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 4003–4012. https://aclanthology.org/2020.lrec-1.494

[48] Wikipedia contributors. 2021. Jōyō kanji — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=J%C5%8Dy%C5%8D_kanji&oldid=1039289460 [Online; accessed 28-January-2022].

[49] Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogue. , 917–929 pages. https://doi.org/10.18653/v1/2020.emnlp-main.66

[50] Menglin Xia and Emilio Monti. 2021. Multilingual Neural Semantic Parsing for Low-Resourced Languages. In *The Tenth Joint Conference on Lexical and Computational Semantics*.

[51] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On Layer Normalization in the Transformer Architecture. , 10524–10533 pages. https://proceedings.mlr.press/v119/xiong20b.html

[52] Weijia Xu, Batool Haider, and Saab Mansour. 2020. End-to-End Slot Alignment and Recognition for Cross-Lingual NLU. , 5052–5063 pages. https://doi.org/10.18653/v1/2020.emnlp-main.410

[53] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. , 483–498 pages. https://doi.org/10.18653/v1/2021.naacl-main.41

[54] Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification.

[55] Steve J. Young. 2002. Talking to machines (statistically speaking). In *INTERSPEECH*.

A   MODEL HYPERPARAMETERS

See Table 7 for Stage 1 teacher model hyperparameters and Table 8 for hyperparemeters used with models associated with our full-system experiments (Section 3.4).

Hyperparam	700M Teacher	2.3B Teacher	9.3B Teacher
Number of Layers	20	29	46
Hidden size	1536	2560	4096
FFN inner hidden size	6144	10240	16384
Attention heads	16	32	32
Attention head size	64	80	128
Dropout	0.1	0.1	0.1
Attention Dropout	0.1	0.1	0.1
Warmup Steps	1k	5k	10k
Peak Learning Rate	1e-3	1.5e-4	1.4e-4
Min Learning Rate	1e-5	1e-5	1e-5
Max Length	1024	512	512/1024
Batch Size (Sequences)	2048	2048	4096
Batch Size (Tokens)	2M	1M	2M
Weight Decay	0.1	0.1	0.1
LR Decay steps	500k	500k	600k
Max Steps		950k	570k
Learning Rate Warmup	Exponential	Exponential	Linear
Learning Rate Decay	Linear	Linear	Linear
Adam epsilon	1e-8	1e-8	1e-8
Adam beta1	0.9	0.9	0.9
Adam beta2	0.99	0.9	0.99
Gradient Clipping	1.0	1.0	1.0

Table 7: Hyperparameters used for Stage 1 Teacher Models

	Base Teacher	Base Distillation	AlexaTM 170M Distillation Stage 1	AlexaTM 170M Distillation Stage 2	AlexaTM MLM of 170M	AlexaTM 17M Distillation
Number of Layers	12	4	16	16	16	4
Hidden Size	768	312	1024	1024	1024	768
FFN Inner Hidden Size	3072	1200	3072	3072	3072	1200
Attention Heads	12	12	16	16	16	12
Dropout	0.1	0.1	0.1	0.1	0.1	0.1
Peak Learning Rate	0.5	0.5	1.50E-03	2.00E-04	1.00E-05	1.00E-03
LR Warmup Type	Noam	Noam	Exponential	Exponential	Exponential	Exponential
LR Decay	Noam	Noam	Linear	Linear	Linear	Linear
Warmup Steps	3250	3250	100k	10k	10k	500
LR Decay Steps	N/A	N/A	1M	250k	15.6k	195k
Max Length	512	512	512	30	512	512
Tokens per Batch	64k	128k	393k	492k	24.6k	8192
Number of Updates	40k	40k	1.5M	360k	1M	5M / 12.5M
Adam Epsilon	1.00E-09	1.00E-09				
Adam Beta1	0.9	0.9				
Adam Beta2	0.98	0.98				
Lamb Epsilon			1.00E-08	1.00E-08	1.00E-08	1.00E-08
Lamb Beta1			0.9	0.9	0.9	0.9
Lamb Beta2			0.999	0.999	0.999	0.999

Table 8: Hyperparameters used for full-system experiments described in Section 3.4

.