Multi-Stage Distillation Framework for Massive Multi-lingual NER

Subhabrata Mukherjee
Microsoft Research
Redmond, WA
submukhe@microsoft.com

Ahmed Awadallah
Microsoft Research
Redmond, WA
hassanam@microsoft.com

Abstract

Deep and large pre-trained language models are the state-of-the-art for various natural language processing tasks. However, the huge size of these models could be a deterrent to use them in practice. Some recent and concurrent works use knowledge distillation to compress these huge models into shallow ones. In this work we study knowledge distillation with a focus on multi-lingual Named Entity Recognition (NER). In particular, we study several distillation strategies and propose a stage-wise optimization scheme leveraging teacher internal representations that is agnostic of teacher architecture and show that it outperforms strategies employed in prior works. Additionally, we investigate the role of several factors like the amount of unlabeled data, annotation resources, model architecture and inference latency to name a few. We show that our approach leads to massive compression of MBERT-like teacher models by upto $35x$ in terms of parameters and $51x$ in terms of latency for batch inference while retaining 95% of its F_1-score for NER over 41 languages.

1 Introduction

Motivation: Pre-trained deep language models have shown state-of-the-art performance for various natural language processing applications like text classification, named entity recognition, question-answering, etc. A significant challenge facing many practitioners is how to deploy these huge models in practice. For instance, BERT Large and GPT 2 contain $340M$ and $1.5B$ model parameters respectively. Although these models are trained offline, during prediction we still need to traverse the deep neural network architecture stack involving a large number of parameters. This significantly increases latency and memory requirements.

Knowledge distillation (Hinton et al., 2015; Ba and Caruana, 2014), originally developed for computer vision applications, provides one of the techniques to compress huge neural networks into smaller ones. In this, shallow models (called students) are trained to mimic the output of huge models (called teachers) based on a transfer set. Similar approaches have been recently adopted for language model distillation.

Limitations of existing work: Recent works (Liu et al., 2019; Zhu et al., 2019; Tang et al., 2019; Turc et al., 2019) leverage only the soft output (logits) from the teacher as optimization targets for distilling student models, with some notable exceptions from concurrent work. Sun et al. (2019); Sanh (2019); Aguilar et al. (2019)[1]; Zhao et al. (2019)[1] additionally use internal representations from the teacher to provide useful hints for distilling better students. However, these methods are constrained by the teacher architecture like embedding dimension in BERT and transformer architectures. This makes it difficult to massively compress these models (without being able to reduce the network width) or adopt alternate architectures. For instance, we observe BiLSTMS as students to be more accurate than Transformers for low latency configurations. Some of the concurrent works (Turc et al., 2019)[1]; (Zhao et al., 2019)[1] adopt pre-training or dual training to distil student models of arbitrary architecture. However, pre-training is expensive both in terms of time and computational resources.

Additionally, most of the above works are geared for distilling language models for GLUE tasks. There has been very limited exploration of such techniques for NER (Izsak et al., 2019; Shi et al., 2019) or multi-lingual tasks (Tsai et al., 2019). Moreover, these works also suffer from the same drawbacks as mentioned before.

Overview of our method: In this work, we compare distillation strategies used in all the above works and propose a new scheme outperforming

[1] Currently under review at ICLR or alternate.

prior ones. In this, we leverage teacher internal representations to transfer knowledge to the student. However, in contrast to prior work, we are not restricted by the choice of student architecture. This allows representation transfer from Transformer-based teacher model to BiLSTM-based student model with different embedding dimensions and disparate output spaces. We also propose a stage-wise optimization scheme to sequentially transfer most general to task-specific information from teacher to student for better distillation.

Overview of our task: Unlike prior works mostly focusing on GLUE tasks in a single language, we employ our techniques to study distillation for massive multi-lingual Named Entity Recognition (NER) over 41 languages. Prior work on multi-lingual transfer on the same (Rahimi et al., 2019) (MMNER) requires knowledge of source and target language whereby they judiciously select pairs for effective transfer resulting in a customized model for each language. In our work, instead, we adopt Multi-lingual Bidirectional Encoder Representations from Transformer (MBERT) as our teacher and show that it is possible to perform language-agnostic joint NER for all languages with a single model that has a similar performance but massively compressed in contrast to MBERT and MMNER.

Perhaps, the closest work to this work is that of (Tsai et al., 2019) where MBERT is leveraged for multi-lingual NER. We discuss this in details and use their strategy as one of our baselines. We show that our distillation strategy is better leading to a much higher compression and faster inference. We also investigate several unexplored dimensions of distillation like the impact of unlabeled transfer data and annotation resources, choice of multi-lingual word embeddings, architectural variations and inference latency to name a few.

Our techniques obtain massive compression of MBERT-like teacher models by upto $35x$ in terms of parameters and $51x$ in terms of latency for batch inference while retaining 95% of its performance for massive multi-lingual NER, and matching or outperforming it for classification tasks. Overall, our work makes the following *contributions*:

- **Method:** We propose a distillation method leveraging internal representations and parameter projection that is agnostic of teacher architecture.
- **Inference:** To learn model parameters, we propose stage wise optimization schedule with gradual unfreezing outperforming prior schemes.
- **Experiments:** We perform distillation for multi-lingual NER on 41 languages with massive compression and comparable performance to huge models[2]. We also perform classification experiments on four datasets where our compressed models perform at par with huge teachers.
- **Study:** We study the influence of several factors on distillation like the availability of annotation resources for different languages, model architecture, quality of multi-lingual word embeddings, memory footprint and inference latency.

Problem Statement: Consider a sequence $x = \langle x_k \rangle$ with K tokens and $y = \langle y_k \rangle$ as the corresponding labels. Consider $D_l = \{\langle x_{k,l} \rangle, \langle y_{k,l} \rangle\}$ to be a set of n labeled instances with $X = \{\langle x_{k,l} \rangle\}$ denoting the instances and $Y = \{\langle y_{k,l} \rangle\}$ the corresponding labels. Consider $D_u = \{\langle x_{k,u} \rangle\}$ to be a transfer set of N unlabeled instances from the same domain where $n << N$. Given a teacher $\mathcal{T}(\theta^t)$, we want to train a student $\mathcal{S}(\theta^s)$ with θ being trainable parameters such that $|\theta^s| << |\theta^t|$ and the student is comparable in performance to the teacher based on some evaluation metric. In the following section, the superscript 't' always represents the teacher and 's' denotes the student.

2 Related Work

Model compression and knowledge distillation: Prior works in the vision community dealing with huge architectures like AlexNet and ResNet have addressed this challenge in two ways. Works in model compression use quantization (Gong et al., 2014), low-precision training and pruning the network, as well as their combination (Han et al., 2016) to reduce the memory footprint. On the other hand, works in knowledge distillation leverage student teacher models. These approaches include using soft logits as targets (Ba and Caruana, 2014), increasing the temperature of the softmax to match that of the teacher (Hinton et al., 2015) as well as using teacher representations (Romero et al., 2015) (refer to (Cheng et al., 2017) for a survey).

Recent and concurrent Works: Liu et al. (2019); Zhu et al. (2019); Clark et al. (2019) leverage ensembling to distil knowledge from several multi-task deep neural networks into a single model. Sun et al. (2019); Sanh (2019);Aguilar et al. (2019)[1] train student models leveraging architectural knowledge of the teacher models which adds architectural constraints (e.g., embedding dimension) on

[2]We will release code and distilled model checkpoints.

the student. In order to address this shortcoming, more recent works combine task-specific distillation with pre-training the student model with arbitrary embedding dimension but still relying on transformer architectures (Turc et al., 2019)[1];(Jiao et al., 2019)[1];(Zhao et al., 2019)[1].

Izsak et al. (2019); Shi et al. (2019) extend these for sequence tagging for Part-of-Speech (POS) tagging and Named Entity Recognition (NER) in English. The one closest to our work Tsai et al. (2019) extends the above for multi-lingual NER.

Most of these works rely on general corpora for pre-training and task-specific labeled data for distillation. To harness additional knowledge, (Turc et al., 2019) leverage task-specific unlabeled data. (Tang et al., 2019; Jiao et al., 2019) use rule-and embedding-based data augmentation in absence of such unlabeled data.

3 Models

The Student: The input to the model are E-dimensional word embeddings for each token. In order to capture sequential information in the tokens, we use a single layer Bidirectional Long Short Term Memory Network (BiLSTM). Given a sequence of K tokens, a BiLSTM computes a set of K vectors $h(x_k) = [\overrightarrow{h(x_k)}; \overleftarrow{h(x_k)}]$ as the concatenation of the states generated by a forward ($\overrightarrow{h(x_k)}$) and backward LSTM ($\overleftarrow{h(x_k)}$). Assuming the number of hidden units in the LSTM to be H, each hidden state $h(x_k)$ is of dimension $2H$. Probability of the label at timestep t is given by:

$$p^{(s)}(x_k) = softmax(h(x_k) \cdot W^s) \quad (1)$$

where $W^s \in R^{2H.C}$ and C is number of labels.

We train the student network end-to-end minimizing the cross-entropy loss over labeled data:

$$\mathcal{L}_{\mathcal{CE}} = -\sum_{x_l, y_l \in D_l} \sum_k \sum_c y_{k,c,l} \; log \; p_c^{(s)}(x_{k,l}) \quad (2)$$

The Teacher: Pre-trained language models like ELMO (Peters et al., 2018), BERT (Devlin et al., 2019) and GPT (Radford et al., 2018, 2019) have shown state-of-the-art performance for several tasks. We adopt BERT as the teacher – specifically, the multi-lingual version of BERT (MBERT) with $179MM$ parameters trained on top of 104 languages with the largest Wikipedias. MBERT does not use any markers to distinguish languages during pre-training and learns a single language-agnostic model trained via masked language modeling over Wikipedia articles from all languages.

Tokenization: Similar to MBERT, we use WordPiece tokenization with $110K$ shared WordPiece vocabulary. We preserve casing, remove accents, split on punctuations and whitespace.

Fine-tuning the Teacher: The pre-trained language models are trained for general language model objectives. In order to adapt them for the given task, the teacher is fine-tuned end-to-end with task-specific labeled data D_l to learn parameters $\tilde{\theta}^t$ using cross-entropy loss as in Equation 2.

4 Distillation Features

Fine-tuning the teacher gives us access to its task-specific representations for distilling the student model. To this end, we use different kinds of information from the teacher.

Teacher Logits: Logits as logarithms of predicted probabilities provide a better view of the teacher by emphasizing on the different relationships learned by it across different instances. Consider $p^t(x_k)$ to be the classification probability of token x_k as generated by the fine-tuned teacher with $logit(p^t(x_k))$ representing the corresponding logits. Our objective is to train a student model with these logits as targets. Given the hidden state representation $h(x_k)$ for token x_k, we can obtain the corresponding classification score (since targets are logits) as:

$$r^s(x_k) = W^r \cdot h(x_k) + b^r \quad (3)$$

where $W^r \in R^{C \cdot 2H}$ and $b^r \in R^C$ are trainable parameters and C is the number of classes. We want to train the student neural network end-to-end by minimizing the element-wise mean-squared error between the classification scores given by the student and the target logits from the teacher as:

$$\mathcal{L}_{\mathcal{LL}} = \frac{1}{2} \sum_{x_u \in D_u} \sum_k ||r^s(x_{k,u}) - logit(p^t(x_{k,u}; \tilde{\theta}_t))||^2 \quad (4)$$

4.1 Internal Teacher Representations

Hidden representations: Recent works (Sun et al., 2019; Romero et al., 2015) have shown the hidden state information from the teacher to be helpful as a hint-based guidance for the student. Given a large collection of task-specific unlabeled data, we can transfer the teacher's knowledge to the student via its hidden representations. However, this poses a challenge in our setting as the teacher and student models have different architectures with disparate output spaces.

Consider $h^s(x_k)$ and $z_l^t(x_k; \tilde{\theta}_t)$ to be the representations generated by the student and the l^{th} deep layer of the fine-tuned teacher respectively for a token x_k. Consider $x_u \in D_u$ to be the set of unlabeled instances. We will later discuss the choice of the teacher layer l and its impact on distillation.

Projection: To make all output spaces compatible, we perform a non-linear projection of the parameters in student representation h^s to have same shape as teacher representation z_l^t for each token x_k:

$$\tilde{z}^s(x_k) = Gelu(W^f \cdot h^s(x_k) + b^f) \quad (5)$$

where $W^f \in R^{|z_l^t| \cdot 2H}$ is the projection matrix, $b^f \in R^{|z_l^t|}$ is the bias, and *Gelu* (Gaussian Error Linear Unit) (Hendrycks and Gimpel, 2016) is the non-linear projection function. $|z_l^t|$ represents the embedding dimension of the teacher. This transformation aligns the output spaces of the student and teacher and allows us to accommodate arbitrary student architecture. Also note that the projections (and therefore the parameters) are shared across tokens at different timepoints.

The projection parameters are learned by minimizing the KL-divergence (KLD) between the student and the l^{th} layer teacher representations:

$$\mathcal{L}_{\mathcal{RL}} = \sum_{x_u \in D_u} \sum_k KLD(\tilde{z}^s(x_{k,u}), z_l^t(x_{k,u}; \tilde{\theta}_t)) \quad (6)$$

Multi-lingual word embeddings: A large number of parameters reside in the word embeddings. For MBERT a shared multi-lingual WordPiece vocabulary of $V = 110K$ tokens and embedding dimension of $D = 768$ leads to $92MM$ parameters. To have massive compression, we cannot directly incorporate MBERT embeddings in our model. Since we use the same WordPiece vocabulary, we are likely to benefit more from these embeddings than from Glove (Pennington et al., 2014) or FastText (Bojanowski et al., 2016).

We use a dimensionality reduction algorithm like Singular Value Decomposition (SVD) to project the MBERT word embeddings to a lower dimensional space. Given MBERT word embedding matrix of dimension $V \times D$, SVD finds the best E-dimensional representation that minimizes sum of squares of the projections (of rows) to the subspace.

5 Training

We want to optimize the loss functions for *representation* $\mathcal{L}_{\mathcal{RL}}$, *logits* $\mathcal{L}_{\mathcal{LL}}$ and *cross-entropy* $\mathcal{L}_{\mathcal{CE}}$. These optimizations can be scheduled differently to obtain different training regimens as follows.

Algorithm 1: Multi-stage distillation.

```
Fine-tune teacher on D_l and update θ̃^t ;
for stage in {1,2,3} do
    Freeze all layers l ∈ {1 ··· L};
    if stage=1 then
        output = z̃^s(x_u) ;
        target = teacher representations on D_u from
          the l^th layer as z_l^t(x_u; θ̃^t) ;
        loss = R_RL ;
    end
    if stage=2 then
        output = r^s(x_u) ;
        target = teacher logits on D_u as
          logit(p^t(x_u; θ̃^t)) ;
        loss = R_LL ;
    end
    if stage=3 then
        output = p^s(x_l) ;
        target = y_l ∈ D_l ;
        loss = R_CE ;
    end
    for layer l ∈ {L ··· 1} do
        Unfreeze l ;
        Update parameters θ^s_l, θ^s_{l+1} ··· θ^s_L by
          minimizing the optimization loss between
          student output and teacher target
    end
end
```

5.1 Joint Optimization

In this, we optimize the following losses jointly:

$$\frac{1}{|D_l|} \sum_{\{x_l, y_l\} \in D_l} \alpha \cdot \mathcal{L}_{\mathcal{CE}}(x_l, y_l) + \frac{1}{|D_u|} \sum_{\{x_u, y_u\} \in D_u} \Big(\beta \cdot \mathcal{L}_{\mathcal{RL}}(x_u, y_u) + \gamma \cdot \mathcal{L}_{\mathcal{LL}}(x_u, y_u) \Big) \quad (7)$$

where α, β and γ weigh the contribution of different losses. A high value of α makes the student focus more on easy targets; whereas a high value of γ leads focus to the difficult ones. The above loss is computed over two different task-specific data segments. The first part involves cross-entropy loss over labeled data, whereas the second part involves representation and logit loss over unlabeled data.

5.2 Stage-wise Training

Instead of optimizing all loss functions jointly, we propose a stage-wise scheme to gradually transfer most general to task-specific representations from teacher to student. In this, we first train the student to mimic teacher representations from its l^{th} layer by optimizing $\mathcal{R}_{\mathcal{RL}}$ on unlabeled data. The student learns the parameters for word embeddings (θ^w), BiLSTM (θ^b) and projections $\langle W^f, b^f \rangle$.

In the second stage, we optimize for the cross-entropy $\mathcal{R}_{\mathcal{CE}}$ and logit loss $\mathcal{R}_{\mathcal{LL}}$ jointly on both

Dataset	Labels	Train	Test	Unlabeled
NER				
Wikiann-41	11	705K	329K	7.2MM
Classification				
IMDB	2	25K	25K	50K
DBPedia	14	560K	70K	-
AG News	4	120K	7.6K	-
Elec	2	25K	25K	200K

Table 1: Full dataset summary.

labeled and unlabeled data respectively to learn the corresponding parameters W^s and $\langle W^r, b^r \rangle$.

The above can be further broken down in two stages, where we sequentially optimize logit loss $\mathcal{R}_{\mathcal{LL}}$ on unlabeled data and then optimize cross-entropy loss $\mathcal{R}_{\mathcal{CE}}$ on labeled data. Every stage learns parameters conditioned on those learned in previous stage followed by end-to-end fine-tuning.

5.3 Gradual Unfreezing

One potential drawback of end-to-end fine-tuning for stage-wise optimization is 'catastrophic forgetting' (Howard and Ruder, 2018) where the model forgets information learned in earlier stages. To address this, we adopt gradual unfreezing – where we tune the model one layer at a time starting from the configuration at the end of previous stage.

We start from the top layer that contains the most task-specific information and allow the model to configure the task-specific layer first while others remain frozen. The latter layers are gradually unfrozen one by one and the model trained till convergence. Once a layer is unfrozen, it maintains the state. When the last layer (word embeddings) is unfrozen, the entire network is trained end-to-end. The order of this unfreezing scheme (top-to-bottom) is reverse of that in (Howard and Ruder, 2018) and we find this to work better in our setting with the following intuition. At the end of the first stage on optimizing $\mathcal{R}_{\mathcal{RL}}$, the student learns to generate representations similar to that of the l^{th} layer of the teacher. Now, we need to add only a few task-specific parameters ($\langle W^r, b^r \rangle$) to optimize for logit loss $\mathcal{R}_{\mathcal{LL}}$ with all others frozen. Next, we *gradually* give the student more flexibility to optimize for task-specific loss by tuning the layers below where the number of parameters increases with depth ($|\langle W^r, b^r \rangle| << |\theta_b| << |\theta_w|$).

We tune each layer for n epochs and restore model to the best configuration based on validation loss on a held-out set. Therefore, the model retains best possible performance from any iteration. Algorithm 1 shows overall processing scheme.

Work	PT	TA	Distil.
Sanh (2019)	Y	Y	D1
Turc et al. (2019)	Y	N	D1
Liu et al. (2019); Zhu et al. (2019); Shi et al. (2019); Tsai et al. (2019); Tang et al. (2019); Izsak et al. (2019); Clark et al. (2019)	N	N	D1
Sun et al. (2019)	N	Y	D2
Jiao et al. (2019)	N	N	D2
Zhao et al. (2019)	Y	N	D2
TinyMBERT (ours)	N	N	D4

Table 2: Different distillation strategies. D1 leverages soft logits with hard labels. D2 uses representation loss. PT denotes pre-training with language modeling. TA depicts students constrained by teacher architecture.

6 Experiments

Dataset Description: We evaluate our model TinyMBERT for multi-lingual NER on 41 languages and the same setting as in (Rahimi et al., 2019). This data has been derived from the WikiAnn NER corpus (Pan et al., 2017) and partitioned into training, development and test sets. All the NER results are reported in this test set for a fair comparison between existing works. We report both the average F_1-score (μ) and standard deviation σ between scores across 41 languages for phrase-level evaluation. Refer to Figure 2 for languages codes and distribution of training labels across languages.

We also perform experiments with data from four other domains (refer to Table 1): IMDB (Maas et al.), SST-2 (Socher et al., 2013) and Elec (McAuley and Leskovec) for sentiment analysis for movie and electronics product reviews, DbPedia (Zhang et al.) and Ag News (Zhang et al.) for topic classification of Wikipedia and news articles.
NER Tags: The NER corpus uses IOB2 tagging strategy with entities like LOC, ORG and PER. Following MBERT, we do not use language markers and share these tags across all languages. We use additional syntactic markers like {CLS, SEP, PAD} and 'X' for marking segmented wordpieces contributing a total of 11 tags (with shared 'O').

6.1 Evaluating Distillation Strategies

Baselines: A trivial baseline (D0) is to learn models *one per language* using only corresponding labels for learning. This can be improved by merging all instances and sharing information across all languages (D0-S). Most of the concurrent and recent works (refer to Table 2 for an overview) leverage logits as optimization targets for distillation (D1).

Strategy	Features	Transfer = 0.7MM	Transfer = 1.4MM	Transfer = 7.2MM
D0	Labels per lang.	71.26 (6.2)	-	-
D0-S	Labels across all lang.	81.44 (5.3)	-	-
D1	Labels and Logits	82.74 (5.1)	84.52 (4.8)	85.94 (4.8)
D2	Labels, Logits and Repr.	82.38 (5.2)	83.78 (4.9)	85.87 (4.9)
D3.1	(S1) Repr. (S2) Labels and Logits	83.10 (5.0)	84.38 (5.1)	86.35 (4.9)
D3.2	+ Gradual unfreezing	86.77 (4.3)	87.79 (4.0)	88.26 (4.3)
D4.1	(S1) Repr. (S2) Logits (S3) Labels	84.82 (4.7)	87.07 (4.2)	87.87 (4.1)
D4.2	+ Gradual unfreezing	87.10 (4.2)	**88.64 (3.8)**	88.52 (4.1)

Table 3: Comparison of several strategies with average F_1-score (and standard deviation) across 41 languages over different transfer data size. S_i depicts separate stages and corresponding optimized loss functions.

A few exceptions also use teacher internal representations along with soft logits (D2). For our model we consider multi-stage distillation, where we first optimize representation loss followed by jointly optimizing logit and cross-entropy loss (D3.1) and further improving it by gradual unfreezing of neural network layers (D3.2). Finally, we optimize the loss functions sequentially in three stages (D4.1) and improve it further by unfreezing mechanism (D4.2). We further compare all strategies while varying the amount of unlabeled transfer data for distillation (hyper-parameter settings in Appendix).

Results: From Table 3, we observe all strategies that share information across languages to work better (D0-S vs. D0) with the soft logits adding more value than hard targets (D1 vs. D0-S). Interestingly, we observe simply combining representation loss with logits (D3.1 vs. D2) hurts the model. We observe this strategy to be vulnerable to the hyper-parameters (α, β, γ in Eqn. 7) used to combine multiple loss functions. We vary hyper-parameters in multiples of 10 and report best numbers.

Stage-wise optimizations remove these hyper-parameters and improve performance. We also observe the gradual unfreezing scheme to improve both stage-wise distillation strategies significantly.

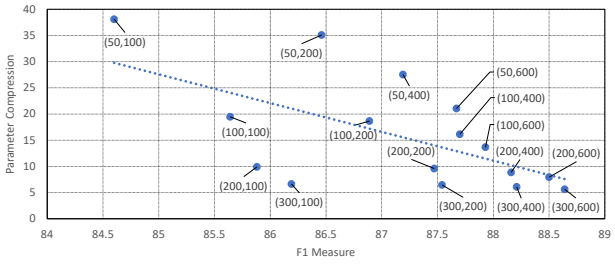
Focusing on the data dimension, we observe all models to improve as more and more unlabeled data is used for transferring teacher knowledge to student. However, we also observe the improvement to slow down after a point where additional unlabeled data does not yield significant benefits. Table 4 shows the gradual performance improvement in TinyMBERT after every stage and unfreezing various neural network layers.

6.2 Performance, Compression and Speedup

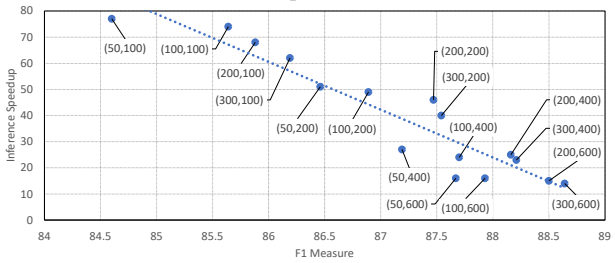
Performance: We observe TinyMBERT in Table 5 to perform competitively with other models. MBERT-single models are fine-tuned per language

Stage	Unfreezing Layer	F_1	Std. Dev.
2	Linear ($\langle W^r, b^r \rangle$)	0	0
2	Projection ($\langle W^f, b^f \rangle$)	2.85	3.9
2	BiLSTM (θ_b)	81.64	5.2
2	Word Emb (θ_w)	85.99	4.4
3	Softmax (W^s)	86.38	4.2
3	Projection ($\langle W^f, b^f \rangle$)	87.65	3.9
3	BiLSTM (θ_b)	88.08	3.9
3	Word Emb (θ_w)	88.64	3.8

Table 4: Gradual F_1-score improvement over multiple distillation stages in TinyMBERT.



(a) Parameter compression vs. F_1-score.



(b) Inference speedup vs. F_1-score.

Figure 1: Variation in TinyMBERT F_1-score with parameter and latency compression against MBERT. Each point in the linked scatter plots represents a configuration with corresponding embedding dimension and BiLSTM hidden states as (E, H).

Model	Avg. F_1	Std. Dev
MBERT-single (Devlin et al., 2019)	90.76	3.1
MBERT (Devlin et al., 2019)	91.86	2.7
MMNER (Rahimi et al., 2019)	89.20	2.8
TinyMBERT (ours)	88.64	3.8

Table 5: F_1-score comparison of different models with standard deviation across 41 languages.

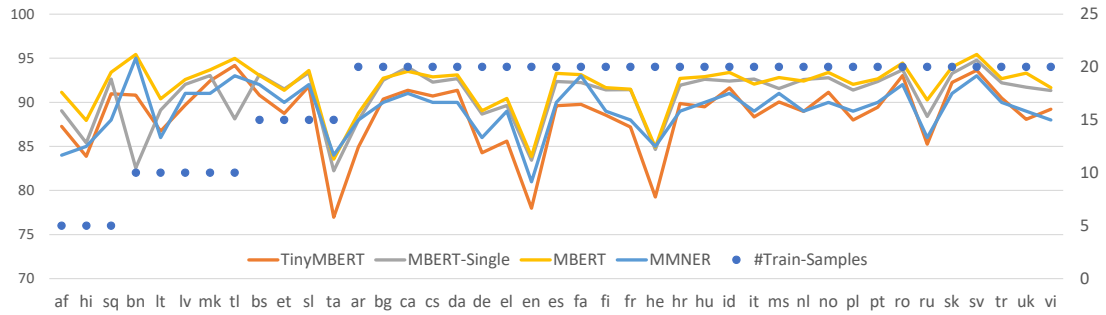


Figure 2: F_1-score comparison for different models across 41 languages. The y-axis on the left shows the scores, whereas the axis on the right (plotted against blue dots) shows the number of training labels (in thousands).

with corresponding labels, whereas MBERT is fine-tuned with data across all languages. MMNER results are reported from Rahimi et al. (2019).

Figure 2 shows the variation in F_1-score across different languages with variable amount of training data for different models. We observe all the models to follow the general trend with some aberrations for languages with less training labels.

Parameter compression: TinyMBERT performs at par with MMNER obtaining atleast $41x$ compression by learning a single model across all languages as opposed to learning language-specific models.

Figure 1a shows the variation in F_1-scores of TinyMBERT and compression against MBERT with different configurations corresponding to the embedding dimension (E) and number of BiLSTM hidden states ($2 \times H$). We observe that reducing the embedding dimension leads to great compression with minimal performance loss. Whereas, reducing the BiLSTM hidden states impacts the performance more and contributes less to the compression.

Inference speedup: We compare the runtime inference efficiency of MBERT and our model in a single P100 GPU for batch inference (batch size = 32) on 1000 queries of sequence length 32. We average the time taken for predicting labels for all the queries for each model aggregated over 100 runs. Compared to batch inference, the speedups are less for online inference (batch size = 1) at $17x$ on Intel(R) Xeon(R) CPU (E5-2690 v4 @2.60GHz) (refer to Appendix for details).

Figure 1b shows the variation in F_1-scores of TinyMBERT and inference speedup against MBERT with different (linked) parameter configurations as before. As expected, the performance degrades with gradual speedup. We observe that parameter compression does not necessarily lead to an inference speedup. Reduction in the word embedding dimension leads to massive model compression, however, it does not have a similar effect on the latency. The BiLSTM hidden states, on the other hand, constitute the real latency bottleneck. One of the best configurations leads to $35x$ compression, $51x$ speedup over MBERT retaining nearly 95% of its performance.

Model	#Transfer Samples	F_1
MMNER	-	62.1
MBERT	-	79.54
TinyMBERT	4.1K	19.12
	705K	76.97
	1.3MM	77.17
	7.2MM	77.26

Table 6: F_1-score comparison for low-resource setting with 100 labeled samples per language and transfer set of different sizes for TinyMBERT.

6.3 Low-resource NER and Distillation

Models in all prior experiments are trained on $705K$ labeled instances across all languages. In this setting, we consider only 100 labeled samples for each language with a total of $4.1K$ instances. From Table 6, we observe MBERT to outperform MMNER by more than 17 percentage points with TinyMBERT closely following suit.

Furthermore, we observe our model's performance to improve with the transfer set size depicting the importance of unlabeled transfer data for knowledge distillation. As before, a lot of additional data has marginal contribution.

6.4 Word Embeddings

Random initialization of word embeddings works well. Multi-lingual $300d$ FastText embeddings (Bojanowski et al., 2016) led to minor improvement due to 38% overlap between FastText tokens and MBERT wordpieces. English $300d-$Glove does much better. We experiment with recent dimensionality reduction techniques and find SVD to work better. Surprisingly, it leads to marginal improvement over MBERT embeddings before reduction. As expected, MBERT embeddings after fine-tuning perform better than that from pre-trained checkpoints (refer to Appendix for F_1-measures).

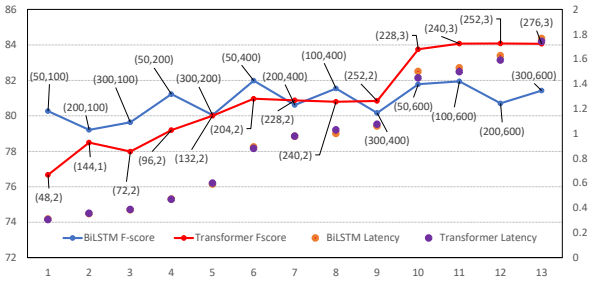


Figure 3: BiLSTM and Transformer F_1-score (left y-axis) vs. inference latency (right y-axis) in 13 different settings with corresponding embedding dimension and width / depth of the student as $(E, W/D)$.

Model	Transfer Set	Acc.
BERT Large Teacher	-	94.95
TinyBERT	SST+Imdb	93.35
BERT Base Teacher	-	92.78
TinyBERT	SST+Imdb	92.89
Sun et al. (2019)	SST	92.70
Turc et al. (2019)	SST+IMDB	91.10

Table 7: Model accuracy on of SST-2 (dev. set).

6.5 Architectural Considerations

Which teacher layer to distil from? The topmost teacher layer captures more task-specific knowledge. However, it may be difficult for a shallow student to capture this knowledge given its limited capacity. On the other hand, the less-deep representations at the middle of teacher model are easier to mimic by shallow student. We observe the student to benefit most from distilling the 6^{th} or 7^{th} layer of the teacher (results in Appendix).

Which student architecture to use for distillation? Recent works in distillation leverage both BiLSTM and Transformer as students. In this experiment, we vary the embedding dimension and hidden states for BiLSTM-, and embedding dimension and depth for Transformer-based students to obtain configurations with similar inference latency. Each of 13 configurations in Figure 3 depict F_1-scores obtained by students of different architecture but similar latency – for strategy D0-S in Table 3. We observe that for low-latency configurations BiLSTMs with hidden states $\{2\times100, 2\times200\}$ work better than 2-layer Transformers. Whereas, the latter starts performing better with more than 3-layers although with a higher latency.

6.6 Distillation for Text Classification

We switch gear and focus on classification tasks. In contrast to sequence tagging, we use the last hidden state of the BiLSTM as the final sentence representation for projection, regression and softmax.

Comparison with baselines: Since we focus only on single instance classification in this work, SST-2 (Socher et al., 2013) is the only GLUE benchmark to compare against other distillation techniques. Table 7 shows the accuracy comparison with such methods reported in SST-2 development set.

Dataset	Student no distil.	Distil (Base)	Distil (Large)	BERT Base	BERT Large
Ag News	89.71	92.33	94.33	92.12	**94.63**
IMDB	89.37	91.22	91.70	91.70	**93.22**
Elec	90.62	93.55	93.56	93.46	**94.27**
DbPedia	98.64	99.10	99.06	**99.26**	99.20

Table 8: Distillation performance with BERT.

Dataset	Student no distil.	Student with distil.	BERT Large
AG News	85.85	**90.45**	90.36
IMDB	61.53	89.08	**89.11**
Elec	65.68	**91.00**	90.41
DBpedia	96.30	**98.94**	**98.94**

Table 9: Distillation with BERT Large on 500 labeled samples per class.

We extract $11.7MM$ sentences from all IMDB movie reviews in Table 1 to form the unlabeled transfer set for distillation. We obtain the best performance on distilling with BERT Large (uncased, whole word masking model) than BERT Base – demonstrating a better student performance with a better teacher and outperforming other methods.

Other classification tasks: Table 8 shows the distillation performance of TinyBERT with different teachers. We observe the student to almost match the teacher performance. The performance also improves with a better teacher, although the improvement is marginal as the student model saturates.

Table 9 shows the distillation performance with only 500 labeled samples per class. The distilled student improves over the non-distilled version by 19.4 percent and matches the teacher performance for all of the tasks demonstrating the impact of distillation for low-resource settings.

7 Conclusions

We develop a multi-stage distillation framework for massive multi-lingual NER and classification that performs close to huge pre-trained models with a massive compression and inference speedup. Our distillation strategy leveraging teacher representations agnostic of its architecture and stage-wise optimization schedule outperforms existing ones. We perform extensive study of several hitherto less explored distillation dimensions like the impact of unlabeled transfer set, embeddings and student architectures, and make interesting observations.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Edward Guo. 2019. Knowledge distillation from internal representations.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2654–2662.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. *CoRR*, abs/1710.09282.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. Bam! born-again multi-task networks for natural language understanding. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Peter Izsak, Shira Guskin, and Moshe Wasserblat. 2019. Training compact models for low resource entity tagging using pre-trained language models.

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

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *CoRR*, abs/1904.09482.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 2011, Portland, Oregon, USA*.

Julian J. McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Victor Sanh. 2019. Introducing distilbert, a distilled version of bert. https://medium.com/huggingface/distilbert-8cf3380435b5.

Yangyang Shi, Mei-Yuh Hwang, Xin Lei, and Haoyu Sheng. 2019. Knowledge distillation for recurrent neural network language modeling with trust regularization. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from BERT into simple neural networks. *CoRR*, abs/1903.12136.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical bert models for sequence labeling. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*.

Sanqiang Zhao, Raghav Gupta, Yang Song, and Denny Zhou. 2019. Extreme language model compression with optimal subwords and shared projections.

Wei Zhu, Xiaofeng Zhou, Keqiang Wang, Xun Luo, Xiepeng Li, Yuan Ni, and Guotong Xie. 2019. PANLP at MEDIQA 2019: Pre-trained language models, transfer learning and knowledge distillation. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 380–388, Florence, Italy. Association for Computational Linguistics.

A Appendices

A.1 Implementation

The model uses Tensorflow backend. Implementation code is included in the supplementary.

A.2 Parameter Configurations

All the analyses in the paper — *except* compression and speedup experiments that vary embedding dimension E and BiLSTM hidden states H — are done with the following model configuration in Table 10 with the best F_1-score. Optimizer Adam is used with cosine learning rate scheduler ($lr\_high = 0.001, lr\_low = 1e-8$).

The model corresponding to the $35x$ parameter compression and $51x$ speedup for batch inference uses $E = 50$ and $H = 2 \times 200$.

Parameter	Value
SVD + MBERT word emb. dim.	$E = 300$
BiLSTM hidden states	$H = 2 \times 600$
Dropout	0.2
Batch size	512
Teacher layer	7
Optimizer	Adam

Table 10: TinyMBERT config. with best $F_1 = 88.64$.

Following hyper-parameter tuning was done to select dropout rate and batch size at the start of the parameter tuning process.

Dropout Rate	F_1-score
1e-4	87.94
0.1	88.36
0.2	**88.49**
0.3	88.46
0.6	87.26
0.8	85.49

Table 11: Impact of dropout.

Batch size	F_1-score
128	87.96
512	**88.4**
1024	88.24
2048	88.13
4096	87.63

Table 12: Impact of batch size.

Layer (l)	F-score	Std. Dev.
11	88.46	3.8
9	88.31	3.8
7	**88.64**	**3.8**
6	88.64	3.8
4	88.19	4
2	88.50	4
1	88.51	4

Table 13: Comparison of TinyMBERT F-score and standard deviation on distilling representations from l^{th} MBERT layer.

Word Embedding	F-score	Std. Dev.
SVD + MBERT (fine-tuned)	**88.64**	**3.8**
MBERT (fine-tuned)	88.60	3.9
SVD + MBERT (pre-trained)	88.54	3.9
PCA + PPA (d=14) (Raunak et al., 2019)	88.35	3.9
PCA + PPA (d=17) (Raunak et al., 2019)	88.25	4.0
Glove (Pennington et al., 2014)	88.16	4.0
FastText (Bojanowski et al., 2016)	87.91	3.9
Random	87.43	4.1

Table 14: Impact of using various word embeddings for initialization on multi-lingual distillation. SVD, PCA, and Glove uses 300-dimensional word embeddings.

BiLSTM					Transformer				
Emb	Hidden	F1	Params (MM)	Latency	Emb	Depth	Params (MM)	Latency	F1
50	100	80.26	4.7	0.311	48	2	4.4	0.307	76.67
200	100	79.21	18.1	0.354	144	1	13.4	0.357	78.49
300	100	79.63	27	0.385	72	2	6.7	0.388	77.98
50	200	81.22	5.1	0.472	96	2	9	0.47	79.19
300	200	80.04	27.7	0.593	132	2	12.5	0.6	80
50	400	81.98	6.5	0.892	204	2	19.7	0.88	80.96
200	400	80.61	20.2	0.978	228	2	22.1	0.979	80.87
100	400	81.54	11.1	1	240	2	23.3	1.03	80.79
300	400	80.16	29.4	1.06	252	2	24.6	1.075	80.84
50	600	81.78	8.5	1.5	228	3	22.7	1.448	83.75
100	600	81.94	13.1	1.53	240	3	24	1.498	84.07
200	600	80.7	22.5	1.628	252	3	25.3	1.591	84.08
300	600	81.42	31.8	1.766	276	3	28	1.742	84.06

Table 15: BiLSTM and Transformer configurations (with varying embedding dimension, hidden states and depth) vs. latency and F_1 scores for distillation strategy $D0 - S$.

Embedding	BiLSTM	Fscore	Std. Dev.	Params (MM)	Params(Compression)	Speedup (bsz=32)	Speedup (bsz=1)
300	600	88.64	3.8	31.8	5.6	14	8
200	600	88.5	3.8	22.5	8	15	9
300	400	88.21	4	29.4	6.1	23	11
200	400	88.16	3.9	20.2	8.9	25	12
100	600	87.93	4.1	13.1	13.7	16	9
100	400	87.7	4	11.1	16.1	24	13
50	600	87.67	4	8.5	21.1	16	10
300	200	87.54	4.1	27.7	6.5	40	15
200	200	87.47	4.2	18.7	9.6	46	16
50	400	87.19	4.3	6.5	27.5	27	13
100	200	86.89	4.2	9.6	18.6	49	15
50	200	86.46	4.3	5.1	35.1	51	16
300	100	86.19	4.3	27	6.6	62	16
200	100	85.88	4.4	18.1	9.9	68	17
100	100	85.64	4.5	9.2	19.5	74	15
50	100	84.6	4.7	4.7	38.1	77	16

Table 16: Parameter compression and inference speedup vs. F_1-score with varying embedding dimension and BiLSTM hidden states. Online inference is in Intel(R) Xeon(R) CPU (E5-2690 v4 @2.60GHz) and batch inference is in a single P100 GPU for distillation strategy $D4$.

Lang	#Train-Samples	TinyMBERT	MBERT-Single	MBERT	MMNER
af	5	87	89	91	84
hi	5	84	85	88	85
sq	5	91	93	93	88
bn	10	91	83	95	95
lt	10	87	89	90	86
lv	10	90	92	93	91
mk	10	92	93	94	91
tl	10	94	88	95	93
bs	15	91	93	93	92
et	15	89	92	91	90
sl	15	92	93	94	92
ta	15	77	82	84	84
ar	20	85	88	89	88
bg	20	90	93	93	90
ca	20	91	94	93	91
cs	20	91	92	93	90
da	20	91	93	93	90
de	20	84	89	89	86
el	20	86	90	90	89
en	20	78	83	84	81
es	20	90	92	93	90
fa	20	90	92	93	93
fi	20	89	91	92	89
fr	20	87	91	91	88
he	20	79	85	85	85
hr	20	90	92	93	89
hu	20	90	93	93	90
id	20	92	92	93	91
it	20	88	93	92	89
ms	20	90	92	93	91
nl	20	89	93	92	89
no	20	91	93	93	90
pl	20	88	91	92	89
pt	20	89	92	93	90
ro	20	93	94	94	92
ru	20	85	88	90	86
sk	20	92	93	94	91
sv	20	94	95	95	93
tr	20	90	92	93	90
uk	20	88	92	93	89
vi	20	89	91	92	88

Table 17: F_1-scores of different models per language.