Training Compact Models for Low Resource Entity Tagging using Pre-trained Language Models

Peter Izsak, Shira Guskin, Moshe Wasserblat

Intel AI Lab

EMC² Workshop @ NeurIPS 2019

Motivation

- Named Entity Recognition (NER) is a widely used Information Extraction task in many industrial applications and use cases
- Ramping up on a new domain can be difficult
  - Lots of *unlabeled* data, little of no *labeled* data and often not good enough for training a model with good performance

Solution A

? Hire a linguist or data scientist to tune/build model

? Hire annotators to label more data or buy similar dataset

? Time/compute resource limitations

Solution B

? Pre-trained Language Models such as BERT, GPT, ELMo are great at low-resource scenarios

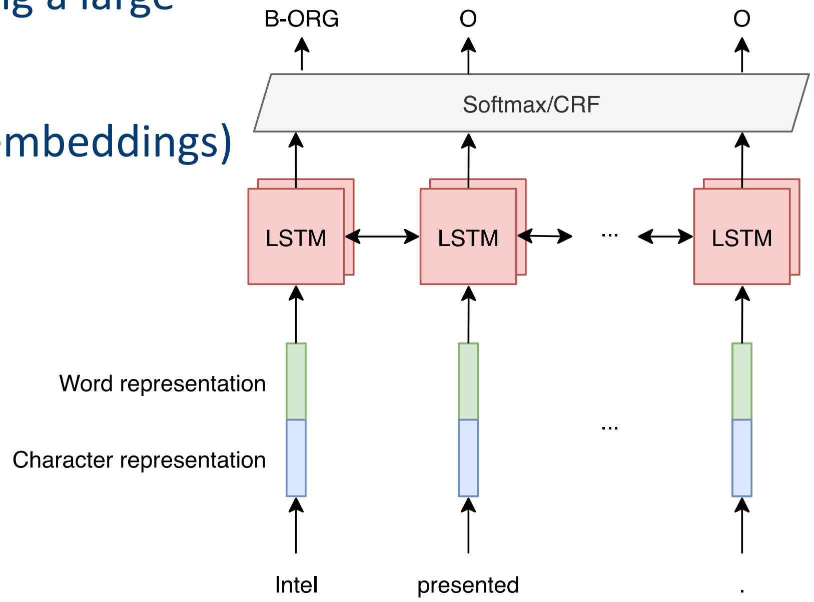
? Require great compute and memory resources and suffer from high latency in inference

? Deploying such models in production or on edge devices is a *major issue*

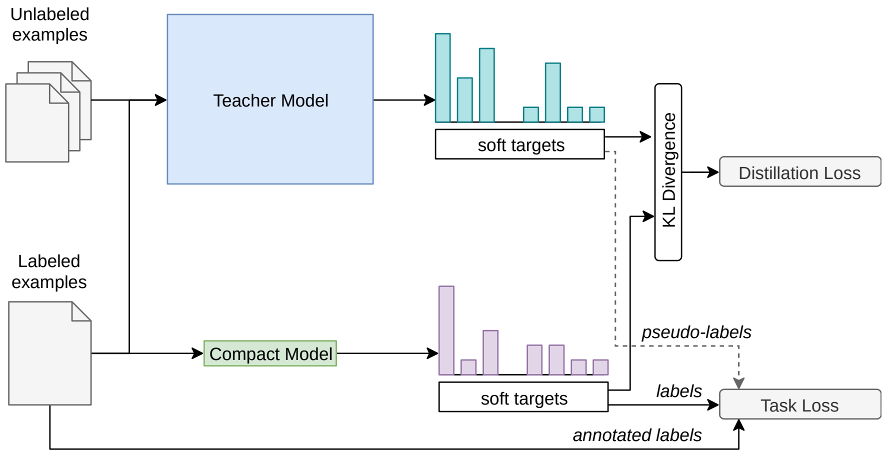
Enhancing a Compact Model

ONGOING WORK-IN-PROGRESS

- Approach:
  - Train a compact model (3M parameters) using a large pre-trained
  - Pre-trained word embeddings (non-shared embeddings)
  - Utilize *labeled* and *unlabeled* data:
    - *Knowledge Distillation*
    - *Pseudo-labeling*



Model training setup



Integrated model knowledge distillation and pseudo-labeling in loss function

$$L_{task} = \begin{cases} \text{CrossEntropy}(\hat{y}, y) & \textit{labeled example} \\ \text{CrossEntropy}(\hat{y}, \hat{y}_{teacher}) & \textit{unlabeled example} \end{cases}$$

$$L_{distillation} = \text{KL}(logits_{teacher} || logits_{compact})$$

$$Loss = \alpha \cdot L_{task} + \beta \cdot L_{distillation}, \qquad \alpha + \beta = 1.0$$

Models

- Teacher – BERT-base/large (110M/340M params.)
- Compact – LSTM-CNN with Softmax/CRF (3M params.)

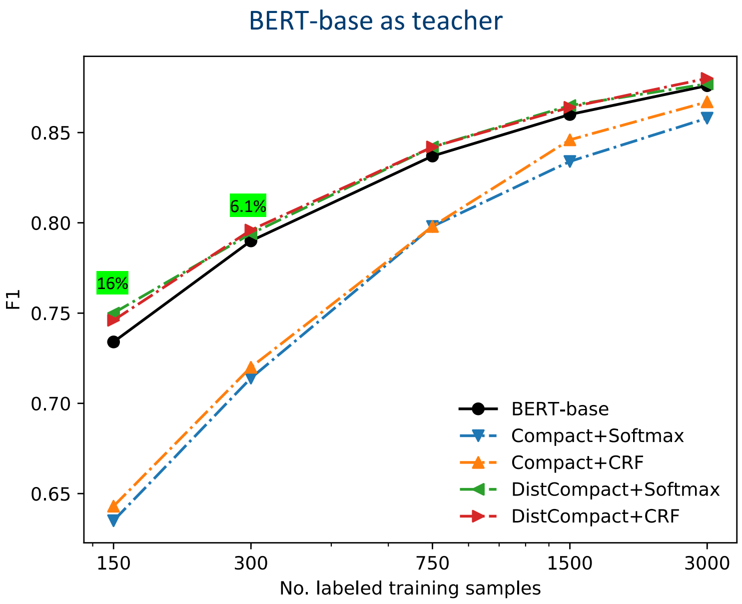
Low-resource Dataset Simulation

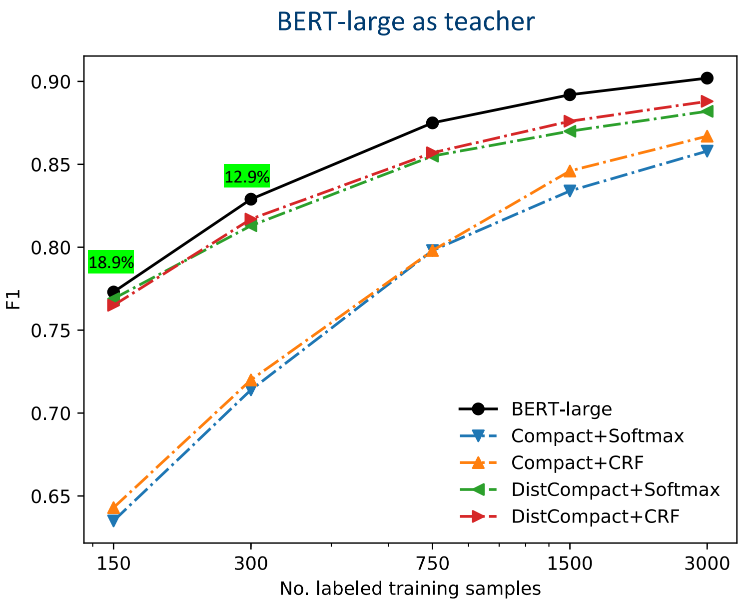
- CoNLL 2003 (English) – `PER/ORG/DATE/MISC`
- Generate random training sets with labeled/unlabeled examples
- Train set size: 150/300/750/1500/3000
- Report averaged F1 (20 experiments per train set size)

Training procedure

1. Fine-tune BERT with labeled data
2. Train compact model using modified loss

Compact model performance





Inference speed on CPU

Batch size	1	32	64	128
Speedup	3.3-4.3	28.6-33.7	40-45.2	49.9-55.6

Batch size	1	32	64	128
Speedup	8.1-10.6	85.2-100.4	109.5-123.8	123.6-137.8

Takeaways

- Compact models perform equally well as pre-trained LM in low-resource scenarios, and with superior inference speed and with compression rate is 36x-113x vs. BERT
- Compact models are preferable for deployment vs. pre-trained LM in such use-cases
- Many directions to explore:
  - Compact model topology – how small/simple can we make the model?
  - Other NLP tasks, pre-trained LM
  - Other ways to utilize unlabeled data
- Code available in Intel AI's NLP Architect open source library

NLP ARCHITECT

NervanaSystems/nlp-architect