On-Device NLP @ Facebook

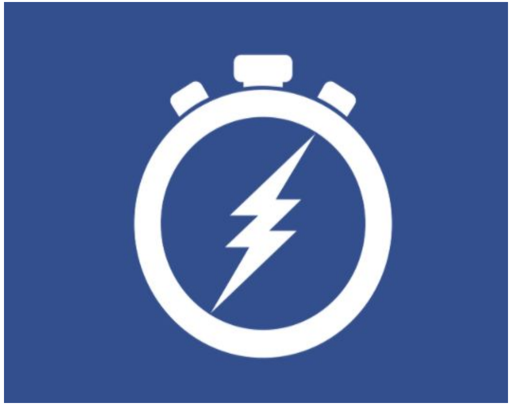
Ahmed Aly, Kshitiz Malik

Agenda

- On-Device NLP: Why?
- Challenges
- On-Device NLP @Facebook: Overview
- Horizontal approaches
- Open problems

On-Device NLP: Why?

Privacy



Latency

Reliability

On-Device NLP: Challenges

- Diverse and Strict compute and memory requirements
  - Diverse set of chipsets with different compute specs
  - Strict memory and compute budgets
- Power consumption and battery considerations
  - Always on Vs Portable
- Toolchain limitations
  - DSP/GPU strict runtime platforms
  - Pytorch Vs Pytorch Edge
  - Tensorflow Vs Tensorflow light
- Model development experience
  - Stricter model releases and deployments
  - Harder benchmarking

On-Device NLP @Facebook: Overview

AI Assistant on Portal

On-Device NLP Tasks

- On-Device Natural Language Understanding (NLU)

Smart Keyboard on Oculus

On-Device NLP Tasks

- On-Device Language Modeling (LM)
- Federated learning

On-Device NLP Research

- Extreme model compression
- Light-weight CNN representations
- Neural architecture search
- On-Device Seq2Seq models (Accepted in NAACL 2021)

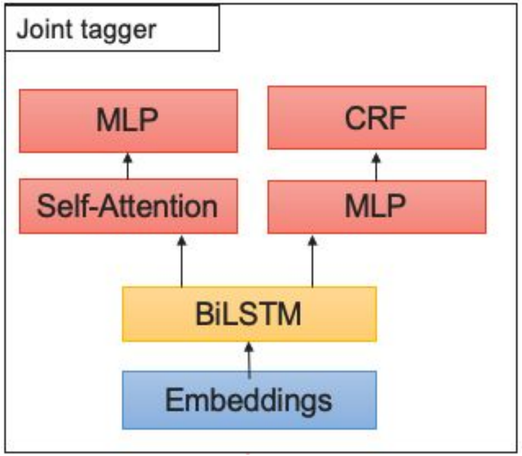
On-Device NLP @Facebook: On-Device NLU on Portal

NLU is the task of converting user utterances to machine understandable representation.

Simple

Call John

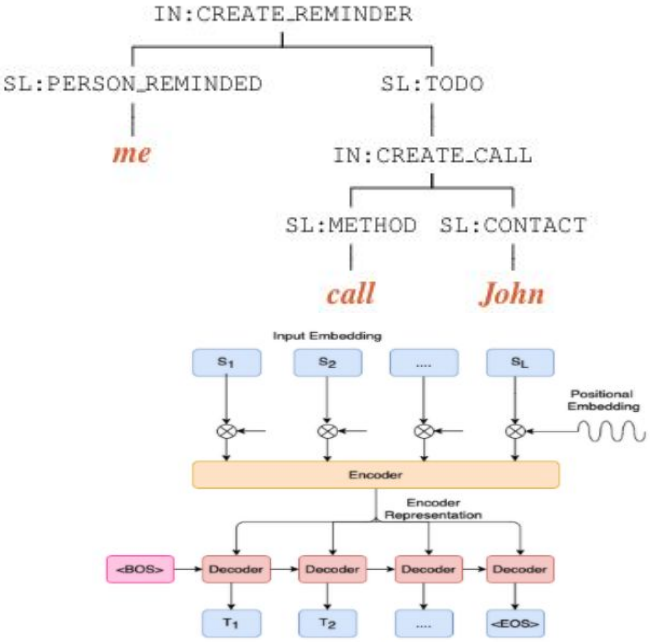
`Call(Contact_name:John)`



Server-side Baseline Model

Hierarchical

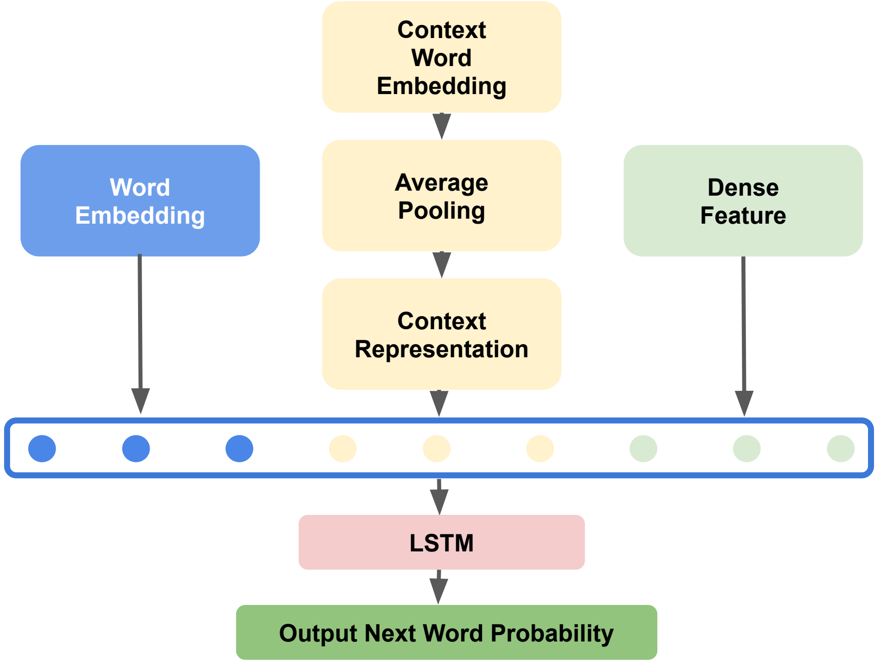
Remind me to call John



Server-side Baseline Model

On-Device NLP @Facebook: On-Device LM on Oculus

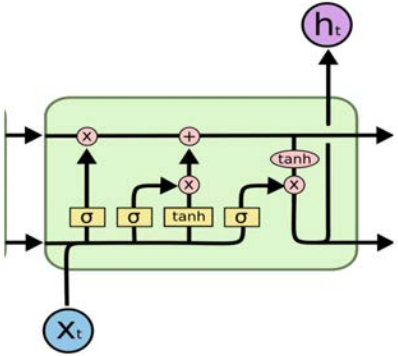
LM (for Smart Keyboard) is the task of predicting the most probable next word given the typed words/characters



Server-side Baseline Model

Horizontal Approaches: Latency

- Recurrence is slow!
  - Encoder: CNNs beat RNNs. LightConv
  - Decoder: Non-autoregressive
- Latency aware Neural Architecture Search (NAS)
- Efficient model layers: question everything
  - Separable Conv Layers
  - Combine Input and Forget Gates
  - Tightly coupled linear layers
- Optimized operator implementation
  - Custom LSTM implementation



Horizontal Approaches: Memory & Tooling

Memory

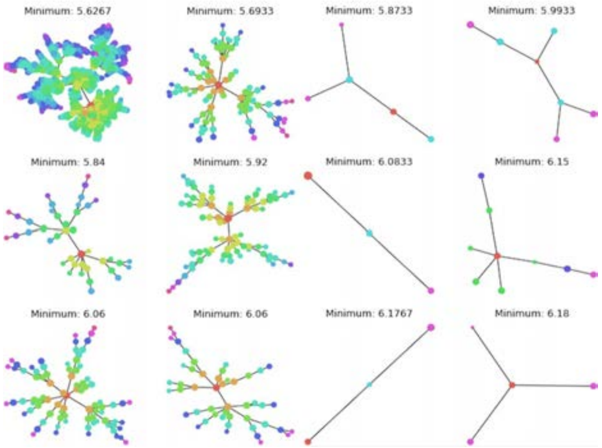
- Byte/Character Embeddings
  - Instead of word/ subword embeddings
- Neural Architecture Search (NAS)
- Quantization
- Sparsification (storage, bandwidth)

Tooling

- Compilers: PyTorch Mobile, Glow
- Benchmarking: AI Bench

Open Problems: Latency & Memory

- Transformers are slow
  - $O(n^2)$ operations
  - Parallelizable - great for GPUs, not mobile processors
- Non-autoregressive decoding quality
- Word embedding compression
- Graceful accuracy degradation
- Network architecture search
  - Hardware-aware NAS
  - NAS search efficiency



Open Problems: Tooling

- Benchmarking
  - Flops != Latency
  - Benchmarking is unreliable, slow
- Taming heterogeneity
  - Many [DSPs, GPUs, NPUs]
  - APIs: Metal (iOS), Vulcan (Android), OpenGL ES
  - Frameworks: Pytorch Mobile, TF, CoreML
- ML Compilers (Glow/TVM)
- Intermediate Representations ([MLIR])