Hybrid AI for the Web: Requirements

Michael McCool, Geoff Gustafson,
Sudeep Divakaran, Muthaiah Venkatachalam

Intel

4 April 2024, W3C WebML WG

Summary of Feedback on “Hybrid AI” Proposal

https://github.com/webmachinelearning/proposals/issues/5

Models/Use Cases:

- MMS – ASR
- SeamlessM4T – general speech tasks
- Others? LLMs?
  - e.g. Mistral-7b, etc.

Comments

- Two forms of hybrid AI
  - Server OR client
  - Split models
- Meaning of “hybrid”?
  - Can also mean “heterogenous hardware”

Pain points (priority?)

Generally: saving space/download latency

1. Sharing/reusing large models
   - Across sites
2. Same resources at different URLs
3. Same resources in different formats
4. Want to expose "built-in" models
5. Generalizing models
   - Categories (e.g. "en-jp text translation")
   - Versions (e.g. ">= 1.3.*")
6. Need solution that can handle adapters
   - Applications sharing a base model
   - Foundational models are large
   - Adapters are small

Other pain points? Quantized representations?

Summary of Feedback on Proposal (original)

https://github.com/webmachinelearning/proposals/issues/5

Models/Use Cases that would benefit from proposal:

- MMS – ASR
  - Large (1B parameters)
  - Uses adapters for different languages (2M each, over 1000 languages)
- SeamlessM4T – general speech tasks
  - Sub-models for specific tasks: S2ST, S2TT, T2ST, T2TT, ASR
  - Sub-model is a model component, e.g. a partial model

Comments

- Two forms of hybrid AI
  - Server OR client
    - E.g. server as fallback if client not powerful enough
    - Need to check if client is powerful enough *before* download
  - Split models
    - Can also enhance model security (complete model not downloaded)
    - Some privacy advantages for user (raw data not sent to cloud)

Pain points

- Sharing large models across sites
  - General solution is hard -> specific solution for ML probably easier and acceptable
- Generalizing models
  - Use case -> potential model choices
  - Model categories (e.g. any ASR model, any noise reducer, …)
  - Dev prioritization to select among options
  - Versioning (e.g. any model >= 1.3.* is acceptable)
  - However - prompt engineering may be specific to particular versions of LLMs
- Same resources
  - At different URLs -> duplicates in cache when using URL as key (e.g. Service Worker Cache)
  - Specification by URL -> ossification, ecosystem brittleness
  - Encoded in different formats
  - Optimized for different clients
- Want to expose built-in models
  - If models provided by system (OS or browser), should avoid downloads
- Need solution that can handle adapters
  - Multiple applications sharing a base model
  - Foundational models large, adapters are small