Role of ***Language Models*** in ASR

AI4Bharat Speech Team

Why do we need LM?

- Speech = Acoustic + Language
  - How much of ***Language** Characteristics* is captured by ***Acoustic** Models?*
  - *Example #1:*
    - *GT: आज सुनेहरा अवसर है*
    - *Output:* आज सुनहरा अवसर है
  - *Example #2:*
    - *GT: आज सुनेहरा अवसर है*
    - *Output:* आज सुनेहरा अफसर है
  - *Example #3:*
    - *GT: आज सुनेहरा अवसर है*
    - *Output:* आज सुनेहरा अवसर हैं

Spelling Errors!

Homophones!

Inflections!

What LM actually computes?

- Probability (Likelihood) of a sentence:

$$P(w_0 \dots w_i) = P(w_i | w_{i-1} \dots w_0) P(w_{i-1} | w_{i-2} \dots w_0) \dots$$

- We generally limit it to N-1 previous words!

$$P(w_i | w_{i-1} \dots w_0) = P(w_i | w_{i-1} \dots w_{i-n+1})$$

N-gram LM!

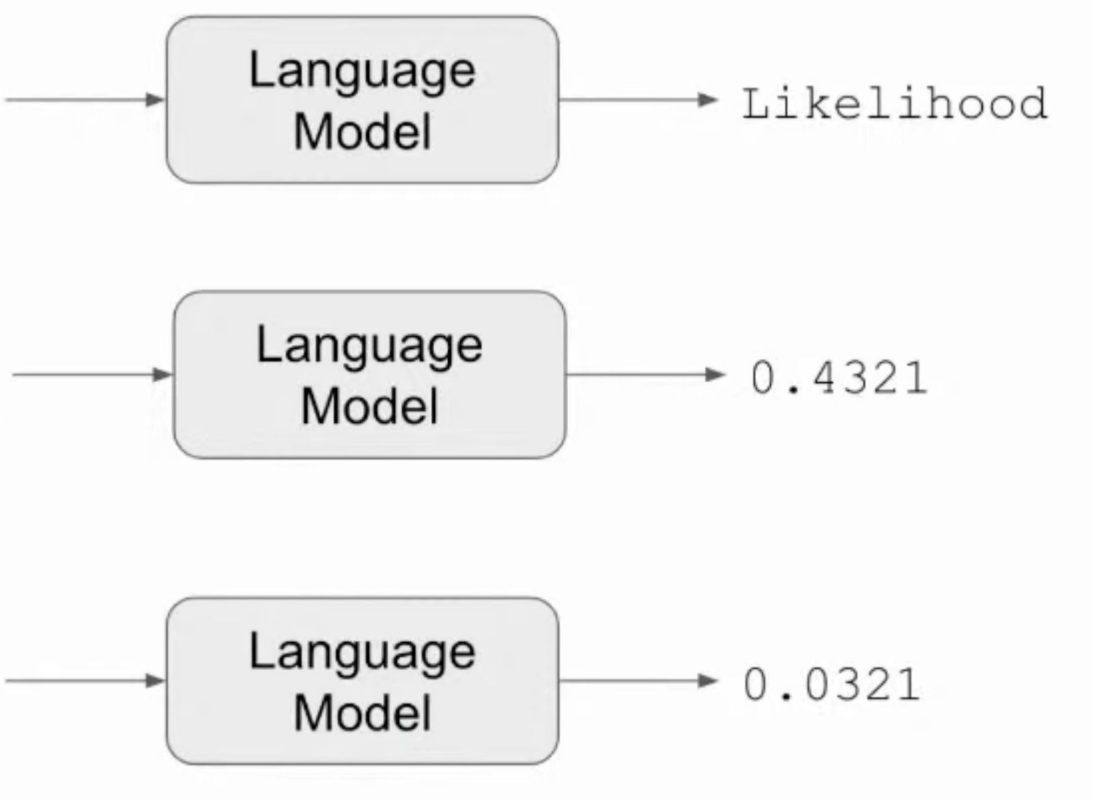
- *Example: Bigram Language Model*
  - text = "<start> आज सुनेहरा अवसर है <end>"
  - P(text) = P(<start> | आज) * P(आज | सुनेहरा) * P(सुनेहरा | अवसर)
    * P(अवसर | है) * P(है | <end>)

What LM actually computes?

Proposed Text

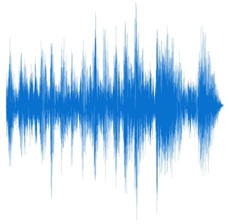
आज सुनेहरा अवसर है

सुनेहरा आज है अवसर

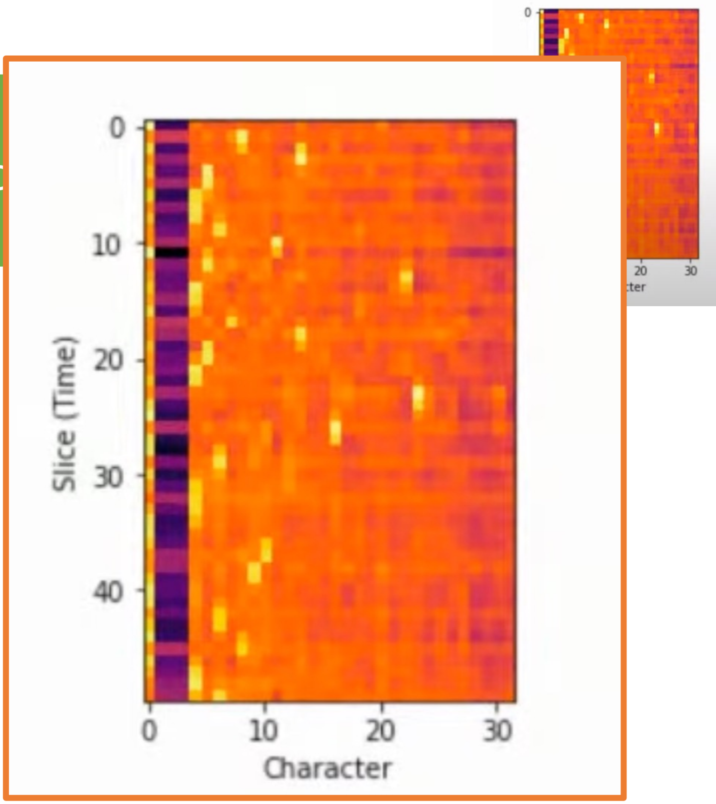


Where do we fit LM?

- Typical ASR pipeline:



Wav → Preprocessing → Audio Features → Acoustic

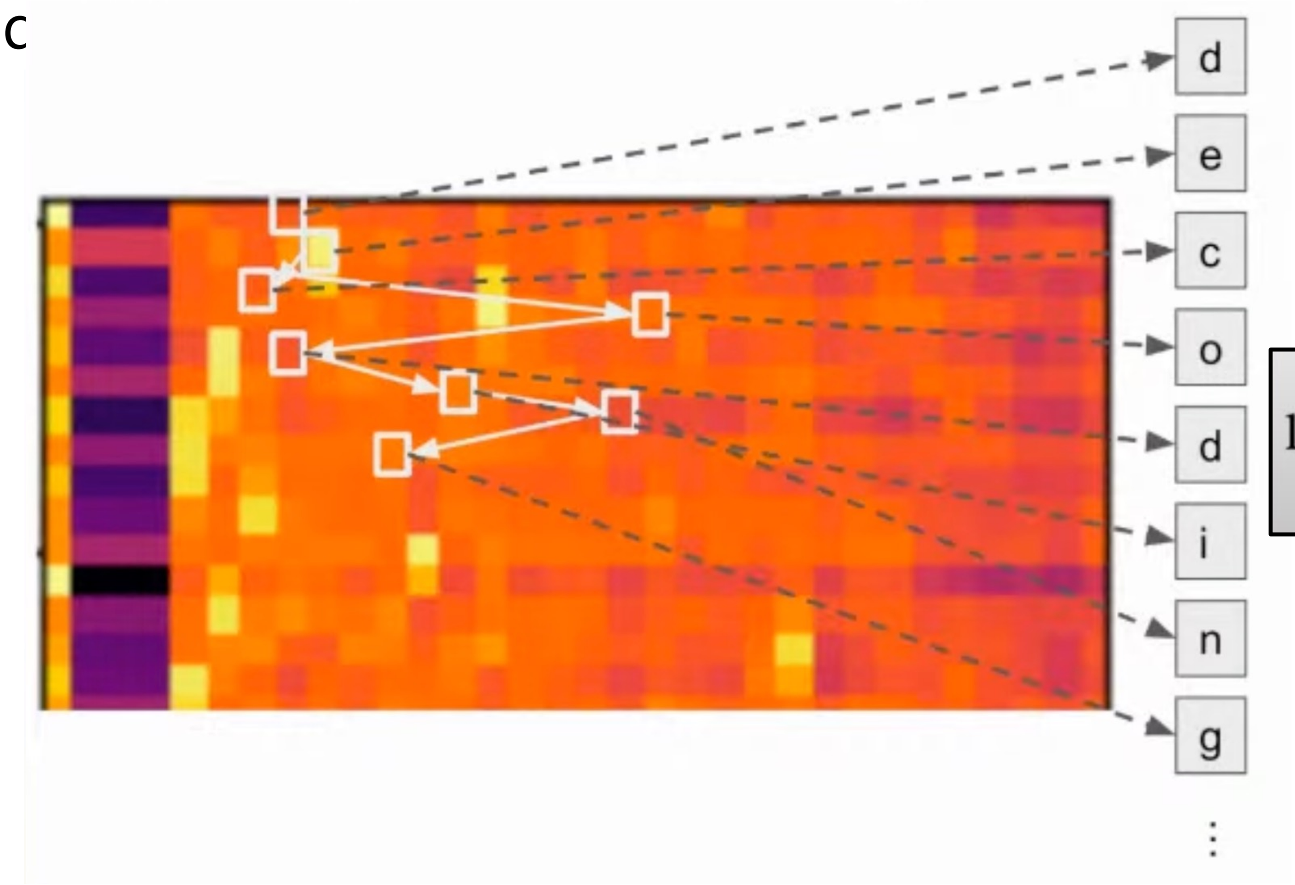


CTC Logit Matrix:

- Leads to softmax output which predicts character probabilities for each timeframe.
- Output Length is NOT proportional to ***Text Length***, but to ***Audio Length***.

Logits Decoding - Concept

- Decoder evaluates Paths through the logit matrix:
  - Greec



$$\log p(\text{path}) = \sum_{i=0}^{n-1} \log \text{softmax}(\text{logits})_{ij_i}$$

i: Time index
j: Character index

Logits Decoding – with LM

- Incorporate LM scores with logits while decoding

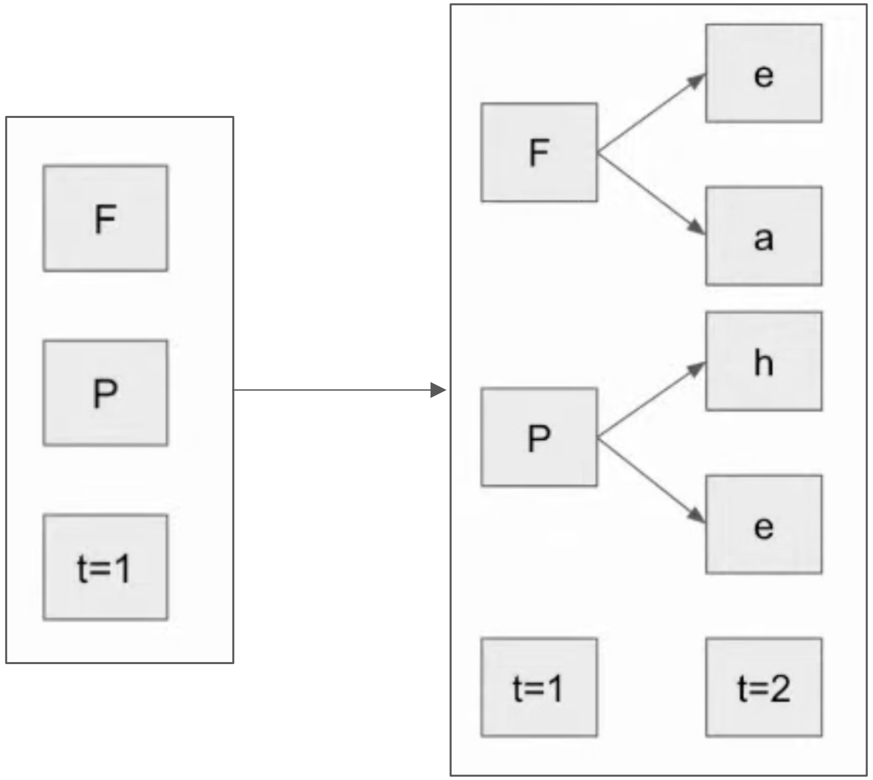
$$\log P_{\mathrm{LM}}(\text{text}) = \log P(\text{text}) + LM(\text{text})$$

- Exact/True Solution
  - Score every possible path through logit matrix with addition of LM
  - Combine scores of equivalent paths
  - Take the highest-scoring text

Extremely Slow!

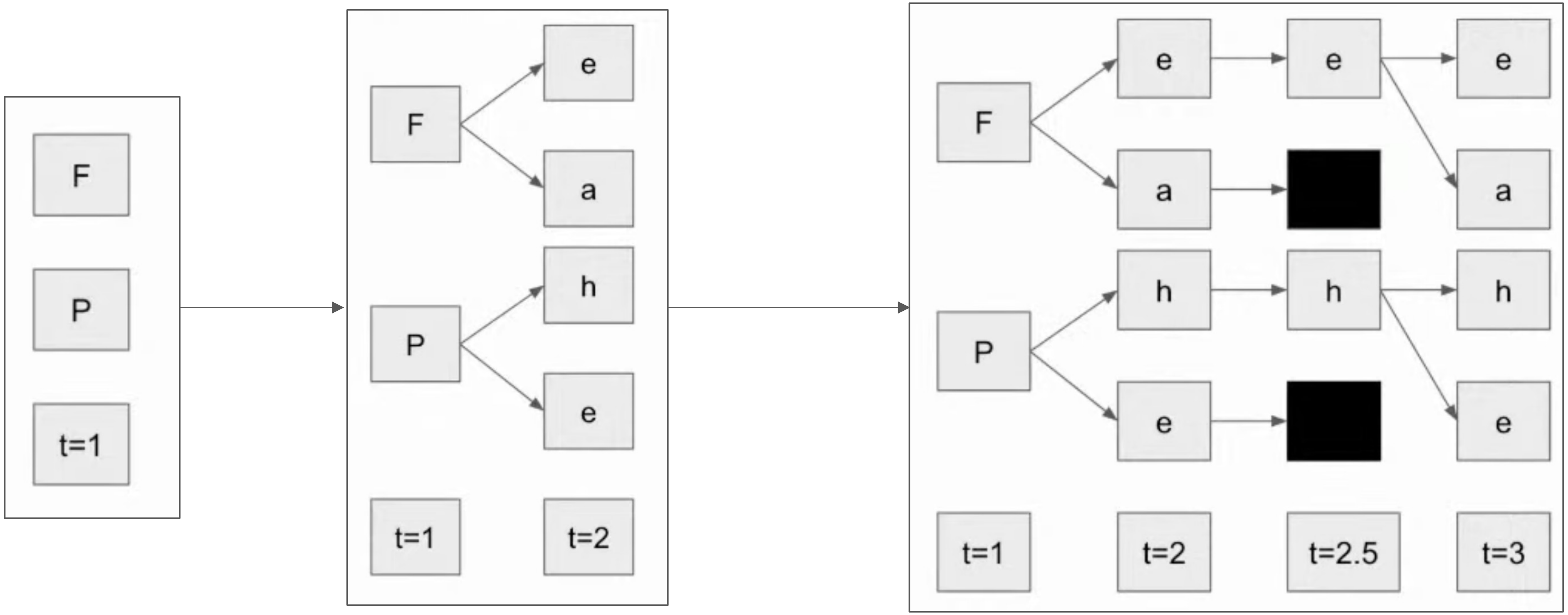
Beam Search: Fast Approximate Solution

- Step 1: Select the N best characters from the first time slice



Beam Search: Fast Approximate Solution

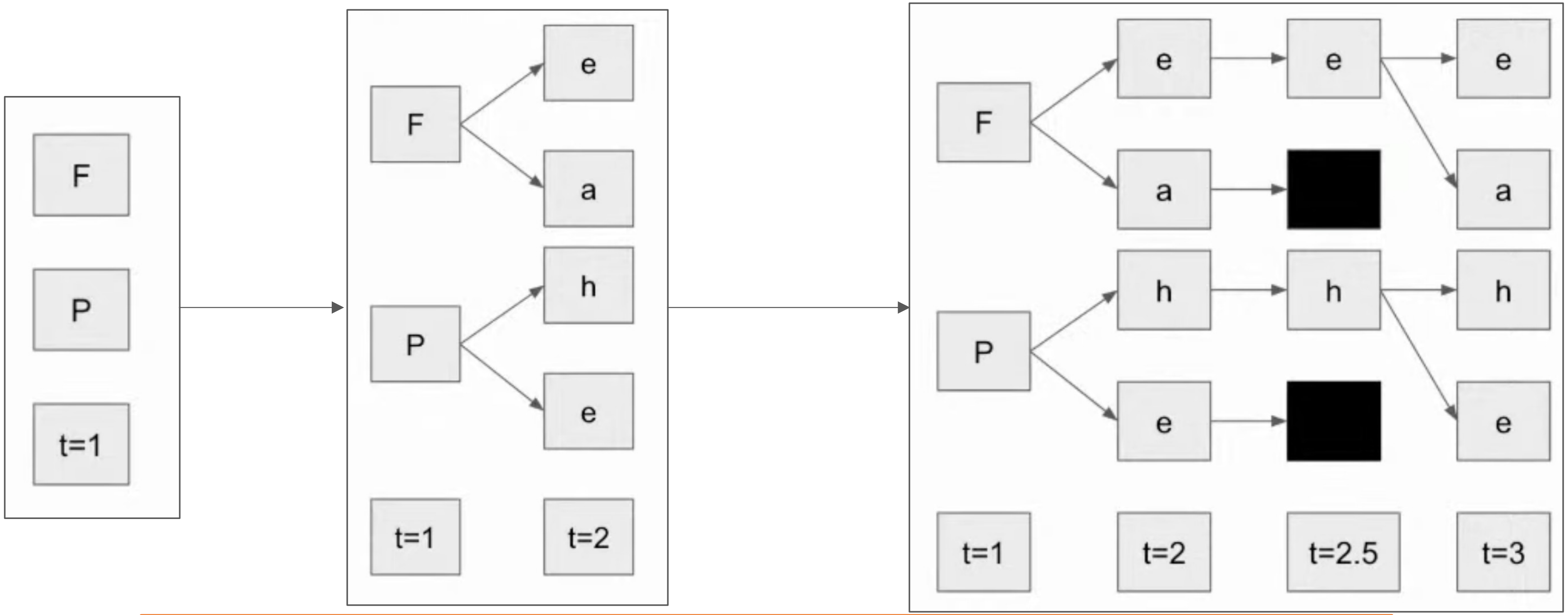
- Step 2.5: Rescore text outputs with the language model.



2.5: ***PRUNE*** on the basis of ***beam size***

Beam Search: Fast Approximate Solution

- Step 3: Continue by adding 3rd timestep, choosing the N best …



If **Beam-size = 1** => Greedy Decoding; if **Beam-size = inf** => Exact Soln

Tools of the Trade

Pyctcdecode

- Support for ***Hotwords*** 🔥
- Easy to *Setup & Experiment*
- Direct *Support* for Huggingface Speech Models, NeMo Speech Models, etc.

- Comparively *Slower* because of Python Implementation 👎
- *No* support for Neural Language Models 😿

Flashlight

- Supports decoding with ***Neural Language Model*** 🚀
- Extremely *Fast* Decoding ✨
- More *Configurations* Options

- Rapidly Evolving Library with *limited* backwards compatibility
- Official Documentation *Only* supports Fairseq

ASR+LM in action

- Training ***Kenlm***
  - Step 1: Creating ***ARPA*** file

```
print("Creating ARPA file ...\n")
subargs = [
        os.path.join(args.kenlm_bins, "lmplz"),
        "--order",
        str(args.arpa_order),
        "--temp_prefix",
        intermediate_dir,
        "--memory",
        args.max_arpa_memory,
        "--text",
        sents_path,
        "--arpa",
        lm_path,
        "--prune",
        *args.arpa_prune.split("|"),
    ]
if args.discount_fallback:
    subargs += ["--discount_fallback"]
subprocess.check_call(subargs)
```

ASR+LM in action

- Training ***Kenlm***
  - Step 2: Filtering from Lexicon

```
print("\nFiltering ARPA file using vocabulary of top-k words ...")
lexicon = open(lexicon_path).read()
subprocess.run(
    [
        os.path.join(args.kenlm_bins, "filter"),
        "single",
        "model:{}".format(lm_path),
        filtered_path,
    ],
    input=lexicon.encode("utf-8"),
    check=True,
)
```

ASR+LM in action

- Training ***Kenlm***
  - Step 3: Quantizing to 8 bit

```python
# Quantize and produce trie binary.
print("\nBuilding lm.binary ...")
if topk is not None:
    binary_path = os.path.join(output_dir, f"topk-{topk}_lm.binary")
else:
    binary_path = os.path.join(output_dir, f"lm.binary")
subprocess.check_call(
    [
        os.path.join(args.kenlm_bins, "build_binary"),
        "-a",
        str(args.binary_a_bits),
        "-q",
        str(args.binary_q_bits),
        "-v",
        args.binary_type,
        filtered_path,
        binary_path,
    ]
```

ASR+LM in action

- Decoding with ***pyctcdecode***
  - Build CTC Decoder

```
decoder = build_ctcdecoder(
    labels = vocab,
    kenlm_model_path = KENLM_MODEL_LOC,
    alpha=0.5,  # tuned on a val set
    beta=1.0,  # tuned on a val set
)
```

  - Decode logits from Acoustic Model

```
text = decoder.decode(logits)
```

ASR+LM in action

- Decoding with ***pyctcdecode***
  - Decoding with hotwords

```
hotwords = ["ivan"]
text = decoder.decode(
    logits,
    hotwords=hotwords,
    hotword_weight=10.0,
)
```

Some Numbers from IndicWav2Vec

- Lexicon Size: 200,000
- "n" in N-grams: 6 grams
- Avg. decrease in ***WER***: 28% reduction (21.6 to 15.6)

Model	**gu**	**ta**	**te**	**gu**	**hi**	**mr**	**or**	**ta**	**te**	**bn**	**ne**
IndicW2V	20.5	22.1	22.9	26.2	16.0	19.3	25.6	27.3	29.3	16.6	11.9
IndicW2V+LM	11.7	13.6	11.0	17.2	14.7	13.8	17.2	25.0	20.5	13.6	13.6

WER Numbers per language

Future Scope

- Decoding with ***Neural Language Models***
- Better integration of ***Language Models***
  - **Deep Fusion of LM**
    External LM is fused directly into the ASR model by combining their hidden states, resulting in a single model with tight integration.
  - **End-to-end Training with LM**
    Can we pass the gradients obtained after decoding with LM?

प्रश्नकाल?

Thank You

https://ai4bharat.iitm.ac.in/