English-to-English Grapheme-to-Phoneme Transliteration using RNN-Transducer

EE698R Course Project

Kumar Kanishk Singh (210544)
Sunny Raja Prasad (218171078)
Tanmey Agarwal (211098)

IIT Kanpur

April 20, 2025

Outline

Problem Statement I

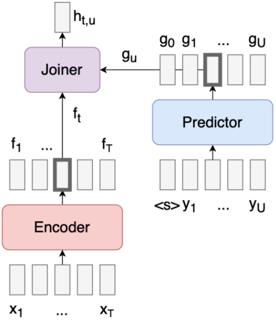
- **Task:** English-to-English Grapheme-to-Phoneme (G2P) Transliteration
  - Converting written text (graphemes) to pronunciation (phonemes)
  - Critical component in text-to-speech and speech recognition systems
  - Challenging due to English's irregular spelling-to-sound correspondence
- **Input:** Sequence of English characters (graphemes)
  - Words like "knight", "phone", "through"
  - Each character processed sequentially through the Encoder
  - Represented as numerical IDs via character vocabulary mapping
- **Output:** Sequence of phonemes (pronunciations)
  - ARPABET phonetic notation (e.g., "N AY T" for "knight")
  - Variable-length sequences predicted by the model
  - Non-aligned with input (single grapheme may map to multiple phonemes)
- **Dataset:** CMU Pronouncing Dictionary (cmudict)
  - Contains 134,000+ English words and their pronunciations

Problem Statement II

- Standard benchmark dataset for G2P tasks
- Available through NLTK library
- **Example:** "knight" $\rightarrow$ [N AY T]
  "phone" $\rightarrow$ [F OW N]
  "through" $\rightarrow$ [TH R UW]

RNN-Transducer Architecture

- **Encoder**: Processes character sequence
- **Predictor**: Models previous phoneme predictions
- **Joiner**: Combines outputs of Encoder and Predictor
- **Loss**: RNN-Transducer loss (via torchaudio)



Model Components

Encoder: Character sequence processor

- Input: Tokenized character sequence $(x_1, x_2, \ldots, x_T)$
- Embedding layer: dim $= 64$, maps character IDs to vectors
- LSTM: hidden_dim $= 128$, processes embeddings sequentially
- Handles variable-length inputs via packed sequences
- Output: f_t represents encoded character features

Predictor: Phoneme context modeler

- Input: Previous phoneme sequence with blank token
- Embedding layer: dim $= 64$, maps phoneme IDs to vectors
- LSTM: hidden_dim $= 128$, captures phoneme dependencies
- Output: g_u represents phoneme context features

Model Components (Continued)

Joiner: Feature combination network

- Takes encoder output f_t and predictor output g_u
- Projects both to joint_dim = 256 using linear layers:

$$h_{t,u} = \text{ReLU}(W_e f_t + W_p g_u)$$
$$\text{logits}_{t,u} = W_o h_{t,u}$$

- Creates a distribution over phoneme vocabulary
- Enables alignment between characters and phonemes

Beam Search Decoder: For efficient inference

- Maintains beam_width = 3 most probable hypotheses
- Iteratively evaluates encoder outputs at each timestep
- Handles the special blank symbol for RNN-T
- Algorithm:
  1. Initialize with empty hypothesis
  2. For each timestep $t = 1 \ldots T$:
     - Extend each hypothesis with top-k predictions
     - Include blank transitions (stay on same alignment)
     - Keep top beam_width hypotheses by score
  3. Return highest scoring complete hypothesis

Preprocessing: Data Source Vocabulary

Data Source and Extraction

- CMU Pronouncing Dictionary via NLTK
- 134,000+ word-pronunciation pairs
- Example: ("knight", ["N", "AY", "T"])
- Convert all words to lowercase

Vocabulary Construction

- Character vocabulary: $|V_{\text{char}}| = 29$ (a-z, apostrophe, etc.)
- Phoneme vocabulary: $|V_{\text{ph}}| = 40$ (ARPABET symbols)
- Special tokens:
  - `<pad>| at index 0`
  - `<blank>| for RNN-T`

Preprocessing: Data Processing

Numerical Conversion

- Convert text to indices:

$$\text{"cat"} \rightarrow [3, 1, 20] \qquad [\text{K, AE, T}] \rightarrow [21, 3, 38]$$

Batching and Padding

- Use custom `collate_fn` for batching
- Handle variable-length sequences with `pad_sequence`

Dataset Setup

- 80% training set (107,200 samples)
- 20% test set (26,800 samples)
- Use `DataLoader` with batch size 64

Training Configuration: Optimization

Optimization Parameters

- **Optimizer:** Adam
  - Default: $\beta_1 = 0.9$, $\beta_2 = 0.999$
  - No weight decay regularization
- **Learning Rate:** 10^{-3} (fixed)
- **Batch Size:** 64
- **Training Epochs:** 5
- **Device:** CUDA if available, CPU as fallback

Training Configuration: Loss and Loop

Loss Function

- **RNN-T Loss** from `torchaudio.functional`:

$$\mathcal{L} = -\log P(y|x)$$

- Aligns predicted and target sequences via all valid paths
- Key steps:
  1. Apply `log_softmax` to logits
  2. Run forward-backward algorithm
  3. Handle blank tokens appropriately
  4. Compute batch mean loss

Training Loop

- CER evaluation on training test sets
- Batch-level progress tracking
- Beam search decoding (width $= 3$)

Evaluation Metric: Character Error Rate (CER)

Definition:

$$\text{CER} = \frac{\text{EditDistance}(\mathit{ref}, \mathit{hyp})}{\text{Length of reference}}$$

What is Edit Distance?

- Levenshtein distance between reference and hypothesis
- Measures:
  - **Substitution:** incorrect phoneme predicted
  - **Insertion:** extra phoneme added
  - **Deletion:** missing phoneme in prediction
- Normalized to [0–100%] range
- Lower CER $\Rightarrow$ Better pronunciation prediction

CER Example

Reference: [K AE T]
Hypothesis: [K AH T]

Edit Distance: 1 (substitution)

$$\text{CER} = \frac{1}{3} = 33.3\%$$

Evaluation Process

- **Evaluation protocol**:
  1. Set model to evaluation mode (disable dropout)
  2. Process each test sample with beam search (width=3)
  3. Convert predicted phoneme IDs to phoneme symbols
  4. Compute CER between reference and hypothesis
  5. Average CER across entire dataset
- **Reported metrics**:
  - Training CER: 13.6% (model fit quality)
  - Test CER: 14.4% (generalization ability)
  - Transformer model comparison: 3.4% CER (both train/test)

Error Analysis

Common Error Patterns

- **Vowel substitutions** (most frequent)
  - Example: "about" → [AH B AW T] instead of [AH B AW T]
  - Cause: Similar phonetic properties between vowels
- **Silent letter handling**
  - Example: "knight" → [N AY T K] instead of [N AY T]
  - Cause: Model struggles with irregular spelling patterns
- **Complex phoneme mappings**
  - Example: "through" → [TH R U] instead of [TH R UW]
  - Cause: Multi-character graphemes to single phoneme mappings

Error Distribution

- Substitutions: 68% of errors
- Deletions: 19% of errors
- Insertions: 13% of errors

Results Comparison

Model	**Train CER**	**Test CER**
RNN-Transducer	13.6%	14.4%
Transformer	3.4%	3.4%

- Transformer models achieve significantly lower error rates (3.4% vs 14.4%)
- Better performance attributed to:
  - Deeper network architecture
  - Self-attention mechanism capturing global dependencies
  - Parallel processing of sequence elements
- RNN-T could potentially achieve similar results with:
  - Deeper LSTM layers
  - Larger hidden dimensions
  - But would require substantially more training time and compute resources

Conclusion

- **Effective G2P Modeling:** The RNN-Transducer architecture demonstrates strong capability in learning grapheme-to-phoneme (G2P) mappings, effectively capturing dependencies between characters and phonemes.
- **Joint Encoder-Predictor Modeling:** By jointly modeling the character encoder and phoneme predictor, the RNN-T allows for dynamic alignment and better phoneme predictions, leading to improved performance over traditional sequence-to-sequence approaches.
- **Simplified Training with TorchAudio:** Leveraging the built-in RNN-T loss function from TorchAudio simplifies implementation and stabilizes training, reducing the need for complex loss function engineering.

Future Work

- **Larger Beam Widths:** Investigate the impact of increasing beam width during decoding to explore more hypotheses and potentially improve final phoneme sequence quality.
- **Advanced LSTM Architectures:** Experiment with deeper LSTM stacks or bidirectional LSTMs in the encoder and predictor to better capture long-range and contextual dependencies in character and phoneme sequences.
- **Multilingual G2P Transliteration:** Extend the current RNN-T framework to handle multilingual datasets, enabling phoneme prediction across different languages and scripts for broader applicability in real-world transliteration tasks.

References

M. Bisani and H. Ney.
Joint-sequence models for grapheme-to-phoneme conversion.
Speech Communication, 50(5):434–451, 2008.
https://dl.acm.org/doi/book/10.5555/2049202

A. Graves.
Sequence transduction with recurrent neural networks.
In *ICML Representation Learning Workshop*, 2012.
https://www.cs.toronto.edu/~graves/icml_2012.pdf

R. Fukuda, H. Yamamoto, and K. Takeda.
Global RNN Transducer Models For Multi-dialect Speech Recognition.
In *Interspeech*, 2022.
https://www.isca-archive.org/interspeech_2022/fukuda22_interspeech.pdf

Carnegie Mellon University.
CMU Pronouncing Dictionary.
https://en.wikipedia.org/wiki/CMU_Pronouncing_Dictionary

Natural Language Toolkit.
nltk.corpus.reader.cmudict.
https://www.nltk.org/_modules/nltk/corpus/reader/cmudict.html

J. Jiang, Y. Zhang, S. Wang, et al.
Pretraining Enhanced RNN Transducer.
Artificial Intelligence Research, 2024.
https://www.sciopen.com/article/10.26599/AIR.2024.9150039

H. Kang and K. Choi.
An Ensemble of Grapheme and Phoneme for Machine Transliteration.
Proc. of the International Conference on Natural Language Processing, 2005.
https://aclanthology.org/I05-1040.pdf

End Credits

Thank You!