Pronounce this!

Tarun Sunkaraneni

THE UNIVERSITY OF UTAH

Introduction

- CMU pronunciation dictionary maps over 134,000 words to their pronunciations.
- There are 39 phenomes, not accounting for lexical stress variations.
- 84 total symbols with vowel stress variations.
- Stress markers: 0(None), 1(primary), 2(secondary)

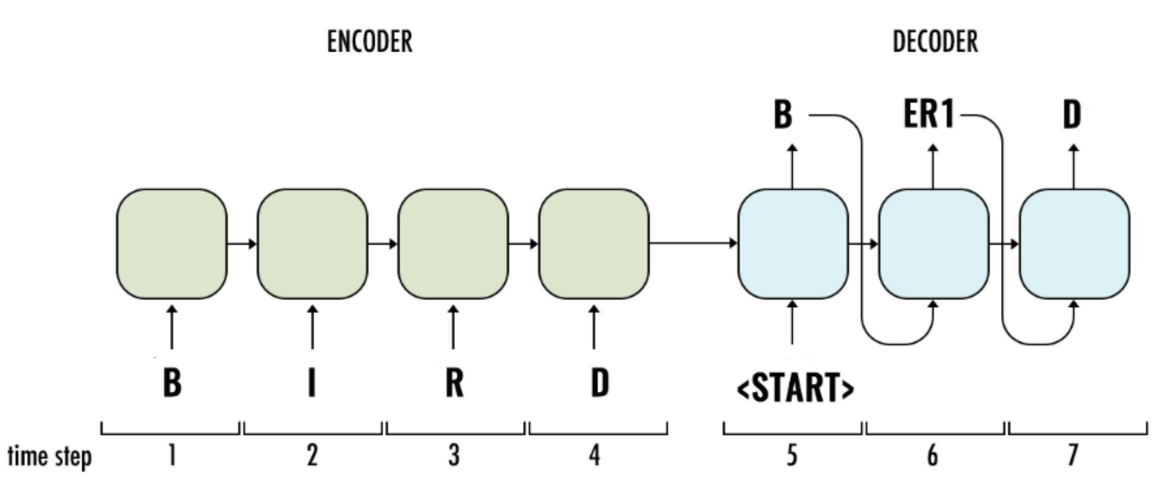
```
LANSING --> L AE1 N S IH0 NG
SHIMKUS --> SH IH1 M K AH0 S
COMOROS --> K AO0 M AO1 R OW2 S
NONINTEREST --> N AA0 N IH1 N T R AH0 S T
SLOAN --> S L OW1 N
SAW --> S AO1
GUIDROZ --> G W IY1 D R OW0 Z
CLINGMAN --> K L IH1 NG M AH0 N
TABERNACLE --> T AE1 B ER0 N AE2 K AH0 L
CLEARWATER --> K L IH1 R W AO2 T ER0

After cleaning, the dictionary contains 124814 words
and 133569 pronunciations
(8755 are alternate pronunciations).
```

1) Baseline

- One hot encoded input characters and output phenomes.
- No form of regularization
- 1) Syllable accuracy: done by counting number of phenomes with stress markers.
- 2) Perfect accuracy: count examples with correct output phenomes and stress markers.
- 3) Bleau Score: Used in NMT, same principle applies in phenome translation

Architecture




```
(Input) Word Matrix Shape:  (5333, 18, 30)
(Output) Pronunciation Matrix Shape:  (5333, 19, 87)
```

- `Encoder input shape: (# Examples, Max word length, # of characters)`
- `Decoder output shape: (# Examples, Max phenomes length, # of phenomes)`

```
"A" vector:
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0.]
-----
"AH0" vector:
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Encoder Procedure

- Given character at timestamp T, use hidden state to capture sequential information about the sequence.
- This information will yield C_t and H_t, both of which will be fed to the decoder the same way in both test and train phases

Decoder Procedure

- Training: Given the *correct* phoneme, predict the next phenome using hidden state.
- Testing: Using the phoneme derived at timestamp T-1, predict T.
- Requires a universal <start> input to decoder at first timestamp in both phases
- Feed X_{t-1} until <stop> fed in as input

Predictions

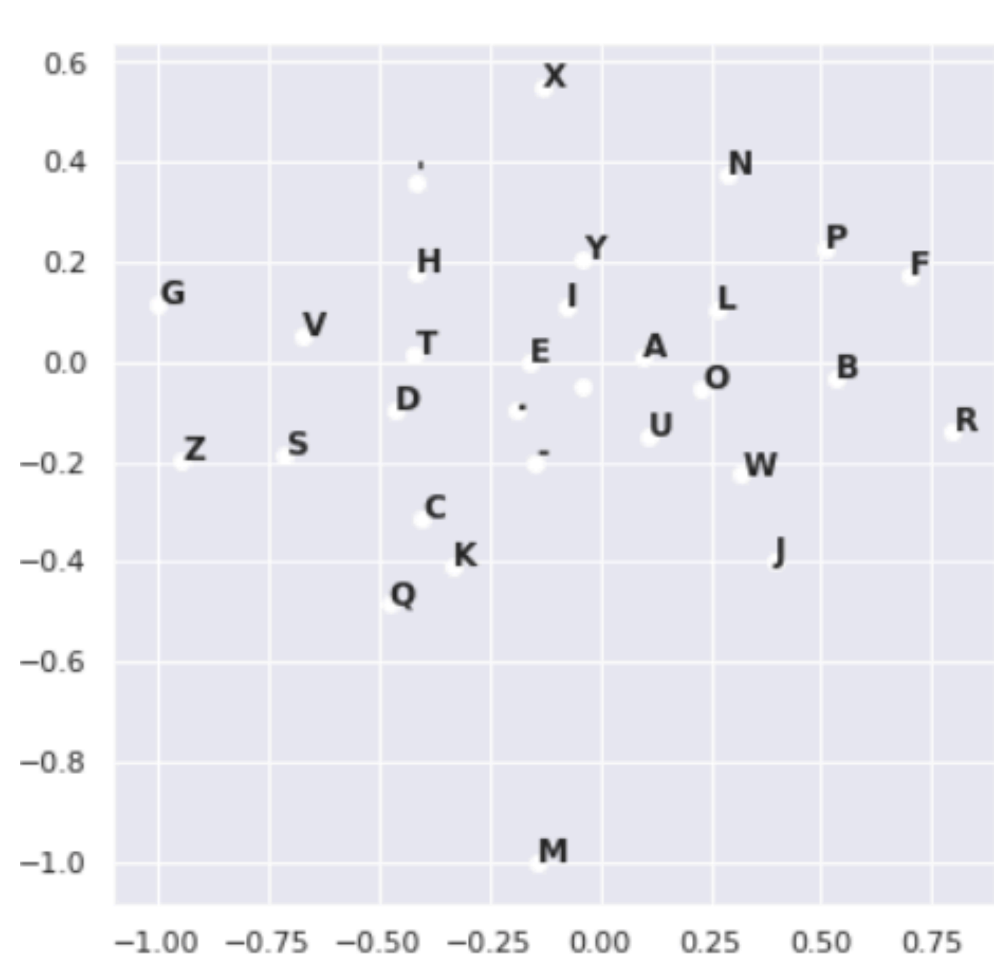
```
✗ RECOGNISE --> R IY0 K AO1 G N AY2 Z
✓ SEASE --> S IY1 Z
✗ BUESCHER --> B UW1 SH ER0
✓ SCHMOKE --> SH M OW1 K
✓ ALTUS --> AE1 L T AH0 S
✗ ZAVODNY --> Z AA0 V AA1 D N IY0
✗ MALEFACTORS --> M AE2 L AH0 F AE1 K T ER0 Z
✗ MANSEAU --> M AE1 N S UW2
✗ JAMIESON --> JH EY1 M IY0 AH0 S
✓ KIDDE --> K IH1 D
✗ KANAN --> K AE1 N AH0 N
✓ ENLISTING --> EH0 N L IH1 S T IH0 NG
✓ COOVER --> K UW1 V ER0
✓ GUADAGNO --> G W AA2 D AA1 G AH0 N
✗ SCISSOR --> S IH1 S ER0
✓ EYESHADE --> AY1 SH EY2 D
✓ BLOWN --> B L OW1 N
✗ EKATERINA --> EH2 K AH0 T IH1 R IY0 AH0 N
✓ CARICO --> K AA0 R IY1 K OW0
✗ MCADOW --> M AH0 K D AW1
✓ BABBLING --> B AE1 B AH0 L IH0 NG
✓ SLINGERLAND --> S L IH1 NG G ER0 L AH0 N D
✓ DIPRIMA --> D IH0 P R IY1 M AH0
✗ KENNEBREW --> K EH1 N AH0 B R UW2
✗ AUDITORY --> AO1 D AH0 T AO2 R IY0
✗ CARPORTS --> K AA1 R P ER0 T S
✓ COLM --> K OW1 L M
✓ NASHBURG --> N AE1 SH B ER0 G
✓ KINGTON --> K IH1 NG T AH0 N
✓ SINGLES --> S IH1 NG G AH0 L Z
```

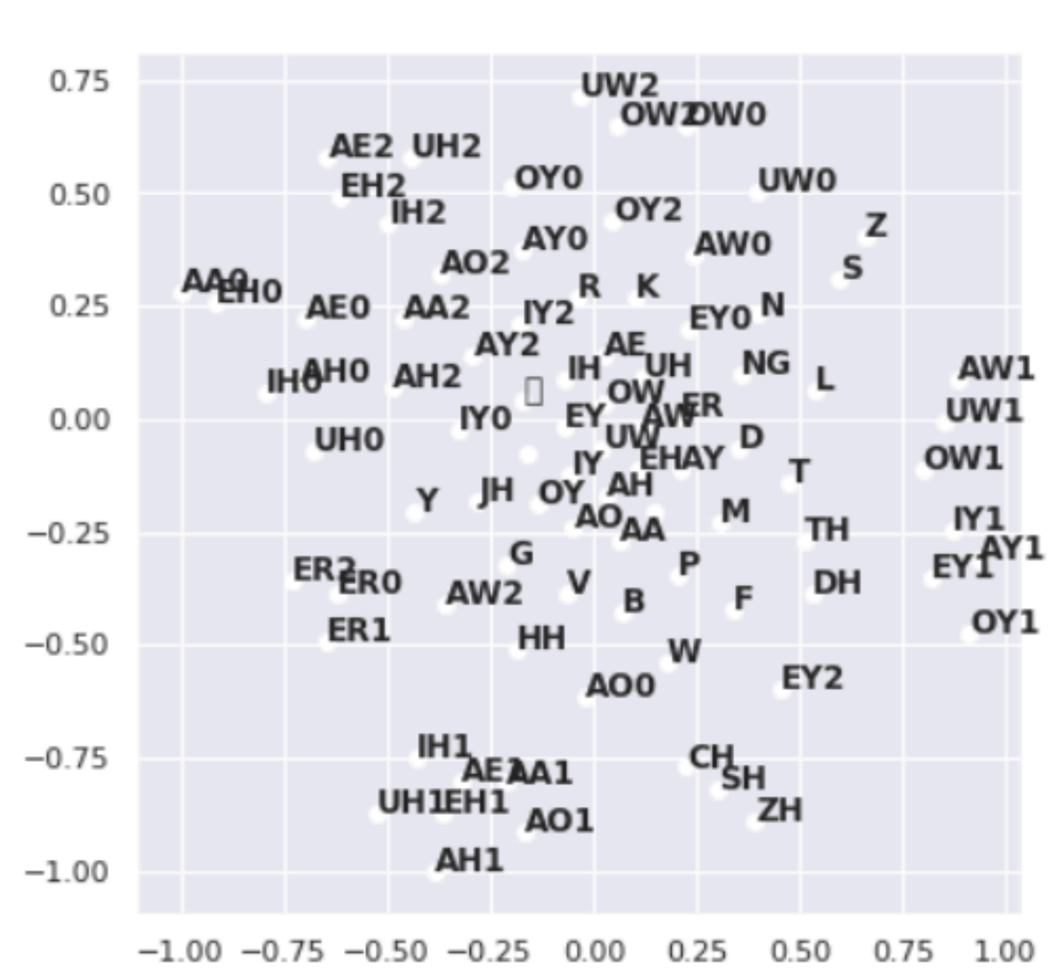
2) Character Embeddings

- Learnable character and phoneme embeddings provides model more flexibility to learn, and makes training less memory intensive
- `Encoder input shape: (# Examples, Max word length)`
- `Decoder input shape: (# Examples, Max phenomes length)`
- `Char embedding shape: (# Examples, Embedding dim)`
- `Phenome embedding shape: (# Examples, Embedding dim)`
- Fix overfitting by adding dropout

```
(Input) Word Matrix Shape:  (5333, 18)
(Output) Pronunciation Matrix Shape:  (5333, 19)
```

TSNE-visualizations of characters and phonemes





Evaluation

Model	Baseline	Embedding	Embedding + Dropout
Syllable Acc	91.0	93	96.2
Perfect Acc	52.3	55.3	64.0
Bleu Score	0.668	0.692	0.744

- Embedding layer gives model flexibility to perform better, but can result in overfitting.
- After adding Dropout, the perfect accuracy score increases
- Embedding lowered memory constraints significantly, although adding many more trainable parameters

```
predict_word = 'NICK'
predict_char_input = np.array([np.zeros(MAX_CHAR_SEQ_LEN)])
for idx, letter in enumerate(predict_word):
    predict_char_input[0][idx] = char_to_id[letter]
print(predict_word, ':', predict_char_input)

NICK : [[17. 12.  6. 14.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.]]
```

```
predict_emb(predict_char_input, emb_testing_encoder_model,
            emb_testing_decoder_model)

'N IH1 K'
```

Unaddressed problems

- Our information flow only goes in one direction. Word pronunciation is very much dependent on the next character as it is on the previous character.
- Use Bi-Directional RNN connections
- Only connection between Encoder and Decoder is through the state "handoff." These states alone might not be sufficient for long words.
- Use Attention mechanism.
- Predication is not robust and versatile, chooses only most likely sequence.
- Use BeamSearch

```
✗ DITOMMASO --> D IH0 T OW0 M AA1 S OW0
✗ PROGRESSIVELY --> P R AH0 G R EH1 S IH0 V L IY0
✗ AIS --> EY1 Z
✗ GOULASH --> G UW1 L AE2 SH
✗ ARABELLA --> AA2 R AH0 B EH1 L AH0
✗ COLLOPY --> K AA1 L AH0 P IY0
✗ ACRID --> AE1 K R AH0 D
✗ COHENOUR --> K AA1 HH AH0 N UH0 R
✗ ION --> AY1 AA0 N
✗ FRIEDBERG --> F R IY1 D B ER0 G
✗ FEDERMAN --> F EH1 D ER0 M AH0 N
✗ SOCIETAL --> S OW0 S AY1 AH0 T AH0 L
✗ ADVISING --> AH0 D V AY1 Z IH0 NG
✗ ARDMORE --> AA1 R D M AO0 R
✗ COLLOPY --> K AA1 L AH0 P IY0
✗ LOELLA --> L OW0 EH1 L AH0
✗ CORRELATES --> K AO1 R AH0 L EY2 T S
✗ LOELLA --> L OW0 EH1 L AH0
✗ KANAN --> K AE1 N AH0 N
✗ DEGAS --> D IY1 G AH0 S
```

Improvements

- Very little hyperparameter training was done due to compute requirements.
- Separate Stress marker model to predict intensity of vowels

Acknowledgements

- Predicting Pronunciations with Syllabification and Stress with Recurrent Neural Networks - Daan Esch
- Text-to-Phoneme mapping using neural Networks – Eniko Bilicu
- English Phoneme predictions using RNN – Ryan Epp
- Joint-Sequence Models for Grapheme-to-Phoneme Conversion- Max Bisani