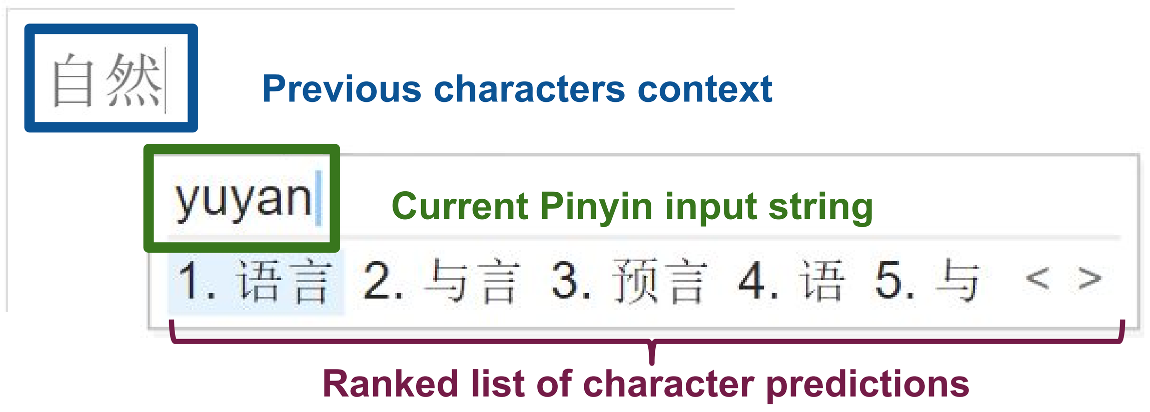
Neural Input Method Engine

CSE 481 NLP - Team Very Natural : Jingchen Hu, Jianyang Zhang, Qingda Wen

Background & Objectives

- An **Input method engine** facilitates the input of non-english characters into digital devices.
  - A Chinese Pinyin IME "translates" from Pinyin tokens (pronunciation symbols) to Chinese characters.
- Traditional Input Method Engine:
  - n-gram models, no long-term memory.
- Our Goal: more contextual information, more accurate predictions, less keystrokes, faster typing.



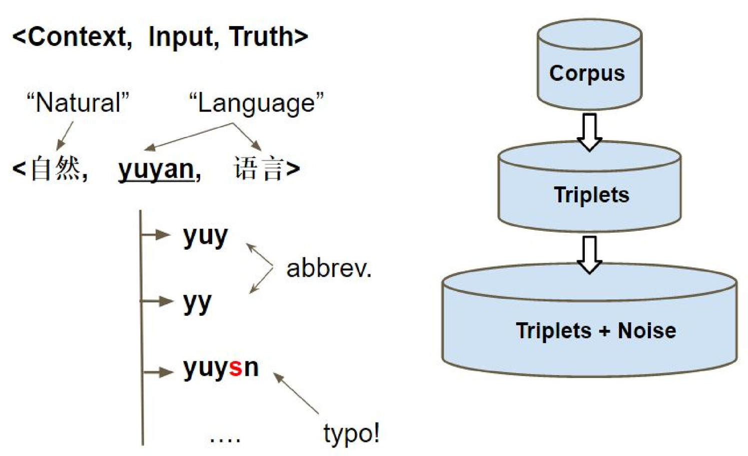
Methodology

- Push the fuzzy match logic to the dataset.
- Use **encoder-decoder** with beam search to generate the ranked list of predictions per input length.
- Merge the results of different lengths as the final suggestions list.
- Query the same model when the user makes new selections.
- Use Google's Seq2Seq library on top of Tensorflow.

Datasets

- Extract tuples from corpora and **add noise** to emulate user input. Use sliding window on word boundary.

The Lancaster Corpus of Mandarin Chinese (LCMC)	2000k
NUS SMS dataset	500k
Weibo dataset (Twitter equivalent)	5000k
Wikipedia abstracts	35M

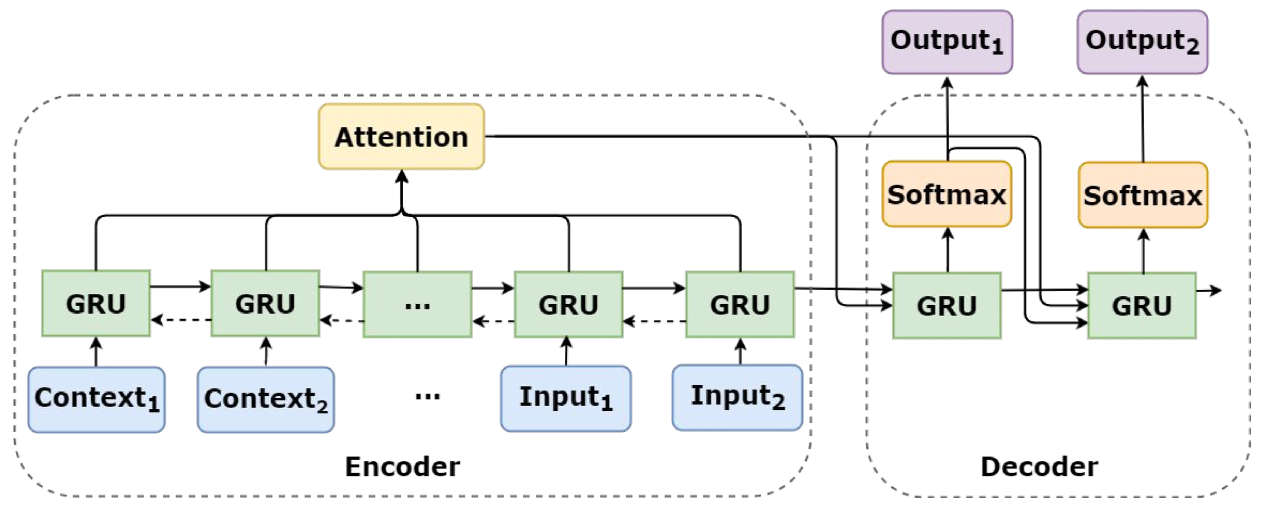


Chrome Extension

- Allows users to input on any webpage.
- Detects keystrokes, buffers the current pinyin input, and queries the backend as the user types.

Seq2Seq with Attention

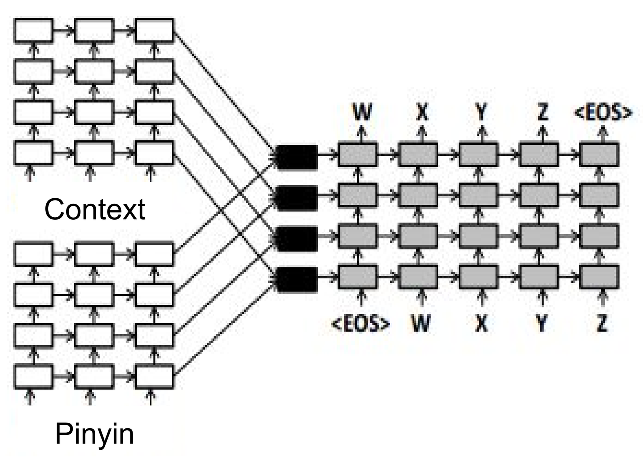
- Input: **concatenation** of Chinese characters (previous context) and pinyin tokens (current user input)
- Output: predicted Chinese characters based on previous context and pinyin tokens



Model Variants

Multi-encoder

Uses two separate encoders and map the output of encoders to decoder using a bridging layer.



$$h = \tanh(W[h_c; h_p] + b)$$

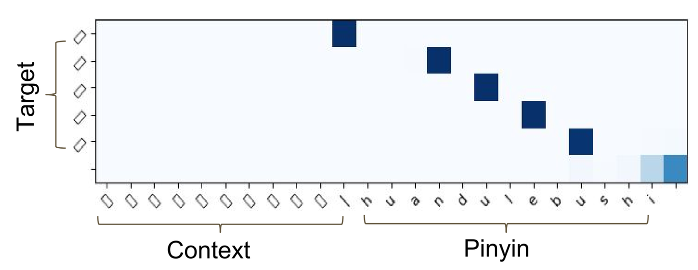
Separate Attention

Separate softmax on attention scores, parameterized / direct concatenation:

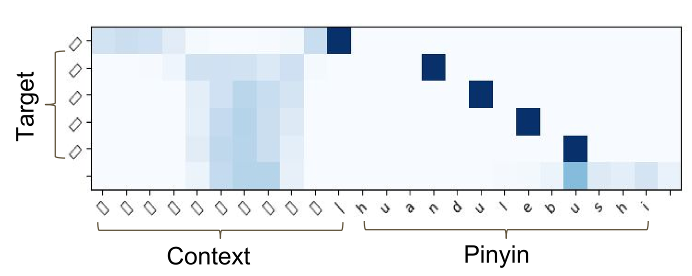
$[k * softmax(att_{context}); (1-k) * softmax(att_{pinyin})]$

$[softmax(att_{context}); softmax(att_{pinyin})]$

- Joint attention



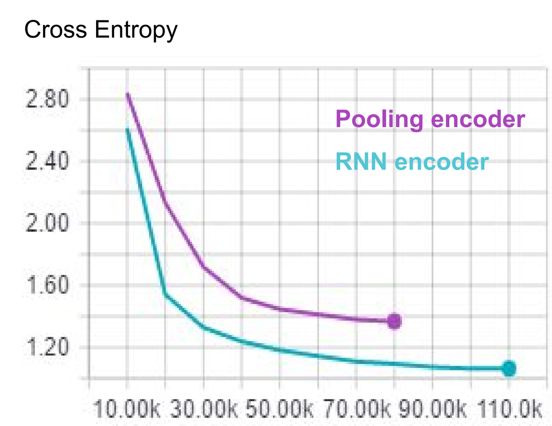
- Dual attention (direct concat):



Pooling encoder

Averages the embeddings of k consecutive words.

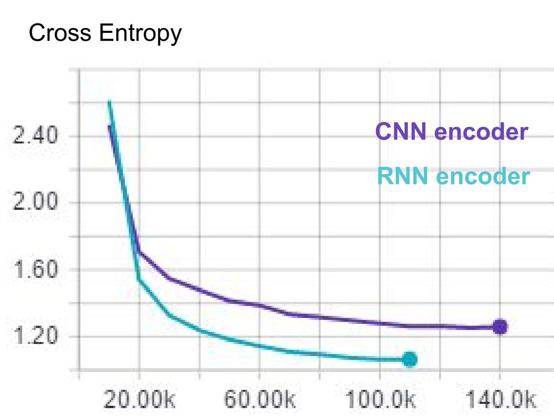
Source embedding = word embedding + position embedding



Deep CNN encoder

Two stacked convolutional networks for encoding:

- Encoder output used for attention
- Conditional input used by decoder

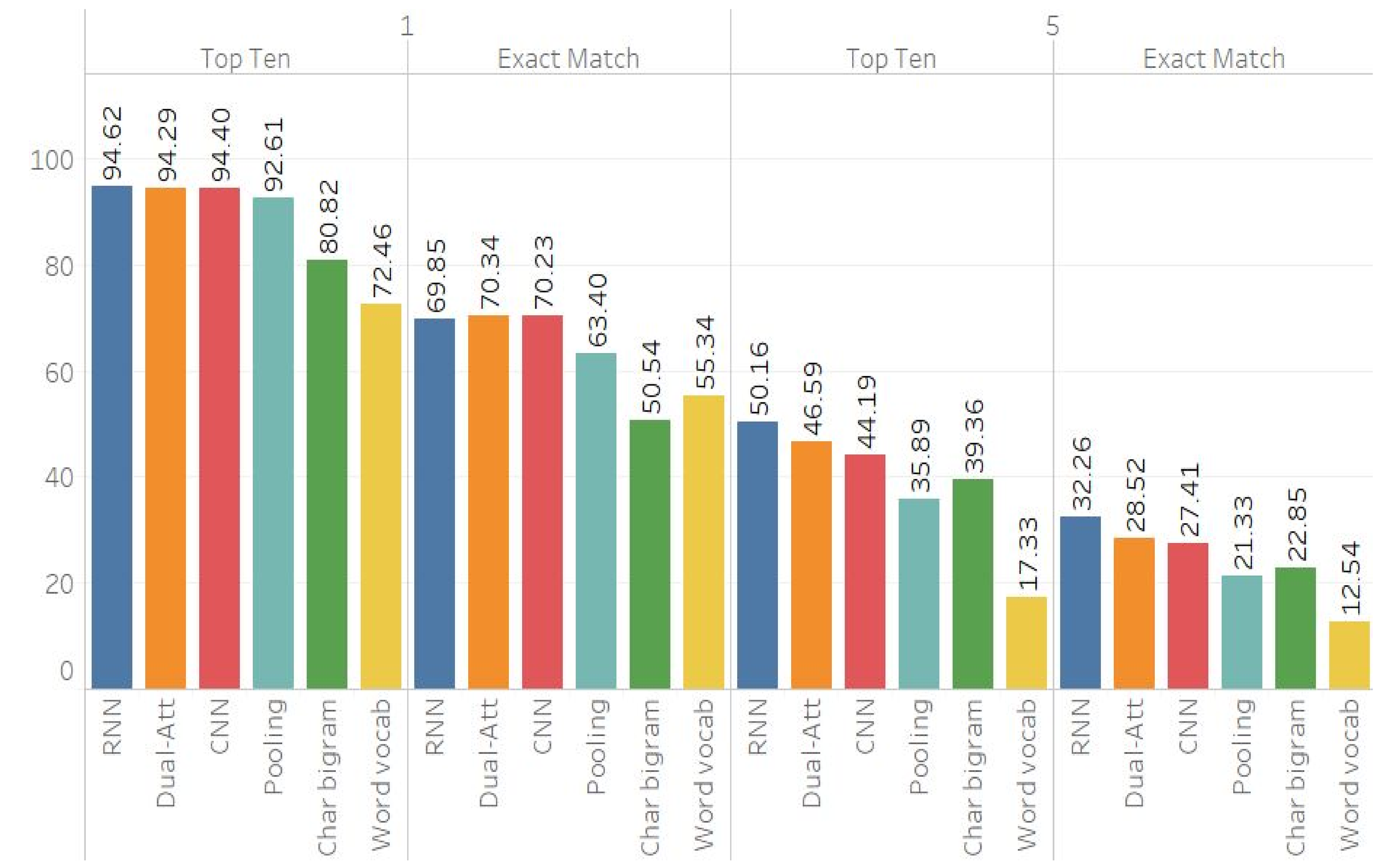


Beam Search Result Scoring

Rank the predictions using a scoring function - the weighted sum of bigram score of each token and its log-probability.

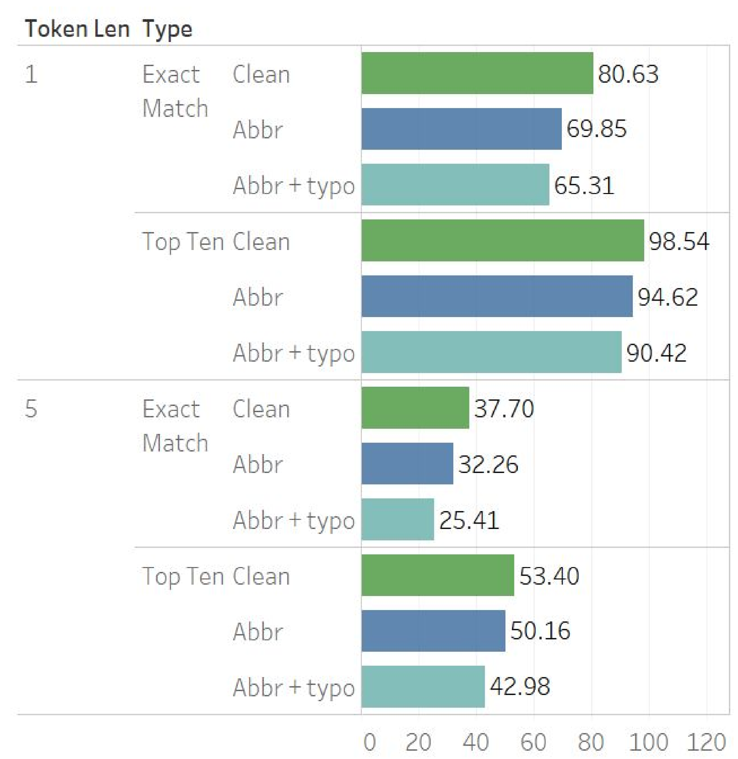
Experiments

Prediction accuracy for different input lengths

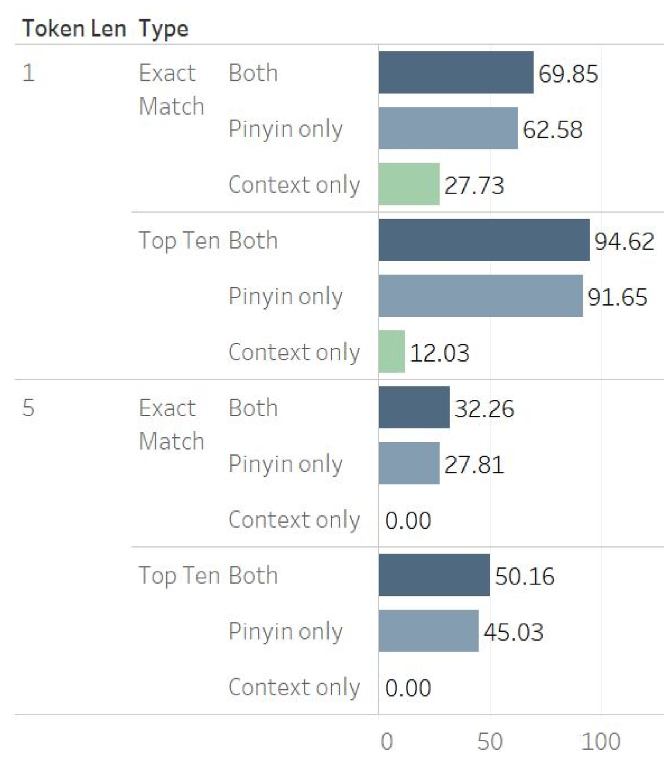


*Exact match: correct prediction ranked first in the returned list.
*Top Ten: correct prediction appeared in top ten of the returned list.

Dataset noise level comparison



Effectiveness of context



Conclusion

- All of our encoder-decoder models achieve similar accuracies.
  - Significantly better than the baseline bigram model.
  - The default attentional RNN Seq2Seq model performs the best.
- The models learn Pinyin-to-character mapping pretty well.
- Context does help, but not as much as we expected.

Work Cited:

- "Multi-Source Neural Translation", Zoph, Barret, and Kevin Knight. (2016).
- "A Convolutional Encoder Model for Neural Machine Translation", Gehring, Jonas, et al.(2016)
- "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation", Wu, Yonghui, et al.(2016).
- D. Britz, A. Goldie, T. Luong, and Q. Le. Massive Exploration of Neural Machine Translation Architectures. ArXiv e-prints, March 2017.