English Instructions to Unix Commands Translation

Hao Wu | CS221 | Mentor: Sydney Li

Motivation & Problem Definition

From Amazon's Alexa to MicroSoft's Cortana, these AI assistants rely on the ability to translate natural language instructions to machine commands, e.g launch itunes, search for the weather. The performance of these commercially available AI assistants are great, but they are not perfect.

Although this project does not aim to surpass Alexa or Cortana, it strives to come up with an AI program that can assist users of Unix systems by *translating English instructions to Unix commands;* specifically, instructions to navigate directories, finding files, opening files and listing directory content. It will especially benefit users who are unfamiliar with or unaware of the command line utilities and would like to exploit such utilities. It will be a good aid to developers working in Unix environment, or simply serve as a shortcut to regular users.

Challenges

- Learning/extracting the same meaning behind different phrasings of an instruction
- Distinguish actions, arguments, and rhetoric with in English instructions, ignore useless/decorative words/expressions, e.g. "please", "can you"; while learning important things such as "find", "go".
- Ensure correct ordering during translation

Data

Training set: 5035 pairs of instruction and command

Validation set: 298 pairs.

The types of instructions are: finding a file/folder, listing what's in a directory or catting a file, go to certain directory, go certain levels up directory tree, and go back to previous folder.

Note on Validation Set:

The validation set contains only instructions that have structures/phrasing previously **unseen** in the training set to measure how good the model is at learning the underlying structures of this translation problem.

Generation Method:

1. Collecting synonyms of different action words, e.g. find, and different structures of instructions, e.g. "Can you do something", "do something for me".
2. Plug in different action words into different phrasings in a meaningful way
3. For instructions involving arguments, e.g. find somefile, try different file names
4. For instructions of iterative nature, e.g. go up 3 levels, use different numbers

Implementation

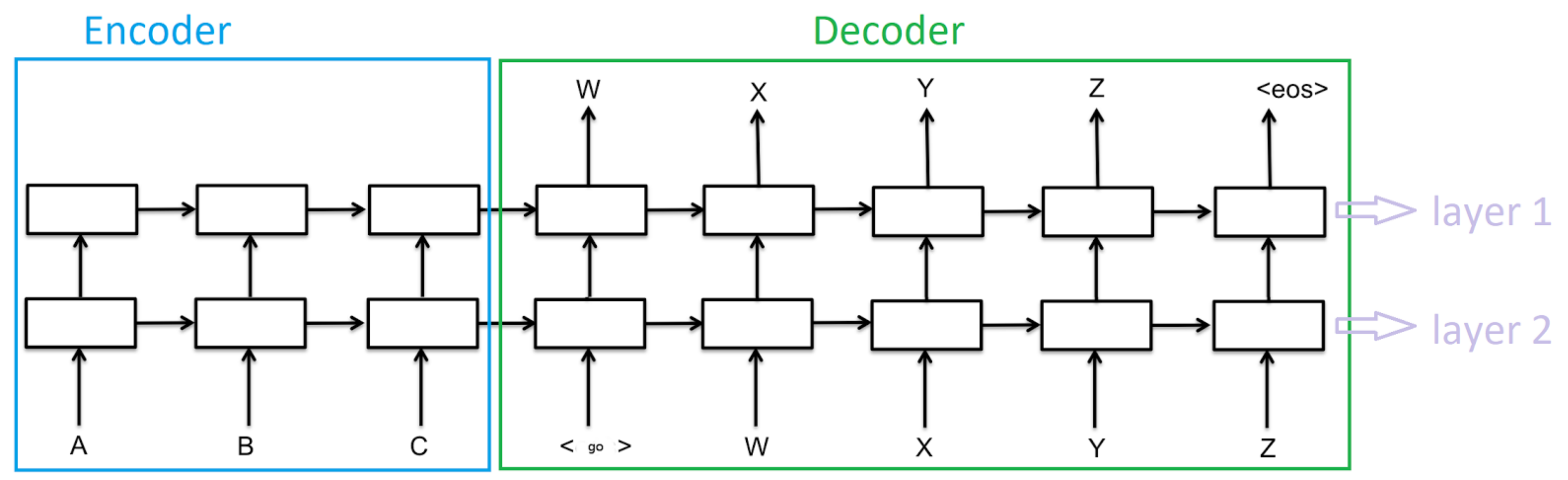


Figure. The Model - 2-layered RNN with LSTM Cells

Each small rectangle represent a hidden state, implemented with a LSTM cell.

Each English Instruction (input sequence) and each Unix command (output/target sequence) need to be encoded into a sequence of integers.

A special integer ARG_ID is reserved to denote the "argument" in the instructions (see "Argument Detection" in the "Key improvements" section)

Since usually we need one RNN model for every pair of (input length, output length), both input sequences and output sequences are padded, with special character 'PAD', to length 50, so only 1 model is needed to handle the (50, 50) case.

The RNN is bi-directional, though not shown on above graph

<go> represents a special end-of-line character, signaling the end of this input sequence, and the decoder part of the RNN will begin

For each of encoder and decoder, the weights on the edge between any time t-1 and t is the same, by the definition of RNN

The encoder portion and decoder portion do not share weights

Hyperparameters

Learning Rate	Learning Rate Decay Factor	# Layers	Batch Size	Max Gradient Norm	# Neurons per Layer
• 0.5	• 0.99	• 2	• 16	• 5.0	• 128

Key Additions to Vanilla RNN

Argument Detection

- During pre-processing of data, use a linear classifier with feature extraction to detect whether a segment in instructions/commands is an argument or not.
- Examples of argument: "somefile.txt" in "can you please fine somefile.txt for me"; "somefolder" in "can you open somefolder".

Semantic Equivalence

- Try to uncover meaning of numbers in English instructions, e.g. "go up 3 levels" which suggests a loop structure
- During pre-processing of data, map to equivalent flattened expressions, e.g. "go up levels levels levels", for better Sequence to Sequence translation performance

Early Stopping

- Prevent overfitting to training data, better validation set performance
- Stop when the current evaluation/dev loss is larger than all past 10 values
- Usually stops at around 1500 iterations

Results

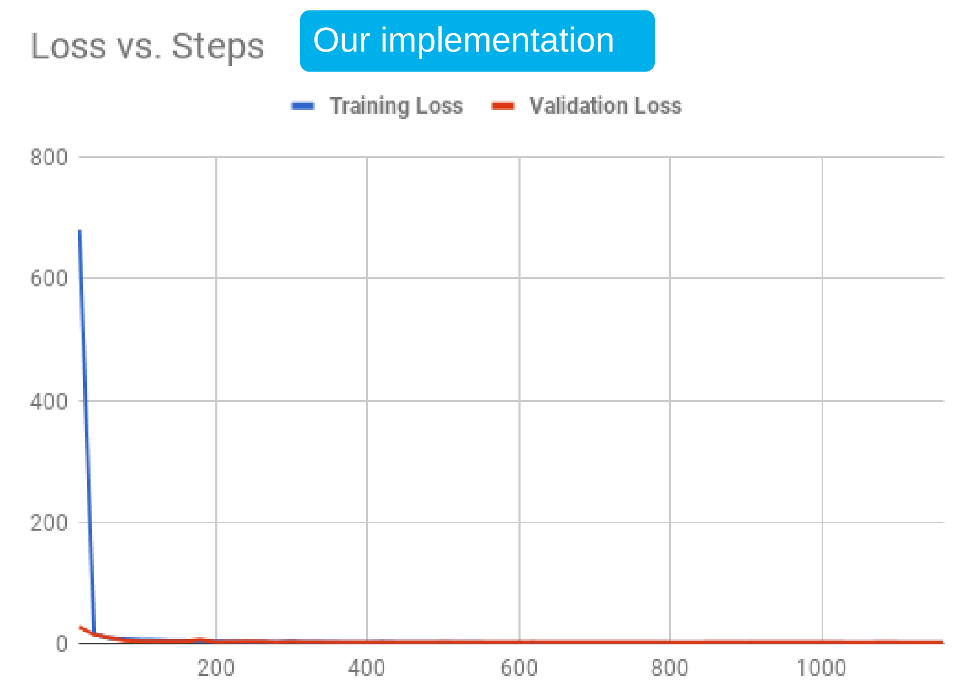
	PM	CO	CM
Training	99.6%	100%	97.7%
Validation	99.6%	100%	97%

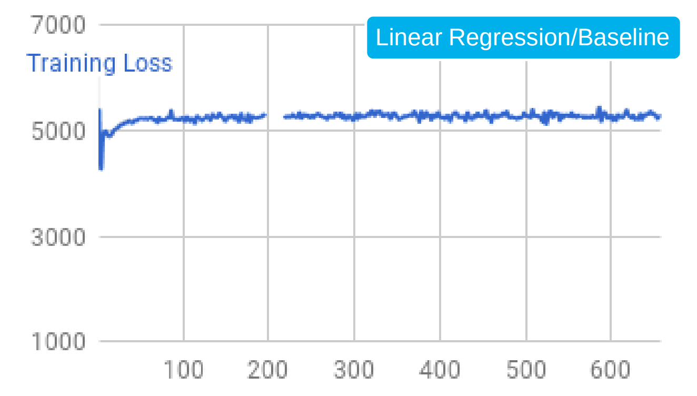
PM: On average, the percentage of segments of each command that are correct. E.g: when the prediction is (find / -name somefile.txt) and oracle/target is (find . –name somefile.txt), PM is 3/4 = 75%

CO: The percentage of output sequences that has the correct ordering. For example, if the command sequence of the oracle is (A B C D), our output (A B F D) will be considered as having the right ordering.

CM: Percentage of completely correct output/command

The sampled softmax loss should serve as a good general indicator of how the model is performing as it approximates how many parts/tokens of a translated command matches with the target command





Analysis and Future Directions

The 2 layered RNN is great at learning different phrasing and their underlying meanings. It especially produces good results with the "Key Additions". After through examination, the model is able to detect arguments with high accuracy, and "flattening" iterative instructions also greatly improved the translation accuracy of them.

Given the promising results of this project, it is possible that some of the techniques used can be extrapolated to other types of machine translations. There are still areas for improvement, such as detecting arguments considering the context rather than just analyzing the features of the argument alone, and I would love to continue exploring after this course.

Works Cited

TensorFlow, *"Sequence-to-Sequence Models"*
https://www.tensorflow.org/versions/r1.0/tutorials/seq2seq

Leonardo Araujosantos, *"Machine Translation Using Recurrent Neural Networks"*
https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/recurrent_neural_networks/machine-translation-using-rnn.html