Final Project Status Update: Improving Pseudo-code from Source Code Generation using Neural Machine Translation

Vincent Nguyen and **Austin Shin**
CS 287: Statistical Natural Language Processing
Harvard University
Cambridge, MA 02138
{vnguyen01, ashin01}@college.harvard.edu

1 Overview

We have set up a repository for our code and have finished preprocessing the data. We are in the process of running machine translation and optimizing our results as well as finding new models and metrics to include. We have preliminary translations that are very comparable to the ones translated by software engineers in the paper by Oda et al. [1]. All code written for this project exists on github[1].

2 Data Preprocess

The dataset of code to pseudo-code annotations is publicly available. [2] This dataset was curated by programmers who commented existing source code in Django with English pseudo-code. Additionally, the source code in Python with Japanese pseudo-code is also available publicly. The Python to English pseudo-code annotations do not belong to a dataset. Rather, these annotations were learned by the Django-to-English data set.

An example of a Django source code is

```
inner = NonCapture(result[start:])
```

and the corresponding English pseudo-code translation is

```
inner is an instance of NonCapture,
created with elements of result from
            start index to the end.
```

A summary of the public datasets can be found in the table below: Up until this point, we have only been working with the English pseudo-code data. We anticipate working with the Japense pseudo-code data in the upcoming week.

Data	Examples
Django-to-English	18805
Python-to-Japanese	722
Python-to-English	NA

We note that the line translations appear to be mostly independent of one another. That is, there does not exist substantial dependency through lines of code. The only time that context dependency might arise is through conjunctions, such as

```
    and evaluated boolean expression
s_opt.nargs doesnt equal to integer 0
```

As such, we train-test-split by line. We have 15044 lines to train and 3761 lines to validate.

3 Models

Attentional Neural Network We run the Sequence-To-Sequence Attention Neural Network on the data to learn translation. Initially, we were obtaining nonsensical results from the seq2seq, but a recent code push fix gave more interpretable translations. We trained with 1 layer with an embedding size of 50 and 50 hidden units. Additionally, we are training on a CPU. After 3 epochs, we obtain a training perplexity of 23.2650 and a validation perplexity of 21.088. This perplexity is consistent with what Professor Rush obtained when doing a preliminary run of the seq2seq model.

We adjusted the parameters of the model to obtain a lower training perplexity score of 15.9065 and a validation perplexity score of 16.28. The new model parameters included extening the layer size to 2 but keeping roughly the same embedding size and number of hidden units. We trained for 10 epochs. This reduction in perplexity is evaluated on translation samples shown in the following section.

4 Preliminary Results

After comparing the results of the seq2seq model on this task with the feature-based model from Oda et al.[1], we believe our vanilla neural model does a very good job. We have not yet computed the BLEU metrics

[1] https://github.com/vnguyen01/src2anno
[2] http://ahclab.naist.jp/pseudogen/

to compare. Included below are sample translations from the Django source code to English pseudo-code for the initial model of 3 epochs and a validation perplexity of 21.088:

```
  if not e . connection_reset :
```

if f is not None,

```
self . attachments . append (
filename )
```
append a result to the result.

```
@ register . filter ( is_safe = True
)
```
decorator function register.filter with an argument is_safe set to boolean True.

```
return bytes ( ( i , ) )
```
call the function force_str with an argument value, return the result.

The translations are mostly accurate and appear to describe what the code does to some extent. We also show results for a better trained model that achieved a validation perplexity of 16.28:

```
  if not e . connection_reset :
```

if self.verbosity is false,

```
self . attachments . append (
filename )
```
get the value from the base class from the base dictionary.

```
@ register . filter ( is_safe = True
)
```
call the method parser.add_argument with an argument is_safe set to boolean True.

```
return bytes ( ( i , ) )
```
return a tuple with an entry: return the result.

5 Future Steps

In the upcoming days, we are planning on doing the following:

- Running code on the cluster to have more layers and intensive computations
- Concatenating sentences to evaluate predictions on longer source code
- Sentence summarising on source code and pseudo-code
- Coming up with a way to do capture context of surrounding source code lines for translation
- Summarization of code functions

References

[1] Yusuke Oda and Hiroyuki Fudaba and Graham Neubig and Hideaki Hata and Sakriani Sakti and Tomoki Toda and Satoshi Nakamura. *Learning to Generate Pseudo-code from Source Code using Statistical Machine Translation*. 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2015.

[2] Dzmitry Bahdanau and KyungHyun Cho and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. ICLR 2015.

[3] Illya Sutskever and Oriol Vinyals and Quoc Le. *Sequence to Sequence Learning with Neural Networks*. NIPS 2014.