Documentation Is All You Need

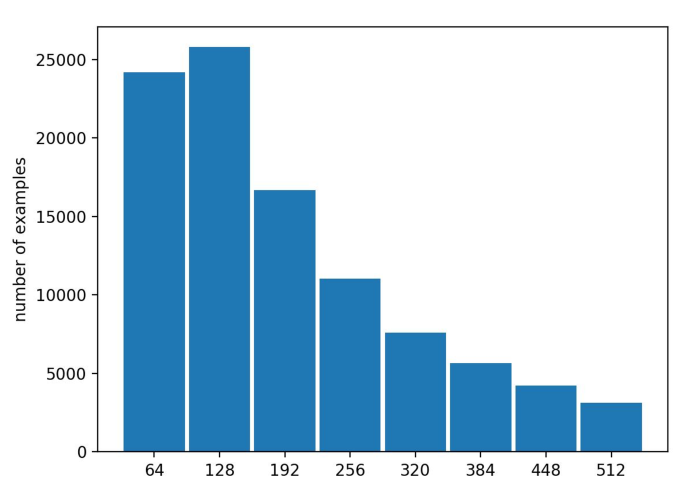
Felipe Meneses
fbomfim@stanford.edu

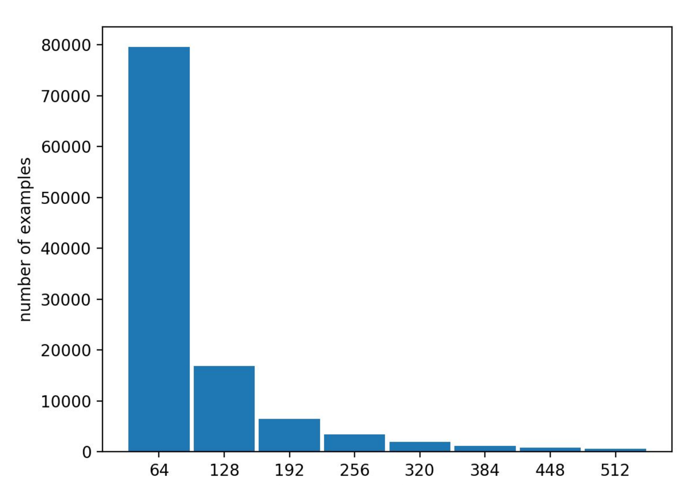
Introduction

- **Motivation -** Documentation, which provides a high-level description of the task performed by the source code, is a must-have for team-based software development groups. Conversely, even with various tools that have been developed to aid the programmer during its creation, crafting correct and consistent documentation for source codes remains a labor-intensive task. Thus, significant gaps in documentation are ubiquitous and considerably contribute to the inefficiency of program comprehension by software engineers.
- **Goal** – generate high-level natural language descriptions from raw low-leve Python source code.
- **What we built -** our final product is a 178M parameter Transformer with pre-trained embeddings that takes as input a function's source code and predicts its description.

Data | Features | Preprocessing

- **Dataset:** For our dataset we use 108,726 Python function-description pairs taken from Wan et al.'s paper [9].





- **Preprocessing:** We split the dataset into 80:10:10 for train, dev, and test sets, respectively; assign the examples into buckets based on the source code length; split data points into batches, tokenize them and apply padding based on buckets.

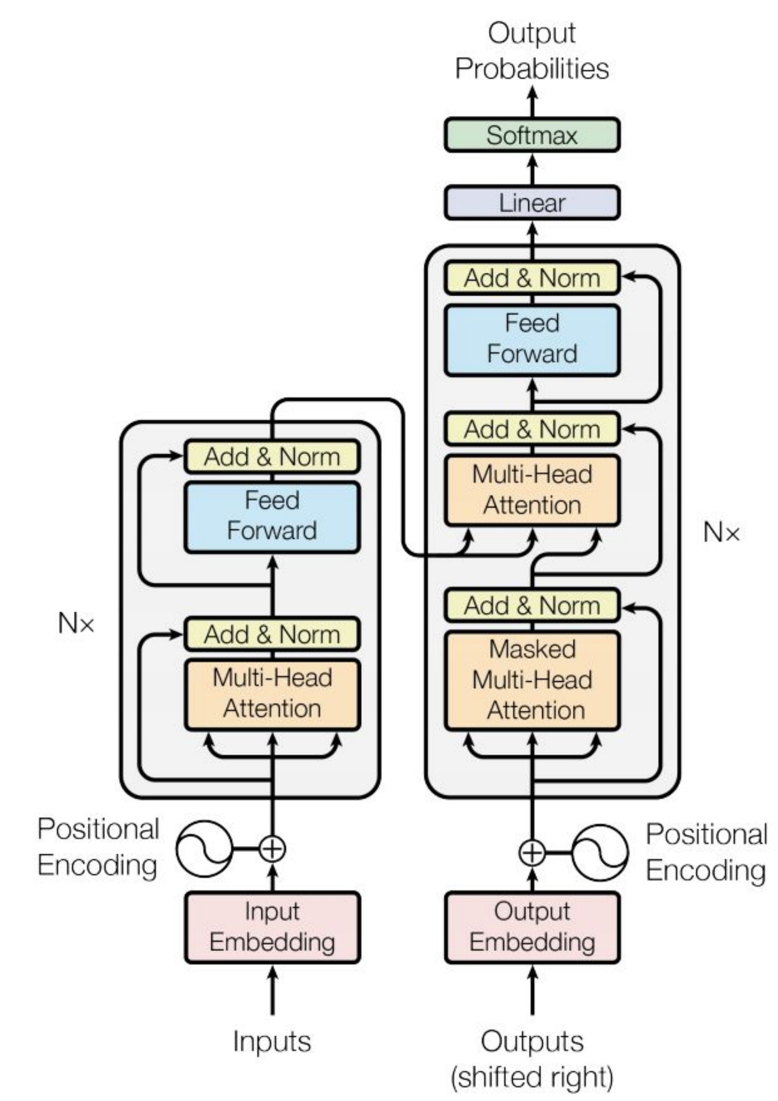
Model

- We use a **Transformer**[8] with pre-trained GloVe[5], and GPT-2[6] embeddings for the inputs and outputs, respectively.
- **Multi-Head attention** – masking, combined with output embeddings offset by one position, to ensure predictions would depend only on the known outputs (the previous output tokens).

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- **Embeddings** – For our enconder, we pre-trained a GloVe language model on Python code for functions and lines. For our decoder, we used pre-trained embeddings from Hugging Face's implementation of GPT-2 small [10].

Transformer Architecture



Epochs	30
d_{model}	768

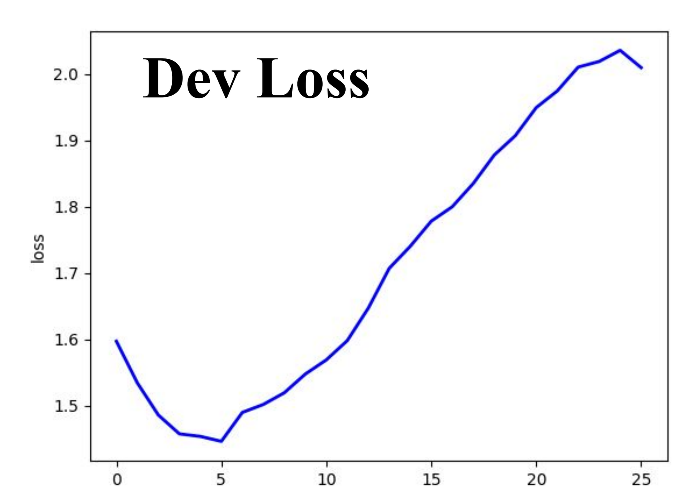
N (encoder-decoder layers)	6
Multi-Head Attention Layers	8

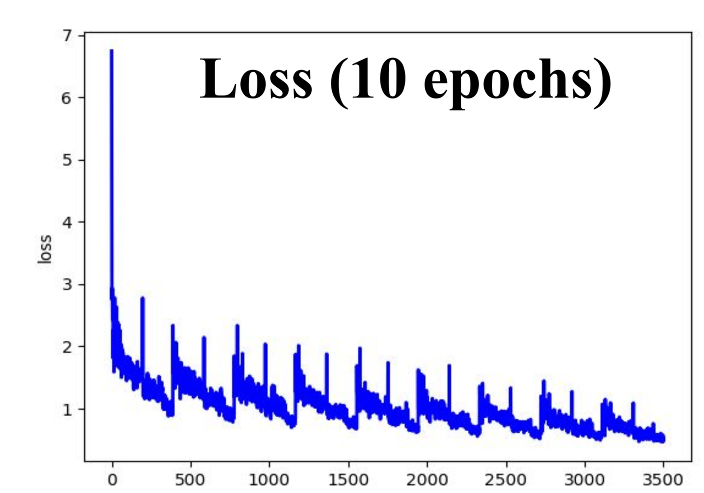
# of parameters	178 M
Optimizer	Adam

Results

- **Loss function -** we use negative log likelihood (NLL) with a log-softmax activation applied on their linear output matrix.

$$loss(x, t) = -log\left(\frac{e^{x_t}}{\sum_j e^{x_j}}\right)$$





Discussion

- There is a considerable variance problem in the model. The fact that the dev set loss distances itself so early from the training set loss indicates that the model might be overtly big for our dataset.
- The striking disconnect between the sets seems to indicate that there was a representation issue in our algorithm. We suspect the GloVe embeddings were insufficient in capturing the structural essence of the code. This dissipated most of the meaning through that layer and hurt the generalizability of the model.
- A more interesting approach would be to use abstract synthax trees, such as in code2vec[2], with some form of tree encoder which would capture more of the structural nature of source code.

Future Work

- Given more time and computational resources, we would have trained the model for longer to find the sweet spot for train and validation loss.
- This model could be improved by using better code embeddings such as the one in code2vec [2], and larger embeddings for the decoder, such as GPT-2 Large.
- We would also try to produce a utility metric for a given description and try to combine Wan et al.'s deep reinforcement learning approach [9], with our model.

References - **[1]** Uri Alon, Omer Levy, and Eran Yahav. "code2seq: Generating Sequences from Structured Representations of Code" (2018). **[2]** Uri Alon et al. "Code2Vec: Learning Distributed Representations of Code" (Jan. 2019). **[3]** Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". **[4]** Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library" (2019) **[5]** Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation" (2014). **[6]** Alec Radford et al. "Language Models are Unsupervised Multitask Learners" (2019). **[7]** Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. "Unsupervised Pretraining for Sequence to Sequence Learning" (2016). **[8]** Ashish Vaswani et al. "Attention is All you Need" (2017). **[9]** Yao Wan et al. "Improving Automatic Source Code Summarization via Deep Reinforcement Learning" (2018). **[10]** Thomas Wolf et al. "Hugging Face's Transformers: State-of-the-art Natural Language Processing" (2019). **[11]** X. Xia et al. "Measuring Program Comprehension: A Large-Scale Field Study with Professionals" (Oct. 2018).