Code Documentation to C#

By: Muhammad Ali Qadri
Cristian Diaz

Problem Statement

- Models currently exist for code generation
- Models currently exist for code summarization
- We want to create a pipeline, that uses a combination of the above, to perform a task which was not done before
- Documentation String to C#

Data Description

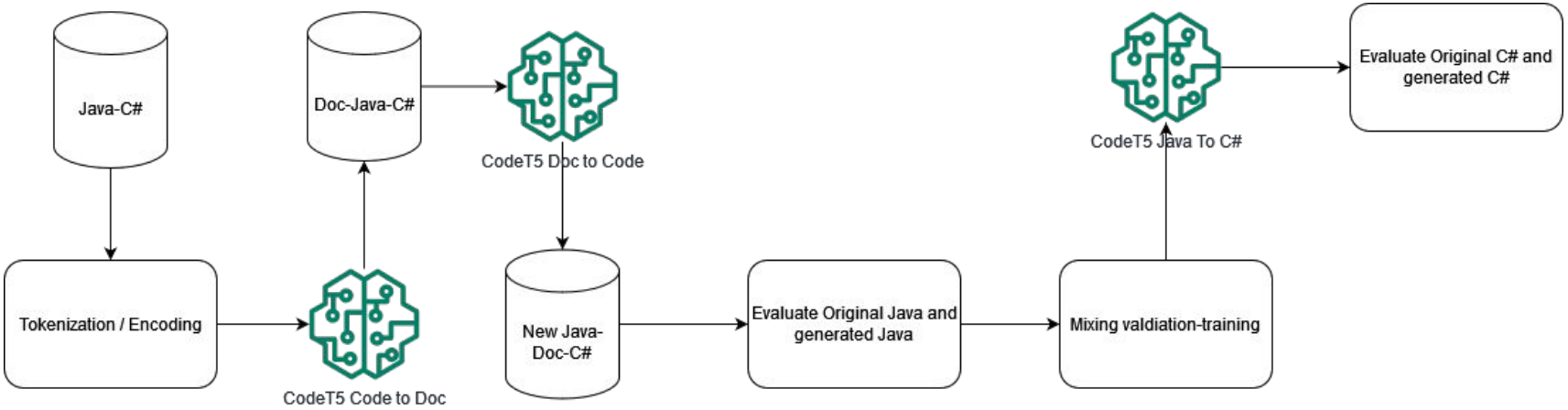
Existing Datasets Used

- Used CodeTrans, part of CodeXGLUEs paper
- Off handedly made use of CONCODE (used to fine-tune our model)

Our Dataset

- Used pre-trained CodeT5 to generate Documentation Strings for Java
- Grouped Documentation Strings, with original Java, and C#
- Tokenize and encode code and document instances
- Training set contained ~10000 samples
- Validation set contained ~500 samples
- Testing set contained ~1000 samples

Model Building Pipeline



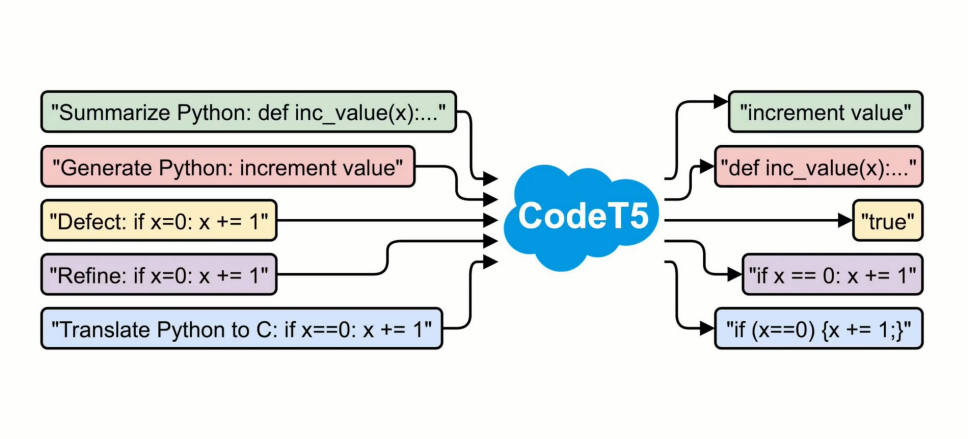
- Trained on A100 Nvidia GPU Instance
- Perform loss and validation
- Beam search for generating predictions

Data Processing and Exploration

- Downloaded pretrained CodeT5, (tuned on CONCODE)
  - Used it to generate to Documentation Strings
- Created Pytorch Data-loaders
  - First we grouped the Documentation Strings with the original Java, and C# code
  - We then extracted input encodings, and attention masks for training tasks

Documentation String to Java

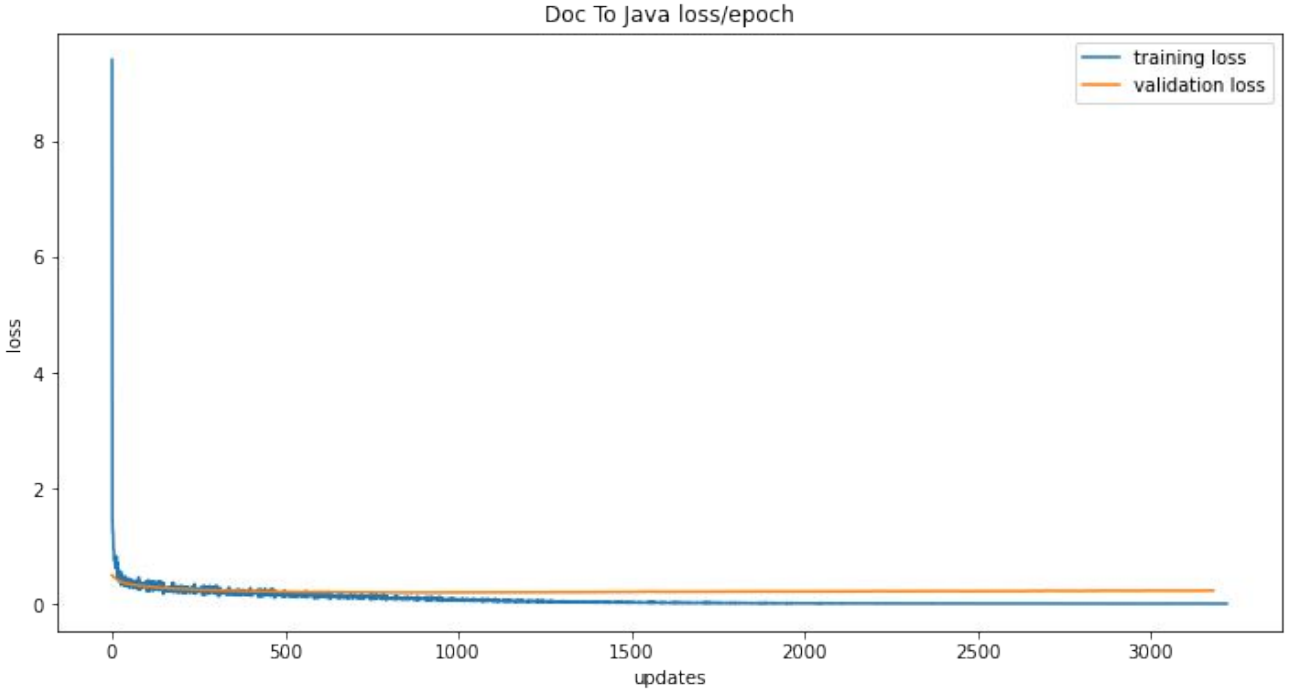
- Used CodeT5s pre-trained and fine-tuned model for this task
- T5 uses transfer learning for tasks
- transformer models to perform NL-PL downstream tasks
- Train for 3000 updates
- Fits our situation



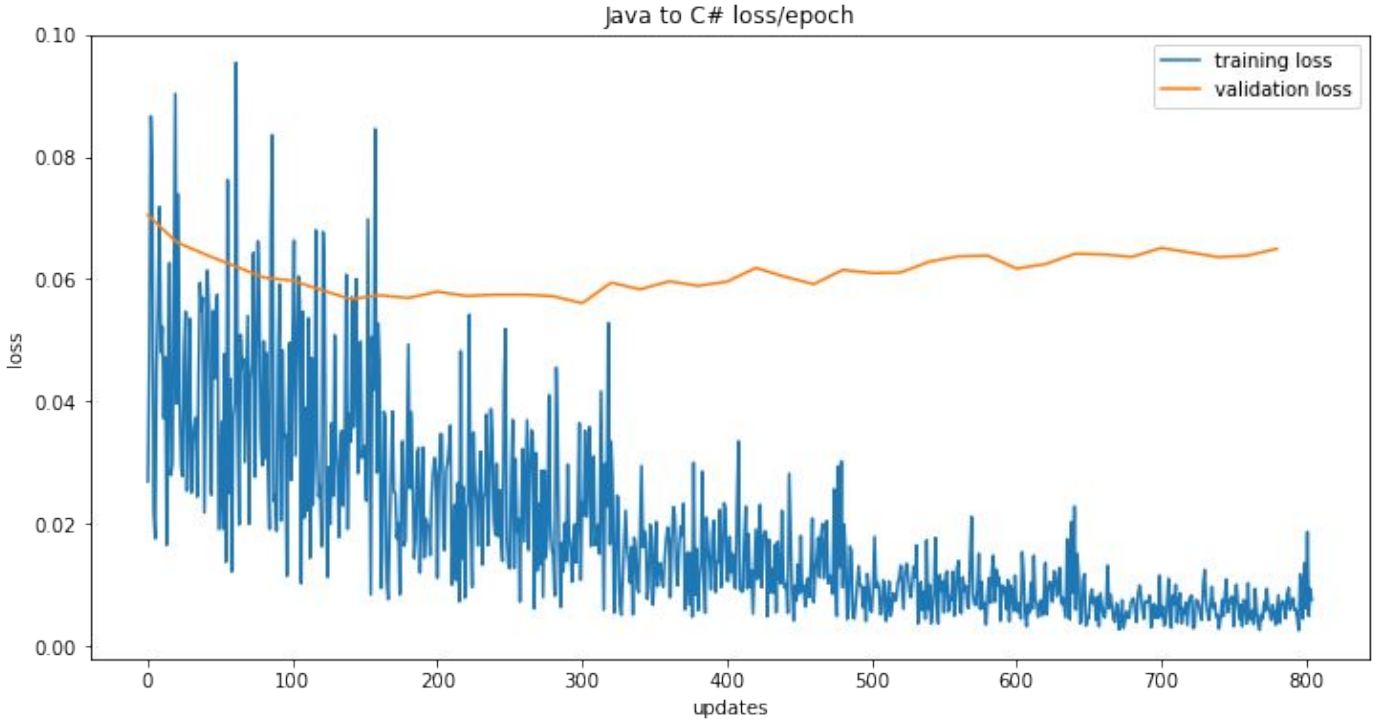
Java to C#

- Use CodeT5s pre-trained model for task
- Trained for 800 updates
- Plotted loss and validation
- Evaluate against original Java and C#

Evaluation (Document to Java)



Evaluation (Java to C#)



Evaluation (BLEU and CodeBLEU)

	Training	Validation	Testing
Java BLEU	86.2	31.45	21.9
Java CodeBLEU	89.32	42.29	**42.06**
C# BLEU	2.98	1.75	2.85
C# CodeBLEU	11.41	9.73	**11.21**

Lessons Learned

- Learned different state-of-the-art models
- Many models to choose form
- Need to pick model carefully that fits needs
- Need to pre-process data to increase performance
- Model complexity increases as model size
- Need high computation power to fine-tune bigger models

Broader Impacts

- Opens doors to making multiple models work together
- Proof of concept for making use of multiple pre-trained models in a pipe
- Solve more complex problems efficiently