CS5814: Evaluation of Transformer for Code Summarization

Ashwin Shenolikar
NCR Campus
Virginia Tech

October 10, 2017
Crystal O'Graphie, Chemistry Department, Virginia Tech

VIRGINIA TECH™

Problem Statement

Code summaries and comments are a vital part of code understandability.

Instead of relying on each programmer to generate inherently varying qualitative comments, we can train a model to always generate comprehensible summaries for given code snippets.

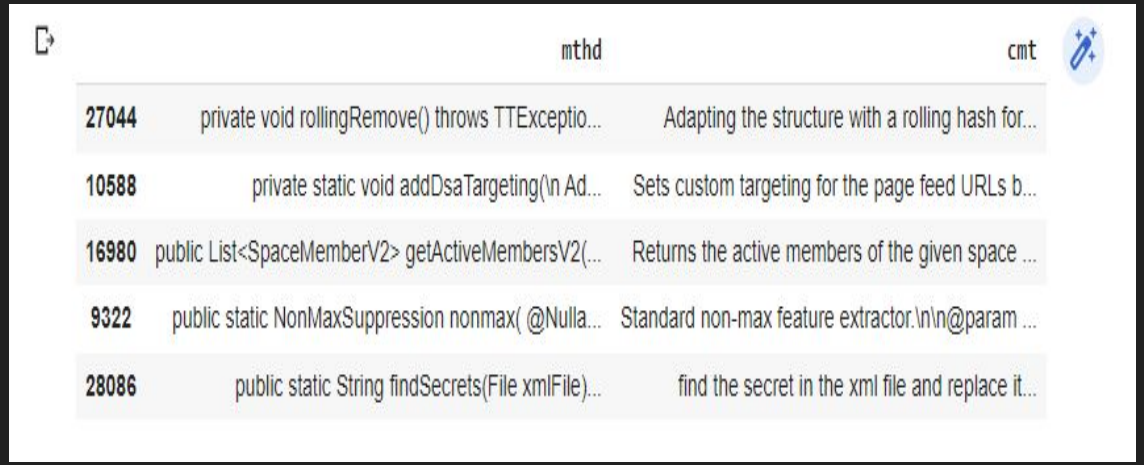
The main task: For a given block of code, such as a function or method, generate a text summarization describing the functionality of said code using transformers.

Data Description

CodeSearchNet: A massive dataset comprising of 2 million code-comment pairs in programming languages: Java, Python, Javascript, Go, Ruby, PHP.

We focus on Java code summarization for the scope of this project.

	Number of Functions	
	w/ documentation	All
Go	347 789	726 768
Java	542 991	1 569 889
JavaScript	157 988	1 857 835
PHP	717 313	977 821
Python	503 502	1 156 085
Ruby	57 393	164 048
All	2 326 976	6 452 446



	mthd	cmt
27044	private void rollingRemove() throws TTExceptio...	Adapting the structure with a rolling hash for...
10588	private static void addDsaTargeting(\n Ad...	Sets custom targeting for the page feed URLs b...
16980	public List<SpaceMemberV2> getActiveMembersV2(...	Returns the active members of the given space ...
9322	public static NonMaxSuppression nonmax(@Nulla...	Standard non-max feature extractor.\n\n@param ...
28086	public static String findSecrets(File xmlFile)...	find the secret in the xml file and replace it...

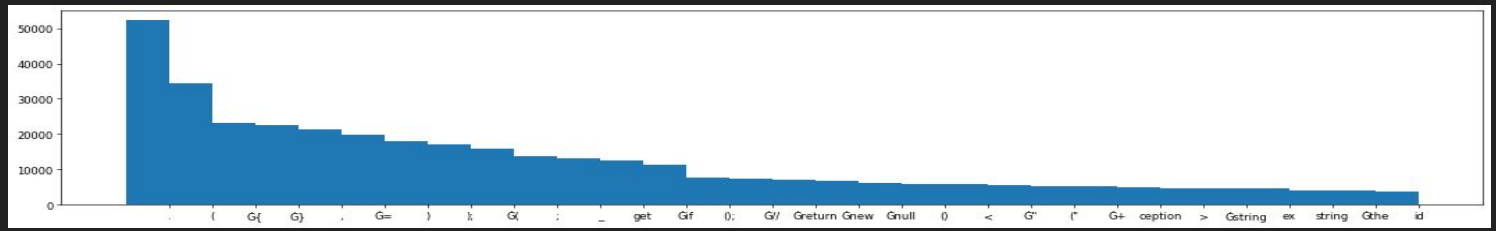
Data Processing and Exploration

Data Processing:

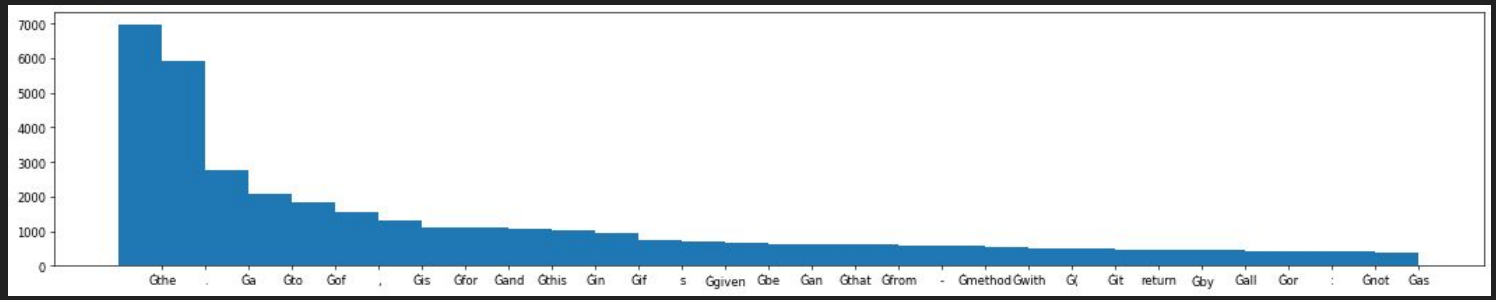
1. Removal of non-ASCII characters for English predictions only.
2. Remove outdated comments(When code is updated but not its comment)
3. Remove very short codes,i.e. if code length is less than comment length.
4. Otherwise,also remove
5. Remove codes which have snippets from other languages, for eg. HTML.

Data Exploration

1. Top 30 most frequent tokens in code



2. Top 30 most frequent tokens in comments



Model Building

High-level procedure:

1. Base model used : Huggingface’s pretrained CodeBERT-base Transformer model.
2. Use different hyperparameters and train on CodeSearchNet, save best bleu scoring models and save state-Dict
3. Use the previously trained model with a Seq2Seq RoBERTa based model. Use this final model for validation and testing.

Evaluation

Based on project proposal feedback, discarded standard evaluation metrics such as f-score, accuracy, et cetera and opted for upto BLEU-4 score and Perplexity score.

Bilingual

The BLEU-4 score we got was

BLEU Score: 21.263

Perplexity: ~78

Lessons Learned

I did not have any prior experience working with Natural Language tasks and so lots of theoretical and practical knowledge gained about state-of-the-art NLP-based models such as Transformers, encoder-decoders in depth.

Introduction to huggingface framework and using it for the project. The framework is amazingly useful for NL tasks.

Realistic usage and actual data processing was required for this project. Even popular datasets such as CodeSearchNet had a lot of code which would confuse the model! Learned how to preprocess data better in this project.

Other techniques to include

Even after multiple data processing techniques, there were still many remaining samples which were not ideal for the model but were still trained on it.

Examples with placeholder comments

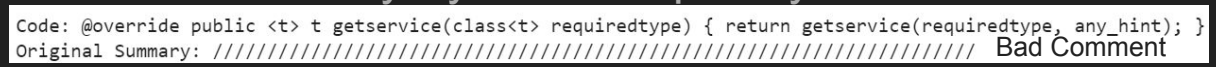
Examples with nonoptimal variable names.

Also, with increased resources available to train the model on so that it can be better trained would be a factor, with more parameters that can be tested.

Challenges faced

Data Cleaning was done but there is still data which is hard to process with coding, such as functions with placeholder texts and comments.

I am sure I missed some data processing techniques which would be very helpful for the model to perform better and would likely try with completely different techniques on a second go.



Apart from that, spent a lot of time understanding and cleaning data. Also, figuring out how to make the model perform better: based on either problematic data or simply misconfiguration of the model.

Overall, it was a challenging problem and could have done a better job with better time management since would not have been stuck on some problems.

Broader Impacts

Deep Learning in the software development sector is a great way to boost productivity and efficiency by automating relatively mundane tasks, such as code commentation and documentation.

Despite being instructed to always comment code, many people do not do it or write illegible comments which do not yield any clarity to the reader. Automating this using well-trained and performing models will have a company not have minor delays in deliverables by eliminating the human component which is liable to underperform on occasion.

Thank You!