

Learning to Recommend Method Names with Global Context

2022 44th IEEE/ACM International Conference on Software Engineering (ICSE)

Presenter: Zhu Jie

2022.6.17

Authors: Key Lab of High Confidence Software Technology, MoE



李戈

北京大学副教授
计算机学院软件研究所

Applications of probabilistic methods for machine learning, including Program Language Processing, Natural Language Processing, and Software Engineering

<https://hcst.pku.edu.cn/jgry/rydw.htm>



金芝

北京大学教授
计算机学院软件研究所

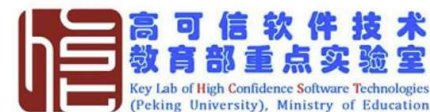
Knowledge engineering and software engineering, knowledge-based software engineering

6. Code Generation as a Dual Task of Code Summarization

会议: NeurIPS 2019.

作者: Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, Zhi Jin

链接: papers.nips.cc/paper/88...



行政机构

实验室主任

梅宏 中国科学院院士, 北京大学教授

实验室常务副主任:

金芝 北京大学教授

实验室副主任:

黄锦辉 香港中文大学教授, 香港分实验室主任

张路 北京大学教授

王腾蛟 北京大学教授

谢涛 北京大学教授

秘书:

徐松青、李连芳

学术委员会

荣誉主任:

杨芙清 中国科学院院士, 北京大学教授

Authors: Key Lab of High Confidence Software Technology, MoE



李戈

北京大学副教授

计算机学院软件研究所

Applications of probabilistic methods for machine learning, including Program Language Processing, Natural Language Processing, and Software Engineering

<https://hcst.pku.edu.cn/jgry/rydw.htm>



金芝

北京大学教授

计算机学院软件研究所

Knowledge engineering and software engineering, knowledge-based software engineering

研究小组及主要骨干人员

1. 可信软件理论研究组

学术带头人：陆汝钤研究员/院士、胡振江教授

学术骨干：许进、王捍贫、焦文品、曹永知教授，陈一峯研究员，刘田副教授，张昕助理教授。

2. 可信软件工程研究组

学术带头人：杨芙清教授/院士，金芝、谢冰、谢涛教授

学术骨干：张路、周明辉教授，赵俊峰研究员，熊英飞、麻志毅、赵海燕、孙艳春、李戈、张伟、邹艳珍副教授。

3. 可信软件平台研究组

学术带头人：梅宏教授/院士，黄罡教授

学术骨干：陈向群、王腾蛟、高军教授，崔斌、曹东刚研究员，郝丹、刘讚哲、金鑫副教授，张颖、陈薇副研究员。

4. 信息安全研究组

学术带头人：陈钟、王平

学术骨干：郭耀教授，关志副研究员，李锭助理教授，王昭讲师。

5. 可信软件应用研究组：

学术带头人：张世琨研究员、张大庆

学术骨干：吴中海、王亚沙、刘云淮教授，赵文、黄雨研究员，刘学洋副教授，胡文蕙、孙基男副研究员，王乐业助理教授，高庆助理研究员。

Why Choose This Paper?

- New: ICSE'22 Paper
- Well-known Method: Transformer-based model
- Idea: Multi-context
- Related Area: a Method Name Recommendation paper
- Open: Replication Package available on GitHub
- Discussion on Model Explainability: how and why?



Background

Background on Deep Learning

How to improve our model performance?

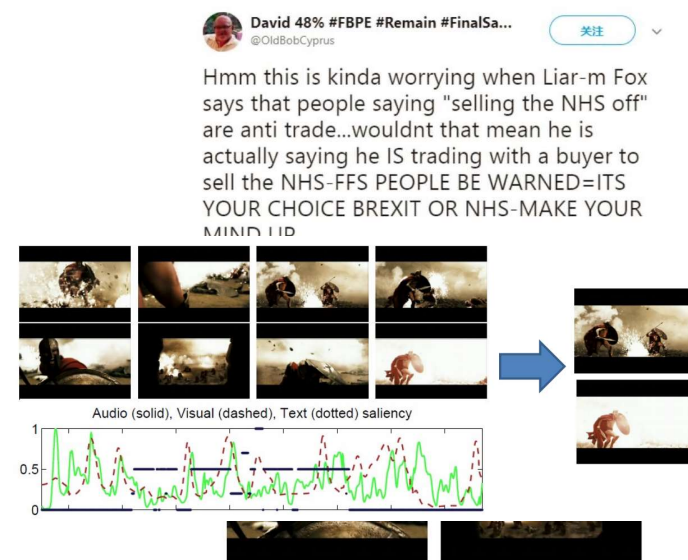
- Larger and Better Dataset: T5 Model (C4 dataset)
- Novel Task Formulation: Code Review Task, Seq2Seq
- Better Preprocess Tricks: BPE(byte pair encoding), Data cleaning, stopwords
- Better Model Design: TextCNN/LSTM => Transformer
- Other tricks: Regularization, Data augmentation, prompt
- Evaluation Metrics: Rouge/Bleu, Human Evaluation
 - The comparison should be fair, which means the baselines must be suitable for the task and the parameters(size) of models should be similar.

Is that all?

Background on Deep Learning

How to improve our model performance?

- Larger and Better Dataset: T5 Model (C4 dataset)
- Novel Task Formulation: Code Review Task, Seq2Seq
- Better Preprocess Tricks: BPE(byte pair encoding), Data cleaning, stopwords
- Better Model Design: TextCNN/LSTM => Transformer
- Other tricks: Regularization, Data augmentation, prompt
- Evaluation Metrics: Rouge/Bleu, Human Evaluation
 - The comparison should be fair, which means the baselines must be suitable for the task and the parameters(size) of models should be similar.

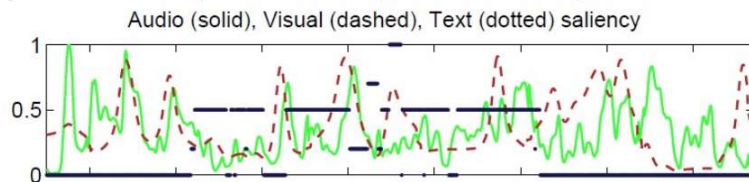
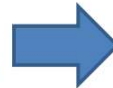


Is that all? No, we could try Multi-Modal

Background on Deep Learning

Multi-Modal Summarization

- Text Summarization could make use of information from video and images.
- Movie Summarization could utilize the features from voice and text subtitles.



David 48% #FBPE #Remain #FinalSa...
@OldBobCyprus

关注

Hmm this is kinda worrying when Liar-m Fox says that people saying "selling the NHS off" are anti trade...wouldnt that mean he is actually saying he IS trading with a buyer to sell the NHS-FFS PEOPLE BE WARNED=ITS YOUR CHOICE BREXIT OR NHS-MAKE YOUR MIND UP



Background

Why is method name recommendation so important?

- Methods are the most **minor named units** for **indicating the program behavior** in most programming languages
- But a good function name can **degrade into a poor one** when the semantics of the function change or the function is used in a new context.
- Meaningful and conventional method names are **vital** for developers to **understand the behavior of programs or APIs**
- Once the name of a method is decided, it is **laborious to change**, especially when used for an API
- Programmers **strive to** choose meaningful and appropriate method names (As researchers found that developers usually change the method names without any change to the corresponding body code in many cases)

Table 8: Examples of generated summaries given Java methods.

Examples	
Method 1	<pre> /** * Adds a path (but not the leaf folder) if it does not already exist. */ protected void ____ (List<String> path, int depth) { int parentSize = path.size() - 1; String name = path.get(depth); Folder child = getChild(name); if (child == null) { child = new Folder(name); ... } } </pre>
Human-written GTNM	<p>"add" "add", "path", "if", "not", "exists"</p>
Method 2	<pre> /** * Append the longs in the array to the selection, each separated by a comma */ private void ____ (long[] objects) { for (int i = 0; i < objects.length; i++) { selection.append(objects[i]); if (i != objects.length - 1) { selection.append(','); } } } </pre>
Human-written GTNM	<p>"join", "in", "selection" "append", "selection"</p>
Method 3	<pre> /** * Calculates the DefinitionUseCoverage fitness for the given DUPair on the given ExecutionResult */ public double ____ () { if (isSpecialDefinition (goalDefinition)) return calculateUseFitnessForCompleteTrace (); double defFitness = calculateDefFitnessForCompleteTrace (); if (defFitness != 0) return 1 + defFitness; return calculateFitnessForObjects (); } </pre>
Human-written GTNM	<p>"calculate", "d", "u", "fitness" "calculate", "fitness", "for"</p>
Method 4	<pre> /** * Validate removal of invalid entries. */ public void ____ () { RightThreadedBinaryTree<Integer> bt = new RightThreadedBinaryTree<Integer> (); assertFalse (bt.remove(99)); bt = buildComplete(4); assertFalse (bt.remove(99)); assertFalse (bt.remove(-2)); } </pre>
Human-written GTNM	<p>"test", "invalid", "removals" "test", "remove", "invalid"</p>

Background

Characteristics of Method Name

- **Significance:** Meaningful and conventional method names are vital for developers to **understand the behavior of programs or APIs**
- **Dynamicity:** But a good function name can **degrade into a poor** one when the semantics of the function change or the function is used in a new context
- **Hard to change:** Once the name of a method is decided, it is **laborious to change**, especially when used for an API
- **Consistency within project:** Consistency with other names in the same project or the codebase is also important. Especially when collaborating, they need to obey a **project's coding conventions**.

Table 8: Examples of generated summaries given Java methods.

Examples	
Method 1	<pre> /** * Adds a path (but not the leaf folder) if it does not already exist. */ protected void ____ (List<String> path, int depth) { int parentSize = path.size() - 1; String name = path.get(depth); Folder child = getChild(name); if (child == null) { child = new Folder(name); ... } } </pre>
Human-written GTNM	<p>"add" "add", "path", "if", "not", "exists"</p>
Method 2	<pre> /** * Append the longs in the array to the selection, each separated by a comma */ private void ____ (long[] objects) { for (int i = 0; i < objects.length; i++) { selection.append(objects[i]); if (i != objects.length - 1) { selection.append(','); } } } </pre>
Human-written GTNM	<p>"join", "in", "selection" "append", "selection"</p>
Method 3	<pre> /** * Calculates the DefinitionUseCoverage fitness for the given DUPair on the given ExecutionResult */ public double ____ () { if (isSpecialDefinition (goalDefinition)) return calculateUseFitnessForCompleteTrace (); double defFitness = calculateDefFitnessForCompleteTrace (); if (defFitness != 0) return 1 + defFitness; return calculateFitnessForObjects (); } </pre>
Human-written GTNM	<p>"calculate", "d", "u", "fitness" "calculate", "fitness", "for"</p>
Method 4	<pre> /** * Validate removal of invalid entries. */ public void ____ () { RightThreadedBinaryTree<Integer> bt = new RightThreadedBinaryTree<Integer> (); assertFalse (bt.remove(99)); bt = buildComplete(4); assertFalse (bt.remove(99)); assertFalse (bt.remove(-2)); } </pre>
Human-written GTNM	<p>"test", "invalid", "removals" "test", "remove", "invalid"</p>

Background

Method Name Recommendation Task

- Given a method definition and its body(implementation)
- Predict/Recommend a method name

Task Formulation/Method

- Code Summarization
- Information Retrieval
- Template-based
- Language Model
- Graph Neural Network
-

Context (Model Input)

- Function Signature(Definition)
- Method Body(Implementation)
-

Table 8: Examples of generated summaries given Java methods.

Examples	
Method 1	<pre> /** * Adds a path (but not the leaf folder) if it does not already exist. */ protected void ____ (List<String> path, int depth) { int parentSize = path.size() - 1; String name = path.get(depth); Folder child = getChild(name); if (child == null) { child = new Folder(name); ... } } </pre>
Human-written GTNM	<p>"add" "add", "path", "if", "not", "exists"</p>
Method 2	<pre> /** * Append the longs in the array to the selection, each separated by a comma */ private void ____ (long[] objects) { for (int i = 0; i < objects.length; i++) { selection.append(objects[i]); if (i != objects.length - 1) { selection.append(','); } } } </pre>
Human-written GTNM	<p>"join", "in", "selection" "append", "selection"</p>
Method 3	<pre> /** * Calculates the DefinitionUseCoverage fitness for the given DUPair on the given ExecutionResult */ public double ____ () { if (isSpecialDefinition (goalDefinition)) return calculateUseFitnessForCompleteTrace (); double defFitness = calculateDefFitnessForCompleteTrace (); if (defFitness != 0) return 1 + defFitness; return calculateFitnessForObjects (); } </pre>
Human-written GTNM	<p>"calculate", "d", "u", "fitness" "calculate", "fitness", "for"</p>
Method 4	<pre> /** * Validate removal of invalid entries. */ public void ____ () { RightThreadedBinaryTree<Integer> bt = new RightThreadedBinaryTree<Integer> (); assertFalse (bt.remove(99)); bt = buildComplete(4); assertFalse (bt.remove(99)); assertFalse (bt.remove(-2)); } </pre>
Human-written GTNM	<p>"test", "invalid", "removals" "test", "remove", "invalid"</p>

Related Work: IR-based Method

ICSE19' Learning to Spot and Refactor

- Paper Information: ICSE'19 (from University of Luxembourg)
- Motivation: To reduce the manual efforts of resolving inconsistent method names
- Idea: propose a novel automated approach to debugging method names based on the analysis of consistency between method names and method code
- Technique: CNN + Information Retrieval

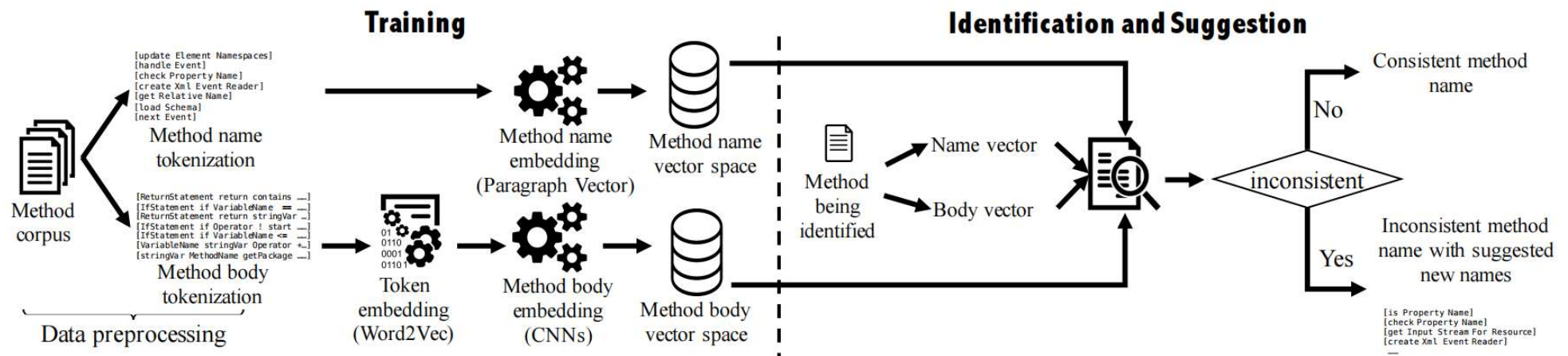


Fig. 3. Overview of our approach to spotting and refactoring inconsistent method names.

Related Work: Code Summarization

ICML16' A Convolutional Attention Network for Extreme Summarization of Source Code

- Paper Information: **Miltiadis Allamanis** (University of Edinburgh)
- Task: Code summarization(predict a short and descriptive name of a source code snippet)
- Technique: CNN + attention (Machine translation model)
- Detail: This problem resembles a summarization task, where the method name is viewed as the summary of the code. However, extreme source code summarization is drastically different from natural language

<code>boolean shouldRender()</code>	<code>void reverseRange(Object[] a, int lo, int hi)</code>
<pre>try { return renderRequested isContinuous; } finally { renderRequested = false; }</pre>	<pre>hi--; while (lo < hi) { Object t = a[lo]; a[lo++] = a[hi]; a[hi--] = t; }</pre>
<p>Suggestions:</p> <ul style="list-style-type: none"> ▶ is, render (27.3%) ▶ is, continuous (10.6%) ▶ is, requested (8.2%) ▶ render, continuous (6.9%) ▶ get, render (5.7%) 	<p>Suggestions:</p> <ul style="list-style-type: none"> ▶ reverse, range (22.2%) ▶ reverse (13.0%) ▶ reverse, lo (4.1%) ▶ reverse, hi (3.2%) ▶ merge, range (2.0%)

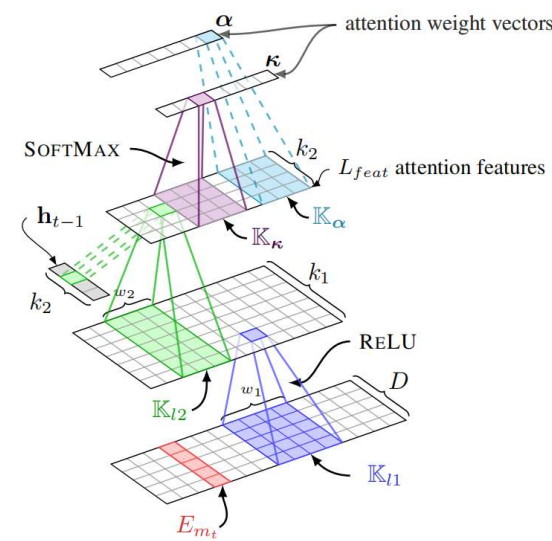




Figure 1. The architecture of the convolutional attentional network. **attention_features** learns location-specific attention features given an input sequence $\{m_i\}$ and a context vector h_{t-1} . Given these features **attention_weights** —using a convolutional layer and a SOFTMAX— computes the final attention weight vectors such as α and κ in this figure.

A Top Researcher

Miltiadis Allamanis Homepage



Miltos Allamanis
Researcher
Machine Learning for Code
Home
Publications
Talks, Tutorials, etc
Quotes I Like
Software
Contact me



- 🚀 I research AI tools and methods that learn from highly structured data. Given the central role of software in our lives, the challenges in developing it, and its rich structure, I aim to create AI that makes creating and maintaining software productive, enjoyable, and less error-prone.
- 💡 I work on machine learning methods for structured data, specifically deep learning methods for source code, programming languages, and software engineering.
- 👉 I enjoy modeling and solving real-world problems using machine learning and optimization methods.

📁 Learning for Software Engineering

- 🔍 Learned program analyses (program understanding, finding bugs 🐛)
- 📄 Code generation and synthesis
- 🧠 Machine learning models, representations, and learning methods
- 🔧 Practical developer tools

🧠 Learning for Structured & Dynamic Data

- 📊 Understanding, reasoning about, and predicting structured data
- 🤖 Graph Neural Networks and Transformers







🔥 Highlighted Publications (👉 Full List)

- 📄 [Learning to Complete Code with Sketches](#). 2022
- 📄 [HEAT: Hyperedge Attention Networks](#). 2022
- 📄 [Self-Supervised Bug Detection and Repair](#). 2021
- 📄 [Graph Neural Networks on Program Analysis](#). 2021
- 📄 [Fast and Memory-Efficient Neural Code Completion](#). 2021
- 📄 [Learning to Represent Edits](#). 2019
- 📄 [A Survey of Machine Learning for Big Code and Naturalness](#). 2018
- 📄 [Learning to Represent Programs with Graphs](#). 2018





📖 Publications

Below you can find a list of my publications, ordered chronologically.

2022

	CoRGi: Content-Rich Graph Neural Networks with Attention. Jooyeon Kim, Angus Lamb, Simon Woodhead, Simon Peyton Jones, Cheng Zheng, Miltiadis Allamanis. KDD 2022. TLDR: A method to embed content information within nodes in a GNN. Evaluated on imputation scenarios.
	AdaptivePaste: Code Adaptation through Learning Semantics-aware Variable Usage Representations. Xiaoyu Liu, Jinu Jang, Neel Sundaresan, Miltiadis Allamanis, Alexey Svyatkovskiy. 2022. TLDR: Learn to adapt pasted snippets within some code context using transformers.
	Deep End-to-end Causal Inference. Tomas Geffner, Javier Antoran, Adam Foster, Wenbo Gong, Chao Ma, Emre Kiciman, Amit Sharma, Angus Lamb, Martin Kukla, Nick Pawlowski, Miltiadis Allamanis, Cheng Zhang. 2022. TLDR: Causal Discovery and Inference End-to-End.
	HEAT: Hyperedge Attention Networks. Dobrik Georgiev, Marc Brockschmidt, Miltiadis Allamanis. 2022. TLDR: Generalize transformers and GNN to typed and qualified hypergraphs.
	Learning to Complete Code with Sketches. Daya Guo, Alexey Svyatkovskiy, Jian Yin, Nan Duan, Marc Brockschmidt, Miltiadis Allamanis. ICLR 2022. TLDR: Automatically generate (code) sketches, placing holes where ambiguity prevents us predicting terminal tokens.
	NS3: Neuro-Symbolic Semantic Code Search. Shushan Arakelyan, Anna Hakhverdyan, Miltiadis Allamanis, Christophe Hauser, Luis Garcia, Xiang Ren. 2022. TLDR: The natural language query is parsed and its structure instantiates neural modules that break down the search problem.

2021

	Copy that! Editing Sequences by Copying Spans. Sheena Panthapackel, Miltiadis Allamanis, Marc Brockschmidt. AAAI 2021. TLDR: Learn seq2seq models that can edit sequence by copying long spans.
	Fast and Memory-Efficient Neural Code Completion. A. Svyatkovskiy, S. Lee, A. Hadjitofi, M. Riechert, J. Franco, M. Allamanis. Mining Software Repositories 2021. TLDR: Lightweight yet accurate code completion use neural reranking models.
	Graph Neural Networks on Program Analysis. M. Allamanis. Graph Neural Networks: Foundations, Frontiers, and Applications 2021. TLDR: A survey of GNNs for learned program analyses
	Self-Supervised Bug Detection and Repair. M. Allamanis, H. Jackson-Flux, M. Brockschmidt. NeurIPS 2021. TLDR: Learn to detect a variety of bugs in source code by asking two models to play a hide-and-seek game: one model inserts a bug, the other tries to find it.

<https://miltos.allamanis.com/publicationlist.html>

Related Work: Summarization-based Method

ICSE20' Mnire: Suggesting Natural Method Names to Check Name Consistencies

- Paper Information: Son Nguyen (University of Texas-Dallas)
- Task: Code summarization(predict a short and descriptive name of a source code snippet)
- Technique: Summarization-based Method (encoder-decoder)
- Idea: MNire, a machine learning approach to check the consistency between the name of a given method and its implementation.
- Detail: MNire first generates a candidate name and compares the current name against it. If the two names are sufficiently similar, we consider the method as consistent.
- Contribution: **Context Extraction**
 - ENC Context: enclosing(class name)
 - INF Context: interface (method input/output)
 - IMP Context: implementation (set of tokens of the variables/fields and methods that are used in the method body)

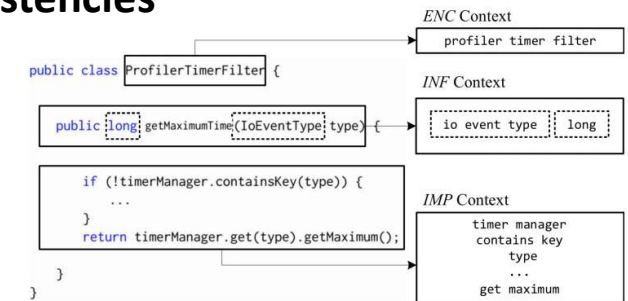


Figure 6: Context Extraction

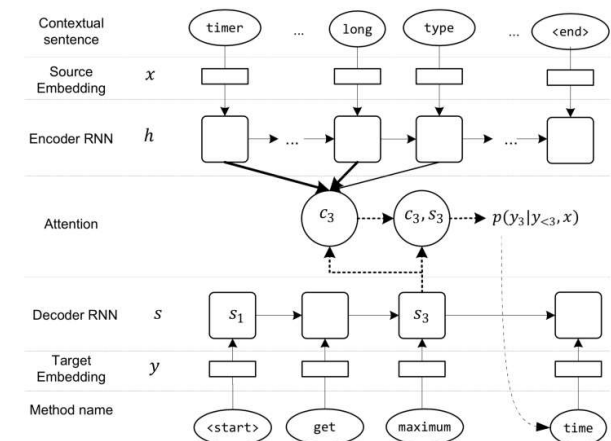
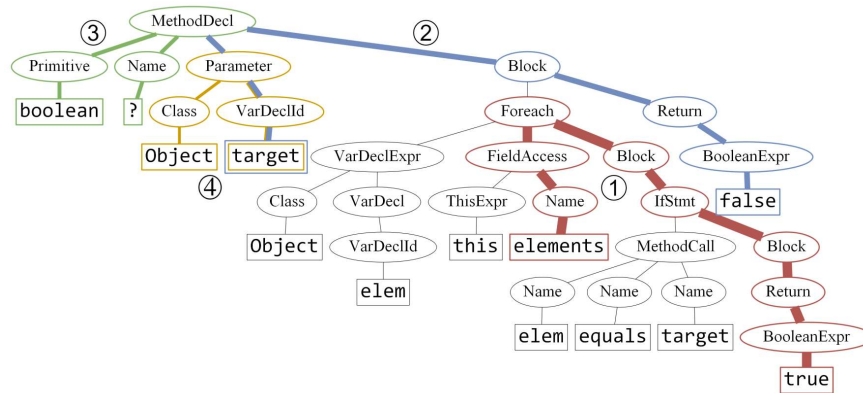


Figure 7: Abstractive Summarization Model

Related Work: AST Structure Context

ICLR19' code2vec POPL19' Code2Vec



CODE2SEQ: GENERATING SEQUENCES FROM STRUCTURED REPRESENTATIONS OF CODE

Uri Alon
Technion
urialon@cs.technion.ac.il

Shaked Brody
Technion
shakedbr@cs.technion.ac.il

Omer Levy
Facebook AI Research
omerlevy@gmail.com

Eran Yahav
Technion
yahave@cs.technion.ac.il

code2vec: Learning Distributed Representations of Code

URI ALON, Technion
MEITAL ZILBERSTEIN, Technion
OMER LEVY, Facebook AI Research
ERAN YAHAV, Technion

Related Work: Code Summarization

ICSE21' A Context-based Automated Approach for Method Name Consistency Checking and Suggestion

- Steps:
 1. Context building: Break Var/Method Name into sub-tokens
 2. Context Representation: Convert sub-tokens into vector representation
 3. Context-based Method Representation Learning: The model considers **four contexts: internal/interaction/sibling/enclosing**
 4. Consistency Checking and Method Name Suggestion: CNN two-channel classifier + modified RNN Model(non-copy mechanism)
- Example on Fig.1: This example shows a common case in which **during the course of software development, the name of a method has become confusing** and inconsistent with its functionality. Thus, an automated tool to detect inconsistent method names is helpful for developers to avoid confusing and mistakes.

```
1 private void declareGrouping(BoltDeclarer boltDeclarer,
    Node parent, String streamId, GroupingInfo
    grouping) {
2 // the old inconsistent method name is declareStream
3 if (grouping == null) {
4     boltDeclarer.shuffleGrouping(parent.getComponentId(),
        streamId);
5 } else {
6     grouping.declareGrouping(boltDeclarer, parent.get
        ComponentId(), streamId, grouping.getFields());
7 }
8 }
```

Fig. 1: An Example of Inconsistent Method Name

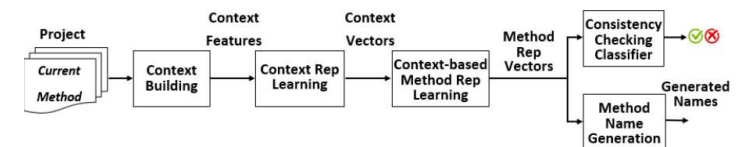


Fig. 3: Overview of DEEPNAME's Architecture

Motivation: How to improve these models

More Context Information(Multi-Modal), More Powerful Model(transformer-based)

- **Limitation 1:** The information of the **whole project (global context)** is ignored in these models
 - A source code file can have references to other files of the same project.
 - Thus, the contexts from other program files which are imported by the file where the target method in are also helpful in understanding the methods.
 - By referring to the global contextual information, the solution space of the method names can be narrowed
- **Limitation 2:** The documentation/**comment of the method** is also ignored
 - the documentation of the method can describe the method's functionality and the role it plays in the project
- **Limitation 3:** RNN has difficulties in remembering long-term dependencies.
 - Transformer is all you need 😊

GTNM: Global Transformer Neural Network

Motivation: Global Context

Why is Global Context so important?

- 1、可以引入项目中其他方法的信息来帮助理解目标方法的意义和行为： A source code file can have references to other files of the same project. Thus, the contexts from other program files which are imported by the file where the target method in are also helpful in understanding the methods.
- 2、文档/注释的重要性： The documentation of the method can describe the method's functionality and the role it plays in the project.
- 3、可以与项目中其他代码方法命名格式保持一致： During the code refinement, the global context can be used to suggest an alternative name if the current name is inconsistent.
- 4、缩小推荐命名的候选空间： By referring to the global contextual information, the solution space of the method names can be narrowed.
- 5、总结 Summary: Thus, when recommending a method name, it is necessary to refer to the global contexts. It can help in following situations: when the method is first created, existing global context can be accessed for suggesting a proper name for it; during the code refinement, the global context can be used to suggest an alternative name if the current name is inconsistent



Design and Definition

Definition: What is context?

Contexts of three different levels

Local Context: we extract the following code entities as the local contexts for the method: **(1) identifiers;** **(2) parameters;** **(3) return type.** We tokenized each of the names from the local contexts following camelcase and underscore naming conventions, then normalized the tokens to lowercase. Finally, all the subtokens are concatenated in the order that they occurred in the source code to form the sequential representation of the local feature.

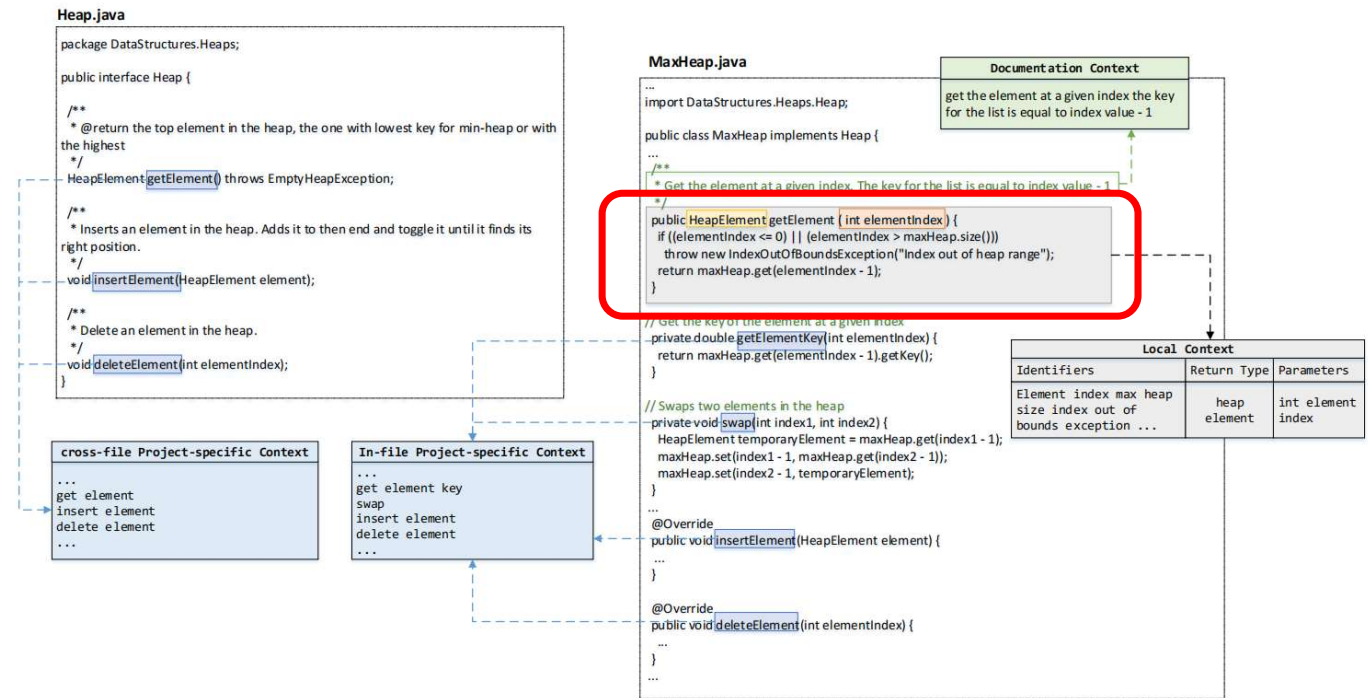


Figure 2: Different levels of contexts for method name suggestion.

Definition: What is context?

Contexts of three different levels

Project-specific Context Extraction: We define the project-specific context of one method as its **in-file methods** (other methods in the same file with the target method) and **cross-file contextual methods** (methods in the files imported by the file containing the target method). For simplicity and efficiency, we **extract the name of the contextual methods as the project-specific context**. Then we perform a similar process to these names as to local context. The concatenation of the lower-cased subtokens serves as the representation of the project-specific feature.

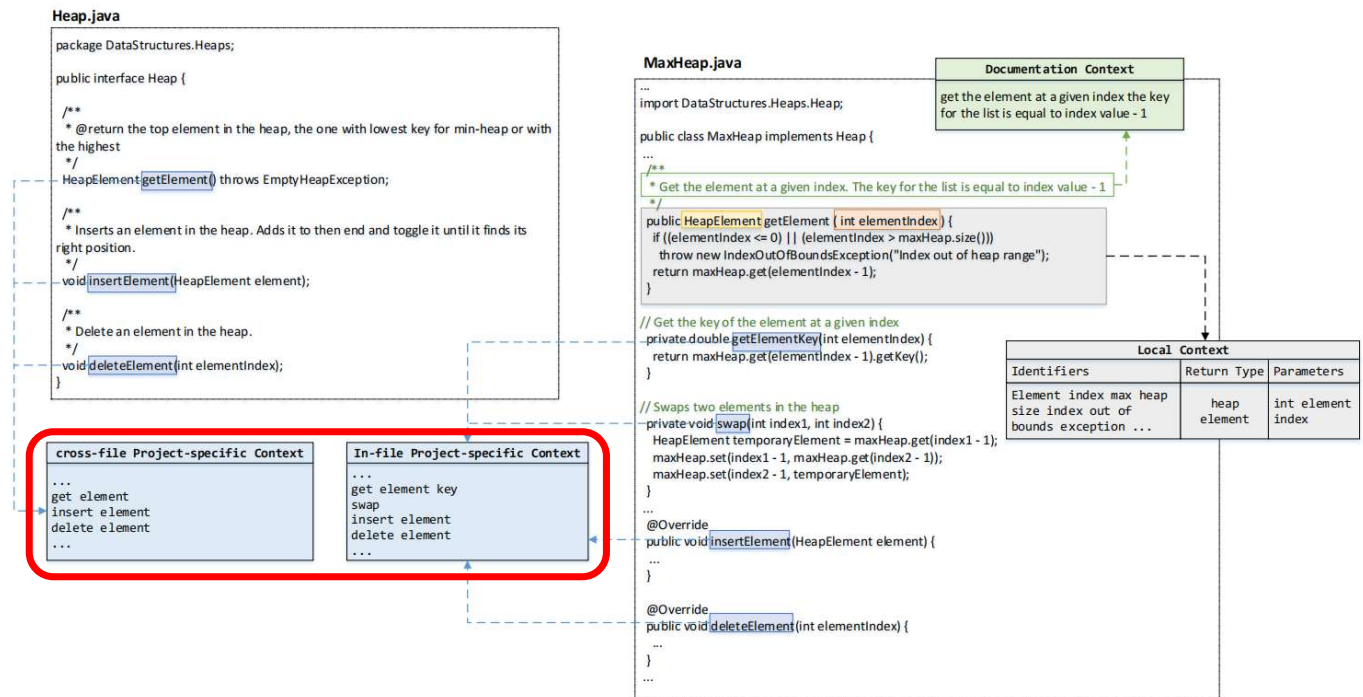


Figure 2: Different levels of contexts for method name suggestion.

Definition: What is context?

Contexts of three different levels

Documentation Context Extraction: For each method with a comment, to get its documentation context, we **extract the first sentence that appeared in its Javadoc description** since it typically describes the functionalities of the method. Then we delete the punctuations and split the sentence with space to get words and lowercase the words. All the words are concatenated to form the documentation context.

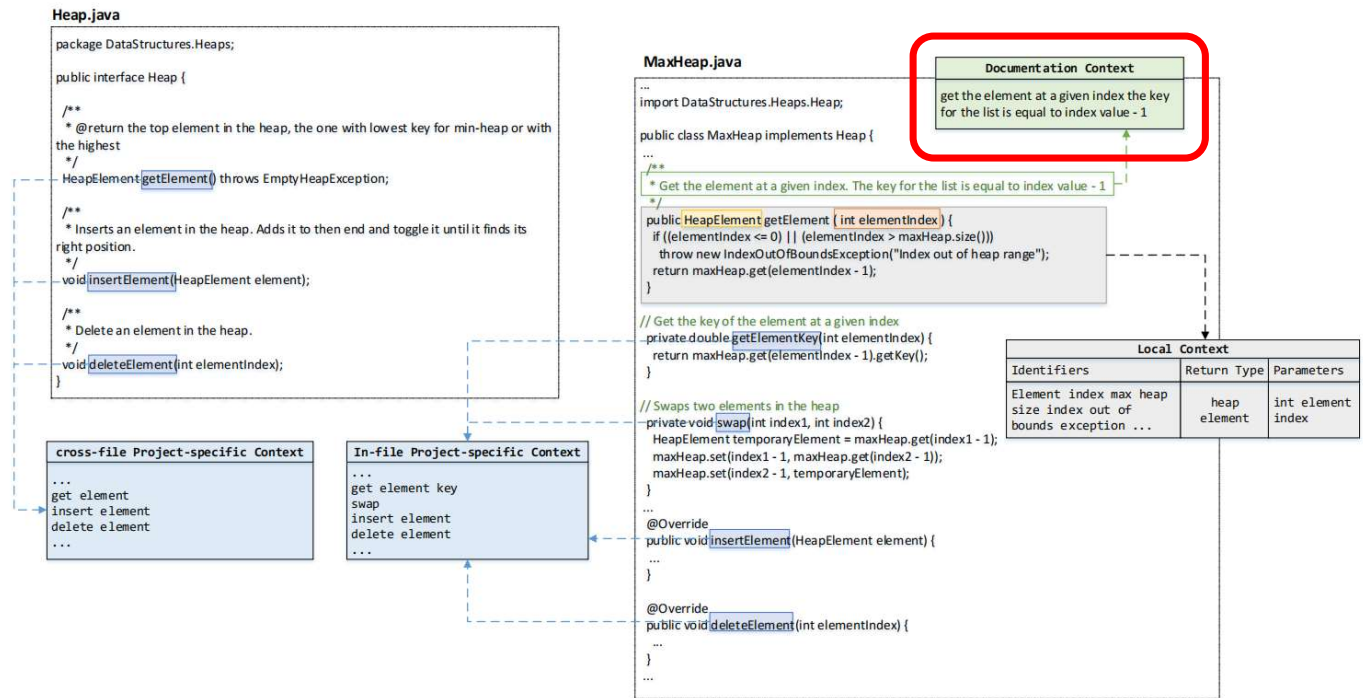


Figure 2: Different levels of contexts for method name suggestion.

Definition: What is context?

Contexts of three different levels

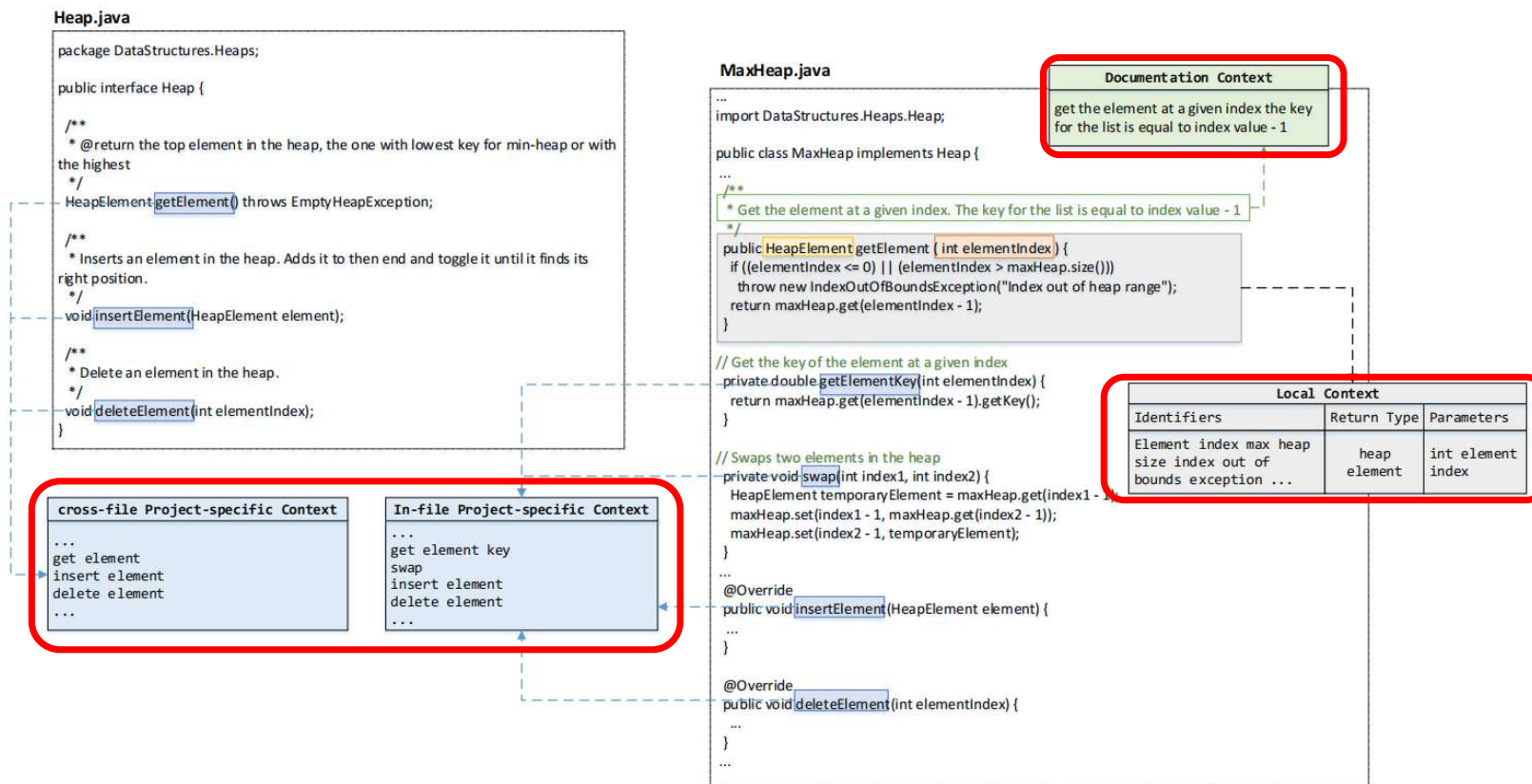


Figure 2: Different levels of contexts for method name suggestion.

Model Overview

GTNM: Global Transformer-based Neural Network

- **Context Extraction:** Local/Document/Project-Specific Context
- **Encoder:** Code Encoder and Project Context Encoder
- **Invoked Weight:** to emphasize some methods in the same project
- **Decoder**

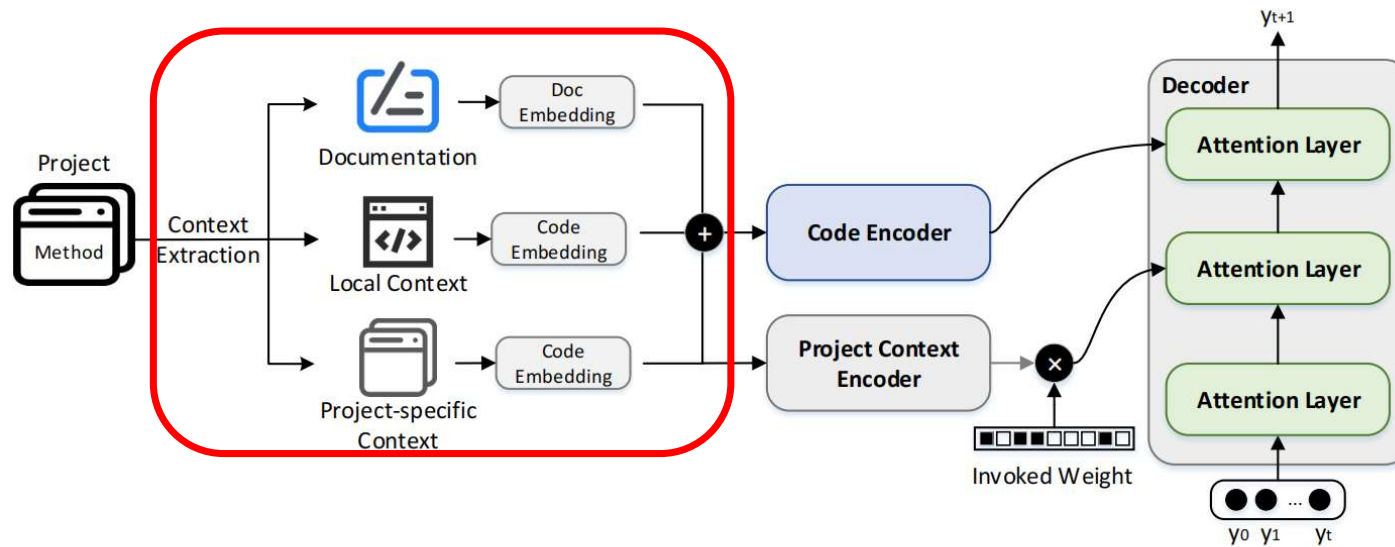


Figure 1: The overall framework of GTNM.

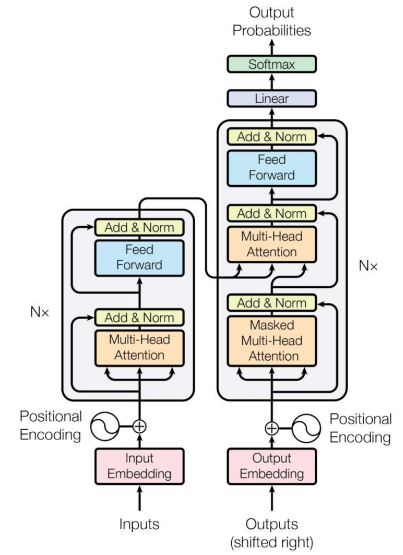


Figure 1: The Transformer - model architecture.

Transformer

- Seq2Seq: Encoder-Decoder (RNN/LSTM)
- Self Attention
- Layer Normalization(compared to Batch Normalization)
- Masked Multi-Head Attention
- Implementation using Tensor2Tensor

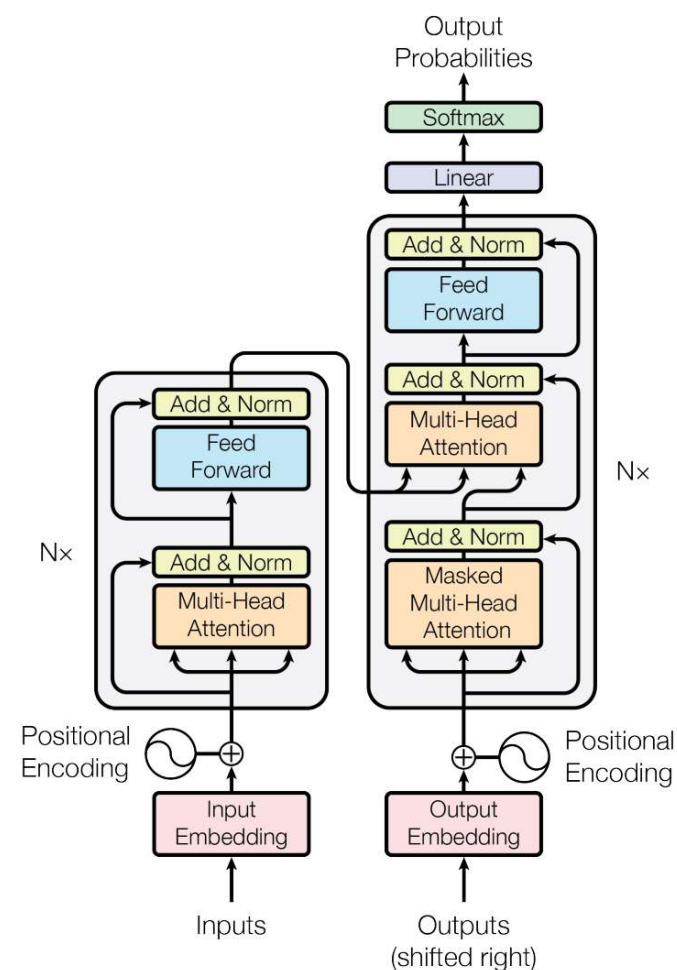


Figure 1: The Transformer - model architecture.

[1] 国科大自然语言处理 (刘洋) <https://www.bilibili.com/video/BV1qy4y1r7M7>
[2] 李宏毅2021机器学习 Self-Attention机制 <https://www.bilibili.com/video/BV1154y1J76o?p=9>
[3] Transformer论文逐段精读【论文精读】李沐 <https://www.bilibili.com/video/BV1pu411o7BE>
[4] 斯坦福cs224n word2vec介绍: <https://www.bilibili.com/video/BV1pt411h7aT?p=2>
[5] <https://github.com/km1994/NLP-Interview-Notes/tree/main/DeepLearningAlgorithm/transformer>

Transformer

- Seq2Seq: Encoder-Decoder (RNN/LSTM)

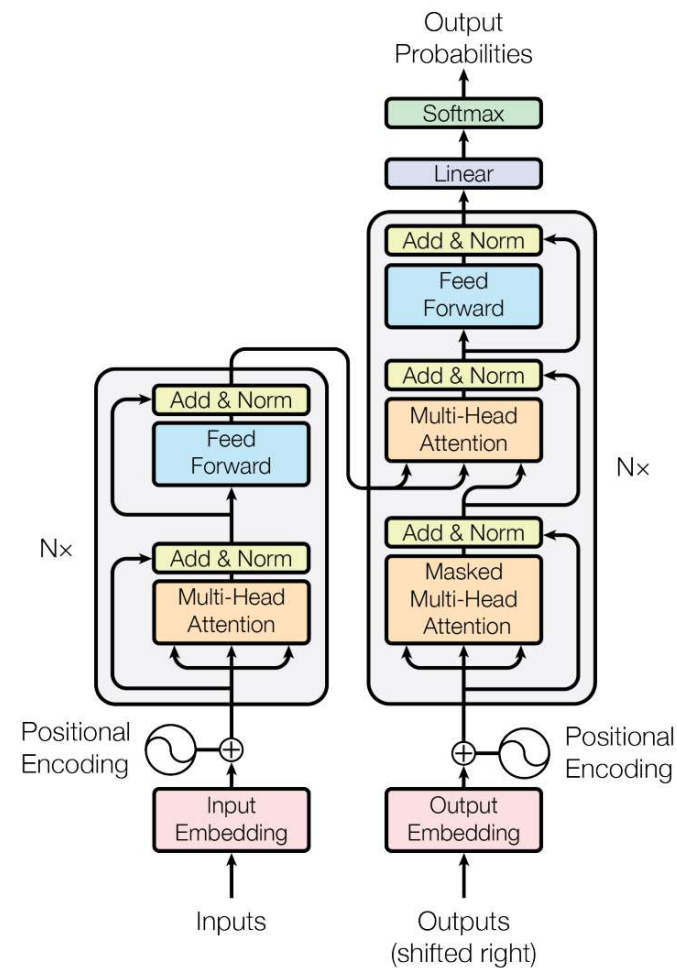
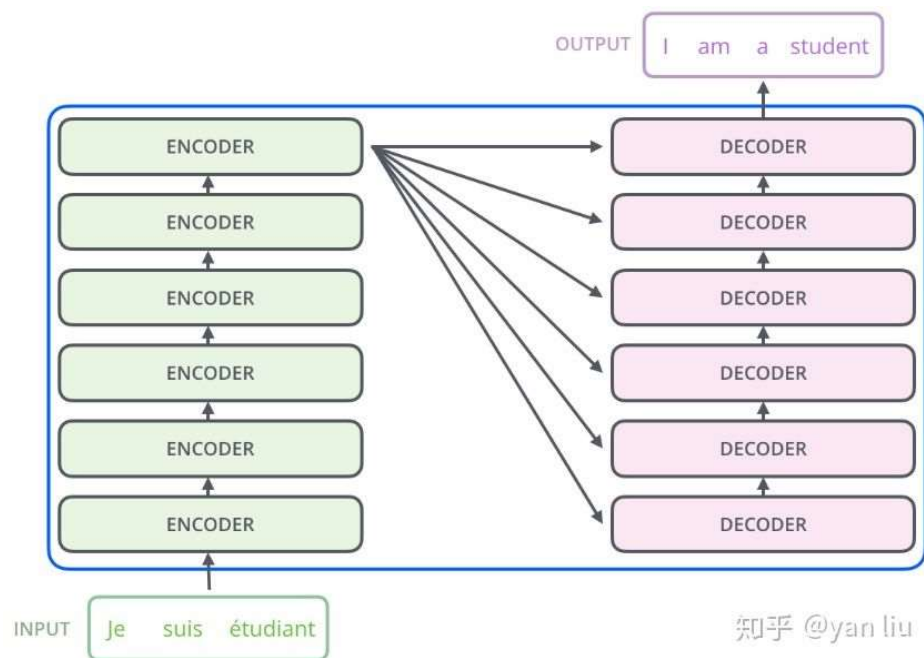


Figure 1: The Transformer - model architecture.

Model Encoder

GTNM: Global Transformer-based Neural Network

- **Encoder Design: Code Encoder(All Context) + Project Context Encoder(project-specific context)**
- We build a Code Encoder to encode the whole context x including the local context, project-specific context, and documentation for the method name generation, and **build an extra Project Context Encoder to encode the project context x_{pro} for enhancing the attention to the project-specific context.**

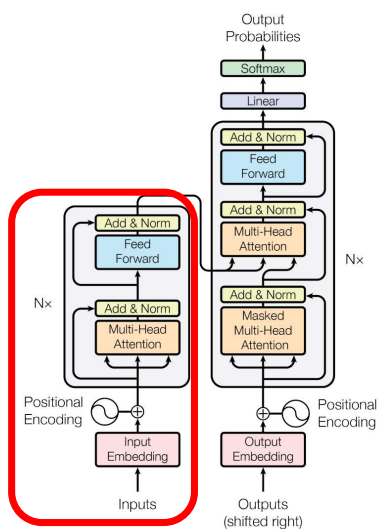


Figure 1: The Transformer - model architecture.

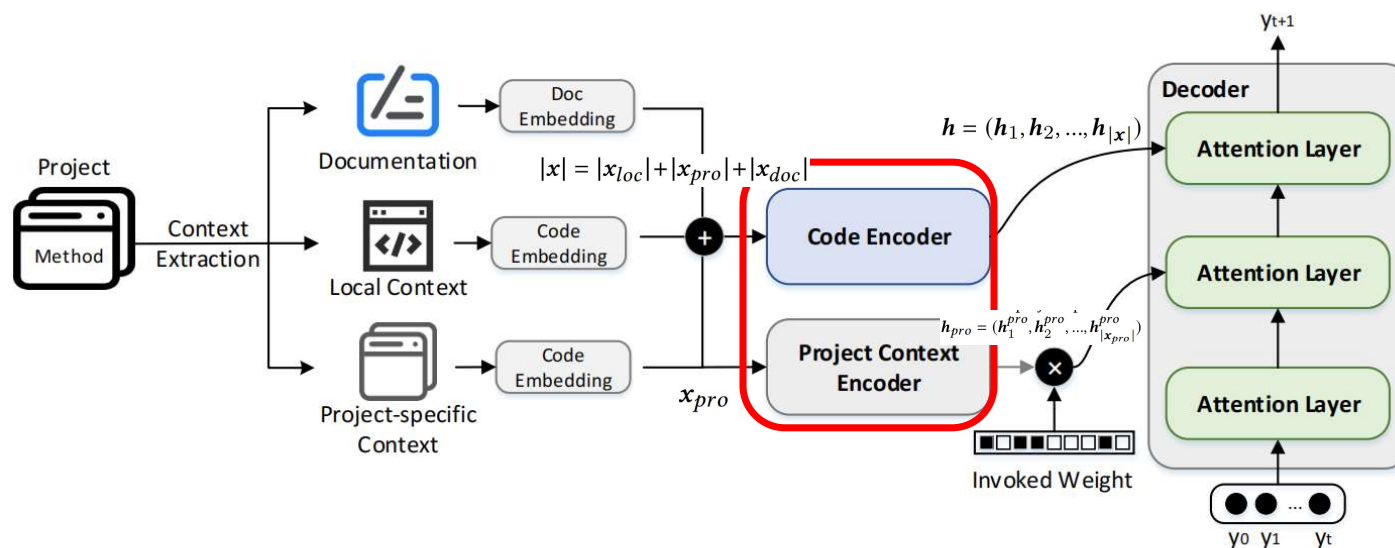


Figure 1: The overall framework of GTNM.

Model Design: Invoked Weight

GTNM: Global Transformer-based Neural Network

- **Invoked Weight:** We use a mask vector M to record the methods that are invoked by the local context. M_i is 1 if the i -th method in the project-specific context is invoked by the local context else is 0
- **Reason:** Intuitively, the methods in the project-specific context invoked by the local context are more important and relative to the target method. Thus we give these methods more attention by multiplying the invoked weight on the project-specific hidden vector

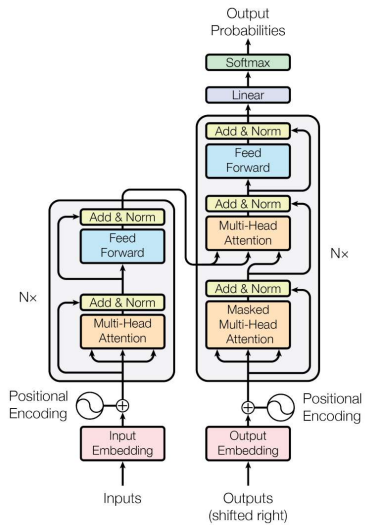


Figure 1: The Transformer - model architecture.

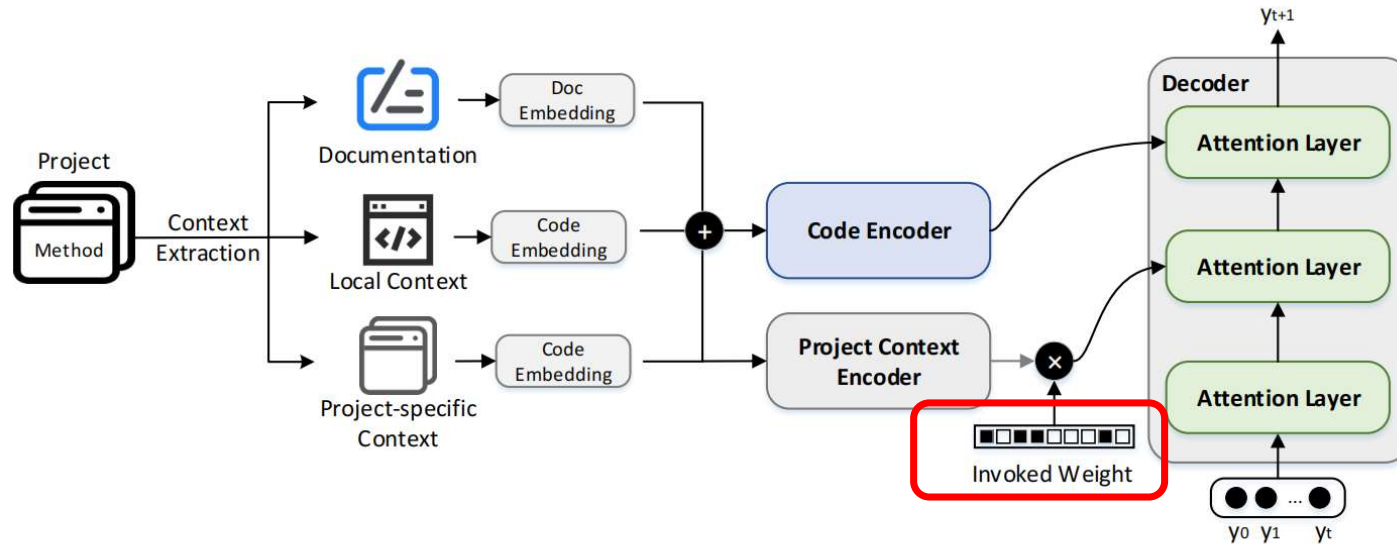
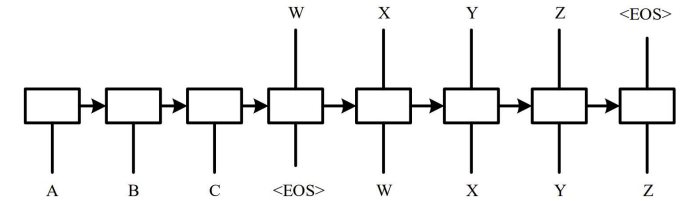


Figure 1: The overall framework of GTNM.

Model Decoder

GTNM: Global Transformer-based Neural Network

- Decoder: multi-head attention
- The decoder aims to generate the target method name by sequentially predicting the subtoken y_{t+1} conditioned on the context vectors \mathbf{h} and $\tilde{\mathbf{h}}_{pro}$, and the previous generated subtokens $\mathbf{y}_{1:t}$



$$p(y_{t+1}) = \text{softmax}(FFN(\mathit{dec}_2))$$

$$\mathit{dec}_2 = \text{Attention-Layer3}(\mathbf{h}, \mathit{dec}_1)$$

$$\mathit{dec}_1 = \text{Attention-Layer2}(\tilde{\mathbf{h}}_{pro}, \mathit{dec})$$

$$\mathit{dec} = \text{Attention-Layer1}(\mathbf{y}_{1:t})$$

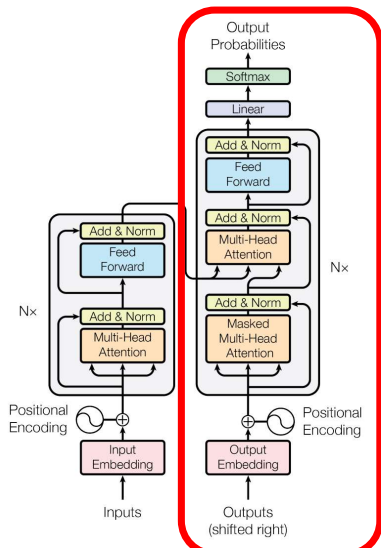


Figure 1: The Transformer - model architecture.

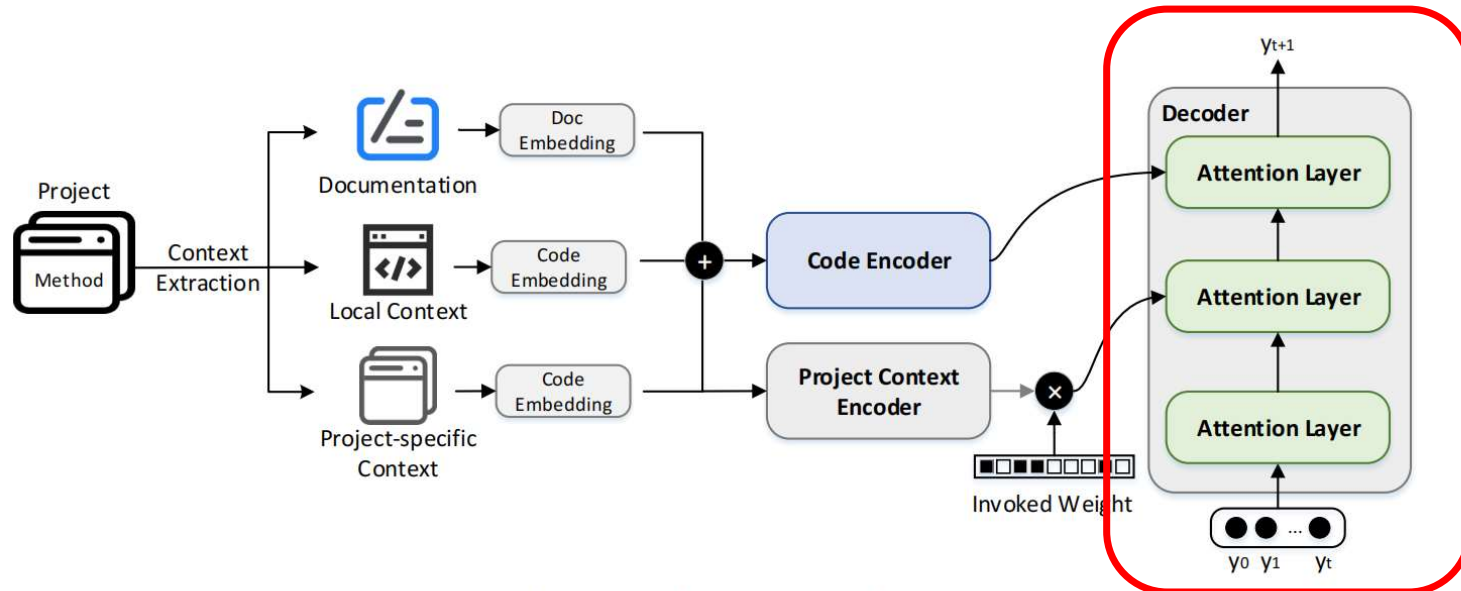


Figure 1: The overall framework of GTNM.

Experiment

Experiment: Dataset

Dataset from Mnire and Code2vec

- Nguyen et al. provide the list of java repositories, which contains 10K top-ranked, public Java projects on GitHub. They used the same setting as in code2vec to shuffle files in all the projects and split them into 1.7M training and 61K testing files. Following their setting, we download the repositories they provide and follow the same way to build the dataset.

Table 1: Statistics of the datasets.

	Train	Validation	Test
Files	1,700,000	393,327	61,000
Methods	18,230,509	4,283,580	636,816
Methods with doc	4,264,852	964,078	143,913

```
/** encodes a bounding box into the provided byte array */
private static void encode(final int minX, final int maxX, final int minY, final int maxY, byte[] b) {
    if (b == null) {
        b = new byte[4 * LatLonShape.BYTES];
    }
    LatLonShape.encodeTriangleBoxVal(minY, b, 0);
    LatLonShape.encodeTriangleBoxVal(minX, b, BYTES);
    LatLonShape.encodeTriangleBoxVal(maxY, b, 2 * BYTES);
    LatLonShape.encodeTriangleBoxVal(maxX, b, 3 * BYTES);
}

@Override
protected Relation relateRangeBBoxToQuery(int minXOffset, int minYOffset, byte[] minTriangle,
                                           int maxXOffset, int maxYOffset, byte[] maxTriangle) {
    Relation eastRelation = compareBBoxToRangeBBox(this.bbox, minXOffset, minYOffset, minTriangle, maxXOffset,
    maxYOffset, maxTriangle);
    if (this.crossesDateline() && eastRelation == Relation.CELL_OUTSIDE_QUERY) {
        return compareBBoxToRangeBBox(this.west, minXOffset, minYOffset, minTriangle, maxXOffset, maxYOffset,
        maxTriangle);
    }
    return eastRelation;
}
```


Experiment Detail

Some details of the experiment

- We use Transformer with 6 layers, hidden size 512, and 8 attention heads for both encoders and decoders. The inner hidden size of the feed-forward layer is 2048. (Parameters of model: $12 * 512 * 64 * 8 * 4 + 512 * 2048 * 2 + \text{vocab_size} * 512$, smaller than BERT-base)
 - Vocab size is 30000 (20000 for source code, 10000 for documentation/comment), OOV tokens are replaced by <UNK>
 - We use javalang2 to parse the java code to extract the contexts
 - Tesla V100 (16GB) for 20 epochs
-
- Truncation: In our experiments, we set the in-file project-specific context length to 30, the cross-file project-specific context length to 30, the local context length to 55 (variable length (50) + parameter and return type length (5)), the documentation context length to 10. And the maximum target name length is set to 5

javalang

build passing pypi package 0.13.0

javalang is a pure Python library for working with Java source code. javalang provides a lexer and parser targeting Java 8. The implementation is based on the Java language spec available at <http://docs.oracle.com/javase/specs/jls/se8/html/>.

The following gives a very brief introduction to using javalang.

Getting Started

```
>>> import javalang
>>> tree = javalang.parse.parse("package javalang.brewtab.com; class Test {}")
```

This will return a `CompilationUnit` instance. This object is the root of a tree which may be traversed to extract different information about the compilation unit.

```
>>> tree.package.name
u'javalang.brewtab.com'
>>> tree.types[0]
ClassDeclaration
>>> tree.types[0].name
u'Test'
```

The string passed to `javalang.parse.parse()` must represent a complete unit which simply means it should represent a complete, valid Java source file. Other methods in the `javalang.parse` module allow for some smaller code snippets to be parsed without providing an entire compilation unit.

Experiment Detail

Truncation Details

- Truncation: In our experiments, we set the in-file project-specific context length to 30, the cross-file project-specific context length to 30, the local context length to 55 (variable length (50) + parameter and return type length (5)), the documentation context length to 10. And the maximum target name length is set to 5
- In-file project-specific context 30: the name of the methods within the same file
- Cross-file project-specific context 30: the name of the methods within the same project but not in the same file
- Local context 55: variable length (50) + parameter and return type length (5)
- Documentation context 10: first line of code comment

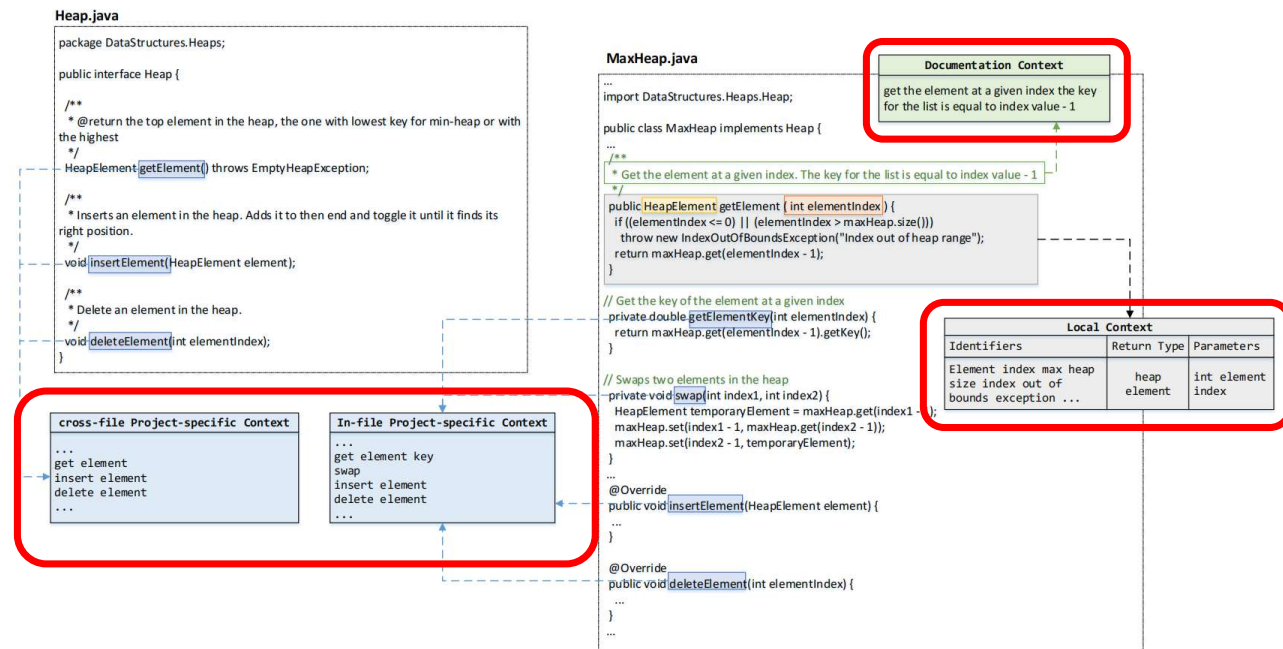


Figure 2: Different levels of contexts for method name suggestion.

Evaluation Metrics

Precision/Recall/F1-Score + Exact Match Accuracy

- To evaluate the quality of the generated method name, we adopted the metrics used by previous works, which measured Precision, Recall, and F-score over sub-tokens. Specifically, for the pair of the target method name t and the predicted name p , the $precision(t, p)$, $recall(t, p)$, and $F1(t, p)$ score are computed as
- Exact Match Accuracy

$$precision(t, p) = \frac{|\text{subtoken}(t) \cap \text{subtoken}(p)|}{|\text{subtoken}(p)|}$$
$$recall(t, p) = \frac{|\text{subtoken}(t) \cap \text{subtoken}(p)|}{|\text{subtoken}(t)|}$$
$$F1(t, p) = \frac{2 \times precision(t, p) \times recall(t, p)}{precision(t, p) + recall(t, p)}$$

Experiment Result

4 Baselines to compare

- **Code2vec 19'**: an **attention-based neural model**, which performs attention mechanism over **AST paths** and aggregates all of the path vector representations into a single vector. They considered the method name prediction as a **classification problem** and predicted a method's name from the vector representation of its body.
- **Code2seq 19'**: an extended approach of code2vec, which employs seq2seq framework to represent **AST paths** of the method body node-by-node using **LSTMs** and then attend to them while generating the target subtokens of the method name.
- **MNire ICSE20'**: an **RNN-based seq2seq model** approach to suggest a method name based on the **program entities' names** in the method body and the enclosing **class name**.
- **DeepName ICSE21'**: an **RNN-based approach** for method name consistency checking and suggestion, using both **internal and interaction contexts** for method name consistency checking and suggestion, which achieves the **state-of-the-art results** on java method name suggestion task.

Table 3: Method name recommendation comparison results.

Model	Precision	Recall	F1	EM Acc
code2vec[7]	51.93%	39.85%	45.10%	35.59%
code2seq[6]	68.41%	60.75%	64.36%	41.50%
MNire[34]	70.10%	64.30%	67.10%	43.10%
DeepName[24]	73.60%	71.90%	72.70%	44.30%
GTNM	77.01%	74.15%	75.60%	62.01%

code2vec and code2seq only use the context in the method body

MNire utilizes the enclosing class, DeepName further considers the interaction context and sibling context, which might appear in other program files.

Experiment Result

F1 or Exact Match, which is a better evaluation metric?

- Table 4 shows two examples where the exact-match didn't occur but F1 was good. In the first case, the semantics of two names are reverse although they shared most of the sub-tokens with a high F1 score. Thus, exact match accuracy can evaluate the generated name more precisely, which plays a crucial role in method name suggestion.

Ablation study on different contexts

- **Token seq vs Local context:** almost equal. But local context is much more shorter than original seq. (computation efficiency)
- **In-file Project context:** offer knowledge about the project information, which is essential and efficient for improving the performance
- **Documentation context:** only 20% of the methods have the documentation context, so the authors did another study on methods with documentation

Table 4: Examples where the exact match did not occur but F1 was good.

Prediction	Ground Truth
'before', 'attach', 'primary', 'storage' 'reset', 'buffer'	'before', 'detach', 'primary', 'storage' 'reset'

Table 5: Performance of using different contexts.

Model	Precision	Recall	F1	EM Acc
Token seq	70.25%	64.75%	67.39%	49.44%
Local cxt	69.60%	64.38%	66.89%	50.95%
+ In-file Project cxt	75.16%	71.83%	73.46%	59.51%
+ Documentation cxt	77.01%	74.15%	75.60%	62.01%

Table 6: The results on the extracted documented methods.

Model	Precision	Recall	F1	EM Acc
GTNM	85.36%	82.54%	83.93%	70.60%
- doc	80.31%	76.65%	78.44%	64.14%

Experiment Result

We already know that document/in-file context are useful, how about the cross-file context?

- When considering the cross-file project-specific context, we need to **preserve the project structure of the programs** in the dataset.
- So, this experiment is more difficult to operate.
- The authors assume that the model can be trained in a low-resource setting, that is, fewer programs are needed for training the model since more contextual information can be accessed. Thus, they **only use a subset of the whole training dataset** in this experiment.
- Specifically, they sample 4000 projects from the big training set as a small training set and extract the cross-file project-specific context for the programs in the sampled projects.
- We compare with the results of our model setting without using project-specific context.
- The cross-project setting is **challenging** and **reflects better the real-world usage of the method name recommendation** where the model is trained on the set of existing projects and used to check for a new project.

Table 7: Performance of using cross-file project-specific context under cross-project and low-resource setting.

Model	Precision	Recall	F1	EM Acc
w/o cross-file cxt	67.25%	64.66%	65.93%	49.71%
w/ cross-file cxt	73.52%	70.65%	72.06%	60.69%

Qualitative Analysis

4-Representative Situations

- We performed qualitative analysis to obtain opinions from participants on the quality of the generated-name, aiming at getting the feedback on our approach and directions for future-work. We invited **8 volunteers with 3-5 years of Java development experience** to evaluate the generated names of the sampled **200 cases** in the form of a **questionnaire**. Each participant is asked to answer several questions, **including whether the human-written-names or generated-names are good, what are the differences between two names**, etc. According to the questionnaire results, we **summarize top-4 representative situations** (The proportion of each situation is 19.4%/43.6%/6.6%/11.9%)

Table 8: Examples of generated summaries given Java methods.

Examples	
Method 1	<pre>/** * Adds a path (but not the leaf folder) if it does not already exist. */ protected void ____ (List<String> path, int depth) { int parentSize = path.size() - 1; String name = path.get(depth); Folder child = getChild(name); if (child == null) { child = new Folder(name); ... } }</pre>
Human-written GTNM	"add" "add", "path", "if", "not", "exists"

Method 2	<pre>/** * Append the longs in the array to the selection , each separated by a comma */ private void ____ (long[] objects) { for (int i = 0; i < objects.length; i++) { selection.append(objects[i]); if (i != objects.length - 1) { selection.append(','); } } }</pre>
Human-written GTNM	"join", "in", "selection" "append", "selection"

Qualitative Analysis

4-Representative Situations

- **1、Contain More Detailed Information 比人写的更详细 19.4%:** GTNM tends to generate a longer name that contains more information about the method's functionality.
- **2、Synonyms 同义词 43.6%:** The human-written name and the name generated by our model have the same meaning, and the verbs used in these two names are synonyms
- **3、Acronym 无法写出缩写 6.6%:** The human-written name contains an acronym for the specific entities, i.e., “du” for “definition use”, which our model cannot correctly infer
- **4、Different Word Orders 词序不对 11.9%:** The order of sub-token are different, but might not change the semantics

Table 8: Examples of generated summaries given Java methods.

Examples	
Method 1	<pre> /** * Adds a path (but not the leaf folder) if it does not already exist. */ protected void ____ (List<String> path, int depth) { int parentSize = path.size() - 1; String name = path.get(depth); Folder child = getChild(name); if (child == null) { child = new Folder(name); ... } } </pre>
Human-written	"add"
GTNM	"add", "path", "if", "not", "exists"
Method 2	<pre> /** * Append the longs in the array to the selection, each separated by a comma */ private void ____ (long[] objects) { for (int i = 0; i < objects.length; i++) { selection.append(objects[i]); if (i != objects.length - 1) { selection.append(','); } } } </pre>
Human-written	"join", "in", "selection"
GTNM	"append", "selection"
Method 3	<pre> /** * Calculates the DefinitionUseCoverage fitness for the given DUPair on the given ExecutionResult */ public double ____ () { if (isSpecialDefinition (goalDefinition)) return calculateUseFitnessForCompleteTrace (); double defFitness = calculateDefFitnessForCompleteTrace (); if (defFitness != 0) return 1 + defFitness; return calculateFitnessForObjects (); } </pre>
Human-written	"calculate", "d", "u", "fitness"
GTNM	"calculate", "fitness", "for"
Method 4	<pre> /** * Validate removal of invalid entries. */ public void ____ () { RightThreadedBinaryTree<Integer> bt = new RightThreadedBinaryTree<Integer> (); assertFalse (bt.remove (99)); bt = buildComplete (4); assertFalse (bt.remove (99)); assertFalse (bt.remove (-2)); } </pre>
Human-written	"test", "invalid", "removals"
GTNM	"test", "remove", "invalid"

Length Analysis

Generating longer method names is challenging but important

- Among all the methods, **only 13.78% of the names generated by our model are shorter than the ground truth(Human).**
- We apply the **Wilcoxon Rank Sum Test (WRST)** to test whether the **increase in the method name length is statistically significant**, and all the p-values are less than $1e-5$, which indicates a significant increase. We also use **Cliff's Delta** to measure the effect size, and the values are **non-negligible**.
- Thus, our model tends to suggest more detailed names for the method. Besides, we also give the exact match accuracy of different lengths. As the length increase, the method naming task becomes harder. Even though our model can still achieve more than 50% accuracy for the names of length 5.

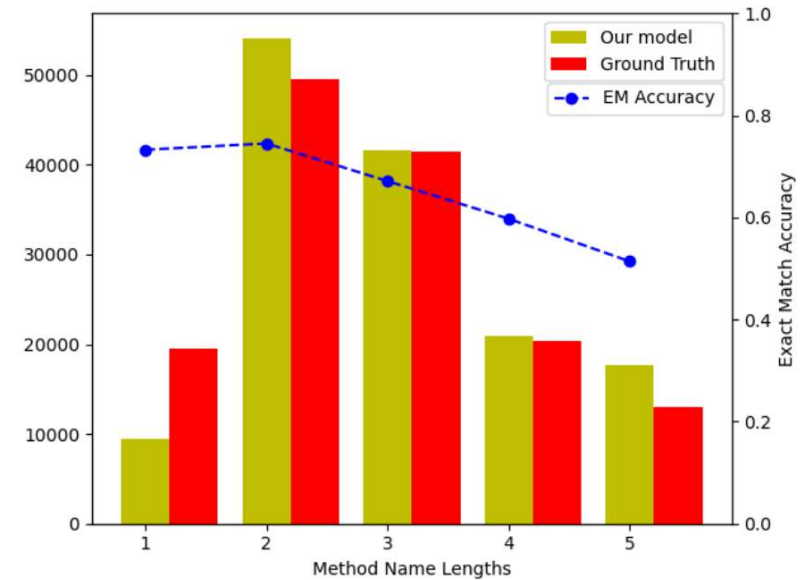


Figure 3: The method name length distribution and the exact match accuracy of different name lengths

Explainability Analysis

Explainability: How confident is the model about its result?

- Lack of explainability is an important concern in many complex AI/ML models in SE. It is crucial to ensure that the model is learned correctly and the logic behind the model is reasonable, which is also important for method name recommendation task. In this section, we analyze the explainability of GTNM.
- We employ **model's confidence** about its prediction to decide whether to accept the model's recommendation. **Prediction Confidence Score (PCS)** which depicts the **probability difference between the two classes with the highest probabilities** is a measure for evaluating model's confidence. In our model, the **Pearson Correlation Score between PCS and F1-score** of the generated names is **0.612** and p-value <0.05 , **demonstrating that the correctness of the generated name is closely related to the model's confidence about its prediction**. Thus, users can decide whether to accept the generated names depending on the case's error tolerance and the model's confidence.
- 我觉得这一段非常精彩，把一个简单的生成结果的可靠性评估讲的非常好，PCS计算概率最高的类别之间的概率差作为模型的置信度，然后用皮尔森相关系数来计算模型对生成结果的置信度和结果的F1打分的相关性。这都是非常非常简单基础的内容但是用在这个可解释性和模型结果可靠性的说明上感觉非常的好。

Reflection

- **A Solid work:** Substantial analysis on the results and brilliant discussion on the model explainability
- **Multi-context:** a “cheating” method when you find that no improvement could be made on model/dataset
- Concrete Examples: takes almost one page
- Fancy definitions: Local/In-file Project-Specific/Cross-file Project-Specific/Documentation Context
- What could we conduct on Transformer?

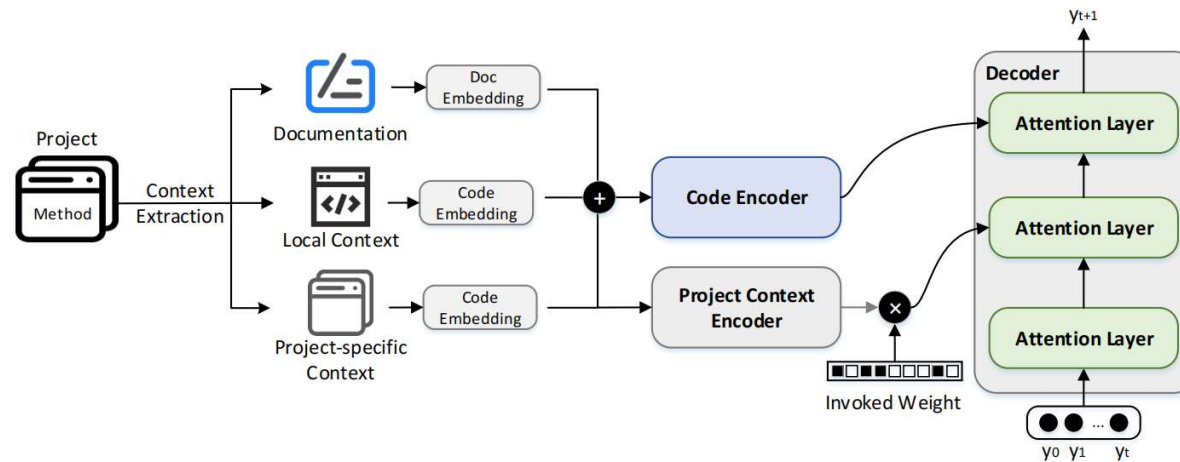


Figure 1: The overall framework of GTNM.

Thanks

Zhu Jie

2022.06.17