AutoTransform: Automated Code Transformation to Support Modern Code Review Process

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

Presenter: Zhu Jie

2022.4.28

Authors

Patanamon Thongtanunam

Lecture (2017 - Now)

University of Melbourne

- *Code Review*
- *Mining Software Repositories*
- *Understanding and Improving Developer collaboration practices*

Chanathip Pornprasit

PhD student of Monash University

- *Natural Language Processing*
- *Software Defects Prediction*

Chakkrit Tantithamthavorn

Senior Research Fellow (2017 - Now) Monash University

Monash's Software Engineering Discipline Group Lead

- *Software Defects Prediction*

Authors

Patanamon Thongtanunam

Lecture (2017 - Now) University of Melbourne

- *Code Review*
- *Mining Software Repositories*
- *Understanding and Improving Developer collaboration practices*

Theses

1. Studying Reviewer Selection and Involvement in Modern Code Review Processes
   Author: Patanamon Thongtanunam
   Degree: Doctor of Engineering
   Venue: Graduate School of Information Science, Nara Institute of Science and Technology
   Publication Year: 2016
2. An Approach to Recommend Reviewers using File Path Similarity for Peer Code Review Process
   Author: Patanamon Thongtanunam
   Degree: Master of Engineering
   Venue: Graduate School of Information Science, Nara Institute of Science and Technology
   Publication Year: 2014

https://patanamon.com/

Patanamon (Pick) Thongtanunam

About | Publications | Recognition | Talks

About me

I'm a lecturer at the School of Computing and Information Systems, the University of Melbourne. My research goals are directed towards uncovering empirical evidence and extracting knowledge from data recorded in software repositories by using statistical analysis. In particular, my research is focused on understanding and improving developer collaboration practices. Nowadays, the variety of collaboration activities which can be found in large software repositories have provided opportunities and challenges for software engineering researchers and practitioners. Therefore, I am keen to perform research that (1) incorporates the various types of collaboration activities, (2) gleans actionable insights for software engineering management, and (3) provides tool support for software developers in order to improve software quality.

Contact: patanamon.t{at}unimelb.edu.au

I'm open to work with enthusiastic Master/Ph.D. students who are keen to learn about mining big data, discovering knowledge, and developing supporting tools for our Software Engineering communities! Feel free contact me if you are interested ;)

Publications

1. Towards Reliable Agile Iterative Planning via Predicting Documentation Changes of Work Items
   Authors: Jirat Pasuksmit, Patanamon Thongtanunam, Shanika Karunasekera
   Venue: The International Conference on Mining Software Repositories (MSR)
   Acceptance Rate: 34% (45/137)
   Preprint: PDF
2. AutoTransform: Towards Automated Code Transformation to Support Modern Code Review Process
   Authors: Patanamon Thongtanunam, Chanathip Pornprasit, Chakkrit Tantithamthavorn
   Venue: The International Conference on Software Engineering (ICSE)
   Acceptance Rate: 26% (200/751)
   Preprint: PDF
3. Where Should I Look at? Recommending Lines that Reviewers Should Pay Attention To
   Authors: Yang Hong, Chakkrit Tantithamthavorn, Patanamon Thongtanunam
   Venue: The International Conference on Software Analysis, Evolution and Reengineering (SANER)
   Acceptance Rate: 36.2% (72/199)
   Preprint: PDF
4. PyExplainer: Explaining the Predictions of Just-In-Time Defect Models
   ⭐ **ACM SIGSOFT Distinguished Paper Award** ⭐
   Authors: Chanathip Pornprasit, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Micheal Fu, Patanamon Thongtanunam
   Venue: The International Conference on Automated Software Engineering
   Acceptance Rate: 19% (82/427)
   Preprint: PDF

What's news

Recognition

x 9 Grants

x 12 Awards

Publications

ASE x 1 | SIGSOFT-SEN x 1 | TSE x 2 | PROMISE x 1 | EMSE x 4 | ICSE x 4 | MSR x 3 | SANER x 3 | ICSME x 2 | IWESEP x 1 | CHASE x 1 | RSS x 1 | KBSE x 1

Authors

Chanathip Pornprasit

PhD student of Monash University

- *Natural Language Processing*
- *Software Defects Prediction*

AutoTransform: Automated Code Transformation to Support Modern Code Review Process

Patanamon Thongtanunam*
patanamon.t@unimelb.edu.au
The University of Melbourne
Australia

Chanathip Pornprasit†
chanathip.pornprasit@monash.edu
Monash University
Australia

Chakkrit Tantithamthavorn
chakkrit@monash.edu
Monash University
Australia

PyExplainer: Explaining the Predictions of Just-In-Time Defect Models
Chanathip Pornprasit; Chakkrit Tantithamthavorn; Jirayus Jiarpakdee; Michael Fu; Patanamon Thongtanunam
2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)
Year: 2021 | Conference Paper | Publisher: IEEE
Cited by: Papers (1)
Abstract HTML

JITLine: A Simpler, Better, Faster, Finer-grained Just-In-Time Defect Prediction
Chanathip Pornprasit; Chakkrit Kla Tantithamthavorn
2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)
Year: 2021 | Conference Paper | Publisher: IEEE
Cited by: Papers (4)
Abstract HTML

Enhancing CNN Based Knowledge Graph Embedding Algorithms Using Auxiliary Vectors: A Case Study of Wordnet Knowledge Graph
Chanathip Pornprasit; Pattararat Kiattipadungkul; Peeranut Duangkaew; Suppawong Tuarob; Thanapon Noraset
2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)
Year: 2020 | Conference Paper | Publisher: IEEE
Abstract HTML

Automatic Classification of Algorithm Citation Functions in Scientific Literature
Suppawong Tuarob; Sung Woo Kang; Poom Wettayakorn; Chanatip Pornprasit; Tanakitti Sachati; Saeed-Ul Hassan; Peter Haddawy
IEEE Transactions on Knowledge and Data Engineering
Year: 2020 | Volume: 32, Issue: 10 | Journal Article | Publisher: IEEE
Cited by: Papers (10)

https://ieeexplore.ieee.org/author/37088457315

Authors

Chakkrit Tantithamthavorn

Senior Research Fellow (2017 - Now) Monash University

Monash's Software Engineering Discipline Group Lead

- *Software Defect Prediction*

Education/Academic qualification

Software Engineering, Doctor of Engineering, Nara Institute of Science and Technology
Award Date: 26 Sep 2016

Software Engineering, Master of Engineering, Nara Institute of Science and Technology
Award Date: 31 Mar 2014

2022

An empirical study of model-agnostics techniques for defect prediction models
Jiarpakdee, J., Tantithamthavorn, C., Dam, H. K. & Grundy, J., 1 Jan 2022, In: IEEE Transactions on Software Engineering. 48, 1, p. 166-185 21 p.
Research output: Contribution to journal › Article › Research › peer-review

DeepLineDP: towards a deep learning approach for line-level defect prediction
Pornprasit, C. & Tantithamthavorn, C., 21 Jan 2022, (Accepted/In press) In: IEEE Transactions on Software Engineering. 16 p.
Research output: Contribution to journal › Article › Research › peer-review

GPT2SP: a transformer-based Agile Story Point Estimation approach
Fu, M. & Tantithamthavorn, C., 10 Mar 2022, (Accepted/In press) In: IEEE Transactions on Software Engineering. 16 p.
Research output: Contribution to journal › Article › Research › peer-review

Search-based fairness testing for regression-based machine learning systems
Perera, A., Aleti, A., Tantithamthavorn, C., Jiarpakdee, J., Turhan, B., Kuhn, L. & Walker, K., May 2022, In: Empirical Software Engineering. 27, 3, 36 p., 79.
Research output: Contribution to journal › Article › Research › peer-review

Open Access File

2021

Actionable analytics: stop telling me what it is; please tell me what to do
Tantithamthavorn, C., Jiarpakdee, J. & Grundy, J., Jul 2021, In: IEEE Software. 38, 4, p. 115-120 6 p.
Research output: Contribution to journal › Article › Research › peer-review

Assessing the students' understanding and their mistakes in code review checklists: an experience report of 1,791 code review checklist questions from 394 Students
Chong, C. Y., Thongtanunam, P. & Tantithamthavorn, C., May 2021, *Proceedings - 2021 IEEE/ACM 43rd International Conference on Software Engineering: Joint Track on Software Engineering Education and Training, ICSE-JSEET 2021.* Erdogmus, H. & M. Moreno, A. (eds.). Piscataway NJ USA: IEEE, Institute of Electrical and Electronics Engineers, p. 20-29 10 p. (Proceedings - International Conference on Software Engineering).
Research output: Chapter in Book/Report/Conference proceeding › Conference Paper › Research › peer-review

Why Choose This Paper?

- New: ICSE'22 Paper
- Well-known Method: Transformer-based model + BPE(byte pair encoding)
- Related Area: a Code Review paper
- Open: Replication Package available on GitHub
- Compare with a Top-research-conference paper: a shortcut to publish?

Background

Code Comment/Summarization/Review

What is the difference among code comment, summarization and review comment?

- Code Comment: Explain how your program works, and your intentions behind it
- Code Summarization: Generate a readable summary that describes the functionality of a program
  - ➢ Code summarization focuses more on the logic and functionality of code
  - ➢ Code comment is more flexible
- **Code Review Comment: Point out the problem (Focus on the changed part)**

```
    public class DockerOptions {
      for (int i = 0; i < maxContainerCount; i++) {
        node.add(caps, new DockerSessionFactory(clientFactory, docker, image, caps));
      }
-     LOG.info(String.format(
+     LOG.finest(String.format(
          "Mapping %s to docker image %s %d times", caps, name,
```

Ground truth: This change prevents a user understanding how their server is configured. Best to leave at `info` level.	
T5: I don't think this change is needed.	BLEU:4.93
CodeT5: Why do we need this change?	BLEU:2.97
Codex: I think this is a good example of how to use the docker	BLEU:8.83
CodeReviewer: This is an important debugging help and shouldn't be lower than the default visible INFO.	BLEU:7.97

(a) An example of the review generation task. The Codex output is obtained by Copilot.

Code (Java)	<pre>private void attachPlot (SVGPlot newplot) {
	    this.plot = newplot;
	    if (newplot == null) {
	        super.setSVGDocument(null);
	        return;
	    }
	    newplot.synchronizeWith(synchronizer);
	    super.setSVGDocument(
	        newplot.getDocument());
	    super.setDisableInteractions(
	        newplot.getDisableInteractions());
	}</pre>

Summ.	Attach to a new plot and display.

Code (Python)	<pre>def get_change_lines_in_file_for_tag(tag,
	                                     change_dict):
	    cleaned_lines = []
	    data_list = change_dict.get('data', [])
	    for data_dict in data_list:
	        block = data_dict.get('block', '')
	        lines = block.split('\\n')
	            for line in lines:
	                index = line.find(tag)
	                if (index >(-1)):
	                    line = line[index:]
	                    cleaned_lines.append(line)
	    return cleaned_lines</pre>

Summ.	The received change_dict is the jsonified version of the changes to a file in a changeset being pushed to the Tool Shed from the command line. This method cleans and returns appropriate lines for inspection.

Table 1: Task samples of code summarization, where summ. refers to the output summary.

[1] 北京智源研究院 青源LIVE第43期 | MSRA卢帅：自动化代码审查过程的研究 https://www.bilibili.com/video/BV1c34y147AQ

Background

Overview of Code Review Process: Three Automation Tasks

- Task1: Quality Estimation (binary classification: code pair => accept or reject)
- Task2: Review Comment Generation (code pair => review text)
- Task3: Code Refinement (code pair + review text => refined code)

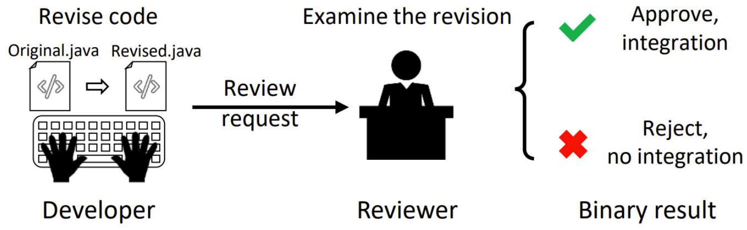


Figure 1: The process of code review.

Automatic Code Review by Learning the Revision of Source Code (AAAI19)

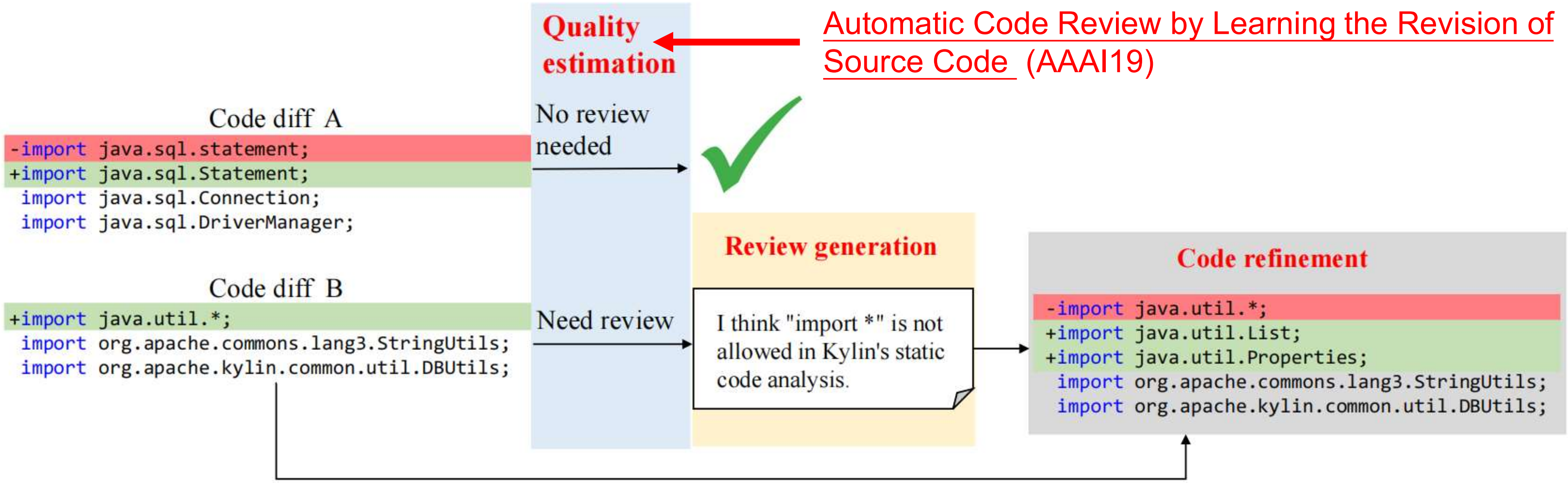


Figure 2: Overview of code review automation tasks.

[1] 北京智源研究院 青源LIVE第43期 | MSRA卢帅：自动化代码审查过程的研究 https://www.bilibili.com/video/BV1c34y147AQ

Background

Overview of Code Review Process: Three Automation Tasks

- Task1: Quality Estimation (binary classification: code pair => accept or reject)
- Task2: Review Comment Generation (code pair => review text)
- Task3: Code Refinement (code pair + review text => refined code)

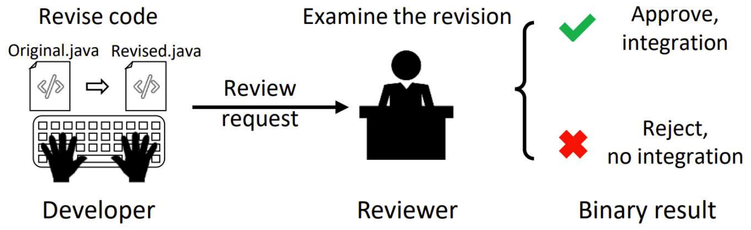


Figure 1: The process of code review.

Where Should I Look at? Recommending Lines that Reviewers Should Pay Attention To (SANER22)

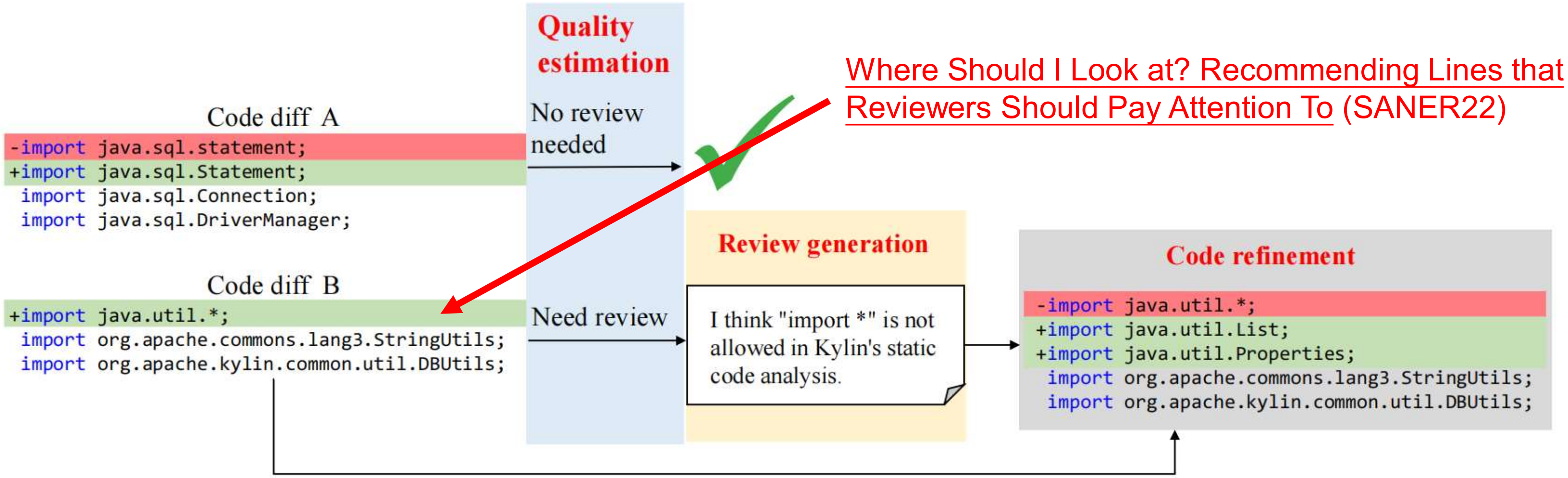


Figure 2: Ove… task…

Defect Prediction
Code Smell

Bug Fixes
Program Repair
Refactoring

[1] SANER22. Where Should I Look at? Recommending Lines that Reviewers Should Pay Attention To

Related Work: Task1

Automatic Code Review by Learning the Revision of Source Code

- Paper Information: AAAI'19 (from NJU lamda + David Lo)
- Motivation: learning the revision features
- Task: Binary classification (approve or reject)
- Technique: CNN + BiLSTM + Auto Encoder

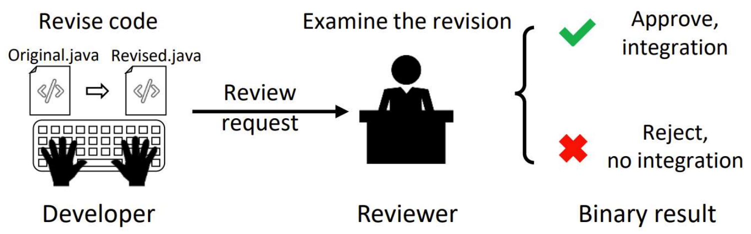


Figure 1: The process of code review.

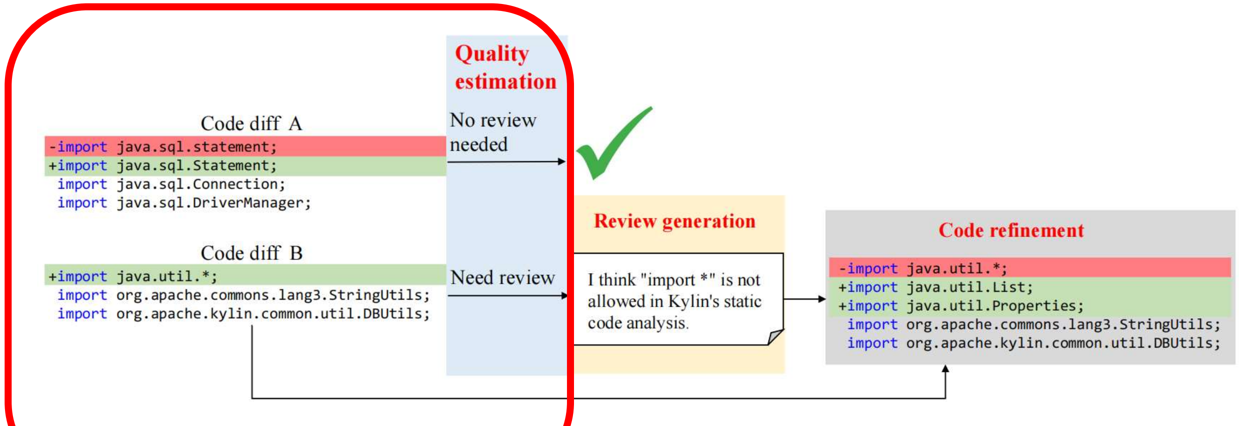


Figure 2: Overview of code review automation tasks.

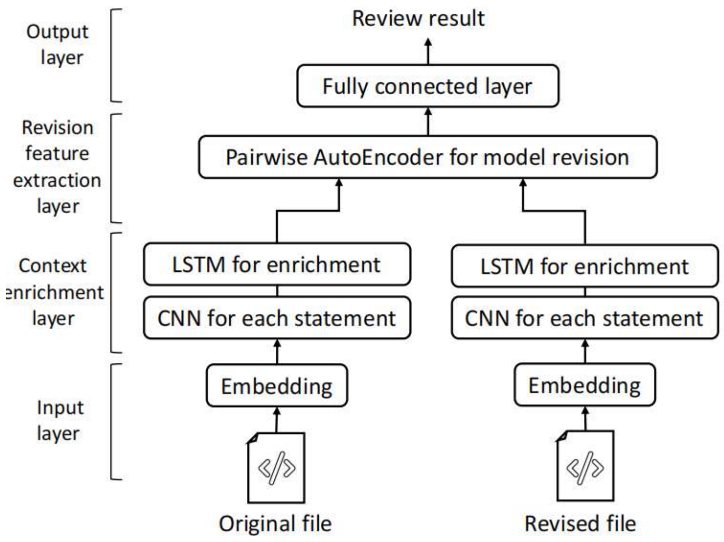


Figure 3: The general framework of DACE.

[1] AAAI19. Automatic Code Review by Learning the Revision of Source Code (NJU lamda + David Lo)

Related Work: Task2

AUGER: Automatically Generating Review Comments with Pre-training Models

- Paper Information: By Li Lingwei
- Task: Review comment generation (text generation)
- Technique: T5-based Pretraining Model

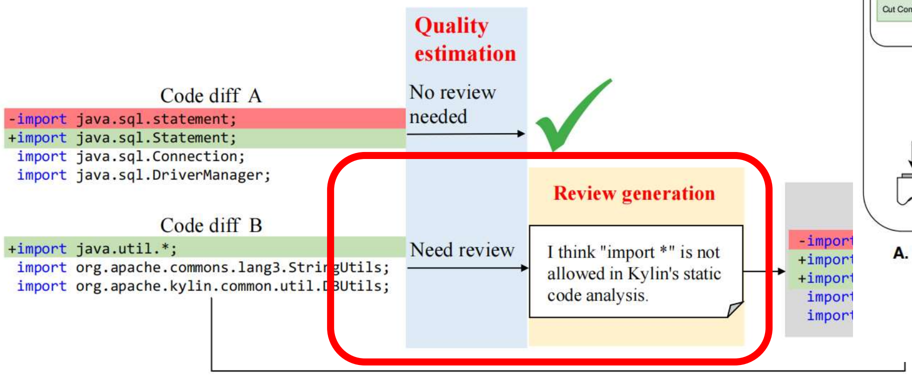


Figure 2: Overview of code review automation tasks.

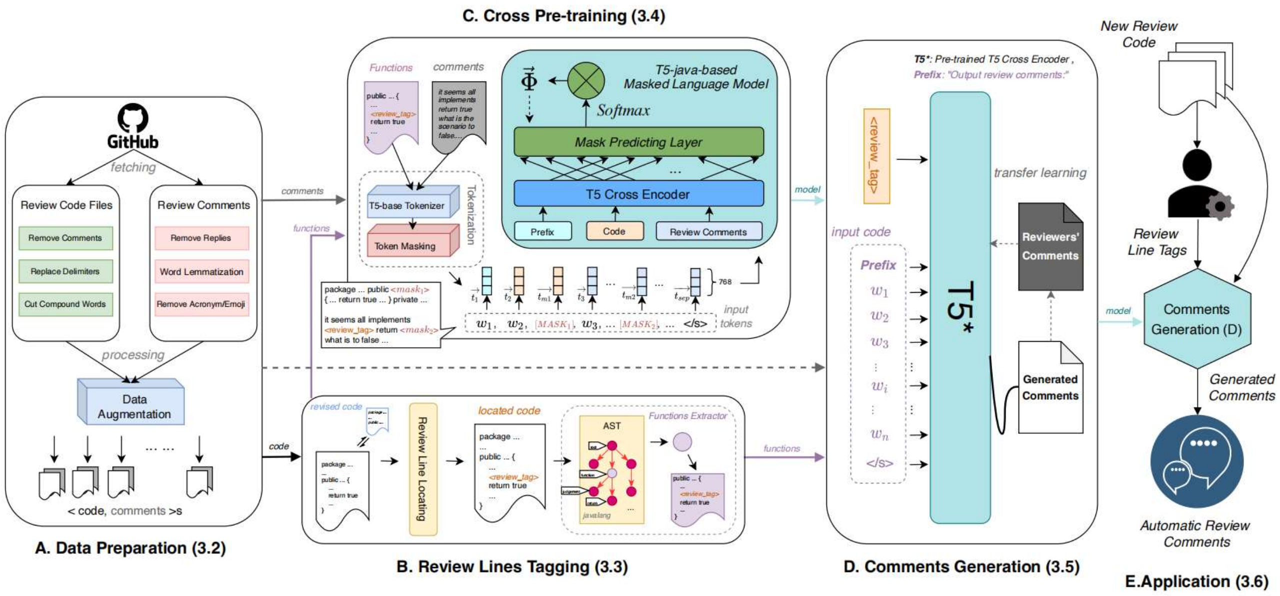


Figure 2: Overview of AUGER

Related Work: Task3

On Learning Meaningful Code Changes via Neural Machine Translation

- **Paper Information**: ICSE'19 from Tufano
- **Related Task:** Bug-Fixes / Code Edit
- **Method**: LSTM-based NMT + **Code abstraction**
- **Expression**:
  - ➢ From **quantitative** analysis to **qualitative** analysis (empirical study of the ability of an NMT model)
  - ➢ **Meaningful** Code Changes (Reviewed and Merged PRs)

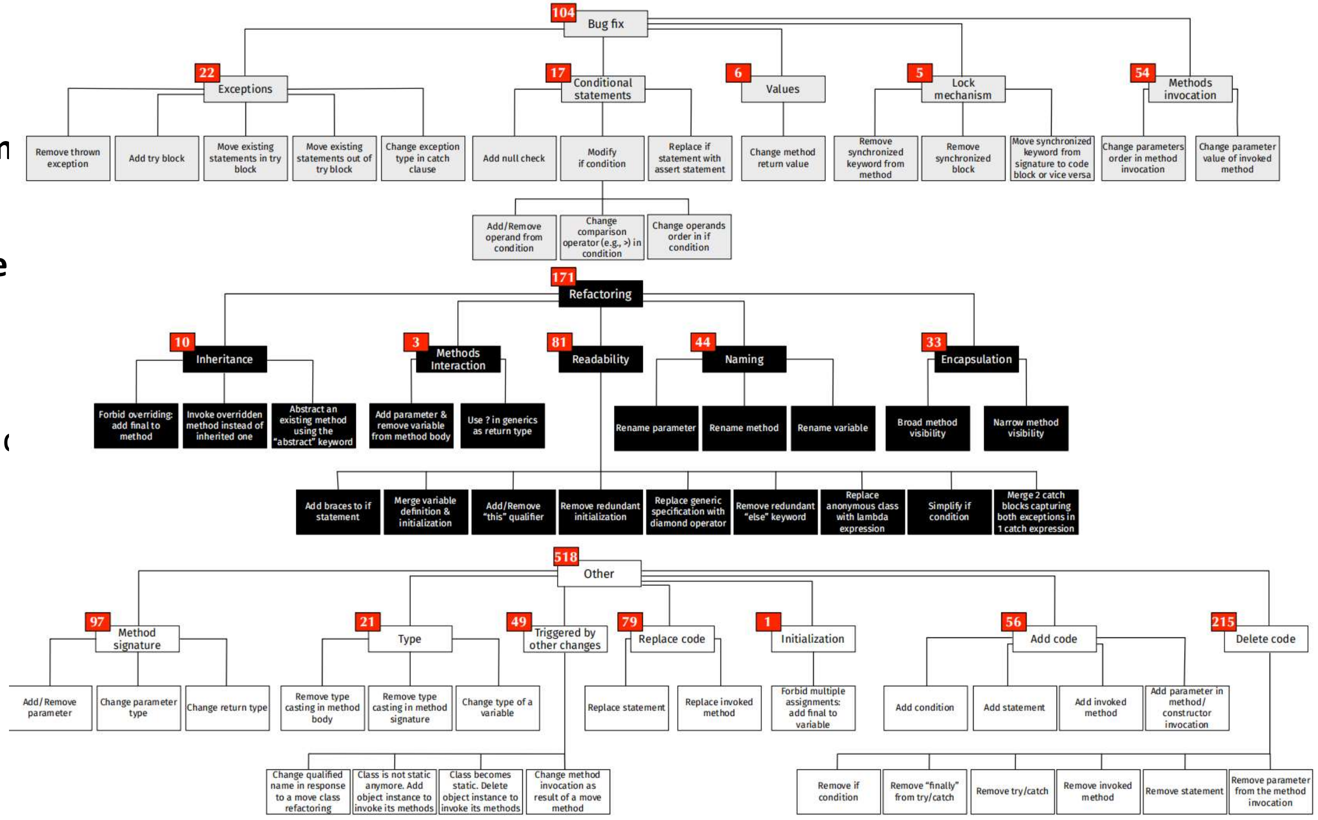


Fig. 1. Taxonomy of code transformations learned by the NMT model

[1] EMNLP19. Encode, Tag, Realize: High-Precision Text Editing (from Google Research)

Related Work: Task3

Towards Automating Code Review Activities

- **Paper Information**: ICSE’21 from Tufano
- **Target Task**: Code Review Comment Generation + Code Refinement

Towards Automating Code Review Activities

Rosalia Tufano*, Luca Pascarella*, Michele Tufano†, Denys Poshyvanyk‡, Gabriele Bavota*

*SEART @ Software Institute, Università della Svizzera italiana (USI), Switzerland

†Microsoft, USA

‡SEMERU @ Computer Science Department, William and Mary, USA

Rosalia Tufano

Università della Svizzera italiana

- AI
- NLP

Luca Pascarella

Università della Svizzera italiana

- Software Engineering

Michele Tufano

Microsoft

- Deep Learning
- Software Engineering
- Software Evolution and maintenance
- Mining Software Repositories
- Software Testing

Denys Poshyvanyk

William and Mary

- Software Engineering
- Software Evolution and Maintenance
- Program Comprehension

Gabriele Bavota

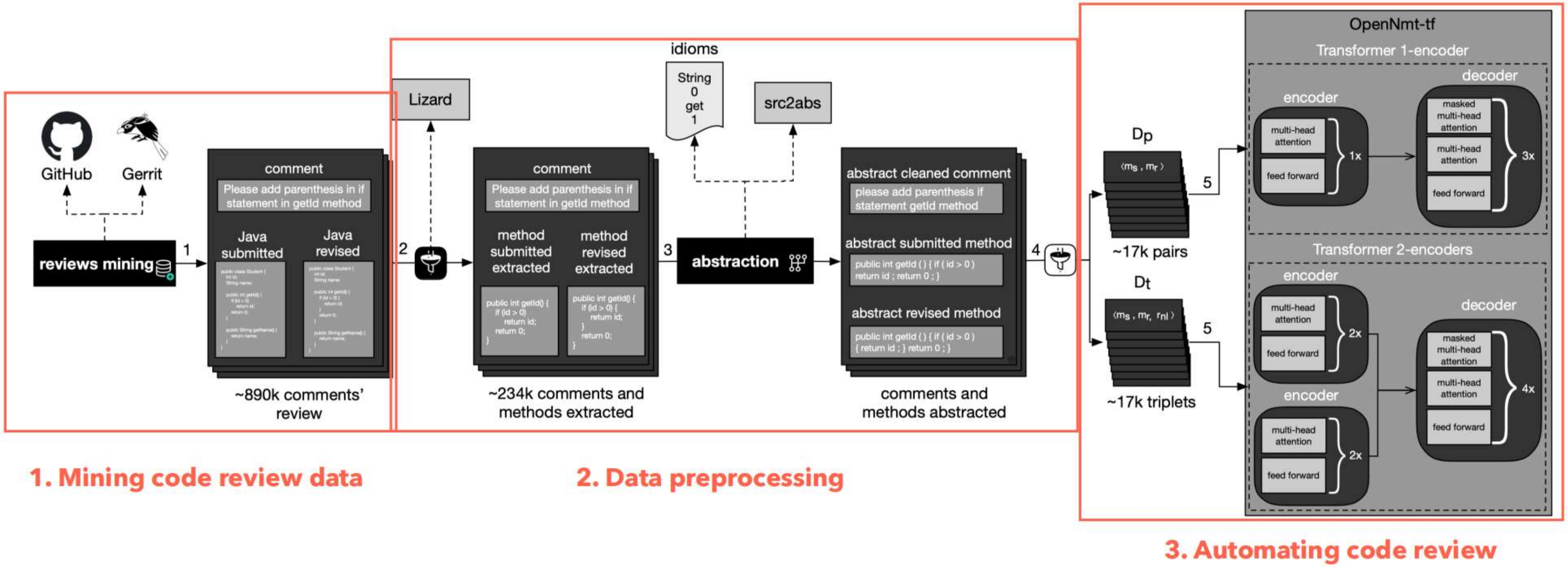
Università della Svizzera italiana

- Software Evolution and Maintenance
- Mining Software Repositories
- Empirical Software Engineering

Related Work: Task3

Towards Automating Code Review Activities

- **Paper Information**: ICSE’21 from Tufano
- **Target Task**: Code Refinement



[1] EMNLP19. Encode, Tag, Realize: High-Precision Text Editing (from Google Research)

Motivation: Limitation of RNN-based Method

On Learning Meaningful Code Changes via Neural Machine Translation (ICSE'19 Tufano)

- **Limitation 1**: Unknown identifiers/literals for the **new tokens** appearing in the after version.
  - ➢ New tokens: tokens that did not appear in the before version. (80% methods contain "New tokens")
  - ➢ Due to the limitation of **code abstraction**.
- **Limitation 2**: Suboptimal performance when the sequences become longer.
  - ➢ **RNN** has difficulties in remembering **long-term dependencies**.

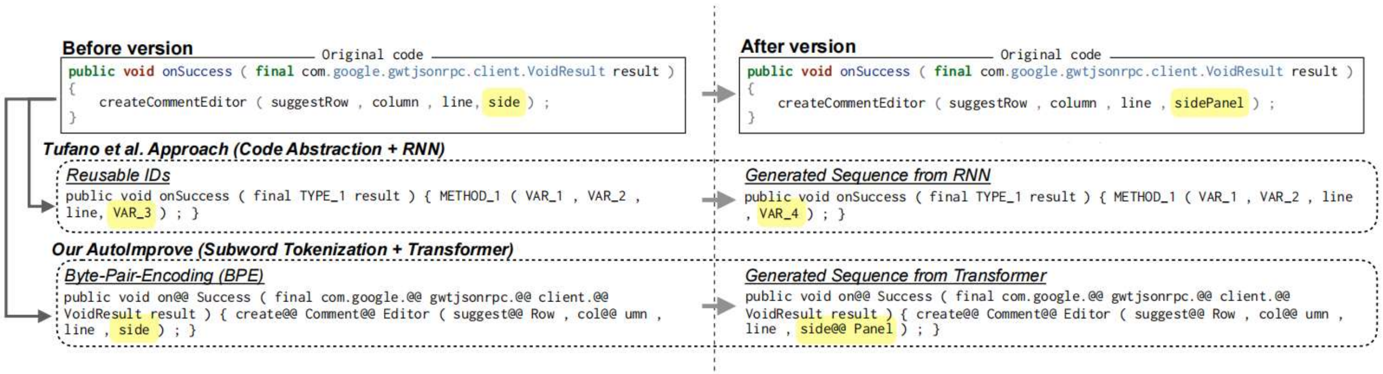


Figure 1: A motivating example for an unknown identifier/literal for the newly-introduced abstracted token (Limitation 1).

Dataset Detail

eval.code_after.txt | eval.code_before.txt ×

E: > AutoTransform > scripts > dataset > with_new_tokens > google > medium > eval.code_before.txt

```
1 public com.google.gwtjsonrpc.client.VoidResult run ( final com.google.gerrit.reviewdb.ReviewDb db ) throws com.google.gerrit.httpd.rpc.
  account.Failure
  get ( groupId )
  singleton ( grou
2 public void onSu
  google.gerrit.cl
  client.account.A
  AccountSuggestOr
3 public void test
  ) ; assertGone (
  commit = repo.br
  . setHostName (
  ) . setRepositor
  "a=commit;p=test
4 public com.googl
  google.gerrit.se
5 private void loa
  CallbackGroup gr
  ) , rev.name ( )
  > ( ) { @ java.l
  info ) ; } @ jav
```

eval.code_after.txt × | eval.code_before.txt

E: > AutoTransform > scripts > dataset > with_new_tokens > google > medium > eval.code_after.txt

```
1 public com.google.gwtjsonrpc.client.VoidResult run ( final com.google.gerrit.reviewdb.ReviewDb db ) throws com.google.gerrit.httpd.rpc.
  account.Failure , com.google.gwtorm.client.OrmException { final com.google.gerrit.reviewdb.AccountGroup group = db.accountGroups ( ) .
  get ( groupId ) ; assertAmGroupOwner ( db , group ) ; group.setExternalNameKey ( bindTo ) ; db.accountGroups ( ) . update ( java.util.
  Collections.singleton ( group ) ) ; groupCache.evict ( group ) ; return com.google.gwtjsonrpc.client.VoidResult.INSTANCE ; }
2 public void onSuccess ( com.google.gwt.core.client.JsArray < com.google.gerrit.client.info.AccountInfo > in ) { java.util.List < com.
  google.gerrit.client.ui.AccountSuggestOracle.AccountSuggestion > r = new java.util.ArrayList ( in.length ( ) ) ; for ( com.google.gerrit.
  client.info.AccountInfo p : com.google.gerrit.client.rpc.Natives.asList ( in ) ) { r.add ( new com.google.gerrit.client.ui.
  AccountSuggestOracle.AccountSuggestion ( p ) ) ; } cb.onSuggestionsReady ( req , new com.google.gerrit.client.ui.Response ( r ) ) ; }
3 public void show ( ) throws java.lang.Exception { assertGone ( com.google.gitiles.GitwebRedirectFilterTest.newRequest ( "a=commit" ) ) ;
  assertGone ( com.google.gitiles.GitwebRedirectFilterTest.newRequest ( "a=commit;p=test" ) ) ; org.eclipse.jgit.revwalk.RevCommit commit
  = repo.branch ( "refs/heads/master" ) . commit ( ) . create ( ) ; assertRedirectsTo ( com.google.gitiles.GitilesView.revision ( ) .
  setHostName ( com.google.gitiles.TestGitilesUrls.HOST_NAME ) . setServletPath ( com.google.gitiles.FakeHttpServletRequest.SERVLET_PATH )
  . setRepositoryName ( "test" ) . setRevision ( commit ) . toUrl ( ) , com.google.gitiles.GitwebRedirectFilterTest.newRequest ( (
  "a=commit;p=test&h=" + ( org.eclipse.jgit.lib.ObjectId.toString ( commit ) ) ) ) ) ; }
4 public com.google.gerrit.extensions.common.RevisionInfo addRevisionActions ( @ com.google.gerrit.common.Nullable com.google.gerrit.
  extensions.common.ChangeInfo changeInfo , com.google.gerrit.extensions.common.RevisionInfo to , com.google.gerrit.server.change.
  RevisionResource rsrc ) throws com.google.gwtorm.server.OrmException { java.util.List < com.google.gerrit.extensions.api.changes.
  ActionVisitor > visitors = visitors ( ) ; if ( ! ( visitors.isEmpty ( ) ) ) { if ( changeInfo != null ) { changeInfo = copy ( visitors ,
  changeInfo ) ; } else { changeInfo = changeJson ( ) . format ( rsrc ) ; } } to.actions = toActionMap ( rsrc , visitors , changeInfo ,
  copy ( visitors , to ) ) ; return to ; }
5 private void loadCommit ( final com.google.gerrit.client.info.ChangeInfo.RevisionInfo rev , com.google.gerrit.client.rpc.CallbackGroup
  group ) { if ( rev.isEdit ( ) ) { return ; } com.google.gerrit.client.changes.ChangeApi.commitWithLinks ( changeId.get ( ) , rev.name (
  ) , group.add ( new com.google.gwt.user.client.rpc.AsyncCallback < com.google.gerrit.client.info.ChangeInfo.CommitInfo > ( ) { @ java.
  lang.Override public void onSuccess ( com.google.gerrit.client.info.ChangeInfo.CommitInfo info ) { rev.setCommit ( info ) ; } @ java.
  lang.Override public void onFailure ( java.lang.Throwable caught ) { } } ) ) ; }
```

Dataset Detail

```
public com.google.gwtjsonrpc.client.VoidResult run (final com.google.gerrit.reviewdb.ReviewDb db) throws
com.google.gerrit.httpd.rpc.account.Failure, com.google.gwtorm.client.OrmException {
    final com.google.gerrit.reviewdb.AccountGroup group = db.accountGroups().get(groupId);
    assertAmGroupOwner (db, group );
    group.setType ( newType ) ;
    db.accountGroups ( ) . update ( java.util.Collections.singleton ( group ) ) ;
    groupCache.evict ( group ) ;
    return com.google.gwtjsonrpc.client.VoidResult.INSTANCE ;
}
```

Long-term Dependency

Code Refinement

```
public com.google.gwtjsonrpc.client.VoidResult run ( final com.google.gerrit.reviewdb.ReviewDb db) throws
com.google.gerrit.httpd.rpc.account.Failure, com.google.gwtorm.client.OrmException {
    final com.google.gerrit.reviewdb.AccountGroup group = db.accountGroups().get(groupId) ;
    assertAmGroupOwner ( db , group ) ;
    group.setExternalNameKey ( bindTo ) ;
    db.accountGroups ( ) . update ( java.util.Collections.singleton ( group ) ) ;
    groupCache.evict ( group ) ;
    return com.google.gwtjsonrpc.client.VoidResult.INSTANCE ;
}
```

New Tokens

Overview

AutoTransform: Automated Code Transformation to Support Modern Code Review Process

- **Contribution:** RNN(LSTM) => Transformer, Code Abstraction => BPE(byte pair encoding)
- **Task:** Buggy Code => Refined Code
- **Experiment:** Ablation study to quantify the contributions of the two components (BPE and Transformer)

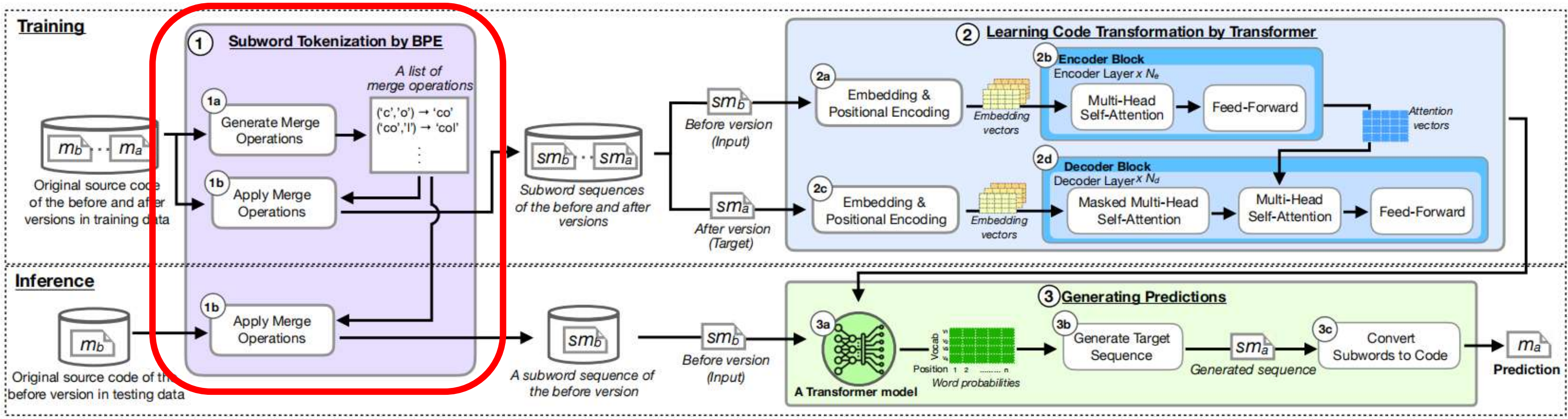


Figure 2: An overview of our AutoTransform.

Overview

AutoTransform: Automated Code Transformation to Support Modern Code Review Process

- **Contribution:** RNN(LSTM) => Transformer, Code Abstraction => BPE(byte pair encoding)
- **Target Task:** Buggy Code => Refined Code
- **Experiment:** Ablation study to quantify the contributions of the two components (BPE and Transformer)

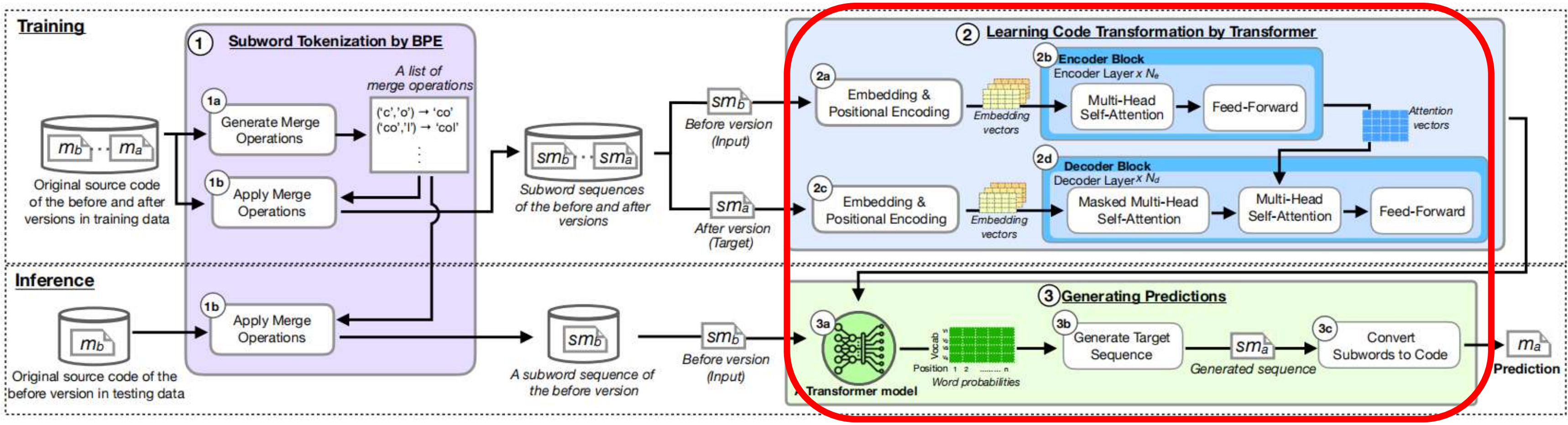
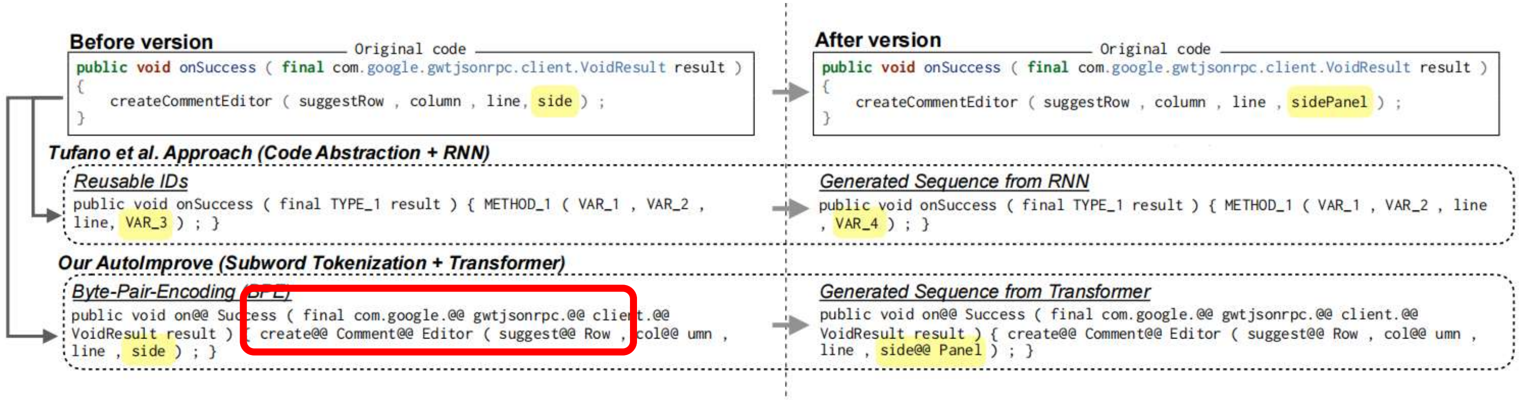


Figure 2: An overview of our AutoTransform.

BPE: Byte Pair Encoding

子词切分（subword）

- 子词
  - 顾名思义，子词
  - 缓解以词语为单
    符为单元的低效
    为“先进”、“



- 子词切分的代表
  - 字节对编码，Byte Pair Encoding (BPE)
  - 句子片段算法，sentencepiece

Figure 1: A motivating example for an unknown identifier/literal for the newly-introduced abstracted token (Limitation 1).

[1] ACL’16. Neural Machine Translation of Rare Words with Subword Units

BPE: Byte Pair Encoding

字节对编码

{'low': 5, 'lower ': 2, 'newest ': 6, 'widest': 3}

转换为字符序列

{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}

Iter1: 合并频率最高的相邻字节对

{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w es t </w>': 6, 'w i d es t </w>': 3}

Iter2: 合并频率最高的相邻字节对

{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w est </w>': 6, 'w i d est </w>': 3}

Iter3: 合并频率最高的相邻字节对

{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3}

继续迭代

……

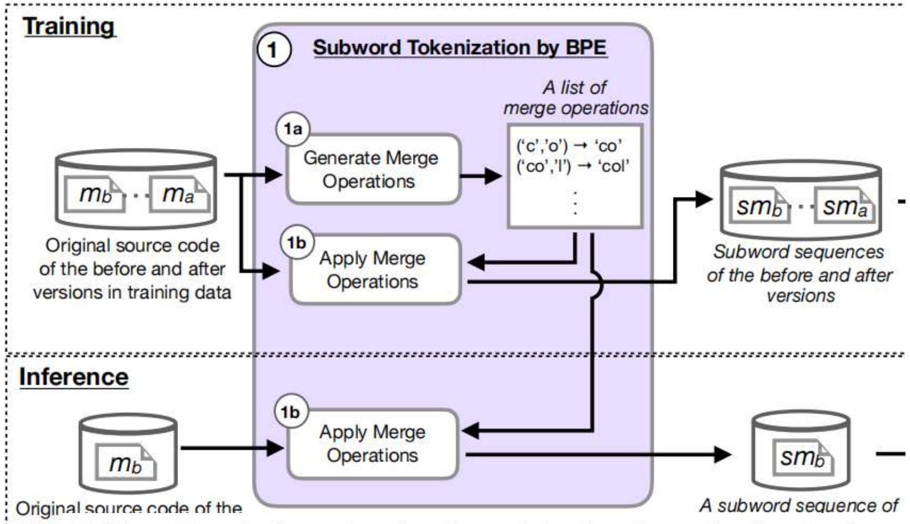


Table 2: Vocabulary size for the original, subword tokenized, and abstracted methods in the training datasets.

Dataset (Method size)	Change Type	Original	Subword Tokenized BPE2K	BPE5K	Abs
Android (Small)	w/o new tokens	12,052	2,702	4,230	356
	w/ new tokens	43,795	7,448	9,247	408
Google (Small)	w/o new tokens	5,012	1,719	2,751	333
	w/ new tokens	13,737	3,417	4,884	383
Ovirt (Small)	w/o new tokens	9,772	1,992	3,575	306
	w/o new tokens	30,562	4,243	6,042	355
Android (Medium)	w/o new tokens	22,296	5,165	6,860	447
	w/ new tokens	76,264	15,585	17,874	496
Google (Medium)	w/o new tokens	9,340	2,831	4,046	371
	w/ new tokens	22,140	6,334	8,052	422
Ovirt (Medium)	w/o new tokens	17,680	3,231	4,958	353
	w/o new tokens	44,317	7,528	9,674	422

*The w/ and w/o new tokens change types are mutually exclusive sets.

[1] ACL'16. Neural Machine Translation of Rare Words with Subword Units

Code Abstraction

- Extract the methods from all the Java files
- Represent each method as a stream of tokens:
  - Java keywords and punctuation symbols are preserved
  - The role of each identifier as well as the type of a literal is discerned
  - Idioms are not abstracted
  - Comments are removed

raw source code

```
public PageProperties getProperties() {
    if (hasProperties()) {
        return properties;
    } else {
        return null;
    }
}
```

abstracted code

```
public TYPE_1 METHOD_1 ( ) { if ( METHOD_2 ( ) )
{ return properties ; } else { return null ; } }
```

[1] ICSE'21. Towards Automating Code Review Activities (from Tufano)

Experiment: Ablation Study

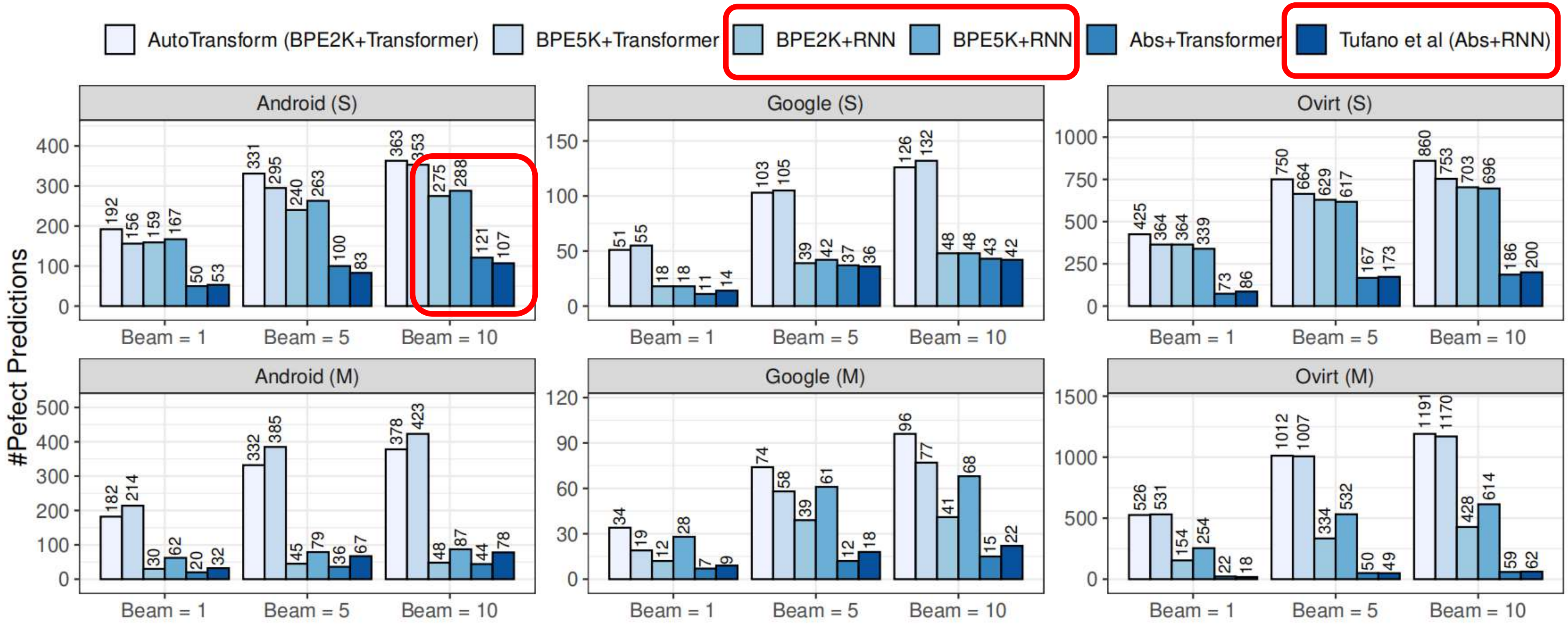


Figure 3: The perfect prediction of our AutoTransform when a component is varied. The y-axis shows the total number of perfect predictions of changed methods *with* and *without* new tokens.

Overview

AutoTransform: Automated Code Transformation to Support Modern Code Review Process

- **Contribution:** RNN(LSTM) => Transformer, Code Abstraction => BPE(byte pair encoding)
- **Task:** Buggy Code => Refined Code
- **Experiment:** Ablation study to quantify the contributions of the two components (BPE and Transformer)

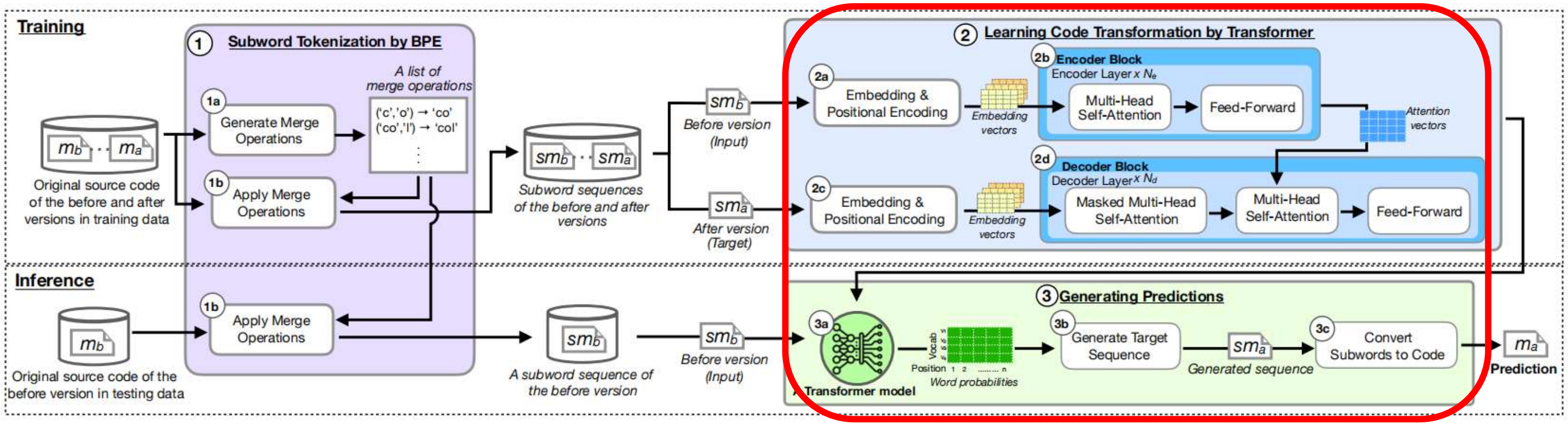


Figure 2: An overview of our AutoTransform.

Transformer

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

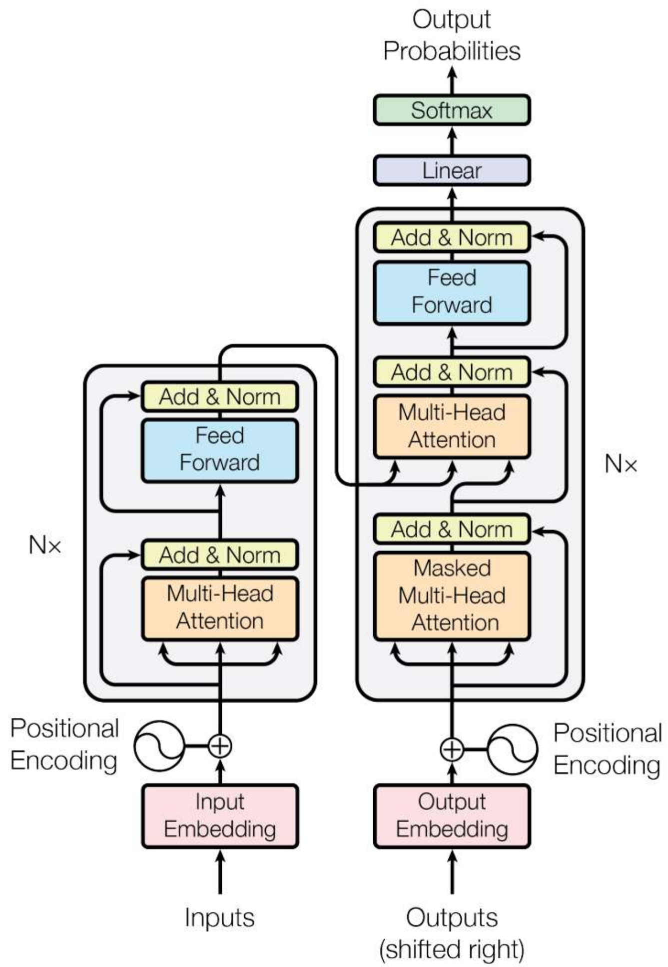
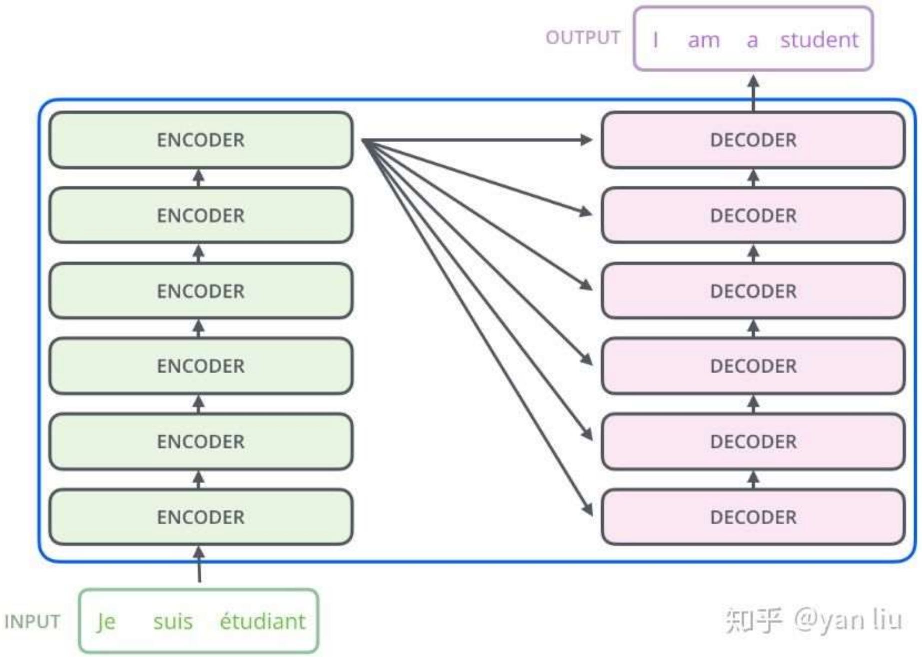


Figure 1: The Transformer - model architecture.

[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)



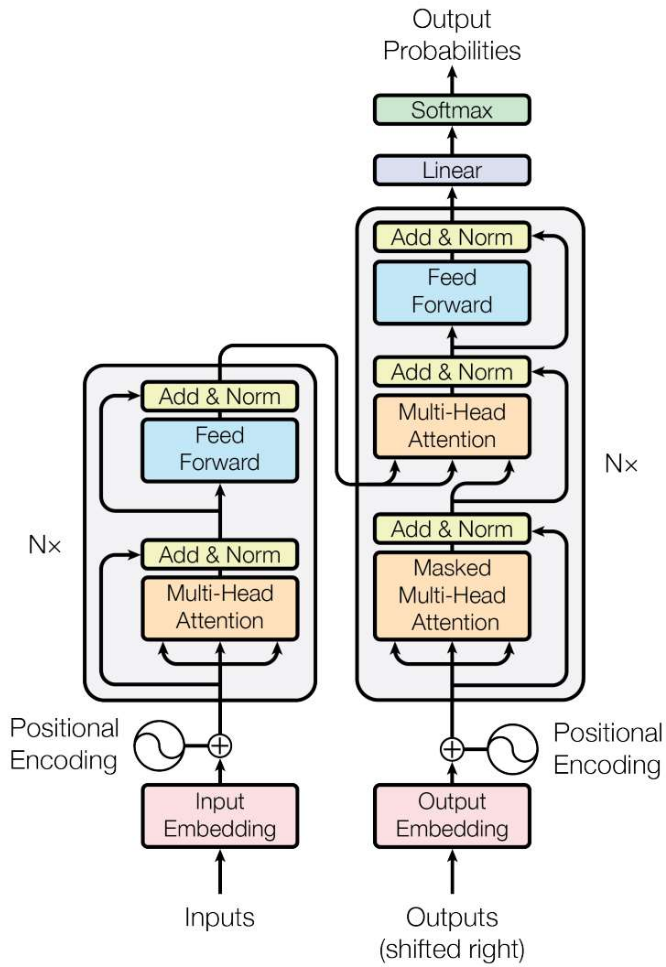
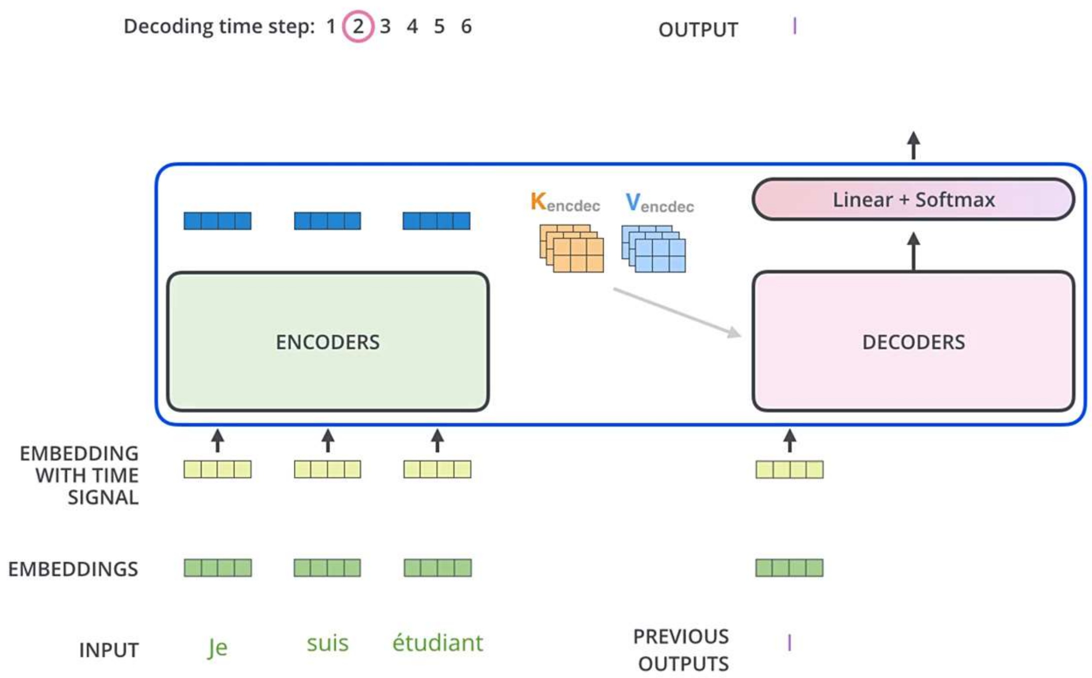


Figure 1: The Transformer - model architecture.

Transformer

- Seq2Seq + Self-Attention



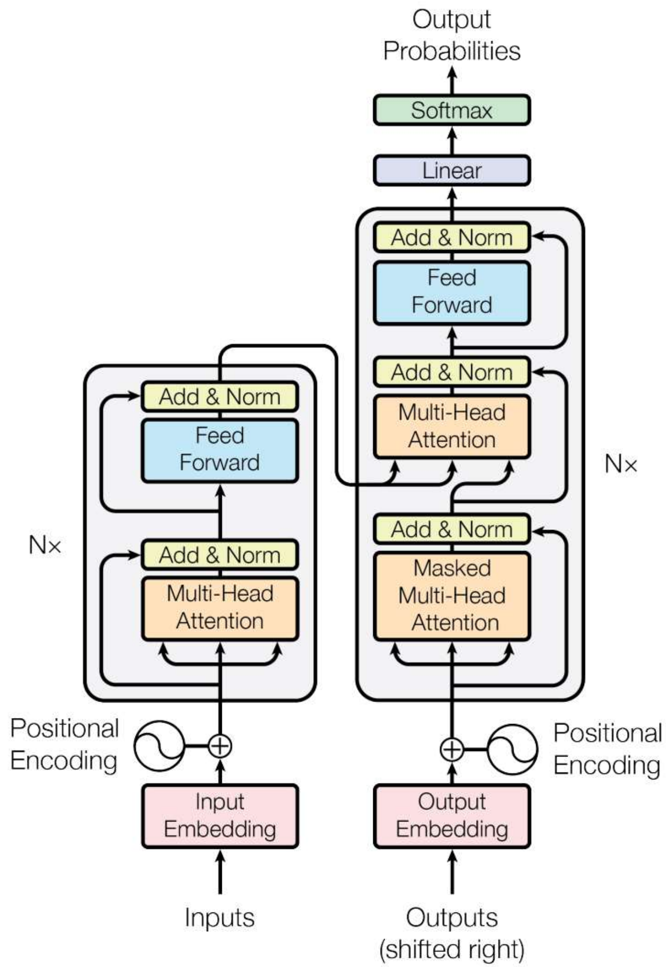
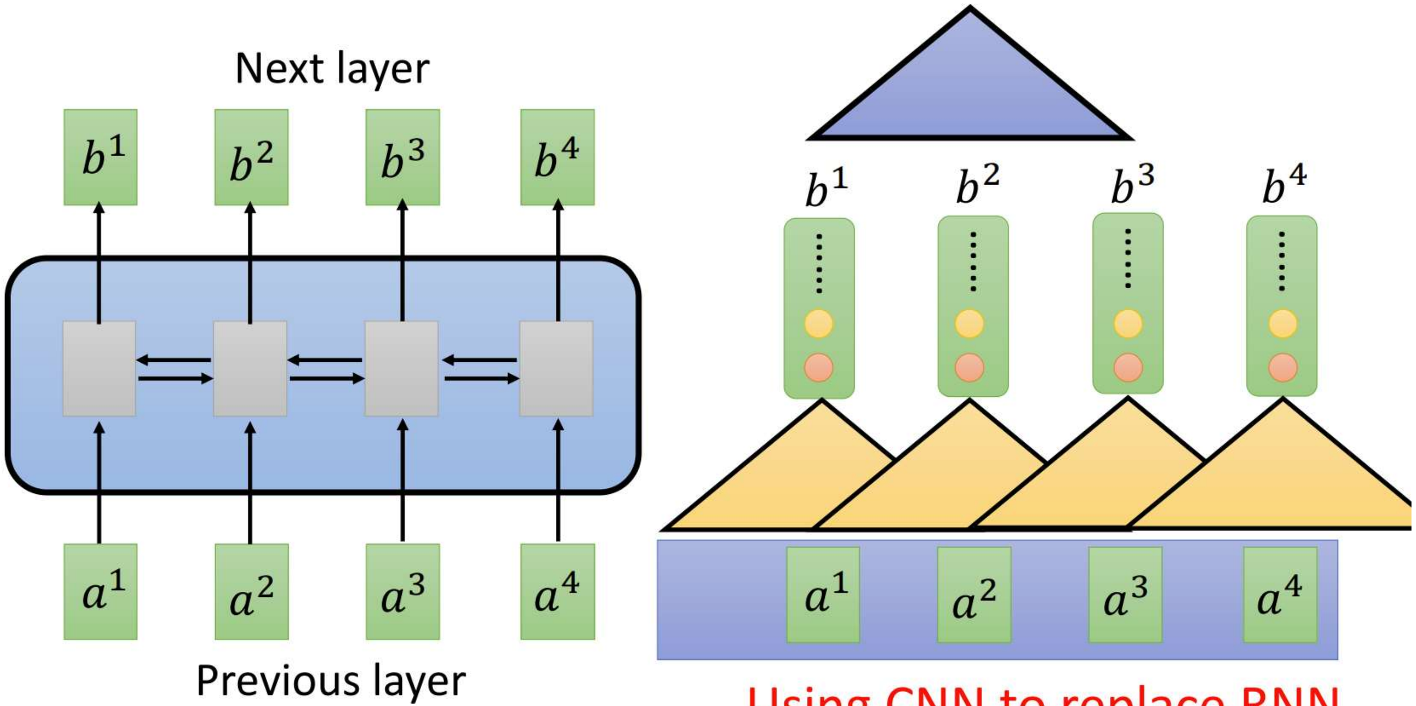


Figure 1: The Transformer - model architecture.

Why Transformer? Limitation of RNN/LSTM/CNN

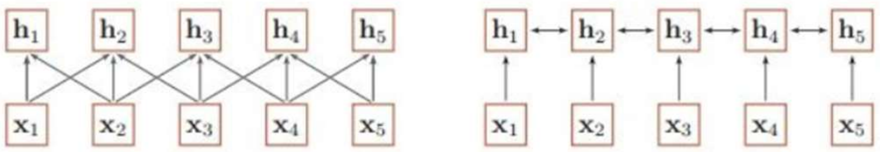


Hard to parallel

Using CNN to replace RNN
(CNN can parallel)

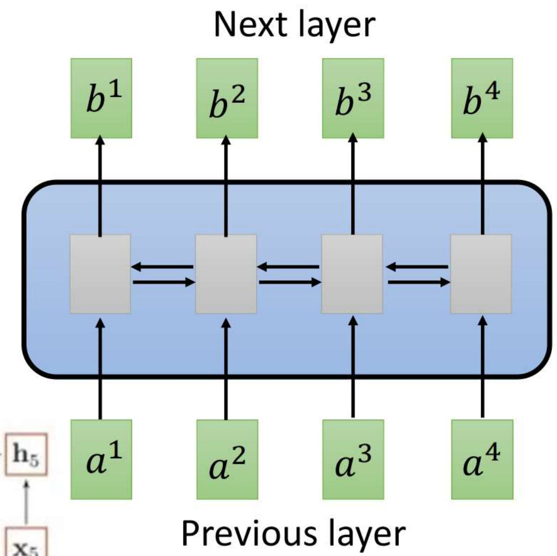
Why Transformer? Limitation of RNN/LSTM

- RNN/LSTM has **difficulties in remembering long-term dependencies** (suboptimal for processing long sequences)
- **Hard to parallel** (Slow)

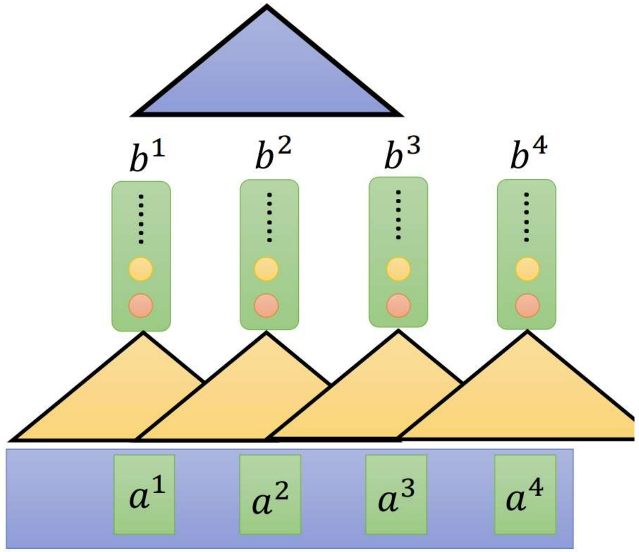


(a) 卷积网络 (b) 双向循环网络

基于卷积网络和循环网络的变长序列编码



Hard to parallel



Using CNN to replace RNN
(CNN can parallel)

Why Transformer? Limitation of TextCNN

- **Could not capture long-term dependencies**: TextCNN is an “improved n-gram model” (**limited-size filter** & **max-pooling**)
- Benefit from word2vec (better word representation)

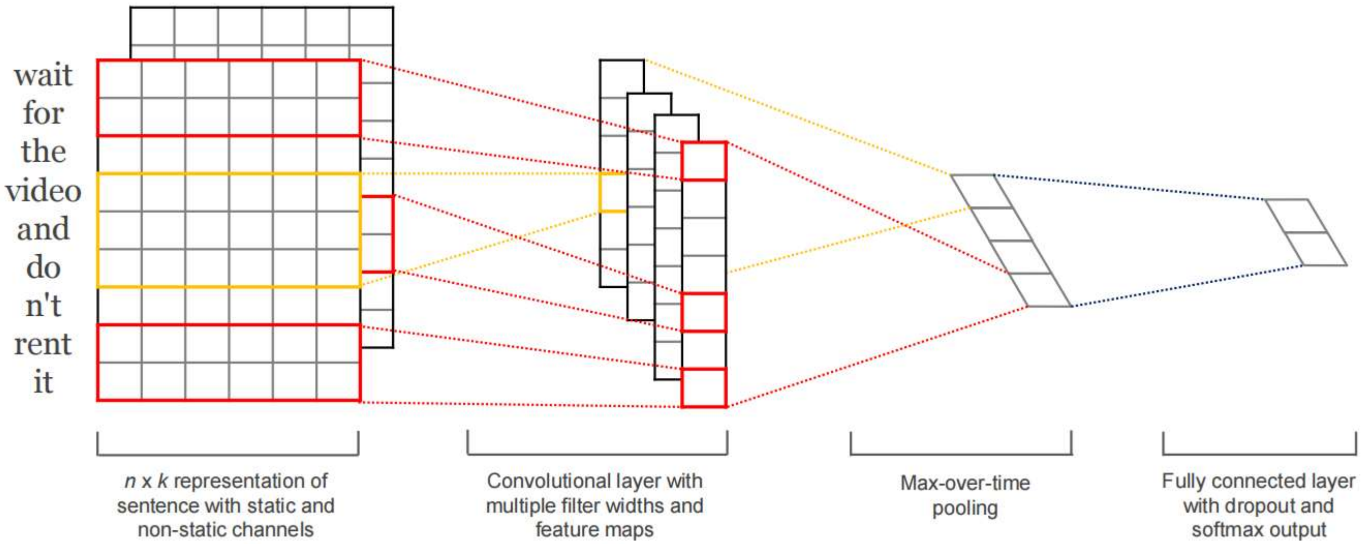
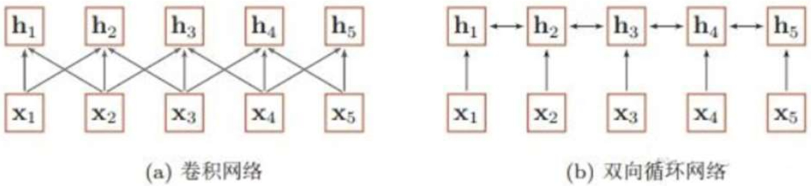


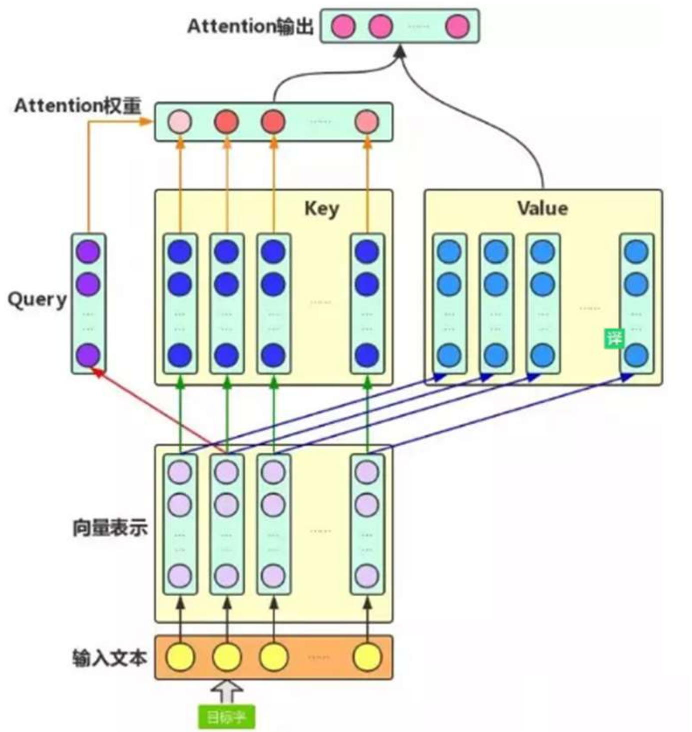
Figure 1: Model architecture with two channels for an example sentence.



基于卷积网络和循环网络的变长序列编码

So, Self-Attention is all you need!

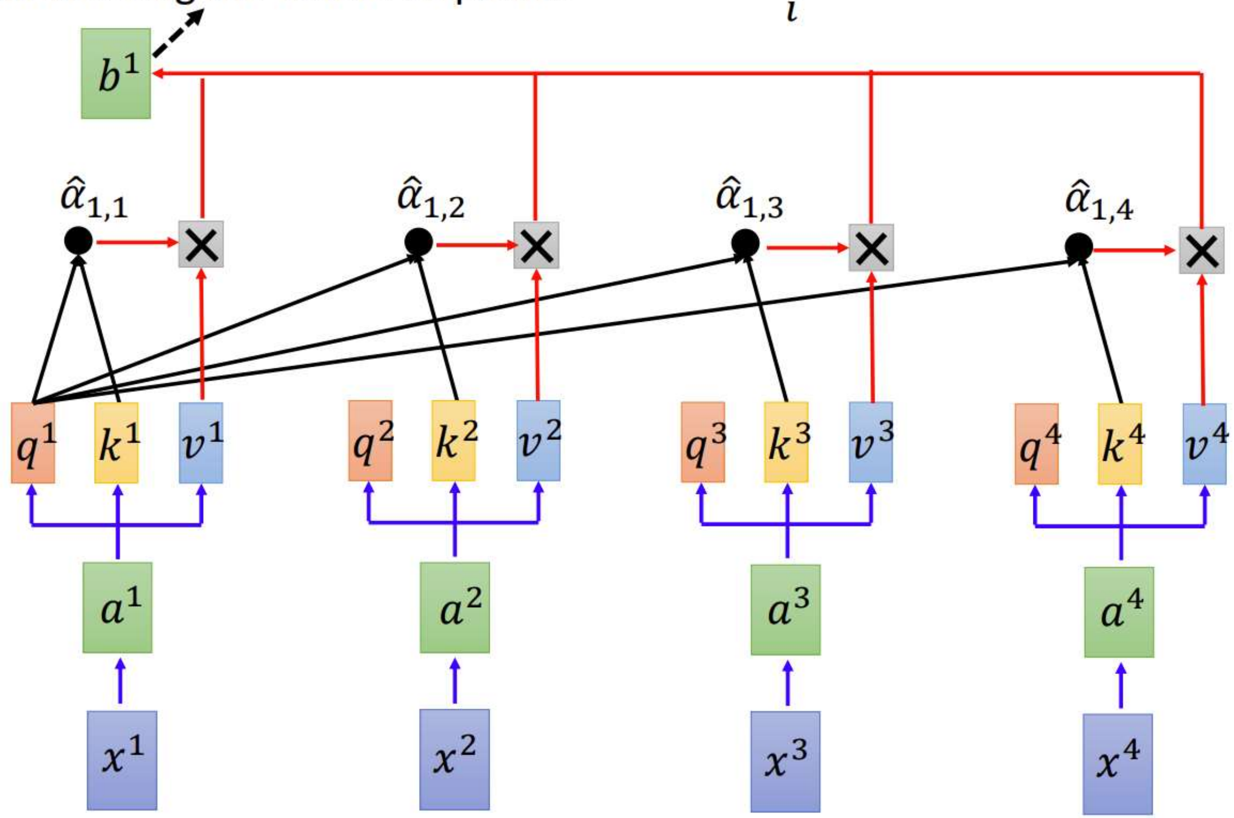
Transformer



Self-attention

Considering the whole sequence

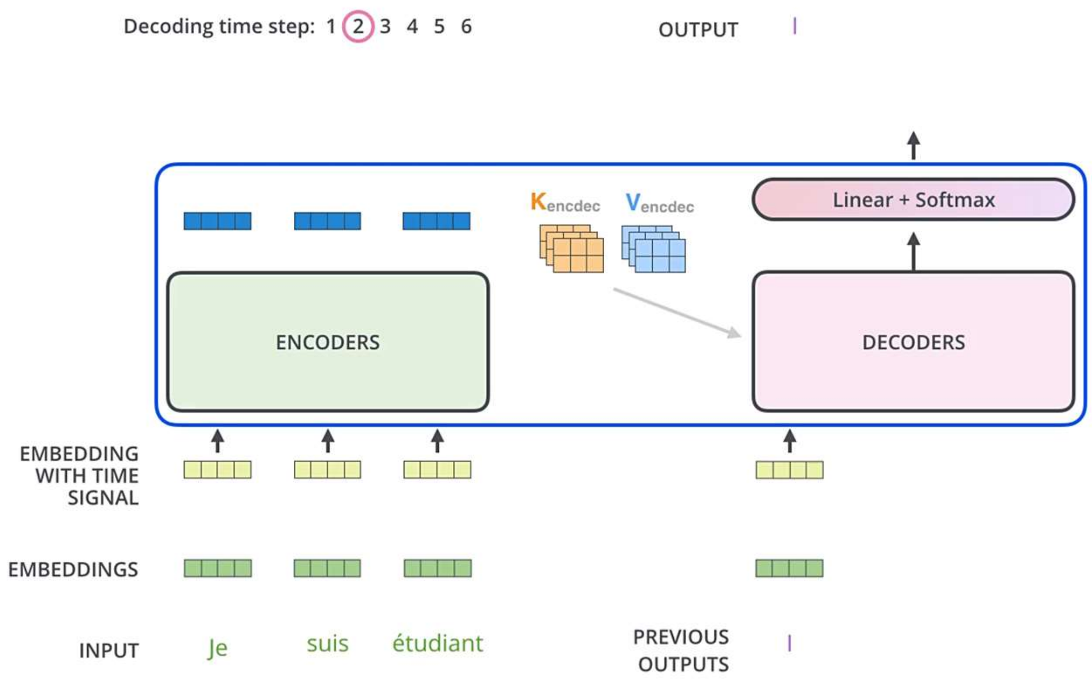
$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$



[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 + Self-Attention



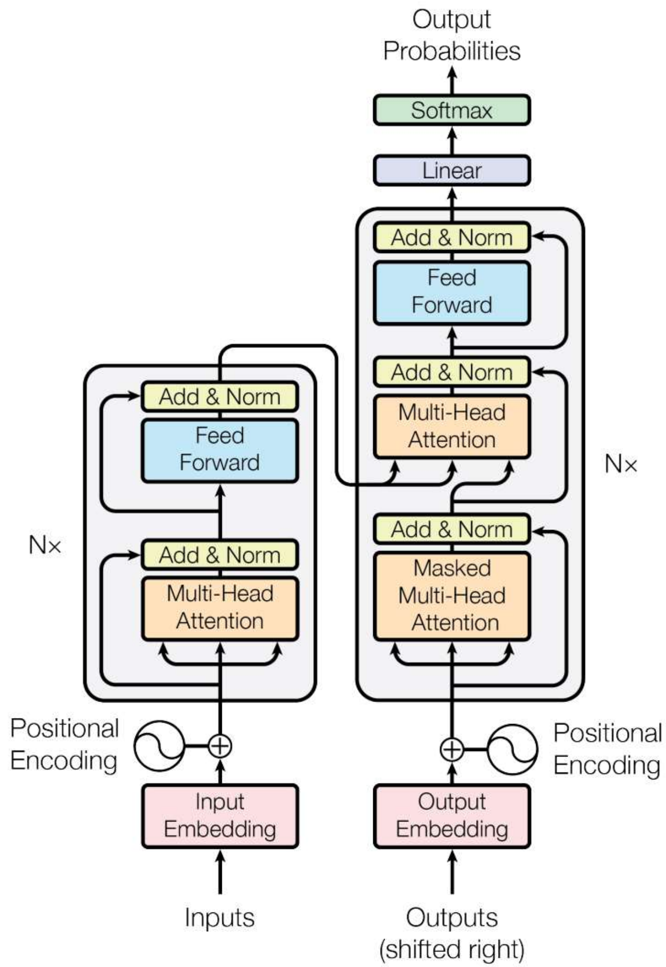


Figure 1: The Transformer - model architecture.

Transformer: Summary

- **Advantages:**
  - ➢ Solve Long-term dependencies: Distance between any token is 1
  - ➢ Easy to parallel: fast training
- **Disadvantages:**
  - ➢ Fail to capture local features
  - ➢ Positional Encoding

- 为什么 CNN 和 RNN 无法解决长距离依赖问题?
  - CNN:
    - 捕获信息的方式:
      - CNN 主要采用 卷积核 的 方式捕获 句子内的局部信息，你可以把他理解为 **基于 n-gram 的局部编码方式**捕获局部信息
    - 问题:
      - 因为是 n-gram 的局部编码方式，那么当 k 距离 大于 n 时，那么 y_t 将难以学习 x_{t-k} 信息;
    - 举例:
      - 其实 n-gram 类似于 人的 视觉范围，人的视觉范围 在每一时刻 只能 捕获 一定 范围内 的信息，比如，你在看前面的时候，你是不可能注意到背后发生了什么，除非你转过身往后看。
  - RNN:
    - 捕获信息的方式:
      - RNN 主要 通过 循环 的方式学习(记忆) 之前的信息x_{t};
    - 问题:
      - 但是随着时间 t 的推移，你会出现**梯度消失或梯度爆炸**问题，这种问题使你只能建立短距离依赖信息。
    - 举例:
      - RNN 的学习模式好比于 人类 的记忆力，人类可能会对 短距离内发生的 事情特别清楚，但是随着时间的推移，人类开始 会对好久之前所发生的事情变得印象模糊，比如，你对小时候发生的事情，印象模糊一样。
    - 解决方法:
      - 针对该问题，后期也提出了很多 RNN 变体，比如 LSTM、 GRU，这些变体 通过引入 门控的机制 来 有选择性 的记忆 一些 重要的信息，但是这种方法 也只能在 一定程度上缓解 长距离依赖问题，但是并不能 从根本上解决问题。

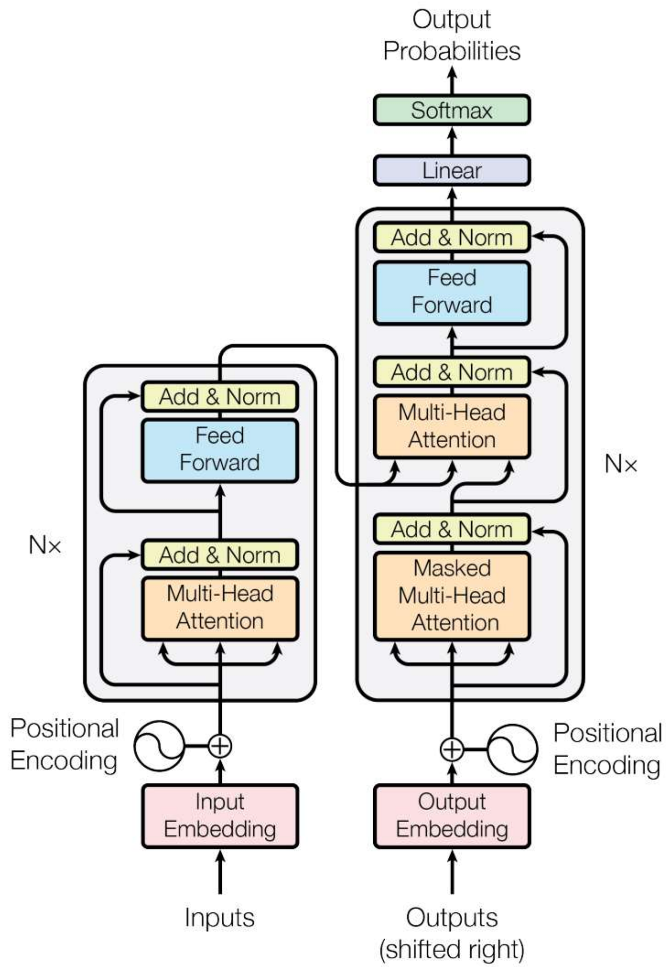


Figure 1: The Transformer - model architecture.

Overview

AutoTransform: Automated Code Transformation to Support Modern Code Review Process

- **Contribution:** RNN(LSTM) => **Transformer**, Code Abstraction => **BPE(byte pair encoding)**
- **Task:** Buggy Code => Refined Code
- **Experiment:** Ablation study to quantify the contributions of the two components (BPE and Transformer)

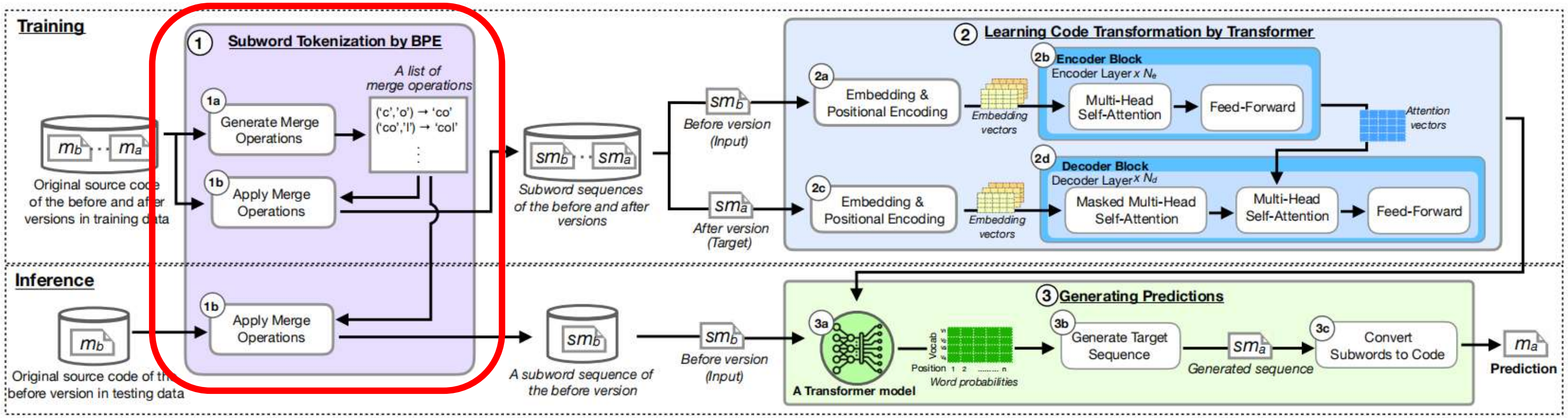


Figure 2: An overview of our AutoTransform.

Overview

AutoTransform: Automated Code Transformation to Support Modern Code Review Process

- **Contribution:** RNN(LSTM) => **Transformer**, Code Abstraction => **BPE(byte pair encoding)**
- **Task:** Buggy Code => Refined Code
- **Experiment:** Ablation study to quantify the contributions of the two components (BPE and Transformer)

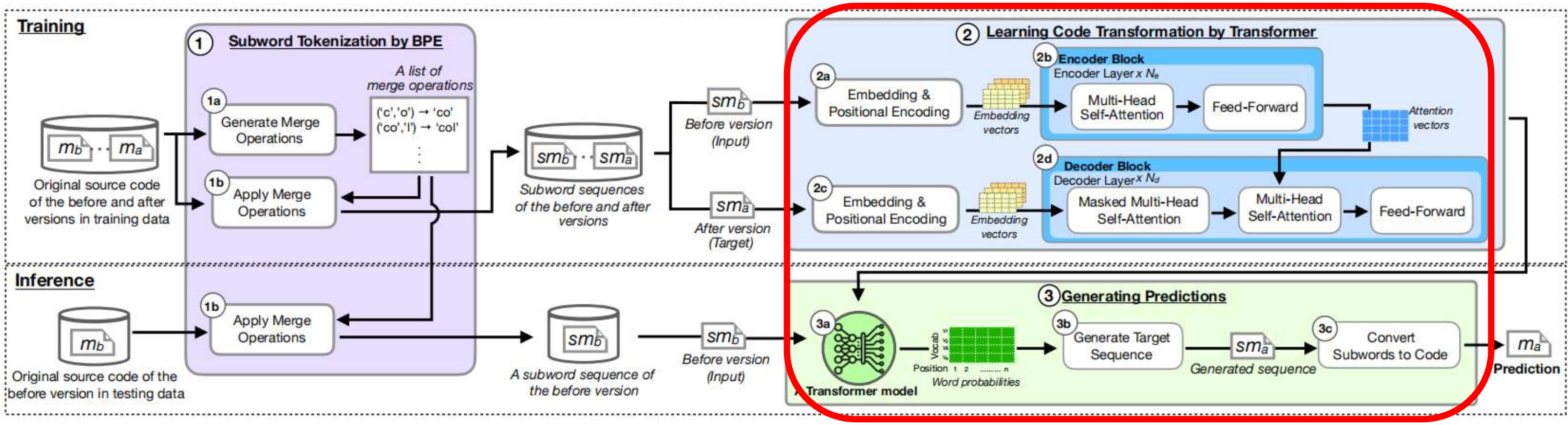


Figure 2: An overview of our AutoTransform.

Experiment

Experiment: Ablation Study

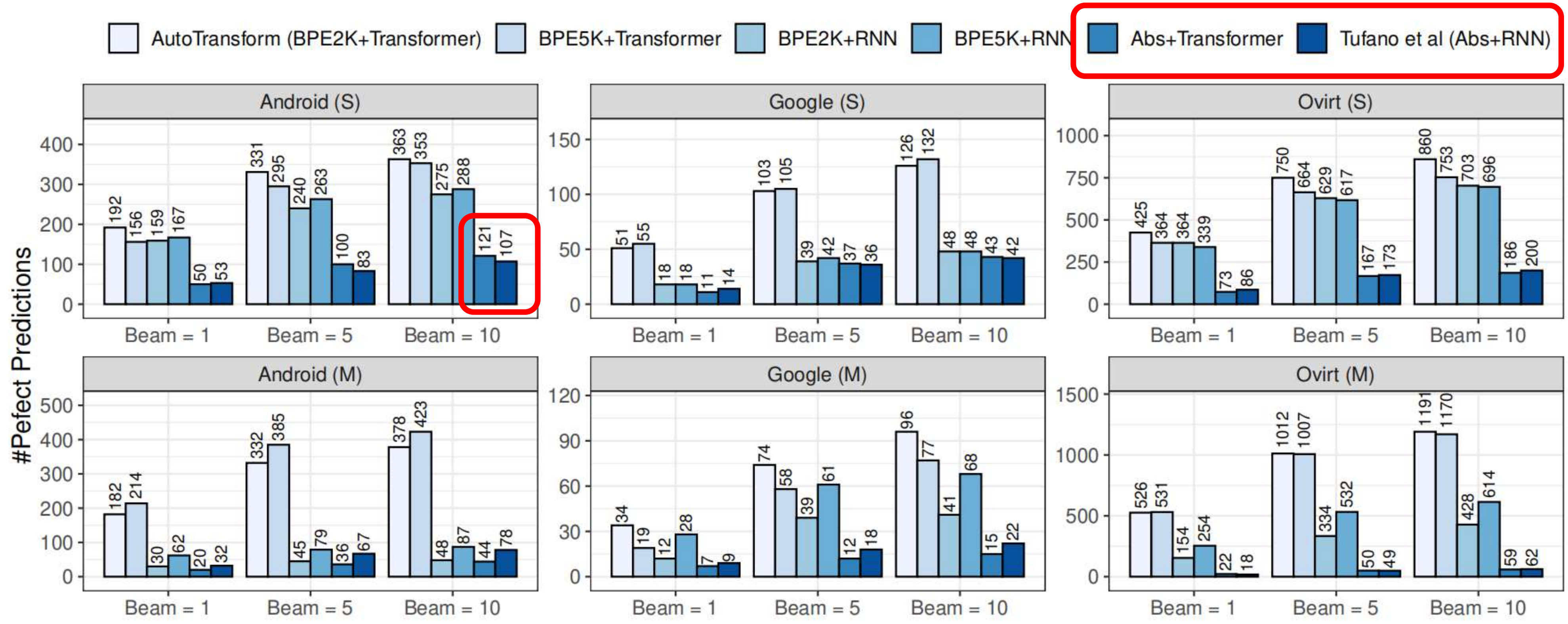


Figure 3: The perfect prediction of our AutoTransform when a component is varied. The y-axis shows the total number of perfect predictions of changed methods *with* and *without* new tokens.

Experiment: Ablation Study

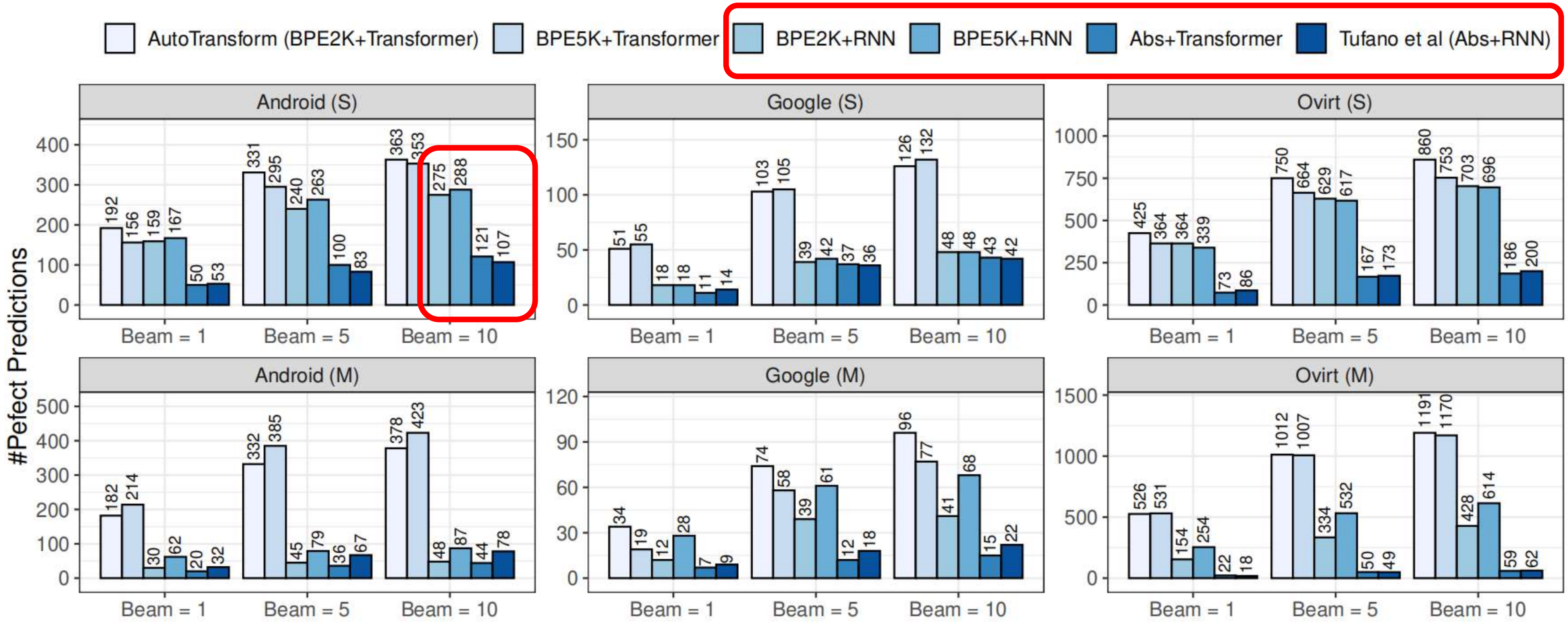


Figure 3: The perfect prediction of our AutoTransform when a component is varied. The y-axis shows the total number of perfect predictions of changed methods *with* and *without* new tokens.

Experiment: Ablation Study

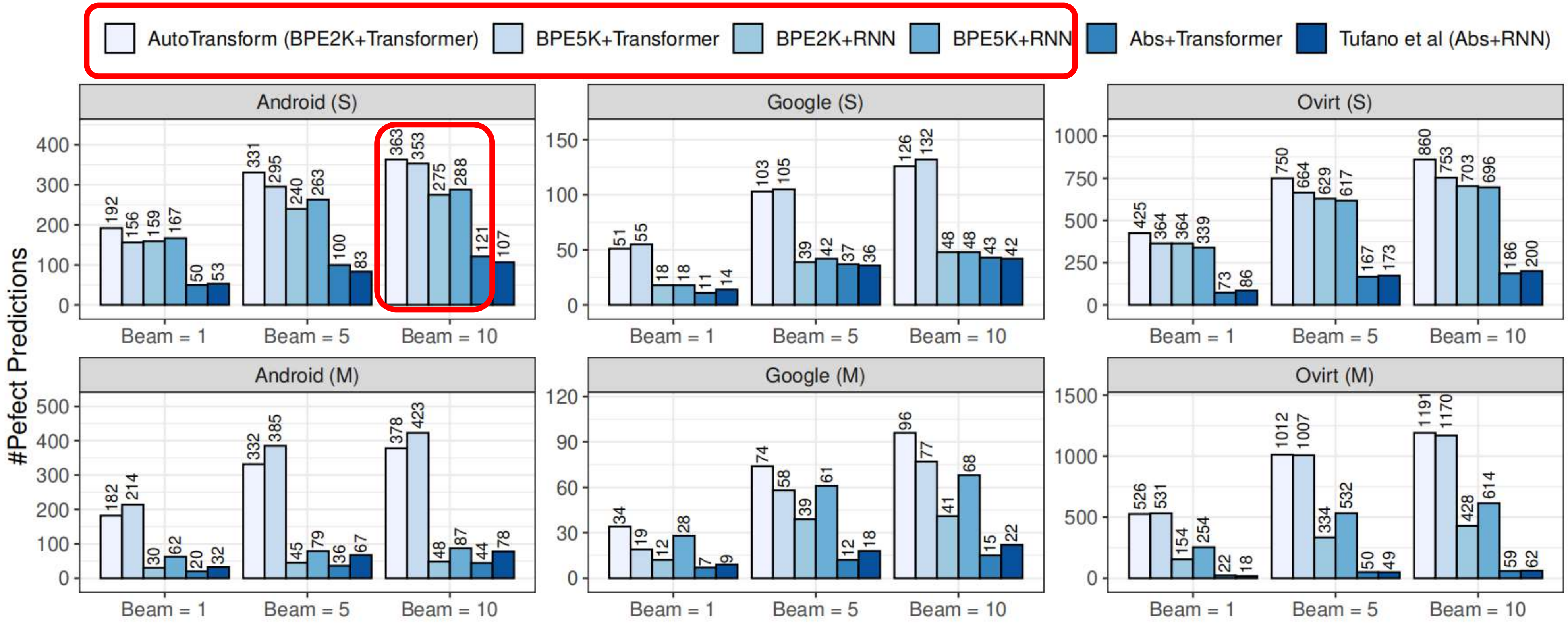


Figure 3: The perfect prediction of our AutoTransform when a component is varied. The y-axis shows the total number of perfect predictions of changed methods *with* and *without* new tokens.

Experiment: Source Code Pre-processing

src2abs

`src2abs` is a tool that *abstracts* Java source code.

It transforms this source code:

```
public static void main(String[] args) {
    console.println("Hello, World!");
}
```

into this abstract textual representation:

```
public static void METHOD_1 ( TYPE_1 [ ] VAR_1 ) { VAR_2 . METHOD_2 ( STRING_1 ) ; }
```

This abstract representations contains:

- Java Keywords;
- Code Separators;
- IDs in place of identifiers and literals;
- Idioms (optionally).

Learning Code Transformations via Neural Machine Translation
Michele Tufano
A Dissertation presented to the Graduate Faculty of The College of William & Mary in Candidacy for the Degree of Doctor of Philosophy

Pre-Prints

[P2] ***Unit Test Case Generation with Transformers***
M. Tufano, D. Drain, A. Svyatkovskiy, S. K. Deng, N. Sundaresan

[P1] ***Generating Accurate Assert Statements for Unit Test Cases using Pretrained Transformers***
M. Tufano, D. Drain, A. Svyatkovskiy, N. Sundaresan

International Journals

[J6] ***SEQUENCER: Sequence-to-Sequence Learning for End-to-End Program Repair***
Z. Chen, S. Kommrusch, **M. Tufano**, L.-N. Pouchet, D. Poshyvanyk, M. Monperrus
*IEEE Transactions on Software Engineering (**TSE 2019**)*

[J5] ***An Empirical Investigation into Learning Bug-Fixing Patches in the Wild via Neural Machine Translation***
M. Tufano, C. Watson, G. Bavota, M. Di Penta, M. White, D. Poshyvanyk
*ACM Transactions on Software Engineering and Methodology (**TOSEM 2019**)*

[J4] ***How Developers Micro-Optimize Android Apps***
M. Linares-Vásquez, C. Vendome, **M. Tufano**, and D. Poshyvanyk
*Journal of Systems and Software (**JSS 2017**)*

[J3] ***When and Why Your Code Starts to Smell Bad (and Whether the Smells Go Away)***
M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia, and D. Poshyvanyk
*IEEE Transactions on Software Engineering (**TSE 2017**)*

[1] https://github.com/micheletufano/src2abs
[2] https://tufanomichele.com/

Evaluation

Table 4: Perfect predictions (#PP) of our AutoTransform and Tufano *et al.* approach approach for the small and medium changed method with and without new tokens in the after version. The percentage value in the parenthesis indicates the percentage improvement of our AutoTransform.

Dataset			Beam width = 1		Beam width = 5		Beam width = 10	
			AutoTransform	**Tufano *et al.***	**AutoTransform**	**Tufano *et al.***	**AutoTransform**	**Tufano *et al.***
(Method Size)	**Change Type**	**#Test**	#PP	#PP	#PP	#PP	#PP	#PP
Android	w/o new tokens	443	84	53	125	83	130	107
(Small)	w/ new tokens	2,064	108	0	206	0	233	0
Google	w/o new tokens	228	11	14	22	36	29	42
(Small)	w/ new tokens	907	40	0	81	0	97	0
Ovirt	w/o new tokens	473	73	86	132	173	145	200
(Small)	w/ new tokens	2,328	352	0	618	0	715	0
Android	w/o new tokens	459	58	32	85	67	89	78
(Medium)	w/ new tokens	2,454	124	0	247	0	289	0
Google	w/o new tokens	283	16	9	28	18	33	22
(Medium)	w/ new tokens	1,162	18	0	46	0	63	0
Ovirt	w/o new tokens	622	111	18	179	49	199	62
(Medium)	w/ new tokens	3,327	415	0	833	0	992	0
Total	w/o new tokens	2,508	353	212	571	426	625	511
	w/ new tokens	12,242	1,060	0	2,031	0	2,389	0
	Both	14,750	**1,413 (+567%)**	212	**2,602 (+511%)**	426	**3,014 (+490%)**	511

1413/14750=9.58% **2602/14750=17.64%** **3014/14750=20.43%**

Evaluation

Compare to ICSE'21?

- Improve by 5%-7%
- No BLEU statistics: BLEU should not be used to evaluate the code transformation since the sequences that are similar

Beam Size	Perfect Predictions #	Perfect Predictions %	BLEU-4 mean	BLEU-4 median	BLEU-4 st. dev.	Levenshtein distance mean	Levenshtein distance median	Levenshtein distance st. dev.
1-encoder								
1	50	2.91%	0.7706	0.8315	0.1929	0.2383	0.2000	0.1670
3	156	9.07%	0.8468	0.8860	0.1419	0.1726	0.1454	0.1427
5	200	11.63%	0.8644	0.8980	0.1317	0.1554	0.1271	0.1348
10	271	15.76%	0.8855	0.9145	0.1166	0.1355	0.1092	0.1247
2-encoder								
1	209	12.16%	0.8164	0.8725	0.1863	0.1849	0.1422	0.1734
3	357	20.77%	0.8762	0.9244	0.1484	0.1321	0.0838	0.1468
5	422	24.55%	0.8921	0.9376	0.1351	0.1173	0.0696	0.1366
10	528	30.72%	0.9142	0.9543	0.1169	0.0953	0.0519	0.1204

ICSE21

Table 4: Perfect predictions (#PP) of our AutoTransform and Tufano *et al.* approach approach for the small and medium changed method with and without new tokens in the after version. The percentage value in the parenthesis indicates the percentage improvement of our AutoTransform.

Dataset (Method Size)	Change Type	#Test	Beam width = 1 AutoTransform #PP	Beam width = 1 Tufano *et al.* #PP	Beam width = 5 AutoTransform #PP	Beam width = 5 Tufano *et al.* #PP	Beam width = 10 AutoTransform #PP	Beam width = 10 Tufano *et al.* #PP
Android (Small)	w/o new tokens	443	84	53	125	83	130	107
	w/ new tokens	2,064	108	0	206	0	233	0
Google (Small)	w/o new tokens	228	11	14	22	36	29	42
	w/ new tokens	907	40	0	81	0	97	0
Ovirt (Small)	w/o new tokens	473	73	86	132	173	145	200
	w/ new tokens	2,328	352	0	618	0	715	0
Android (Medium)	w/o new tokens	459	58	32	85	67	89	78
	w/ new tokens	2,454	124	0	247	0	289	0
Google (Medium)	w/o new tokens	283	16	9	28	18	33	22
	w/ new tokens	1,162	18	0	46	0	63	0
Ovirt (Medium)	w/o new tokens	622	111	18	179	49	199	62
	w/ new tokens	3,327	415	0	833	0	992	0
Total	w/o new tokens	2,508	353	212	571	426	625	511
	w/ new tokens	12,242	1,060	0	2,031	0	2,389	0
	Both	14,750	**1,413 (+567%)**	212	**2,602 (+511%)**	426	**3,014 (+490%)**	511

1413/14750=9.58%

2602/14750=17.64%

3014/14750=20.43%

Evaluation

Compare to ICSE'21?

- Inferior to 2-encoder Transformer(ICSE21')
- Because 2-encoder imports NL comments which could guide the revision

Beam Size	Perfect Predictions #	Perfect Predictions %	BLEU-4 mean	BLEU-4 median	BLEU-4 st. dev.	Levenshtein distance mean	Levenshtein distance median	Levenshtein distance st. dev.
1-encoder								
1	50	2.91%	0.7706	0.8315	0.1929	0.2383	0.2000	0.1670
3	156	9.07%	0.8468	0.8860	0.1419	0.1726	0.1454	0.1427
5	200	11.63%	0.8644	0.8980	0.1317	0.1554	0.1271	0.1348
10	271	15.76%	0.8855	0.9145	0.1166	0.1355	0.1092	0.1247
2-encoder								
1	209	12.16%	0.8164	0.8725	0.1863	0.1849	0.1422	0.1734
3	357	20.77%	0.8762	0.9244	0.1484	0.1321	0.0838	0.1468
5	422	24.55%	0.8921	0.9376	0.1351	0.1173	0.0696	0.1366
10	528	30.72%	0.9142	0.9543	0.1169	0.0953	0.0519	0.1204

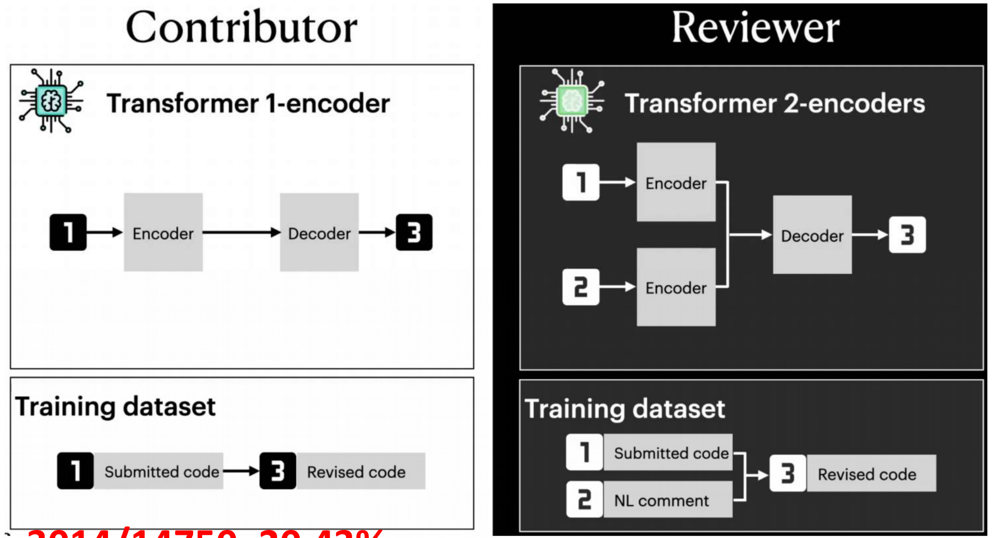


Table 4: Perfect predictions (#PP) of our AutoTransform and Tufano *et al.* approach approach f changed method with and without new tokens in the after version. The percentage value in the percentage improvement of our AutoTransform.

Dataset (Method Size)	Change Type	#Test	Beam width = 1 AutoTransform #PP	Beam width = 1 Tufano *et al.* #PP	Beam width = 5 AutoTransform #PP	Beam width = 5 Tufano *et al.* #PP	AutoT	
Android (Small)	w/o new tokens	443	84	53	125	83		
	w/ new tokens	2,064	108	0	206	0		
Google (Small)	w/o new tokens	228	11	14	22	36		
	w/ new tokens	907	40	0	81	0		
Ovirt (Small)	w/o new tokens	473	73	86	132	173		
	w/ new tokens	2,328	352	0	618	0		
Android (Medium)	w/o new tokens	459	58	32	85	67		
	w/ new tokens	2,454	124	0	247	0		
Google (Medium)	w/o new tokens	283	16	9	28	18		
	w/ new tokens	1,162	18	0	46	0		
Ovirt (Medium)	w/o new tokens	[illegible]	[illegible]	18	[illegible]	[illegible]	[illegible]	[illegible]
	w/ new tokens	[illegible]	[illegible]	0	[illegible]	0	[illegible]	0
Total	w/o new tokens	2,508	353	212	571	426	625	511
	w/ new tokens	12,242	1,060	0	2,031	0	2,389	0
	Both	14,750	**1,413 (+567%)**	212	**2,602 (+511%)**	426	**3,014 (+490%)**	511

1413/14750=9.58%

2602/14750=17.64%

3014/14750=20.43%

Reflection: Transformer + ?

Michele Tufano

Microsoft

Verified email at email.wm.edu - Homepage

Software Engineering Deep Learning Machine Learning

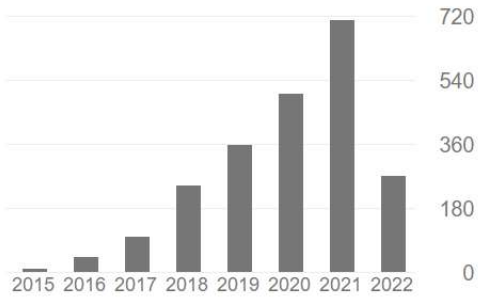
FOLLOW

GET MY OWN PROFILE

TITLE	CITED BY	YEAR
Deep learning code fragments for code clone detection M White, M Tufano, C Vendome, D Poshyvanyk 2016 31st IEEE/ACM International Conference on Automated Software …	453	2016
When and why your code starts to smell bad M Tufano, F Palomba, G Bavota, R Oliveto, M Di Penta, A De Lucia, ... 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering 1 …	301	2015
When and why your code starts to smell bad (and whether the smells go away) M Tufano, F Palomba, G Bavota, R Oliveto, M Di Penta, A De Lucia, ... IEEE Transactions on Software Engineering 43 (11), 1063-1088	169	2017
SequenceR: Sequence-to-Sequence Learning for End-to-End Program Repair Z Chen, S Kommrusch, M Tufano, LN Pouchet, D Poshyvanyk, ... IEEE Transactions on Software Engineering 47 (9), 1943-1959	161	2019

Cited by

	All	Since 2017
Citations	2274	2192
h-index	19	19
i10-index	25	25



[1] https://tufanomichele.com/

Reflection: Transformer + ?

Transformer

- Evaluating Representation Learning of Code Changes for Predicting Patch Correctness in Program Repair Haoye Tian, Kui Liu, Abdoul Kader Kaboreé, Anil Koyuncu, Li Li, Jacques Klein, Tegawendé F. Bissyandé
- Empirical Study of Transformers for Source Code Nadezhda Chirkova, Sergey Troshin
- Global Relational Models of Source Code Vincent J. Hellendoorn, Charles Sutton, Rishab Singh, Petros Maniatis, David Bieber
- Self-Supervised Bug Detection and Repair Miltiadis Allamanis, Henry Jackson-Flux, Marc Brockschmidt
- ProtoTransformer: A Meta-Learning Approach to Providing Student Feedback Mike Wu, Noah D. Goodman, Chris Piech, Chelsea Finn
- Retrieval Augmented Code Generation and Summarization Md Rizwan Parvez, Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, Kai-Wei Chang
- Show Your Work: Scratchpads for Intermediate Computation with Language Models Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, Augustus Odena
- CoTexT: Multi-task Learning with Code-Text Transformer Long Phan, Hieu Tran, Daniel Le, Hieu Nguyen, James Anibal, Alec Peltekian, Yanfang Ye
- Learning Type Annotation: Is Big Data Enough? Kevin Jesse, Premkumar Devanbu, Toufique Ahmed
- Code to Comment Translation: A Comparative Study on Model Effectiveness & Errors Junayed Mahmud, Fahim Faisal, Raihan Islam Arnob, Antonios Anastasopoulos, Kevin Moran
- ConTest: A Unit Test Completion Benchmark featuring Context Johannes Villmow, Jonas Depoix, Adrian Ulges
- Toward Less Hidden Cost of Code Completion with Acceptance and Ranking Models Jingxuan Li, Rui Huang, Wei Li, Kai Yao, Weiguo Tan
- Jointly Learning to Repair Code and Generate Commit Message Jiaqi Bai, Long

- Generating Bug-Fixes Using Pretrained Transformers Dawn Drain, Chen Wu, Alexey Svyatkovskiy, Neel Sundaresan
- Language-Agnostic Representation Learning of Source Code from Structure and Context Daniel Zügner, Tobias Kirschstein, Michele Catasta, Jure Leskovec, Stephan Günnemann
- Distilling Transformers for Neural Cross-Domain Search Colin B. Clement, Chen Wu, Dawn Drain, Neel Sundaresan
- Long-Range Modeling of Source Code Files with eWASH: Extended Window Access by Syntax Hierarchy Colin B. Clement, Shuai Lu, Xiaoyu Liu, Michele Tufano, Dawn Drain, Nan Duan, Neel Sundaresan, Alexey Svyatkovskiy
- Time-Efficient Code Completion Model for the R Programming Language Artem Popov, Dmitrii Orekhov, Denis Litvinov, Nikolay Korolev, Gleb Morgachev
- What do pre-trained code models know about code? Anjan Karmakar, Romain Robbes
- CodeTrans: Towards Cracking the Language of Silicon's Code Through Self-Supervised Deep Learning and High Performance Computing Ahmed Elnaggar, Wei Ding, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Silvia Severini, Florian Matthes, Burkhard Rost
- An Empirical Cybersecurity Evaluation of GitHub Copilot's Code Contributions Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, Ramesh Karri
- CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation Yue Wang, Weishi Wang, Shafiq Joty, Steven C.H. Hoi
- CommitBERT: Commit Message Generation Using Pre-Trained Programming Language Model Tae Hwan Jung
- Improving Code Autocompletion with Transfer Learning Wen Zhou, Seohyun Kim, Vijayaraghavan Murali, Gareth Ari Aye
- DIRECT : A Transformer-based Model for Decompiled Identifier Renaming Vikram Nitin, Anthony Saieva, Baishakhi Ray, Gail Kaiser

Contribute to ML4Code

[1] https://ml4code.github.io/tags.html#Transformer

Thanks

Zhu Jie

2022.04.28