ASE19-SRC-23

# Boosting Neural Commit Message Generation with Code Semantic Analysis

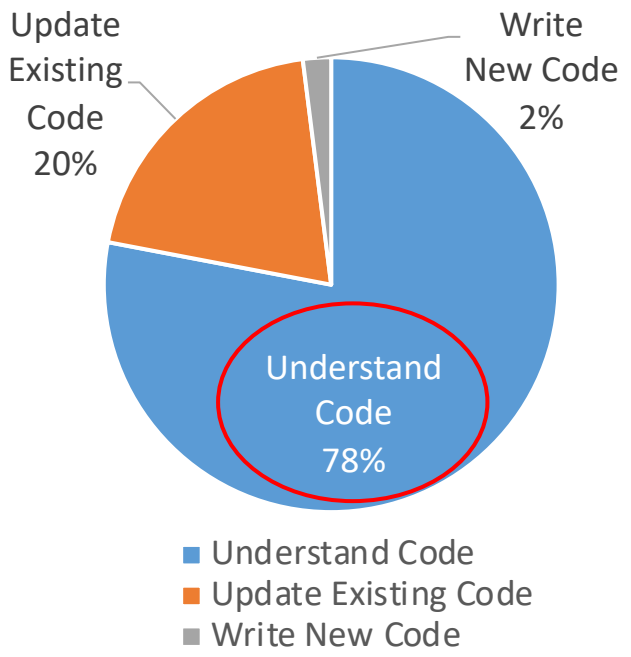
Shuyao Jiang

Fudan University, Shanghai, China

# Introduction

## Modern Software

- **Huge amount** of codes, **frequent** upgrades, and **numerous** team members
- **Code comprehension** is critical
  - Nearly 80% of development time is spent in understanding code[1]

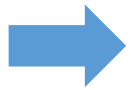



[1] P. Hallam. "What do programmers really do anyway". Microsoft Developer Network (MSDN)—C# Compiler, 2006.

# A Hot Line of Research: Codes → Human language

- Example: **commit messages**
  - A **short** natural language description
  - Summarizing the **contents** and **reasons** of code changes
- Question: How to **automatically** generate descriptions of code changes

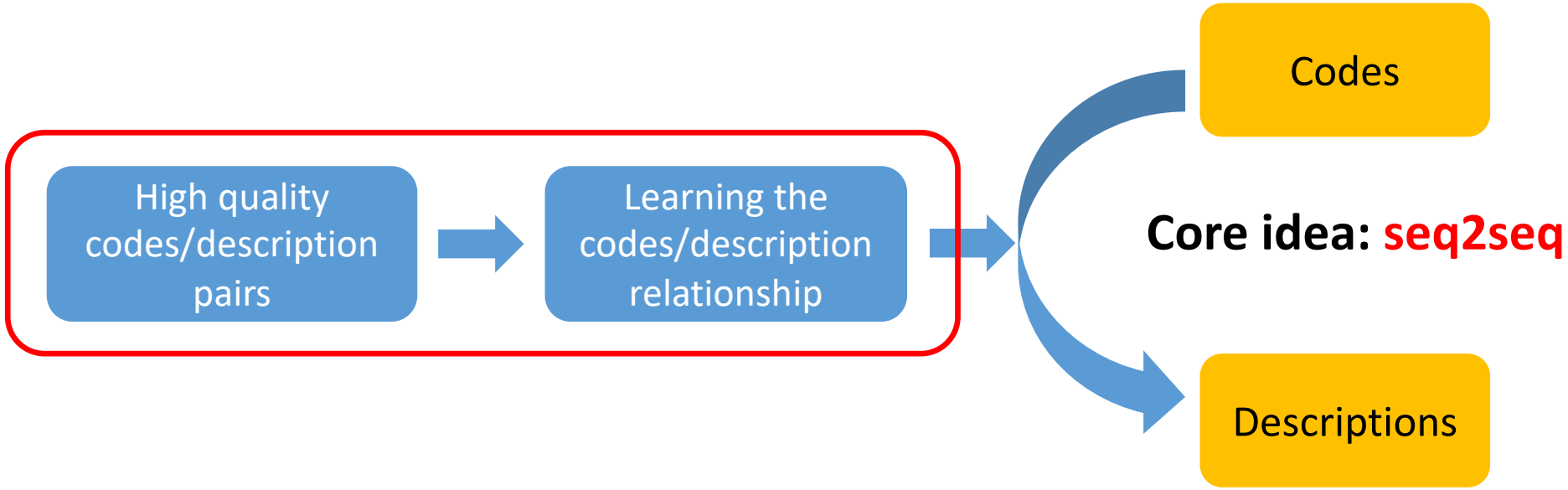
```
for(int i=0; i<len; i++)
{
    int min_index = i;
    for(int j=i+1; j<len; j++)
       if( array[min_index] > array[j])
         min_index = j;
    if( min_index != i )
      Swap(array+min_index,array+i);
}
```



?

# Current State-of-the-art Approaches

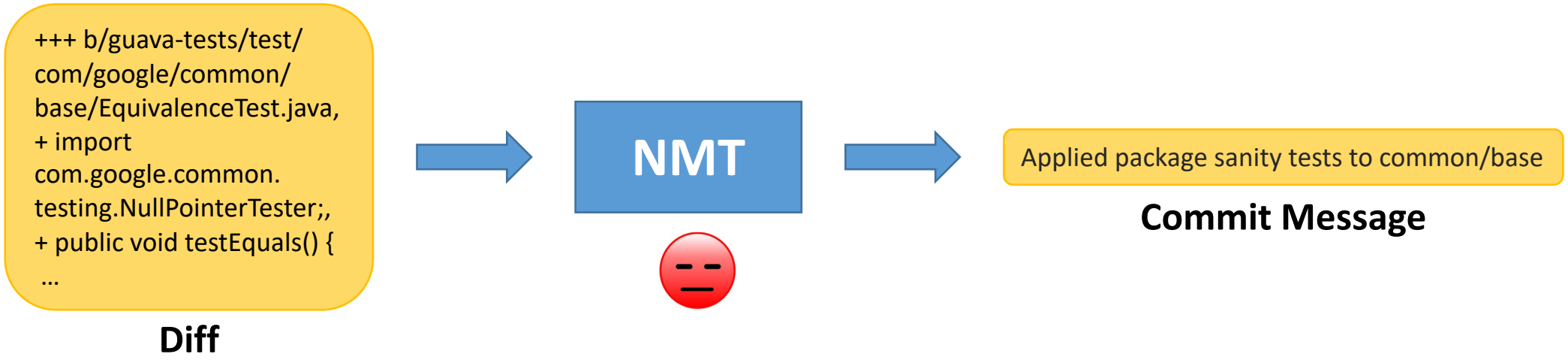
- With machine learning techniques
- “Codes → Human language" is a **translation** problem, i.e., **seq2seq**



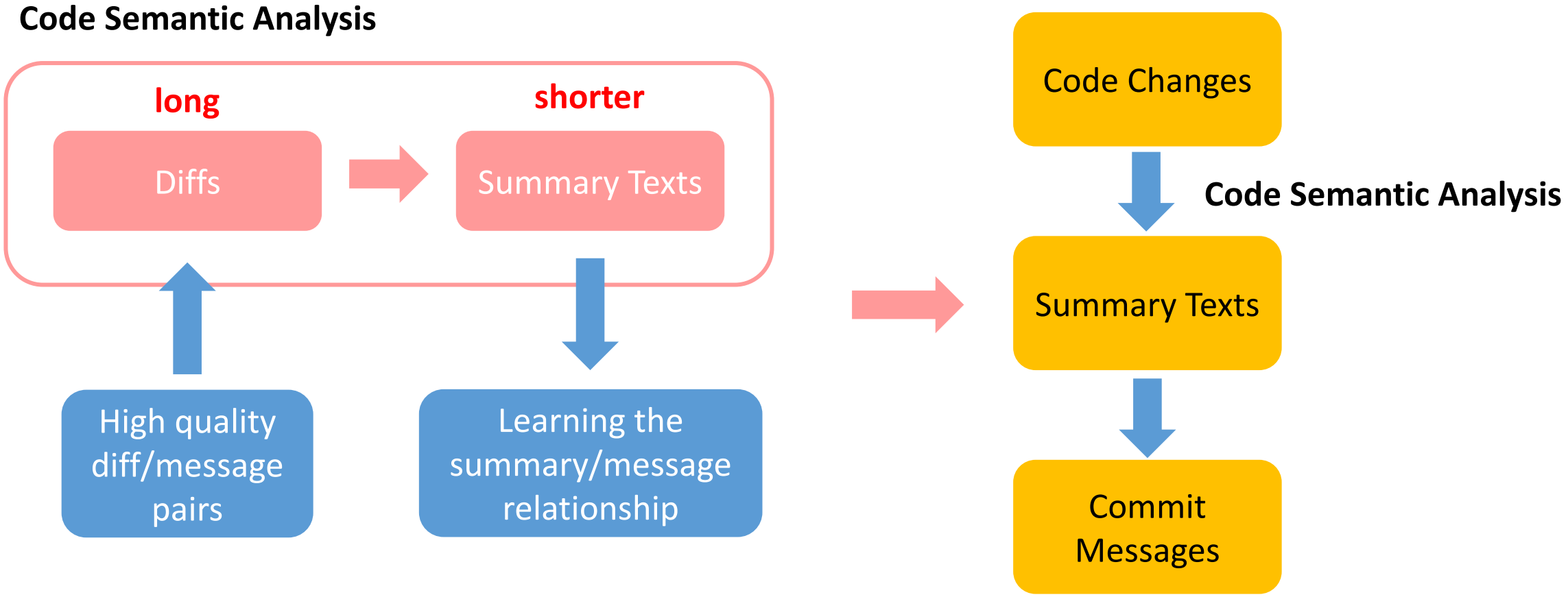

**Neural Machine Translation (NMT)**

# Use NMT to generate commit messages

- **Pros:**
  - Can generate commit messages automatically
  - Can utilize past human experiences (in training cases)
- **Cons:**
  - Generally suitable for inputs/outputs with **comparable lengths**
    - But code diffs are too lengthy:  containing too much **noisy information**



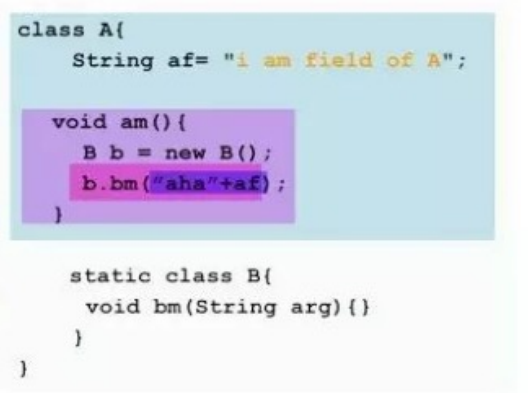
# Key Notion of Our Proposal: **Data Preprocessing**

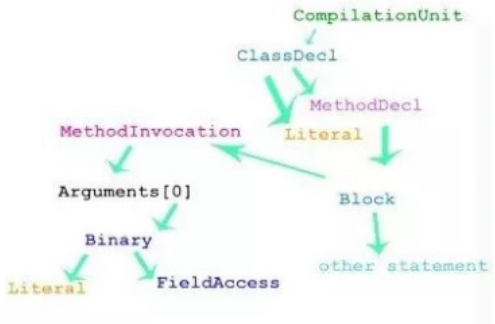



**Neural Machine Translation (NMT)**

# How to Preprocess Data

- **Code Semantic Analysis**
  - Code diffs → Abstract syntax tree (AST) → **Template**-based description
  - Lengthy code diffs → short descriptions




This change set is mainly composed of:
1. Changes to package …
1.1. Modifications to …
1.1.1. Add a constructor method
The added/removed methods triggered changes to …

**Code** **AST** **Summary Text**

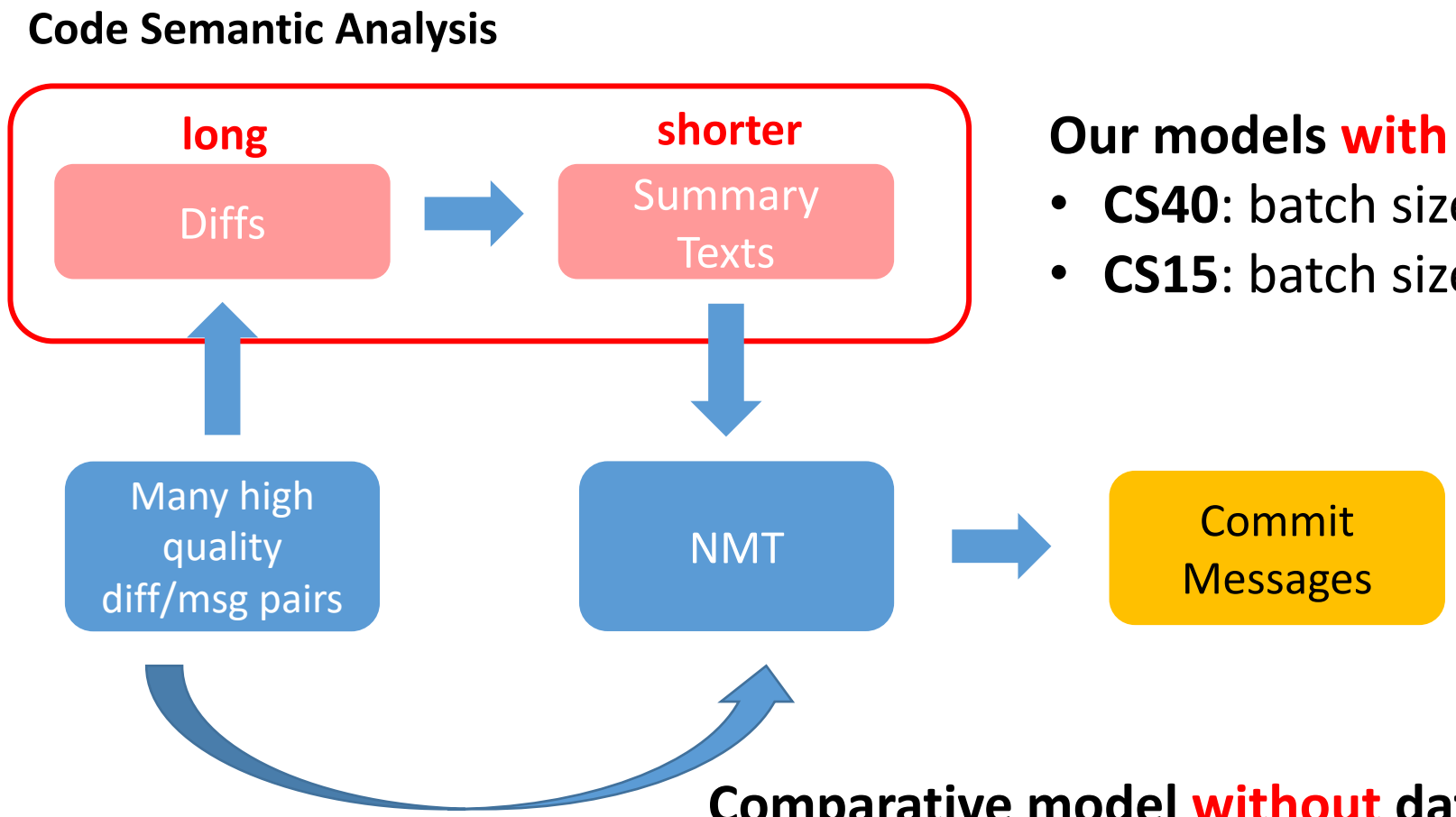
# Implementation and Experiments - Tool

- Our tool implementation / 4 modules
  - **Collection**: Automatically collect all the commits of each project of interest
    - Dataset construction details in next page
  - **Analysis**: For each commit, perform semantic analysis of code diffs
    - Based on ChangeScribe (http://www.cs.wm.edu/semeru/changescribe/)
  - **Generation**: Generate descriptions of all the commits
    - With deep neural networks
  - **Testing**: NMT model testing (quantification & case study)
- Last for 6 months: *2019.01– 2019.06*
  - Most of the time: used to read papers ☺, and to read codes of ChangeScribe

# Implementation and Experiments - Dataset

- Consideration
  - High-quality codes with high-quality commit messages
- So, we collect 18 projects from GitHub (**50k+ commits**)
  - **Popular**: each with 20k+ stars
  - **Large-scale**: each with 100k+ code lines, 3k+ commits
  - **Diverse**: including projects on mobile development, web application, Java core library, etc.
- Collect all the commits (code diffs and commit messages)
- Dataset and tool Online available
  - *https://github.com/ShuyaoJiang/CommitDataset*
  - Facilitate further follow-up research

# Model Training




**Our models with data preprocessing**

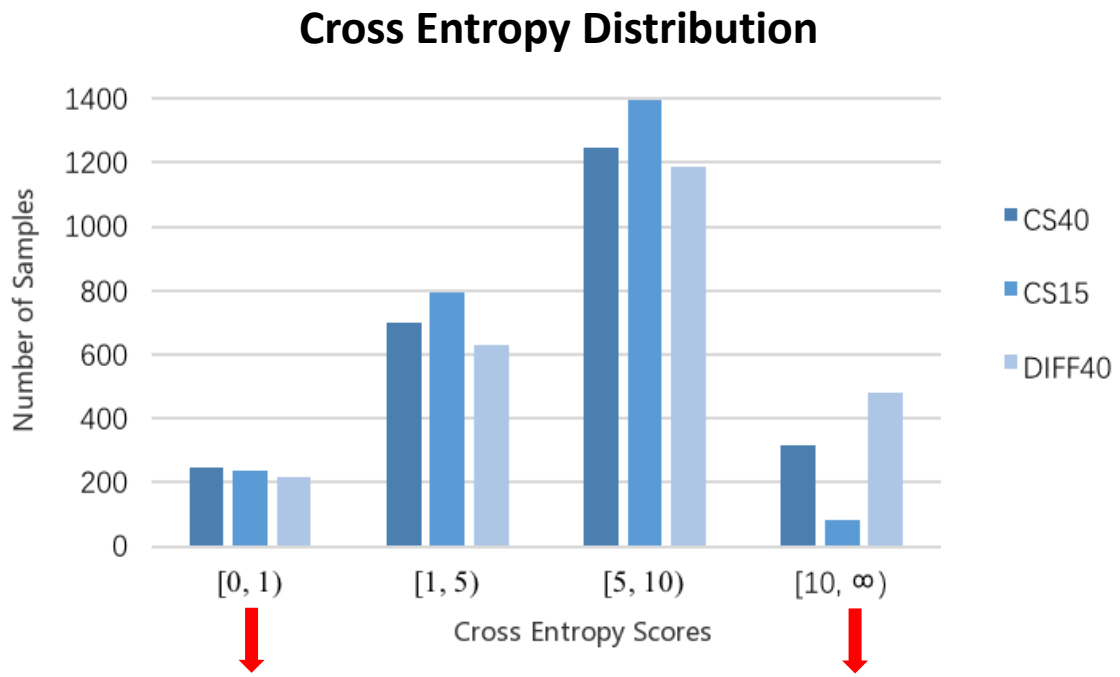
- **CS40**: batch size 40
- **CS15**: batch size 15

**Comparative model without data preprocessing**

- **DIFF40**: batch size 40

# Evaluation Results

- **Cross Entropy**
  - Measuring the **difference** between two probability distributions
  - The lower, the better (closer to the human-written commit messages)



# Evaluation Results

- **BLEU Scores**
  - Measuring the **similarity** between source and target sequence
  - The higher, the better (more similar to human-written commit messages)

**BLEU Scores**

| Model | BLEU-4 | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|---|---|---|---|---|---|
| CS40 | 1.10 | 4.7 | 1.7 | 0.5 | 0.4 |
| CS15 | 0.44 | 9.1 | 3.2 | 0.1 | 0.0 |
| DIFF40 | 0.41 | 3.9 | 0.9 | 0.1 | 0.1 |

$p_n$ is the modified n-gram precision used to calculate BLEU-4

**Our CS40 has the highest BLEU score**

- **Results**
  - Data preprocessing with AST analysis is effective!

# Translation Example

**Diff:**

```
+++ b/guava-tests/test/com/google/
common/base/EquivalenceTest.java,
+ import com.google.common.
testing.NullPointerTester;,
+ public void testEquals() {
 …
```

**DIFF40 result:**

Add <UNK>

**Summary Text:**

Changes to package com.google.common.base: Add a class for package sanity tests. It allows to: Instantiate package sanity tests.

**Our CS40 result:**

Add support for task properties

# Conclusions

- We suggest that **data preprocessing** is critical to commit message generation with NMT.
  - We apply **code semantic analysis** to tailor NMT inputs in commit message generation.
  - We conduct a **comprehensive evaluation** to prove that data preprocessing does improve NMT.
- This work sheds light to how to **properly apply existing DNN models** in software engineering tasks.