Решение задач программной инженерии на основе объединения информации из разных источников

Лобанов Артём Владиславович

Научный руководитель: к.т.н., Брыксин Тимофей Александрович
Рецензент: к.ф.-м.н., Фильченков Андрей Александрович

НИУ ВШЭ СПбШФМКН, 2021

Введение

- За 2020 создано более 60 миллионов[1] проектов и совершено более 1.9 миллиардов[1] коммитов
- Программная инженерия занимается оптимизацией связанных с разработкой процессов
- Во многих задачах приходится работать с разнородными данными
  - Классификация коммитов
  - Разметка issue и pull request-ов
  - Поиск ошибок
  - Анализ стек-трейсов

[1] Отчёт о статистике GitHub за 2020 год: https://octoverse.github.com

Существующие решения

- Классификация коммитов
  - Levin et al., “Boosting automatic commit classification into maintenance activities by utilizing source code changes”, 2017
    - BoW описаний коммитов + BoW типов изменений
  - Hoang et al., “PatchNet: Hierarchical Deep Learning-Based Stable Patch Identification for the Linux Kernel”, 2019
    - CNN для векторизации и текста, и изменений кода
- Определение ошибкоопасных коммитов
  - Hoang et al., “DeepJIT: an end-to-end deep learning framework for just-in-time defect prediction”, 2019.
    - CNN для векторизации и текста, и изменений кода
- Поиск кода по текстовому описанию
  - Gu et al., “Deep code search”, 2018.
    - RNN и для кода, и для текста

Мотивация

- Не во всех задачах применяются комбинированные решения
- Зачастую для анализа кода используются методы NLP
- Использование нерепрезентативных выборок
- Использование ансамблей позволит объединять любые модели

Цель и задачи

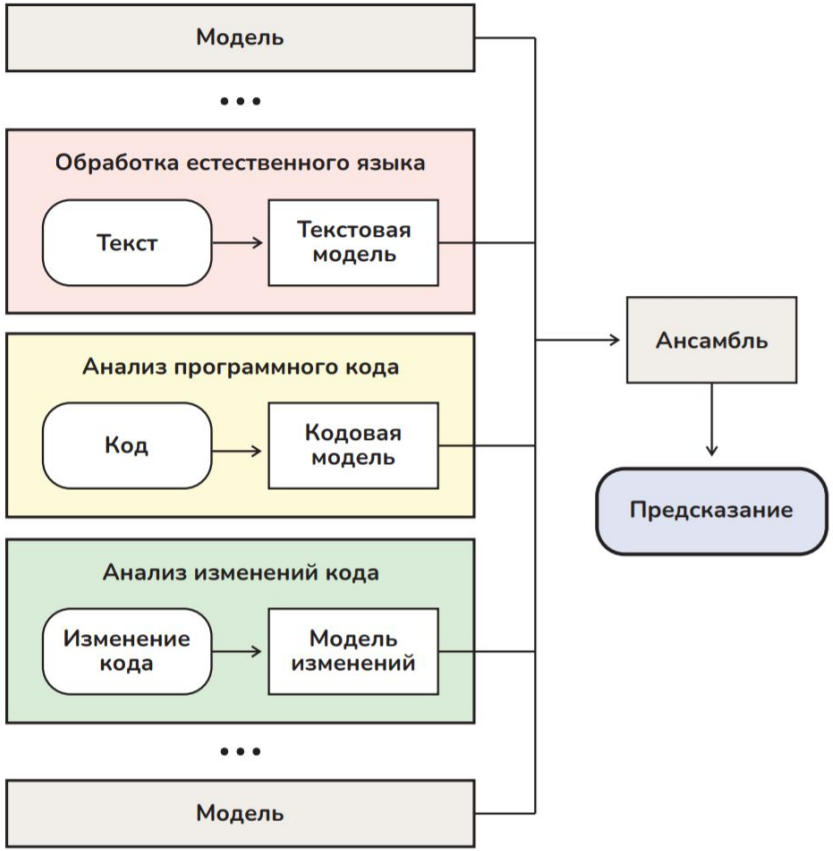
Цель: исследовать применимость ансамблей для объединения информации из разных источников в задачах программной инженерии

Задачи:

- Выделить основные типы информации в задачах программной инженерии
- Выбрать архитектуру решения, позволяющую эффективно объединять информацию из разных источников
- Выбрать модели, специфичные для выделенных типов информации
- Провести апробацию предложенной архитектуры на проблеме предсказания тегов задач спортивного программирования
- Провести апробацию предложенной архитектуры на задаче определения коммитов, исправляющих ошибки

Общая архитектура

- Обучение независимых специфичных для вида данных моделей
- Объединение моделей в ансамбль



Обработка естественного языка

- BoW
  - Не учитывается порядок слов
- RNN[1], LSTM[2], GRU[3]
  - Взрыв градиента[4]
  - Вымывание градиента[5]
- BERT[6]
  - Предобучение на большом корпусе текста
  - BPE для токенизации текста

[1] Medsker et al., “Recurrent neural networks”, 2001.
[2] Sundermeyer et al., “LSTM neural networks for language modeling”, 2012.
[3] Dey et al., “Gate-variants of gated recurrent unit (GRU) neural networks”, 2017.
[4] Kanai et al., “Preventing gradient explosions in gated recurrent units”, 2017.
[5] Hu et al., “Overcoming the vanishing gradient problem in plain recurrent networks”, 2018.
[6] Devlin et al., “Bert: Pre-training of deep bidirectional transformers for language understanding”, 2019.

Анализ исходного кода

- Метрики кода
  - Потеря информации
- code2seq[2], code2vec[1]
- CuBERT[3]
  - Переиспользование подхода из NLP
- GGNN[4, 5, 6]
  - Использует графовое представление структуры кода

[1] Alon et al., “code2seq: Generating Sequences from Structured Representations of Code”, 2018.
[2] Alon et al., “code2vec: Learning distributed representations of code”, 2019.
[3] Kanade et al., “Learning and Evaluating Contextual Embedding of Source Code”, 2020.
[4] Allamanis et al., “Typilus: neural type hints”, 2020.
[5] Ruiz et al., “Gated graph recurrent neural networks”, 2020
[6] Wang et al., “Detecting code clones with graph neural network and flow-augmented AST”, 2020

Представление изменений

- Построчная разница[1, 2]
  - Векторизация с помощью BoW
- Сценарии редактирования[3, 4, 5]
  - Векторизация с помощью BoW
- Графы изменений[6, 7]

[1] Kim et al., “lassifying software changes: Cleanor buggy?”, 2008.
[2] Giger et al., “Comparing fine-grained source code changes and code churn for bug prediction”, 2011.
[3] Falleri et al., “Fine-grainedand accurate source code differencing”, 2014.
[4] Kreutzer et al., “Automaticclustering of code change”, 2016.
[5] Matsumoto et al., “Beyond gumtree: A hybrid approach to generate edit scripts”, 2019.
[6] Nguyen et al., “Graph-based mining of in-the-wild, fine-grained, semantic code change patterns”, 2019.
[7] Golubev et al., “Changes from the trenches: Should we automate them?”, 2021.

Предсказание тегов: Постановка задачи

- Архив задач размечен фиксированным набором тегов (35 уникальных)
  - Примеры тегов: *графы*, *математика*, *динамическое программирование*
- Задача имеет условие на английском языке и решения на языке C++
- Автоматизация определения тегов позволит размечать новые задачи и выявлять ошибки в существующей разметке
- Датасет с платформы Codeforces[1]

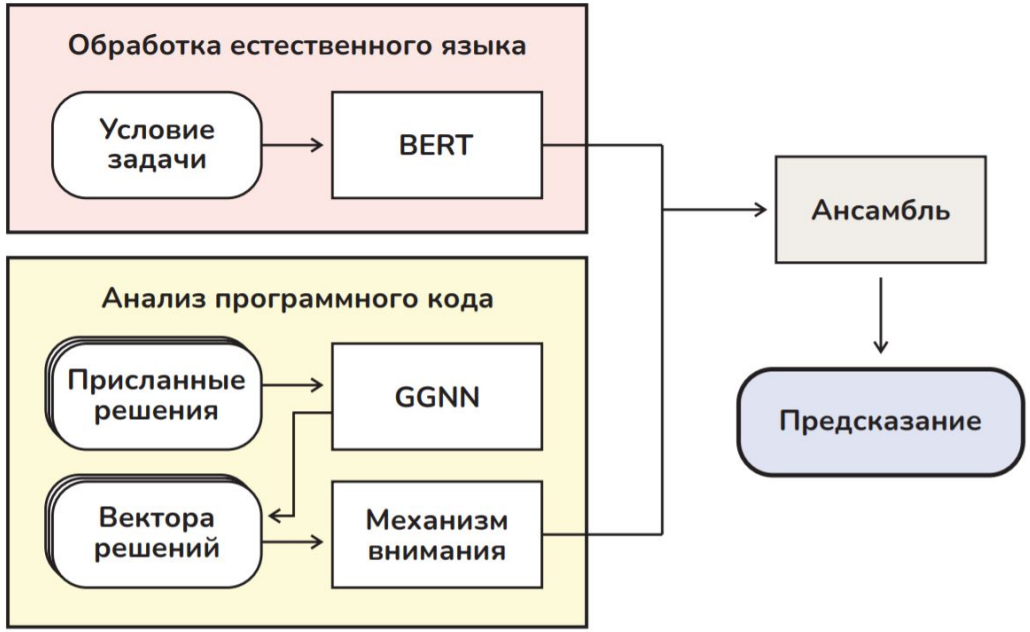
Выборка	Контексты	Задачи	Решения
Обучающая	716	3 837	2 493 745
Валидационная	169	1 020	664 578
Тестовая	159	1 046	734 612
Всего	1 044	5 903	3 892 935

[1] Платформа Codeforces: http://codeforces.com/

Предсказание тегов: Обучение модели

Обучение в 4 этапа:

1. BERT на условиях задач
2. GGNN на отдельных решениях
3. GGNN с механизмом внимания на всех решениях
4. Объединение моделей в ансамбль



Предсказание тегов: Результаты

Подход	PR-AUC	F1	P	R
Анализ текста				
TF-IDF + DT [1]	0.250	0.207	0.219	0.203
LSTM [2]	0.185	0.255	0.252	0.362
CNN Ensemble [3]	0.278	0.289	0.296	0.338
BERT (эта работа)	0.273	0.308	0.268	0.417
Анализ кода				
metrics + RF [4]	0.388	0.389	0.386	0.467
char-CNN [5]	0.372	0.360	0.377	0.414
GGNN + Attention (эта работа)	0.514	0.517	0.487	0.594
Комбинированные подходы				
BERT + GGNN (эта работа)	**0.542**	**0.537**	0.489	0.618

[1] Iancu et al., “Multi-label Classification for Automatic Tag Prediction in the Context of Programming Challenge”, 2019.
[2] Bora et al., “Predicting algorithmic approach for programming problems from natural language problem description”, 2016.
[3] Athavale et al., “Predicting Algorithm Classes for Programming Word Problem”, 2019.
[4] Shalaby et al., “Automatic algorithm recognition of source-code using machine learning”, 2017.
[5] Sudha et al., “Classification and Recommendation of Competitive Programming Problems Using CNN”, 2017.

Классификация коммитов: Постановка задачи

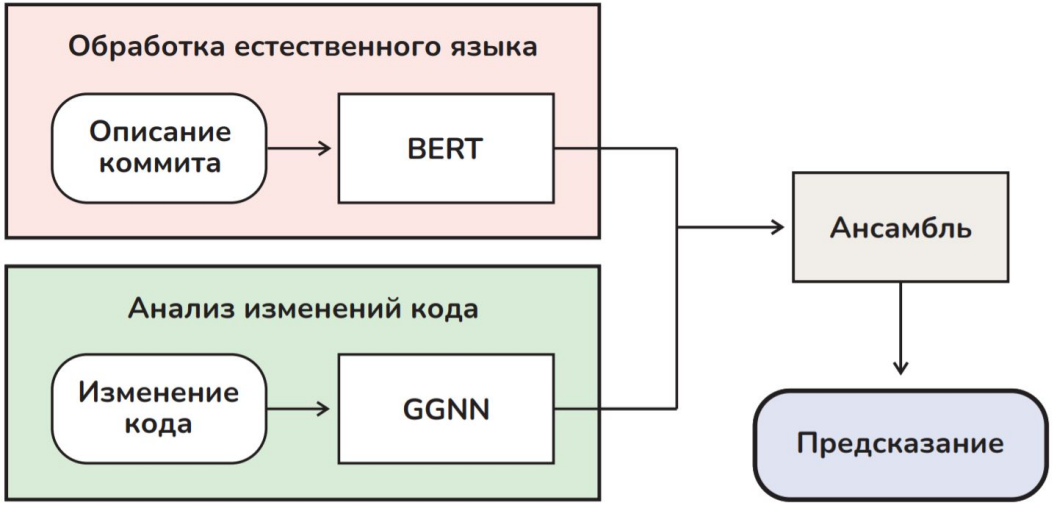
- На GitHub есть истории изменений множества проектов
- Каждый коммит имеет текстовое описание и ассоциированные с ним изменения кода
- Хотим по коммиту определять, исправляет ли он ошибку в проекте
  - Может быть полезно для локализации и автоисправления ошибок

Выборка	Проекты	Коммиты
Обучающая	4 438	70 365
Валидационная	583	9 247
Тестовая	552	2675
Всего	5 573	89 287

Классификация коммитов: Обучение модели

Обучение в 3 этапа

1. BERT на описания коммитов
2. GGNN на изменениях кода
3. Объединение моделей в ансамбль



Классификация коммитов: Результаты

Подход	ROC-AUC
Анализ текста	
RF (эта работа)	0.713
BERT (эта работа)	0.742
Анализ изменений	
GGNN (эта работа)	0.663
Комбинированные подходы	
PatchNet [1]	0.692
CC2Vec [2]	0.703
BERT + GGNN (эта работа)	**0.761**

[1] Hoang et al., “PatchNet: Hierarchical Deep Learning-Based Stable Patch Identification for the Linux Kernel”, 2019.
[2] Hoang et al., “CC2Vec: Distributed representations of code change”, 2020.

Итоги

- Показана эффективность ансамблей из моделей в задаче объединения информации из разных источников
- Получена модель для предсказания тегов
  - Показано, что модель значительно превосходит аналоги
  - Результаты представлены на конференции SEIM’21
- Получена модель для классификации коммитов
  - Собран датасет коммитов на основе пул-реквестов
  - Показано, что модель превосходит аналоги

Ответы на рецензию

Претензия: Неочевидна релевантность раздела 4.3.2 исследованию

Ответ: Этот раздел посвящён сравнению сложности датасетов

Подход	ROC-AUC (GitHub)	ROC-AUC (Linux)
PatchNet [1]	0.692	0.860
CC2Vec [2]	0.703	0.916
RF (эта работа)	0.713	0.959

[1] Hoang et al., “PatchNet: Hierarchical Deep Learning-Based Stable Patch Identification for the Linux Kernel”, 2019.
[2] Hoang et al., “CC2Vec: Distributed representations of code change”, 2020.

Ответы на рецензию

Претензия: Почему для сравнения предсказания тэгов использовались только работы [1] и [2] (обе 2017 года), а не более новые: [3], [4], [5]?

Ответ: Мы сравнивались не с двумя, а с пятью работами ([3], [6], [7], [8], [9]), в том числе с одной из работ, указанных рецензентов – [3]. Работы [3] и [6] датируются 2019 годом.

[1] Sudha et al., “Classification and Recommendation of Competitive Programming Problems Using CNN”, 2017.
[2] Shalaby et al., “Automatic algorithm recognition of source-code using machine learning”, 2017.
[3] Athavale et al., “Predicting algorithm classes for programming word problems”, 2019.
[4] Ohashi et al., “Convolutional neural network for classification of source codes”, 2019.
[5] Rahman et al., “Source code assessment and classification based on estimated error probability using attentive LSTM language model and its application in programming education”, 2020.
[6] Iancu et al., “Multi-label Classification for Automatic Tag Prediction in the Context of Programming Challenge”, 2019.
[7] Bora et al., “Predicting algorithmic approach for programming problems from natural language problem description”, 2016.
[8] Shalaby et al., “Automatic algorithm recognition of source-code using machine learning”, 2017.
[9] Sudha et al., “Classification and Recommendation of Competitive Programming Problems Using CNN”, 2017.

Ответы на рецензию

Претензия: Почему для двух классификационных задач (предсказания тэгов и выявления bug-fix коммитов) используются разные метрики?

Ответ:

- ROC-AUC и PR-AUC довольно похожи
  - Кривая модели доминирует в пространстве ROC если и только если она доминирует в пространстве PR[1]
- PR-AUC более устойчива в случае несбалансированных выборок[1, 2]

[1] Davis et al., “The relationship between Precision-Recall and ROC curves”, 2006.
[2] Ozenne et al., “The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases”, 2015.

Предсказание тегов: Результаты (текст)

Подход	PR-AUC	F1	P	R
TF-IDF + DT [1]	0.250	0.207	0.219	0.203
LSTM [1]	0.203	0.259	0.212	0.409
word2vec + LSTM [1]	0.210	0.280	0.252	0.400
BoW + LR [2]	0.227	0.252	0.252	0.306
LSTM [2]	0.177	0.236	0.188	0.385
word2vec + LSTM [2]	0.185	0.255	0.252	0.362
TWE + CNN [3]	0.268	0.303	0.295	0.372
GloVe + CNN [3]	0.276	0.292	0.263	0.388
CNN Ensemble [3]	0.278	0.289	0.296	0.338
BERT (эта работа)	0.273	0.308	0.268	0.417

[1] Iancu et al., “Multi-label Classification for Automatic Tag Prediction in the Context of Programming Challenge”, 2019.
[2] Bora et al., “Predicting algorithmic approach for programming problems from natural language problem description”, 2016.
[3] Athavale et al., “Predicting Algorithm Classes for Programming Word Problem”, 2019.

Предсказание тегов: Результаты (код, решения)

Подход	PR-AUC	F1	P	R
char-CNN [1]	0.249	0.289	0.306	0.323
metrics + RF [2]	0.254	0.303	0.282	0.376
metrics + AdaBoost [2]	0.225	0.270	0.208	0.474
metrics + SVM [2]	0.225	0.272	0.245	0.495
GGNN (эта работа)	0.393	0.430	0.467	0.446

[1] Sudha et al., “Classification and Recommendation of Competitive Programming Problems Using CNN”, 2017.
[2] Shalaby et al., “Automatic algorithm recognition of source-code using machine learning”, 2017.

Предсказание тегов: Результаты (код, задачи)

Подход	PR-AUC	F1	P	R
char-CNN [1]	0.372	0.360	0.377	0.414
metrics + RF [2]	0.388	0.389	0.386	0.467
metrics + AdaBoost [2]	0.325	0.356	0.341	0.452
metrics + SVM [2]	0.363	0.339	0.297	0.564
GGNN (эта работа)	0.514	0.517	0.487	0.594

[1] Sudha et al., “Classification and Recommendation of Competitive Programming Problems Using CNN”, 2017.
[2] Shalaby et al., “Automatic algorithm recognition of source-code using machine learning”, 2017.

Предсказание тегов: Корреляция