НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Санкт-Петербургская школа физико-математических и компьютерных наук

Построение векторных представлений для изменений программного кода

Выпускная квалификационная работа бакалавра

Студент: Правилов Михаил Егорович
Научный руководитель: Брыксин Тимофей Александрович, к.т.н.
Рецензент: Луцив Дмитрий Вадимович, к.ф.-м.н.

Санкт-Петербург, 2020

Введение в область

Программный код постоянно меняется

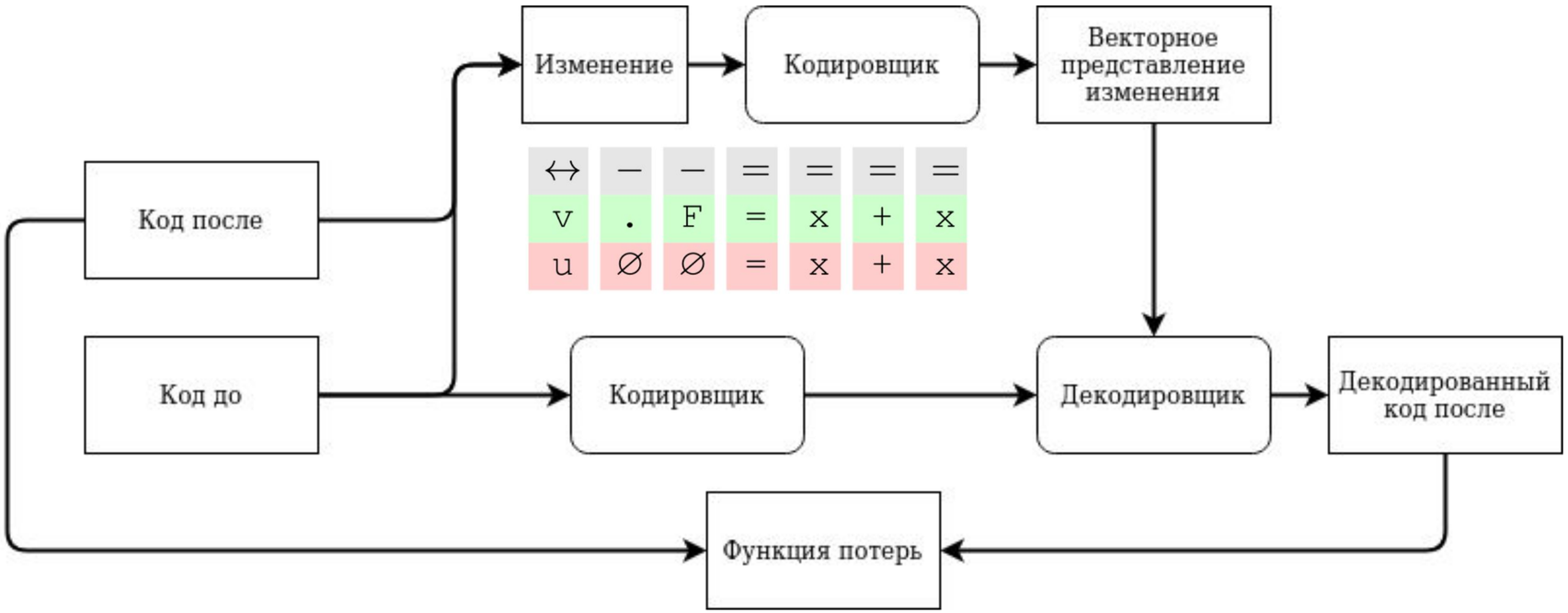
Во многих задачах программной инженерии именно изменения играют одну из ключевых ролей:

- Генерация сообщений к изменениям (коммитам)
- Исправление ошибок в программном коде
- Классификация изменений (уязвимых, стабильных патчей)

Введение в область

Yin et al., Learning To Represent Edits, ICLR 2019:

- Задача для обучения **без учителя**: применить закодированное изменение к коду
- Оценка на **размеченном** датасете C# Fixers, задача: применить мелкий рефакторинг



Цель и задачи

Цель: предложить подход, не требующий ручной разметки данных, для задач исправления ошибок в коде и генерации сообщений к коммитам

Задачи:

1. Реализовать модель, предложенную Yin et al.
2. Предложить улучшения модели
3. Провести апробацию на задаче исправления ошибок в коде
4. Провести апробацию на задаче генерации сообщений к коммитам

Реализованная модель

- Код = последовательность токенов
- Основа модели: машинный перевод
- Кодирование и декодирование с помощью LSTM
- Реализованы механизмы внимания и копирования

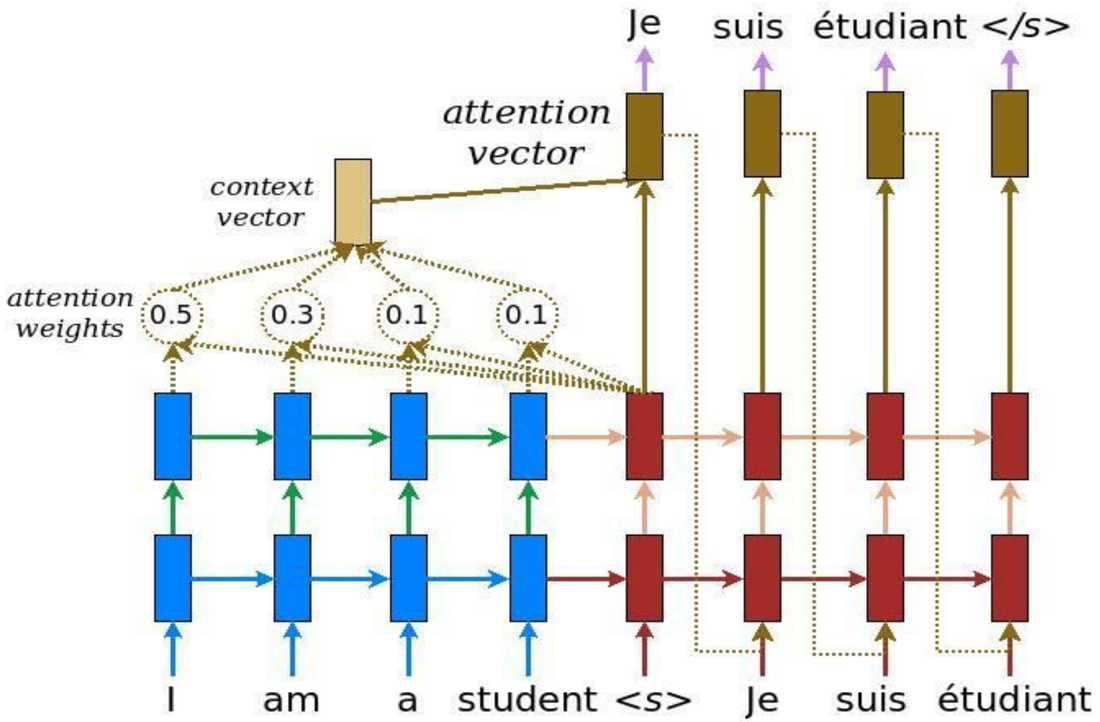
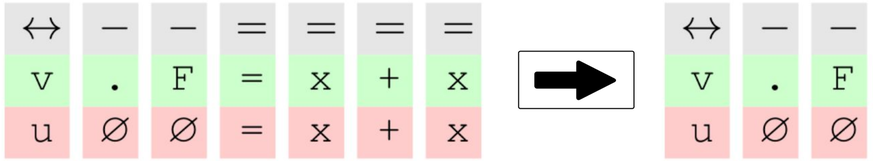


Схема машинного перевода на основе архитектуры Кодировщик-Декодировщик

https://www.tensorflow.org/images/seq2seq/attention_mechanism.jpg

Улучшение модели

1. Оставить в редакционной последовательности только измененные токены:
   - Позволит сделать векторные представления более обобщенными
   - 

2. Уменьшить размерность векторного представления изменений с 512 до 16
   - Позволит сделать векторные представления менее разреженными

Исправление ошибок в коде

Датасет (Tufano et al., An Empirical Study on Learning Bug-Fixing Patches in the Wild via Neural Machine Translation, TOSEM 2019): ~58 тыс. пар методов с ошибкой и без нее

Проблема: на тестовых данных невозможно построить изменение

- Решение 1:
  - Представить, что у нас есть код после изменения
- Решение 2 (388 классифицированных примеров, 64 класса):
  - Разметить данные и брать изменение из того же класса
- Решение 3:
  - Закодировать код с ошибкой
  - Найти ближайший к нему код с ошибкой из тренировочной выборки
  - Взять у найденного примера его изменение

Исправление ошибок в коде

Метрика: точность	Yin et al.	Tufano et al.	Мой подход
Решение 1	**84%**	-	62%
Решение 2	3%	7%	**10%**
Решение 3	11%	**15%** (9%)	**15%**

Выводы:

- Решение 1: **обе модели ожидаемо работают очень хорошо**
- Решение 2: **предложенный подход лучше обобщает векторные представления**

Исправление ошибок в коде

- Решение 3:
  - Точность жадного декодирования с использованием k ближайших векторов изменений

top-1	top-3	top-5	top-10	top-50
13%	18%	21%	25%	32%

  - Вывод: **среди ближайших существуют вектора изменений, позволяющие значительно повысить точность исправления ошибок в программном коде**

Генерация сообщений к коммитам

Датасеты:

1. Jiang et al., Automatically Generating Commit Messages from Diffs using Neural Machine Translation, ASE 2017: ~32 тыс. пар изменений и сообщений к ним
2. Его отфильтрованная версия:
   a. Изменения только в одном файле
   b. Нет удаления и добавления файлов (т.е. только “честные” изменения файлов)
   c. После фильтрации осталось 80% датасета
   d. После этого половина данных была отложена для предобучения

Генерация сообщений к коммитам

Результаты перекрестной проверки на 10 частях (метрика BLEU):

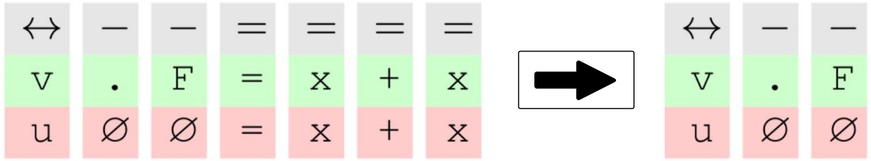
Датасет	Подход Jiang et al.	Подход Liu et al.	Мой подход
Jiang et al.	39.25 ± 0.96 (32.81)	40.47 ± 1.03 (39.01)	39.25 ± 0.99
После фильтрации	40.04 ± 1.24	41.86 ± 0.96	40.48 ± 1.01

Вывод: **обученные без учителя вектора изменений содержат достаточно информации для описания их на естественном языке**

Результаты

- Реализована модель для построения векторных представлений изменений с помощью обучения без учителя
- Предложены улучшения данной модели
  - https://github.com/JetBrains-Research/embeddings-for-code-diffs/tree/model_3
- Произведена апробация на задаче исправления ошибок в коде
  - Выявлен потенциал векторных представлений изменений значительно улучшить точность исправления ошибок до 32%
- Произведена апробация на задаче генерации сообщений к коммитам
  - Обучение без учителя достигает значения метрики BLEU в 40%, такого же, как и у других подходов

Улучшение модели

1. Оставить в редакционной последовательности только измененные токены:
   - Позволит сделать векторные представления более обобщенными
   - 

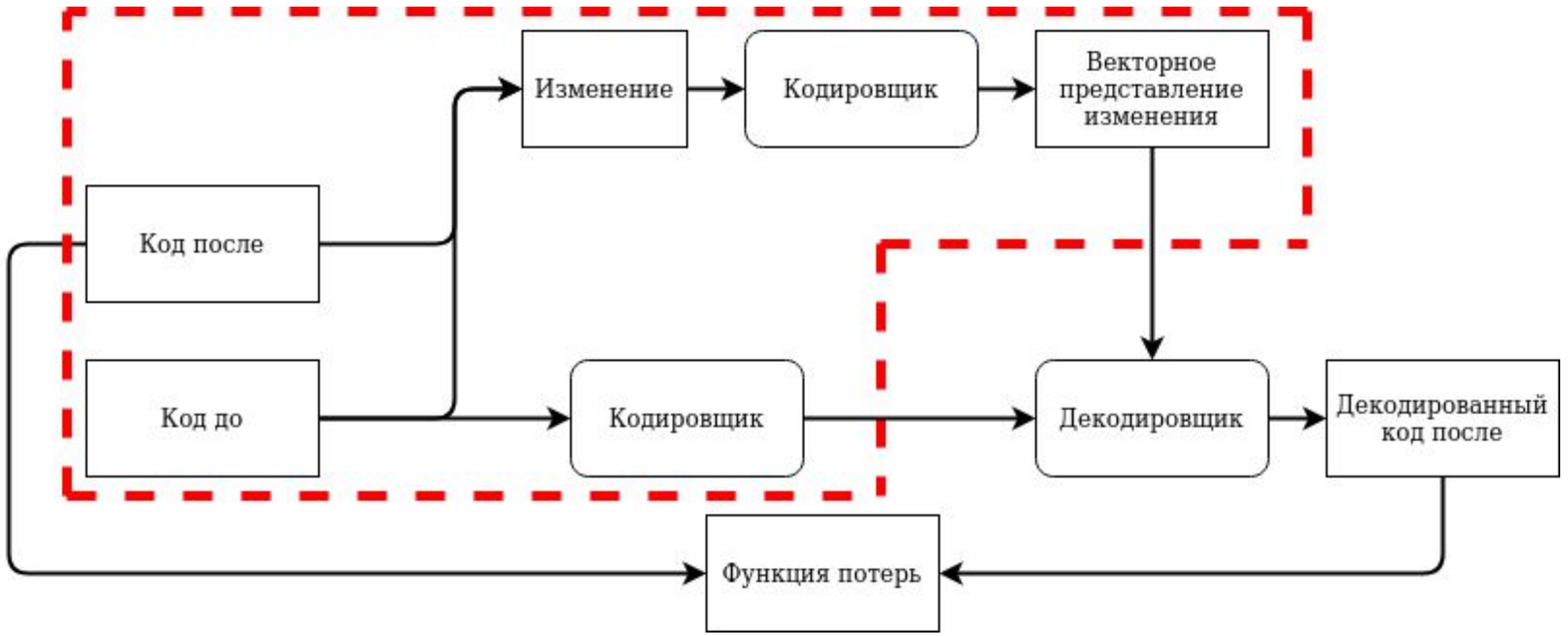
2. Уменьшить размерность векторного представления изменений с 512 до 16
   - Позволит сделать векторные представления менее разреженными
3. Использовать при обучении teacher forcing
   - Позволит сделать сходимость модели более быстрой и более устойчивой

Генерация сообщений к коммитам

Процесс обучения моей модели:

- Предобучение векторов изменений без учителя
- Обучение декодировщика на обученных векторах генерировать сообщения

Метод оценки: перекрёстная проверка на 10 частях, метрика: BLEU



Исправление ошибок в коде

(Размерность векторного представления изменения, тип редакционной последовательности)	Точность
(512, все токены)	2.8%
(16, все токены)	2.5%
(512, только измененные токены)	8.9%
(128, только измененные токены)	9.3%
(16, только измененные токены)	9.6%