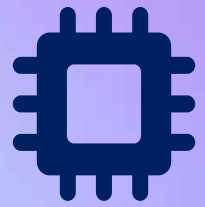
Исследование возможностей файн-тюнинга LLM для решения задачи повышения читабельности декомпилированного кода на языке Си

Команда и роли

Кислов Константин Александрович, С23-712
< Файн-тюнинг >

Божко Артем Александрович, С23-712
< Декомпиляция >



< Наставник – Бехтин Артем Владимирович, Б22-515 >

Ременяко Владислав Денисович, С23-712
< Сбор датасета >

Лялин Максим Андреевич, Б22-702
< Тестирование>

Введение

```
#include <windows.h>
#include <defs.h>


//-------------------------------------------------------------------------
// Function declarations

int printf(const char *const Format, ...);
static time_t __cdecl time(time_t *const Time);
int __fastcall main(int argc, const char **argv, const char **envp);
__int64 __fastcall _main(_QWORD, _QWORD, _QWORD); // weak
// void __cdecl srand(unsigned int Seed);
// int __cdecl rand();


//----- (0000000140001580) ----------------------------------------------------
int __fastcall main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  __int64 v5; // [rsp+20h] [rbp-30h]
  __int64 v6; // [rsp+28h] [rbp-28h]
  int v7; // [rsp+40h] [rbp-10h]
  unsigned int v8; // [rsp+44h] [rbp-Ch]
  unsigned int v9; // [rsp+48h] [rbp-8h]
  int i; // [rsp+4Ch] [rbp-4h]

  _main(argc, argv, envp);
  v3 = time(0i64);
  srand(v3);
  printf("Weather Simulation:\n\n");
  for ( i = 0; i <= 6; ++i )
  {
    v9 = rand() % 40;
    v8 = rand() % 100;
    v7 = rand() % 20;
    LODWORD(v6) = rand() % 100;
    LODWORD(v5) = v7;
    printf(
      "Day %d:\nTemperature: %d degC\nHumidity: %d%%\nWind Speed: %d m/s\nRainfall: %d mm\n\n",
      (unsigned int)(i + 1),
      v9,
      v8,
      v5,
      v6);
  }
  return 0;
}
// 1400016D4: variable 'v5' is possibly undefined
// 1400016D4: variable 'v6' is possibly undefined
// 140001780: using guessed type __int64 __fastcall _main(_QWORD, _QWORD, _QWORD);

// nfuncs=141 queued=1 decompiled=1 lumina nreq=0 worse=0 better=0
// ALL OK, 1 function(s) have been successfully decompiled
```

Пример вывода декомпилятора **Hex-Rays**

В большинстве случаев декомпилированный программный код трудно поддается **анализу**: названия переменных и функций лишены изначального заложенного смысла и трудно прослеживается логика программы. В ходе работы над проектом был обучен **адаптер** для языковой модели **CodeLlama**, предназначенный для улучшения декомпилированного кода на языке **Си**: приближения к исходному коду программы и **упрощения** для человеческого восприятия. Также исследованы возможности адаптера и проведена **оценка** его эффективности при решении данной задачи.

Исследовательская составляющая

Гипотеза: мощная предобученная LLM при относительно небольших затратах на ее тонкую настройку сможет продемонстрировать хорошие результаты при решении задачи повышения читабельности декомпилированного кода

Объект исследования - декомпиляция программного кода

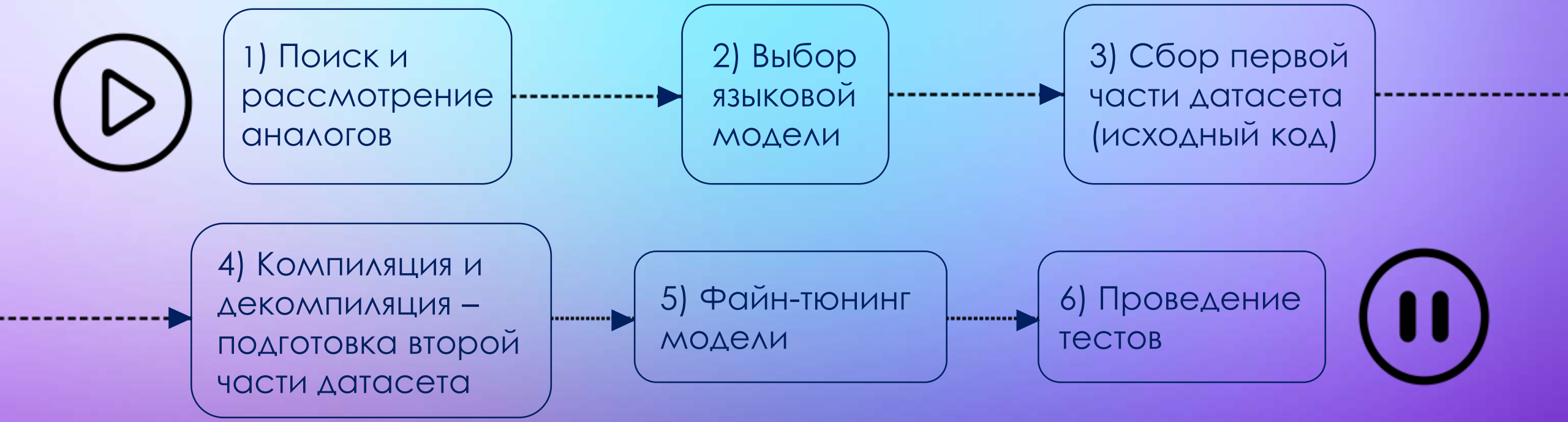
Предмет исследования – применение языковых моделей для анализа программного кода

Методы исследования: поиск, анализ, сравнение, моделирование, программирование, тестирование, измерение

Цель и задачи

Цель: разработка адаптера на основе предобученной языковой модели для решения задачи интеллектуальной обработки (в нашем случае – повышения читабельности) декомпилированного кода, а также оценка его работоспособности

Задачи:



Стек технологий

Основа датасета - FormAI Dataset (176 тыс. примеров из исходного кода на Си)

Декомпиляторы - Hex-Rays (8.3.0.230608), RetDec (v5.0)

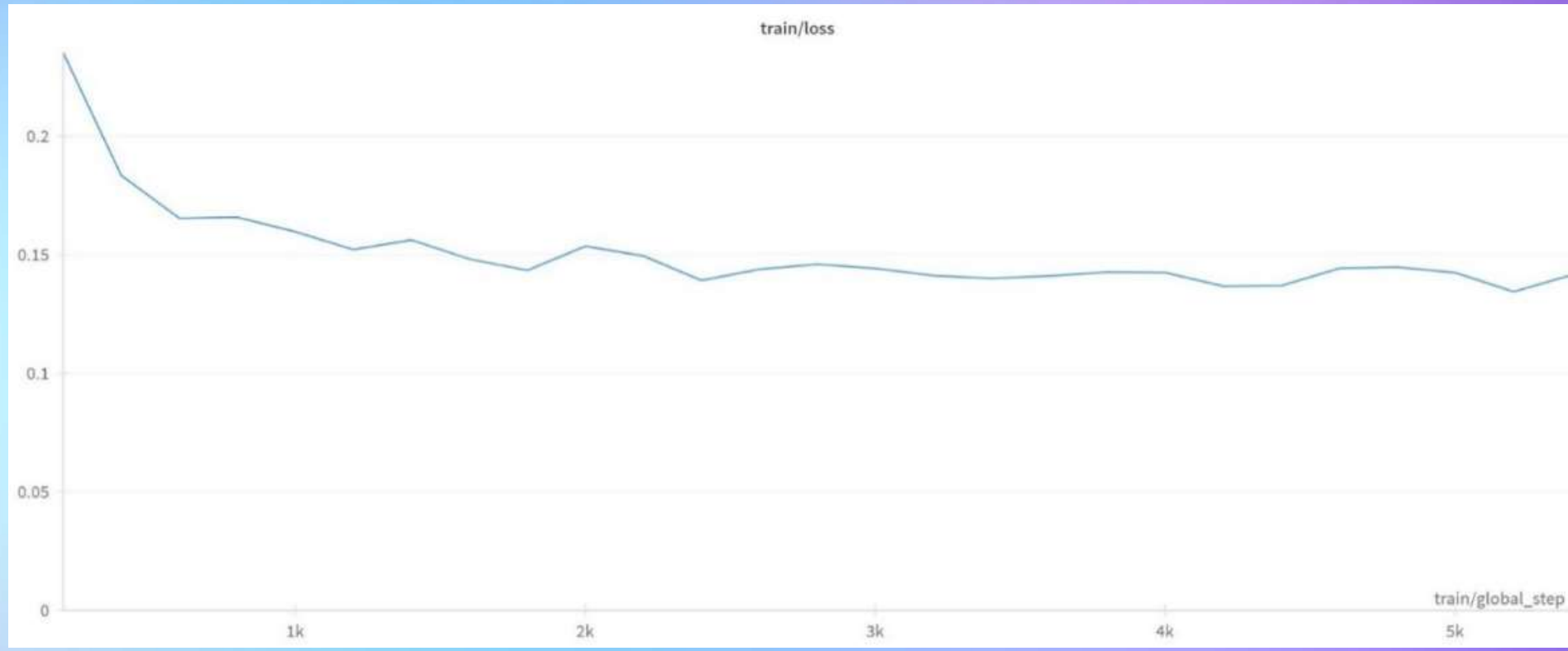
Адаптер
с 1,05 млн. обучаемых параметров (0,015 % от всех параметров модели)

Компилятор - GCC 11.4.0

Языковая модель - CodeLlama-7b (7 миллиардов обучаемых параметров)

Fine-tuning

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(32016, 4096)
    (layers): ModuleList(
      (0-31): 32 x LlamaDecoderLayer(
        (self_attn): LlamaSdpaAttention(
          (q_proj): lora.Linear4bit(
            (base_layer): Linear4bit(in_features=4096, out_features=4096, bias=False)
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=4096, out_features=1, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=1, out_features=4096, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (k_proj): lora.Linear4bit(
            (base_layer): Linear4bit(in_features=4096, out_features=4096, bias=False)
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=4096, out_features=1, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=1, out_features=4096, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (v_proj): lora.Linear4bit(
            (base_layer): Linear4bit(in_features=4096, out_features=4096, bias=False)
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=4096, out_features=1, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=1, out_features=4096, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (o_proj): lora.Linear4bit(
            (base_layer): Linear4bit(in_features=4096, out_features=4096, bias=False)
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.05, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=4096, out_features=1, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=1, out_features=4096, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (rotary_emb): LlamaRotaryEmbedding()
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear4bit(in_features=4096, out_features=11008, bias=False)
          (up_proj): Linear4bit(in_features=4096, out_features=11008, bias=False)
          (down_proj): Linear4bit(in_features=11008, out_features=4096, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm()
        (post_attention_layernorm): LlamaRMSNorm()
      )
    )
    (norm): LlamaRMSNorm()
  )
  (lm_head): Linear(in_features=4096, out_features=32016, bias=False)
)
```

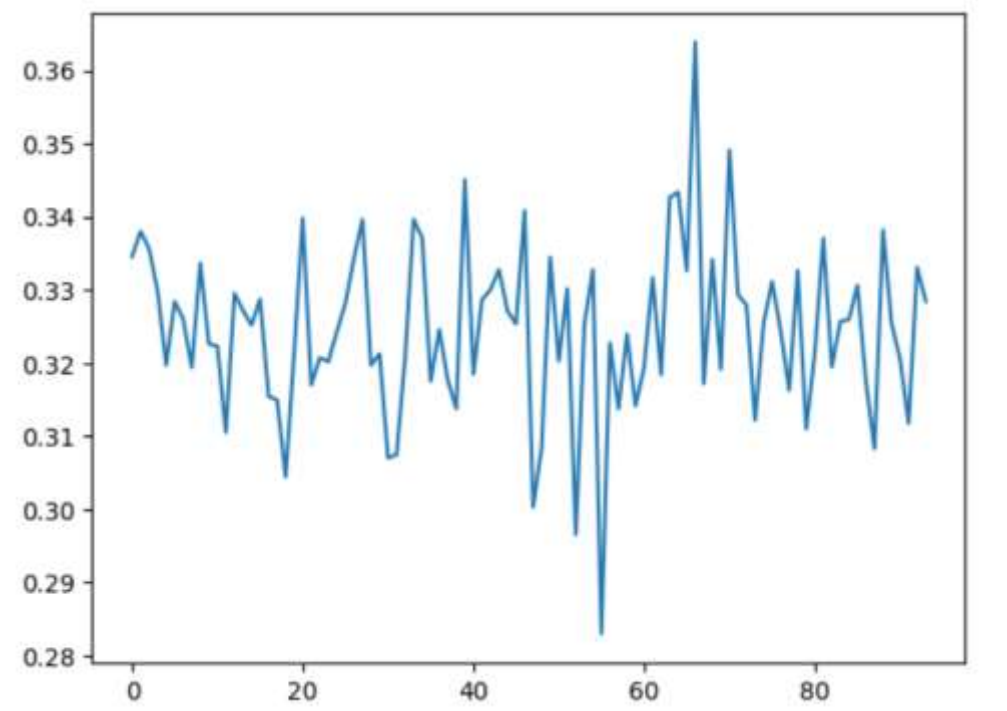


Демонстрация динамики функции потерь от количества итераций в ходе обучения

< - Архитектура LLM + адаптер

Последний лучший loss = **0.1495**

Тестирование



Динамика Sentence BLEU при тестировании

Corpus BLEU = 0.41
Sentence BLEU = 0.32

1) Оригинал
2) Сгенерированный моделью код

1

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int temperature, humidity, windSpeed, rainfall;
    srand(time(NULL)); // initialize random seed based on time
    printf("Weather Simulation:\n\n");
    for(int i = 0; i < 7; i++) { // simulate 7 days of weather
        temperature = rand() % 40; // generate random temperature between 0 and 39 degrees Celsius
        humidity = rand() % 100; // generate random humidity between 0% and 99%
        windSpeed = rand() % 20; // generate random wind speed between 0 and 19 meters per second
        rainfall = rand() % 100; // generate random rainfall between 0mm and 99mm
        printf("Day %d:\nTemperature: %d degC\nHumidity: %d%%\nWind Speed: %d m/s\nRainfall: %d mm\n\n", i+1, temperature, humidity, windSpeed, r
ainfall);
    }
    return 0;
}
```

2

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    srand(time(NULL)); // Seed the random number generator

    printf("Weather Simulation:

");

    for (int i = 0; i < 7; i++) { // Loop through each day of the week
        int temperature = rand() % 40; // Generate a random temperature between 0 and 39 degrees Celsius
        int humidity = rand() % 100; // Generate a random humidity between 0 and 99%
        int windSpeed = rand() % 20; // Generate a random wind speed between 0 and 19 m/s
        int rainfall = rand() % 100; // Generate a random rainfall between 0 and 99 mm

        printf("Day %d:
Temperature: %d degC
Humidity: %d%%
Wind Speed: %d m/s
Rainfall: %d mm

", i+1, temperature, humidity, windSpeed, rainfall);
    }

    return 0;
}
```

Заключение

Проведенные тесты показывают, что модель с адаптером, несмотря на относительно небольшие для NLP объем датасета и количество эпох, достигает приличных результатов, в том числе при обработке кода, декомпилированного при помощи программ, примеров вывода которых не было в обучающей выборке (например, RetDec). В дальнейшие планы работы над проектом входят: продолжение обучения модели на датасете большего объема и с примерами работы других декомпиляторов, проведение более масштабного тестирования как с классическими для задачи seq2seq метриками (BLEU, AED и т.д.), так и с оценкой при помощи опроса специалистов и с проверкой возможности перекомпилирования результатов работы нейросети.

< To be continued... >

II Весенняя научная сессия СНО НИЯУ МИФИ — 2024

Исследование возможностей файн-тюнинга LLM для решения задачи повышения читабельности декомпилированного кода на языке Си

Наш GitHub - >

< - Адаптер на Hugging Face