ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO CUỐI KÌ GR1

Đề tài: Tóm tắt nội dung đoạn code được viết bằng ngôn ngữ Ruby

Sinh viên	:	Nguyễn Đình Hiếu
Mã số sinh viên	:	20215049
Giáo viên hướng dẫn	:	PGS.TS. Lê Thanh Hương
Bộ môn	:	Nghiên cứu tốt nghiệp 1
Viện	:	Trường Công nghệ Thông tin và Truyền thông

Hà Nội, 7-2024

Lời cảm ơn

Em xin bày tỏ lòng biết ơn sâu sắc đến Trường Công nghệ Thông tin và Truyền thông, đã tạo điều kiện cho em được học môn học này. Em đã có cơ hội tiếp xúc với cách làm việc với một mô hình AI đặc biệt là LLM, điều mà không có môn nào trong chương trình học của em dạy về điều đó cả.

Đặc biệt, em xin gửi lời cảm ơn chân thành tới cô Lê Thanh Hương, người đã tận tình hướng dẫn, động viên và hỗ trợ em trong suốt quá trình thực hiện đồ án. Nhờ những sự chỉ dạy của cô mà em đã có những kiến thức cơ bản về NLP, cách đọc paper, cách research và cách giải quyết bài toán bằng mô hình ngôn ngữ lớn. Nhờ những sự chỉ bảo tận tình của cô đã giúp em vượt qua những giai đoạn khó khăn nhất.

Em xin chân thành cảm ơn.

Mục lục

Chương 1

Đề tài thực hiện

1.1 Mô tả đề tài

Trong quá trình học IT của bản thân và cũng như những người bạn của em luôn có một hoạt động là tham khảo code từ những người giỏi hơn, từ những repo trên github. Điều này giúp em có thêm nhiều kiến thức và kĩ năng hơn trong việc viết code. Tuy nhiên trong quá trình đó em gặp khá nhiều khó khăn trong việc đọc hiểu code, ý nghĩa của các tham số truyền vào cũng như đầu ra như thế nào. Bởi vậy nên em mới chọn đề tài này với mong muốn giải quyết một bài toán thực tế đang còn tồn đọng khá nhiều đặc biệt với sinh viên IT như chúng em.

1.2 Dữ liệu sử dụng

Em sử dụng bộ dữ liệu mã nguồn mở CodeSearchNet (Husain et al. (2019)). Đây là bộ dữ liệu được lấy từ Github với 6 ngôn ngữ Go, Java, JavaScript, PHP, Python, và Ruby. Đối với mỗi ngôn ngữ dữ liệu đều chia thành train, valid và test. Mỗi điểm dữ liệu bao gồm code (Dạng function) và docstring đi kèm của nó. Dưới đây là mô tả về số lượng dữ liệu

Ngôn ngữ	Function
Go	726768
Java	1569889
JavaScript	1857835
PHP	977821
Python	1156085
Ruby	164048
All	6432446

Bảng 1.2.1: Số lượng hàm

Số lượng các bộ dữ liệu của ngôn ngữ Ruby như sau

Bộ dữ liệu	Số lượng dữ liệu
Train	48791
Valid	2209
Test	2279

Bảng 1.2.2: Số lượng hàm

Mỗi điểm dữ liệu bao gồm các trường:

- **repo**: Tên repo trên github chứa đoạn code.
- **path**: Đường dẫn đến file chứa đoạn code trong repo đó.
- **func_name**: Tên hàm.
- **original_string**: Đoạn text gốc (Bao gồm cả text và document) chưa qua tokenize.
- **language**: Ngôn ngữ được sử dụng.
- **code**: Phần code gốc chưa qua tokenize.
- **code_tokens**: Phần code đã qua tokenize.
- **docstring**: Phần comment top-level ở đầu mỗi hàm.
- **docstring_tokens**: Phần docstring đã tokenized.
- **partition**: Một trong 3 giá trị ”train”, ”test”và ”valid”tương ứng với bộ dữ liệu thuộc về.
- **url**: đường dẫn url trên github chỉ ra vị trí đoạn code trong 1 file.

Ví dụ một điểm dữ liệu của ngôn ngữ Ruby như sau:


```
{
  "repo": "rails/rails",
  "path": "activesupport/lib/active_support/current_attributes.rb",
  "func_name": "ActiveSupport.CurrentAttributes.set",
  "original_string": "def set(set_attributes)\n      old_attributes
  ↪  = compute_attributes(set_attributes.keys)\n
  ↪  assign_attributes(set_attributes)\n      yield\n    ensure\n
  ↪  assign_attributes(old_attributes)\n    end",
  "language": "ruby",
```

```
"code": "def set(set_attributes)\n      old_attributes =
 ↪  compute_attributes(set_attributes.keys)\n
 ↪  assign_attributes(set_attributes)\n      yield\n    ensure\n
 ↪  assign_attributes(old_attributes)\n    end \end{tabular}}",
"code_tokens": [
  "def",
  "set",
  "(",
  "set_attributes",
  ")",
  "old_attributes",
  "=",
  "compute_attributes",
  "(",
  "set_attributes",
  ".",
  "keys",
  ")",
  "assign_attributes",
  "(",
  "set_attributes",
  ")",
  "yield",
  "ensure",
  "assign_attributes",
  "(",
  "old_attributes",
  ")",
  "end"
],
```

```
"docstring": "Expose one or more attributes within a block. Old
 ↪  values are returned after the block concludes.\n Example
 ↪  demonstrating the common use of needing to set Current
 ↪  attributes outside the request-cycle:\n\n   class
 ↪  Chat::PublicationJob < ApplicationJob\n     def
 ↪  perform(attributes, room_number, creator)\n
 ↪  Current.set(person: creator) do\n
 ↪  Chat::Publisher.publish(attributes: attributes, room_number:
 ↪  room_number)\n       end\n     end\n   end",
"docstring_tokens": [
  "Expose",
  "one",
  "or",
  "more",
  "attributes",
  "within",
  "a",
  "block",
  ".",
  "Old",
  "values",
  "are",
  "returned",
  "after",
  "the",
  "block",
  "concludes",
  ".",
  "Example",
  "demonstrating",
  "the",
  "common",
  "use",
  "of",
  "needing",
  "to",
  "set",
```

```
    "Current",
    "attributes",
    "outside",
    "the",
    "request",
    "-",
    "cycle",
    ":"
  ],
  "sha": "85a8bc644be69908f05740a5886ec19cd3679df5",
  "url":
  ↪  "https://github.com/rails/rails/blob/85a8bc644be69908f05740a5886ec19cd3679d
  "partition": "train"
}
```


Với những khám phá thêm về bộ dữ liệu cô có thể tham khảo thêm ở đây

1.3 Phạm vi bài toán

Em vốn định giải bài toán sinh docstring (bao gồm mô tả về nội dung/chức năng của đoạn code kèm theo mô tả về các tham số đầu vào, đầu ra) tuy nhiên vì độ khó cũng như thời gian của em hơi hạn chế ở học kì này (em đăng ký 24 tín kì này để các kì sau có thể tập trung hơn vào đồ án) nên em sẽ làm bài toán sinh mô tả nội dung/chức năng của đoạn code thôi còn phần mô tả đi kèm em sẽ cố gắng nghiên cứu thêm vào thời gian nghỉ hè cũng như các kì đồ án sau.

Chương 2

Phương pháp giải quyết

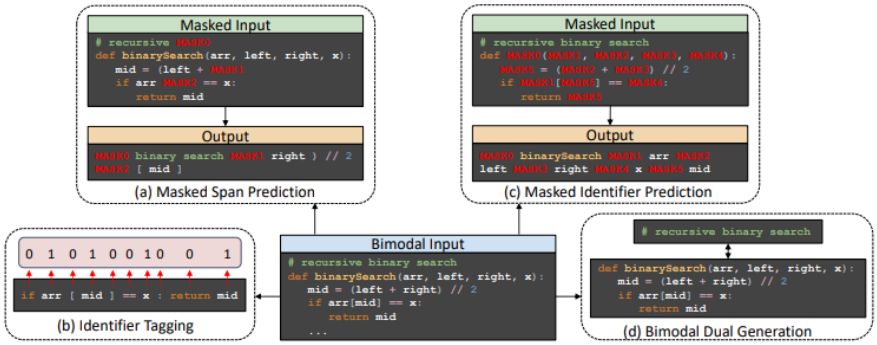
Với bài toán Code Summarization dựa trên bộ dữ liệu CodeSearchNet, em có tham khảo leaderboard của PaperWithCode thì mô hình codeT5 Yue Wang (2005) đang đạt SOTA ở bài toán này. Thang đánh giá được dùng ở đây là smoothed BLEU-4.

Rank	Model	Go	Java	Javascript	PHP	Python	Ruby	Paper	Code	Result	Year	Tags
1	**CodeT5**	19.76	20.46	16.24	26.53	20.36	15.73	CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation			2021	
2	**CodeT5+ 220M**	19.60	20.53	16.27	26.78	20.16	15.51	CodeT5+: Open Code Large Language Models for Code Understanding and Generation			2023	
3	**CodeBERT**	19.06	17.65	14.9	25.16	18.07	12.16	CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation			2021	
4	**Redcoder**		22.94			21.01		Retrieval Augmented Code Generation and Summarization			2021	
5	**CodeT5+ 770M**		20.83	17.93	26.39	20.47	15.63	CodeT5+: Open Code Large Language Models for Code Understanding and Generation			2023	

Hình 2.0.1: Bảng đánh giá các model với bài toán code Summarization.

Do phần lớn các đoạn tóm tắt code khi sinh ra rất là ngắn vậy nên điểm khá là thấp, loanh quanh chỉ là 20 nhưng vẫn ngữ nghĩa sinh ra vẫn diễn tả đầy đủ nội dung và chức năng của đoạn code. Nếu chúng ta muốn cải thiện điểm có thể chuyển sang bài toán sinh docstring, khi đó đoạn văn bản sẽ dài hơn và có nhiều thứ để đánh giá hơn.

Mô hình codeT5 trong quá trình pretrain đã mask những identifier (là các tên biến, tên hàm) và họ cho rằng việc dự đoán những identifier này sẽ giúp mô hình hiểu rõ hơn về code.



Hình 2.0.2: Pretrain-task của codeT5, như ở ý c, các identifier là midd, arr, x là tên các biến.

Từ đó em có hai hướng giải quyết dựa trên pretrain model của họ là

- Fine tune lại trên bộ data cho code ruby.
- Pretrain khác là mask một vài từ trong document và dự đoán các từ này để model có thể hiểu rõ hơn về code.

Chương 3

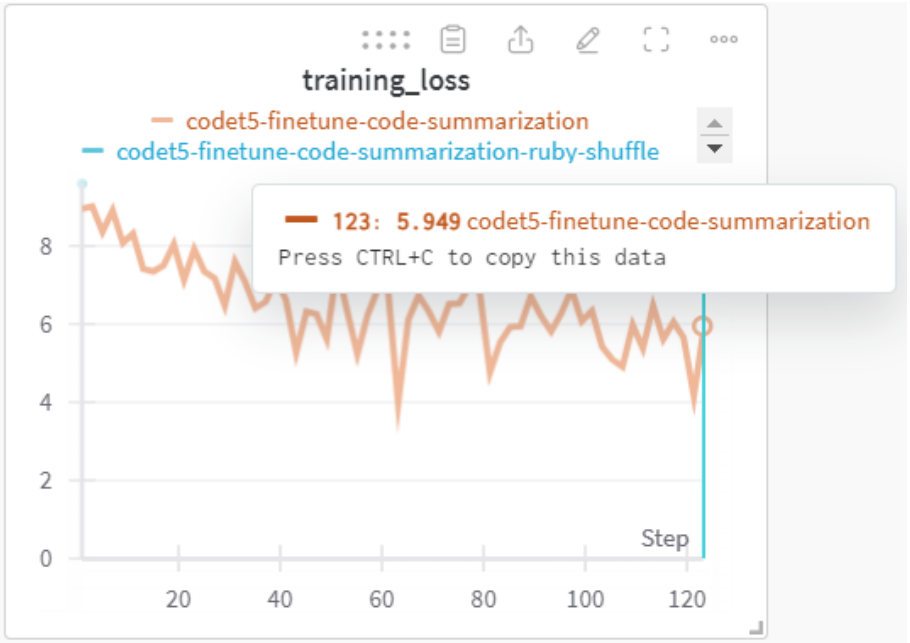
Các thí nghiệm và kết quả đạt được

3.1 Fine-tune model

Em đã thực hiện fine tune trên mô hình codeT5 small 60.5M

```
{
    "learning rate": 5e-5,
    "train epochs": 15,
    "warmup steps": 1000,
    "batch size": 8
}
```

Sau khi train qua 37 tiếng, em thu được loss giảm khá nhiều tuy nhiên vẫn khá là cao



Hình 3.1.1: Loss cuối cùng thu được vẫn khá là cao

Và kết quả dự đoán ở một số test case có cấu trúc rõ ràng thì kết quả thu được khá tốt

```
Code: def ensure_org(organization, members = true)
      org = db[:users].first(:login => organization, :type => 'org')

      if org.nil?
        org = ensure_user(organization, false, false)

        # Not an organization, don't go ahead
        if org[:type] != 'ORG'
          warn "User #{organization} is not an organization"
          return nil
        end
      end
      if members
        retrieve_org_members(organization).map do |x|
          ensure_participation(ensure_user(x['login'], false, false)[:login],
                               organization, false)
        end
      end
      org
    end
Ground truth: Make sure that an organization exists

 ==Parameters:
 [organization]  The login name of the organization
Generated docstring: Ensure the user is an organization.

 @param [String] organization
 @param [Boolean] members
 @return [Array<User>]
```

Hình 3.1.2: Kết quả thu được thậm chí còn mô tả thêm về các tham số

Tuy nhiên cũng có một số số test case mà cấu trúc chưa rõ ràng thì kết quả thu được

chưa được tốt

```
Code: def find_bad_files_from_kubectl_output(line)
      # stderr often contains one or more lines like the following, from which we can extract the file path(s):
      # Error from server (TypeOfError): error when creating "/path/to/service-gqq5oh.yml": Service "web" is invalid:

      line.scan(%r{"(/\S+\.ya?ml\S*)"}).each_with_object([]) do |matches, bad_files|
        matches.each do |path|
          content = File.read(path) if File.file?(path)
          bad_files << { filename: File.basename(path), err: line, content: content }
        end
      end
    end
Ground truth: Inspect the file referenced in the kubectl stderr
 to make it easier for developer to understand what's going on
Generated docstring: Find bad files from the kubeadmectl output
```

Hình 3.1.3: Kết quả thu được khác xa so với ban đầu

Và kết quả tổng quát đánh giá trên tập test với thang smoothed BLEU-4 vẫn khá là thấp

```
!python evaluator.py reference.txt < predictions.txt

Total: 1261
9.600254059282024
```

Hình 3.1.4: Kết quả thu được khác xa so với ban đầu

3.2 Pre-train model

Với phương pháp này em có thử mask các từ trong top level comment với xác suất 0.15 và dự đán chúng. Sau đó tiếp tục fine tune trên bộ dữ liệu code Ruby. Tuy nhiên, kết quả thu được còn kém hơn cả không pre-train vậy nên em sẽ không trình bày về phương pháp này.

Chương 4

Kết luận và hướng phát triển

Mặc dù kết quả của mô hình đem lại chưa thực sự tốt nhưng cũng đã phần nào giải quyết được bài toán đã đề ra ban đầu. Trong thời gian rảnh của kì hè và trong các lần đồ án môn học sau, em dự định sẽ thử tiếp với những cách tiếp cận mà cô đã đưa ra và một số cách dự kiến của em:

- Can thiệp vào các phần block của code để hiểu về luồng hoạt động hơn thay vì chỉ tác động vào phần comment.
- Thử các cách instruction tuning.
- Nghiên cứu bản nâng cấp của codeT5 chính là codeT5+ với một số cải thiện đáng kể về cách pretrain, hàm loss, data
- Thử nghiệm với các bộ data khác như HumanEval hoặc với bộ dữ liệu CodeSearchNet nhưng với ngôn ngữ khác
- Tìm hiểu mô hình CodeBERT, CodeTrans-MT-Base cho bài toán sinh code document.

Trong giai đoạn cuối kì, do sự dồn dập của các bài tập lớn của môn học và sức nặng kiến thức của 24 tín học phần nên em chưa thực sự chuyên tâm vào môn học, em mong cô thông cảm cho em. Vào kì sau, lượng học phần của lại của em đã giảm bớt nhiều, em xin hứa em sẽ tập trung hơn nữa vào đồ án để không phụng sự kì vọng của cô.

Em xin chân thành cảm ơn cô.

Tài liệu tham khảo

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv:1909.09436*, 2019.

Shafiq Joty Steven C.H. Hoi Yue Wang, Weishi Wang. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *EMNLP*, 2005.