

Automatic Semantic Augmentation of Language Model Prompts (for Code Summarization)

Toufique Ahmed
University of California, Davis
Davis, California, USA
tfahmed@ucdavis.edu

Kunal Suresh Pai
University of California, Davis
Davis, California, USA
kunpai@ucdavis.edu

Premkumar Devanbu
University of California, Davis
Davis, California, USA
ptdevanbu@ucdavis.edu

Earl T. Barr
University College London & Google Brain
London, UK
e.barr@ucl.ac.uk

ICSE '24, April 14–20, 2024, Lisbon, Portugal

背景

コード要約タスク

ソースコードから関数の機能などを説明する文章を生成

例

この関数は与えられた整数配列から最大値を見つけて返します。

空の配列が渡された場合は`null`を返します。

配列内の要素を順に走査し、現在の最大値と比較しながら更新していきます。

利点

- ・コード理解の支援
- ・保守が容易になる

コード要約手法の遷移

テンプレートベース

コードの多様性を網羅することが難しい

情報検索ベース

コードと要約がペアになっているデータベースから入力コードに似たものを検索する

事前学習言語モデルの利用

CodeBERT, CodeT5などのモデルがコード要約で最高性能を示す

LLMベース

多くのタスクでCodeBERTといったモデルを上回るようになってきた

この研究で用いる手法

この研究ではLLMベースの手法を提案する

(理由)

学習のための大規模なデータセットを生成するのは困難で時間がかかる

高価なハードウェアが必要なくコストが低い

Few-shot学習によってデータが少量であっても効果的にタスクを実行できる

中間ステップの明示

“Chain of thought”や“step by step”といったアプローチでは中間結果を生成してタスクに活用することを要求している

コード解析ツールを利用し，中間結果をプロンプトに明示する

→開発者がコードを理解するときに考慮する内容を示せる

目的

- | プロンプトに意味的情報を自動追加する手法の提案
- | この手法のコード要約タスクにおける有効性の評価

提案手法

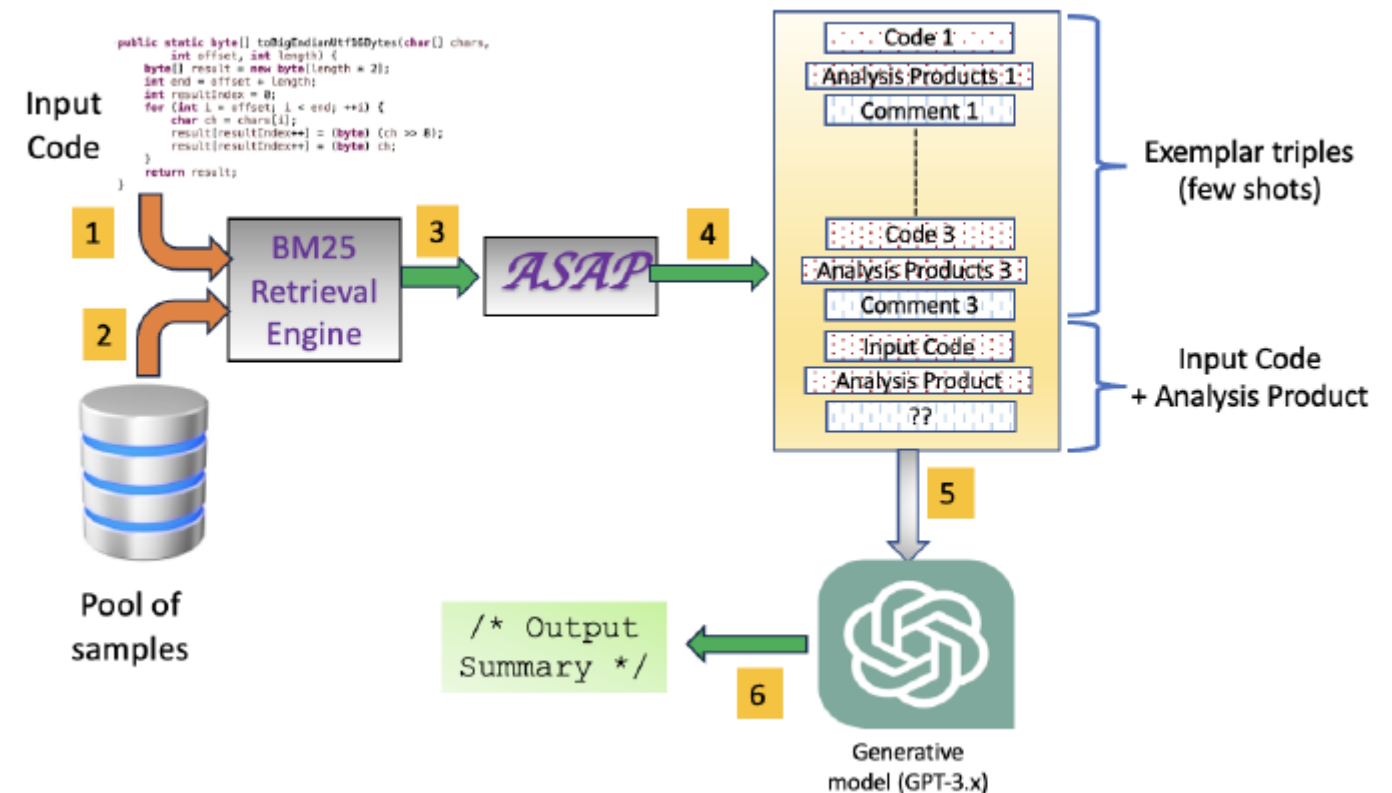
Automatic Semantic Augmentation of Prompts(ASAP)

(1~3)

入力コードと類似する 3 つのコードと要約のペアを取得

(4)

意味的情報の抽出



3 組のコード，意味的情報，要約のセット，
入力コードとその意味的情報をプロンプトに含めて要約生成

抽出する意味的情報

リポジトリ名とパス

リポジトリ名やリポジトリへの完全なパスが追加の情報を提供する

識別子のASTタグ

コード内の識別子（変数名、クラス名など）に抽象構文木内での役割を表すタグをつける

データフローグラフ

これを使用するGraphcodeBERTモデルがSEタスクで優れた性能を発揮したため、これを組みこむ

これらが開発者がコードを理解する際に注目する情報としている

意味的情報の抽出

リポジトリ名とパス

GitHubから抽出する

識別子のASTタグ

TreeSitterでASTをたどる

We categorized the identifiers into different classes.

Please find the information below.

Function name: OffsetUtil.deserializeOffsetMap

Parameters of the function: ['lastSourceOffset']

Identifier to be returned: ['offsetMap']

Method Invocation: ['lastSourceOffset', 'isEmpty', 'JSON_MAPPER', 'readValue']

Method Arguments: ['lastSourceOffset']

Variable declaration: ['offsetMap']

One Exemplar

Source Code

Repo Info

DataFlow Info

Variable Names, scopes

Output Comment

```
@SuppressWarnings("unchecked")
public static Map < String, String >
    deserializeOffsetMap(String lastSourceOffset) throws IOException {
    Map < String, String > offsetMap;
    if (lastSourceOffset == null || lastSourceOffset.isEmpty()) {
        offsetMap = new HashMap < > ();
    } else {
        offsetMap = JSON_MAPPER.readValue(lastSourceOffset, Map.class);
    }
    return offsetMap;
}
```

Please find some info about the location of the function in the repo.
Repo: streamsets/datacollector
Path: stagesupport/src/main/java/com/.../OffsetUtil.java
Function_name: OffsetUtil.deserializeOffsetMap

Please find the dataflow of the function. We present the source and list of target indices.
String-[9] [11, 15, 23, 25]
Map-[7] [21, 62]
lastSourceOffset-[16] [31, 35, 60]
offsetMap-[27] [42, 54, 69]

We categorized the identifiers into different classes. Please find the information below.
Function name: OffsetUtil.deserializeOffsetMap
Parameters of the function: ['lastSourceOffset']
Identifier to be returned: ['offsetMap']
Method Invocation: ['lastSourceOffset', 'isEmpty', 'JSON_MAPPER', 'readValue']
Method Arguments: ['lastSourceOffset']
Variable declaration: ['offsetMap']

"Deserialize String offset to Map of table to offset"

GitHub



TreeSitter

データフローグラフ

ASTを辿り， データが生成・使用される箇所を特定

Please find the dataflow of the function.

We present the source and list of target indices.

String-[9] [11, 15, 23, 25]

Map-[7] [21, 62]

lastSourceOffset-[16] [31, 35, 60]

offsetMap-[27] [42, 54, 69]

データセット・メトリクス・使用モデル

CodeSerchNetデータセット

メトリクス

- BLEU-CN
- ROUGE-L
- METEOR

使用モデル

OpenAIのcode-davinci-002

例示に使うデータ

トレーニングデータ

- ・ここから入力に似たコードを探す
- ・時間的に早く開発されたコード

Language	#of Training Samples	#of Test Samples
Java	164,923	1000
Python	251,820	1000
Ruby	24,927	1000
JavaScript	58,025	1000
Go	167,288	1000
PHP	241,241	1000

→例示に使用できるのは既に存在しているものであり，現実的

テストデータ

- ・これに似たコードをトレーニングデータから探し，例示とする
- ・各言語から均等にランダムに1000サンプルを選択

結果

各意味的情報の貢献（BLEUスコア）

Language	BM25	BM25+repo	BM25+id	BM25+DFG	$\mathcal{ASAP}$	Comparing with BM25	
						Gain (%) over BM25	p-value
Java	22.87	25.23	23.39	23.13	25.41	+11.11%	<0.01
Python	21.78	24.22	22.54	21.82	24.26	+11.39%	<0.01
Ruby	17.21	19.67	19.19	17.55	19.62	+14.00%	<0.01
JavaScript	23.27	25.11	24.21	24.04	25.36	+8.98%	<0.01
Go	22.67	24.41	23.2	23.42	24.51	+8.12%	<0.01
PHP	28.15	32.07	29.8	28.92	32.73	+16.27%	<0.01
Overall	22.66	25.12	23.72	23.15	25.32	+11.74%	<0.01

3つの意味的情報を考慮することで改善がみられる

結果

モデル間での比較

Language	Model	BM25	$\mathcal{ASAP}$	Gain	p-value
Java	Code-davinci-002	23.90	25.78	+7.87%	<0.01
	Text-davinci-003	18.98	22.31	+17.54%	<0.01
	Turbo-GPT-3.5	16.68	16.96	+1.68%	0.95
Python	Code-davinci-002	22.00	24.78	+12.64%	<0.01
	Text-davinci-003	16.74	18.93	+13.08%	<0.01
	Turbo-GPT-3.5	15.01	16.38	+9.13%	<0.01
PHP	Code-davinci-002	28.42	33.52	+17.95%	<0.01
	Text-davinci-003	21.67	25.72	+18.69%	<0.01
	Turbo-GPT-3.5	18.48	19.99	+8.17%	<0.01

Gpt-3.5-turboが劣っている

プロンプト設計を変えればより自然な要約が出来るかもしれない

Code-davinci-002が最も高いスコアを示した

結果

コード補完タスクへの適用

EM指標：参照テキストと完全に一致する割合

ES指標：参照テキストとの類似性を測定

Language	# of Samples	Exact Match (EM)				Edit Similarity (ES)			
		Zero-shot	$\mathcal{ASAP}$ (Zero-shot)	Gain (%)	p-value	Zero-shot	$\mathcal{ASAP}$ (Zero-shot)	Gain (%)	p-value
Java	9292	20.75	22.12	+6.6%	<0.01	55.35	59.66	+7.79%	<0.01
Python	6550	14.05	14.58	+3.77%	0.13	49.71	50.12	+0.82%	<0.01
Overall	15842	17.97	19.01	+5.79%	<0.01	53.01	55.72	+5.11%	<0.01

意味的情報を加えることで改善が見られた

結果

他の指標

Language	BLEU-DC				ROUGE-L				METEOR			
	BM25	ASAP	Gain (%)	p-value	BM25	ASAP	Gain (%)	p-value	BM25	ASAP	Gain (%)	p-value
Java	14.09	15.94	+13.13%	<0.01	36.85	38.41	+4.23%	<0.01	35.66	36.10	+1.23%	0.32
Python	12.63	14.49	+14.73%	<0.01	35.32	37.74	+6.85%	<0.01	33.05	35.63	+7.81%	<0.01
Ruby	9.16	11.01	+20.2%	<0.01	28.19	30.55	+8.37%	<0.01	27.65	29.20	+5.61%	0.03
JavaScript	14.89	16.71	+12.22%	<0.01	32.28	33.88	+4.96%	<0.01	32.08	33.02	+2.93%	0.15
Go	17.10	18.57	+8.60%	<0.01	41.04	42.43	+3.39%	<0.01	36.78	37.26	+1.31%	0.27
PHP	16.97	20.63	+21.57%	<0.01	40.48	44.90	+10.92%	<0.01	40.14	43.35	+8.00%	<0.01
Overall	14.14	16.23	+14.78%	<0.01	35.69	37.99	+6.44%	<0.01	34.23	35.76	+4.47%	<0.01

すべての指標で改善が見られた

結論

- | プロンプトに意味的情報を追加することで
コード要約タスクにおいてLLMの性能を向上させられた
- | 3つのモデルで提案手法による性能の改善が確認できた
- | コード補完タスクにおいても性能を向上させられた

