

How Effective are Large Language Models in Generating Software Specifications?

文献

背景

ソフトウェア仕様(Software Specifications)

- プログラムがどのような動作をするのか明記した設計書.
⇒プログラムのバグを見つけたり, テストを作ったりするのにとても有効.
- 開発者がコード内にコメント, ドキュメントとして保存.
- 自然言語で書かれている.

テスト生成やバグ検出を自動化させる

- 機械が読める形式に変換しなければならない

問題

自然言語から機械が読める仕様に変換する研究は存在.

問題

- 特定の分野でしか使えない.
 - 人がルールを1つずつ作らないといけない.
- ⇒汎用性が低い.

LLMの登場

- 自然言語とコードの両方を理解できる.
- ⇒コメントから自動で「仕様」を作れる可能性がある.

目的

- | RQ1 : Few-shot Learningを用いたLLMは, 従来のルールベース手法と比べてどの程度性能があるか?
- | RQ2 : プロンプトの作り方は, LLMの性能にどのような影響を与えるか?
- | RQ3 : LLMと従来手法には, どのような強みと弱みがあるか?
- | RQ4 : 異なるLLM同士を性能とコストの観点で比べるとどうなるか?

既存の手法データセットと評価指標

データセット

- Jdoctor-data(Jdoctor手法)
 - 関数の入力・出力・例外などの条件を抽出する.
- DocTer-data(DocTer手法)
 - パラメータに関する制約を抽出する（例：型や値の制約など）
- CallMeMaybe-data(CallMeMaybe手法)
 - 関数呼び出しの順序関係を抽出

評価指標

- 正答率：正しく生成された仕様の数/仕様の総数
 - Jdoctor, CallMeMaybe
- F1値
 - DocTer

LLMを用いた仕様生成

Few-shotを用いて, LLMに学習させる.

- CodeLlama-13B

プロンプトの作り方

- ランダム選択
 - データセットの中からランダムにK個選ぶ.
- 意味的類似度選択
 - RoBERTa-largeを用いて, 意味類似度を測定.

出力結果の後処理

- 人間が目視で正しい移民お出力になっているか確認.
 - 意味的には同じでも文字列等が違う可能性があるため.

RQ4：使用LLM

ベンチマークLLM(RQ1~3で使用)

- CodeLlama-13B

比較LLM

- 最新の汎用モデル・コード特化モデル
オープンソースモデル・商用モデル
- 15B未満のモデル

Model	#Param	Token limit	Price (per 1K)	Open-source?
GPT4 [46]	Unknown	8,192	\$0.03	✗
GPT3.5 [47]	Unknown	16,384	\$0.0015	✗
CodeLlama [48]	13B, 7B	16,384	-	✓
Llama3 [49]	8B	8,192	-	✓
Llama2 [50]	13B, 7B	4,096	-	✓
deepseek-coder [51]	6.7B	16,384	-	✓
StarCoder2 [52]	15B, 7B	16,384	-	✓
StarCoder [53]	15.5B	8,192	-	✓
CodeGen2 [54]	16B, 7B	2,048	-	✓

RQ1:従来の手法と比較

Jdoctor

Approach	K	@param		@return		@throws		Overall	
		Raw	Processed	Raw	Processed	Raw	Processed	Raw	Processed
Jdoctor	-	97.0	97.0	69.0	69.0	79.0	79.0	83.0	83.0
CodeLlama-13B	10	81.1	89.7	35.3	48.9	76.3	87.3	71.0	81.7
CodeLlama-13B	20	84.4	92.6	39.6	55.4	73.9	88.0	71.3	84.0
CodeLlama-13B	40	92.2	94.2	48.2	61.2	79.9	90.3	78.2	86.7
CodeLlama-13B	60	91.8	94.7	48.9	56.8	83.3	92.4	80.1	87.3

DecTer

Approach	K	TensorFlow	PyTorch	MXNet	Overall
DocTer	-	90.0/74.8/81.7	78.4/77.4/77.9	87.9/82.4/85.1	85.4/78.2/81.6
CodeLlama-13B	10	68.3/74.3/71.2	72.9/69.1/70.9	66.4/71.8/69.0	68.2/72.2/70.1
CodeLlama-13B	20	77.9/72.5/75.1	76.6/72.2/74.3	71.7/69.7/70.7	74.7/71.1/72.8
CodeLlama-13B	40	77.8/80.4/79.1	78.7/72.7/75.5	75.3/75.2/75.2	76.7/76.6/76.6
CodeLlama-13B	60	80.9/79.8/80.4	79.6/76.4/77.9	78.3/80.4/79.4	79.4/79.5/79.5

CallMeMaybe

Approach	K	Accuracy
CallMeMaybe	-	70.0
CodeLlama-13B	10	48.3
CodeLlama-13B	20	57.3
CodeLlama-13B	40	62.9
CodeLlama-13B	60	70.8

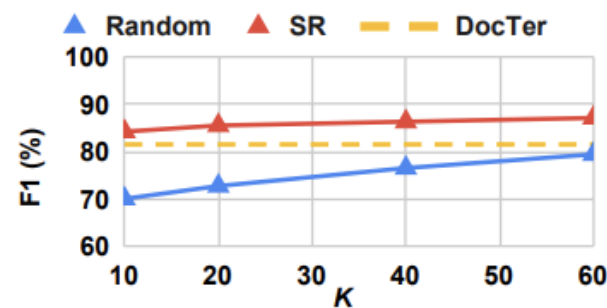
RQ2：プロンプト工夫すると性能はどの程度変化するか？

意味的選択(SR)は、ランダム選択よりも
常に高い性能

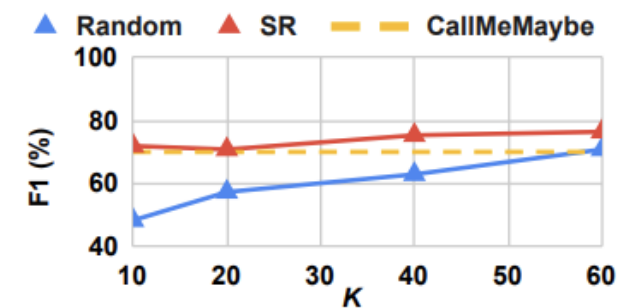
- 最大約10%



(a) Jdoctor-data



(b) DocTer-data



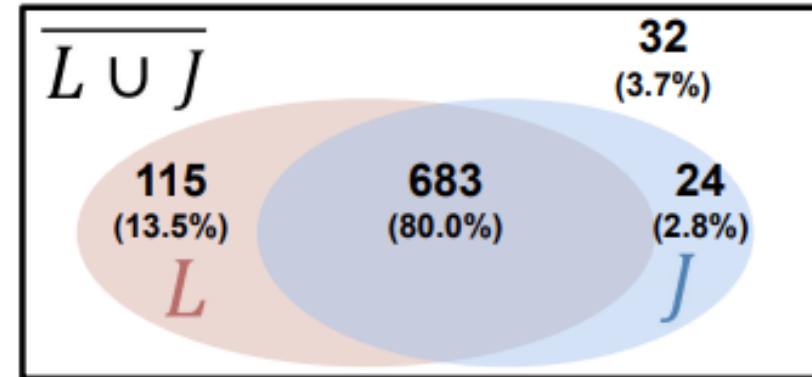
(c) CallMeMaybe-data

適切なプロンプト設計がFSLの性能向上に重要

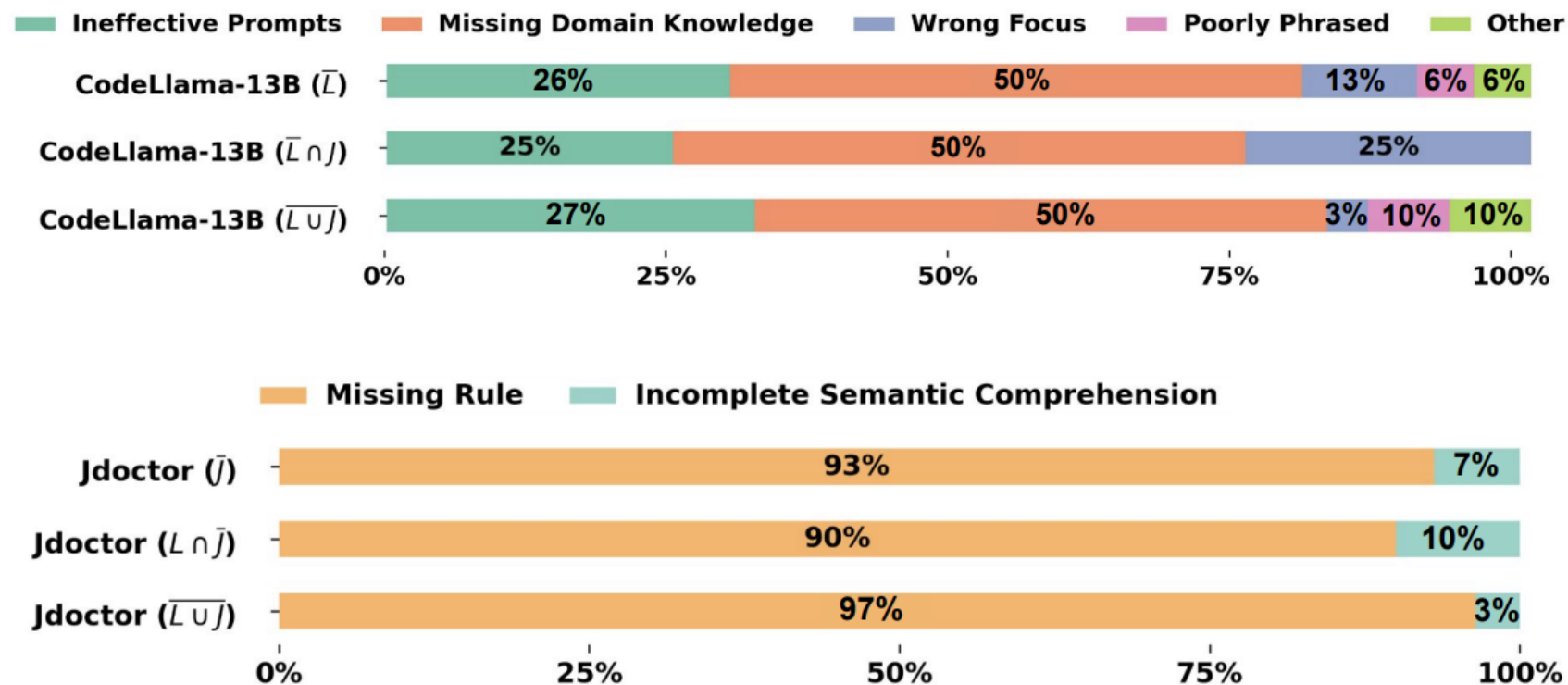
RQ3：LLMと従来手法の強みと弱み

失敗事例の比較分析

- L : CodeLlama-13B
- J : Jdoctor



- 図の各領域からランダムに30件ずつサンプルを抽出



失敗の原因分析

原因特定方法

- 反事実的推論法：変数を1つずつ変更して結果への影響を観察。
 - 失敗を修正できる方法を見つけ、その修正の性質から原因を特定する。

LLMの失敗原因

- 効果の薄いプロンプト
 - 特定分野の知識欠如
- ⇒この2つが全体の75%を占める。

従来手法の失敗原因

- ルールの欠如
- 文脈等の要素間の関係の理解の不完全さ

RQ4：モデル比較

CocTerとの比較

Approach / Model (+SR)	K	TensorFlow	PyTorch	MXNet	Overall	Cost (\$)	
DocTer	-	90.0/74.8/81.7	78.4/77.4/77.9	87.9/82.4/85.1	85.4/78.2/81.6	-	
GPT	4	60	84.7/87.5/86.1	82.8/87.3/85.0	86.8/89.9/88.3	85.4/88.6/ 87.0	78
	3.5	20	81.9/83.7/82.8	77.3/81.2/79.2	84.3/87.3/85.8	82.3/85.0/ 83.6	5.2
CodeLlama	13B	60	86.5/86.3/86.4	85.0/84.3/84.6	88.3/89.2/88.7	87.1/87.4/ 87.2	-
	7B	60	86.4/83.6/85.0	84.4/80.4/82.4	88.2/88.8/88.5	86.9/85.6/ 86.2	-
deepseek-coder	6.7B	60	85.9/85.4/85.7	85.6/84.6/85.1	87.5/89.1/88.3	86.6/87.0/ 86.9	-
Llama3	8B	60	86.1/82.8/84.4	83.3/83.0/83.1	87.7/85.6/86.7	86.4/84.2/ 85.3	-
Llama2	13B	20	83.9/80.0/81.9	80.4/74.2/77.2	84.4/81.3/82.8	83.6/79.6/81.5	-
	7B	20	80.5/79.2/79.8	77.7/75.5/76.6	84.0/80.6/82.3	81.7/79.3/80.5	-
StarCoder2	15B	60	87.9/87.2/87.6	85.8/85.4/85.6	88.6/89.4/89.0	87.9/88.0/ 87.9	-
	7B	60	86.8/86.7/86.7	84.8/85.4/85.1	87.7/89.1/88.4	86.9/87.6/ 87.2	-
StarCoder	16B	60	85.0/86.7/85.8	83.4/84.6/84.0	87.4/89.2/88.3	85.9/87.5/ 86.7	-
CodeGen2	16B	10	79.7/81.4/80.6	77.6/79.3/78.5	85.0/86.3/85.7	81.9/83.4/ 82.7	-
	7B	10	77.7/77.3/77.5	76.6/77.2/76.9	82.2/84.0/83.1	79.7/80.5/80.1	-

13モデル中

- 13モデルがJdoctorを上回る
- 10モデルがDocTerを上回る
- 9モデルがCallMeMaybeを上回る

課題と今後

ハイブリッドアプローチ

- LLMの文脈理解，汎化性能を活かしつつ，ルールベースが得意とする正確な制約処理，ドメイン知識の補完が可能に.

プロンプト設計の改善

- より柔軟でドメイン特化したプロンプト作り
 - 自動プロンプト最適化やタスクごとのカスタムテンプレートが研究中.

まとめ

ほとんどのLLMが従来手法と同等またはそれ以上の性能を示した.

LLMの主な課題

- 正しいプロンプト設計不足
- ドメイン知識の欠如

今後の展望

- ハイブリッドアプローチ
- プロンプト設計の改善

