

A Comparative Study on Large Language Models for Log Parsing

Merve Astekin¹, Max Hort¹, Leon Moonen¹²

¹Simula Research Laboratory, Oslo, Norway

²BI Norwegian Business School, Oslo, Norway

ログとは

システムが記録する動作履歴をログという

- ログは、デバッガが使えないような本番環境での不具合原因特定に有用[1].

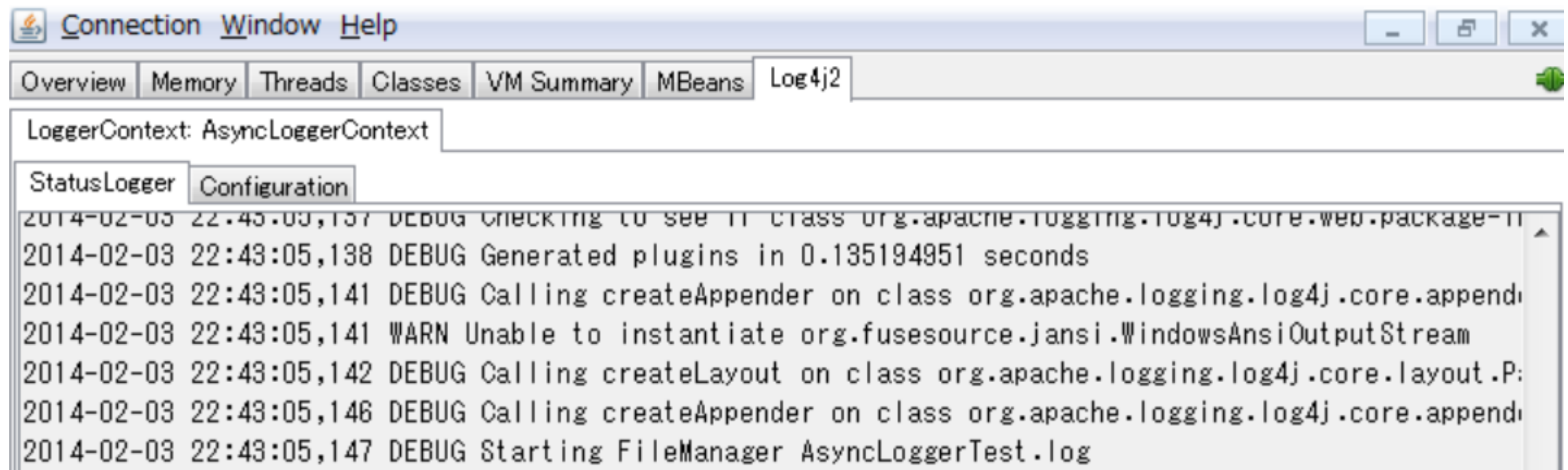


図 1 : Log4jのログステートメントにより出力されるログの例

[1] Yuan, D., Park, S., & Zhou, Y. (2012). Characterizing logging practices in open-source software. 2012 34th International Conference on Software Engineering (ICSE), 102-112.

ログ構文解析

ログは非構造化形式で提供されている

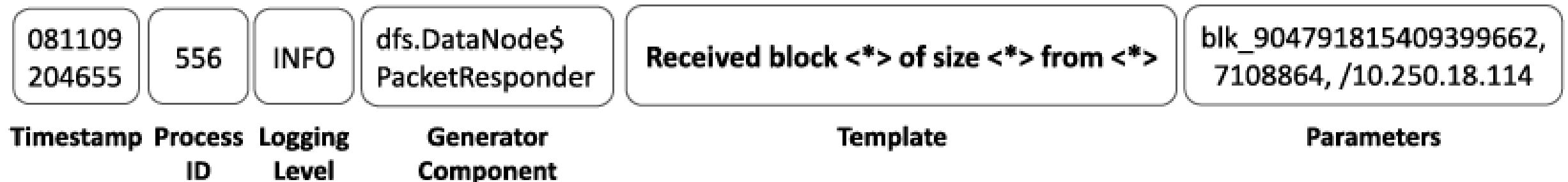
- ログは基本的にJSONのような形式でなく、1つの文(String)として出力されるため、構造化されているとは言えない

分析するにあたり、扱いやすくするため、構造化することがある

➤ そのために **ログ構文解析** を行う

```
081109 204655 556 INFO dfs.DataNode$PacketResponder: Received block blk_3587508140051953248 of size 67108864 from /10.251.42.84
```

Raw log message



ログ構文解析の手法

主に2つに分けられる

- **構文ベースのアプローチ**

- 事前に設定されたルールや特徴(テンプレート)に基づいて、構文解析を行う
- ルールや特徴の例外となるようなログは原則解析できない

- **セマンティックベースのアプローチ**

- ログのセマンティクスを学習させたディープニューラルネットワーク(DNN)等を用いて、解析を行う
- 発想としては、入力したログに共通する**テンプレート**をDNNやLLMに考えさせ、テンプレートとパラメータを出力させる

Received block <*> of size <*> from <*>

テンプレートの例

ログ構文解析の手法

主に2つに分けられる

- 構文ベースのアプローチ

- 事前に設定されたルールや特徴(テンプレート)に基づいて、構文解析を行う
- ルールや特徴の例外となるようなログは原則解析できない

- セマンティックベースのアプローチ

- ログのセマンティクスを学習させたディープニューラルネットワーク(DNN)等を用いて解析を行う
- 発想と構文解析の発想は異なるが、LLMに考えさせ、テンプレートを作成する

本来、文字や数値が入力される箇所に
一時的に置く文字や値をプレースホルダ
という

Received block <*> of size <*> from <*>

テンプレートの例

ログ構文解析におけるLLM

ログ構文解析においても、LLMの活用が注目を浴びている

- セマンティックベースのアプローチの一環

実際にChatGPTのようなLLMにおいて、有望な結果が出ている

しかし、研究はまだまだ不十分である

- 他の最先端LLMとの比較研究がない
- 当該タスクにおいて、ChatGPT並みの性能を発揮するオープンソースLLMがあるかどうか不明

研究目的

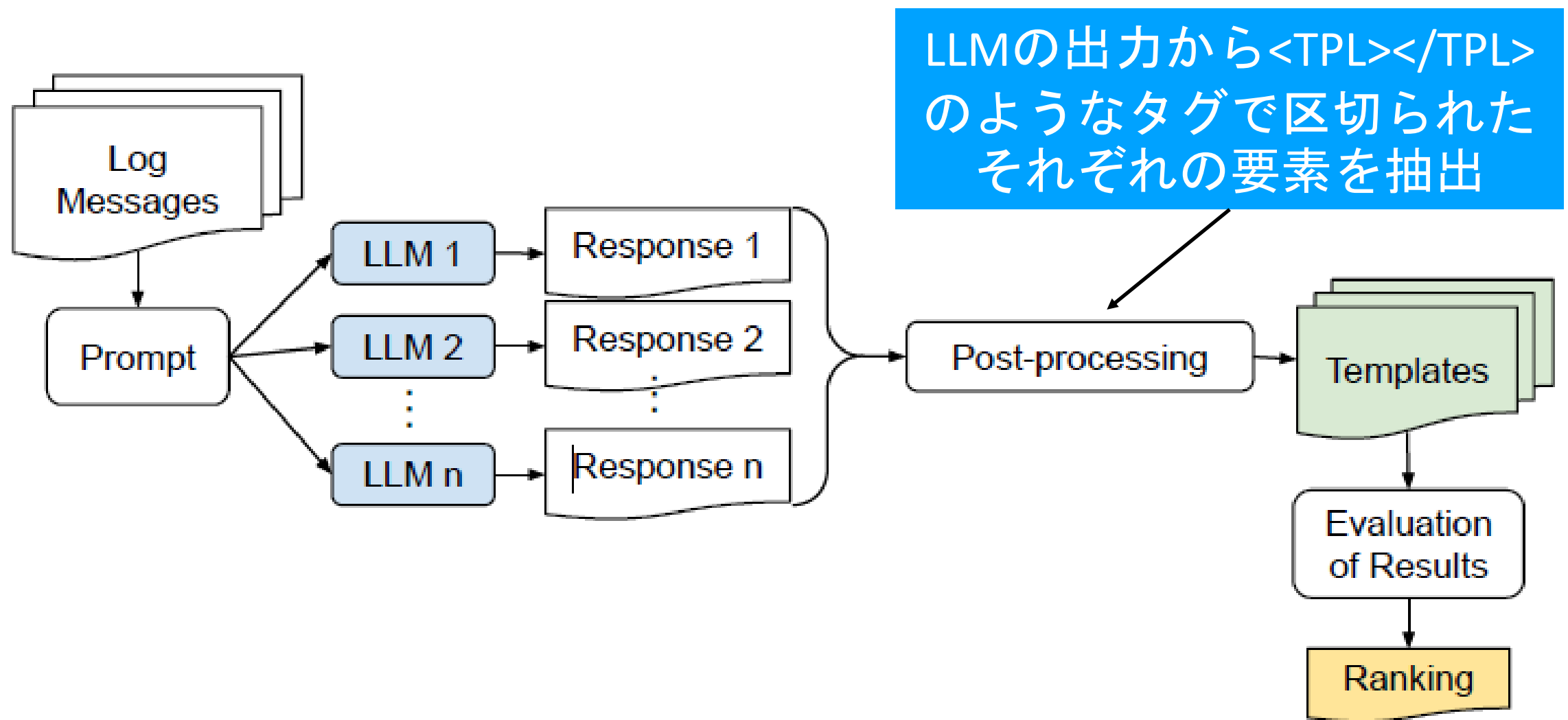
研究目的：

- ログ構文解析を実行する最先端のLLMの現在の能力の調査
- 有料のプロプライエタリモデル(proprietary model)とオープンソースのモデルを含む6つの最新のLLMの比較

アプローチ：

- 共通のプロンプトに対する各LLMの構文解析タスク遂行能力(=**テンプレート作成能力**)をメトリクスを用いて評価
 - また、LLMの構文解析タスク遂行能力を評価するには、どのメトリクスが適切かについても評価を行う
- LLMの出力そのものを観察し、定性的に評価

実験設定：概要



共通のプロンプトに対する各LLMの構文解析タスク遂行能力をメトリクスを用いて評価

実験設定：プロンプト，評価用データセット

Khan et alにより修正されたLogHub を使用

- 現実世界のログメッセージのデータセット

Type	Project	# Original [31]	# Khan et al. [13]
Distributed Systems	HDFS	14	14
	Hadoop	114	114
	Spark	36	36
	Zookeeper	50	50
	OpenStack	43	43
Super computers	BGL	120	120
	HPC	46	46
	Thunderbird	149	149
Operating systems	Windows	50	50
	Linux	118	118
	Mac	341	341
Mobile systems	Android	166	158
	HealthApp	75	75
Server applications	Apache	6	6
	OpenSSH	27	26
Standalone software	Proxifier	8	8
Total		1363	1354

オリジナルのLogHubと修正されたLogHubそれぞれの固有テンプレートの内訳

実験設定：プロンプト，評価用データセット

| Khan et alにより修正されたLogHub を使用

- 現実世界のログメッセージのデータセット

Type	Project	# Original [31]	# Khan et al. [13]
	HDFS	14	14
	Hadoop	114	114

テンプレートに関する問題を解決した結果、
データセット内の
固有テンプレートの量が少しだけ減っている

Server applications	OpenSSH	27	26
Standalone software	Proxifier	8	8
Total		1363	1354

オリジナルのLogHubと修正されたLogHubそれぞれの固有テンプレートの内訳

実験設定：プロンプトの概要

Zero-shot, Few-shotそれぞれのプロンプトを用意

- Few-shotにおける例は, Q&A形式の4組となっている
 - 2つはLogHubから, もう2つはLogHub以外のデータセットからとってきている

プロンプトプレフィックスの採用

- The log message typically consists of two parts:のような形式の接頭辞を与えることで, LLMのタスク理解を促進

動的変数は山括弧内にプレースホルダーを置くよう指示

プロンプト全文は補足資料として掲載

実験設定：使用LLM

有料プロプライエタリモデル:

- GPT-3.5、Claude 2.1

無料利用可能オープンモデル:

- Llama2、CodeLlama、Zephyr、CodeUp

Model	Version/ Release time	Base model	Availability
GPT-3.5	gpt-3.5-turbo-0613/ June, 2023	-	Closed-source
Claude 2.1	November, 2023	-	Closed-source
Llama 2	llama2:7b/ July, 2023	-	Public
CodeLlama	codellama:7b-instruct/ August, 2023	Llama 2	Public
Zephyr Beta	zephyr:7b/ October, 2023	Mistral-7B	Public
CodeUp	codeup:13b/ July, 2023	Llama 2	Public

実験設定：評価指標

解析精度（PA）：

- 正しく解析されたログメッセージの数を評価
 - 抽出されたテンプレートとパラメータが正解データと完全一致していれば True

編集距離（ED）：

- 生成されたテンプレートと正解データ間のテキスト類似性を測定（最小化）

最長共通部分列（LCS）：

- 生成されたテンプレートと正解データ間のテキスト類似性を測定（最大化）

EDとLCSについては、複数の評価法があるため、それらのどれを用いるべきかについても評価を行う

- Ex. EDの平均値 or 中央値をとり、それを比較する

実験設定：評価指標

編集類似度

- 編集距離を用いて、2つの文字列がどれだけ似ているかを示す指標
 - 値が高いほど類似している
- $EDnorm(\text{正規化されたED}) = \frac{ED(T, GT)}{\max(\text{len}(T), \text{len}(GT))}$
- $ESnorm = 1 - EDnorm$
 - 後ろの項で編集距離が正規化されていることから、この論文では「正規化された編集類似度」と表現されていると考えられる

正規化されたLCS

- $LCSnorm = \frac{LCS(T, GT)}{\text{len}(GT)}$

結果

3つのリサーチクエスチョンを設定

RQ1:最新LLMの構文解析タスクに対する性能比較

RQ2:今回の実験におけるメトリクスの影響

RQ3:LLMの出力の実用性

- LLMの出力からテンプレート部分を簡単に抽出できるか？
 - 抽出が機械的にできると嬉しい

RQ1:最新LLMの構文解析タスクに対する性能比較

解析精度：

- **CodeLlama**が最高のパフォーマンスを発揮
 - 1354のログメッセージ中586を正しく解析(43.3%)
 - 次点でGPT3.5とCluade2
 - 原則Zero-shotよりFew-shotの方が精度が優れている
 - Zepherは長いプロンプトへの対応が厳しいため、このような結果になったと考えられる

	GPT-3.5		Claude2.1		Llama 2		CodeLlama		CodeUp		Zephyr	
	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few
合計(1346)	309	531	346	439	107	281	470	586	193	393	126	63

各LLMの構文解析精度

RQ1:最新LLMの構文解析タスクに対する性能比較

編集距離（ED）と最長共通部分列（LCS）：

- EDではCodeLlamaが、LCSではGPT3.5-fewとClaude 2-zeroが最高のパフォーマンス

	GPT-3.5		Claude2.1		Llama 2		CodeLlama		CodeUp		Zephyr	
	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few
合計	11.0	4.0	14.0	11.0	31.0	11.0	7.0	3.0	36.0	9.0	35.0	37.0

各LLMの編集距離(ED)

	GPT-3.5		Claude2.1		Llama 2		CodeLlama		CodeUp		Zephyr	
	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few	Zero	Few
合計	38.0	39.0	39.0	38.0	12.0	35.0	37.0	38.0	35.0	38.0	33.0	28.0

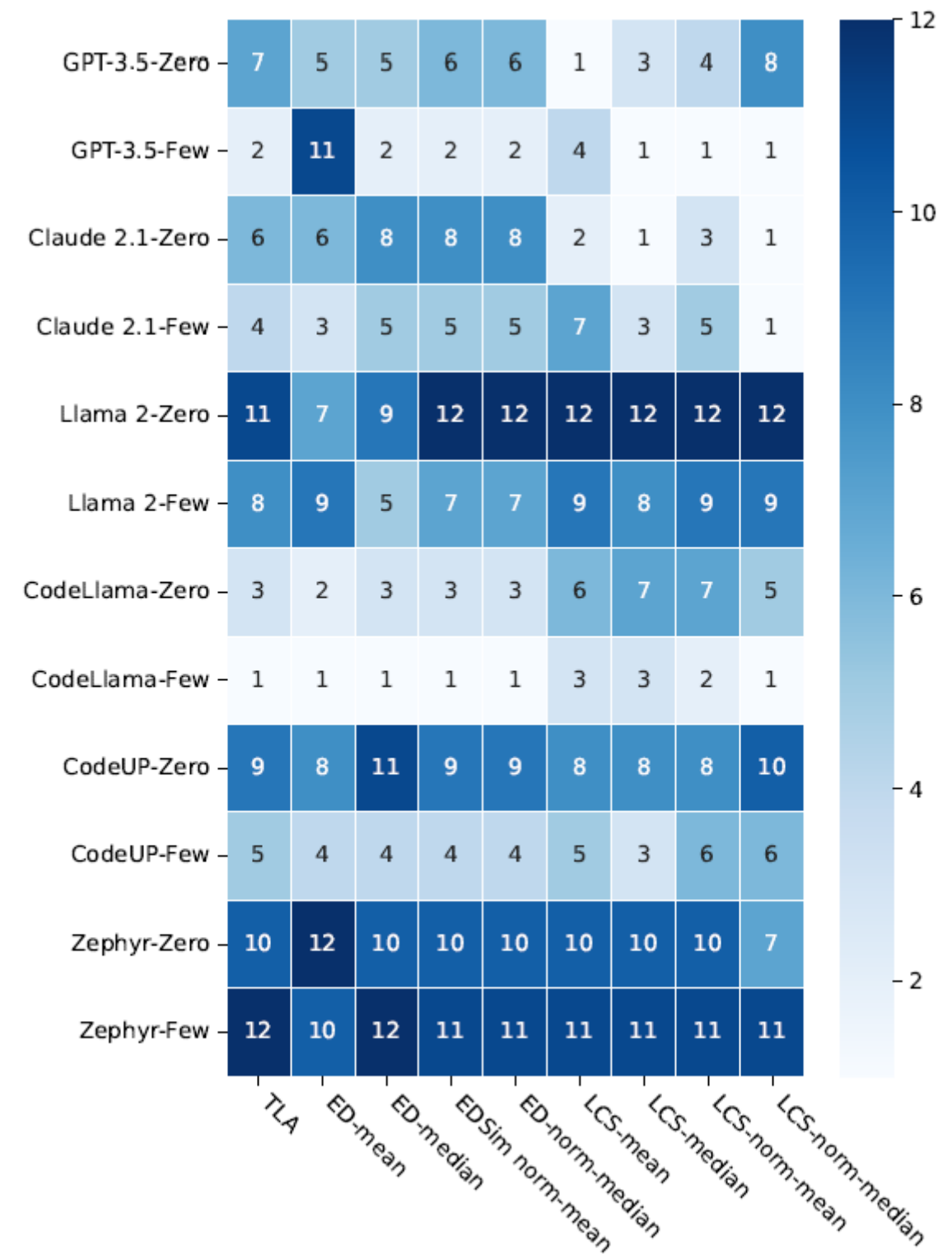
各LLMの最長共通部分列(LCS)

RQ2:今回の実験におけるメトリクスの影響

右図は各評価指標でLLMをランク付けした際のランクの一覧

- 明度が**高い**ほど順位が**高く**,
明度が**低い**ほど順位が**低い**

最高順位と最低順位は
すべての指標において
およそ一貫している



RQ2: 今回の実験におけるメトリクスの影響

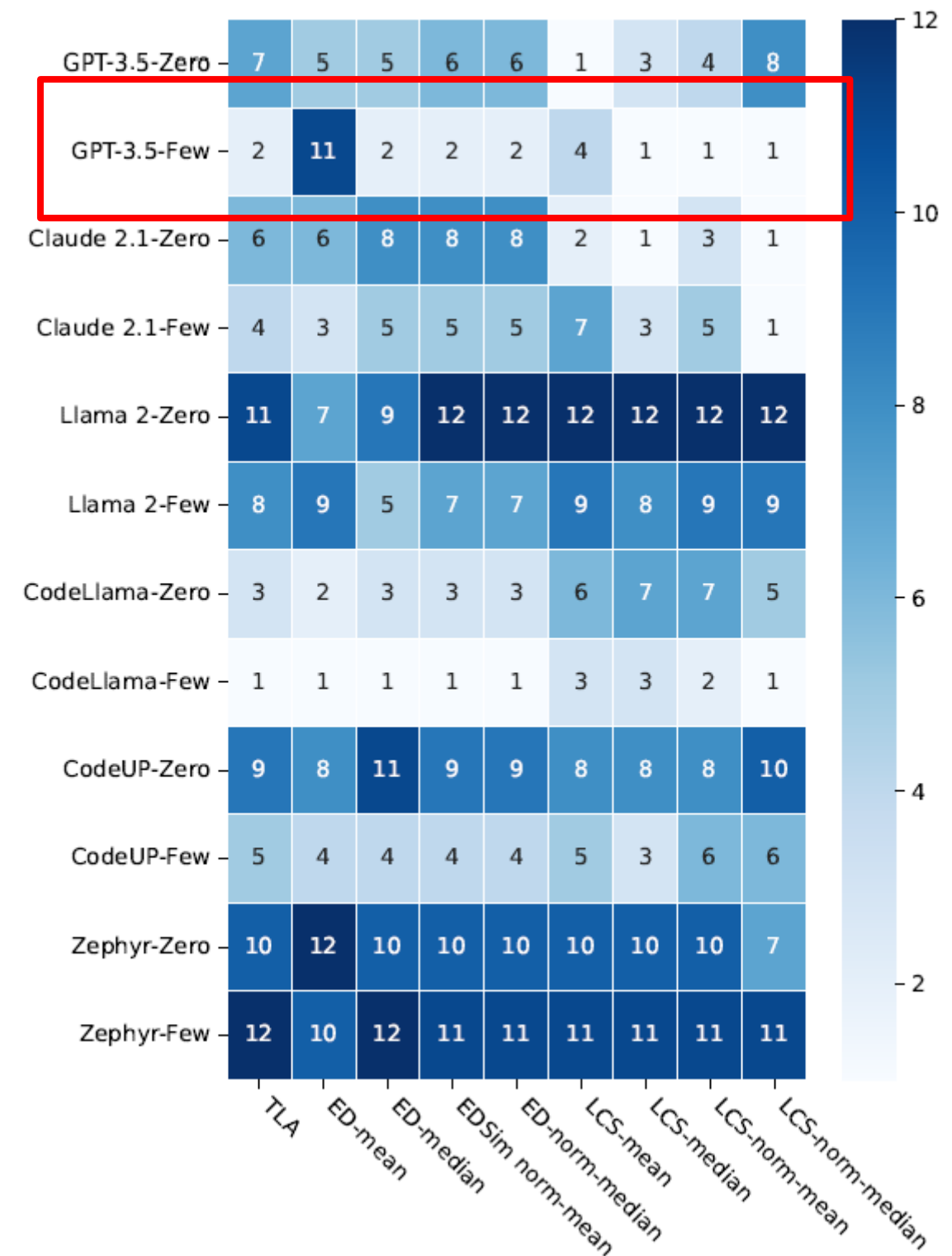
GPT-3.5-Fewにおいて、ED-meanだけ
極端に順位が低い



外れ値による影響が強く出ている



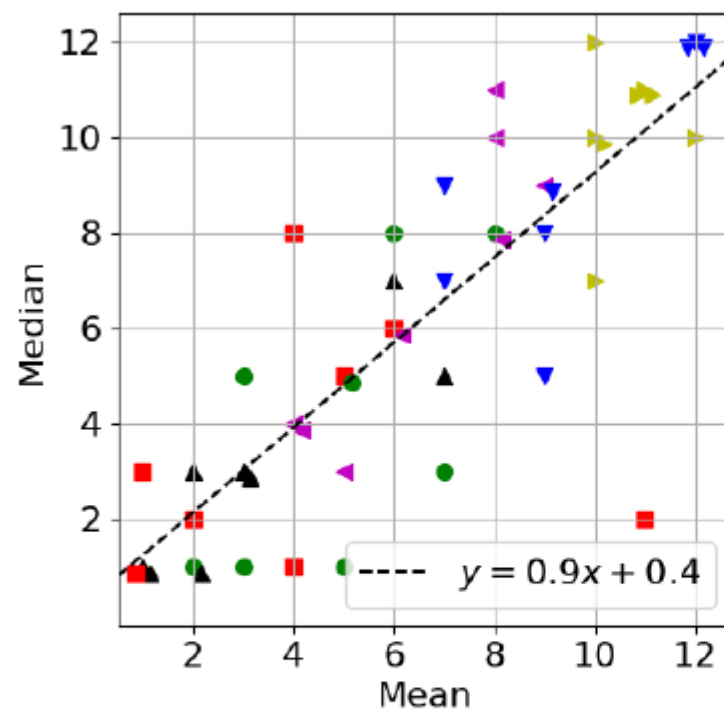
正規化しているED-norm-meanでは
他と同じような順位であること
から、平均の正規化の必要性を
主張



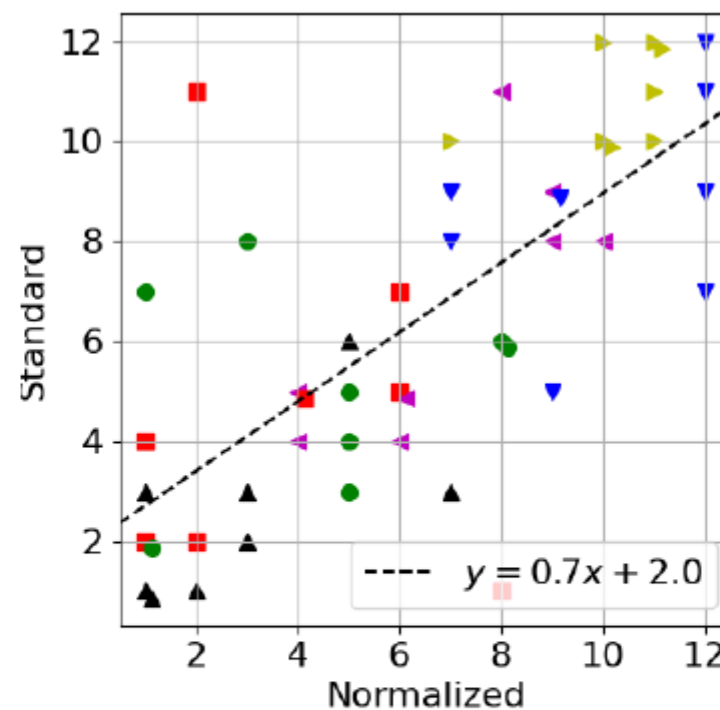
RQ2:今回の実験におけるメトリクスの影響

下の図はそれぞれ平均値と中央値, 標準値と正規化値, LCSとEDの相関係数を計測した図となっている

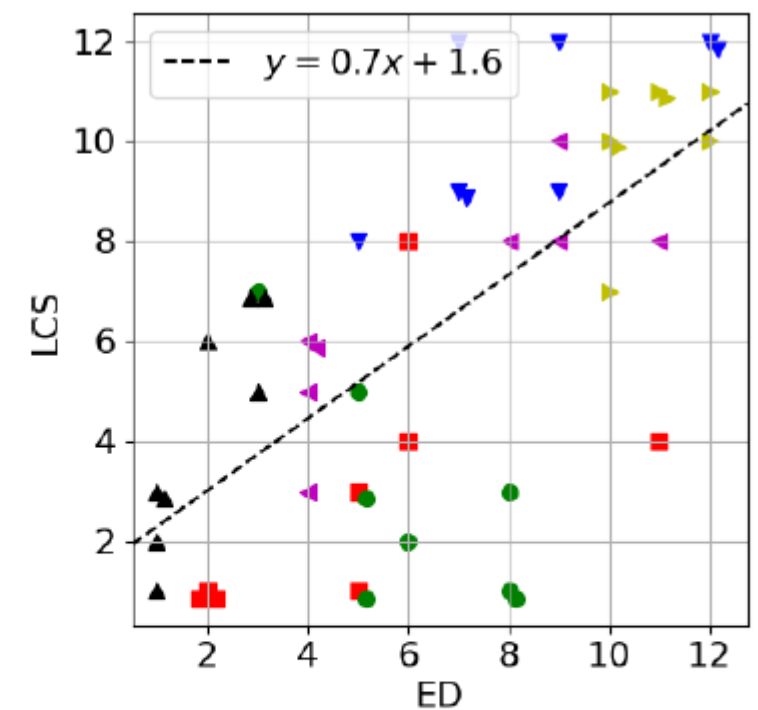
- 相関係数が高い→2つの評価指標には互換性がある(≒ランキングが同じような感じになる)



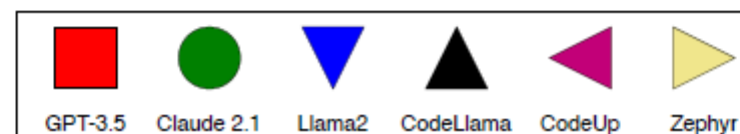
(a) Mean and median.
Correlation coefficient = 0.83



(b) Standard and normalized.
Correlation coefficient = 0.72



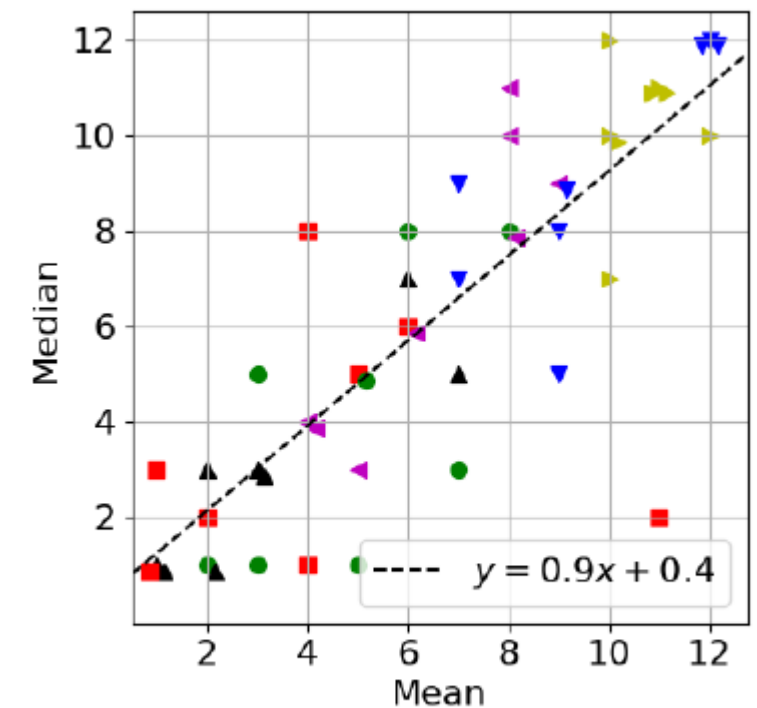
(c) Edit distance and Longest Common Subsequence.
Correlation coefficient = 0.68



RQ2: 今回の実験におけるメトリクスの影響

平均と中央の相関係数は0.83

- 平均値と中央値のそれぞれによるランク付けは、どちらも同じようなランキングになるということが示唆されている
- それでも外れ値の影響から中央値を使うことが推奨される。

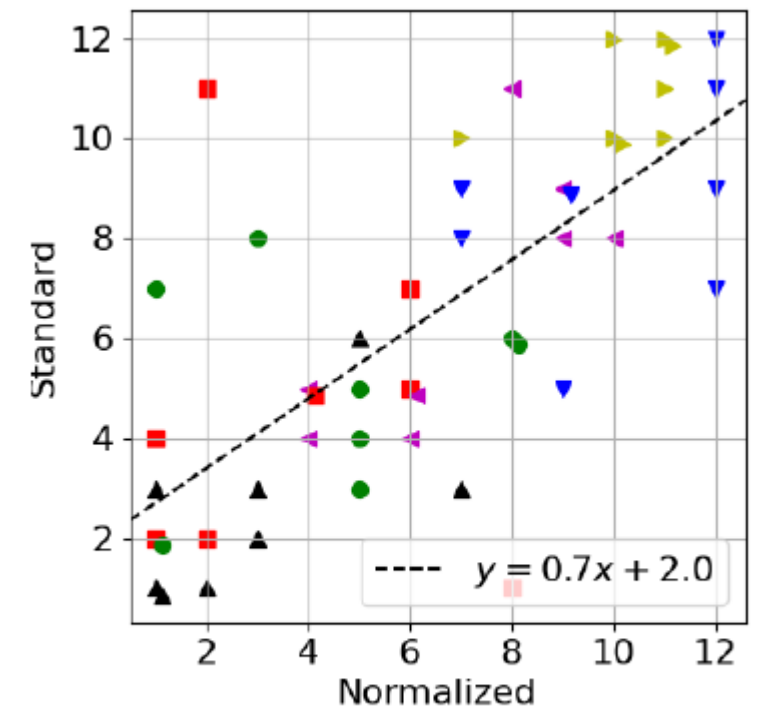


(a) Mean and median.
Correlation coefficient = 0.83

RQ2: 今回の実験におけるメトリクスの影響

標準と正規化の相関係数は0.72

- 標準値と正規化値のそれぞれによるランク付けは、どちらもだいたい同じようなランキングになるということが示唆されている
- すなわち、正規化によるランクの変動はそこまで大きくないといえる

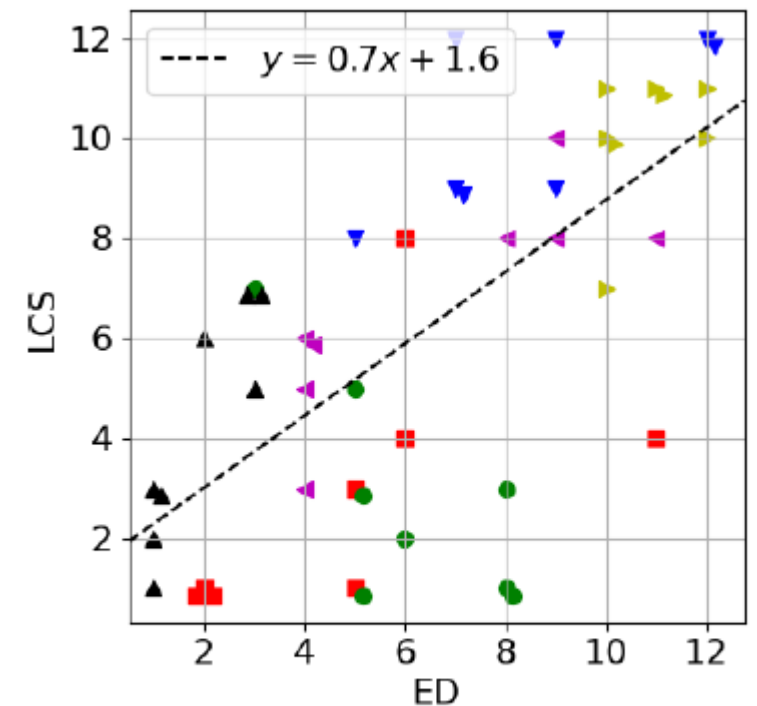


(b) Standard and normalized.
Correlation coefficient = 0.72

RQ2:今回の実験におけるメトリクスの影響

EDとLCSの相関係数は0.68

- タイムスタンプ/メッセージスコープとプレースホルダーの扱い方の違いによるもの
- EDとLCSそれぞれによるランキングは異なるといえる.
- そのため, どちらを使うかは検討する必要がある.



(c) Edit distance and Longest Common Subsequence.
Correlation coefficient = 0.68

RQ3: 応答の実用性

プレースホルダーの選択と変数フォーマット:

- LLMは、プロンプトで明示的に指示しても、仕様を満たさない出力を行うことが多い
 - Ex. 変数のプレースホルダーやフォーマットに関して一貫性がない

CodeLlamaとCodeUp:

- 十分な処理能力がある場合、無料利用可能なモデルとして推奨される。生成される結果が比較的解析しやすい

GPT-3.5:

- 有料モデルの中では、Claude 2.1よりも安価であり、解析しやすい結果を生成するため、代替手段として推奨される。

結論

- | ログ解析における6つのLLM（有料2、無料4）のパフォーマンスを比較
- | CodeLlamaとGPT3.5が最高のパフォーマンスを発揮
- | CodeLlama(無料)は高性能であり、ログ解析のための有料APIの実行可能な代替手段となる
- | ログ解析アプローチを評価する際に、正しいメトリクスを選択することが重要である

所感

LLMによる構文解析はまだ実用レベルではなさそう

- そもそも、解析精度が一番高くて43.4%と50%もいってない
- 出力のフォーマットのブレもまた大きな問題

評価指標に関する評価は参考になった

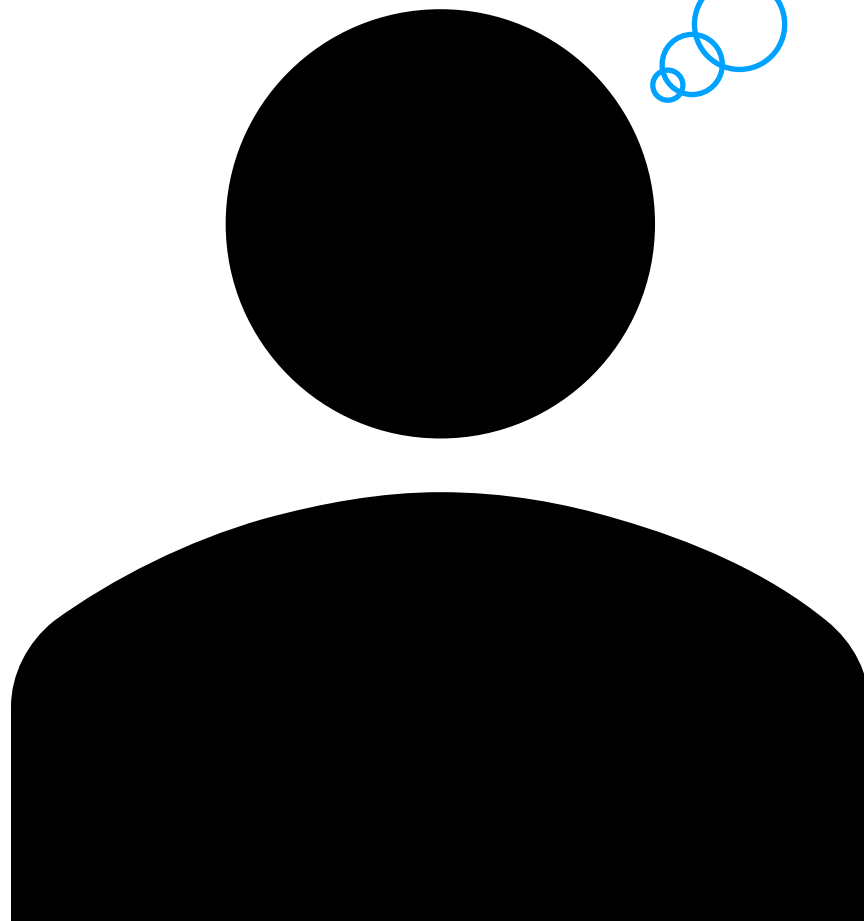
- 結果そのものは「それはそう」としか言いようがなかった

動機

- | 現在ログステートメントに関する研究を行っている
- | ログステートメントは構造化されたログを出力するのに役立つ
- | ログが構造化されているとログを構文ベースで解析する際に役立つ
- | しかし、ログ解析はLLM等を用いてセマンティックに解析するならそもそもログを構造化する必要性はないともいえる

要するに

どんなに汚くて雑で適当なログでも
LLMがいい感じに解析してくれる
なら「良いログの出力方法」は
考えなくてもいいのでは？



研究の根本が揺るぎ
かねないため、今回の論文
を読むことにした

補足：プロンプト(1/2)

You will be provided with a log message delimited by **<MSG>** and **</MSG>**.

The log texts describe various system events in a software system. A log message usually contains a header that is automatically produced by the logging framework, including information such as timestamp, class, and logging level (INFO, DEBUG, WARN etc.).

The log message typically consists of two parts:

- 1. Template** - message body, that contains constant strings (or keywords) describing the system events;
- 2. Parameters/Variables** - dynamic variables, which reflect specific runtime status.

You must identify and abstract all the dynamic variables in the log message with suitable placeholders inside angle brackets to extract the corresponding template.

You must output the template corresponding to the log message. Print only the input log's template surrounded by **<TPL>** and **</TPL>**. Never print an explanation of how the template is constructed.

補足：プロンプト(2/2)

Here are a few examples of log messages (labeled with Q:) and corresponding templates (labeled with A:):

Q: `<MSG>[081109 204453 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.11.85:50010 is added to blk_2377150260128098806 size 67108864]</MSG>`

A: `<TPL>[BLOCK* NameSystem.addStoredBlock: blockMap updated: <*>:<*> is added to <*> size <*>]</TPL>`

Q: `<MSG>- 1129734520 2005.10.19 R17-M0-N0-I:J18-U01 2005-10-19-08.08.40.058960 R17-M0-N0-I:J18-U01 RAS KERNEL INFO shutdown complete</MSG>`

A: `<TPL>shutdown complete</TPL>`

Q: `<MSG>20231114T101914E ERROR 14 while processing line 123: cannot find input '42'</MSG>`

A: `<TPL>ERROR <*> while processing line <*>: cannot find input <*></TPL>`

Q: `<MSG>2023-01-14 23:05:14 INFO: Reading data from /user/input/file.txt</MSG>`

A: `<TPL>Reading data from <*> </TPL>`

Here is the input log message:<MSG>...</MSG>
Please print the corresponding template.

補足：編集距離と最長共通部分列

編集距離：

- 一方の文字列をもう一方の文字列に変形するのに必要な手順の最小回数

最長共通部分列

- 双方に共通して現れる部分文字列の中で最長の文字列
- Ex. AGCATとGACの最長共通部分列はAC, GC, そしてGA
 - 部分文字列は連続して並んでいる必要はない