言語処理学会第 30 回年次大会 併設ワークショップ JLR2024 -日本語言語資源の構築と利用性の向上

Prompt Tuning から Fine Tuning への移行時期推定

アマゾンウェブサービスジャパン合同会社

久保 隆宏, 呉 和仁, 前川 泰毅

アジェンダ

1. 会社概要
2. 取り組んだ問題の背景 / 先行研究
3. 課題解決のための実験設計
4. 実装
5. 実験結果
6. 今後の展望

アジェンダ

1. **会社概要**
2. 取り組んだ問題の背景 / 先行研究
3. 課題解決のための実験設計
4. 実装
5. 実験結果
6. 今後の展望

アマゾンウェブサービスジャパン合同会社

日本で AWS を利用する時は AWS Japan が契約当事者になります。

AWS には 200 以上のサービスがあるため、目的を達成するのにどのサービスを組み合わせればよいのか迷うことがあります。そんな課題を解決するソリューションアーキテクト (SA) などが働いています。

AWS の SA には目黒セントラルスクエアの 17F にある AWS Loft というコワーキングスペースに行くと質問できたりします。

アマゾンウェブサービスジャパン合同会社

AWS は生成 AI の領域でも Amazon のように「品揃え」を拡充し「顧客体験をよくする」ことを指向しています。

LLM 開発支援プログラムでは総額 8 億円超のクレジットと学習用インスタンスを確保し、採択した 17 社の基盤モデル開発を支援しました。

開発後の収益化を支援すべく
AWS Marketplace を通じた販売や Amazon SageMaker JumpStart を通じた販路拡大を支援しています。

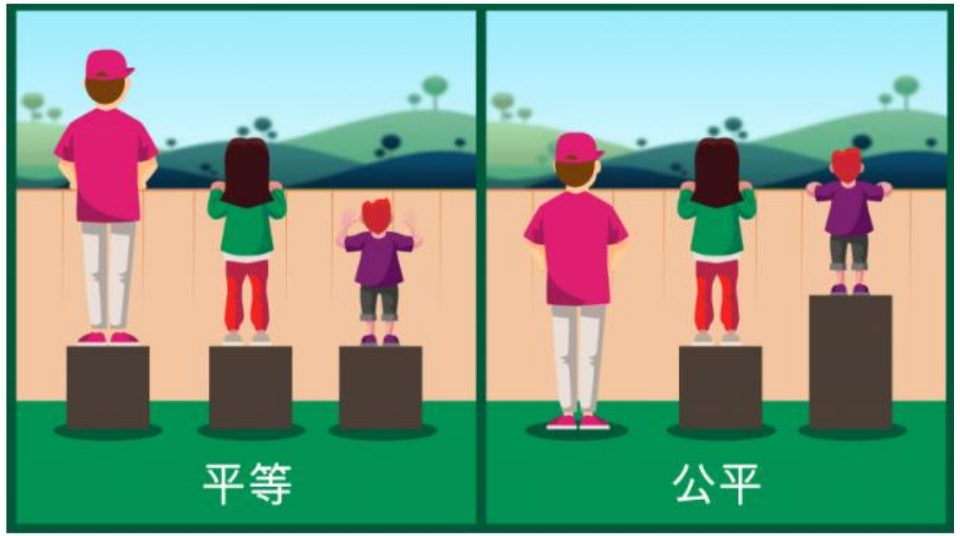
参考: AWSのAIインフラで何を作った？　NTTやストックマークが成果を公開

日本語大規模言語モデル（ 言語資源 ）の利用を促進する調査研究をしてきたので発表します！

アジェンダ

1. 会社概要
2. **取り組んだ問題の背景 / 先行研究**
3. 課題解決のための実験設計
4. 実装
5. 実験結果
6. 今後の展望

日本語の大規模言語モデル公開が進む一方、素の精度重視の評価が利用の移行を阻んでいる



公開モデル(※)の精度は上昇傾向だが、高性能な企業独自のモデル (Claude や ChatGPT) とは差がある。

公開モデルの利点はカスタマイズ（追加学習）できることなので、その特性を活かさない素の精度のみ評価されるのは、平等ではあるが公平ではないのでないか？

※公開されているモデルの中には厳密にはオープンソースの定義を満たさないものも含まれるため、本資料では一貫し「公開モデル」と表記します

平等と公平の画像 :特定非営利活動法人　ホップすてーしょん より引用

先行研究

モデルの評価には様々なベンチマークがあるが、Zero-shot / Few-shot の精度をみており学習データによる Fine Tuning が行われていない。

※ llm-jp-evalについては、zero-shotを使用し、各testデータの100問に対する評価を計算しています。Wikiのデータについては、全体で100問となるようにデータ数を設定しています。

Overall average = (llm-jp-eval + MT-bench/10) / 2

Nejumi リーダーボードは zero-shot 評価

runs.summary["leaderboard_table"]

	run_name	Overall average	AVG	EL	FA	MC	MR	NLI	QA	RC
2	gpt-4-0125-preview	0.7722	0.6463	0.3066	0.2547	0.96	0.97	0.762	0.4693	
33	gpt-4-0613	0.7622	0.6463	0.3004	0.2199	0.96	0.97	0.768	0.415	
37	gpt-4-1106-preview	0.7479	0.6295	0.317	0.2388	0.93	0.96	0.74	0.4002	
30	gpt-3.5-turbo	0.6701	0.5161	0.2913	0.1886	0.79	0.67	0.56	0.2723	
12	anthropic.claude-v2:1	0.6682	0.5188	0.201	0.129	0.84	0.84	0.676	0.2882	
10	gemini-pro	0.6402	0.5636	0.2188	0.1838	0.91	0.79	0.676	0.4196	

実際企業で利用する際はすでに対象ドメインのデータが一定量蓄積しているはずであり
(例:顧客応対のデータなど) 、公開モデルはチューニングしたうえで使える

明らかにしたいこと

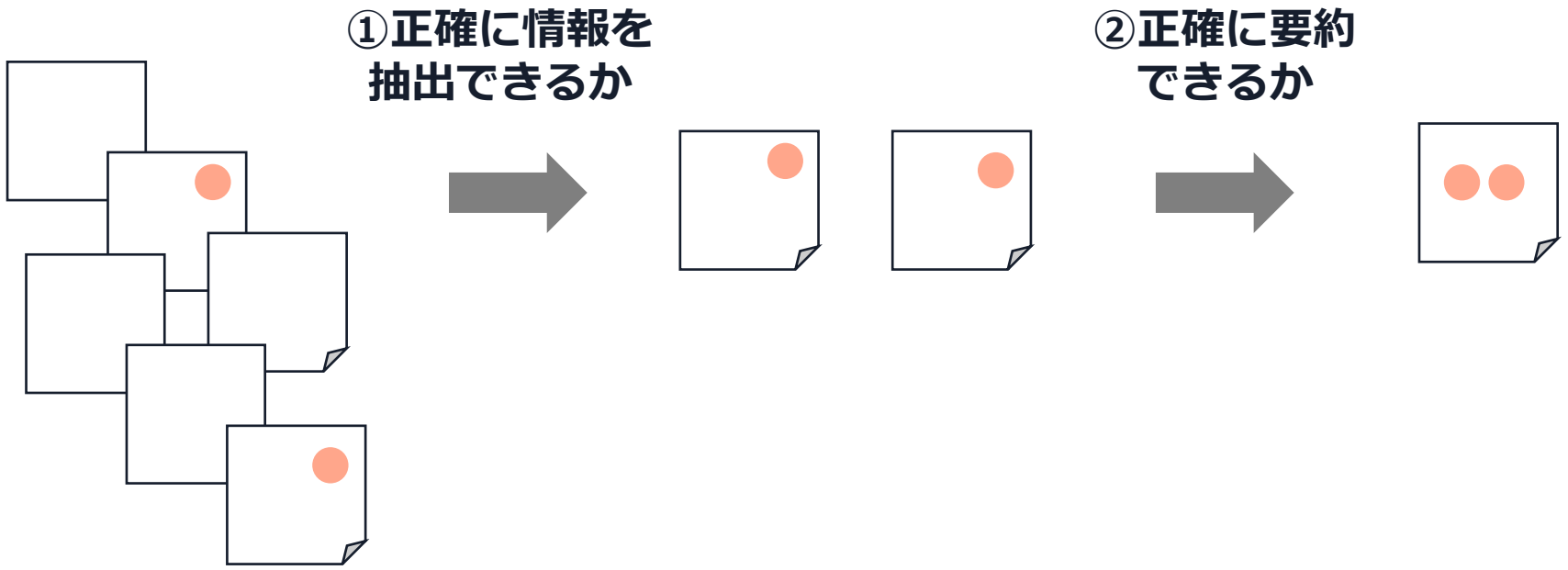
- 追加学習を前提とした場合 API と公開モデルの精度の差はどの程度になるのか？
- 追加学習にコストをかけることは費用対効果があるのか？

アジェンダ

1. 会社概要
2. 取り組んだ問題の背景 / 先行研究
3. **課題解決のための実験設計**
4. 実装
5. 実験結果
6. 今後の展望

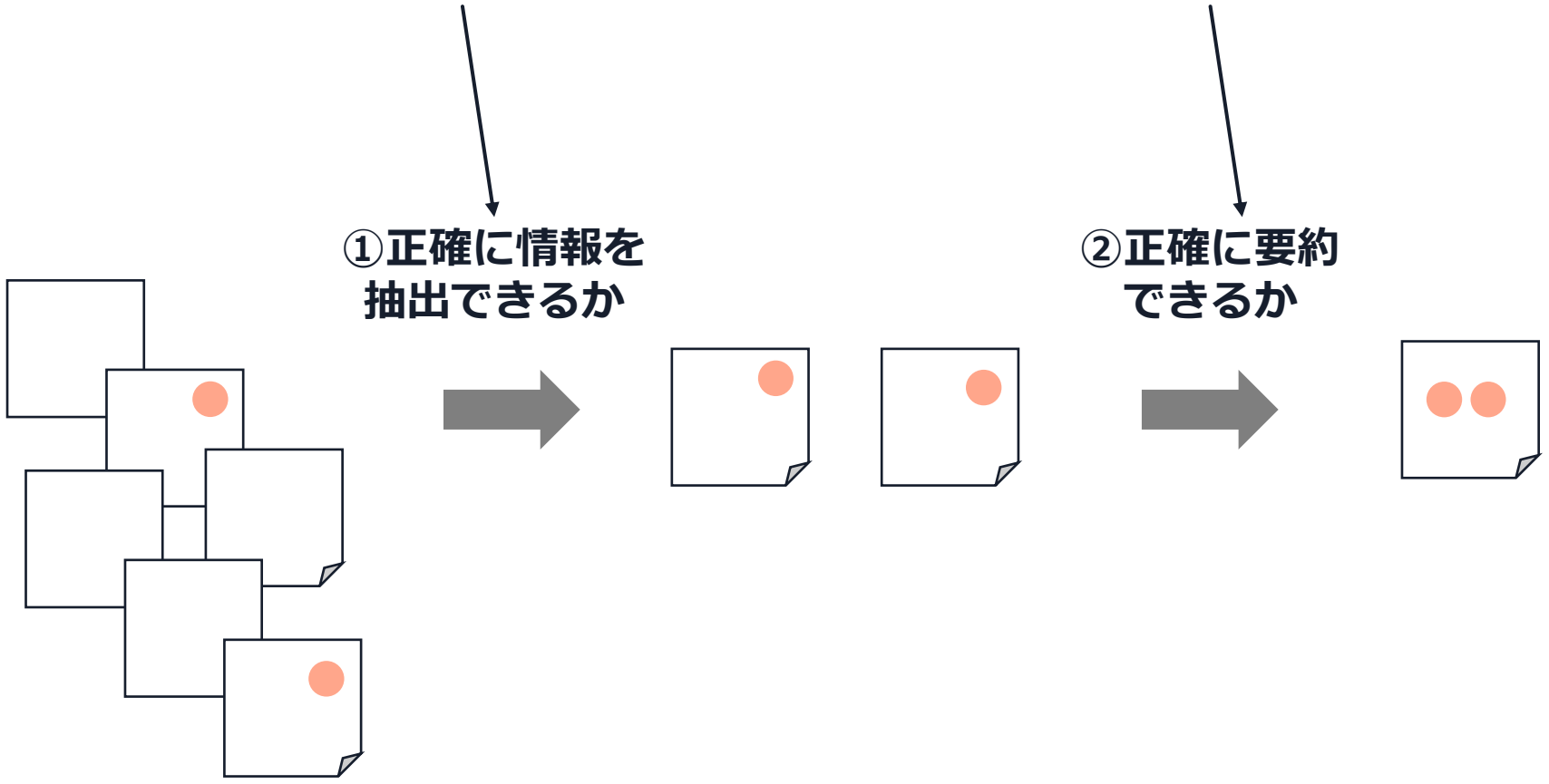
実験設計 (1/3)

基盤モデルの活用例として代表的な検索拡張検索 (RAG: Retrieval-Augmented Generation) での公開モデル活用を想定し、①情報抽出性能と②要約性能の 2 種類のタスクを評価する。

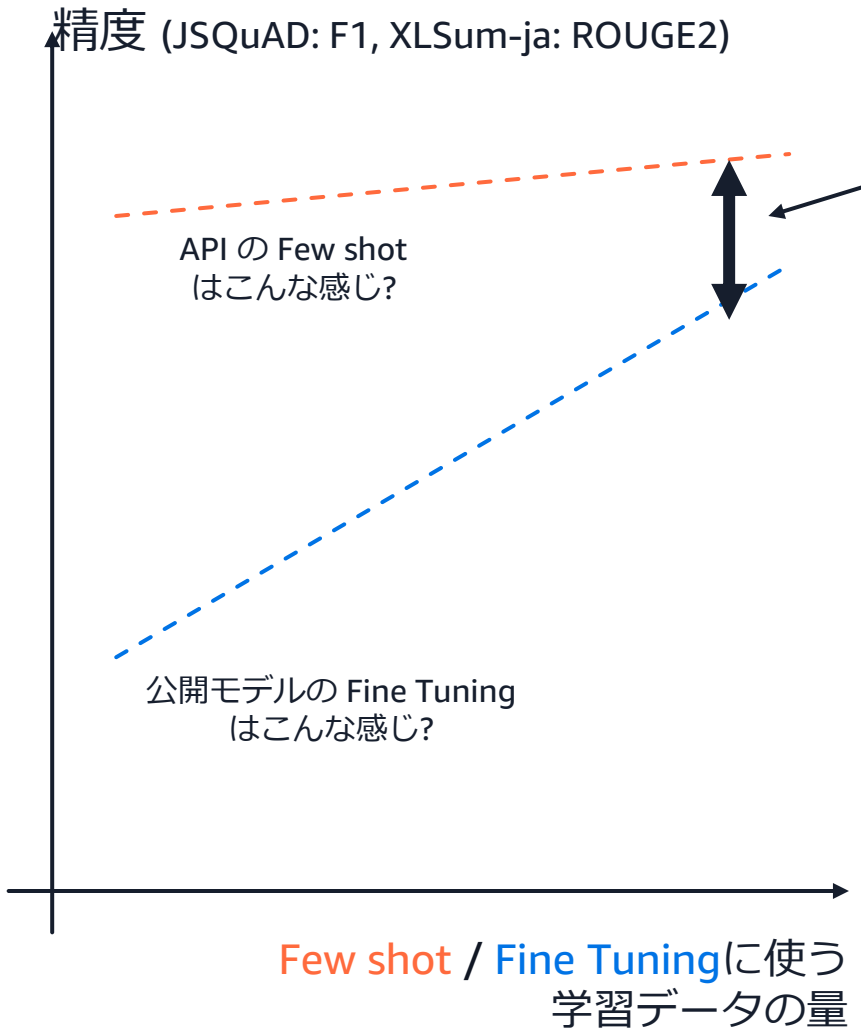


実験設計 (2/3)

①情報抽出性能として JSQuAD 、②要約性能として XLSum-ja で評価。



実験設計 (3/3)



何件ぐらいのデータがあれば、 API のモデルと同等の精度が得られるのかを知りたい。少ない件数、学習で良ければ公開モデルのコスト効率が高いことになる。

データを全く与えない場合を 0、そこから 2 、4 、 8 、 16 ・・・と 2 の倍数刻みでデータを増やしていき精度の変化を観測する。 API 側も Few-shot にデータを使い公平な設定で実験。

実験に使用するデータセットの詳細

JSQuAD

Wikipedia の記事 (context) に対する質問 (question) と回答 (answers) が収録されている。評価は完全一致 (exact match) と、文字単位の部分一致を加味する f1 。

同じ context のデータは類似性が高いため、学習データ (Instruction) を作る際は context が重複しないよう設計 (データ数が 15,000 件までは重複なしで作成可) 。

XLSum-ja

XLSum のデータセットから日本語のデータのみを抽出。 title 、 text 、 summary の 3 つ組から成る。評価は ROUGE2 (bi-gram の一致を評価) 。

実験に使用する公開モデル

1B 、 3~4B 、 7B 、 10B 超の 4 種類を用意

モデル	公開元	パラメータ数	概要
open-calm-1b	CyberAgent	1B	株式会社サイバーエージェントから公開された GPT-NeoX ベースの日本語大規模言語モデル。
japanese-gpt-neox-3.6b-instruction-ppo	rinna	3.6B	rinna 株式会社から公開された日本語で学習された GPT-NeoX ベースの大規模言語モデル。対話形式のデータで教師あり学習、強化学習が行われた日本語大規模言語モデル。
bilingual-gpt-neox-4b-instruction-ppo	rinna	4B	rinna 株式会社から公開された日英両言語で学習された GPT-NeoX ベースの大規模言語モデル。対話形式のデータで教師あり学習、強化学習を行っている。
ELYZA-japanese-Llama-2-7b-instruct	ELYZA	7B	株式会社 ELYZA から公開された、 Meta の Llama2 をもとに日本語コーパスで継続学習した大規模言語モデル。独自データでの教師あり学習を行っている。
Swallow-13b-instruct-hf	東工大 / 産総研	13B	東工大と産総研の研究チームから公開された、Meta の Llama2 をもとに日本語コーパスで継続学習した大規模言語モデル。

実験に使用する API のモデル

Claude 3 は間に合いませんでした！君の目で確かめてください

公開元	種別	概要
Claude v2.1	Anthropic	Anthropic の提供する高性能な基盤モデル。 Hugging Face の Leaderboard では GPT-4 などに次ぐ精度。日本語性能でも、 Rakuda Ranking などでトップレベルの性能を示す。20 万トークンという長大なテキストを扱える。
Claude Instant	Anthropic	高速な応答に重点を置いたモデル。 10 万トークンという長大なテキストを扱える。

アジェンダ

1. 会社概要
2. 取り組んだ問題の背景 / 先行研究
3. 課題解決のための実験設計
4. **実装**
5. 実験結果
6. 今後の展望

評価：JP Language Model Evaluation Harness を使用

llm-jp-eval にはまだ要約のデータセットがなかったので (2024/2/28 時点) 、双方が入っている Evaluation Harness を利用。

※ ただ、 JSQuAD の評価において F1 は文字単位で計算する必要があるが、 lm-evaluation-harness は JSQuAD の評価をトークン単位で計算しているので、本来の値とはずれてしまう (ただ、結論への影響は軽微と判断) 。

公開モデルの学習：Hugging Face で PEFT (LoRA) を実装

Full Fine Tuning でなく部分的にパラメーターを学習する PEFT (Parameter-Efficient Fine Tuning) を採用し、実装には Hugging Face (peft) を使用。

epoch 数は 3 まで実施。

Amazon SageMaker を使うと、

学習データ、学習スクリプト (次スライド)、実行環境

の 3 つを用意すれば簡単に学習できる。

Amazon SageMaker Training で機械学習のモデル開発を楽にする【ML-Dark-01】【AWS Black Belt】

Amazon SageMaker で LoRA 学習

Fine-tuning

```
base_job_name="OpenCALM"
hyperparameters={
    'base_model':'cyberagent/open-calm-7b',
    # 'load_in_8bit': True,
    'load_in_4bit': True,
    'pad_token_id': 1,
    'data_path': '/opt/ml/input/data/train/databricks-dolly-15k-ja.json',
    'num_epochs': 1, # default 3
    'cutoff_len': 512,
    'group_by_length': False,
    'output_dir': '/opt/ml/model',
    # 'resume_from_checkpoint': '/opt/ml/checkpoints',
    'lora_target_modules': '[query_key_value]',
    'lora_r': 16,
    'batch_size': 32,
    'micro_batch_size': 4,
    'prompt_template_name': 'alpaca',
}
```

学習データ

```
huggingface_estimator = HuggingFace(
    base_job_name=base_job_name,
    role=role,
    entry_point='finetune.py',
    source_dir='./scripts/code',
    instance_type='ml.g5.2xlarge',
    instance_count=1,
    volume_size=200,
    transformers_version='4.26',
    pytorch_version='1.13',
    py_version='py39',
    use_spot_instances=True,
    max_wait=86400,
    hyperparameters=hyperparameters,
    metric_definitions=[{'Name': 'eval_loss', 'Regex': "'eval_loss': (¥d¥.¥d+)"},
                        {'Name': 'train_loss', 'Regex': "'loss': (¥d¥.¥d+)"}],
    # checkpoint_s3_uri=f"s3://{bucket}/{base_job_name}/checkpoint/",
)
huggingface_estimator.fit({'train': input_train})
```

学習スクリプト、

実行環境 (A10G の g5)

fit で実行！

API モデルの推論：Amazon Bedrock でバッチ推論 (Preview)

JSQuAD の Validation dataset は 4000 件ぐらいあるので、普通に API を叩いているとあっという間にレートリミットにかかる。そのため、バッチ推論機能を使用。

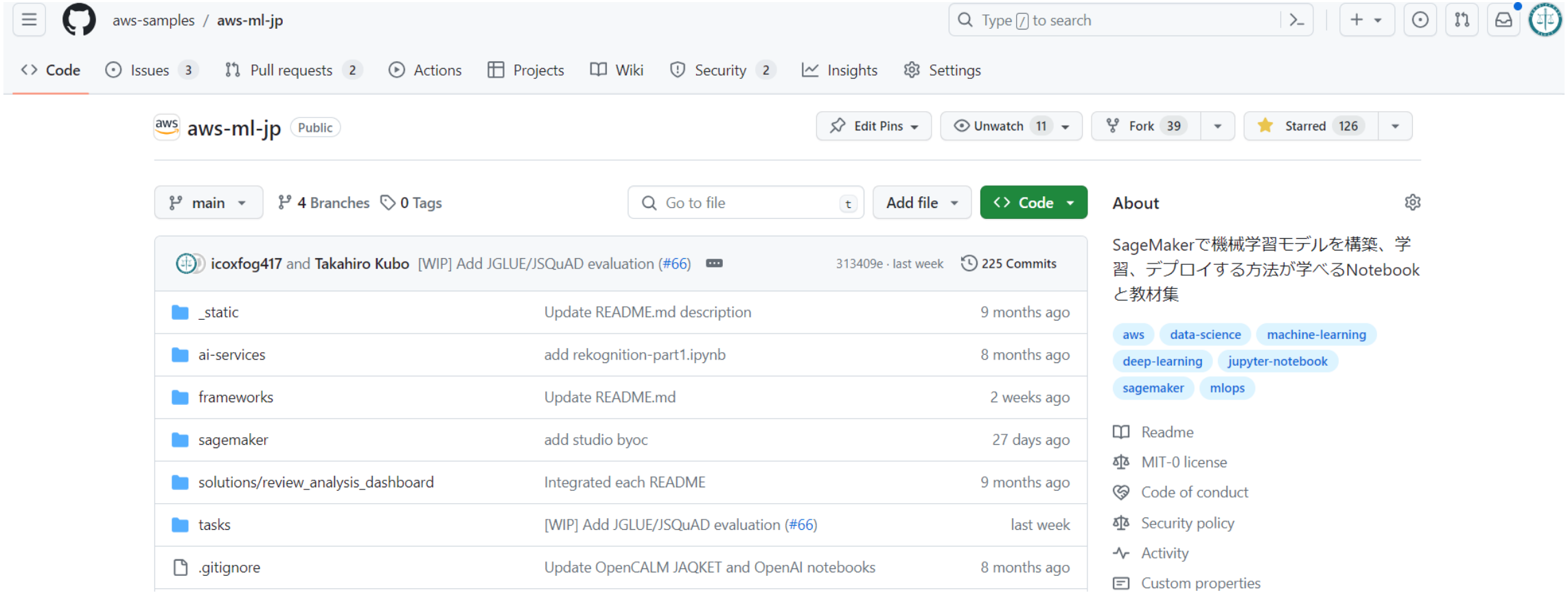
```
# Create batch inference job
inputDataConfig = {
    "s3InputDataConfig": {"s3Uri": f"s3://{bucket_name}/{input_key}"}
}

outputDataConfig = {
    "s3OutputDataConfig": {"s3Uri": f"s3://{bucket_name}/output/{job_name}/"}
}

_role_arn = role_arn if role_arn else self.identity["Arn"]
# Todo: fix patch code
_role_arn = (
    _role_arn.replace(":sts:", ":iam:")
    .replace("assumed-role/", "role/service-role/")
    .replace("/SageMaker", "")
)
response = self.batch_client.create_model_invocation_job(
    roleArn=_role_arn,
    modelId=model_id,
    jobName=job_name,
    inputDataConfig=inputDataConfig,
    outputDataConfig=outputDataConfig,
)
```

← 学習データ

← モデルを指定しジョブを発行

※バッチ推論は資料作成時点では Preview の機能です

実装は公開済みです

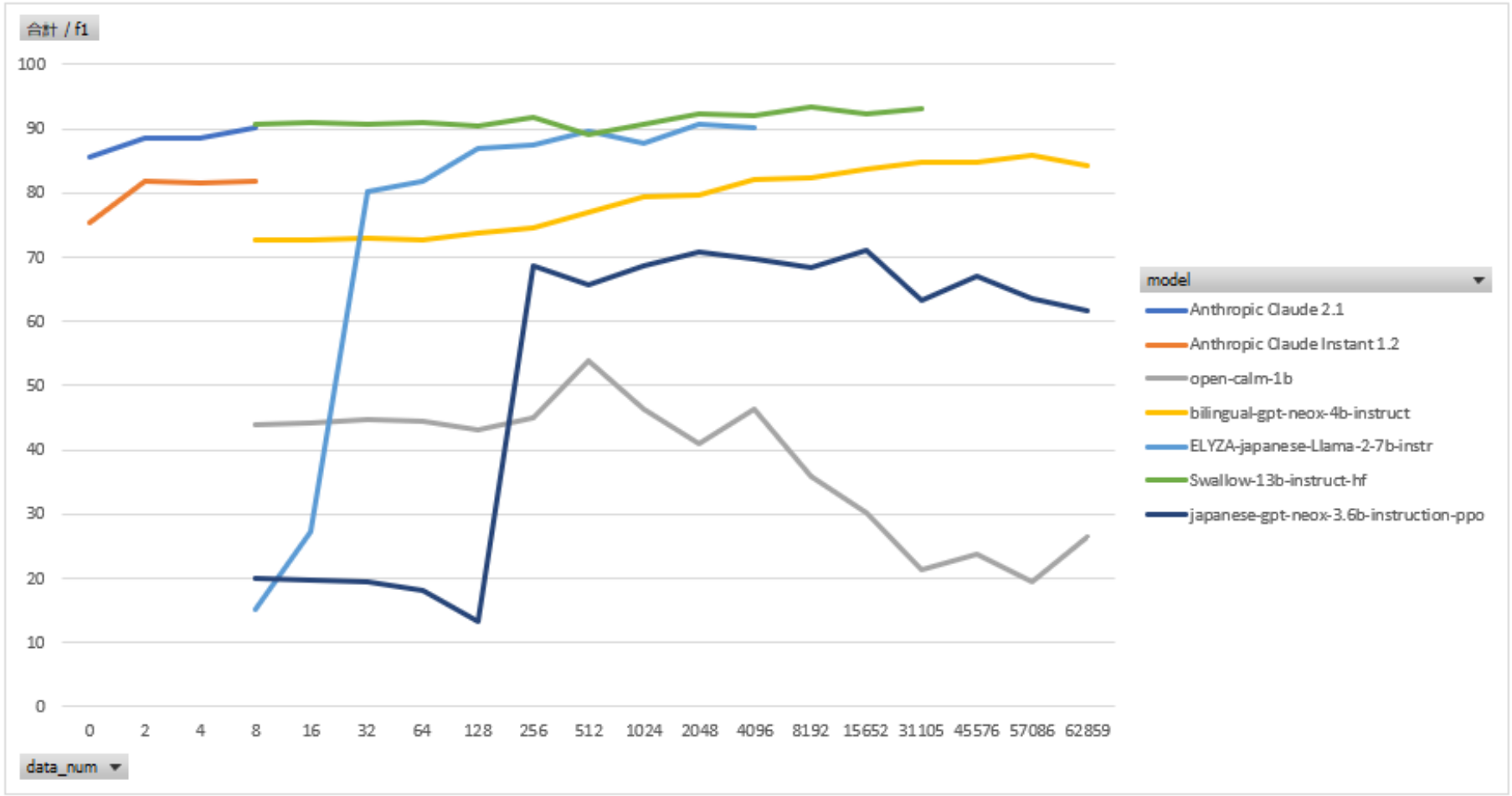


https://github.com/aws-samples/aws-ml-jp/tree/main

アジェンダ

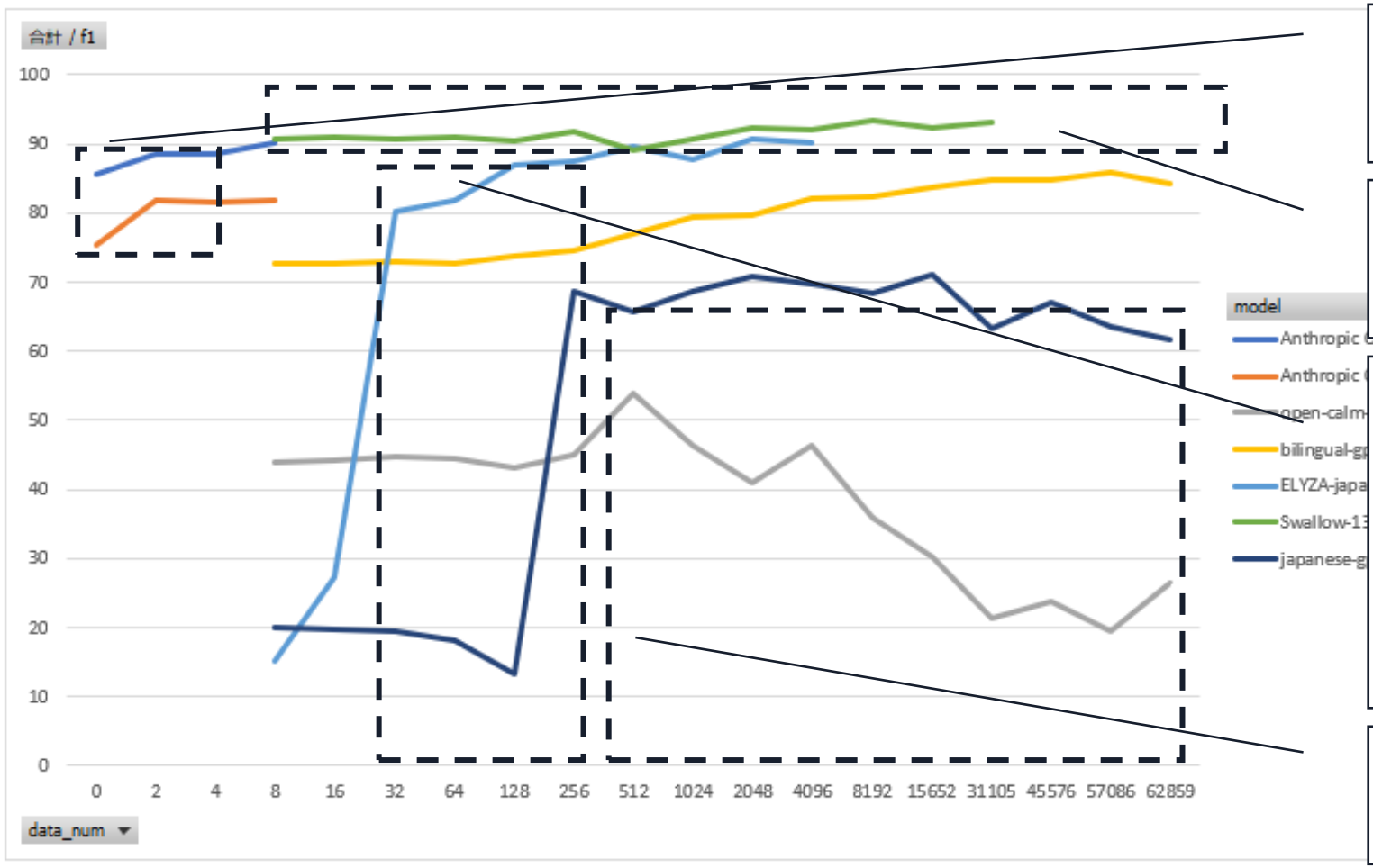
1. 会社概要
2. 取り組んだ問題の背景 / 先行研究
3. 課題解決のための実験設計
4. 実装
5. **実験結果**
6. 今後の展望

JSQuAD の精度を、学習データを増やしながら計測した結果



縦軸は F1 、横軸は使用した JSQuAD の学習データの件数 (対数)

JSQuAD の精度を、学習データを増やしながら計測した結果



Few-shot の恩恵は 2 件でかなりの部分が得られる。

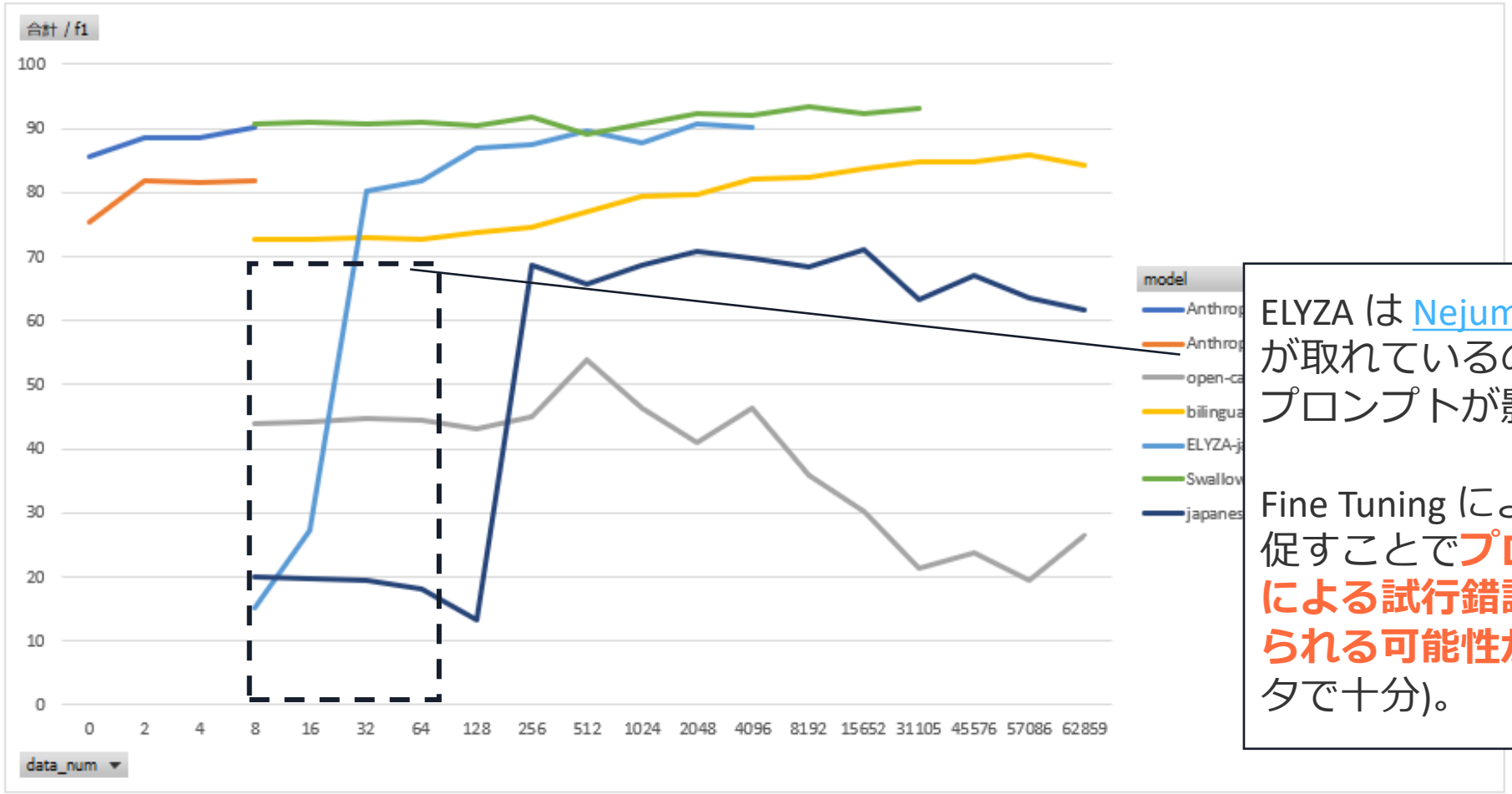
13B の Swallow は学習なしの段階で Claude 2.1に匹敵する精度

7B の ELYZA は 30 件~ で Claude Instant 、500 件~ あれば Claude 2.1 に及ぶ精度。4B の bilingual-rinna もデータを入れるほど精度が上がるが、 Claude Instant に追いつくために必要なデータは数千件のオーダーになる

3B の rinna 、 1b の OpenCALM は過学習？により精度の減衰が見られる

縦軸は F1 、横軸は使用した JSQuAD の学習データの件数 (対数)

JSQuAD の精度を、学習データを増やしながら計測した結果

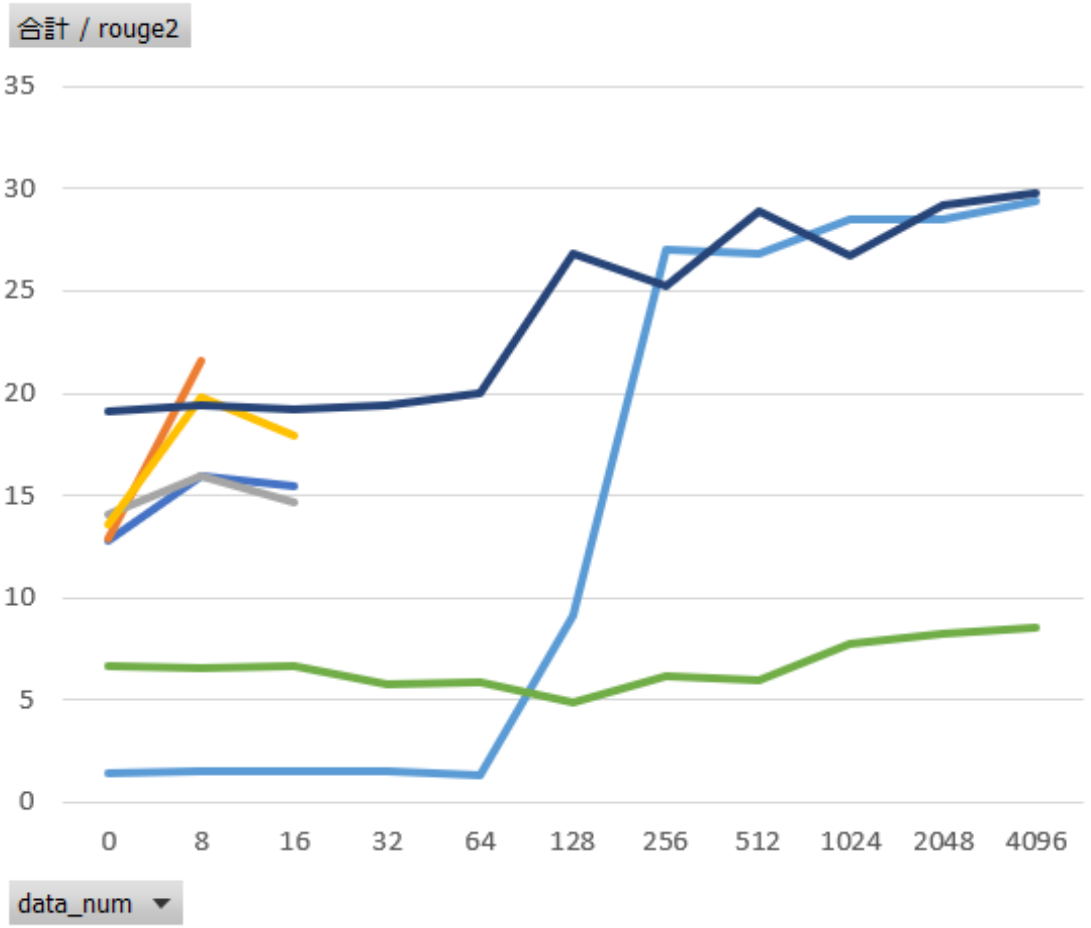


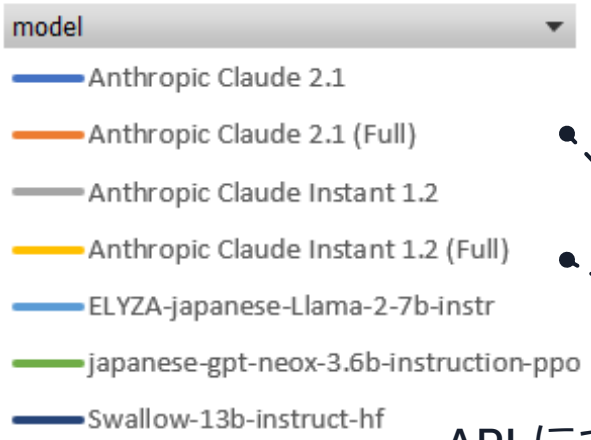
ELYZA は Nejumi Leaderboard では 0.5352 が取れているので、初期低いのは評価用プロンプトが影響している可能性あり。

Fine Tuning によりプロンプトへの適応を促すことで**プロンプトエンジニアリングによる試行錯誤より効果的に精度を上げられる可能性がある** (30~件ぐらいのデータで十分)。

縦軸は F1 、横軸は使用した JSQuAD の学習データの件数 (対数)

XLSum-ja の精度を、学習データを増やしながら計測した結果

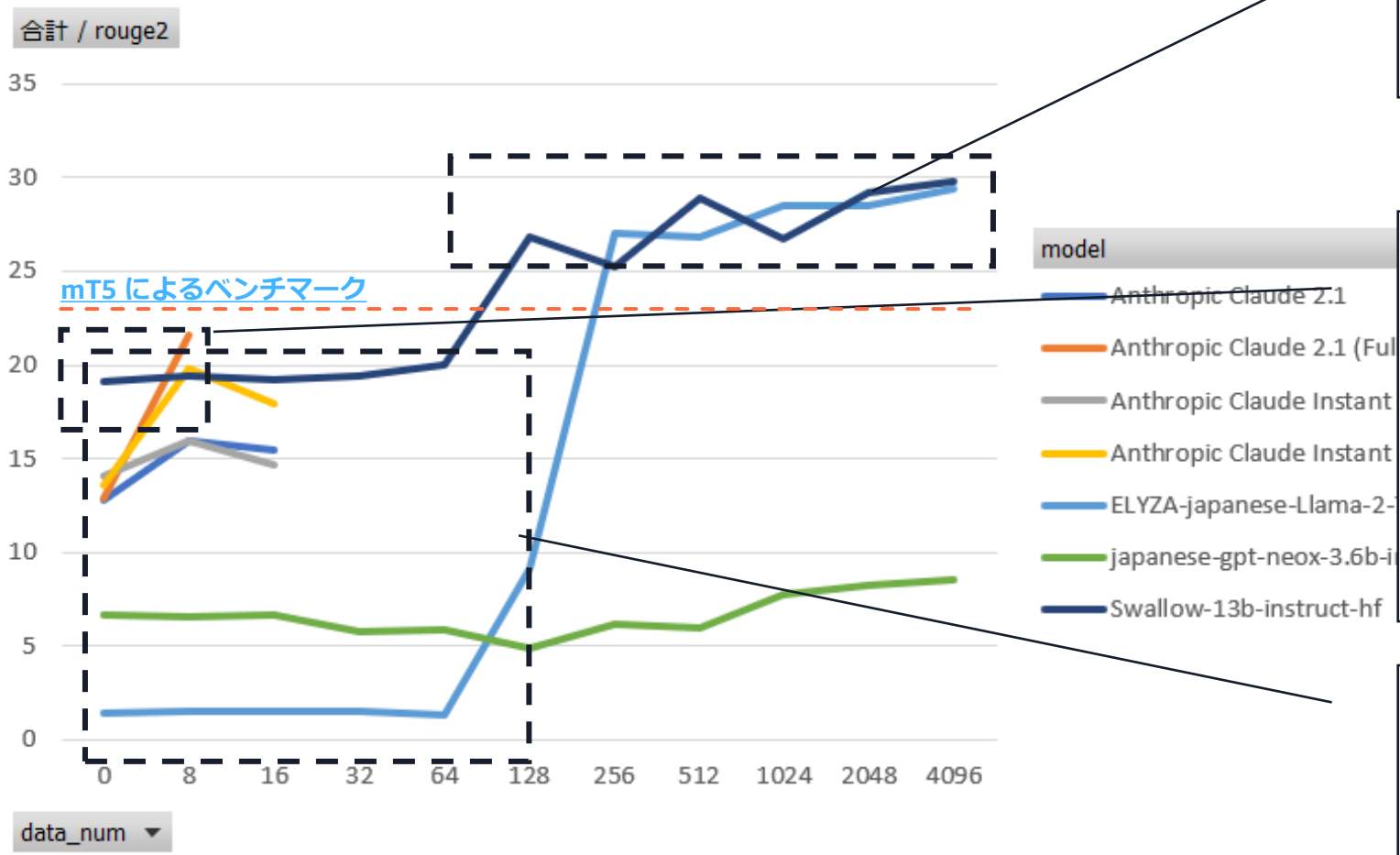




API については、要約結果のみ Few-shot で与える場合と、要約元文書まで与える Full の 2 つで検証。
[XLSum の Rouge2 スコアは、現時点のトップは 27 くらい]

縦軸は ROUGE2 、横軸は使用した XLSum-ja の学習データの件数 (対数)

XLSum-ja の精度を、学習データを増やしながら計測した結果



13B の Swallow 、 7B の ELYZA は 128 件以上のデータで **Claud 2.1 より高いスコアに到達**

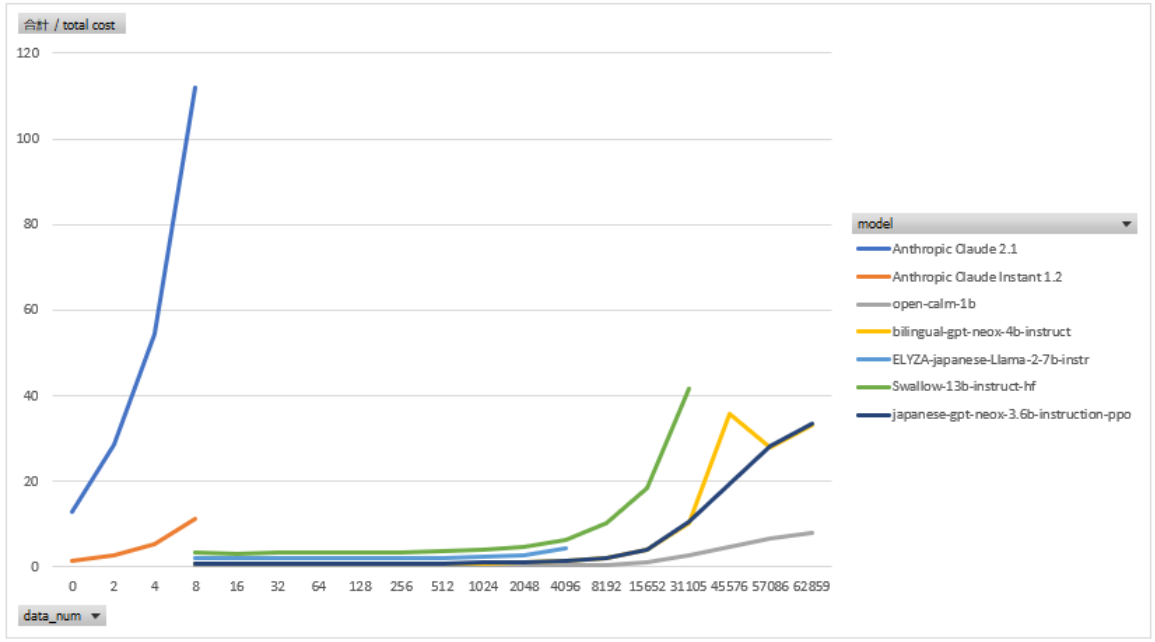
スコア上、 API のモデルは要約が得意ではないように見える。
要約元の文章を含む例示を与えないと Few-shot の恩恵が少ない (= **API のモデルで要約を制御する場合、コスト高になる可能性がある**)

60 件を超すまで ROUGE2 にほぼ反応が見られない。要約は一定量のデータが必要な可能性あり。

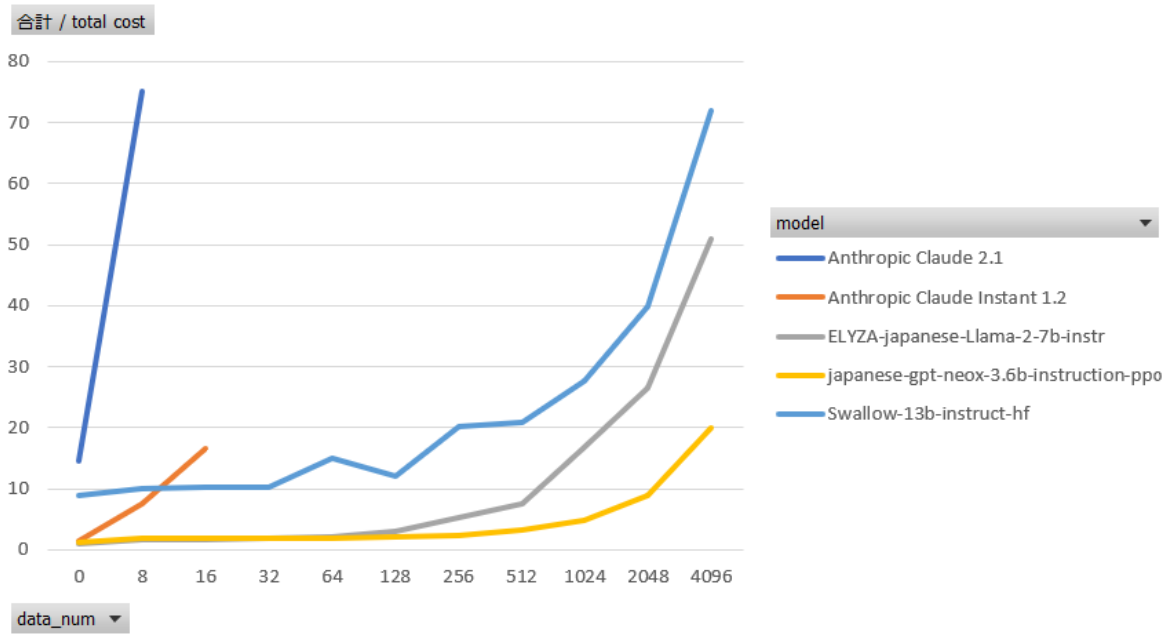
縦軸は ROUGE2 、 横軸は使用した XLSum-ja の学習データの件数 (対数)

API の Few-shot 推論、公開モデル学習 + 推論にかかるコストの比較

JSQuAD



XLSum-ja



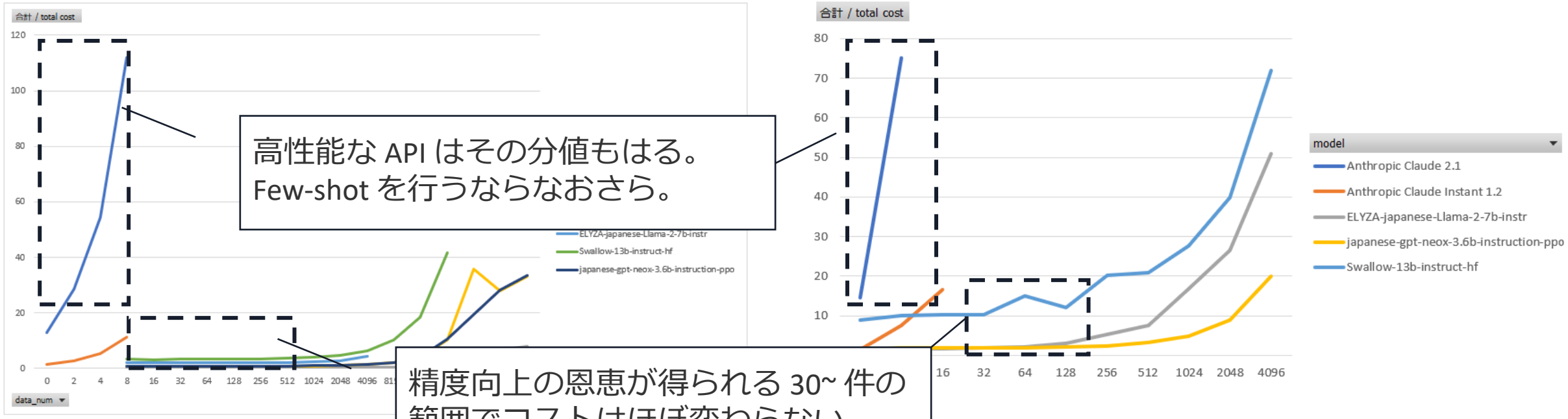
※コストはオンデマンド価格で計算しており、スポットインスタンスの使用、さらに推論特化の AWS Inferentia2 を使用することでさらに下げられます。

縦軸は金額 ($) 、横軸は使用した JSQuAD / XL Sum-ja の学習データの件数 (対数)

API の Few-shot 推論、公開モデル学習 + 推論にかかるコストの比較

JSQuAD

XLSum-ja



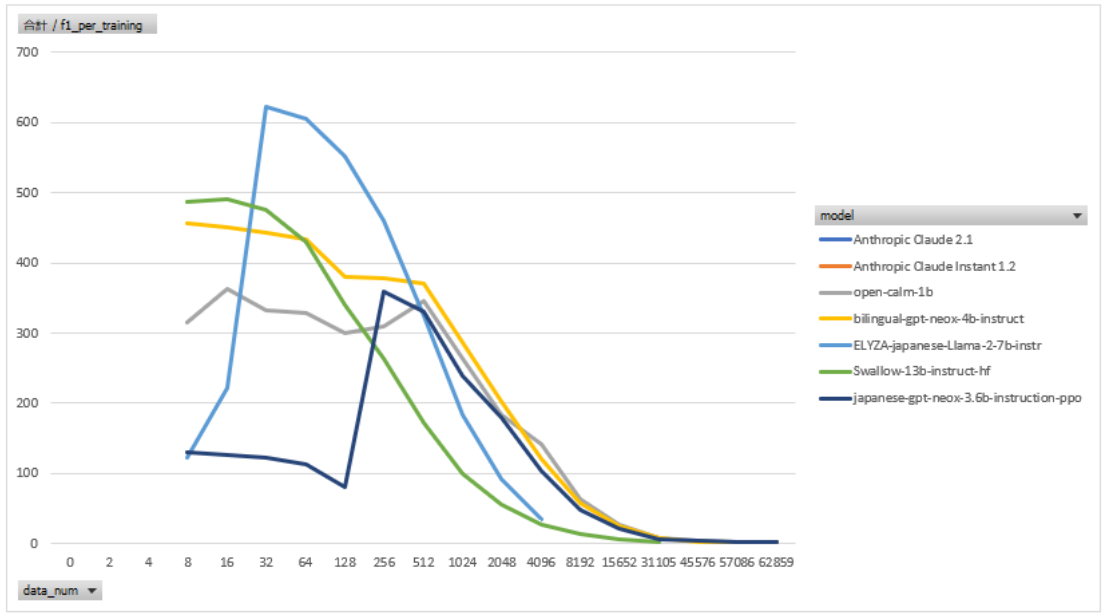
高性能な API はその分値もはる。
Few-shot を行うならなおさら。

精度向上の恩恵が得られる 30~ 件の範囲でコストはほぼ変わらない

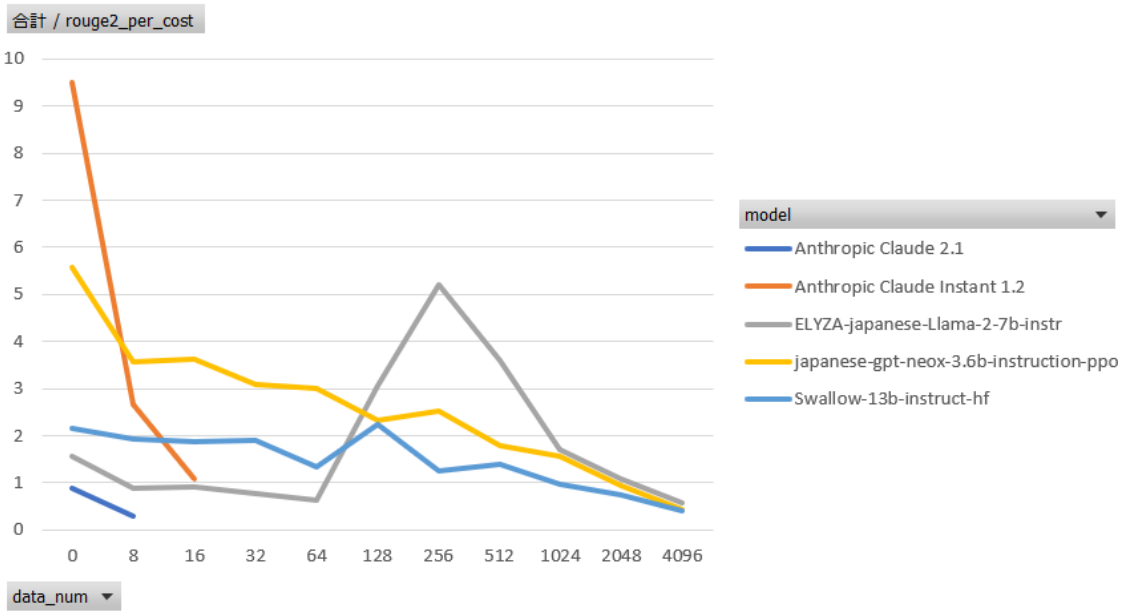
縦軸は金額 ($) 、横軸は使用した JSQuAD / XL Sum-ja の学習データの件数 (対数)

$1 で上げられる評価指標の大きさ

JSQuAD



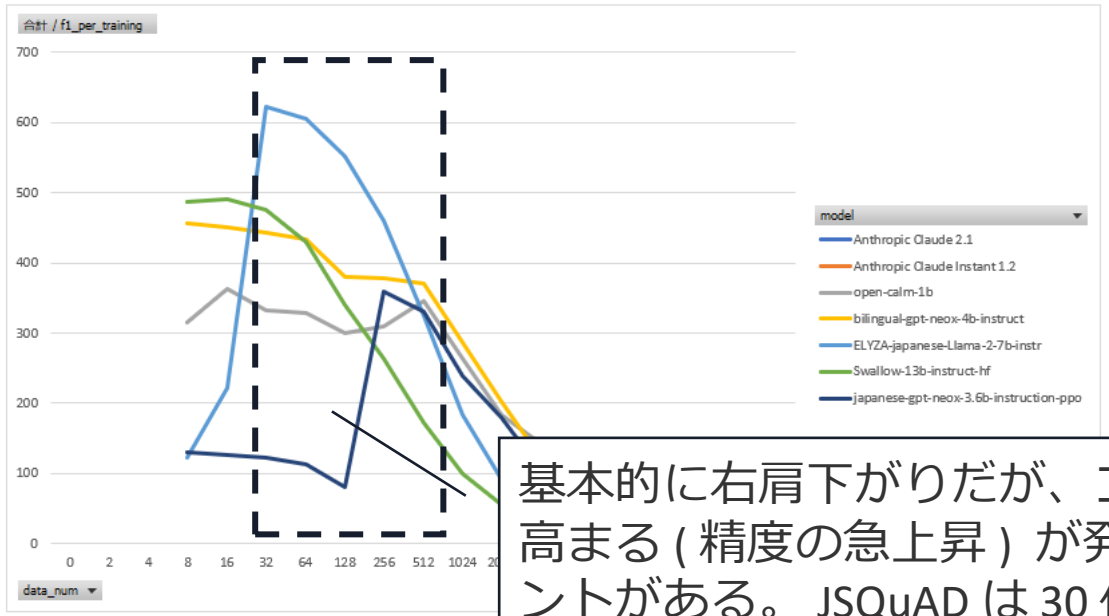
XLSum-ja



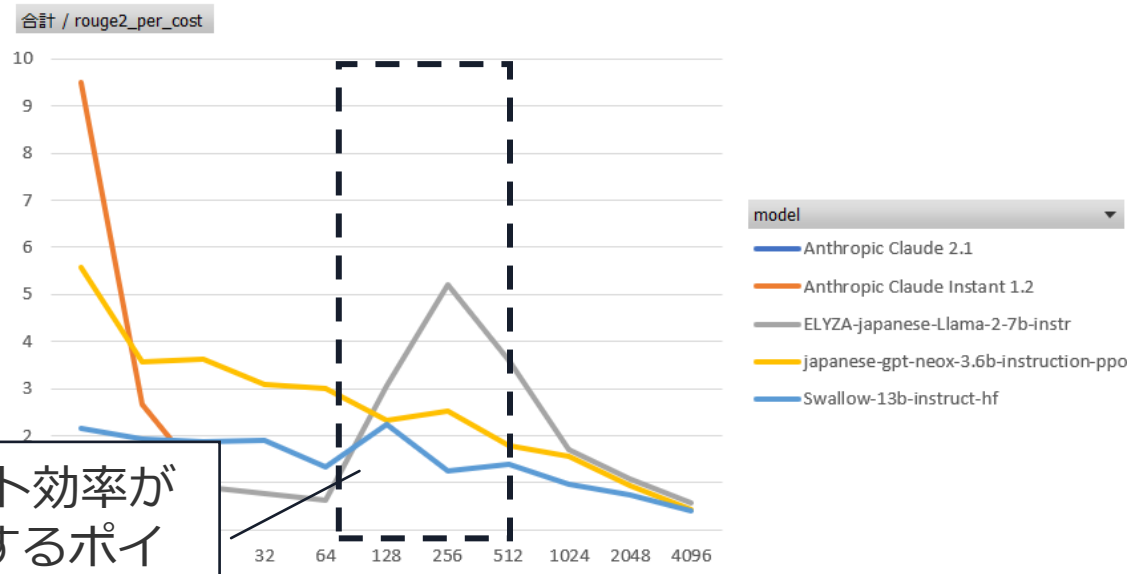
縦軸は F1 / ROUGE2 をコスト ($) で割った値 、横軸は使用した JSQuAD / XL Sum-ja の学習データの件数 (対数)

$1 で上げられる評価指標の大きさ

JSQuAD



XLSum-ja



基本的に右肩下がりだが、コスト効率が高まる (精度の急上昇) が発生するポイントがある。 JSQuAD は 30 件前後、XLSum-ja は 256 件前後。

縦軸は F1 / ROUGE2 をコスト ($) で割った値 、横軸は使用した JSQuAD / XL Sum-ja の学習データの件数 (対数)

推奨される「移行時期」 ※下記結論を一般化できるかは議論の余地あり

API は 2 件 Few-shot まで、コスト効率や安定性に不満なら 30~200 件データを用意して公開モデルの Fine Tuning へ

1. まず、 API 経由で利用し精度に課題がある場合、 2 つ程度プロンプトに例示入れることで確かな精度の向上を確認できる。
2. Claude Instant / ChatGPT 3.5 など軽量な API モデルの精度に満足している一方、速度、コスト、サービス安定性に課題を感じている場合 7B クラスのモデルを 30~200 件程度のデータで追加学習し、精度を計測してみる。
3. Claude 2.1 、あるいは GPT-4 相当の精度が必要な場合、 1) 13B クラスの OSS モデルを使用するか、 2) 7B クラスの OSS モデルを 500 件程度のデータで追加学習し精度を計測してみる。

その他の考察

プロンプトエンジニアリングと Fine Tuning (ここでは Instruction Tuning を指す) は対立的にみられることが多いが、Fine Tuning は「事前プロンプトエンジニアリング」のようにも捉えられる。

「精度を引き出すためのプロンプト」は探索空間が非常に広いが、 Fine Tuning では ELYZA の例が示す通り 30 件程度の例示でモデルの出力を API 並みの精度に高められる。 API でも Few-shot により例示に沿わせることができるが、要約の例示は原文の入力なしには精度の改善が見られず、例示による精度向上はコストがかさむ。

そのため、 Fine Tuning はプロンプトエンジニアリングよりもコストパフォーマンスに優れる可能性がある。

アジェンダ

1. 会社概要
2. 取り組んだ問題の背景 / 先行研究
3. 課題解決のための実験設計
4. 実装
5. 実験結果
6. **今後の展望**

今後の展望

① 本検証結果をもとにしたお客様 (皆様) への検証の提案

データ整備や情報抽出などバッチで良いケースでは活用余地十分

複数タスクや会話のスタイルなど複雑なケースではどうなるか？

② 7B の OSS をサーバーレス (AWS Lambda 等) で推論できないか検証

Fine Tuning 済みの 7B を API で使えたら活用シーンが広がる

③ Amazon SageMaker 等による Easy to train / deploy / call の体験実現

データとモデルからワンクリックで学習、 API エンドポイント化

Thank you!