

Smudge error: Error downloading scatter/pipeline/inputs/example-polis.csv (Object d

```
Encountered 6 files that should have been pointers, but weren't:
scatter/pipeline/inputs/togikai_gijiroku_questions.csv
scatter/pipeline/outputs/togikai-gijiroku-questions/args.csv ...
```

上記のようなエラーが発生して悩まされました。

原因

Talk to the Cityのリポジトリはサンプルデータの配布にGit LFSをつけています。しかも.gitattributeでワイルドカードでの指定がされているので、サンプルデータに限らず新しいファイルの一部はGit LFS管理下に入っていました。ところがこのことがチーム内で共有されていなかったためにGit LFSをenableした人とそうでない人が混在し、一部のファイルはポインターだけあって実体がない状態、一部のファイルはポインターのはずなのに実体がある状態になっていて、その不整合のためにトラブルが起きていました。

解決

解決策は二つありました。

案1: Git LFSを使わない側に倒す。現状のユースケースだとそれで問題なく進められると思います。が、100%大丈夫という自信がなくて、データが巨大になった時にGitHubにpushできないエラーになるはず、分析結果がどの程度のサイズまで大きくなるかが初期時点ではわかりませんでした。

案2: Git LFSを徹底する。しかし、リポジトリはすでに不整合が発生しているのでさらに見慣れないエラーが発生する可能性が高くて分析プロセスが止まるリスクがあります。

このトラブルは選挙戦序盤の時間がない時に起きたので、数日止まることが出足の遅さとして大きなダメージにつながる事が懸念されました。そこで案1を選びました。選挙戦の間に30件以上のレポートを作成しましたが、データサイズの問題は発生しませんでした。

具体的アクション: scatter/.gitattributes, scatter/pipeline/inputs/example-polis.csv, scatter/pipeline/inputs/example-videos.csv の削除

内部向けプレビューサーバの作成

自分一人で試している時には気づかなかったのですが、チーム内でTalk to the Cityを使う場合、「Aさんがレポートを作る」「Bさんがそれを見る」を実現するためには、Aさんが全世界公開することなく結果をBさんに共有する手段が必要です。

Talk to the Cityは静的レンダリングを行うので、自分の手元でプレビューするなら `python -m http.server` でサーバを立ち上げるだけでいいし、公開するなら `nojekyll` ファイルを置いてGitHub Pagesでホストすればいいです。しかし非公開のまま他人にシェアしようとしたとき、GitHub Pagesでプライベートリポジトリからデプロイしようすると有償プランが必要になることがハードルになりました。

解決

内部向けプレビューサーバを開発してNetlifyでホストしました。Netlifyでもprivateリポジトリからの継続的デプロイは有償機能なのですが、1ヶ月の試用期間があったので選挙期間中は無償で実行できました。

これにより、誰がレポートを作成してもGitHubにpushすれば数分で公開した場合と同様の見た目でのプレビューできるパーマリンクURLが生成され、Slackでシェアできるようになりました。最終的に色々なニーズに合わせて30件以上のレポートが作成されました。

おまけで一覧のページも自動更新するようにしていましたが、一覧に表示するものはレポートのフォルダ名ではなく人間可読なタイトルにした方がよかったかもね、と今回見返していて思いました。

ソースコード

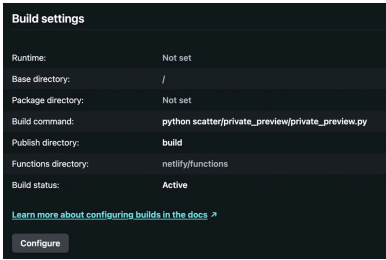
```
1 import os
2 import shutil
3 import urllib.parse
4
5 REPORT_DIR = "scatter/pipeline/outputs"
6
7 os.makedirs("build", exist_ok=True)
8
9 links = []
```

```
9     links = []
10     for target in os.listdir(REPORT_DIR):
11         indir = f"{REPORT_DIR}/{target}/report"
12
13         # outdir sholud be URL safe
14         safe_target = urllib.parse.quote(target)
15         outdir = f"build/{safe_target}"
16
17         shutil.copytree(indir, outdir, dirs_exist_ok=True)
18         links.append(f"<li><a href='{safe_target}/index.html'>{target}</a></li>")
19
20     with open("build/index.html", "w") as f:
21         f.write(
22             f"""
23 <!DOCTYPE html>
24 <html>
25 <head>
26     <title>Private Preview</title>
27 </head>
28
29 <body>
30     <h1>Private Preview</h1>
31     <ul>
32         { ''.join(links) }
33     </ul>
34 </body>
35 </html>
36 """
37         )
```

private_preview.py hosted with ❤ by GitHub

[view raw](#)

設定



日本語化パッチ

Talk to the Cityは多言語化されています。これは言語の選択肢を指定すると、それぞれの言語へLLMを使って翻訳を行う仕組みです。

ところがこの設計は暗黙に英語をメイン言語と仮定しているように見えます。日本語をデフォルトにする機能が(たぶん)なく、レポートを都民の皆さんにシェアした場合に、まず英語で表示されてから右上の言語切り替えボタンで日本語に切り替える形になりそうでした。これは説明が複雑で、正しく伝わらず、英語画面を見て離脱されるのでは、と心配になりました。

これをどう解決するかは悩ましい問題です。あらゆる人の意見を聞くブロードリスニングの思想としては、あらゆる言語で意見を受け付け、あらゆる言語に翻訳して提供するのが理想です。しかし、それには無限にコストが掛かります。

今回は初の試みなので「まずは最も低コストな方法で実験をし、実験を繰り返して改善すべきだろう」と考えました。であるなら不確実性を拡大する翻訳プロセスを介在させない方が良くと判断し、日本語のみで提供することにしました。

解決

LLMによる翻訳プロセスを介在させずに、英語で記述されているUIをデフォルトで日本語に翻訳させるために、多言語化のコードをハックしました。また、UIの日本語訳データをコードに埋め込みました。

```
1 From 18a3164dd9f51e97d9e01d9ea7f79cef7bc9a426 Mon Sep 17 00:00:00 2001
2 From: NISHIO Hirokazu <nishio.hirokazu@gmail.com>
3 Date: Sat, 15 Jun 2024 10:55:22 +0900
4 Subject: [PATCH] Make UI Japanese
5
6 ---
7 .../hooks/useTranslatorAndReplacements.ts | 68 ++++++-----
8 1 file changed, 59 insertions(+), 9 deletions(-)
9
10 diff --git a/scatter/next-app/hooks/useTranslatorAndReplacements.ts b/scatter/next-
11 index_d2b4c08..5106ead.100644
```

```

11 index 02048907.71008ad 100044
12 --- a/scatter/next-app/hooks/useTranslatorAndReplacements.ts
13 +++ b/scatter/next-app/hooks/useTranslatorAndReplacements.ts
14 @@ -4,6 +4,50 @@ import * as OpenCC from "opencc-js";
15
16   let missing: { [key: string]: boolean } = {};
17
18   +const JapaneseUI: { [key: string]: string } = {
19   +  Argument: "議論",
20   +  "Original comment": "元のコメント",
21   +  "Representative arguments": "代表的な議論",
22   +  "Open full-screen map": "全画面地図を開く",
23   +  "Back to report": "レポートに戻る",
24   +  "Hide labels": "ラベルを非表示にする",
25   +  "Show labels": "ラベルを表示",
26   +  "Show filters": "フィルターを表示",
27   +  "Hide filters": "フィルターを非表示",
28   +  "Min. votes": "最小投票数",
29   +  Consensus: "コンセンサス",
30   +  Showing: "表示中",
31   +  arguments: "議論",
32   +  "Reset zoom": "ズームをリセット",
33   +  "Click anywhere on the map to close this":
34   +    "このメッセージを閉じるには地図のどこかをクリックしてください",
35   +  "Click on the dot for details": "詳細を見るには点をクリックしてください",
36   +  agree: "同意する",
37   +  disagree: "同意しない",
38   +  Language: "言語",
39   +  English: "英語",
40   +  "of total": "合計",
41   +  Overview: "概要",
42   +  "Cluster analysis": "クラスター分析",
43   +  "Representative comments": "代表的なコメント",
44   +  Introduction: "導入",
45   +  Clusters: "クラスター",
46   +  Appendix: "付録",
47   +  "This report was generated using an AI pipeline that consists of the following :
48   +    "このレポートは、以下のステップで構成されるAIパイプラインを使用して生成されました",
49   +  Step: "ステップ",
50   +  extraction: "抽出",
51   +  "show code": "コードを表示",
52   +  "hide code": "コードを非表示",
53   +  "show prompt": "プロンプトを表示",
54   +  "hide prompt": "プロンプトを非表示",
55   +  embedding: "埋め込み",
56   +  clustering: "クラスタリング",
57   +  labelling: "ラベリング",
58   +  takeaways: "まとめ",
59   +  overview: "概要",
60  +};
61  +
62  const useTranslatorAndReplacements = (
63    config: Config,
64    translations: Translations,
65    @@ -67,15 +111,21 @@ const useTranslatorAndReplacements = (
66    const t = useCallback(
67      (txt?: string) => {
68        if (!txt) return txt;
69        - if (langIndex === 0) return replaceAll(txt);
70        - const result = translations[txt]?.[langIndex - 1];
71        - if (!result) {
72        -   if (!missing[txt]) {
73        -     missing[txt] = true;
74        -     console.log(`Missing translation for "${txt}"`);
75        -   }
76        - }
77        - return fixLocalLang(replaceAll(result || txt));
78        +
79        + // force Japanese UI
80        + const result = JapaneseUI[txt];
81        + if (!result) return txt;
82        + return result;
83        +
84        + // if (langIndex === 0) return replaceAll(txt);
85        + // const result = translations[txt]?.[langIndex - 1];
86        + // if (!result) {
87        + //   if (!missing[txt]) {
88        + //     missing[txt] = true;
89        + //     console.log(`Missing translation for "${txt}"`);
90        + //   }
91        + // }
92        + // return fixLocalLang(replaceAll(result || txt));
93      },
94      [langIndex, replaceAll]
95    );
96  --

```

上記のパッチは git am コマンドでapplyできます。

最後に

実は私は本家Talk to the Cityのコントリビュータでもあるのですが、日本人が日本語で使っていく上ではまだまだ震が多いと感じています。



ブロードリスニングは、為政者が使うことによって多くの人の声を聞くことができるようになるだけでも価値がありますが、さらに先の理想として多くの人が使えるようになることが好ましいです。多くの人の声を聞いて理解する人が増えることが、社会をより良い方向へとアップデートすることにつながるからです。

そのためにも、Talk to the City もしくはこの種のブロードリスニングのツールをもっと多くの人が使って、改善していけると良いなと思っています。ツールがオープンソースであることによって、みんなで改善していくことができます。新しいより良い社会をプログラミングによって実装できる時代が来たのです。

最後にオードリー・タンのドキュメンタリーの一節を引用します。

Q: オードリー・タンさん、世界の民主主義に対する最大の希望は何ですか？
A: 人々が民主主義を社会的技術と捉え、共に構築し、今ここで改善できるものだ と認識することです。私たちは政治システムをコーディングすることができます。

Good Enough Ancestor

