

GitHub Copilot は 何でもできる。

プログラミング用 AI で、動画制作
石崎 充良

自己紹介

石崎 充良 (@mishi_cs)

C# Tokyo コミュニティ管理メンバー

GitHub :

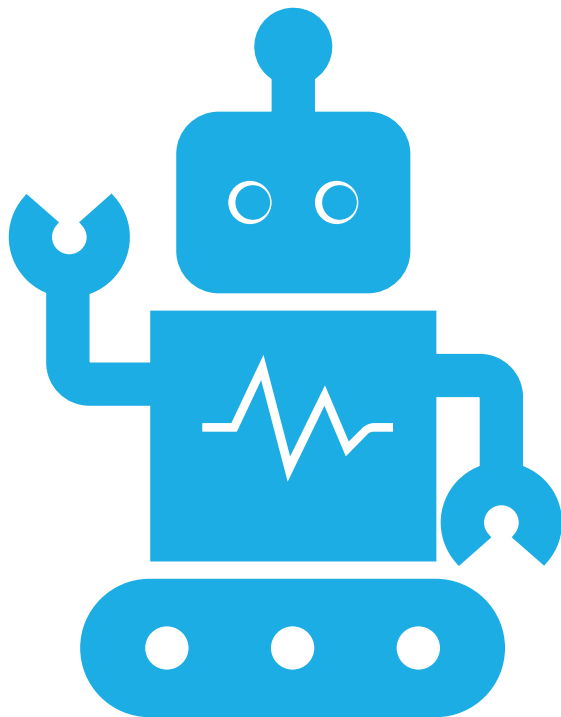
<https://github.com/m-ishizaki>

blog :

<https://rksoftware.hatenablog.com/>



「何でも」とは



- GitHub Copilot は AI
- プログラミング向けのプロダクト
- 以前は「プログラミング専用」だった
- 今は一般的な作業にもかなり素直に答えてくれる
- 規約も「プログラムだけ」と書かれていない様子 (次ページ)

規約上「プログラム限定」はあるか

結論：プログラム専用に使おうべしという規約類は見当たらなかった

	公式ドキュメント	説明
①	GitHub Terms of Service	全体の基本規約
②	Terms for Additional Products and Features	追加製品向け
③	#J AI Features	AI機能のデータ取り扱い

- ①の「C. Acceptable Use」はおおむね「違法行為に使うの禁止」。プログラム限定の記述はなし
- Copilot は②にセクションがあるが内容は薄く、プラン別のリンクが載るだけ
- そのリンク先は③へ。オプトアウトすれば学習には使われず、出力は自己責任で使う旨のみ

実際にこれ
から行う

プログラムそのものではない作業を行う

具体的には、プログラム系 YouTube 動画を作る作業

C# コミュニティ「C# Tokyo」の動画制作を行います
その素晴らしい動画制作を GitHub Copilot で！

まずは文字起こし
データを手

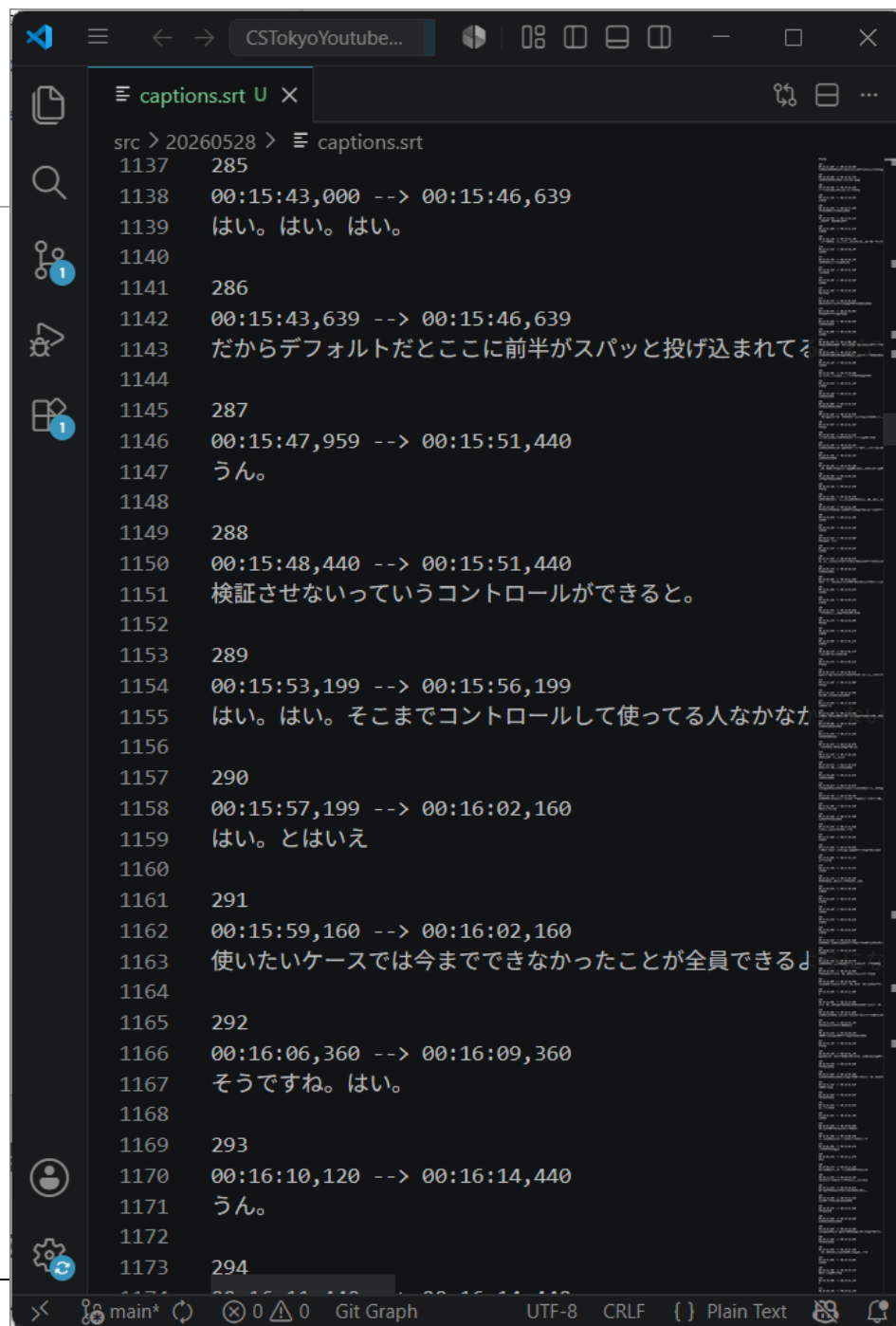
文字起こしデータを入手する

- C# Tokyo はイベントはライブ配信で限定公開し、を編集し後日動画として公開している
- ライブ配信時の文字起こしが YouTube 上に残っている
- 「字幕」内の日本語（自動字幕起こし）からデータが取得できる

YouTube Studio の「動画の字幕」画面のスクリーンショット。左側のサイドバーには「チャンネルのコンテンツ」があり、動画「【配信】.NET 11 Preview 4! C# Tokyo」がリストアップされている。中央の「動画の字幕」セクションには、字幕のリストが表示されている。リストには「日本語 (動画の言語)」と「日本語 (自動字幕起こし)」の2つの字幕がある。右側の「日本語 (自動字幕起こし)」の行には、編集、削除、およびオプションのアイコンがある。オプションアイコン（3つの点）は赤い枠で囲まれ、青い矢印で拡大されたメニューに移動している。このメニューには「ダウンロード」ボタンがあり、これも赤い枠で囲まれている。さらに拡大されたメニューでは、「.srt」オプションが赤い枠で囲まれ、青い矢印で「.srt (個人の好みです)」というテキストボックスに指し示されている。他のオプションとして「.vtt」と「.sbv」も表示されている。

文字起こしデータ

- 形式は中身を確認しやすい .srt を選択
(個人の好みです)



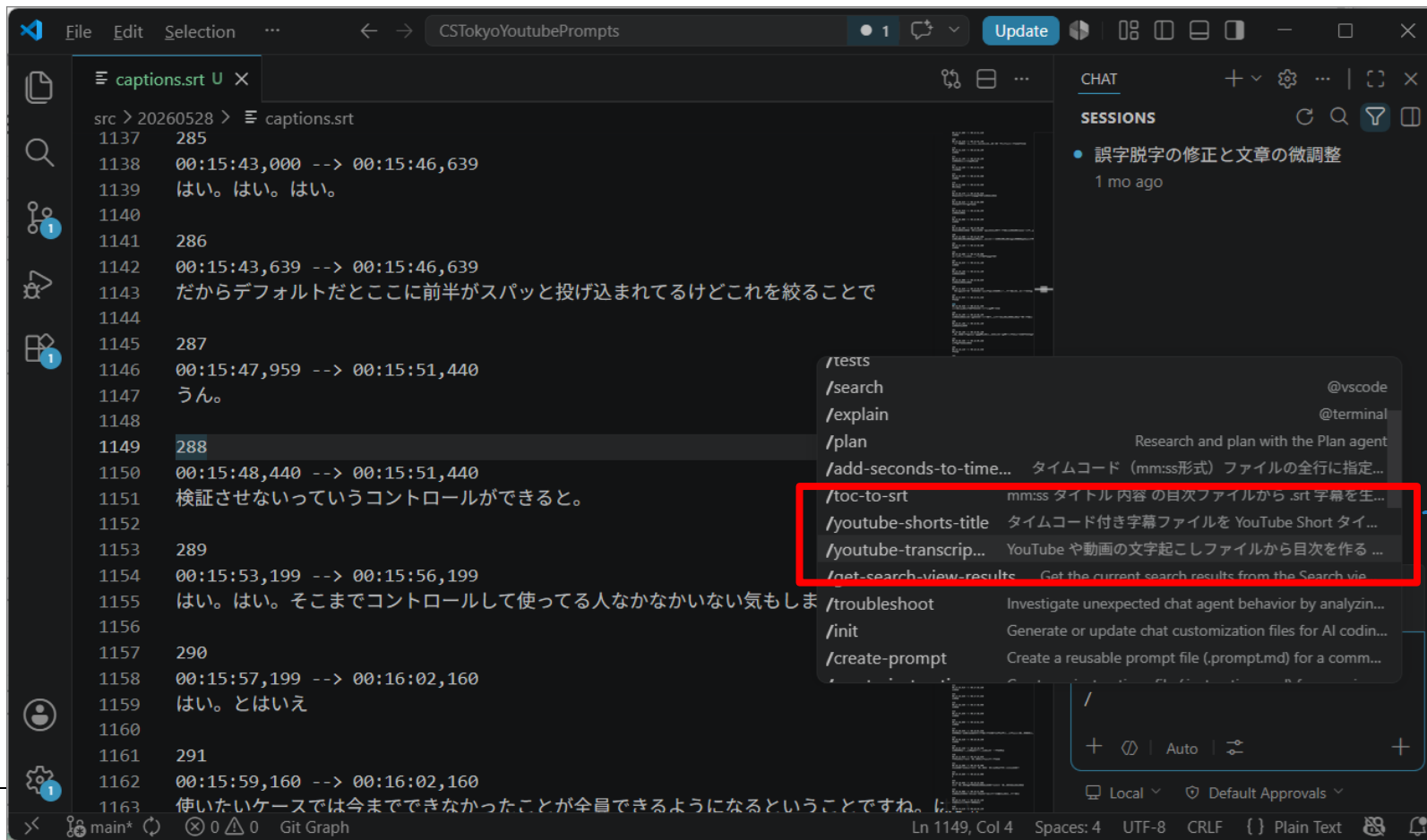
The screenshot shows a VS Code editor window with a file named 'captions.srt' open. The editor displays a list of subtitle entries, each consisting of a line number, a time range, and the subtitle text. The text is in Japanese. The editor interface includes a sidebar on the left with icons for Explorer, Search, Source Control, and Run and Debug. The bottom status bar shows 'main*' and 'UTF-8 CRLF Plain Text'.

```
src > 20260528 > captions.srt
1137 285
1138 00:15:43,000 --> 00:15:46,639
1139 はい。はい。はい。
1140
1141 286
1142 00:15:43,639 --> 00:15:46,639
1143 だからデフォルトだとここに前半がスパッと投げ込まれてる
1144
1145 287
1146 00:15:47,959 --> 00:15:51,440
1147 うん。
1148
1149 288
1150 00:15:48,440 --> 00:15:51,440
1151 検証させないっていうコントロールができればと。
1152
1153 289
1154 00:15:53,199 --> 00:15:56,199
1155 はい。はい。そこまでコントロールして使ってる人なかなか
1156
1157 290
1158 00:15:57,199 --> 00:16:02,160
1159 はい。とはいえ
1160
1161 291
1162 00:15:59,160 --> 00:16:02,160
1163 使いたいケースでは今までできなかったことが全員できるよ
1164
1165 292
1166 00:16:06,360 --> 00:16:09,360
1167 そうですね。はい。
1168
1169 293
1170 00:16:10,120 --> 00:16:14,440
1171 うん。
1172
1173 294
```

GitHub Copilot で加工する

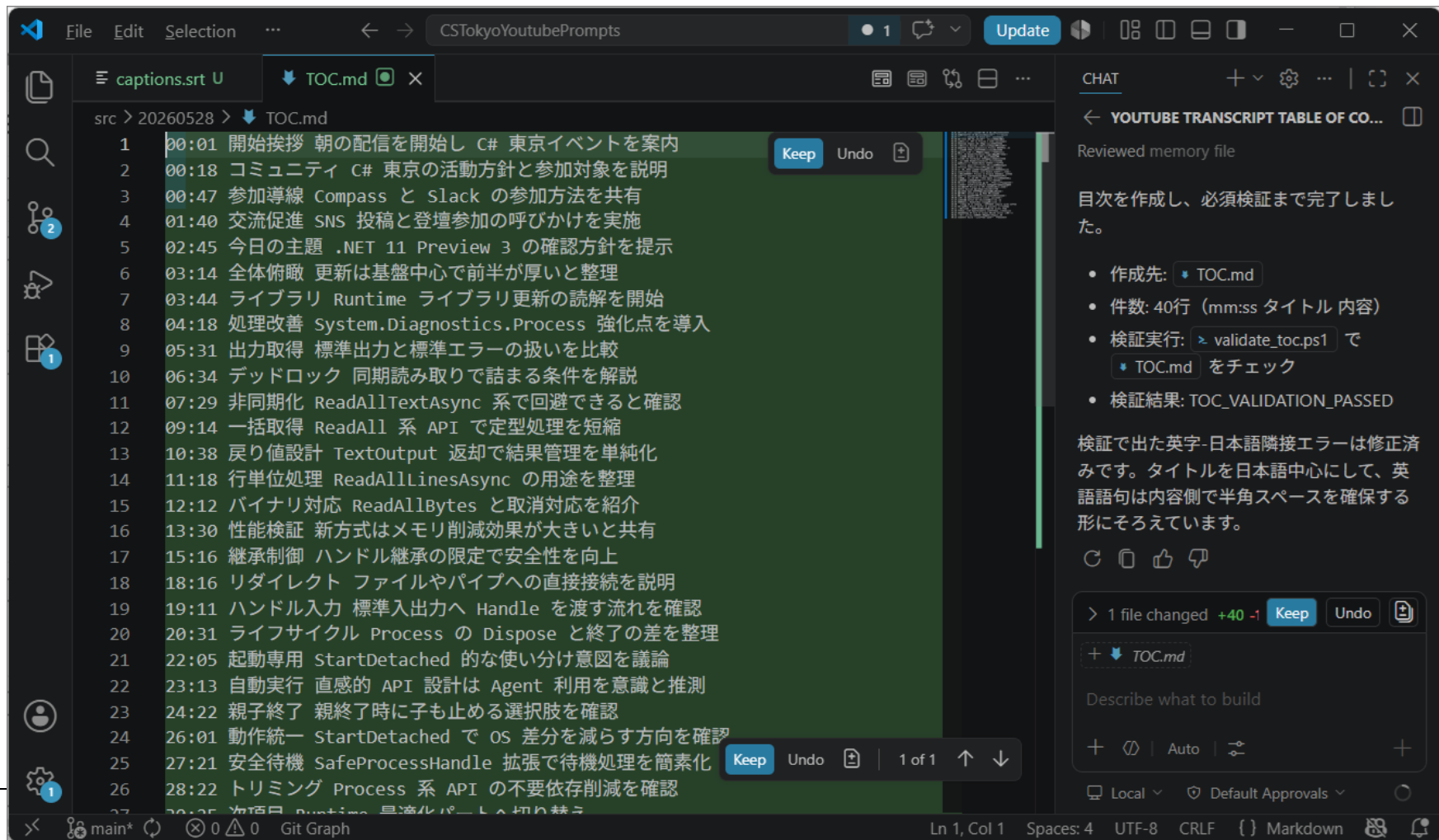
文字起こしを Skill で加工

- 文字起こしデータの加工にも GitHub Copilot は使える
- プログラミング内容なら固有名詞・用語の補正も得意
- ※自動字幕は用語のカタカナ化や不正確さがある



文字起こしを Skill で加工

- VS Code で開いたまま加工できてとても自然
- Skill を作れば一発。数分ごとの見出しを自動生成



課題：タイトルがわかりにくい

- 生成された見出しが少しわかりにくかった
- 「何についての話か」が抜け落ちていた
- 例：「処理改善」だけでは何の処理かわからない（動画には複数の技術要素があるのに主語が不明）
……追加プロンプトで再挑戦しても、まだまいち

The screenshot shows a VS Code editor window with the file explorer on the left displaying 'captions.srt U' and 'TOC.md U'. The main editor area shows the content of 'TOC.md' with a list of 13 items, each with a timestamp and a title. The first item is '00:01 開始挨拶 朝の配信を開始し C# 東京イベントを案内'. The right sidebar shows a chat window titled 'CHAT' with a message: 'YOUTUBE TRANSCRIPT TABLE OF CO... ておきます。'. Below the message, it says 'Reviewed memory file' and '目次を作成し、必須検証まで完了しました。'. At the bottom, there is a list of actions: '作成先: TOC.md', '件数: 40行 (mm:ss タイトル 内容)', '検証実行: validate_toc.ps1 で TOC.md をチェック', and '検証結果: TOC_VALIDATION_PASSED'.

src > 20260528 > TOC.md

- 1 00:01 開始挨拶 朝の配信を開始し C# 東京イベントを案内
- 2 00:18 コミュニティ C# 東京の活動方針と参加対象を説明
- 3 00:47 参加導線 Compass と Slack の参加方法を共有
- 4 01:40 交流促進 SNS 投稿と登壇参加の呼びかけを実施
- 5 02:45 今日の主題 .NET 11 Preview 3 の確認方針を提示
- 6 03:14 全体俯瞰 更新は基盤中心で前半が厚いと整理
- 7 03:44 ライブラリ Runtime ライブラリ更新の読解を開始
- 8 04:18 処理改善 System.Diagnostics.Process 強化点を導入
- 9 05:31 出力取得 標準出力と標準エラーの扱いを比較
- 10 06:34 デッドロック 同期読み取りで詰まる条件を解説
- 11 07:29 非同期化 ReadAllTextAsync 系で回避できると確認
- 12 09:14 一括取得 ReadAll 系 API で定型処理を短縮
- 13 10:38 戻り値設計 TextOutput 返却で結果管理を単純化

CHAT

← YOUTUBE TRANSCRIPT TABLE OF CO...
ておきます。

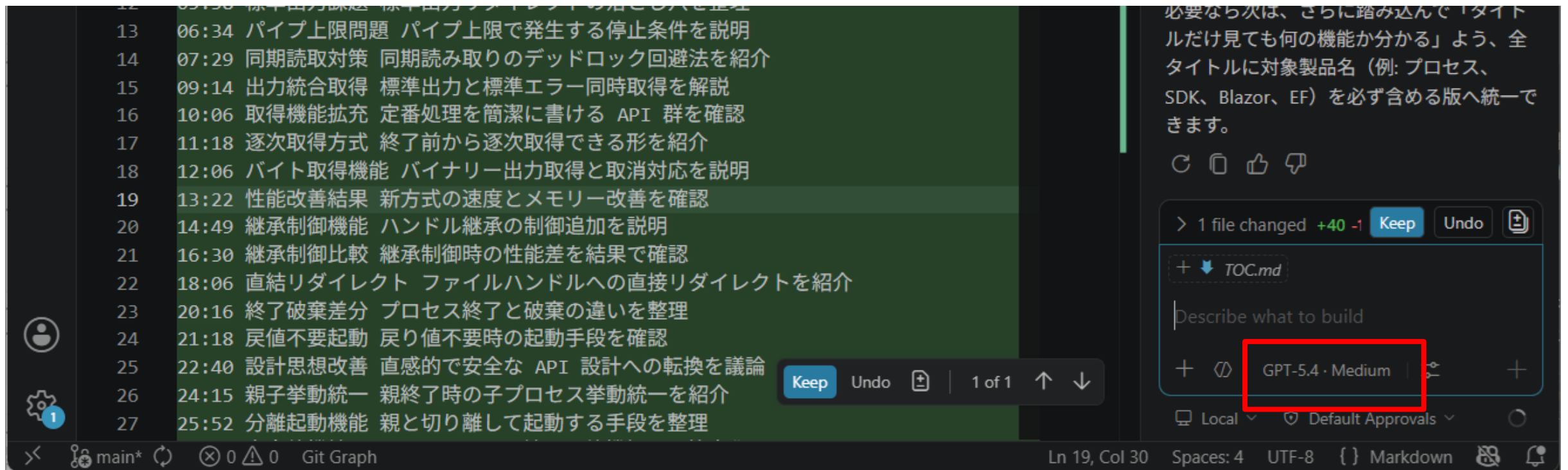
Reviewed memory file

目次を作成し、必須検証まで完了しました。

- 作成先: TOC.md
- 件数: 40行 (mm:ss タイトル 内容)
- 検証実行: validate_toc.ps1 で TOC.md をチェック
- 検証結果: TOC_VALIDATION_PASSED

モデルを変更してみる (GPT 系)

- GPT 系の少し良いモデルに変えてみる
- 見出しの精度は改善されるか
- 結果は……無念。まだいまいちだった



Claude 系で大きく改善

- さらにモデルを変更（Claude 系）、かなりいい仕上がりになった
- これがプログラミング用 AI の力！
- しかし、最後までやり切れず（60分中45分あたりまで）（対処は次ページ）

The screenshot shows a VS Code editor with a list of improvements on the left and a Claude AI chat window on the right. The list of improvements includes:

- 24 29:00 パフォーマンスまとめ Process 改善で全プラットフォームの性能と省メモリを向上
- 25 30:33 非同期ランタイム改善 async/await をランタイム側で実装し最適化を強化
- 26 32:15 ハードウェア最適化 SIMD 命令を活用した低レベル演算の最適化を追加
- 27 33:26 Ready2Run 改善 Native AOT の知見をフィードバックして起動を高速化
- 28 33:55 CoreCLR 改善 GC などランタイム細部のパフォーマンス改善をまとめ
- 29 34:33 SDK セクション dotnet コマンドの強化と新機能の概要を説明
- 30 34:43 デバイス選択コマンド dotnet コマンドでデプロイ先デバイスを指定可能に
- 31 35:39 WinUI コマンド対応 コマンドから WinUI アプリを操作する方向性が加速
- 32 37:29 シェル補完改善 dotnet の補完を Bash など他のシェルでも統一して動作
- 33 39:50 Static Web Assets SDK ASP.NET Core コンポーネント向けの発行フローを改善
- 34 40:29 OpenTelemetry 移行 アプリインサイトから oTel への標準テレメトリー移行
- 35 40:55 dotnet CLI 高速化 Native AOT エントリーポイントで特定コマンドの起動を短縮
- 36 42:33 SDK サイズ縮小 アーカイブの継続的なサイズ削減の取り組みを説明
- 37 42:45 ReadyToRun 除外宣言 Mac/iOS 向けに不要フレームワークを事前コンパイルから除外
- 38 44:29 .NET Standard 廃止 役割を終えた .NET Standard の廃止と移行方針を説明
- 39 44:58 C# 言語セクション C# 14 の shebang エラーメッセージ改善を解説
- 40 45:38 コンパイルキャッシュ C# スクリプトで同一ビルド繰り返し時にキャッシュが有効
- 41

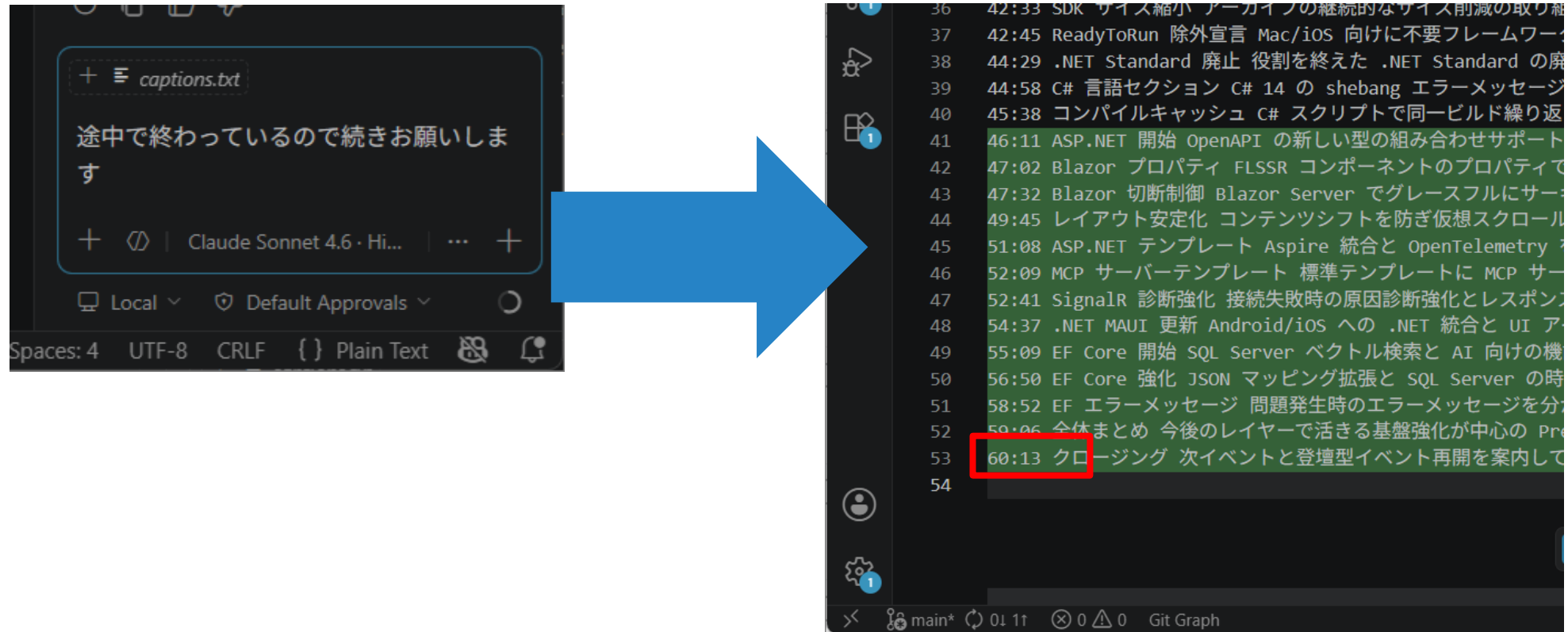
The Claude AI chat window on the right shows the following main sections:

- 00:00-02:45 — 開会・テーマ紹介
- 04:05-29:00 — Process API 大幅改善（出力取得、デッドロック解消、ハンドル制御、AOT対応）
- 30:33-33:55 — Runtime（非同期最適化、SIMD、Ready2Run、CoreCLR）
- 34:33-42:45 — SDK（dotnet コマンド強化、WinUI、シェル補完、AOT高速化）
- 44:29-45:38 — .NET Standard 廃止・C# 14 改善

The chat window also shows a prompt "Describe what to build" and a response "Claude Sonnet 4.6 · Hi...".

アドリブ力も必要

- 「続きもお願い」と頼めば最後まで生成してくれた
- このあたりは AI の使い手側にもアドリブ力が必要
- 途中で止まっても、追加依頼でカバーできる



タイムテーブルを調
整する

タイムスタンプを一括調整

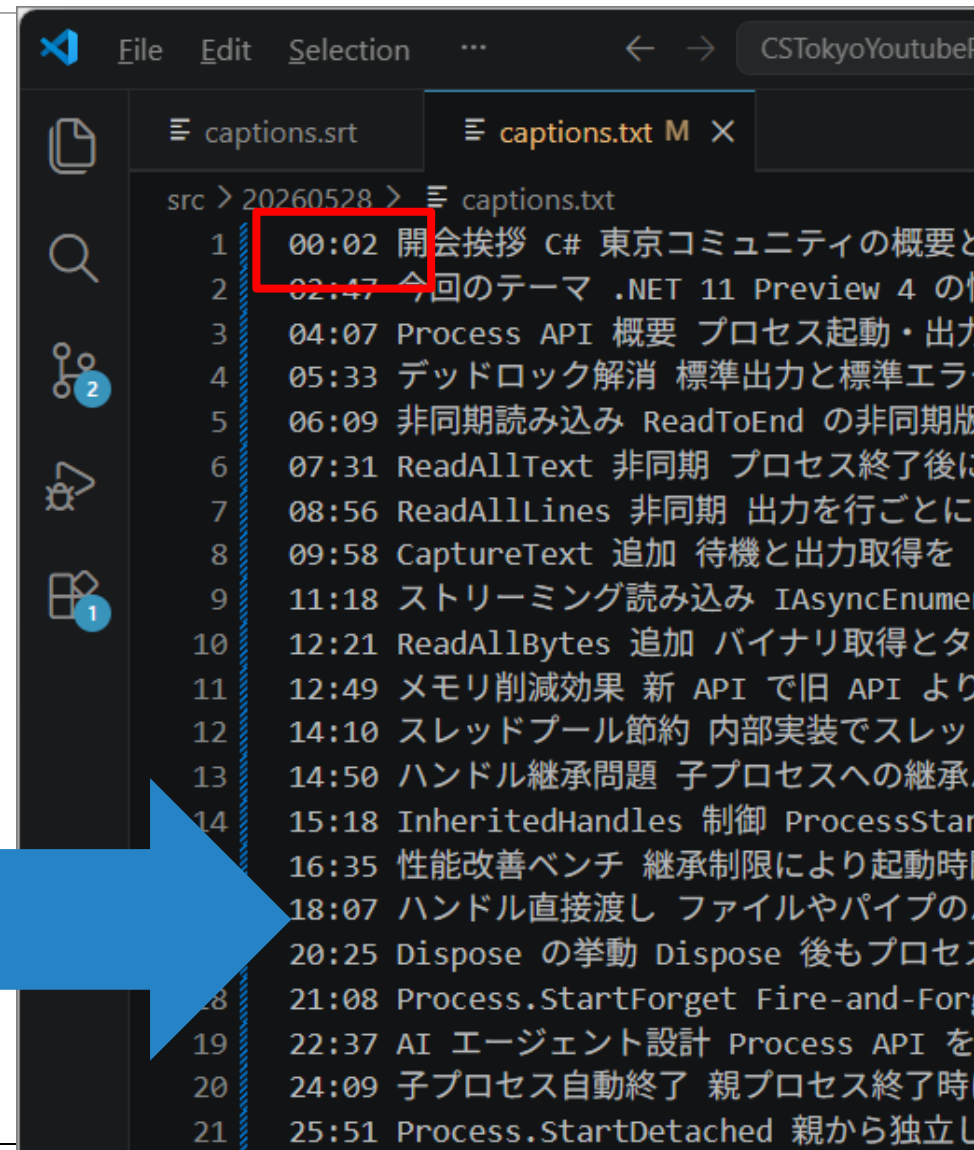
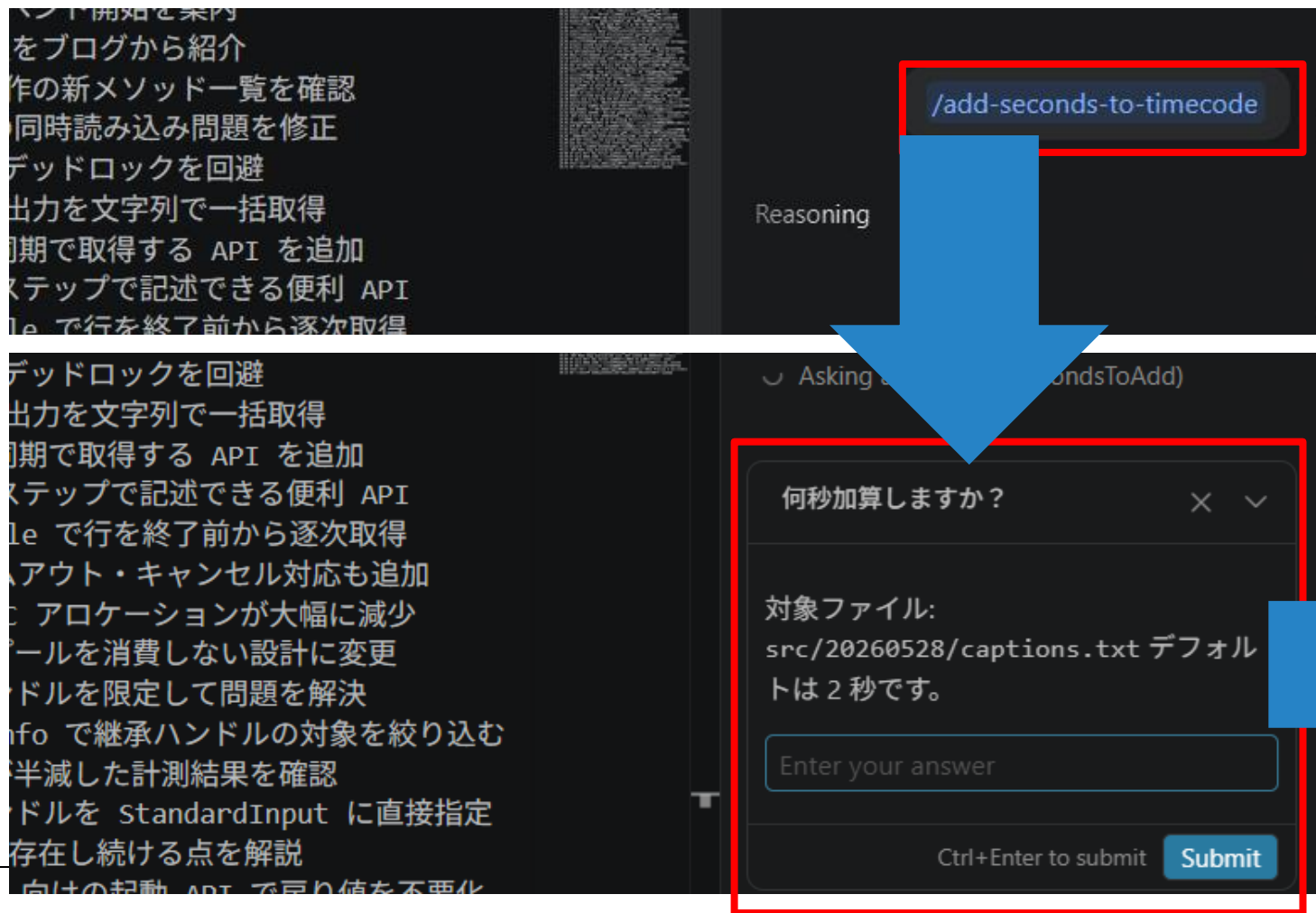
動画にするときにはオープニング（タイトルロゴ約3秒）を入れる

その結果、編集後は文字起こしと数秒ずれる

タイムスタンプの一括調整が必要になる

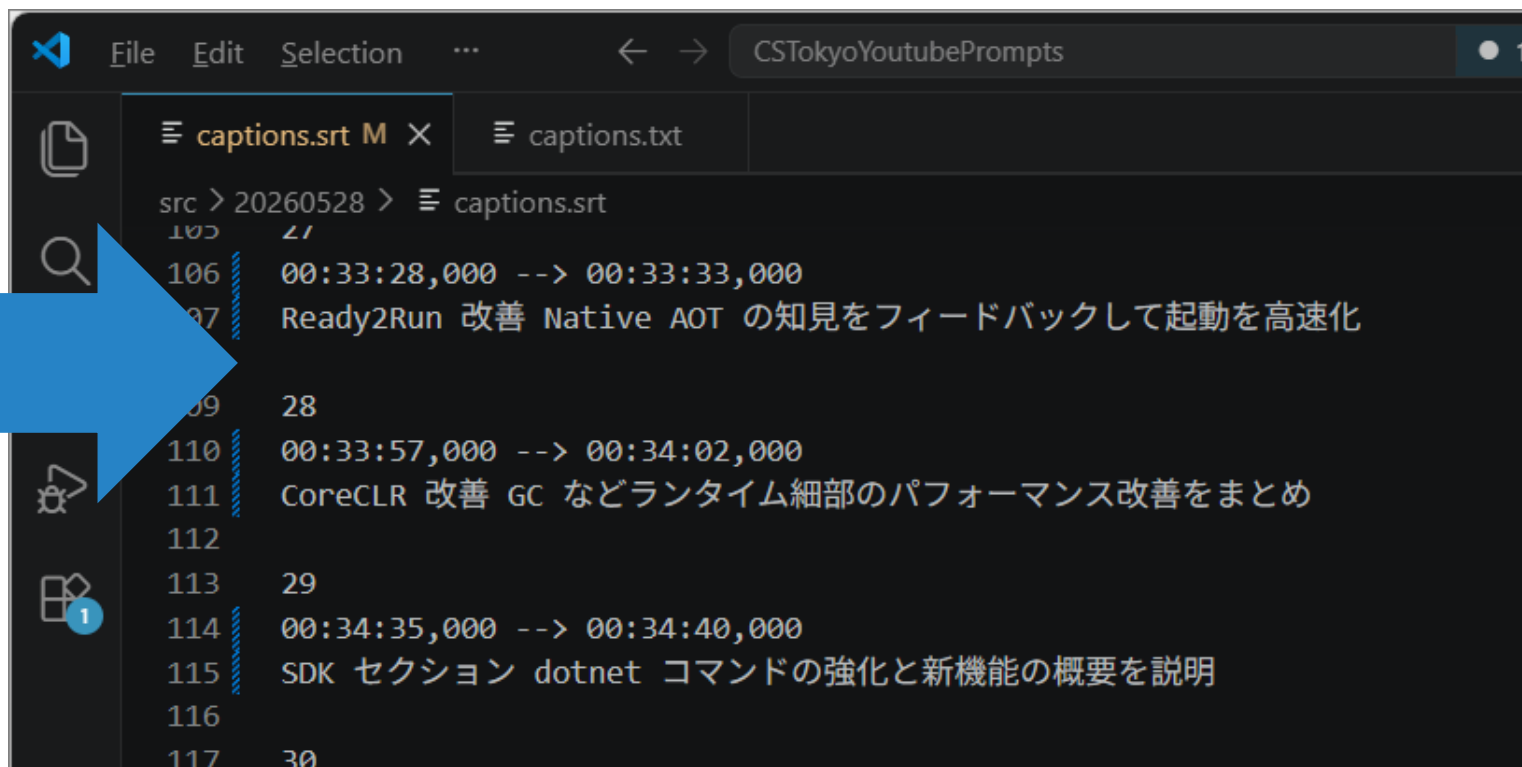
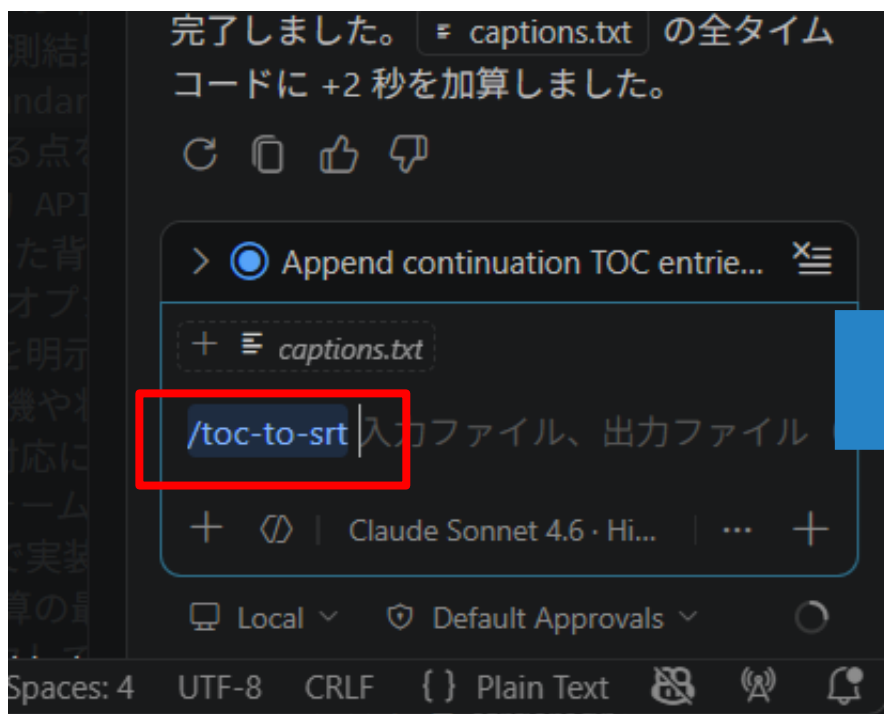
タイムスタンプを一括調整

- これも GitHub Copilot の Skill で一発
- 何秒ずらすか聞いてくれて、完璧に調整できた



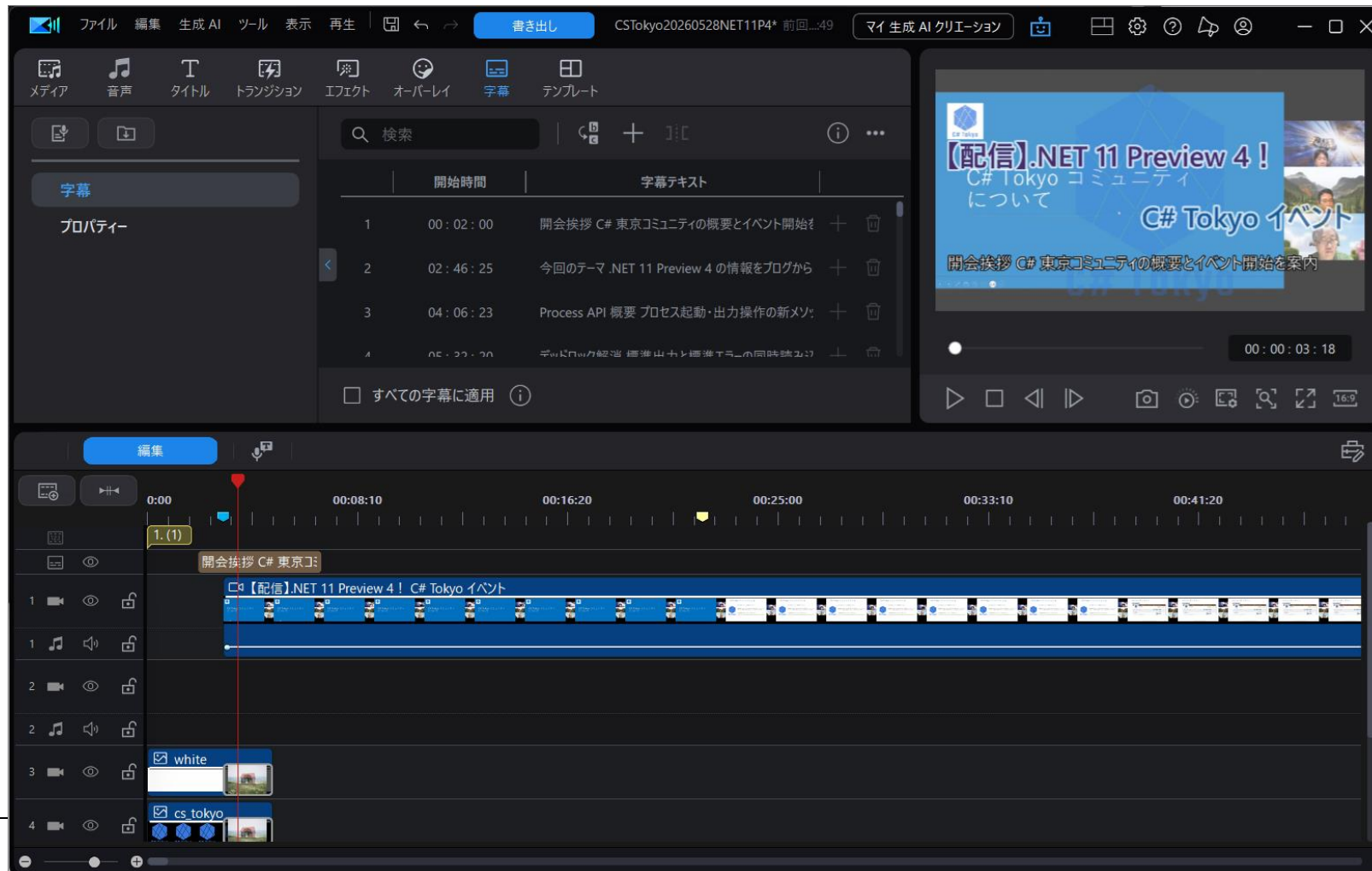
ファイル形式を変換する

- 字幕・文字起こしデータは、しょせんテキストデータ
- テキスト変換なら言語変換が得意な Copilot で十分
- ここでも Skill を用意している



ファイル形式を変換する

- 変換成功。動画編集ソフトにインポートできる
- かなり実用的な仕上がり



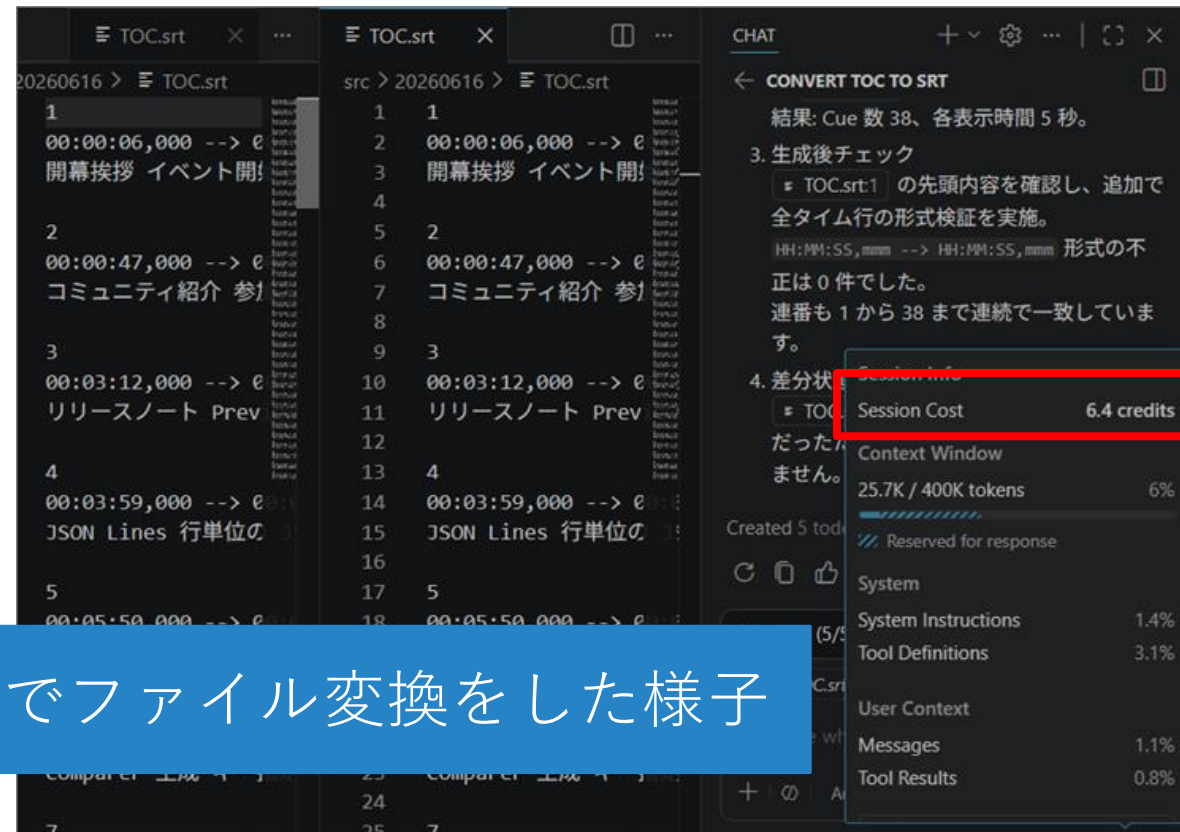
アポカリプス

2026/06 世界は AI Credits に包まれた

前セクションは 2026/05 に作成した手法です。

実際のところ、GitHub Copilot による書式変換は今でも可能。ただし、AI Credits を消費してしまう。

※2026/06 の GitHub Copilot の価格変更で AI Credits という単位での従量料金になり、多くの人がこれまでの数倍～数十倍の料金となってしまう、非常に話題になっています。



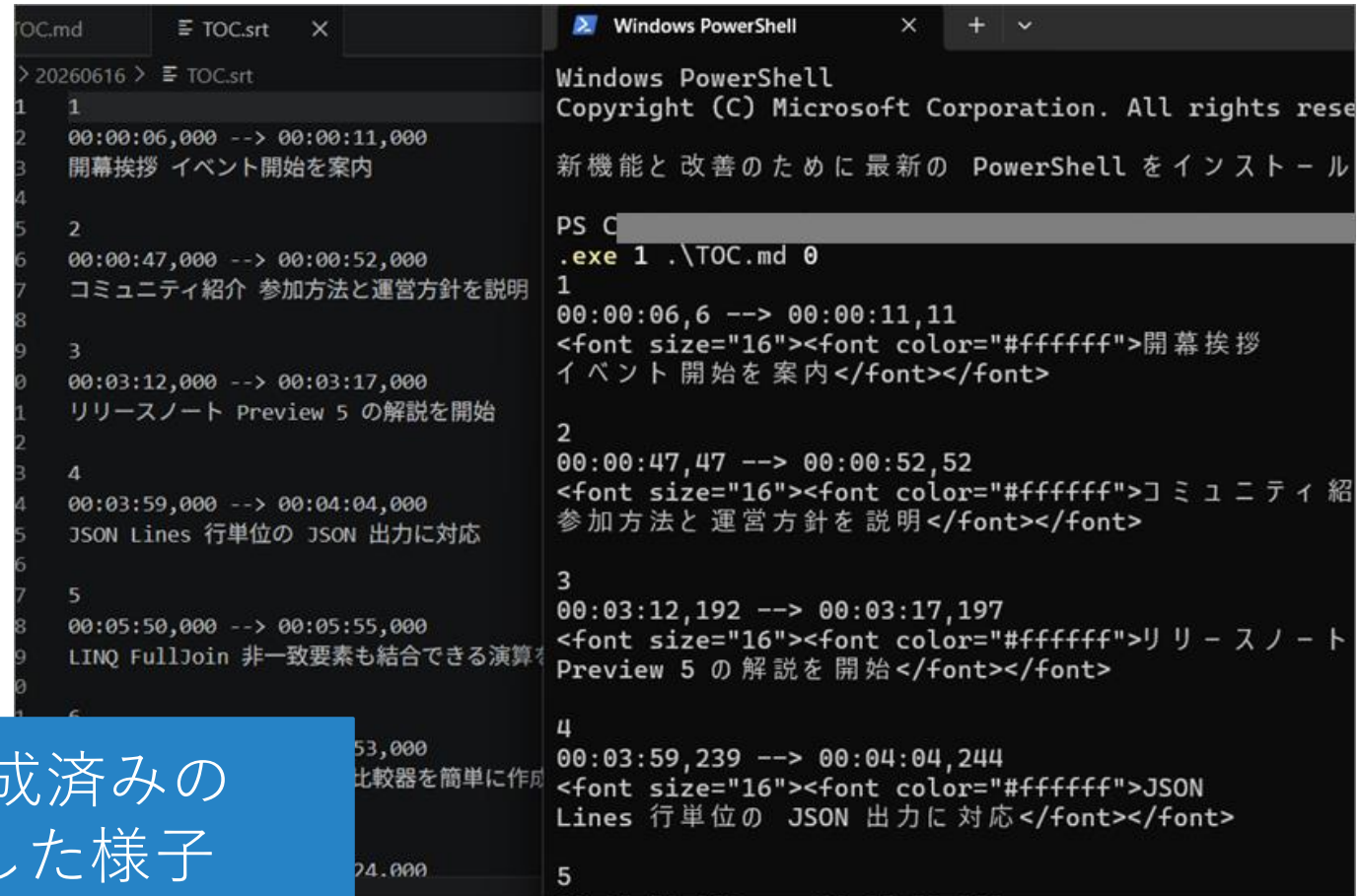
GitHub Copilot でファイル変換をした様子

ポストアポカリプスの救世主 C# コード

そこで登場させたいのが、C# コードです。

ファイル変換ツールを作っておき、
ツール側で変換することで、

AI Credits 消費はなんと驚愕の **0**。
 すごくないですか？ 画期的です。



C# コードで書いた事前作成済みの ツールでファイル変換をした様子

C# コード

小さな C# ツールで変換ルールを固定化し、毎回 AI に頼らない再利用可能な処理にします。

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4
5  namespace ToSrt;
6
7  internal class ToSrt
8  {
9      internal static string Format(long no, Line line, long start, long span)
10     {
11         var time_start_second = line.Time + start;
12         var time_end_second = time_start_second + span;
13         var time_start = Commons.TimeFormat(time_start_second);
14         var time_end = Commons.TimeFormat(time_end_second);
15         var title = line.Title.Trim();
16         var description = string.Join(" ", line.Description).Trim();
17
18         var result = @$"{no}
19 {time_start},{time_start_second} --> {time_end},{time_end_second}
20 <font size=""16""><font color=""#ffffff"">{title}
21 {description}</font></font>";
22         return result;
23     }
24 }
```

変換工程の

AI Credits 消費 **0**

変換ツールを作成してコスト削減

皆様のメリット

コスト予測：従量料金の増加を抑え、見積りしやすい

品質安定：同じルールで変換し、表記ゆれを低減

横展開：一度作ったツールを複数動画へ再利用

まとめ

結論：Copilot で動画制作もできる

- 「プログラミング専用」ではない：規約上も制限はなく、一般的な作業にも素直に応えてくれる
- Skill 化で作業が一気に効率化：文字起こし加工・タイムスタンプ調整・形式変換まで一発
- モデル選びとアドリブ力が鍵：Claude 系で品質が大きく向上し、実用レベルに到達

ありがとうございました

石崎 充良