



# 衛星データメタデータ検索エージェントの深掘り調査

## 実現方法（実装アプローチ）

- ・**自然言語から検索条件を抽出するNLP**: ユーザが入力した自由記述のテキストから、場所（AOI）、期間（日時範囲）、衛星種別、雲量条件などを解析します。例えば「2023年夏ごろの北海道東部で雲が少ない衛星画像」といった問い合わせから、**時間的条件**（2023年6-8月）、**空間的条件**（北海道東部の範囲）、**データ種別**（光学衛星）や**雲量フィルタ**（雲量<10%など）を抽出します。最新のアプローチでは、この解析にLLM（Large Language Model）を用い、ユーザの文章を**Temporal（時間）エージェント**・**Spatial（空間）エージェント**・**Collection（衛星コレクション）エージェント**・**Filter（フィルタ条件）エージェント**といった専用サブモジュールで処理する例があります。Development Seedのプロトタイプでは、「2023年のパリ上空の雲のないSentinel-2画像を探して」といった質問に対し、AIエージェントが日付や場所、センサ種別を理解しSTAC検索パラメータを組み立てています。このように、LLMにより「ユーザの意図→検索クエリ構造」への自動変換が行われます。
- ・**座標・地名の正規化（ジオコーディング）**: 自然言語中の地名や地域表現から正確な空間範囲を取得するために、ジオコーディングサービスを利用します。例えば「パリ周辺」「北海道の東側」といった入力に対し、まず地名「パリ」「北海道」を検出し、対応する緯度経度の範囲（バウンディングボックス等）を取得します。曖昧な表現（例：「東側」）については、地名から得たエリアを適宜分割したり、代表地点を選ぶなどの工夫が必要です。実装面では、オープンソースのNominatim（OpenStreetMap）やGoogle Geocoding API、あるいはDevelopment Seedが開発中の**Geodini**のような自然言語ジオコーディングAPIを使う方法があります。ユーザが入力した地名をそのまま投げて座標を取得し、STAC検索用に**bbox**（西端経度、南端緯度、東端経度、北端緯度）形式に変換する流れです。
- ・**データカタログAPIの選定（STAC API/Earth Search 等の比較）**: 衛星データ検索には、近年**STAC API**が標準化されつつあります。STACは衛星画像など時空間データのメタデータを統一構造で提供するAPIで、検索時にコレクションID（データセット種別）、時間範囲、空間範囲、プロパティ（例：雲量%）などを指定できます。例えば**Element84社のEarth Search**はAWS上のオープンデータを統合した無料のSTAC APIで、Landsat Collection 2全件やESA Sentinel-1/2など複数データセットを横断検索できます。Earth Searchでは日付範囲指定や雲量%しきい値、空間BBOX/GeoJSONフィルタといった標準機能により目的のシーンを絞り込みます。Microsoft Planetary Computerも同様に多種の公開衛星データをSTAC互換APIで提供しており、Development Seedのシステムでもデフォルトで利用されています。一方、USGSの**EarthExplorer**やESAの**Copernicus Open Access Hub**（現Data Space Ecosystem）など従来ポータルは、ウェブUI上で場所・日時を指定して検索する形式で**APIは独自**・認証必要な場合があります。例えばEarthExplorerは住所/地名や座標でエリア指定し日時範囲を絞り込むテキストベース検索機能を提供していますが、自然文によるクエリ入力には対応ていません。個人開発レベルで無料のカタログを使うなら、**Earth Search (STAC)**や**Planetary Computer (STAC)**が有力です。対して**Sentinel Hub API**は商用サービスですが、無料枠でSentinel/Landsatの検索・処理が可能で、独自のクエリ言語やEvalscriptを用います。総合すると、**STAC API**採用が汎用性・拡張性で有利であり、一つ実装すれば他のSTAC対応カタログにも容易に適用できます。

- ・**LLMエージェントによるツール利用設計**: LLMを中心据え、周辺のツール（API）を使いこなすエージェントとして設計します。具体的には、LLMに対して「まず地名を座標に変換し、その結果を使ってSTAC検索しなさい」という手順をシステムプロンプトで指示し、LLMが段階的にAPI呼び出しを行えるようにします。たとえばOpenAI GPT系モデルに対し、Pythonのツール呼び出し（関数）を組み込む形や、LangChainなどを用いて**ReActスタイル**（推論とツール実行を交互に行う手法）のエージェントを構築します。ユーザの質問を受け取ったLLMエージェントは、まず `geocode_location("地名")` を呼び出し座標JSONを取得→次に `search_items(パラメータ)` を呼び出しメタデータ一覧を取得→最後に結果を要約・推薦する回答を生成、という一連の操作を自律的に行います。JSONで得た検索結果から「最適なシーン」を選ぶロジックもLLMに組み込みます。例えばメタデータ中の雲量%や取得日時を比較して一番条件に合うものを選定したり、LLM自体に**再ランク付け (reranking)** させることも可能です。このようなツール指示付きプロンプトにより、エージェントがAPIを連携して使う高度な対話型システムが実現できます。
- ・**メモリ機能（履歴・頻出AOIの保存）**: ユーザが繰り返し利用する中で、検索履歴やお気に入りエリアを学習・保存する仕組みも考えられます。簡易な実装では、直近の対話コンテキスト内に過去のAOIや条件を保持し、ユーザが「前回と同じ場所で今度は冬の画像を」と言えば前の場所を再利用できます。より長期の個人メモリとしては、サーバ側に履歴データベースを用意し**ユーザID毎に過去クエリとそのAOI/期間**を保存しておきます。これにより**頻出する地域**（例えばユーザがよく検索する観測地点）をキャッシュして高速に再検索したり、ユーザに名前付きでエリアを保存（「マイフィールド」等）させることも可能です。LLMを使ったアプローチでは、履歴からユーザの意図を推測して補助することも考えられますが、誤推論の危険もあるため、まずは**明示的な履歴データ**として保持・検索する方が堅実です。ベクトルデータベースを用いた質問の類似度で履歴を検索することも可能ですが、開発初期では単純に全文ログや構造化クエリログを参照するだけでも十分でしょう。
- ・**MVP（最小実用製品）とフル実装の違い**: MVP段階では、基本的な**单発クエリ検索→結果提示**の流れを実現します。具体的には、単一のデータソース（例: Sentinel-2のSTAC）に対し、LLMが一回の対話で**入力文を解析→API検索→上位のシーン1件を推薦**する程度が目標です。MVPでは大規模な最適化は省き、たとえば**自然言語解析**はルールベース+LLM簡易プロンプトで実装、**地名→座標**は外部API使用、**結果の提示**もメタデータ項目をフォーマットして表示するだけなど、シンプルにまとめます。一方、フル実装では以下の高度化が考えられます。(1) **複数データソース対応**: SentinelだけでなくLandsatや他衛星も横断検索し、LLMが動的に「どのデータセットが適切か」判断。（例: レーダー画像が必要ならSentinel-1を選ぶ等）。(2) **バッチ処理・継続監視**: 一度のリクエストで複数結果を取得しユーザに一覧提示、あるいは定期実行ジョブとして例えば「毎週同じ条件で検索し新しいシーンがあれば通知」といった機能。(3) **履歴最適化**: 過去のユーザ選好から、例えば「以前選んだシーンと似た条件を優先」などレコメンド精度を上げる。(4) **対話的リファインメント**: ユーザが追加の質問「もっと雲の少ないのは？」などを投げ、それに応じて再検索・結果更新する対話をLLMがハンドリングする。(5) **スケーラビリティ**: 大量リクエストに対応するためキャッシュ導入や非同期処理、並列検索。（Planetary Computerなどは一度に数百件取得も可能だが応答が遅い場合があるため、進捗表示や部分結果ストリーム表示も検討）。MVPでは最小限の一問一答に留め、フル実装でこれら追加機能を盛り込む計画です。

## 新規性（Novelty）

- ・**既存ツールとの違い**: 現在一般的な衛星データ検索ツール（USGS EarthExplorer、ESA SciHub/Copernicus Data Spaceブラウザ、Google Earth Engine Data Catalog、NASA Earthdata Searchなど）は、基本的に**ユーザがGUIフォームに従って場所・日時・センサを選択し条件設定する形**です。キーワー

ド検索機能があっても、それはデータセット名や地名の一致によるもので、**自然言語での問い合わせ**（例：「去年の冬に北海道で撮影された雲の少ない画像を」）には対応していません。提案のエージェントは、このギャップを埋める**対話的インターフェース**です。従来は専門知識が必要だったSTACクエリ構造を隠蔽し、ユーザは**日常言語で要望を述べるだけでよくなります**。この「自然言語で検索」という点は、既存ツールにはほぼ見られない新規性です（2025年時点では一部プロトタイプを除き未実装）。

- ・**既存システムでの自然言語検索の現状:** いくつか関連する試みが近年出てきています。例えば、Development Seedによる**STAC Semantic Search**はまさにLLMを組み合わせてSTAC検索を行う実験的ツールであり、ユーザは「2023年のパリの雲がないSentinel-2画像を見せて」のように英語で質問できます。このシステムは内部でLLMエージェントがクエリを解析し、Planetary ComputerのSTAC APIから該当画像を取得するしくみです。また、Sparkgeo社のプロトタイプでは、英国のEO Data HubのSTACカタログを対象に**自然言語クエリで衛星画像を検索**し、インタラクティブ地図に可視化する試みが報告されています（2025年7月）※。さらに、Element 84社は**Queryable Earth**という研究プロジェクトで、衛星画像に対して「テキスト→画像特徴」の検索（例：「ケープコッド近くのアオコの発生場所を示して」）を**大規模ビジョン・ランゲージモデル**で実現しています。こちらは画像内容そのものをテキストで検索する先進的事例ですが、本提案のエージェントは主に**メタデータ検索**にフォーカスしており、まずは従来GUI操作を置き換える自然言語インターフェースという位置付けです。その意味で、「特定分野向けChatGPTプラグイン」のような役割を果たします。既存ツールでは地名入力程度の簡易NL対応はありますが、**複数条件を一文で指定→最適データ推薦**まで行うのは新規性が高い部分です。
- ・**LLMを組み合わせたエージェントならではの拡張性:** LLMエージェントを介在させることで、単なる検索に留まらない柔軟な対応が可能になります。例えば、ユーザのあいまいな要求に対し**追加の質問を自動で行って条件を絞り込む**、検索後に「この中でどれが一番新しいですか？」と尋ねれば即答する、といった対話的な利点があります。LLMは検索結果のメタデータを読み込んで要約・評価する能力もあるため、「このシーンは雲量5%で取得日時は午前10時です。こちらのシーンは雲量1%ですが夕方の撮影です。」のように**シーン比較レポート**を生成することも可能です。これは静的な既存検索UIにはない対話型レコメンド機能です。また、LLMに知識を持たせれば、例えば**衛星ごとの特性**（解像度やバンド情報）を踏まえて「森林観測にはLandsatよりSentinel-2の方が空間解像度が高いので適しています」といった助言も付加できるでしょう（知識グラフやプロンプト設計次第）。さらに、LLMは**非構造データ**との統合も得意です。将来的に画像内容記述や過去の分析結果と組み合わせ、「○○のような現象が映っている画像」といった高度な検索条件にも発展させられる拡張性があります。総じて、エージェント方式は**ユーザ意図に柔軟に寄り添うフロントエンド**として既存システムをラップし、新たなユーザ体験を創出する点が新規と言えます。
- ・**STACクエリ構造の抽象化と意図に基づく動的クエリ生成:** 従来、衛星データ検索APIを使うにはユーザが**経度緯度や日時を明示的に指定し、クラウドカバー等の数値を調整**する必要がありました。提案システムでは、これらの技術的パラメータを**NL→構造化クエリ変換**により自動生成します。例えばユーザが「解像度の高い衛星画像」と言えばLLMが自動で「**解像度=10m級 → Sentinel-2を選択**」といった推論をしてクエリを構築できるかもしれません。このようにユーザの意図を高レベルで理解し、それを**低レベルAPIパラメータにマッピング**すること自体が価値を生みます。ユーザはSTACや衛星毎の制約を知らずとも、自分の問い合わせに集中できますし、エージェント側で**最適なデータソースやフィルタ条件を動的に決定**できるため、従来の固定的な検索フォームを超えた柔軟性があります。この抽象化レイヤは、新規性が高い部分です。特に複数のデータカタログをまたいで統一インターフェースを提供する点は、専門家以外への障壁を下げる効果が大きいでしょう。

・関連研究・事例（LLM × リモートセンシング）の類似性と差分：2023～2025年にかけ、生成AIを地球観測に応用する動きが活発化しています。代表例としてGoogleは「**Google Earth AI**」と称し、衛星画像の質問応答や解析を行う地理AIシステムを開発中です<sup>①</sup>。彼らは強力なビジョン・ランゲージ基盤モデルと**地理推論エージェント**を組み合わせ、「**洪水後の画像で冠水した道路をすべて見つけて**」といった問い合わせに応える技術を発表しています。ESA（欧州宇宙機関）も**ChatGPTスタイルの地球観測アシスタント「EVE」**を研究中で、人間の質問から衛星データ分析まで行うチャットボットを目指しています。これらは本提案と思想は共通しますが、**画像内容の解析**まで含む点でより野心的です。対して本提案はまず**メタデータ検索と推薦**にフォーカスしており、実現ハードルは低めですがユーザの利便性向上に寄与する部分に注力しています。類似オープンソースとしては前述のDevelopment SeedのSTAC Semantic Searchや、OpenGeoAIプロジェクト内の**GeoAgents**によるSTAC検索エージェント実装があります。後者ではLLMに手順を覚えさせ、地名→BBox取得や検索API呼び出しをシステムチックに行わせる工夫が見られます。本提案はこれら先行事例を参考にしつつ、日本語を含むマルチ言語対応や、ドメイン特化の拡張など差別化を図る点が考えられます。現状、LLMとリモートセンシングの融合は黎明期であり、競合もプロトタイプ段階が多いため、この分野のフロンティアとして開発する意義は大きいでしょう。

## 課題性・リスク

- ・**入力表現のあいまいさ（地名や時期の解釈問題）**：自然言語には曖昧さが付きまといます。例えば「北海道の東側」と言った場合、どの程度の範囲を指すのか明確ではありません。北海道全域の東半分？それとも根室や釧路など具体的地域名を再特定すべきか？エージェントが勝手に範囲を決め打ちするとユーザ意図とズレる可能性が高く、**フォローアップの質問**（「北海道東側とは具体的にどの辺りを指していますか？」）が必要になるかもしれません。一方ユーザはなるべく一度で答えを得たいでしょうから、このさじ加減は難しいところです。また時間表現も、「秋頃」「去年の冬」といった曖昧表現は厄介です。年度によって「冬」は年をまたぎますし、北半球か南半球かで季節が逆転します。LLMに現在日付を与え**相対期間を解釈させること**もできますが、誤解のリスクがあります。実例として、ある試作では「季節や月が指定されたら適切な期間を割り当てる（例：秋=8～9月）」というルールを組み、曖昧な日付を機械的に決めています<sup>②</sup>。しかしその解釈がユーザの意図と常に一致する保証はありません。従って、**曖昧さをどう扱うか**は重要課題です。必要に応じ**ユーザに確認を取るUI**（候補期間を提示し選ばせる等）や、**デフォルト解釈**（例えば「去年の冬」は直近の11月～翌3月とするなど）を明示するなどの対策が考えられます。
- ・**クラウドカバー（雲量）の定義差とメタデータ限界**：衛星毎にメタデータのクラウドカバー定義や精度が異なるため、単純に数値%で比較すると誤解を招く場合があります。例えばSentinel-2では画像タイトル全体の雲被覆率がメタデータ `eo:cloud_cover` に格納されています<sup>③</sup>が、Landsatも同様にシーン全体の雲率を持ちます。しかし**クラウドマスク手法の違い**（Sentinel-2はSen2Corによる分類マスク、LandsatはCFMaskなど）や、雲に含める条件（薄い巻雲の扱い等）が異なるため、**数値の絶対比較は厳密ではありません**。ユーザが「雲量10%未満」と指定したとき、SentinelとLandsatで若干解釈がずれるリスクがあります。ただ実用上は大きな問題ではなく、**相対的な目安**として受け入れる想定です。もっと問題なのは**部分的な雲の見逃し**です。メタデータの雲量はシーン全域に対する割合なので、例えば「対象地域Aは晴れているがシーンの他の部分に雲が集中して全体では50%雲」という場合、クラウドカバー50%のため除外されてしまう恐れがあります。本エージェントは画像そのものの解析は行わず**メタデータに頼る**ため、この種の誤排除・誤採用が起こります。ユーザ意図が「AOI内が雲無しの画像」なのに、メタデータはシーン全体で判断するため完全には一致しません。これは**メタデータ検索の限界**であり、将来的にはAOIクロップ後の雲判定など高度な処理が必要になるでしょう。現段階ではユーザにその限界を説明す

る、あるいは若干緩めの閾値を使い候補を多めに提示してユーザ自身に確認してもらう対応が考えられます。

- **APIレート制限・レスポンス遅延:** スクリプト的にAPIを叩くため、レートリミットや待ち時間の問題も無視できません。Planetary ComputerやEarth Searchは比較的寛容ですが、過剰な連続リクエストはブロックされる可能性があります。また検索結果が大量になると、レスポンスが大きなJSONデータとなりLLMに与えるにはサイズオーバーという問題も出ます。例えば「全球から10年間のデータを検索」などと指定された場合、何万件ものアイテムがヒットしかねません。すべて取得するのは現実的でなく、**結果数が膨大な場合の扱い**（例えば「該当シーンが非常に多いため上位100件を対象にします」とメッセージを返す等）も設計課題です。またユーザに待ち時間を意識させない工夫も必要です。LLMを対話で使う場合、人は即座の回答を期待しますが、バックエンドでSTAC検索に数秒～数十秒かかるとストレスになります。非同期処理で「検索中...」と表示する、部分的に結果が揃ったら逐次LLMに食わせてストリーム出力する、といったUI上の工夫で緩和可能ですが、**リアルタイム性**は完全には保証できません。さらに、LLM自体のAPIも高遅延（数秒以上）なので、全体で10秒以上応答にかかるケースも考えられます。このあたりはMVP段階では割り切りつつ、ユーザエクスペリエンスの課題として残るでしょう。
- **LLMの誤答・誤操作リスク:** LLMが生成する検索条件や回答に誤りが入り込む危険も指摘されます。たとえばユーザが「ランドサットの画像」と言ったのに、LLMが文脈を誤りSentinel-2を検索してしまう、あるいは「<5% cloud」と言ったのに {"eo:cloud\_cover": {"lt": 5}} ではなく誤ったフィールド名でクエリを組んでしまう、といったツール使用ミスです。LLMは訓練データにない新しいAPI仕様については平気でハルシネーションするため、システムプロンプトで厳格にフォーマットを教え込む必要があります。それでも想定外の入力に対してはバリデーションが必要でしょう。対策として、エージェントからのAPI呼び出しは必ず検証して、エラー応答なら再度プロンプトで訂正させる、ないしユーザに「内部エラー：検索条件の解釈に失敗しました」と伝え再試行を促す、といったロバスト性向上が考えられます。また、LLMが最終回答で存在しないシーンIDやリンクを捏造してしまう恐れもあります。検索結果に基づいて回答させる場合も、引用符号や数値の単位など細かい誤りが紛れ込む可能性があります。そのため、**出力テンプレートを厳格に定義**し、例えば「Scene IDはJSONレスポンスからそのまま出力させる（LLMに再生成させない）」などの方策が必要です。誤答リスクは信頼性に直結するため、エージェントの応答を監視・ログ分析し、問題のあったケースをフィードバックしてプロンプトやロジックを修正する運用も欠かせません。
- **多言語対応の困難さ:** 本システムを日本語など多言語で使えるようにするには、LLMとジオコーダそれぞれで課題があります。LLM自体は多言語モデル（例えばGPT-4や多言語に精通したモデル）を使えば、日本語の解析もかなり高精度にできます。しかし**地名の扱い**が問題です。ジオコーダは多くの場合英語や主要言語の地名データを持ちますが、日本語ローカルな通称や表記ゆれには弱いことがあります。例えば「秩父山地」などを英語の地名データで引いても出ない可能性があります。この場合、LLMに地名を翻訳させてしまうと誤訳のリスクがあるため危険です（「秩父」を無関係な言葉に翻してしまう等）。理想は**ジオコーダ自体が日本語対応**していることですが、Nominatimは主要言語の別名もある程度扱えるものの万能ではありません。GoogleのGeocoding APIは日本語も通りますがコストがかかります。さらに、クラウドカバーなど専門用語を日本語でユーザが書いた場合（例：「雲量一割未満」）、LLMが正しく"cloud cover <10%"と解釈できるか、といった問題もあります。多言語対応は一気にハードルが上がるため、**まず英語のみ対応→日本語は将来対応**とフェーズ分けするのも現実的選択です。ただ国内利用を考えると日本語対応は外せない要件でもあるため、対策として**ユーザ入力をバックエンドで英訳→英語LLMで処理→結果を和訳**というパイプラインも検討されます。この場合、翻訳誤差で意図がねじ曲がるリスクがあります。従って、可能なら**多言語に堪能なLLM**（多言語モデル or 日本語特化モデル + 英語モデルのハイブリッド）を用い、ジオコーディング部分だけ慎重に設計（地名はそのまま投げ、ローマ字なども試す）す

る必要があります。要約すると、**多言語対応は高コストかつ高リスク**であり、MVPでは单一言語に絞ることも視野に入るべきでしょう。

- ・**メタデータ検索のみの場合の限界:** 本エージェントは画像そのものの解析や表示は行わず、あくまで **メタデータ（付帯情報）** にもとづき「最適」シーンを推薦します。そのため、**画像内容に関する要求**には応えられません。例えば「洪水の被害が写っている衛星画像を探して」という問いは、メタデータ上には洪水の有無など記載がないため対応不能です。また例えば「この地域の変化が分かる画像を」と言われても、実際には複数日の画像を比較しなければ分からないので、本システムの範囲外です。要するに、**本エージェントは所望データを探す所までの役割**であり、分析・判断自体は人間または別システムに委ねることになります。これ自体は「検索エージェント」という定義からすれば問題ありませんが、ユーザの期待値次第では**物足りなく映る可能性**があります。昨今のAIブームで、「衛星画像も分析してくれて答えを出してくれる」ことを期待するユーザには、本システムは「**ただデータを見つけてくれるだけ**」なのでギャップがあります。従って将来的な発展として、後述するような**画像解析エージェントとの連携**や、ユーザに対しては「**これはデータ検索ツールであり解析結果を出すものではない**」旨の説明が必要でしょう。加えて、メタデータに依存するために発生する限界（例：前述のクラウドカバーや日時精度の問題）も合わせ、**ユーザ教育やUI上の注意書き**によって正しく理解してもらう配慮が求められます。

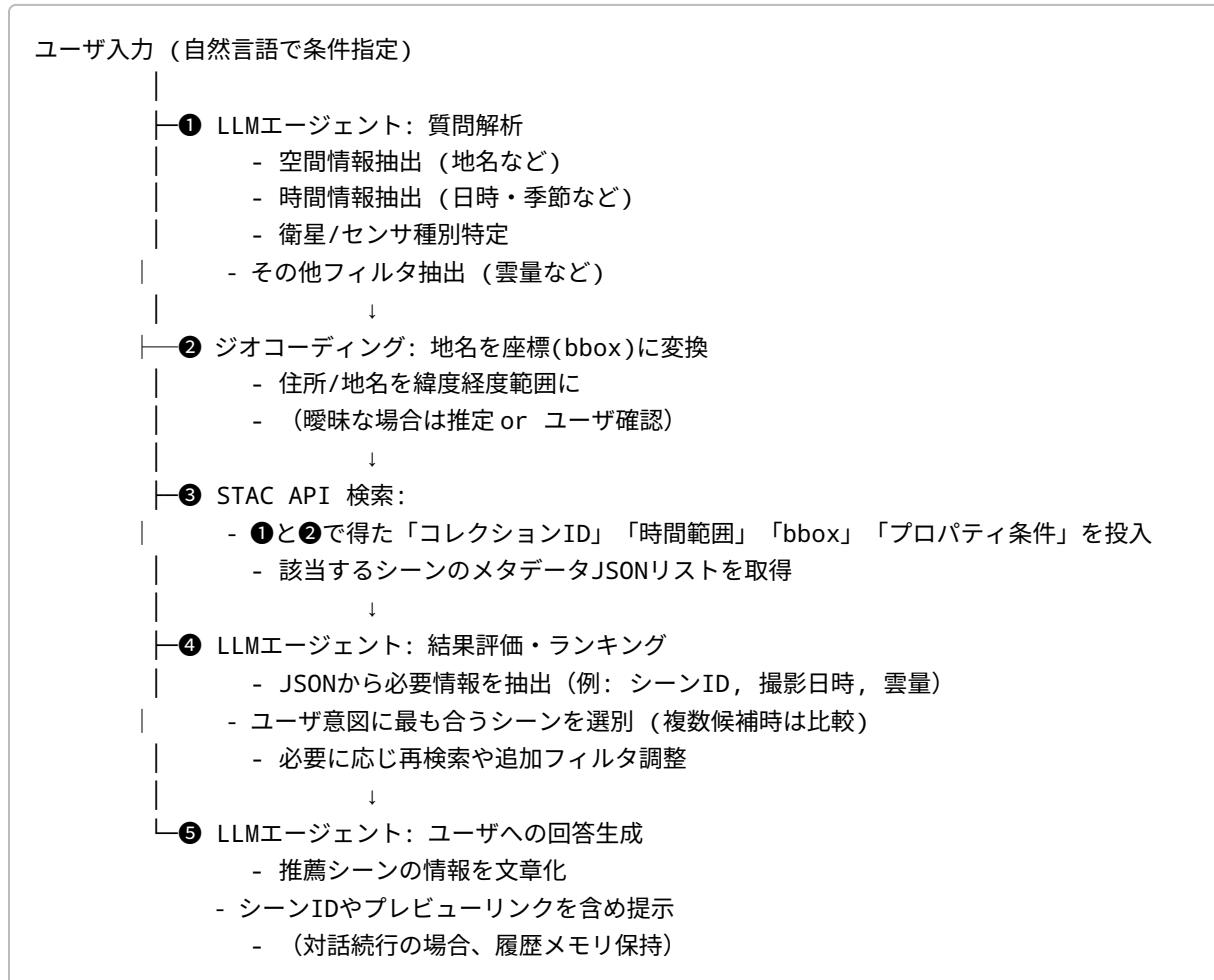
## 応用性・発展案

- ・**自動前処理エージェントとの連携:** データ検索エージェントで見つけたシーンに対し、さらに自動処理を行うエージェントを組み合わせれば、**ワンストップで解析結果を提供**することも夢ではありません。例えば、検索エージェントが「このSentinel-2シーンが適しています」と出力した後、**別のエージェントがNDVI（正規化植生指数）を計算**して画像をユーザに提示したり、**雲マスクを適用**して雲を除去した画像を返す、といったフローです。これには画像データの取得と計算が必要ですが、幸いGoogle Earth EngineやSentinel Hub、EO-LabといったプラットフォームをAPI経由で利用すれば実現可能です。例えばLLMエージェントにEarth Engine用のPythonコードを生成させNDVI画像を書き出すとか、Sentinel Hub APIの処理リクエストを組み立てさせる方法が考えられます。これが実現できれば、ユーザは「去年の夏のNDVIが高い場所を教えて」のような高度な質問を投げ、エージェントが適切な衛星データを検索・取得・処理し、**解析画像や数値指標まで提供**するという流れも可能になります。まずは検索部分のMVP完成が先ですが、将来的なモジュール連携として「**検索エージェント + 処理エージェント**」のマルチエージェント化は大きな発展軸です。
- ・**時系列モニタリングエージェントへの発展:** 一度きりの検索だけでなく、**定期的なデータ収集・監視**に役立てる方向性もあります。例えば農業分野では「毎週この圃場の衛星画像をチェックしてNDVIの推移を追跡したい」というニーズがあります。エージェントがユーザから定期ジョブを登録され、バックグラウンドで**継続的にSTAC検索を実行**し、新しい画像が出たら自動で解析・通知するといった機能拡張です。同様に災害監視では、洪水シーズン中毎日特定流域の画像を集めて変化を検出するなど、**サブスクリプション型**の時系列エージェントが考えられます。これをLLMとどう結びつけるかは課題ですが、例えばChatGPTプラグインのような形で「この条件で今後3ヶ月監視して」と命令すると裏で定期実行に登録される、といったUXも面白いでしょう。技術的にはスケジューラとストレージが必要になりますが、**検索部分は既存**なので再利用できます。さらにLLMに時系列データの簡易な解釈（増減傾向など）をさせ、レポートを生成することも可能です。こうした発展は、**単なる単発検索からソリューション型への昇華**であり、ユーザ価値を高める次のステップといえます。

- ・**特化型検索エージェントへの拡張（災害・農業・都市など）**：衛星データの利用用途別に、エージェントをドメイン特化させる展開も考えられます。例えば**災害対応向け**には、洪水・火災・地震後の衛星画像を迅速に検索する機能を持たせ、最新時刻のSAR画像（悪天候でも可視可能なSentinel-1など）を優先するといったカスタマイズが考えられます。**農業管理向け**なら、作物生育を見るため過去数ヶ月のNDVI合成を探すとか、クラウドフリー期間の長い衛星を選ぶ（Landsatは16日間隔、Sentinel-2は5日間隔なので頻度優先なら後者を選択など）知識を埋め込むことができます。**都市解析向け**には高解像度衛星（もし利用可能なオープンデータがあれば）を検索対象に加え、都市名からその市境界ポリゴンを取得して精密に検索する、といった工夫もあるでしょう。それぞれの分野で**ユーザが求める基準**（例えば災害では「被害がはっきり写っている画像」、農業では「曇っていない生育期の画像」など）が異なるため、評価エージェントのロジックを分野毎に最適化するイメージです。LLMにドメイン知識をプロンプト注入しておけば、ユーザの曖昧な依頼でも推測して補完する度合いを調整できます。最終的には**複数の特化エージェント**（災害くん、農業ちゃん、のような）が内部で同時稼働し、ユーザの質問内容から適切なエージェントが応答する、といった**マルチエージェントシステム**への発展も可能でしょう。
- ・**ChatGPTプラグインやAPIサービスとしての公開**：エージェントが軌道に乗れば、その機能を**外部提供**することも視野に入れます。具体的には、OpenAIのChatGPTプラグインとしてエンドユーザに使ってもらう構成です。プラグイン化する場合、本システムの検索API（独自サーバでLLM制御下にSTAC検索するエンドポイント）を用意し、そのOpenAPI仕様をChatGPTに認識させることで、ユーザはChatGPTに対し「～な衛星データを探して」と尋ねると裏で本システムAPIが呼ばれて回答する、という流れが実現します。これはまさに**自然言語→ツール実行**という本システムの思想と合致します。プラグインにすればUIやユーザ管理はOpenAI側に任せられる利点があります。また自前でサービス提供するなら、**汎用API**として（例えばRESTで `/search?query=...` に対し結果JSONやシーン一覧を返す）公開し、他のアプリや研究者が利用できるようにする展開もあります。この場合ドキュメント整備や利用制限の設計も必要ですが、**地理・衛星データ版の検索GPTサービス**としてポテンシャルがあります。もっとも、LLMのAPI利用にはコストがかかるため、ビジネスモデルや運用費用も考慮しなければなりません。個人開発レベルでは難しいかもしれません、オープンソース公開してコミュニティ貢献・フィードバックを募る道もあります。
- ・**マルチエージェント化（役割分担制御）**：システム内部をさらに高度化するアイデアとして、**複数のエージェントが連携して答えを導く構成**も考えられます。例えば、**検索担当エージェント**はユーザ入力を受け取りSTAC APIから生データ一覧を取得する。その後、**評価担当エージェント**が一覧を検証し、雲量や画質、重複などをチェックして候補を絞り込む。最後に**回答生成エージェント**がユーザ向けに説明文や推奨理由を含めた出力を行う、という三段構えです。LLMのチェーンオブソート的に、一度に全部やらせるよりステップ毎に専門家エージェントが処理するイメージです。このメリットは、各エージェントに専用知識やプロンプトを与え最適化できる点です。評価担当には画像品質に関する知識（太陽高度が高い方が良い、オフナadirの角度が小さい方が幾何歪みが少ない等）を組み込み、検索担当にはSTAC仕様の知識を埋め込む、といった役割特化ができます。また結果の検証段階を設けることで、**誤った検索結果をユーザに見せない品質管理**の効果もあります。もっと発展すれば、異なるモデル（例えば規模の小さい軽量LLMで検索し、大型モデルで最終生成するなどコスト管理）を使い分けることも可能です。マルチエージェントは複雑度が増しますが、LangChainや独自フレームワークで段階実行すること自体は比較的容易です。将来的にシステムが大規模化し要求も多様になった場合、**内部エージェントの協調**によって拡張性・保守性を高めることができます。

## システム構成図（テキストによる簡易図）

以下に本エージェントの基本構成フローを示します。ユーザからの質問入力から、衛星データ検索・結果推薦までの流れをテキストで表現します。



※上記④と⑤は実質同じLLMでも段階を分けています。また①②③④⑤をすべて一つのLLMがReAct的に行う実装も可能です。

## 実現ステップの詳細

1. **要件定義と技術選定:** まず扱う衛星データの範囲や利用APIを決定します（例: **Planetary Computer STAC**を使用しSentinel-2/Landsatを対象とする）。次に使用するLLMを選定（精度重視ならGPT-4、コスト重視ならGPT-3.5やLlama2など）。地名解釈にはNominatimなどジオコーダを選びます。日本語対応要件もここで明確化します（最初は英語のみなど）。
2. **プロンプト設計とテスト（NLP部分）:** LLMに対し「入力文から検索条件を抽出する」プロンプトを開発します。例えばシステムメッセージに「あなたは地理データ検索エージェントです。ユーザの質問から場所・日付・衛星種別・雲量をJSONで返してください...」と指示し、いくつかの例示を与えます。OpenAI

のGPT系ならワンショット/少ショットで実験し、所望のJSON（または内部思考用テキスト）を出力できるか確認します。Temporal/Spatial/Filterなどサブ要素毎に抽出させる方や、一括でSTACクエリ構造を作らせる方を比較します。テストとして「東京の画像」「2020年夏の北海道東部」「雲ゼロのLandsat画像」など多様な入力で想定どおりパースできるかを検証します。

3. **ジオコーディング実装:** 抽出された場所情報をジオコーダAPIに投げ、座標を取得するコードを実装します。Nominatimの場合は地名文字列をURLクエリで投げJSONを取得します。複数候補が返る場合は先頭を採用、ヒットしない場合はLLMに再確認させるかユーザに訊ねるフローを作ります。Python等で関数化し、後でLLMエージェントから呼び出せるよう準備します。
4. **STAC検索モジュール実装:** STAC APIエンドポイントに対しHTTPリクエストを送り、検索結果（Itemのリスト）を取得するコードを用意します。検索パラメータ（bboxやdatetime、query(JSON））を組み立てる部分は、LLMの出力を使うか、LLMには高レベル条件を出させて**サーバ側で組み立てる方法**もあります。MVPではLLMから具体的なコレクションIDや日時文字列を受け取り、バックエンドでHTTPリクエストを構築→JSONをパースして主要項目を抜き出す処理を実装します。PythonのSTAC clientライブラリ（例えばpystac）を利用して良いでしょう。
5. **LLMエージェント統合（ツール使用）:** LangChain等を使い、LLMが手順②③を自発的に実行できるよう統合します。LangChainのTool機能でgeocode\_locationとsearch\_stac関数を登録し、エージェントにそれらを呼ばせるポリシーを決めます。システムプロンプトに具体例を含め、「地名があればまずgeocodeせよ、その結果のbboxを使ってsearchせよ」という手順を教示します。仮にLangChainを使わない場合は、自分でLLMの思考ステップをパースしツール呼び出し→結果をLLMに与えるループを組みます。まずはシンプルに、一回の質問につき一回検索を行う流れを完成させます。
6. **結果要約・推薦のロジック:** 得られた検索結果（例えば最大10件程度のメタデータ）をLLMに解析させ、ユーザへの回答文を生成する段階です。LangChainならOutput Parser等でやらせるか、単純にsearch\_stacの結果をLLMにプロンプトとして与え「この中から最適なものを選び説明してください」と指示します。ここでは**選定基準**をプロンプトに記述し、例えば「雲量が少なく、日時が新しいものを優先して選んで下さい」のようにルールを与えます。LLMが安定して最良の一件を選ぶか検証し、時に複数件提案もできるように調整します。
7. **エンドツーエンドテスト（MVP完成）:** 以上を繋げ、ユーザから自然文を受け取り→エージェントがツールを駆使→回答を返す一連の流れをテストします。想定質問に対し適切な衛星シーンIDや日時が出ているか、人間が検証します。不具合（例えば地名解釈ミス、該当なし時の応答など）を洗い出し修正します。この段階で基本機能MVPは完成です。
8. **高度化機能の追加:** ここからフル実装に向けて、課題対策や機能拡張を行います。（順不同）
9. **対話継続対応:** ユーザが追加条件を後から言った場合に、履歴のクエリと統合して再検索するロジックを追加します。LLMに直近の検索条件を要約させ、それに新条件を加味してクエリ更新するフローなどを実装します。
10. **結果プレビュー:** メタデータだけでなくサムネイル画像URLが取得できる場合、回答にそれを埋め込むようにします（UIが画像表示対応なら）。STACのassets.thumbnailリンクを取得し、回答に含めます。

11. **エラー処理とガード:** ジオコーダやSTAC APIがエラーになった場合のfallbackを実装します。リトライや、「該当するデータが見つかりませんでした。条件を変えてください。」といったユーザへの案内をLLMに言わせます。LLMが想定外の出力（ツール実行指示以外の余計な発話など）をした場合に備え、正規表現や出力検証でガードし、再試行させる仕組みも入れます。
12. **性能改善:** キャッシュの導入（同じ地名→bbox結果をメモリ保持、同じクエリなら前回結果を再利用など）や、検索結果が多い場合に上位n件だけ取得する調整をします。LLMへのプロンプト長も制御し、必要最低限の情報のみ渡すよう最適化します。
13. **分野特化調整:** もし特定用途にフォーカスするなら、この段階でLLMへの指示や評価基準を変更します（例えば農業用プリセットでは雲量を特に厳しくする等）。また衛星コレクションの追加（Sentinel-1や他のSTACコレクションを検索対象に含める）も行います。
14. **ユーザテストとフィードバック反映:** 試作システムを想定ユーザに触ってもらい、インターフェースや回答内容のフィードバックをもらいます。特に**自然言語インターフェースの使い勝手**（どのように尋ねれば良いか分かるか、応答の説明は十分か）を確認します。必要に応じプロンプトを改善し、応答に信頼性指標（例えば「データ提供: Planetary Computer」等付記）を加えるなど調整します。また、多言語対応予定であれば日本語で試してもらい問題点を洗います。
15. **展開準備:** システムを継続利用できる形にデプロイします。バックエンドでLLM APIキーとSTACエンドポイントURL、レートリミット設定などを環境変数化し、サーバにデプロイ。ChatGPTプラグイン化するならOpenAPI仕様を整備し審査に出します。オープンソース公開ならREADMEや使用例を整理します。以上で一通り完成となります。

## 実現可能性ランク（A～D評価）

- ・**全体の実現可能性: A（高い）** - 本コンセプトは既に類似のプロトタイプが存在することからも分かるように、現在の技術で**十分実現可能**です。特にMVP範囲（自然言語→STAC検索→メタデータ推薦）は、LLM APIと公開データカタログを組み合わせれば個人でも構築できます。開発者コミュニティからもいくつか事例報告があり、技術的な障壁は低いと判断されます。
- ・**NLP解析精度: A** - 地名・日時の抽出などはLLMの得意分野であり、高精度に期待できます。多少の調整は必要ですが、GPT-4クラスなら日本語含めかなり柔軟に意図を読み取れるため、大きなリスクはありません。
- ・**ジオコーディング部分: A** - オープンなNominatim等を使えばコストなく実装可能であり、技術的困難も低いです。極度の曖昧地名以外は概ねヒットするでしょう。
- ・**データ検索部分: A** - STAC APIは成熟しており、Planetary ComputerやEarth Searchなど無料で使える信頼性高いサービスがあります。API仕様もRESTで簡明、認証も不要or簡易なので、接続性も問題ありません。したがってこの部分も実現上のリスクはほぼありません。
- ・**LLMエージェントの統合: B** - 単純なLLM応答ではなくツール実行を絡める部分は若干複雑です。とはいってLangChain等の既成ライブラリが充実しているため、多少学習コストがありますが十分対応可能です。LLMへの細かな指示調整やエラー時のハンドリング実装に工数がかかるため、評価はAより一段下げる程度です。

- ・日本語対応: B** – 日本語だけで完結させる場合、対応可能なLLM選定（GPT-4など有料モデルの利用）が前提となり、コスト面・依存面でややハードルがあります。ただし技術的には可能であり、翻訳を挟むなど代替策もあります。多少の精度劣化リスクはあるためBとします。
- ・リアルタイム性・効率: B** – 応答速度や大量データ時の処理など、スケーラビリティ面では最適化が必要です。初期版では許容範囲でも、ユーザ数やデータ量増大時に課題が出る可能性があります。この点で将来的リスクがあるためB評価です。
- ・画像解析など高度機能: C** – メタデータ検索を超えて、画像内容理解や自動解析まで踏み込むと一気に難度が上がります。外部のAIモデルとの連携や計算資源確保が必要で、個人開発レベルでは難しい段階です。ただ必須ではない拡張要素のため、コア部分には影響しません。
- ・総合評価: A寄りのB（かなり実現しやすいが一部要調整）** – 基本コンセプトは十分手の届く範囲で、早期にプロトタイプ構築→実用化できる見込みです。細部の調整や非機能要件（速度、多言語）での課題は残るもの、研究段階を脱して実務適用できるレベルに達しています。

## 新規性が高い部分・低い部分

| 機能/要素          | 新規性の評価 | 解説   |
|----------------|--------|--|
| 自然言語インタフェースで検索 | 高い     | 衛星データ検索にチャット形式の自然言語UIを持ち込む試みは新しく、従来ツールがないユーザ体験です。専門用語不要で問い合わせできる点に独自価値があります。               |
| LLMによるクエリ自動生成  | 高い     | ユーザ意図から機械用クエリへの変換をAIが行う点は、静的フォームとは異なるダイナミックなアプローチであり、特にLLMの推論能力を活かした高度なマッピングは革新的です。        |
| 複数データセット横断検索   | 中程度    | 複数の衛星カタログを統合して検索すること自体は、STACの思想に沿えば可能であり、新規というよりは技術統合です。ただしLLMが自動でデータセット選択まで行う部分は新規性があります。 |
| ツール指示付きエージェント  | 中程度    | LLMにAPI操作をさせる設計は最近普及し始めた手法で、先行事例はいくつか存在します。本システム固有というよりAIエージェント全般の新潮流ですが、衛星分野では目新しいです。     |
| メタデータのみでのシーン評価 | 低い     | メタデータ（雲量や日時）を基に最適シーンを選ぶこと自体は、従来手動で分析者が行っていた作業です。これを自動化することに新規性というよりは利便性向上の意味合いがあります。       |
| STAC/API利用そのもの | 低い     | STAC APIの使用や、既存ジオコーダサービスの利用自体は既知技術の組み合わせです。システム全体としての新しさは、それらを統合しNLで操作する部分にあります。           |

| 機能/要素            | 新規性の評価 | 解説  |
|------------------|--------|---|
| (将来案) 画像解析エージェント | 高い     | 画像内容までLLMや他AIで理解・検索する段階になれば、現状の技術フロンティアに突入するため非常に新規性が高いです（例: Queryable Earth のような取り組み）。 |

## 潜在的な重要課題の優先度

- ・曖昧なクエリの解釈問題 – **Critical (最重要)** : ユーザ入力の曖昧さによる誤検索は本質的問題です。特に地理・期間表現の不確実さは検索精度に直結するため、まず対処すべき課題です。場合によってはユーザ確認を促すなどUX面も含めた解決が必要でしょう。
- ・LLMの信頼性（誤った検索・誤回答） – **Critical**: LLMエージェントが暴走したり間違った結果を返すと、システム全体の信用失墜に繋がります。誤フィルタや幻データの提示は避けねばならず、プロンプト整備や出力検証などあらゆる手段で信頼性を担保する必要があります。
- ・大量結果・性能スケーラビリティ – **High (高い)** : ユーザの指定範囲が広大だった場合の結果爆発や、複数ユーザ同時利用での応答遅延は現実的に起こり得ます。UXに大きく影響するため高優先度での対策（結果制限や並列処理・キャッシュ導入）が望まれます。
- ・クラウドカバーなどメタデータ精度 – **High**: メタデータ依存による見落とし（AOI局所の雲など）は、ユーザが「おすすめを信じたら雲だらけだった」等の不満に直結します。完全解決は難しくとも、例えばプレビュー画像表示でユーザ自身が確認できるようにする等、優先度高めのケアが必要です。
- ・多言語サポート – **Medium (中程度)** : 対応言語を広げることは利用者層拡大に重要ですが、技術的には後から追加可能なため優先度は中程度です。まず英語で堅牢に動かし、その後日本語対応を検討する段取りで支障ないでしょう。
- ・UI/UX上の期待値コントロール – **Medium**: ユーザが本エージェントの出来ること・出来ないことを誤解しないようにすることも課題です。検索結果にどこまで責任を持つか、注意書きやヘルプの充実などは中程度の優先度で対処すべきです。
- ・API利用制限・費用 – **Medium**: 外部API（LLMや地図）の使用量によっては料金や制限に抵触します。個人開発では特にLLM利用料がボトルネックになります。これは技術課題ではありませんが、継続運用には無視できない点であり、中程度の優先度でモニタリング・対策（必要に応じモデル変更等）します。
- ・画像そのものを扱わないことの限界 – **Low (低い)** : これは根本仕様なので優先度は低いですが、将来的発展の機会損失にはなります。重要課題というより認識すべき限界事項としてユーザと開発者が共有すべき点です。

## 競合・類似技術一覧

| ツール/サービス名   | 自然言語検索対応          | 概要・特徴  |
|---|-------------------|--|
| <b>USGS EarthExplorer</b> (米国地質調査所)                 | 非対応               | 衛星・航空写真の検索ポータル。住所・座標でエリア指定し日付やセンサをフィルタ可能。自然文クエリは不可で、操作に専門知識を要する。                           |
| <b>Copernicus Open Access Hub</b> (現Data Spaceプラウザ) | 非対応               | 欧州のSentinelデータ検索サイト。マップでエリア指定し期間やプロダクトで絞り込み。キーワード検索はあるが基本はメタデータ項目指定。NLインターフェースなし。          |
| <b>NASA Earthdata Search</b>                        | 部分的               | NASAの地球観測データ検索。キーワードと日時・範囲フィルタで検索。データセット名や場所名は検索できるが、自由文章クエリには対応しない。                       |
| <b>Google Earth Engine Data Catalog</b>             | 非対応               | 衛星データを含むカタログのブラウズサイト。データセット毎のメタデータ閲覧や位置検索は可能だが、自然文での検索機能は提供されていない。                         |
| <b>Sentinel Hub EO Browser</b> (Sinergise)          | 非対応               | ウェブ上でSentinelやLandsatを閲覧・検索できるGUI。地図上でエリア指定し日時や雲量をスライダーで調整して検索。直感的だがテキストによる条件入力はできない。      |
| <b>Microsoft Planetary Computer STAC API</b>        | 非対応 (APIのみ)       | 無料のSTAC APIサービス。ブラウザ上の簡易検索UIもあるが、基本はプログラムからJSONクエリ投げて使う。自然言語インターフェースは無し。                   |
| <b>Element 84 Earth Search</b> STAC API             | 非対応 (API+Console) | AWS上のオープンデータを統合したSTAC API。地図ベースのConsoleも提供。複数データを横断検索可能だが、入力はフォーム形式。                       |
| <b>Development Seed STAC-Semantic-Search</b>        | 対応 (英語)           | 開発中のオープンソースNL検索システム。ChatGPTなどを使い「～を探して」と聞くとSTAC検索し地図表示まで行うプロトタイプ。セマンティック検索やLLM再ランクイング機能あり。 |
| <b>Sparkgeo Natural Language Search</b> (試作)        | 対応 (英語)           | Sparkgeo社ブログで紹介。自然言語クエリで衛星画像を検索し、UKのEO DataHub STACから該当画像をマップ表示するデモ。実装詳細は非公開だがLLM等を用いた可能性。 |
| <b>Element 84 Queryable Earth</b>                   | 対応 (英語)           | 自然言語で地理特徴を検索する実験。衛星画像そのものをVLMでエンコードし、「テキスト→画像内容」の検索を可能にする先端例。特定地域（マサチューセッツ州）でのデモ。          |

| ツール/サービス名                                      | 自然言語検索対応 | 概要・特徴   |
|--|----------|---|
| <b>Google Earth AI (試験段階)</b>                  | 対応 (英語)  | Googleの研究プロジェクト。チャット形式で「画像中の対象物検出」「多ステップ地理分析」が可能。基盤モデルとLLMエージェントを組み合わせた包括的システム。一般公開はされていない。 |
| <b>ESA Φ-lab Earth Virtual Assistant (EVE)</b> | 対応予定     | ESAが開発中の地球観測チャットボット。自然言語で衛星データ分析まで問合せできる将来像を描く。2024年に実験開始と報じられた。                            |

※上記のうち、本提案システムが直接競合しうるのは、Development SeedやSparkgeoの類似実装です。ただしまだいすれもプロトタイプ段階であり、完成度次第では本提案がリードできる可能性があります。

## 参考文献・リンク

- Development Seed, “**STAC Semantic Search**” (GitHub README) – 「自然言語で衛星画像を検索するAIエージェント」のOSS実装概要
- OpenGeoAI, **Geo Agents module** (source code) – LLMにSTAC検索を行わせる手順を定義したコード抜粋
- Tatsuyuki Sekine, “**Web Visualization of Remote Sensing Data via GEE with LLM**” – LLMで地名・期間を抽出しEarth Engineから衛星画像を取得・表示する試作解説 <sup>2</sup>
- Element 84, “**Earth Search**” – AWS上のSTAC APIサービス（対応データセットや検索機能の紹介）
- Stephen Chege, “**Unveiling ESA's Earth Observation AI Assistant**” – ESAのChatGPT型衛星データアシスタント開発に関する記事 (*Medium*, 2024)
- Google Research, “**Google Earth AI: geospatial insights with foundation models**” – Googleにおける地理AIエージェントの最新動向 (2025)

(その他、USGSやNASAの既存ツール紹介ページ、Sparkgeoブログ記事などを内容把握に参考にしました)

## 総合評価（現時点での価値と将来性）

本提案の「自然言語で衛星データを検索し推薦するエージェント」は、現時点で十分に構築する価値があるアイデアと評価できます。技術的な実現性は高く、プロトタイプから有用なツールへ発展させるハードルも低めです。特に衛星データ初心者や非専門家にとって、複雑な検索条件を意識せずデータにアクセスできる意義は大きく、オープンサイエンスの推進や教育用途にも貢献し得ます。新規性の面では、既存ツールにない自然言語UIという強みがあり、うまく成熟させればユーザ体験の質を大幅に向上できるでしょう。また将来性についても、画像解析エージェントや分野特化型AIへの拡張など発展の余地が広く、単なるデータ検索に留まらないプラットフォームへ成長する可能性があります。GoogleやESAといった大手も類似コンセプトに注目しており、市場やコミュニティの関心も高まっています。

もっとも、LLMの誤動作リスクや曖昧入力の問題など、実用化に際して解決すべき課題はいくつか残ります。特に結果の信頼性確保と応答の安定性は慎重なチューニングが必要です。しかしこれらは解決不能なわけではなく、段階的改善が可能な領域です。総合的に見て、本エージェントは「衛星データ利用の民主化」を推し進める有望なアプローチであり、早期に試作しユーザフィードバックを得ながら改良を重ねる価値があります。現段階ではまず限定的な範囲でMVPを構築し、有用性を実証した上で機能拡張・スケール拡大を図るのが賢明でしょう。

う。将来的には、専門家だけでなく一般市民が衛星データを会話しながら取得・活用できる時代の一助となることが期待されます。

---

- ① Google Earth AI: Unlocking geospatial insights with foundation models and cross-modal reasoning  
<https://research.google/blog/google-earth-ai-unlocking-geospatial-insights-with-foundation-models-and-cross-modal-reasoning/>
- ② Web Visualization of Remote Sensing Data via Google Earth Engine with the LLM | by Tatsuyuki Sekine | Remote Sensing Tech by ELSPINA VEINZ Inc.  
<https://space.elspina.tech/web-visualization-of-remote-sensing-data-via-google-earth-engine-with-the-lm-63521a437f11?gi=858685467262>
- ③ Sentinel-2 Data Dictionary | U.S. Geological Survey  
<https://www.usgs.gov/centers/eros/science/sentinel-2-data-dictionary>