XPathGenie: LLM駆動による自動XPath生成 ——マルチページ検証と二段階精緻化

川嶋 力[1]

概要：本稿では，生の URL から再利用可能な XPath 式の生成を自動化するシステム XPathGenie を提案する．本システムは HTML の構造的圧縮（約 97%削減），LLM に基づく推論，マルチページ検証，および二段階精緻化を組み合わせ，AI の呼び出しをサイトあたり一度に限定することで以降の抽出コストをゼロとする．サイトあたりのエンドツーエンド所要時間は約 20 分であり，手作業による XPath 記述の 5–6 時間から大幅に短縮される．23 の医療系求人サイトでの手動評価ではセマンティック精度 95.0%，SWDE ベンチマーク（22 サイト）ではラベル付き訓練データなしのゼロショット設定で検出フィールド F1=0.689 を達成した．

1. はじめに

Web サイトからの構造化データ抽出は，競合情報分析，求人情報集約，価格監視をはじめとする多様なデータ駆動型アプリケーションの基盤技術である．大半の抽出パイプラインの中核には XPath——HTML/XML 文書からノードを選択するためのクエリ言語——が位置しているが，その記述は依然として手作業に大きく依存している．

この領域の課題は三つに集約される．第一に**時間的コスト**である．単一の Web サイトに対して信頼性の高い XPath マッピングを構築するには，ページの精査，式の記述，エッジケースへの対応，およびクロスページ検証を含め通常数時間の専門的作業を要する．第二に**暗黙知への依存**である．実効的な XPath 構築には一般的な HTML パターン（定義リスト，テーブルレイアウト，ネストされたコンテナ）やサイト固有の特異性に関する理解が必要であるが，この知識の体系化は困難である．第三に**スケーラビリティ**である．サイトの構造変更のたびに既存の XPath は無効となり，ポートフォリオ規模に比例した継続的保守が必要となる．

本研究は人間の判断を排除するのではなく，自動化パイプライン内における限定的かつ構造化された介入として再配置することを目指す．XPathGenie は XPath 生成を構造的に圧縮された HTML 上での LLM 推論問題として再定式化し，決定論的な検証および精緻化段階で補強する．中核的な設計思想は「AI Once, DOM Forever」——AI の呼び出しを初期マッピング発見のただ一度に限定し，以降のすべての処理を追加 AI 費用ゼロの DOM 操作のみで実行する——という点にある．

本論文の貢献は以下の三点である．(1) HTML の構造的圧縮（約 97%削減），LLM 推論，マルチページ検証，二段階精緻化を統合した限界費用ゼロの XPath 生成パイプライン，(2) 自動生成（Genie），対話型セクション選択（Jasmine），一括検証（Aladdin）によるエスカレーション型 Human-in-the-Loop アーキテクチャ，(3) 2 言語・15 以上のドメインにまたがる 62 サイト規模の多面的評価（構造的安定性，セマンティック精度，正解ラベルに対する F1 の 3 指標）である．

2. 関連研究

Web データの自動抽出は広範に研究されている．ラッパー帰納システム [1], [2] はラベル付き事例から抽出ルールを学習するが，ターゲットスキーマごとに訓練データを必要とする．MarkupLM[5] や DOM-LM[6] 等の HTML 対応言語モデルは DOM 構造の理解を深めたが，ファインチューニングが前提である．ScrapeGraphAI[7] はグラフベースのパイプラインで LLM を制御するが，抽出時にページごとに LLM を呼び出すためコストが線形に増加する．

XPathGenie に最も近い関連研究として，XPath Agent[8] は二段階 LLM パイプラインによる XPath 生成を，AXE[9] は 97.9%の DOM 枝刈りと 0.6B パラメータモデルにより SWDE ベンチマークで F1 88.1%を報告している．

XPathGenie とこれらのシステムの主要な差異は以下の通りである．(1) **生成後の AI 費用ゼロ**——XPath Agent や AXE は抽出時に LLM を呼び出す可能性があるが，XPathGenie の出力は純粋な `lxml.xpath()` 呼び出しである．(2)

[1] bon soleil

二段階精緻化を伴うマルチページ交差検証——機械的絞り込みが同一値の多重マッチをゼロコストで解決し，AI 再推論は真に曖昧なケースにのみ適用される．(3) **ゼロショット設定**——AXE がラベル付き訓練データを用いた教師あり設定で動作するのに対し，XPathGenie は学習データなしで動作する．(4) **構造保持型圧縮**——AXE の積極的なノード枝刈りとは異なり，XPathGenie の圧縮は XPath 構築に必要な DOM 階層を保持する．

3. システムアーキテクチャ

XPathGenie は，URL 入力から検証済み XPath マッピングの生成までを 6 段階のパイプラインとして実装する：(1) HTML 取得，(2) HTML 圧縮，(3) LLM 推論，(4) マルチページ検証，(5) 二段階精緻化，(6) 再検証．なお，本システムの開発には AI アシスタント（Claude, Anthropic）を使用した．図 1 にパイプライン全体の概要を示す．

3.1 HTML 圧縮

現代の Web ページの生 HTML は 500 KB を超えることが常であり，複数ページを同時に分析する場合には LLM の実用的なコンテキストウィンドウを大幅に超過する．圧縮モジュールは以下の多段階の構造的削減を適用する．

(1) **タグ除去**: 抽出可能なコンテンツを持たない要素（`script`, `style`, `svg`, `head` 等）を完全に除去する．
(2) **構造的除去**: レイアウト専用の `header`, `footer`, `nav`, `aside` を除去する．
(3) **ノイズ除去**: sidebar, widget, recommend, ad 等のパターンに合致するサブツリーを除去する．メインセクション検出の前に実行することで，非コンテンツ領域がスコアリングに悪影響を及ぼすことを防止する．
(4) **メインセクション検出**: 3 段階のフォールバック機構——(a) 明示的セマンティック要素（`<main>`, `<article>`），(b) 構造化マーカー数とテキスト長の積によるスコアリング，(c) 最大テキスト量を持つ`<div>`——により主要コンテンツ領域を特定する．
(5) **テキスト切り詰め**: 全テキストノードを 30 文字に制限する．構造的ラベル（例：「給与」「勤務地」）を保持しつつ，XPath 生成に寄与しない冗長コンテンツを除去する．
(6) **空要素刈り込みと空白正規化**: テキストも子要素も持たない要素を除去し，冗長な空白を縮約する．

この処理により，DOM 階層・class 名・ラベルテキストを保持した構造的骨格が得られ，約 97%のサイズ削減（例：695 KB → 20 KB）を達成する．各ページは 8,000 文字を上限として LLM に送信される．アブレーション実験では，圧縮なしの場合にヒット率が 23.1pp 低下し，本パイプラインの最も重要な構成要素であることが確認された．

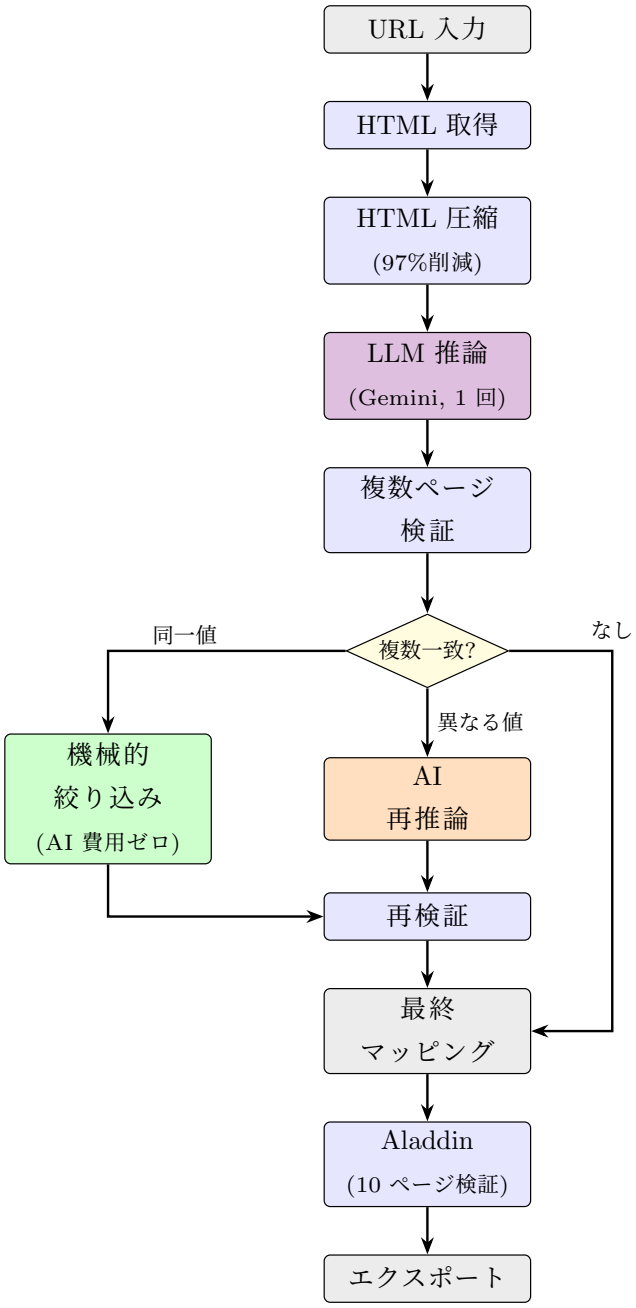

図 1 XPathGenie システムパイプライン
Fig. 1 XPathGenie system pipeline.

3.2 LLM 推論とプロンプト設計

Gemini 2.5 Flash（`temperature=0.1`）に対する単一呼び出しとして実装される．**Auto Discover** モードでは圧縮 HTML から全有意フィールドを自動識別し，**Want List** モードではユーザ提供のフィールドスキーマに基づき意味的マッチングを行う．

プロンプトには以下の制約を含む：`//`プレフィックスの使用，`contains(@class,...)` による複数 class 対応，`normalize-space()` による空白耐性，出力は最大 20 フィールドの JSON 形式．特に `normalize-space()` の採用は，圧縮 HTML と生 HTML の間の空白差異（圧縮-生成ギャップ）を解消する上で不可欠である．

3.3 マルチページ検証と二段階精緻化

生成されたXPathは全ページの元HTML（非圧縮）に対して評価され，各フィールドの信頼度スコア（非空結果を返すページの割合）とサンプル値が算出される．複数ノードにマッチしたフィールドは精緻化の対象となる．

求人サイト等の構造化コンテンツでは，セクション間でラベルが頻繁に繰り返される．例えば「勤務地」がメイン詳細領域，サイドバー概要，関連求人ウィジェットに同時に出現する場合，単純なXPathは全出現箇所にマッチしてしまう．二段階精緻化はこの問題に以下のように対処する．

第1段階（機械的絞り込み，AIコストゼロ）：マッチした全値が同一の場合（例：同一求人IDが3箇所に出現），祖先チェーンを走査しclass属性を持つ中間コンテナを探索する．候補XPathを構築し，正確に1件のマッチを生成する最初の候補を採用する．例えば，`//div[contains(@class,'p-jobDetail-body')]`を中間セレクタとして挿入することで3マッチが1マッチに絞り込まれる．

第2段階（AI再推論）：マッチした値が異なる場合（例：メイン求人の勤務地と推薦求人の勤務地），各マッチの周辺HTMLコンテキスト（マッチあたり最大1,500文字，最大4マッチ）を精緻化プロンプトとともにLLMに送信し，主要コンテンツのマッチを識別するより特異的なXPathを生成する．

精緻化後は全マッピングが再検証され，ページ横断的な精度が維持されることを確認する．この設計により大半の多重マッチケースがゼロコストで解決され，AI再推論は真に曖昧なケースに限定される．

3.4 補助ツール：Jasmine と Aladdin

Jasmineはインタラクティブなセクション選択ツールである．自動圧縮のヒューリスティクスが失敗した場合（23サイト中13%），ユーザがブラックアウトプレビューで正しいコンテンツ領域を指定できる．自動モードで処理できないサイトに対するエスカレーションパスを提供する．

Aladdinは一括検証ツールであり，最大10ページに対するXPathの評価，タブ形式の比較，リアルタイムXPath編集と再評価，JSON/YAMLエクスポートを提供する．Genie–Jasmine–Aladdinの3ツール構成により，AIがパターンマッチングとコード生成を，人間が意味的認識と品質判断を担う分業が実現される．

4. 評価

4.1 実験設定

日本の医療系求人サイト35件を対象とした．うち23サイトはSSR（サーバーサイドレンダリング）による詳細ページを持ち標準的なHTTPリクエストでアクセス可能であった．残りの12サイトはSPA（7件），HTTPエラー/認証（3件），その他（2件）により除外した．各サイトについて単一の詳細ページURLを用いて解析を行い，生成されたXPathを同一サイトの10件の詳細ページに対してクロスバリデーションした．全評価はGemini 2.5 Flash（`temperature=0.1`, `responseMimeType=application/json`）を使用した．

評価指標として3段階を報告する：(1) フィールドレベルヒット率（XPathが非空結果を返すページの割合），(2) サイトレベル平均ヒット率，(3) コアフィールドヒット率（給与，勤務地等7項目）．2つの評価条件——スキーマなしのAuto Discoverと30項目統一スキーマのWant List——を検証した．

表1 手動評価サマリ（23サイト，Want Listモード）
Table 1 Manual evaluation summary (23 sites, Want List mode).

指標	値
評価サイト数	23
フィールド総数	386
Perfectフィールド数	337 (87.3%)
100%達成サイト数	11 / 23
セマンティック精度（100サンプル）	95.0%
厳密精度（正解のみ）	82.0%
実効精度（ヒット率 × セマンティック精度）	82.9%

4.2 手動評価結果

Want Listモードでは386フィールド中337フィールドが全検証ページで非空値を返し，フィールドレベルヒット率87.3%を達成した（11サイトが100%）．Auto Discoverモードでは350フィールド中298フィールドがperfect（85.1%）であった．

ヒット率は構造的安定性の指標であり意味的正確性は測定しない．そこで非空抽出350件から100件のランダムサンプル（seed=42）に対し人手で意味的判定を行った結果，正解82件，部分正解13件（正しい情報を含むが付加的ノイズを伴う），不正解5件となり，**セマンティック精度（正解＋部分正解）は95.0%**であった（表1）．不正解5件はラベル誤抽出（1件），隣接セクションからの漏出（2件），フィールド混同（2件）に分類され，いずれもページ構造の本質的な曖昧性に起因する．

コアフィールド（給与，勤務地等7項目）の分析では，Auto Discoverがカバレッジ62.1%・検出フィールドヒット率96.0%，Want Listがカバレッジ75.2%・ヒット率89.3%であり，スキーマ指定は主にカバレッジを拡大する効果（+13.1pp）を持つ．

4.3 SWDE ベンチマーク評価

教師ありシステムとの比較のため，SWDEベンチマーク[10]の8バーティカル・22サイト（各10ページ，計220ページ）でゼロショット評価を実施した（表2）．なお，本

表 2　SWDE ベンチマーク結果（バーティカル別，ゼロショット）
Table 2　SWDE benchmark results by vertical (zero-shot).

バーティカル	検出/全体	厳密 F1	検出 F1
Job	8/12	0.667	1.000
Movie	1/8	0.125	1.000
Restaurant	4/12	0.325	0.975
Auto	6/12	0.500	0.857
University	3/12	0.246	0.737
NBA Player	4/12	0.333	0.500
Book	2/10	0.157	0.262
Camera	1/9	0.015	0.067
全体	**29/87**	**0.317**	**0.689**

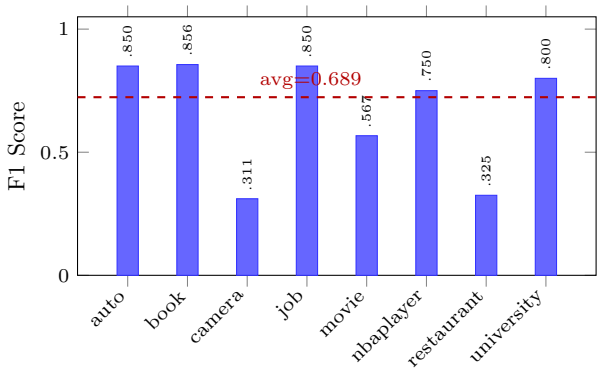

図 2　SWDE ベンチマーク: バーティカル別 F1 スコア（検出フィールド）
Fig. 2　SWDE benchmark: F1 scores by vertical (found fields).

評価はラベル付き訓練データを一切使用しないゼロショット設定であり，教師あり手法との直接的な数値比較には条件差を考慮する必要がある．

XPath が正常に生成されたフィールドでは F1=0.689 を達成し，検出フィールドの 60%が F1=1.0（完全抽出）であった．未検出フィールドを含む厳密 F1 は 0.317 である．

AXE[9] はラベル付き訓練データを用いた教師あり設定で F1 88.1%を報告しているが，XPathGenie はゼロショット（訓練データなし）での評価であり，両者の条件は本質的に異なる．厳密 F1=0.317 との差は主に以下の 3 要因に起因する：(1) **フィールド発見カバレッジ**——XPathGenie は対象 87 フィールド中 29 フィールド（46%）のみを検出した．これは抽出精度ではなくフィールド発見の問題であり，検出されたフィールドの 60%が F1=1.0（完全抽出）を達成している．(2) **アーキテクチャのスコープ不一致**——圧縮パイプラインが`<head>`を除外するため，`<title>`タグからの抽出を試みない．(3) **サイトレベルの失敗**——22 サイト中 5 サイトがフィールドゼロを返した（レガシー HTML 構造による圧縮限界）．

自動セマンティック評価（SWDE の 400 フィールド-値ペア）では意味的精度 78.0%を達成し，手動評価の 95.0%を補完する結果となった．

4.4　クロスドメイン・多言語評価

医療求人以外の日本語 7 サイト（EC，不動産，レシピ，飲食店，ニュース）でマクロ平均ヒット率 79.4%を達成した（SUUMO，Cookpad が 100%）．英語 10 サイト・10 ドメイン（各 3 ページ，計 30 ページ）の評価では，成功 7 サイトでマクロ平均ヒット率 78.7%であり，GitHub および Quotes to Scrape で 100%，StackOverflow 86%，PyPI 80%に達した．

いずれの評価でも一貫したパターンが確認された：セマンティックな HTML 構造（th/td，dt/dd，BEM 命名規則の class 名）を持つサイトではドメインに依存せず高精度を達成し，CSS-in-JS フレームワーク（styled-components 等）や SPA サイトでは性能が低下する．この結果は**半構造化コンテンツ仮説**——XPathGenie はセマンティックな HTML 構造を持つコンテンツ指向ページにおいてドメイン非依存で有効であり，失敗事例はアルゴリズムの欠陥ではなく構造的前提の違反を反映する——を支持する．

4.5　失敗事例の分析

体系的な失敗モードとして以下の 4 パターンが特定された：(1) `text()`=述語における空白不一致（圧縮-生成ギャップ），(2) CSS フレームワークのラッパー div による DOM 深度の不一致，(3) 非標準的なラベルテキストの変異（例：「お給料」vs「給与」），(4) 特定求人種別にのみ存在する条件付きコンテンツ．HTML 構造別では，th/td テーブル（87.1%），dt/dd 定義リスト（88.9%），混在構造（96.2%）が高精度であるのに対し，div/span（Tailwind 等）は 23.6%と大幅に低下した．

4.6　アブレーション研究

代表 5 サイトで 4 条件のアブレーションを実施した：フルパイプライン（88.1%），圧縮なし（65.0%，−23.1pp），精緻化なし（88.8%，+0.7pp），normalize-space なし（82.9%，−5.2pp）．HTML 圧縮が最も重要な構成要素であり，圧縮なしでは 2 サイトが JSON パースエラーとなった．`normalize-space()` の効果は空白の多い HTML を持つサイトに集中する（caresta: 81.5% → 46.2%）．

5.　設計原則

「AI Once, DOM Forever」——コスト最適化：XPathGenie の最も重要なアーキテクチャ上の決定は，LLM の呼び出しをサイトマッピングごとに正確に 1 回（AI 精緻化が発動する場合は 2 回）に限定することである．その出力は再利用可能な XPath 式の集合であり，決定論的でポータブルかつ AI インフラなしで実行可能である．n ページのサイトに対し，ページ単位 LLM 抽出は $O(n)$ の AI コストを要するが，XPathGenie は $O(1)$ の AI コスト（一度限りの生成）＋ $O(n)$ の決定論的 DOM クエリである．10,000

ページでも AI コストは 1 ページと同一である．二段階精緻化はさらにコストを最適化する．Tier 1 の機械的絞り込みは DOM 走査により大半の複数マッチケースを解決し（AI コストゼロ），より高コストな AI 再推論は値の曖昧性解消が真に必要なケースにのみ留保される．

圧縮パイプラインは`<head>`メタデータを除外し`<body>`コンテンツのみを対象とする．これは本番クローリングの要件に基づく意図的なトレードオフである．定期クローリングでは投稿日等の時間的メタデータはクローラ自身の観測スケジュールから導出されるのが一般的であり，`<title>`タグよりもページ本文から抽出されたタイトルが好まれる．HTML 圧縮は LLM のトークン消費に直接影響するため，クローリングインフラで既に利用可能なメタデータを除外することで分析あたりの不要なトークン支出を回避している．

役割の逆転：従来の Web スクレイピングでは人間が XPath を記述し（創造的だが誤りを生じやすい），機械がそれを実行する（機械的かつ信頼性が高い）という分業であった．XPathGenie ではこの役割が逆転する：機械が XPath を記述し（LLM 推論＋機械的絞り込み），人間が XPath を検証する（Aladdin による抽出値の目視確認）．一方が作成し他方が検証するという構造的関係は維持しつつ，各エージェントの認知的強みに応じた配置となる．人間は文脈に即した品質判断に長けているが，`//dl[dt[normalize-space()=’給与’]]/dd` をゼロから構築することは得意ではない．Jasmine のセクション選択も同じ原則の適用であり，人間のタスクは構築（XPath の記述，ルールの実装）から認識（値の検証，関連コンテンツの認識）へと移行している．

Why > What——LLM への意図の伝達：プロンプト設計における中核的原則は，制約が何であるかを示すだけでなく，なぜその制約が必要かを LLM に伝えることである．例えば `contains(@class,...)` の使用指示に「クラス属性は複数の値を持つことが多いため」と理由を併記する．Want List モードの意味的記述も同じ原則の適用であり，例えば`"contract": "雇用形態 (employment type)"`という記述により LLM は「雇用形態」「就業形態」「Employment Type」等の異なるラベルを同一フィールドに対応付けることが可能となる．機械的絞り込みにおける中間コンテナの自動検出も同じ原則の発現であり，コンテナのクラス名をハードコーディングするのではなく祖先チェーンを走査して動的に発見することで，サイト固有の設定なしに任意のサイト構造に適応する．

6. 制限事項と今後の課題

SPA 対応：XPathGenie は現在 HTTP リクエストによる生 HTML 取得に依存している．JavaScript でコンテンツを動的にレンダリングするサイト（React, Vue, Angular による SPA）では，空またはスケルトンの HTML が返されるため XPath 生成が不可能となる．評価対象 35 サイト中 7 サイトが SPA であり除外された．ヘッドレスブラウザ（例：Playwright）を利用した JavaScript レンダリング後の HTML 取得は今後の拡張として計画している．

CSS フレームワーク互換性：ユーティリティクラス型フレームワーク（Tailwind CSS 等）では意味的情報が非記述的なクラス名（`w-11/12`, `flex` 等）を持つ深くネストされた要素にエンコードされるため精度が低下する．CSS-in-JS（styled-components 等）はさらに根本的な課題を呈し，クラス名がビルド時のハッシュ値でデプロイごとに変化するため安定した構造的アンカーが存在しない．Yahoo! News（styled-components）のヒット率は 14%であった一方，Cookpad（Tailwind だがセマンティックな HTML 併用）は 100%を達成しており，制限要因はユーティリティクラスそのものではなく DOM 階層における意味的構造の有無である．

圧縮-生成ギャップ：HTML 圧縮パイプラインは空白の正規化やテキストの切り詰めを行うため，LLM が参照する圧縮 HTML と生成された XPath が評価される生 HTML の間に構造的乖離が生じる．例えば`<td>\n 勤務地 \n </td>`は`<td>勤務地</td>`に圧縮され，`text()=’勤務地’` は圧縮 HTML では成功するが生 HTML では失敗する．`normalize-space()` の採用により空白に起因するケースは緩和される．圧縮器が誤ったコンテンツ領域を選択するセクション選択エラーについては，Jasmine が人間介入のエスケープ手段を提供する．

サイト構造の変化：生成された XPath は分析時点の DOM 構造に本質的に依存する．サイトのリデザインにより XPath が無効となる可能性があり，定期的な再分析メカニズムや変更検知システムの導入が望まれる．

モデル依存性：全評価は Gemini 2.5 Flash[11] で実施した．構造化プロンプティングと JSON モード出力に依拠しており他モデルへの移植可能性が示唆されるが，実証的には未検証である．より小規模なモデルでは性能が異なる可能性がある．

再現性：主要 23 サイトの HTML スナップショットは未アーカイブであるが，SWDE および英語評価では全ページをアーカイブし再現可能性を確保した．LLM の非決定性（`temperature=0.1` にもかかわらず）により実行間で軽微な変動が生じうる．21 サイト・3 回実行の再現性調査では，8 サイトが SD=0 の完全安定，全体平均ヒット率 83.1%であった．2 サイト（phget, MRT-nurse）は 0%–100%の極端な分散を示し，非標準的な HTML 構造（div/span レイアウト）と相関していた．

7. 結論

XPathGenie は，HTML 圧縮（97%削減），LLM 推論，マ

ルチページ検証，二段階精緻化を統合し，本番 Web サイトにおいて高い抽出カバレッジを達成した．23 サイトの手動評価でセマンティック精度 95.0%（100 サンプル），SWDE ベンチマーク（22 サイト）では検出フィールド F1=0.689 を達成した．厳密 F1=0.317 との差は主にフィールド発見カバレッジ（46%）に起因し，検出フィールドの 60%が F1=1.0（完全抽出）であることからコア抽出メカニズムは健全である．フィールド発見の改善（圧縮とプロンプティングの強化）が最もレバレッジの高い改善機会である．

圧縮-生成ギャップから重要なエンジニアリング上の知見が得られた．HTML 圧縮器が空白を正規化するため `text()`=述語が検証時に失敗するが，`normalize-space()` の採用により解消され，空白の多いサイトの精度が劇的に向上した（例：ph-10: 0% → 90.8%）．この問題は，LLM 推論の前に入力の表層形式を変換するあらゆるシステムにおいて推論時の表現と実行時の環境との間に意味的整合性のリスクをもたらすという一般的原則を例示している．

「AI Once, DOM Forever」の設計により，初期生成後の運用コストはゼロとなる．n ページのサイトに対し，ページ単位 LLM 抽出の $O(n)$ コストに対して XPathGenie は $O(1)$ の AI コスト＋ $O(n)$ の決定論的 DOM クエリで動作する．実際のサイトあたりのエンドツーエンド所要時間は，自動生成（約 20 秒）と Aladdin による人間の検証（5–15 分）を含めて約 20 分であり，手作業による XPath 記述に通常要する 5–6 時間と比較して大幅に短縮されている．Genie–Jasmine–Aladdin の 3 ツール構成は，機械が作成し人間が検証するという完全なワークフローを確立し，Web データ抽出における従来の分業を逆転させた．

今後の課題として，フィールド発見カバレッジの改善（圧縮とプロンプティングの強化），SPA 対応のためのヘッドレスブラウザ統合，および他の LLM モデルでの性能検証が挙げられる．XPathGenie は，LLM の推論能力を一度限りの知識抽出に集中させ，その成果を決定論的に再利用可能な形式に変換するという設計パラダイムを示した．このアプローチは Web データ抽出に限らず，LLM の出力を永続的なルールとして定着させる応用全般に適用可能であると考える．

謝辞　AI アシスタント テディ（Claude, Anthropic）による開発支援・文書作成支援に感謝する．

参考文献

[1] Kushmerick, N., Weld, D. S. and Doorenbos, R.: Wrapper Induction for Information Extraction, *Proc. IJCAI*, pp. 729–735 (1997).

[2] Dalvi, N., Kumar, R. and Soliman, M.: Automatic Wrappers for Large Scale Web Extraction, *Proc. VLDB Endowment*, Vol. 4, No. 4, pp. 219–230 (2011).

[3] Kohlschütter, C., Fankhauser, P. and Nejdl, W.: Boilerplate Detection Using Shallow Text Features, *Proc. WSDM*, pp. 441–450 (2010).

[4] Lockard, C., Dong, X. L., Einolghozati, A. and Shiralkar, P.: ZeroShotCeres: Zero-Shot Relation Extraction from Semi-Structured Webpages, *Proc. ACL*, pp. 8105–8117 (2020).

[5] Li, J., Xu, Y., Cui, L. and Wei, F.: MarkupLM: Pre-Training of Text and Markup Language for Visually Rich Document Understanding, *Proc. ACL*, pp. 6078–6087 (2022).

[6] Deng, X., Sun, Y., Galley, M. and Gao, J.: DOM-LM: Learning Generalizable Representations for HTML Documents, *arXiv:2201.10608* (2022).

[7] Perini, M., Samardzic, L. and Pozzoli, M.: ScrapeGraphAI: A Web Scraping Python Library That Uses LLM and Direct Graph Logic, *arXiv:2411.13104* (2024).

[8] XPath Agent: Multi-Sample XPath Generation via Two-Stage LLM Pipeline, *arXiv:2502.15688* (2024).

[9] AXE: Adaptive X-Path Extractor: DOM Pruning for Efficient LLM-Based XPath Extraction with Grounded Resolution, *arXiv:2602.01838* (2026).

[10] Hao, Q., Cai, R., Pang, Y. and Zhang, L.: From One Tree to a Forest: A Unified Solution for Structured Web Data Extraction, *Proc. SIGIR*, pp. 775–784 (2011).

[11] Google: Gemini 2.5 Flash, *Google DeepMind* (2025). `https://deepmind.google/technologies/gemini/`

[12] Ferrara, E., De Meo, P., Fiumara, G. and Baumgartner, R.: Web Data Extraction, Applications and Techniques: A Survey, *Knowledge-Based Systems*, Vol. 70, pp. 301–323 (2014).