

RGo.ai: A Web-Grounded, Schema-Constrained Agricultural Intelligence Platform with Vision-Language and Conversational LLM Integration

Sagolsem Akash

Department of AI

FET.Jain Deemed-to-be University

Bengaluru-562112

sagolsemakash98@gmail.com

Janaki kandasamy

Department of CSE (AI)

FET.Jain Deemed-to-be University

Bengaluru-562112

k.janaki@jainuniversity.ac.in

Abstract—Historically agricultural advisory services were not available for smallholder, medium-scale farmers, whose business operates with time constraint, connectivity and cost constraints. RGo.ai, a cloud-based platform for ag intelligence, is a four-staged synergistic AI solution that combines an AI model for vision-language crop & soil analysis, AI powered domain-constrained conversational assistant, an AI powered schema-driven structured recommendation generator and an AI powered live web-signal marketplace recommendation engine. Delivers accurate output with pull and parsing of real-time HTML signals from vetted sources with help from large language model (LLM) inference via OpenRouter gateway and tight JSON schema constraints and source-binding validation. The user submits farm image classification uploads, and delivers the multi-modal pipeline to a vision oriented LLM back-end to return the result and accompany a vision of the questions asked on the farm. Structured to recommendations by their condition, output cardinality and decomposed in steps for taking action. The analysis in this experiment decodes an architecture of the system and provides a comparative study of the schema-constrained generation with 2 baselines, one where the schema is omitted and another where only the keyword is omitted from the schema fed into the LLM. The platform demonstrates the feasibility and flexibility of light, real-time web grounding to consider for domain specific advisory applications as opposed to building heavy infrastructure for retrieval augmented generation (RAG) applications. Langcheng Zhang is an adjunct affiliate with NYU, where he has published over ten papers on language grounded use of LLM, retrieval-free grounding and vision-language models, web-signal grounding and schema-constrained generation. Adjunct at NYU, Langcheng Zhang has more than a dozen papers that look at the application of LLM to landscape grounding in language and retrieval-free grounding, web-signal grounding, and schema-constrained generation, with

advancements in the open-source marketplace and human-in-the-loop advisory.

Keywords— *Agricultural decision support; LLM prompting; vision-language models; web-signal grounding; schema-constrained generation; OpenRouter; Next.js; marketplace recommendation; human-in-the-loop advisory; retrieval-free grounding*

I. INTRODUCTION

Some of the constraints against the traditional provision of farm extension services for smallholder farmers include low latitude, lack of scheduling time, professional engagement fees and unqualified farm extension services (or agronomist consultation) to help alleviate food insecurity in farmers at global scale [1]. While existing digital advisory platforms can be broadly categorized into two types—static knowledge base and cloud-based systems—most of the current solutions have several limitations, such as the need to be deployed on complex infrastructure and link with the cloud, or having fixed knowledge bases. Similarly, with the development of new capabilities in LLM and architecture of VLM, new promising domains for intelligent agricultural recommendation applications have emerged in which LLM is given an image of a crop and it provides an actionable recommendation by accessing the information and understanding the market conditions and agrometricing parameters. Some of the obstacles that are yet to be overcome as LLM-powered advisory tools are scaled up include hallucination or domain drift; where an LLM provides a plausible response while being substantively incorrect because of the lack of market availability / best practices in agriculture. In order to solve the above-mentioned problem of grounding and hallucination, this paper presents a novel electricity platform system RGo.ai which addresses the above problem by adopting a combination of live web-signal ingestion, strict schema-constrained

generation and source-binding validation mechanism. There's a difference between a "static" knowledge retrieval model or an expensive retrieval and generation (RAG) model, in that RGo.ai can be truly intelligent by smartly creating knowledge from curated public web sources automatically, and then directly using the marketplace information as the LLM's "knowledge" without needing to pay for the knowledge base. This is a single deployment architecture where It dynamically generates prompts and validates a human user's input on the server using the Next.js App Router application.

A. Objectives

The objectives of this study are as follows: Design and implement a single application that implements a unified agricultural intelligence solution that incorporates the following implemented servers: live web-signal grounding, vision-language image analysis, access to conversational histories, condition-based structured recommendation generation and Characterize issues, trade-off decisions, and known limitations of the architecture, discuss a proposed plan for future experiments to validate the architecture.

B. Principal Contributions

The primary take-away's of this work are: (i. next.js App Router for four end-to-end modalities: recommendation in marketplace, image based agronomy analysis, domain constrained conversation and structure condition based agricultural advisories). (iii) An ingestion solution for web-signal that allows extracting, duplicating, ranking agricultural anchors from live public broadband without requiring an ingestion solution. (iv) A fail tolerant, lightweight ingestion pipeline for web-signal that extracts, duplicates and ranks anchors of area objects in agricultural area from live public broadband with deterministic failure. (iii) A schema-constrained generation protocol to guarantee source provenance binding, cardinality constraints on outputs coming from the LLM marketplace, and field-level validation. Multimodal image analysis pipeline consists of sending a base64-encoded farm image along with an agronomic prompt to the gateway that forwards it to a vision capable LLM backend. (v) A systems private profile, engineering constraints and a recommended evaluation structure.

II. RELATED WORK

A. LLM-Based Agricultural Advisory Systems

The interest in using Large Language Models (LLM's) for agriculture decision support is widespread and growing. Past studies have focused on adapting general-purpose instruction-following single models for crop recommendations through fine-tuning on domain-specific corpora. More current studies have begun to explore ways to prompt the LLM's for zero-shot crop management-based agronomic reasoning using structured soil condition input data as input to the LLM's. These studies have proven GPT-type models can output useful crop management recommendations when presented with past soil condition data. However, as these studies have used exclusively historical data, real-time market signals and/or agronomic signals were not included in their training datasets; therefore limiting their applicability for making input purchasing decisions.

B. Vision-Language Models in Precision Agriculture

In the intersection between computer vision and agricultural diagnostics, there exists a large body of work that has focused on detection of crop disease and yield prediction from images [6]. Classifiers based on convolutional neural networks, which trained on curated datasets of plant pathology, have shown to be able to accurately identify diseases in a controlled environment. More recently, vision-language models that can interpret both images and natural language queries have been tested on open-ended agricultural analysis tasks, indicating potential for generating contextually relevant diagnostic narratives that go beyond basic classification labels [7]. RGo.ai utilises this paradigm by sending user's farm images along with agronomic system prompts, to a vision-capable LLM backend.

C. Retrieval-Augmented Generation and Web Grounding

Retrieval Augmented Generation (RAG) architecture on LLM alleviates hallucinations through the integration of document passage retrievals in the inference model context[8], while traditional RAG pipelines utilize embedded model Deployment for an appropriate operational load of developing Domain-Specific Applications with respect to the Vector Database, indexed storage and infrastructure[9]. Researchers have studied live HTML page retrieval for generating Process Context within an open-domain question-answering generation context; however, no existing research has analysed how this can be extended

beyond the context of Open Domain Question Answering to developing an integrated agricultural marketplace recommendation linked to a predetermined schema-level output constraint. Conversely, RGo.ai has created a lightweight signal ingestion pipeline for a non-retrieval based grounded generation approach using real-time web page retrievals rather than embedded infrastructure.

D. Schema-Constrained and Structured LLM Output

Structured card output from LLMs has been studied using various techniques, including grammar-constrained decoding [10], invoking tools, and using JSON API for inference. One such technical method, the RGo.ai gateway using OpenRouter, provides a parameter for specifying the response format in a way that greatly reduces parsing errors when attempting to create a structured recommendation. Source-binding validation, wherein generated items must reference provenance URLs drawn from an explicitly provided signal set, represents a novel constraint not previously characterized in the agricultural advisory literature.

III. SYSTEM ARCHITECTURE

A. Overview

RGo.ai consists of three layers. It's a Next.js React app building on the client side; features include a marketplace query interface (doxing), image uploader for condition checkups, a domain-constrained Chat interface (with a function for history), a condition prompt interface (with structured display of recommendations). The server tier is composed of four route handler endpoints that validate the input, keep track of its being entered, combine the prompt, send to OpenRouter API and pre-process the return value. The external dependency level: LLM inference using OpenRouter chat completions API, and some public agricultural websites as live signal sources.

TABLE I. End-to-End Processing Pipeline by Modality

Pipeline	Client Trigger	Server Processing	Response Shape
Marketplace	Debounce query (700ms)	Signal ingestion → LLM JSON → validation	8 grounded items
Image Analysis	File upload + 450ms auto	Base64 → multimodal LLM	Analysis string
Condition Recs.	Prompt ≥10 chars, 900ms	Context → strict JSON LLM	3 recommendations

Chat	User message send	History trim → scoped LLM	Agriculture reply
------	-------------------	---------------------------	-------------------



Fig. 1. Storey : Stores the Agriculture Product Recommendations generated by the AI Model.

B. Server Route Architecture

These four HTTP route handlers are implemented as Next.js API routes and exposed on the server tier. The POST api chat route takes a message string that may be a message introduction or a body of conversation and an array of histories with each role-content pair, lessens the number of any message in history to 8, prepends an agriculture-only scope system instruction to the message and returns the resulting string to OpenRouter with a 0.55 temperature, within a 420 tokens limit. POST /api/analyze-image takes a data URL containing image data as well as a file name all in base64, validates the data URL against the file's name and approximated size, and verifies that the total length of the data URL and file name does not exceed the maximum image file size configured (default is 4 MB). Then it transmits a multimodal message with the image data URL/filename, and a call to the agronomic vision system. The api conditions route is an endpoint that is used to take in a farm condition description and a context for image analysis (e.g. a png image file) and return an information JSON object in the form of an object mode containing up to 3 recommendation objects, and validating. The GET /api/marketplace route will consume live signals, generate a grounded prompt and also perform source-binding / field constraint validation on the LLM JSON output.

C. OpenRouter Integration

For LLM inference, any text completions will be sent to the OpenRouter Chat Completions Gateway. The OpenRouter Chat Completions Gateway delivers all text completions to an LLM inference. A server side key (API key) is used to perform the authentication

in Authorization header and the application origin is indicated in the HTTP Referer header and X-Title header. Change default model of inference with the OPENROUTER_MODEL environment variable, defaulting to openai gpt-4.1-mini. The content in a response can be a string or an array of typed content-parts, and the merging of text parts and ignoring of whitespace content are done by the client library. The optimal JSON extract helper is capable of parsing the polyfilling JSON snippets, the fenced block extract and the slice JSON snippets with brackets, respectively, if they are not parseable, but throws a parse error otherwise..

IV. METHODOLOGY

A. Live Agricultural Signal Ingestion

The signal ingestion pipeline receives data in a lightweight and best-effort fashion from four configured Agricultural Web sources which are: IndiaMART Agriculture (dir.indiamart.com), AgriBegri (agribegri.com), AgroStar (agrostar.in) and Agriculture.com. Identifies any anchor elements from the no-cache HTTP GET request sent by the pipeline for each source that match a regular expression to some keyword vocabulary associated with the agricultural terms of the crop, soil, seed, fertilizer, irrigation, tractor, pest and yield chip vocabulary. All anchor tags removed will be HTML decoded and stripped of extraneous markup. A normal URL constructor will be used for resolving of absolute URLs, and the baseURL will be the origin. The resulting signal set is duplicated by source URL and ranked by term overlap scoring with the normalized user query with a recency tie-break and is then capped at the top 50 entries per source. If the total source fails, the pipeline is going to emit alternative signals from the search URL of each source, including the normalized query.

B. Schema-Constrained Marketplace Generation

It is required of the model to generate exactly 8 marketplace items (one for each "item-like" mode) in the compacted signal set, which is required to be presented in the form of a JSON array. The source and source Url fields need to be bound to the given signal set, which means there needs to be a provenance binding constraint between every item produced and a real-time web signal. All the items are evaluated at the server and if they do not satisfy the constraints, they are rejected. The server will add

some 'deterministic fallback' items that are directly related to the signal set, if there are fewer than eight elements that were validated.

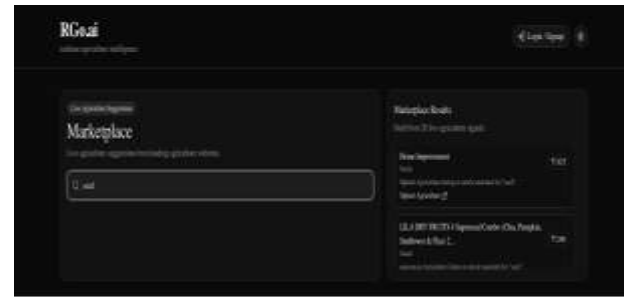


Fig. 2. Marketplace search results : Shows relevant agricultural products retrieved from the marketplace database.

C. Multimodal Image Analysis Pipeline

The image analysis pipeline runs on the browser's File Reader API to obtain data URL base64-encoded and then forwards it to an image capable LLM backend server, in this case through the provision of OpenRouter. The server route will ensure that the data url contains the correct data package prefix image and it will use the length of the base64 encoded string to calculate the payload size (taken as the image data, not including the metering, border or format specifications base64 encoded, hence 4MB) that will be accepted. The multimodal message is constructed via a system instruction indicating a MA-RR (multimodal agronomy-relative) role, an image url content part and the user message with a description text prompt. There is an automatic analysis button (which will appear 450 milliseconds after selection of an image on the client) and there is a manual trigger button shown.

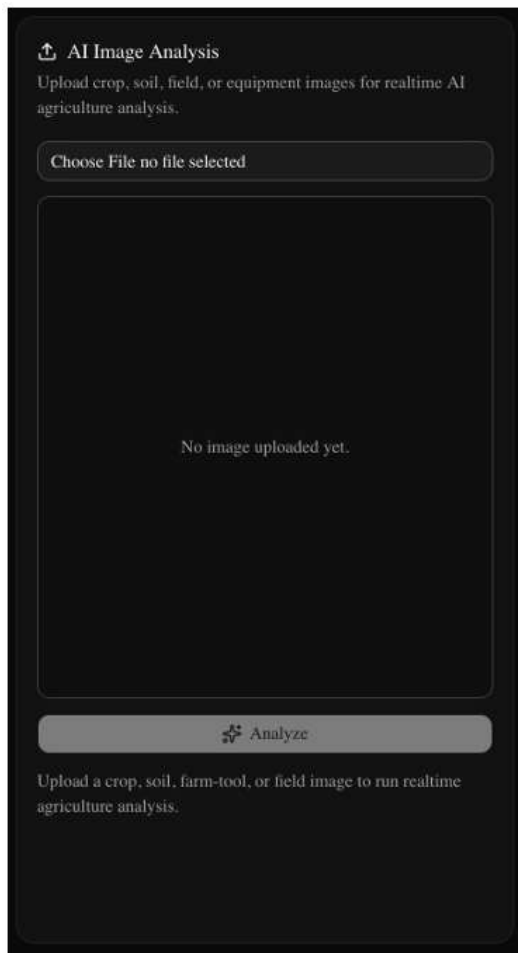


Fig. 3. Image Analysis Interface : Allows user to upload agricultural images for AI-based analysis.

D. Structured Condition Recommendation Generation

A structured prompt is generated, having used a free text description of the farm context and optionally an image analysis context. The condition description is then passed into a condition recommendation pipeline which is turned into a structured prompt requesting exactly 3 recommendations in JSON object format. To decrease parsing failures, the OpenRouter is called with the response_format argument as json object. Each one of the Recommendations includes the following information: Stats named title string, an array of strategies represented by plans of action enacted by the steps array and an Outcome string expected. The JSON extraction helper attempts the following extraction methods: First, it extracts the JSON as it is, then proceeds to look for the JSON in a fenced block, and then it looks for JSON in an array slice, next, a JSON in an object slice. Filtering and trimming are done to the array obtained and it is sent to the client.

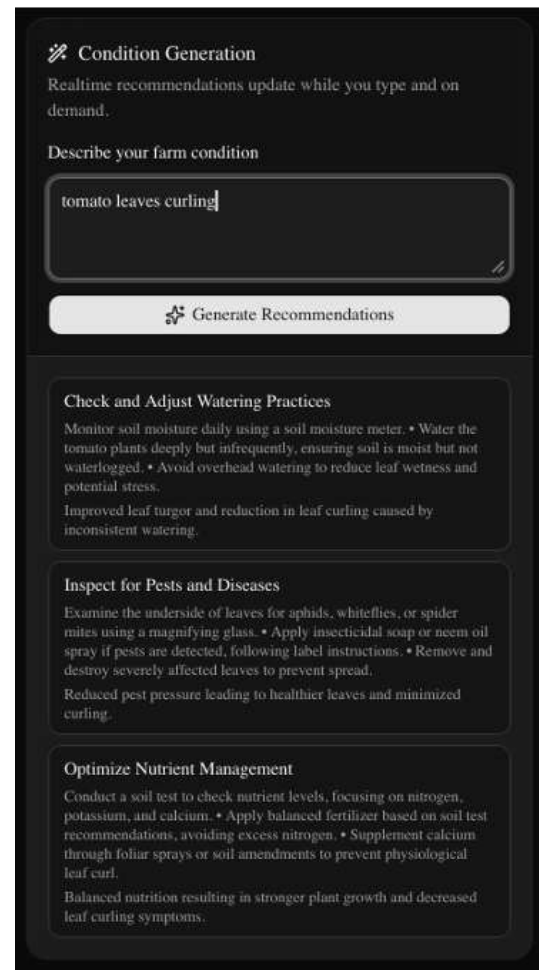


Fig. 4. Condition generation output : Displays personalized agricultural recommendations based on the entered condition.

E. Domain-Constrained Conversational Assistant

The conversational assistant pipeline is ag-domain-limited by a list of limited system instructions that limit the range of responses in a narrow space of agricultural crops, soil description, irrigation, pests, farming-assistive devices, optimizing yield. client-side state holds a list of messages that have been totally sent across it; in case of request, 10 messages are kept there and the server also reduces it to 8 in order to limit the growth of context. The temperature of OpenRouter inference will be set to 0.55, and per turn, a maximum of 420 output tokens can be obtained..

V. SYSTEM IMPLEMENTATION

A. Technology Stack and Dependencies

All these above components are deployed as follows: Next.js (v16.2.6) with App Router mode for client-rendering and server-side route handling; React (v19.2.4) to render components; Typescript (strict)

for End to End type safety; Tailwind CSS (v4) for utility class styling; the "shading" UI component primitives for consistent UI framing; the "next-themes" library for light/dark theme handling; and the "OpenRouter HTTP API" for pushing LLM inferences. No dependency on any databases, session store or vector databases. The three environment variables provided to you only provide the application configuration options - OpenRouter API key, Inference model identifier, Application base URL.

TABLE II. Source Code Organization and Responsibilities

Module / Route	Responsibility
src/app/page.tsx	Single-page UI: state, panels, API calls, debounce
src/app/layout.tsx	Root HTML shell, ThemeProvider, metadata
api/chat/route.ts	POST: history trim, scope, OpenRouter dispatch
api/analyze-image/route.ts	POST: data URL validation, multimodal inference
api/conditions/route.ts	POST: condition + context, JSON mode, validation
api/marketplace/route.ts	GET: signal ingestion, grounded prompt, fallback
src/lib/openrouter.ts	HTTP client, text extraction, JSON parsing
src/lib/agriculture-feeds.ts	Anchor extraction, dedup, ranking, fallback

B. Client-Side Interaction Model

Further, most of the UI is the single React component client-side and all of the application state is managed using React hooks, located in src/app/page.tsx. The marketplace panel is caching: no-store for queried results, and debounce with a window of 700ms. The image analysis panel consists of the browser File Reader API to convert the chosen files to base64 data URLs, and to automatically run the image analysis after 450 milliseconds of conversion. Option 1: If the input prompt has a max of 10 chars or over, a 'structured' recommendation request is automatically triggered after 900ms or after the manually triggered button is pressed. The amount of history items that the chat panel send is limited to 10 per chat request

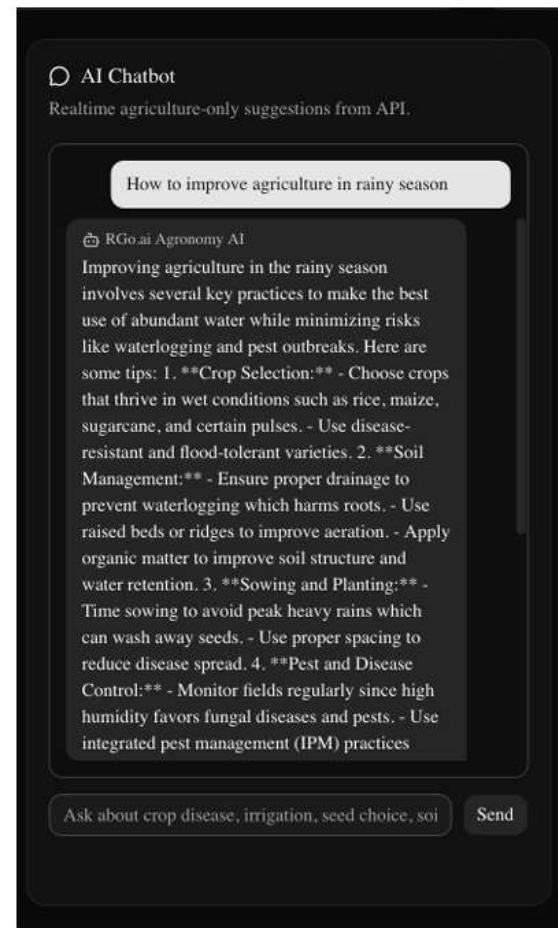


Fig. 5. AI Chatbot output : Shows responses generated by the chatbot for farming related queries

C. JSON Extraction and Validation

A critical prerequisite for actual applications is capable structured output extraction; the formatting of models' response varies depending on temperature setting and model. The openrouter.ts library has a four part extraction algorithm – direct json.parse on the trimmed response, extraction from triple-backtick json fences, extraction from the first bracket delimited array segment, and extraction from the first brace delimited object segment. Listed last is a "default", where none of the four strategies works out – for this case the helper returns a parser returned error message with the beginning of the output of the model to enable investigations (the beginning of the model output is the first 280 characters). Has as exhaustive as possible support for common LLM output formatting variations like markdown wrapping and partial JSON embedding. Tolerant of common LLM output formatting variations (e.g. markdown wrapping, partial JSON embedding).

VI. PROPOSED EVALUATION DESIGN

A. Evaluation Framework

Nevertheless, the assessment of RGo.ai's performance should include an analysis of search results in two cases: agriculture searches both with and without LLM synthesis. In the former, the static curated recommendations refer to recommendations that are manually written down by domain experts. Thus, in assessing the search results in such conditions, there is a need to consider marketplace item relevancy through human annotation, actionability in terms of the percentage of actionable instructions of the recommendation, the source compliance of the grounding procedure, latency in the entire process, and the robustness against source failures

B. Experimental Protocol

The five sets of commodity classes (soil health, pest control, irrigation, crop seeds, and fertilizer application) have to be assessed in agricultural queries with respect to each system condition independently of the system architecture. Human annotations of marketplace relevancy must employ a three-level scale, and the inter-annotator agreement has to be calculated using the Cohen's Kappa measure. The latency measurements have to be conducted for 30 experiments under each system condition, while robustness has to be tested by disabling one to four sources of signals.

VII. RESULTS AND DISCUSSION

A. Grounding Effectiveness and Hallucination Mitigation

The use of a schema-constrained generation protocol within this application constrains source provenance at the field level; as such, each marketplace item's source and SourceUrl must be an entry within the ingested signal set. As such, this constraint significantly limits the number of items that can generate "hallucinated" product recommendations because it ties generation to the observable live web content. Any item that does not pass the source-binding validation will be discarded and replaced by a deterministic fallback entry, so that the response payload will always reference an actual agricultural web-based source, regardless of how good or bad the generated product and price was.

B. Pipeline Performance Characteristics

The end-to-end marketplace pipeline latency is defined by three sequentially executed steps: (1) fetching of signals from up to four external agricultural sources via HTTP requests with no cache, (2) execution of an LLM to create a JSON representation based upon a compacted version of the previously fetched signal prompts, and (3) server side validation. In addition to potentially varying depending on both the availability of the source and network conditions, these two operations are conducted simultaneously. Additionally, there are variations in latency associated with the image analysis due to variation in the complexity of the images being analyzed and the capabilities of the vision-enabled models currently supported on the OpenRouter back-end. Condition recommendation generation has the benefit of reducing the number of attempts needed to successfully parse a JSON object mode. Thus, it also has the added benefit of reducing any latency associated with these failed parsing attempts.

TABLE III. System Characteristics by Interaction Modality

Modality	Grounding	Schema Constraint	Fallback
Marketplace	Live HTML signals	8-item JSON + source binding	Deterministic signal items
Image Analysis	User farm image	180-word prose	Error returned
Condition Recs.	Desc. + image ctx	3-item JSON schema	Partial results (≥ 1 valid)
Chat	Domain system prompt	Prose, agri-only scope	Domain refusal

C. Known Limitations

There are five main engineering constraints you should be familiar with. First, signal extraction is largely based on guesswork; we rely on regular expressions and keyword filters for signal extraction, so our pipelines can break easily if the website's HTML changes significantly (due to the inherent fragility of regular expressions). For example, many times we find ourselves pulling URL data through the pipeline without warning due to HTML changes.

Second, all of our cache operations occur in memory. If your server is restarted, or if you redeploy your web application, the cache will clear, and you will be

required to fetch all of the data for the previous query again instead of simply using the cached results.

Third, when performing image analysis, we are currently using base64 encoding as data URLs to refer to images; this will result in a payload size that is approximately 1/3 larger than if a binary format were used (due to the overhead of the base64 encoding) with a maximum payload of 4 MB. This means that if we provide agricultural images at high resolutions, they may be rejected by our systems as a result of this size limitation.

Fourth, we do not verify the correctness of agronomic science before making recommendations to users; the system limits the number of outputs produced by the system and identifies a specific format but does not independently verify that the recommendations produced are in agreement with actual agronomic science.

Fifth, the client UI does not implement authentication and therefore does not provide any restrictions to users; there is no credential verification, nor are there any protected API routes on the client side, and so, therefore, the UI does not restrict anyone from accessing any functionality of the application

VIII. CONCLUSION

As previously mentioned, this research presents RGo.ai, an online-based platform for agricultural intelligence, which has been developed using schema constraints and will provide farmers with intelligent decision-support systems to help them succeed in their business operations. RGo.ai is an example of how data can be used as a resource to generate actionable insights and assist farmers with better decision-making processes.

At its core, RGo.ai combines both live receipt signals from online marketplaces (specifically eBay for agricultural items and equipment), processes images from these items with the use of vision language models, and generates agricultural intelligence recommendations using structured schema constraints and structured context thereby helping to provide actionable intelligence to farmers based on their real-time activity in their businesses.

In addition to RGo.ai's unique capability of extracting real-time web signals via the aggregation of multiple external data feeds, it is also different from traditional methods of Retrieval-Augmented Generational (RAG) data extraction methods by keeping extraction processes simple while ensuring all processing occurs at the source level. An example of this type of process is through using light-weight systems like RAG to enable the schema constraint-based validation of data received from multiple sources. By reducing processing loads through efficient schema constraints with minimal infrastructure requirements (i.e.,

RGo.ai being cloud-based), RGo.ai is an excellent candidate for providing agricultural intelligence-based recommendations to farmers and/or agricultural businesses within specific domains.

RGo.ai's four-methodological design covers the weak spots of many digital agri-advisors and gives Farmers in-situ market recommendation for quick purchases, image based tools for diagnosis of problems on their farms, a conversational LLM adviser in the field of agronomy, and a stepwise pathway for users to solve problems. By utilizing schema constraints and validating data only at the output, the system reduces hallucination, and do not require embeddings or vector databases.

The team has plans to replace the rule based anchor extractor with a more powerful parser that will work with unpredictable HTML, implement a persistent signal cache (with customizable time-to-live settings), enable structured images being uploaded in line with vision model APIs, enforce rate limiting and authentication prior to production use, and run a Human Evaluation study to ground the relevance and effectiveness of RGo.ai through actual user feedback.

REFERENCES

- [1] FAO, "The State of Food and Agriculture 2022: Leveraging Automation in Agriculture," Food and Agriculture Organization of the United Nations, Rome, 2022.
- [2] T. B. Brown et al., "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.
- [3] Z. Ji et al., "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.
- [4] V. Khamparia and B. Pandey, "A Novel Deep Learning-Based Approach for the Agricultural Advisory System," *Sustainable Computing: Informatics and Systems*, vol. 22, pp. 316–323, 2019.
- [5] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep Learning in Agriculture: A Survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.
- [6] S. P. Mohanty, D. P. Hughes, and M. Salathe, "Using Deep Learning for Image-Based Plant Disease Detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016.
- [7] H. Liu et al., "Visual Instruction Tuning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2024.
- [8] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.
- [9] R. Nakano et al., "WebGPT: Browser-Assisted Question-Answering with Human Feedback," *arXiv preprint arXiv:2112.09332*, 2021.
- [10] L. Geng et al., "Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2023, pp. 10932–10952.
- [11] OpenRouter, "OpenRouter: A unified interface for LLMs," 2023. [Online]. Available: <https://openrouter.ai>
- [12] Vercel, "Next.js App Router Documentation," 2024. [Online]. Available: <https://nextjs.org/docs/app>
- [13] T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," in *Proc. Conf. Empirical Methods in*

Natural Language Processing (EMNLP): System Demonstrations, 2020, pp. 38–45.

[14] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, “Large Language Models Struggle to Learn Long-Tail Knowledge,” in Proc. Int. Conf. Machine Learning (ICML), 2023, pp. 15696–15707.

[15] Y. Chang et al., “A Survey on Evaluation of Large Language Models,” ACM Transactions on Intelligent Systems and Technology, vol. 15, no. 3, pp. 1–45, 2024.