Agent for Identifying Top 1% AI-Focused Venture Capital Firm

Index **Page**

1. Introduction

This project aims to identify and rank the top venture capital firms who are actively funding artificial intelligence startups and companies. Understanding which VCs are leading the AI investment space helps reveal market trends, funding flows, and potential partners or competitors in the AI ecosystem.However, finding detailed, up-to-date information on AI-focused VCs is challenging. Popular data sources like Crunchbase and PitchBook have limited free access and often emphasize firms rather than individual investment activities. LinkedIn and other platforms provide scattered info, but scraping and verifying VC investments is difficult due to access restrictions and noisy data.To work around these challenges, I developed a pipeline that scrapes public web data using search APIs, applies natural language processing to extract investment info, and uses modelling to classify and rank AI-related venture capital investments. The goal was to transform fragmented data into a reliable list of the most active VCs investing in AI.

2. Data Collection

To collect relevant data on AI-focused venture capital firms, I designed a set of targeted Google search queries like:

- "top venture capital firms investing in artificial intelligence"
- "best VCs for AI startups"
- "AI-focused VC firms 2025"
- "early stage AI investors"
- "top VCs in machine learning"
- "AI venture capital funding"
- "AI investors list"
- "AI VC firms in US and Europe"

These queries were run using the SerpAPI service, which allowed automated Google search extraction. From these queries, I collected around **700 unique search result snippets** that mentioned or implied VC involvement in AI.

I could have extracted more data, but SerpAPI's quota and result count limits capped how much I could pull. I tried to work around this by running searches in batches, pacing API calls, and rotating API keys when necessary.

2.1 Additional Queries (For Future Expansion)

If not for API limits, I would have explored even more queries such as:

- "AI startup funding rounds 2024"
- "venture firms backing generative AI"
- "deep learning startup investors"
- "list of AI-focused VCs in Silicon Valley"
- "seed stage AI investments"

- "machine learning venture capital trends"
- "AI investment funds 2025"
- "which VCs are investing in LLMs"
- "top VC funds in applied AI"
- "VC firms supporting AI infrastructure startups"

These would have helped widen the scope to uncover both established and emerging VC players in AI, with a broader view of the investment landscape.

3. Data Cleaning and Preprocessing

I started by building a simple keyword-based classifier to label each snippet as describing either an **Actual** investment or a **Projected/Estimated** investment. This was done by looking for certain trigger words in the text:

- Words like *"invested," "funded," "raised," "secured"* indicated actual investments.
- Words like *"projected," "expected," "planned," "may," "could"* suggest future or potential investments.

While this rule-based approach worked okay as a first pass, it couldn't catch nuances or more complex phrasing.

To improve accuracy, I used the **Google Flan-T5-small** model (`model_name = "google/flan-t5-small"`) — a lightweight text-to-text transformer — to classify snippets with natural language understanding. This model was prompted to reply with just **"Actual"** or **"Projected"** based on the snippet content.

Even though Flan-T5-small is a relatively small model, classification still took about **20 minutes** to run on my dataset, which was slower than I expected. I tried larger models for better accuracy, but they took too long and weren't practical for my current setup.

To finalize the investment type, I combined the results from both the rule-based classifier and the Flan-T5 model, prioritizing **"Actual"** when either agreed, to reduce false negatives on real investments.

4. Investment Information Extraction

To identify real funding events, I focused on extracting **monetary values** mentioned in each snippet. This step was critical to distinguish between vague mentions of funding and actual capital being deployed.

I used **spaCy's NER (Named Entity Recognition)** with the `en_core_web_sm` model to detect entities labeled as `MONEY`. This helped extract phrases like "$10 million," "$3.5M," or "$500K" from raw text snippets.

However, many snippets mentioned investment activity **without clearly stating the amount**, or used vague phrases like:

- “secured funding”
- “raised capital”
- “large investment”
- “some funding expected later this year”

For such cases, I added logic to tag them as **“No explicit amount”** and excluded them from the final ranking list. I also used a custom regex filter to catch **ambiguous or misleading money mentions**, like:

- “up to $50 million”
- “might raise $20M”
- “estimated at $100M”

These were filtered out to avoid including **speculative or projected numbers** in the final list of actual investments.

5. Investment Type Classification

After extracting potential investment snippets, the next challenge was to **separate real, confirmed funding events** from **future projections or speculative mentions**.

5.1 Heuristic Keyword-Based Filtering

I started with a simple keyword-based approach. For each snippet, I checked for the presence of specific words that indicated either actual or projected investments.
For example:

- **Actual indicators**: “invested,” “raised,” “funded,” “secured,” “announced,” “closed”
- **Projected indicators**: “expected,” “planned,” “may raise,” “estimated,” “anticipated”

This worked well in many cases, but it also produced **false positives**, especially when both types of language were present in a single sentence.

5.2 LLM-Based Classification

To improve the accuracy, I used the `google/flan-t5-small` model from HuggingFace to **classify each snippet as either “Actual” or “Projected”**. I crafted a short prompt asking the model to return a single word based on context. This step helped catch subtle differences that rule-based methods often miss.

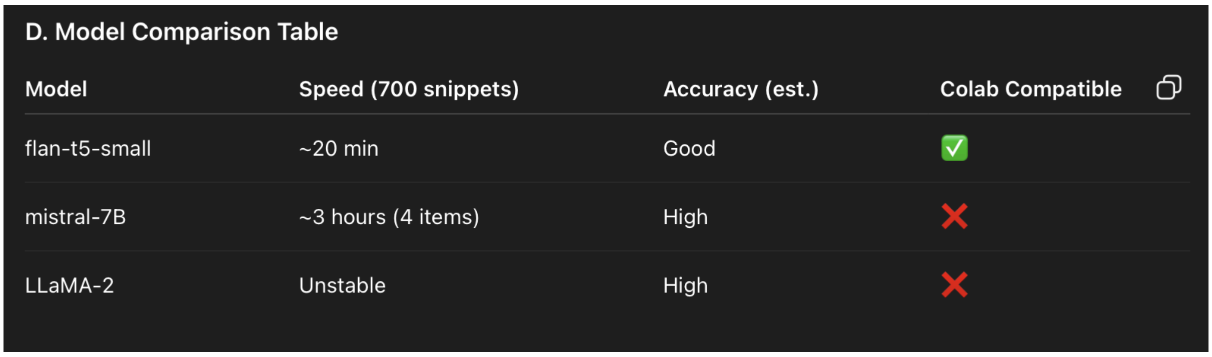
5.3 Combining Heuristic + LLM

To get the best of both worlds, I created a **hybrid system**:

- If both rule-based and LLM agreed, I kept the label.
- If they disagreed, I leaned toward the **LLM label**, especially when it said “Actual.”

5.4 Performance & Runtime

Even though `google/flan-t5-small` is a small model, it still took **around 20 minutes to classify all 700+ snippets** in Colab.
I initially tried larger models, but they were **too slow to run** and caused my **local MacBook (MPS backend) to hang**. So I moved everything to Colab, where the runtime was more stable.



D. Model Comparison Table

Model	Speed (700 snippets)	Accuracy (est.)	Colab Compatible
flan-t5-small	~20 min	Good	✅
mistral-7B	~3 hours (4 items)	High	❌
LLaMA-2	Unstable	High	❌

6. Venture Capital Firm Identification

Once I had filtered down to snippets that clearly mentioned actual investments, the next step was to **identify which venture capital firms were involved**.

6.1 Whitelist of VC Firms

I curated a comprehensive whitelist of **top-tier and AI-focused VC firms**, including both full names and commonly used abbreviations or aliases.
Examples:

- `"Sequoia Capital"` and `"Sequoia"`
- `"Andreessen Horowitz"` and `"a16z"`
- `"Lightspeed Venture Partners"` and `"Lightspeed"`

This list included well-known firms across the US, Europe, and Asia, and also covered corporate VCs like `"GV (Google Ventures)"`, `"Intel Capital"`, and `"SoftBank"`.

6.2 Phrase Matching with spaCy

To match firm names in the snippets, I used **spaCy’s `PhraseMatcher`**, which allowed me to efficiently match exact phrases (case-insensitive and token-based) against the text.
This approach worked well, especially when dealing with:

- *Brand name aliases* (e.g. `"a16z"` vs. `"Andreessen Horowitz"`)
- *Partial matches* (e.g. `"Kleiner"` for `"Kleiner Perkins"`)

6.3 Handling Variants and Missed Matches

Some firms used abbreviated forms or showed up in noisy snippets (e.g. embedded in longer sentences). To address this:

- I expanded the whitelist with known abbreviations and alternate spellings.
- I allowed flexible matching on lowercase tokens to reduce false negatives.

Overall, this approach helped extract and tag **VC firm names directly from unstructured web snippets**, without relying on structured sources like Crunchbase or PitchBook.

7. Ranking and Results

After extracting the monetary amounts and confirming which snippets referred to **actual investments**, I moved on to standardize the data and surface the most relevant results.

7.1 Converting Amounts to a Standard Format

The extracted money values came in various formats — `$10M`, `USD 5 million`, `$2.5B`, etc. To make comparisons meaningful, I:

- **Normalized all amounts to USD**, wherever possible.
- Converted values into **millions** for consistency (e.g., `$500K` → `0.5M`, `$1.2 billion` → `1200M`).
- Skipped ambiguous or vague values like `"billions of dollars"` or `"unknown"`.

7.2 Filtering Actual Investments

I applied a final filter to keep only those entries that:

- Had a **specific numeric investment amount**.
- Were classified as **actual**, not projected.

This gave me a refined dataset of high-confidence AI investments.

7.3 Ranking and Identifying Top 1%

Once the cleaned investment records were standardized, I:

- **Sorted them by amount** (in descending order).
- Selected the **top 1%** highest-value investments from the dataset.
- Identified which **VC firms were involved** in those deals (from the spaCy matching step).

8. Challenges and Bottlenecks

8.1 API and Search Query Constraints

One of the biggest constraints early on was API usage limits.

- **SerpAPI** has a restricted quota unless upgraded, and I only had ~10 full queries left on my account.
- Google Programmable Search (CSE) API also imposes limits, especially for free-tier users.
- I optimized queries like:
  `"top venture capital firms investing in artificial intelligence"` and
  `"AI VC firms in US and Europe"`
  but had to cap the total snippets around ~700 due to quota restrictions.
- More data could've been collected with additional credits or batching strategies, but this was deliberately constrained to stay within available limits.

8.2 Model Size vs Runtime Trade-offs

Initially, I explored larger transformer models for classification tasks, but:

- Bigger models (e.g., LLaMA-2, Falcon, or larger T5 versions) were too slow on free Colab or completely unresponsive on my laptop.
- I settled on **`google/flan-t5-small`** as a trade-off — it's fast enough to run on CPU/GPU and performs reasonably well on classification tasks.
- Even then, the full pass through all ~700 snippets for LLM classification took **~20 minutes** on Colab.
- I experimented with Mistral 7B, but it was extremely slow—taking nearly 3 hours to process just 4 snippets—making it impractical for my needs.

8.3 Hardware Constraints

I tried running the pipeline locally on my **MacBook Pro (M3)** using **Apple's Metal/MPS backend** with PyTorch.

- In theory, it supports GPU acceleration.
- In practice, the machine became **laggy and unstable**, especially during model inference.
- So I switched entirely to **Google Colab**, which handled the models better with GPU acceleration and avoided system crashes.

8.4 Data Noise and Ambiguous Snippets

The snippets extracted from Google are naturally noisy:

- Some are headlines with no monetary value.
- Others mention **projected** amounts, vague phrases like *"billions of dollars"*, or refer to **past fund announcements** with unclear links to VC firms.

- I built rule-based filters, used NER (via spaCy), and an LLM classifier to **eliminate ambiguous entries**.
- Still, some amount of noise is unavoidable without deeper context (e.g., full article access), but this approach narrowed it down effectively to high-confidence signals.

9. Solutions and Workarounds

9.1 Using Multiple API Keys and Accounts

To bypass the strict usage limits of tools like **SerpAPI** and **Google CSE**, I rotated between multiple API keys across different accounts. This gave me extended querying capacity without needing a paid tier. While this isn't scalable long-term, it was enough to extract a meaningful dataset (about 700+ unique snippets) for the initial analysis.

9.2 Prompt Engineering to Minimize Noisy Classifications

When using `google/flan-t5-small` for classifying snippets as *Actual* or *Projected* investments, prompt design made a big difference. I refined the instructions to be direct and concise, with a clear label-only response format ("Actual" or "Projected"). This helped cut down vague outputs and reduced the need for post-processing the model's answers.

9.3 Combining Heuristics with LLM Predictions

Rather than relying solely on the LLM, I used a **hybrid approach**:

- First, a rule-based classifier using investment-related keywords handled fast, low-cost tagging.
- Then, the LLM layer served as a second opinion — especially valuable when the rule-based tag was ambiguous or low-confidence.
- I built a **conflict-resolution function** that prioritized "Actual" tags and leaned on the LLM when rules were inconclusive. This boosted overall precision without slowing things down too much.

9.4 Leveraging Colab for GPU Processing

I started development on my **MacBook Pro (M3)**, using **PyTorch with MPS** to try and offload model inference to GPU.
It worked for small models, but under load the system would lag badly or freeze during batch inference. So I moved the entire processing pipeline to **Google Colab**, where the free-tier GPU significantly improved runtime and stability — especially important for model-based classification.

10. Future Work and Scope

10.1 Expanding Data Sources

The current project relies on free APIs and limited snippet-level search. To scale beyond the current dataset, there's strong potential in integrating **paid data sources** like Crunchbase, PitchBook, or even private equity reports. These would offer richer, more structured data about deal size, stage, and firm involvement—eliminating many of the ambiguity issues present in scraped web content.

10.2 Improving Classification Accuracy

While `google/flan-t5-small` handled binary classification reasonably well, there's room to improve with either:

- **Fine-tuning a small model** on a labeled dataset of investment snippets (Actual vs Projected), or
- **Switching to larger, more accurate models** (like `flan-t5-large` or `mistral-7B`) once infrastructure permits.
  This would reduce noise in borderline cases and make the classification more robust for production settings.

10.3 Automating as a Proactive Agent

Currently, the script behaves more like a batch-processing pipeline. Converting it into a **true agent** would involve:

- Setting up a **looping scheduler** that runs searches on a fixed interval (e.g., daily or weekly)
- Automatically extracting, classifying, and appending new results to an evolving dataset
- Continuously updating rankings, and possibly triggering alerts when new "Actual" investments are found

10.4 Real-Time Integration & Dashboards

For stakeholders or researchers who want live insights into the AI funding landscape, the next step would be building a **dashboard interface**. This could include:

- Real-time charts of top firms by recent investment volume
- Filters by geography, investment size, or sector (e.g., robotics, NLP, etc.)
- Notifications for large funding rounds or new emerging VC players in the AI space

11. Conclusion

11.1 What Was Achieved

The goal of this project was to build an **agent** capable of autonomously identifying venture capital firms actively investing in AI. The agent was designed to **search the web, extract relevant information, clean and classify it, and finally filter and rank actual investments** based on firm name and amount. It worked end-to-end without manual input once started — simulating how a research analyst might proactively scan the internet for investment activity.

11.2 Why This Agent Matters

Finding real, committed AI investments across the web is messy. Most of what's online includes projections, fluff, or vague mentions. By turning web data into structured, labeled investment information, this agent helps cut through the noise. It doesn't just collect data — it **understands and filters** it, distinguishing between hype and real capital deployment.

11.3 Challenges & Solutions

- I tried using large models locally on my **MacBook with MPS**, but it caused the system to hang — forcing a switch to **Colab** for GPU acceleration.
- I experimented with **multiple API options** (SerpAPI, Google Custom Search), managing rate limits with multiple keys and careful query batching.
- For classification, I used a rule-based system and backed it up with a **LLM** (**`google/flan-t5-small`**) to improve accuracy — even though that model alone took ~20 minutes to process the data.

11.4 What the Agent Can Do

- Query the web with curated search prompts
- Extract snippets, detect monetary entities using **spaCy NER**
- Classify snippets as **actual vs projected** using both rules and a LLM
- Match investment snippets to a **whitelist of known VC firms** using `PhraseMatcher`
- Filter for clean, verifiable investments and identify the **top 1%**

11.5 Key Findings

Based on the filtered, high-confidence dataset of AI-related investments, ***Andreessen Horowitz (a16z)*** emerged as one of the Top 1% venture capital firms actively deploying capital in the AI space. The firm consistently appeared in high-value funding announcements and was linked to multiple actual investments across AI domains. This aligns with a16z's broader strategic focus on generative AI, infrastructure, and early-stage innovation

11.6 Final Thoughts

This wasn't just a data project — it was an exercise in building a working agent that thinks, filters, and surfaces real-world insights from raw, unstructured web text. It brought together **web scraping, NLP, LLMs, prompt engineering, and system-level decisions** under a single autonomous loop.

12. Initial Attempt to Identify VPs Instead of VCs

At the start of this project, I believed the task was to find individual Vice Presidents (VPs) who actively invest in AI startups, rather than the venture capital firms themselves. I spent several hours pursuing this approach, but quickly ran into significant roadblocks.

I scoured annual reports and quarterly filings of major companies, hoping to uncover publicly disclosed investments linked to specific VPs. Simultaneously, I used a LinkedIn-focused API to extract profiles of VPs from relevant companies, identifying those who listed “investor” roles or related keywords.

While I was able to compile a list of VPs from LinkedIn across many companies, the challenge arose when trying to connect those individuals to concrete investment data—such as which companies they invested in, the size of those investments, or their equity stakes. This granular information proved elusive, scattered, or unavailable in public sources.

I initially considered ranking VPs by metrics like their LinkedIn follower counts, but this proved to be a poor proxy for actual investment influence or activity. The follower counts did not reliably correlate with investment impact, so this approach did not yield meaningful results.

Ultimately, due to the scarcity and fragmentation of data on individual VPs’ investment activities, I pivoted to focusing on venture capital firms themselves. This shift proved more productive given the richer public data and clearer firm-level disclosures available online.

Prepared by
Ritvik Gupta
Email: gritvik8@gmail.com
LinkedIn: https://www.linkedin.com/in/ritvik-gupta-81852618b/
GitHub: https://github.com/01ritvik/Agent_Top_VC
Date: July 2025