

Handbuch: Lokale Investigativ- & OSINT-Recherche-Pipeline

Version: 2.1

Stand: Oktober 2026

Status: In Entwicklung / Basis-Architektur

Sieht man sich die tägliche Berichterstattung an, bekommt das Wort „NEWS“ eine neue Bedeutung:

„Nicht ein Wort stimmt“

Zielsetzung & Grundphilosophie

Ziel der Pipeline ist eine halbautomatisierte, methodisch abgesicherte und lokalisierte Recherche zu komplexen Themen aus Politik, Wissenschaft, Technologie und Gesellschaft.

Das System adressiert dabei drei zentrale Herausforderungen:

1. Bias-Klärung: Gegenüberstellung von offiziellen Narrativen (Regierungen, Militär, Konzerne) und dem sogenannten „Flur-Funk“ (Grauliteratur, Vor-Ort-Schilderungen, Insider-Blogs).
2. Wissenschaftliche & Technische Klärung: Rekonstruktion harter Fakten und Spezifikationen hinter verkürzten oder PR-getriebenen Darstellungen.
3. Investigation & Zensurfreiheit: Erfassung von Informationen und Quellensegmenten, die in Mainstream-Suchmaschinen untergehen oder von Cloud-LLMs aufgrund restriktiver Inhaltsfilter (Guardrails) abgelehnt werden.

Kern-Prinzip:

- Lokale Verarbeitung: Der Download der Volltexte sowie deren Analyse erfolgen vollständig auf der eigenen Hardware (Pop!_OS, qwen2.5:14b via Ollama).
- Quellen-Pluralismus: Neben Mainstream-Medien werden gezielt Blogs, Foren, Dokumenten-Ablagen (.pdf) und Social-Media-Quellen einbezogen.
- Archivierbarkeit: Jede Recherche erzeugt eine zeitgestempelte, lokale Ordnerstruktur mit unveränderlichen Rohdaten und strukturierter Analyse.

Architektur-Leitfaden: Optimierung mehrstufiger Recherche-Pipelines mit lokalen Sprachmodellen (Qwen2.5 / 14B)

1. Ausgangslage und Problemstellung

Beim Einsatz lokaler Sprachmodelle mittlerer Parametergröße (z. B. Qwen2.5 mit 14 Milliarden Parametern) in automatisierten Recherche-Prozessen (Search/RAG-Pipelines) kommt es häufig zu schwankenden Ergebnissen. Während reine Wissensabfragen und einfache Recherchen zufriedenstellend verlaufen, sinkt die Treffergenauigkeit bei komplexen, mehrsprachigen oder lokal spezifischen Fragestellungen spürbar.

Die Ursache liegt in der fundamentalen Funktionsweise von LLMs bezüglich Spracheffizienz, Tokenisierung und Kontextverarbeitung. Um eine verlässliche Pipeline aufzubauen, muss das Recherche-Modell an die mathematischen und linguistischen Stärken und Schwächen kleinerer Modelle angepasst werden.

2. Grundlegende Funktionsweise und Leistungsgrenzen lokaler LLMs

2.1 Das Ressourcen- und Sprachen-Gefälle (High- vs. Low-Resource)

Sprachmodelle verarbeiten Informationen nicht rein logisch, sondern auf Basis statistischer Muster aus ihrem Trainingsmaterial.

- **Englisch als Hauptdomäne:** Über 50 % bis 80 % der Trainingsdaten bestehen aus englischen Texten. Das Modell besitzt hier seine höchste logische Präzision, das beste Regelverständnis (*Instruction Following*) und die geringste Fehlerquote.
- **Große Weltsprachen (Deutsch, Französisch, Spanisch):** Werden vom Modell gut verstanden und generiert, erfordern bei der internen Rechnungslogik aber mehr Aufwand als Englisch.
- **Spezifische Lokalthemen & Dialekte:** Sobald Anfragen administrative Spezialfälle, seltene Begriffe oder Dialekte betreffen, sinkt die Verlässlichkeit. Das Modell greift dann auf globale Entitäten zu, verliert aber im Fließtext an semantischer Präzision.

2.2 Das Tokenisierungs-Problem

LLMs verarbeiten Text in mathematischen Einheiten (*Tokens*).

- Ein englisches Wort entspricht im Schnitt einem Token.
- Nicht-englische Texte (besonders mit Sonderzeichen, Umlauten oder komplexer Morphologie) werden vom Tokenizer in deutlich mehr Teil-Tokens zerlegt.
- **Konsequenz:** Bei fremdsprachigen Texten füllt sich das Kontextfenster (*Context Window*) um ein Vielfaches schneller. Das Modell verbraucht wertvolle Rechenkapazität für die reine Wort-Rekonstruktion statt für die inhaltliche Logik.

3. Die Engpässe einer 3-stufigen Recherche-Pipeline

In einer klassischen mehrstufigen Recherche (1. *Suchanfrage generieren*, 2. *Ergebnisse abrufen/filtern*, 3. *Antwort synthetisieren*) entstehen bei lokalen 14B-Modellen zwei Hauptfehlerquellen:

3.1 Der Hinweg (Query-Generierung)

Sucht das Modell in einer Fremdsprache nach Informationen, neigt es bei direkter Übersetzung deutscher Prompts zu unnatürlichen Satzkonstruktionen.

- *Erkenntnis:* Für Suchmaschinen sind keine ausformulierten Sätze erforderlich, sondern präzise Schlüsselbegriffe (*Keywords / Keyphrases*).
- *Stärke des Modells:* Das Aufrufen fester Begriffe, Entitäten und behördlicher Fachausdrücke (z. B. *Carte Grise, Péage flux libre*) gelingt auch kleineren Modellen exzellent, da diese als feste Wissensknoten abgespeichert sind.

3.2 Der Rückweg (Verarbeitung der Quelltexte als Haupt-Flaschenhals)

Werden bei der Webabfrage 5 bis 10 rohe Webseiten in den Kontext des Modells geladen, bricht die Genauigkeit ein:

1. **Token-Explosion:** Die Rohdaten (inkl. Navigationsmenüs und Werbemüll) überfordern den Arbeitsspeicher des Modells.
2. **Grammatik-Schatten (Halluzination beim Lesen):** Bei komplexen Satzkonstruktionen in Nicht-Hauptsprachen versteht ein 14B-Modell oft nur die Einzelbegriffe, verfehlt aber grammatikalische Bedingungen (Verneinungen, Fristen, Ausnahmeregelungen).
3. **Detailverlust:** Wichtige Einschränkungen im Quelltext gehen in der Masse des Fließtextes verloren.

4. Ziel-Architektur und Optimierungs-Prinzipien

Um die Verlässlichkeit der Pipeline mit Qwen2.5 drastisch zu steigern, sollte das Recherche-Konzept nach folgenden Prinzipien überarbeitet werden:

- [1. Prompting & Logik] —→ Steuerung & Denkprozesse strikt auf ENGLISCH
- [2. Keyword-Extraktion] —→ Gezielte Abfrage offizieller FACHBEGRIFFE der Zielsprache
- [3. Web-Retrieval] —→ Auf Top-3 bis Top-5 Ergebnisse beschränken (Markdown/Reinextrakt)
- [4. Vorverarbeitung] —→ Vor-Filterung / Kompression der Daten (Chunking / Reranking)
- [5. Synthese] —→ Auswertung auf Englisch/Deutsch, finale Ausgabe im Wunschformat

4.1 Modultrennung und Regelwerk für die Stufen

Stufe 1: Anweisungs- und Denkebene (System-Prompts)

- **Grundsatz:** Die interne Logik, System-Prompts und Anweisungen an das Modell werden auf **Englisch** formuliert.
- **Nutzen:** Das 14B-Modell nutzt seine maximale Kapazität für die Befolgung von Regeln und Strukturvorgaben.

Stufe 2: Generierung der Suchbegriffe (Query Formulation)

- Das Modell wird angewiesen, keine ausformulierten Sätze zu suchen, sondern die **3 bis 5 exaktesten nativen Fachbegriffe** der Zielsprache auszugeben.
- *Beispiel-Prompting:* "Identify the official administrative terms and native keyphrases used in France for [Thema]. Output only the top 4 search terms."

Stufe 3: Datenabruf und Vorfilterung (Web Scraping & Cleaning)

- **Datenmenge reduzieren:** Maximal 3 bis 5 hochrelevante Seiten abrufen statt 10+ Treffer.
- **Boilerplate entfernen:** Vor der Übergabe an das LLM müssen HTML-Ballast, Header, Footer und Skripte entfernt werden (z. B. Umwandlung in sauberes Markdown).
- **Chunking / Verdichtung:** Falls technisch möglich, wird ein vorgeschaltetes Embedding- oder Reranker-Modell genutzt, das nur die relevantesten Absätze an das Hauptmodell übergibt.

Stufe 4: Auswertung und Synthese (Extraction & Output)

- **Strukturierte Extraktion erzwingen:** Das Modell erhält den Auftrag, Fakten in ein klares Zielformat (z. B. JSON oder Listenstruktur mit den Feldern *Bedingung*, *Frist*, *Ausnahme*) zu extrahieren. Das verhindert das "Erzählen" und freie Halluzinieren.
- **Finale Ausgabe:** Die Übersetzung und Aufbereitung der finalen Antwort in die gewünschte Zielsprache (Deutsch) erfolgt erst im allerletzten Schritt der Pipeline.

5. Zusammenfassung der Kernerkenntnisse für die Praxis

1. **Englisch als Steuerungssprache:** Nutze Englisch für alle Zwischenschritte, Prompts und logischen Ketten innerhalb der Pipeline.
2. **Fachbegriffe statt Satzübersetzung:** Lass das Modell native Keyphrases und Entitäten der Zielsprache generieren, statt deutsche Sätze 1:1 zu übersetzen.
3. **Kompression vor Analyse:** Das Nadelöhr ist das Einlesen fremdsprachiger Webseiten. Filtere Müll rigoros heraus, bevor der Text an Qwen2.5 übergeben wird.
4. **Strikte Formate:** Nutze klare Extraktionsvorgaben (z. B. JSON/Key-Value), um die Auswertung auf dem Rückweg stabil zu halten.

System-Architektur im Überblick

Die Pipeline teilt sich in drei aufeinander aufbauende Phasen:

Plaintext

[PHASE A: STRATEGIE]	—> Concept & Prompt-Design (Web-LLM / Lokales LLM)
[PHASE B: TRIAGE]	—> Relevanz-Check & Feinschliff (SearXNG / WebUI / Browser)
[PHASE C: HARVESTING]	—> Volltext-Download, Skript-Analyse & Archivierung (CLI)

Die Phasen A und B sind interaktiv und zyklisch interaktiv, d.h. die Vorschläge werden geprüft und „nach Gefühl“ bewertet, d.h. weisen die erzeugten Linklisten oder eine Kurzauswertung darauf hin, dass die gewünschten Ergebnisse erzielt werden können. In Phase B kann mit mehreren Methoden gearbeitet werden, zwischen denen je nach erzieltm Ergebnis gewechselt werden kann. Ein Teil der Projekte lässt sich vermutlich bereits hier abschließen, wenn sich zeigt, dass

- die gewünschten Ergebnisse durch eine Kurzauswertung vorliegen,
- sich abzeichnet, dass die gewünschten Informationen nicht gefunden werden können,
- die Recherche auf falschen Grundannahmen beruht und sich daher erledigt.

Ziel ist es, die interaktive Phase zeitlich möglichst zu begrenzen und die eigentliche, länger dauernde Auswertung dem System zu überlassen. Die Prompts für die Kurzauswertung sollten so beschaffen sein, dass sie hauptsächlich eine schnelle Entscheidung über weitere Schärfung der Suchanfrage oder Abschluss der Recherche erlauben, aber nicht unbedingt in die Tiefe gehen.

Phase C kann je nach Datenumfang fast im zeitlichen Rahmen der interaktiven Arbeit erledigt werden oder auch eine längere eigenständige Bearbeitung des Themas durch das System umfassen. Die Prompts sind hier schärfer zu fassen und die Auswertung ggf. mehrfach mit angepassten Prompts vorzunehmen.

3. Der Recherche-Workflow im Detail

Phase A: Prompt-Design & Such-Matrix (A__init_queries.py)

Ein vorgeschaltetes LLM (z. B. ein Web-LLM oder ein lokales Modell) wird als Recherche-Strategie eingesetzt. Da das Modell in dieser Phase selbst keine Suchen ausführt, greifen keine Inhaltsfilter.

Das Programm erstellt zunächst ein leeres Datentemplate, in das die Analysenziele eingetragen werden: Thema, Zielrichtung (z. B. "Bias-Analyse"), gewünschte Quellentypen. Das Template ist so aufgebaut, dass in den folgenden Schritten jeweils die kompletten Zusammenhänge in die Ergebnisdateien übertragen werden können. Der Rechercheur ist somit in der Lage, in jeder Datei den Gesamtzusammenhang sehen zu können; ein Querblättern in mehreren Dateien entfällt

Im zweiten Aufruf mit eingetragenen Zielen erzeugt das Programm passende Suchstrings gemäß Vorgabe. Wiederholte Aufrufe führen zur Erzeugung weiterer Suchstrings, wobei über erzeugte Suchstrings Buch geführt wird, also keine Wiederholungen auftreten. Es ist daher jederzeit möglich, die Pipeline neu aufzusetzen, wenn die Ergebnisse nicht zufriedenstellend sind. Vor der nächsten Phase sind manuelle Ergänzungen wie weitere Suchstrings, !Bang, Blacklist und Whitelist möglich

Phase B (1): Automatische Triage & Sichtung (B__search_xng.py)

Die Suche erfolgt mit der Meta-Suchmaschine SearchXNG, die Suchergebnisse im JSON-Format ausliefert, das wiederum recht einfach zu bearbeiten ist. Das Programm wertet die fertige Prompt-Datei aus Schritt 1 aus und versucht, eine vorgegebene Anzahl von Webseiten zu analysieren. Dabei wird mit den reinen Suchstrings und in Kombination bang + Suchstring gearbeitet. Bevor die Volltext-Pipeline, also das LLM, gestartet wird, wird die Güte der Suchstrings überprüft:

- Blacklist-Seiten werden nicht ausgewertet
- Es wird versucht, eine Mindestanzahl von Whitelist-Seiten zu finden.
- Die Keywords müssen in den Snippets auftreten.
- Die Seiten dürfen nicht durch Captchas oder ähnliches blockiert sein.
- Die Seiten müssen eine Mindestlänge haben.

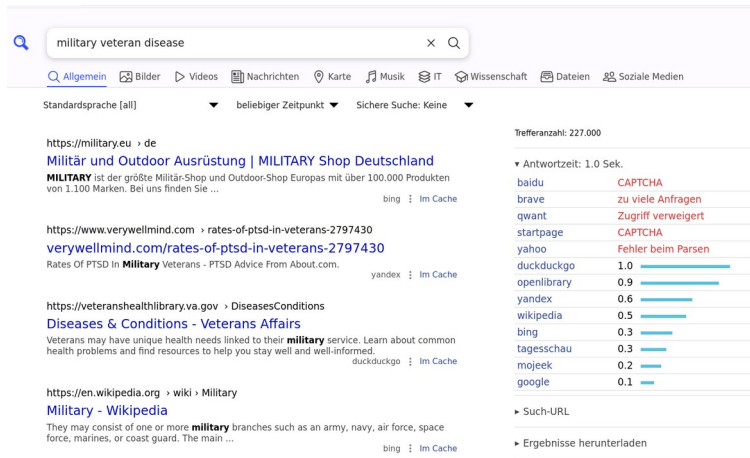
Ausgewertet werden reine HTML-Seiten und Video-Seiten, auf denen die Videos mit Untertiteln versehen sind (z.B. youtube). Gefundene PDF werden zur separaten Analyse herunter geladen.

Die schließlich ausgelesenen Daten werden an das LLM zur Auswertung übergeben. Erzeugt wird ein Dossier, das wiederum alle Daten enthält, um bei einer Sichtung nicht zwischen verschiedenen Dateien wechseln zu müssen.

Phase B (2): manuelle Suche & Sichtung (B__manual_search.py)

Die Suche über die Meta-Suchmaschine SearXNG hat das Handicap, dass zwar sehr viele Suchmaschinen gleichzeitig befragt und ein Konglomerat der Ergebnisse abgeliefert wird, aber viele Suchma-

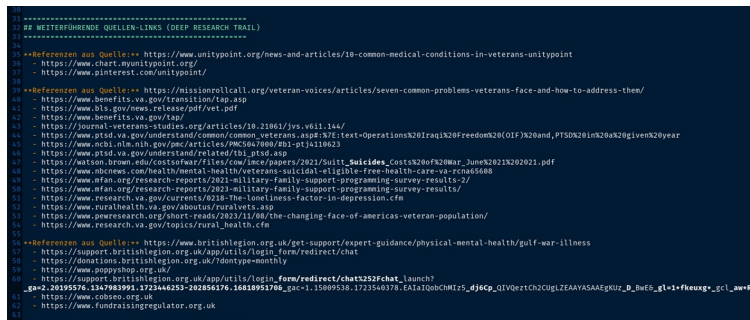
schinen bei automatischer Abfrage die Kooperation verweigern. Auch sind Quellen dabei, die man vielleicht gar nicht dabei haben möchte.



Die zweite Suchstrategie umgeht das Problem, in dem die Suche nun rein manuell mit nur einer Suchmaschine durchgeführt wird. Nach grober Bewertung, ob diese Ergebnisse brauchbar sind, wird die Seite als HTML-Datei auf dem Rechner gespeichert. Das ermöglicht auch die Auswertung von Links, die von den Suchmaschinen erst sehr weit hinten ausgegeben werden.

Anschließend wird das Python-Programm mit Angabe der auszuwertenden gespeicherten Seite aufgerufen. Das

Programm extrahiert die Links und leitet sie an das LLM weiter. Da die Snippet-Auswertung hier nicht eingesetzt werden kann, ist der „Beifang“ unbrauchbarer Seiten, die beim LLM landen etwas größer. Ansonsten entspricht das Verhalten dem von **Phase B (1)**.



Des weiteren liefert dieses Programm auch die Quell-Links, die in den ausgewerteten Seiten genannt werden, im Dossier aus. Dies ermöglicht eine Suche in die Tiefe. Man kopiert die interessanten Links in eine Datei, mit der man das Auswertungsprogramm erneut aufruft. Das gilt generell für alle Webseiten, die als HTML-Datei auf dem Rechner ge-

speichert werden: die enthaltenen Links werden ausgewertet.

Phase B (3): PDF - Auswertung (B__local_search.py, split_pdf.py)

Dieses Skript liefert eine vorgegebene Anzahl von PDF an das LLM aus. Als Vorbereitung werden große PDF in kleinere Chunks von 30 – 50 Seiten zerlegt, um das LLM nicht zu überlasten. Kleinere PDF werden einfach im Dateimanager durchnummeriert. Im Aufruf wird die Startnummer der nummerierten Dateien und deren Anzahl (Batch) aufgerufen. Das LLM erzeugt aus diesem Batch ein Dossier.

Phase C: Zusammenfassung (C__meta_analyse.py)

Je nach Größe des Projekts liegen am Schluss 10 – 100 Dossiers vor – in der Regel zu viel, um sie ohne Mühe auszuwerten.

[illegible]

In der Meta-Analyse werden die Dossiers vom LLM weiter auf die wesentlichen Aussagen verdichtet. Je nach Dichte kann man 10 – 30 Dossiers (muss experimentell ermittelt werden) weiter verdichten, ohne dass das LLM anfängt, wegen Überlastung zu halluzinieren.

4. Die Prompts

Dem LLM muss mitgeteilt werden, wie es die herunter geladenen Daten analysieren soll. Da die Vorgehensweise häufig bei verschiedenen Suchen gleich sein wird, werden die Prompts in Textdateien hinterlegt, die beim Skriptaufruf übergeben werden.

Der analytische Standard: Das Drei-Säulen-Raster

Wir erläutern den Standard am Beispiel einer Recherche im Umfeld des militärisch-zivilen Bereiches. Um konsistente und vergleichende Dossiers zu erzeugen, nutzt die primäre Analyse folgende Struktur:

- Säule A (Militärische & Offizielle Ebene): Dargestellte Einsatzlogik, Regierungs-Narrative, offizielle Dementis, PR-Statements.
- Säule B (Investigative & Lokale Ebene): Technische Details, Enthüllungen vor Ort, Fehlerquoten, Augenzeugenberichte, Nischen-Blogs.
- Säule C (Institutionelle & Völkerrechtliche Ebene): NGO-Berichte (Amnesty, Human Rights Watch), UN-Dokumente, juristische Einordnungen.
- Synthese-Auftrag am Ende jedes Dossiers:
- Welche Fakten sind durch mehrere unabhängige Quellen gesichert?
- Wo liegen direkte Widersprüche, Framing-Unterschiede oder blinde Flecken?

Basis-Prompt:

Du bist ein neutraler Recherche-Assistent. Analysiere die folgenden Quellentexte sachlich im Drei-Säulen-Raster. Fasse die Kernfakten zusammen und benenne etwaige Widersprüche, Framing-Unterschiede oder Lücken im Material.

„Drei-Säulen-Raster“ ist kein fest definierter Begriff, wird aber vom LLM in der Weise interpretiert, dass es die Informationen nach Hauptkategorien interpretieren soll. Die folgenden Anweisungen „Kernfakten, ...“ geben die Richtung vor, deren genaue Ausrichtung in diesem Prompt aber dem LLM überlassen bleiben.

Military-Prompt:

Du bist ein neutraler Recherche-Assistent. Analysiere die folgenden Quellentexte sachlich und ordne die Fakten strikt in das folgende Drei-Säulen-Raster ein:

1. Säule A (Militärische & Offizielle Ebene): Dargestellte Einsatzlogik, Zielsetzung und Rechtfertigung der Akteure.
2. Säule B (Investigative & Lokale Ebene): Technische Details, Enthüllungen, Fehlerquoten und konkrete Abläufe vor Ort.
3. Säule C (Institutionelle & Völkerrechtliche Ebene): Berichte von NGOs/UN, ethische Bewertungen und rechtliche Einordnungen.

Arbeite abschließend prägnant heraus:

- Welche Fakten durch mehrere Quellen gesichert sind.
- Wo direkte Widersprüche, Framing-Unterschiede oder blinde Flecken im Quellmaterial liegen.

Der Military-Prompt verfolgt die gleiche Analyserichtung, gibt dem LLM aber wesentlich schärfere Kriterien vor.

Je nach Ziel der Recherche sind Prompts und Suchanfragen anzupassen. Zeigt die Analyse, dass die Zielrichtung der Recherche gut getroffen ist, kann das LLM beauftragt werden, bestimmte Details in einem erneuten Durchlauf genauer heraus zu arbeiten. Stellt sich heraus, dass diese Details in der aktuellen Suche nicht zu finden sind, kann der Suchstring, ggf. unter Rückkehr zu Phase A, geschärft werden.

Im Prompt-Template ist zusätzlich vorgesehen, alle drei Stufen – init_query, search und meta_analyse – mit speziellen Prompts zu versorgen. Diese können den Haupt-Prompts hinzu gefügt oder völlig individuell gestaltet werden.

5. Bekannte Schwachstellen & Operative Begrenzungen

Knackpunkt der Recherchen sind die verwendeten Suchmaschinen. SearXNG ist eine Meta-Suchmaschine, die eine Vielzahl anderer Suchmaschinen (Liste in der Konfiguration) ansprechen kann. Optionen:

- (a) Wird die Suchanfrage nicht weiter spezifiziert, werden die in der Konfiguration aktivierten Suchmaschinen verwendet. Welche aktiviert werden, erfordert ein wenig Fingerspitzengefühl: je größer die aktive Liste, mit desto mehr Müll muss man in der Antwort rechnen.
- (b) Durch so genannte !bangs kann man die Suche im Einzelfall auf bestimmte Maschinen oder bestimmte Gruppen von Maschinen beschränken.

Dabei ergeben sich zwei Probleme:

- (1) Suchmaschinen erkennen meist, ob ein Mensch anfragt oder eine automatisiert Suche stattfindet. Viele Suchmaschinen blockieren bei automatischen Suchen. Aus diesem Grund sind Suchen nur mit Google oder DuckDuckGo vorgesehen.
- (2) Nicht alle Suchmaschinen verstehen die Syntax in einer Suchanfrage. Beispielsweise ist mit „Lavender“ im oben beschriebenen Beispiel eine militärisch AI gemeint. Obwohl die Suche durch „IDF“ und „AI“ eigentlich eindeutig gemacht wird, ist es durchaus möglich, dass fast sämtliche Links auf Seiten mit dem botanischen Begriff „Lavender“ verweisen. Das LLM merkt das zwar, aber die Recherche lässt sich nicht auflösen.

Wir haben dieses Problem durch 3 verschiedene Auswertemethoden im Bereich „B___“ berücksichtigt.

Problem	Ursache	Auswirkung	Gegenmaßnahme / Workaround
SearXNG CAPT-CHAs	Haupt-Engines (Google, DDG) blockieren automatisierte IP-Anfragen.	SearXNG weicht auf Bing/Wikipedia aus; Treffer-Qualität sinkt ("Pflanzen-Müll").	1. Vereinfachen der Syntax (keine Klammern). 2. Manuelle Linkliste aus Google/DDG direkt ins Skript speisen.
Snippet-Falle	Open WebUI wertet in der GUI oft nur die kurzen Textvorschauen aus.	Tiefere Details, Zahlen und Zitate fehlen im Erst-Check.	Für tiefgehende Dossiers immer Schritt C (search.sh) mit Volltext-Download nutzen.
Pflanzen-/Homonym-Problem	Eigennamen wie "Lavender" werden ohne Operatoren als Botanik interpretiert.	Unrelevante Treffer dominieren die Liste.	Strikte Anführungszeichen erzwingen: "Lavender" + "IDF", oder Bang ! news nutzen.

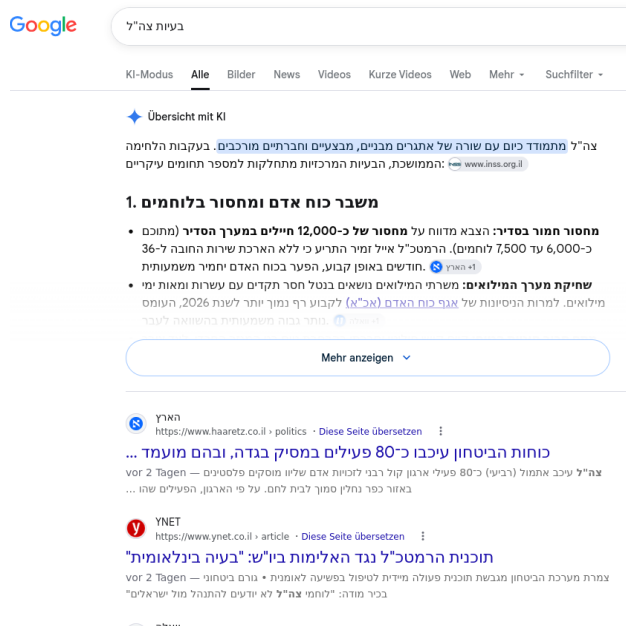
Suchstrings

Ein weiterer Knackpunkt besteht in der Erzeugung der richtigen Suchstrings. Wenn man beispielsweise den Stand des Ukraine-Krieges untersuchen will, ist es sinnlos, die Suche in Deutsch durchzuführen: die Information, dass die Ukraine 50.000 Mann seit 2022 verloren hat, die Russen aber 1,5 Mio, ist Standard im Qualitätsmedienwald, aber sicher nicht real.

Die Suchstrings sind daher in den passenden Sprachen zu formulieren. „Probleme der IDF“ findet deutsche Seiten, „Problems of the IDF“ sicher schon mehr internationale, “בעיות צה"ל” israelische und „مشاكل الجيش الإسرائيلي” den arabischen Counterpart.

Die kleinen LLM stoßen bei der Erzeugung von Suchstrings in Fremdsprachen aber schnell an ihre Grenze, obwohl sie die Sprachen grundsätzlich verstehen (sie sind auf Englisch optimiert; um den gleichen Sachverhalt in russisch darzustellen, werden 5- bis 8-fach so viele Token benötigt, was die Auswertungskapazität massiv beeinträchtigt). Es empfiehlt sich, die Suchen jeweils auf eine oder sehr wenige Sprachen zu begrenzen, was für die Auswertung u.U. sogar vorteilhaft ist.

Eine Option besteht auch darin, die Suchstrings von einem großen LLM im Internet manuell nachschärfen zu lassen. Häufig existieren in andern Sprachen sehr spezielle Fachbegriffe, ohne die die Suche fehlschlägt. Große LLM kennen diese Tücken und umgehen sie.



Bei der manuelle Suche, die man natürlich auch in der entsprechenden Sprache durchführen kann, hat man keine Chance, die Ergebnisse in irgendeiner Form manuell vorzufiltern. Man muss hier einfach sehr viel „Beifang“ in Kauf nehmen.

Nicht zufrieden?

Trotz viel Mühe und Abarbeiten von Jobs im Batchbetrieb während der Nacht kann das Ergebnis nicht zufriedenstellend ausfallen. Häufige Ursachen:

- Die Suchmaschinen liefern vorzugsweise HTML-Ergebnisse der wichtigsten Medien und verweisen spezielle Regierung Webseiten oder soziale Medien auf die hinteren Plätze oder unterdrücken sie ganz.
Möglich ist die gezielte Suche auf bestimmten Webseiten wie Regierungsservern und herunterladen passender PDF oder HTML-Seiten. Ein großes LLM kann hier in der Regel Kandidaten für beides liefern.
- Viele Seiten sind durch Captcha, Logins u.a. für unsere Suchmethoden nicht zugänglich.
Soziale Medien kann man händisch durchsuchen, was aber Mitgliedschaft voraussetzt und sehr mühsam ist. Soziale Medien mit eigener KI (z.B. Grok bei X) können zumindest globale Aussagen liefern.
- Die Suchstrategie ist falsch. Eine Meta-Meta-Analyse, d.h. ein Vergleich von dem, was man hineingesteckt hat mit dem, was herausgekommen ist, kann eine Anpassung ermöglichen. In der Meta-Analyse kann das LLM bereits Vorschläge erarbeiten.

Fazit

Das eine oder andere Problem ist hier sicher noch nicht angegangen. Vieles liegt an der (relativ) leichten Überlastbarkeit der kleinen LLM und dem kaum auszuschließenden Bias der großen LLM. Anders ausgedrückt: es ist derzeit nicht möglich, eine vollautomatisierte Analyse zu starten, an deren Ende wirklich unzweifelhafte Ergebnisse vorliegen. Eine Halbautomatik wie hier ist zwar mühsam, bringt aber doch schon erhebliche Verbesserungen gegenüber Analysen ohne KI-Beteiligung.