Fachhochschule Graubünden
University of Applied Sciences

PROJEKTBERICHT

James Bond Universum

Eine semantische Modellierung und Wissensextraktion rund um die ikonische Agentenfilmreihe.

Verfasserinnen: Tamara Nyffeler
Selina Steiner
Studiengang: MSc Data Visualization

Referent: Prof. Dr. habil. Albert Weichselbraun
Modul: Knowledge Engineering and Extraction
Institut: Institut für Informationswissenschaft (SII)

Bearbeitungszeitraum: 24. Oktober 2025 bis 19. Dezember 2025

Chur, 18. Dezember 2025

Abstract

Das vorliegende Projekt entwickelt einen domänenspezifischen Knowledge Graph für das James-Bond-Universum mit 25 Filmen und über 1'500 semantisch modellierten Entitäten. Durch die Integration strukturierter Daten aus Wikidata mit unstrukturierten Informationen aus Wiki-Fandom, Wikipedia und LLM-basierten Extraktionen entsteht eine umfassende Wissensbasis, die 435 Charaktere, 414 Schauspieler und Schauspielerinnen, 541 Schauplätze, 225 Fahrzeuge, 25 Titelsongs und Performer sowie biografische Informationen der sechs Bond-Darsteller umfasst.

Die methodische Grundlage des Projekts basiert auf den Konzepten und Techniken, die im ersten Teil des Moduls *Knowledge Engineering and Extraction* vermittelt wurden. Die Knowledge Extraction erfolgt mittels einer automatisierten Python-Pipeline, die API-Abfragen, SPARQL-Queries, Named Entity Recognition, Pattern Matching und Large Language Models kombiniert. Eine eigens entwickelte Ontologie basierend auf bestehendem Vokabular (LinkedMDB, DBpedia, Schema.org) formalisiert die Domäne und ermöglicht ein Reasoning zur Inferenz impliziter Beziehungen. Die erreichte Datenqualität liegt bei rund 95% Vollständigkeit.

Die Ergebnisse werden in einer öffentlich zugänglichen Streamlit-App visualisiert, die interaktive Exploration des RDF-Graphs, Analyse wiederkehrender Charaktere, visuelle Darstellung von spezifischen Charakteren wie Bond Girls und Antagonisten, sowie geografische Darstellung der Drehorte ermöglicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Codeverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CSV	Comma-separated values
ID	Identifikator oder Identität (englisch: in development)
JSON	JavaScript Object Notation
LinkedMDB	Linked Internet Movie Database
LLM	Large Language Model
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name

1 Einleitung

Die James-Bond-Filmreihe gehört zu den langlebigsten Erfolgsreihen der Kinogeschichte. Seit der Premiere von *Dr. No* im Jahr 1962 sind 25 offizielle Filme erschienen, die ein vielschichtiges Universum geschaffen haben. Dieses Universum zeichnet sich durch wiederkehrende Elemente aus: ikonische Charaktere wie Bond Girls und Antagonisten, weltweite Schauplätze, legendäre Fahrzeuge sowie prägnante Titelsongs. Über sechs Jahrzehnte hinweg haben unterschiedliche Darsteller die Hauptfigur James Bond verkörpert und dabei jeweils ihre eigene Interpretation des Charakters geprägt.

Die Fülle an Informationen rund um die James-Bond-Filme ist über zahlreiche Quellen verteilt. Von strukturierten Datenbanken wie Wikidata über Community-Plattformen wie Wiki-Fandom bis hin zu enzyklopädischen Einträgen in Wikipedia. In dem vorliegenden Projekt werden diese Informationsquellen zu einer Wissensdatenbank zentralisiert.

Der Aufbau erfolgt in zwei zentralen Schritten. Im Rahmen der Knowledge Extraction (Abschnitt 3) werden sowohl strukturierte als auch unstrukturierte Daten aus verschiedenen Quellen extrahiert und aufbereitet. Das anschliessende Knowledge Engineering (Abschnitt 4) widmet sich der formalen Modellierung des extrahierten Wissens. Es werden Klassen, Properties sowie deren semantische Relationen definiert und in einer Ontologie abgebildet. Die resultierenden RDF-Tripel bilden die Grundlage für den Knowledge Graph, dessen Konsistenz durch automatisiertes Reasoning überprüft wird.

Mittels einer interaktiven Streamlit-App werden die Ergebnisse schliesslich präsentiert (Abschnitt 5). Die App ist unter dem Link James Bond Universe öffentlich zugänglich und kann durch unterschiedliche Visualisierungen exploriert werden.

2 Ausgangslage

2.1 Kerndatensatz

Ausgangspunkt des Projekts bildet das *James Bond Movie Dataset* von Kaggle (Block, 2020). Der Datensatz umfasst 27 Merkmale für 24 James-Bond-Filme, darunter Produktionsinformationen, finanzielle Kennzahlen sowie inhaltliche Elemente wie Schauplätze, die Anzahl konsumierter Martinis oder die Häufigkeit der ikonischen Aussage “Bond, James Bond”.

Da der neueste Film *James Bond 007: No Time to Die (2020)* im Originaldatensatz fehlt und auch keine Angaben zu den Produzenten enthalten sind, werden im Rahmen der Datenaufbereitung folgende Schritte durchgeführt:

- Ergänzung der fehlenden Spalte *Producer* mittels Mapping,
- automatisches Hinzufügen des fehlenden Films *James Bond 007: No Time to Die (2020)*,
- Konstruktion eines reduzierten Kerndatensatzes, der aus sechs inhaltlich relevanten Attributen besteht, welche in Tabelle 1 detailliert erläutert werden.

Tabelle 1: Attribute des Kerndatensatzes (eigene Darstellung).

Spalte	Beschreibung
Year	Erscheinungsjahr des Films.
Movie	Offizieller Filmtitel.
Bond	Schauspieler, der James Bond verkörpert.
Director	Regisseur des Films.
Producer	Produzenten des Films; ergänzt über ein manuelles Mapping.
Avg_User_IMDB	Durchschnittliche Nutzerbewertung auf IMDB.
Avg_User_Rtn_Tom	Durchschnittliche Nutzerbewertung auf Rotten Tomatoes.

Zusätzlich wird für jeden Film die zugehörige Wikidata-ID mittels SPARQL-Query abgefragt und als neue Spalte ergänzt (Abschnitt 3.2, Code 5). Der so angereicherte Datensatz bildet die Grundlage für alle weiteren Schritte der Wissensextraktion und der semantischen Modellierung (Abbildung 1).

	Year	Movie	Bond	Director	Producer	Avg_User_IMDB	Avg_User_Rtn_Tom	wikidata_id
1	1962	Dr. No	Sean Connery	Terence Young	Harry Saltzman & Albert R. Broccoli	7.3	7.7	http://www.wikidata.org/entity/Q102754
2	1963	From Russia with Love	Sean Connery	Terence Young	Harry Saltzman & Albert R. Broccoli	7.5	8.0	http://www.wikidata.org/entity/Q106571
3	1964	Goldfinger	Sean Connery	Guy Hamilton	Harry Saltzman & Albert R. Broccoli	7.8	8.4	http://www.wikidata.org/entity/Q106440
4	1965	Thunderball	Sean Connery	Terence Young	Richard Maibaum & John Hopkins	7.0	6.8	http://www.wikidata.org/entity/Q107724
5	1967	You Only Live Twice	Sean Connery	Lewis Gilbert	Harry Saltzman & Albert R. Broccoli	6.9	6.3	http://www.wikidata.org/entity/Q107761
6	1969	On Her Majesty's Sec…	George Laze…	Peter R. Hunt	Harry Saltzman & Albert R. Broccoli	6.8	6.7	http://www.wikidata.org/entity/Q107894
7	1971	Diamonds Are Forever	Sean Connery	Guy Hamilton	Harry Saltzman & Albert R. Broccoli	6.7	6.3	http://www.wikidata.org/entity/Q107914
8	1973	Live and Let Die	Roger Moore	Guy Hamilton	Harry Saltzman & Albert R. Broccoli	6.8	5.9	http://www.wikidata.org/entity/Q27204

Abbildung 1: Auschnitt des Kerndatensatzes (jamesbond_with_id.csv).

2.2 Zielsetzung

Ziel des Projekts ist die semantische Modellierung und Wissensextraktion des James-Bond-Universums. Auf Basis des Kerndatensatzes sowie zusätzlicher strukturierter und unstrukturierter Datenquellen (Wikidata, Wikipedia, Wiki-Fandom, LLM) wird ein domänenspezifischer Knowledge Graph aufgebaut, der zentrale Entitäten wie Filme, Charaktere, Fahrzeuge, Orte oder musikalische Elemente miteinander verknüpft. Der Knowledge Graph soll folgende Fragestellungen beantworten können:

- **Beziehungsanalyse:** Wie stehen Bond, Antagonisten, Bond Girls, Schauplätze, Fahrzeuge oder Titelsongs miteinander in Beziehung und welche Zusammenhänge können sichtbar gemacht werden?
- **Wiederkehrende Elemente:** Welche Elemente und Charaktere sind über mehrere Filme hinweg wiederkehrend?
- **Entwicklung über die Zeit:** Wie haben sich bestimmte Rollen, geografische Schwerpunkte oder musikalische Themen über die Jahrzehnte hinweg entwickelt?

Das Ergebnis dieser Wissensextraktion wird in einer interaktiven Streamlit-App (James Bond Universe) zugänglich gemacht, welche die modellierten Informationen in verschiedenen Visualisierungen bündelt und so Film- und Charakterinformationen, den semantischen Knowledge Graphen, Bildergalerien sowie eine geografische Darstellung der Drehorte integriert. Damit ermöglicht die App eine intuitive und explorative Analyse des James-Bond-Universums und macht die durch die Ontologie geschaffenen semantischen Verknüpfungen für Nutzerinnen und Nutzer direkt zugänglich.

3 Knowledge Extraction

Die Extraktion von Wissen zur Anreicherung des Kerndatensatzes erfolgt auf zwei unterschiedlichen Wegen, abhängig davon, wie die Daten in ihrer Quelle vorliegen. Strukturierte Daten wie Tabellen erfordern andere Methoden als unstrukturierte Daten, wie beispielsweise Fliesstexte. Abbildung 2 zeigt eine Übersicht der verwendeten Datenquellen zur Wissensextraktion.

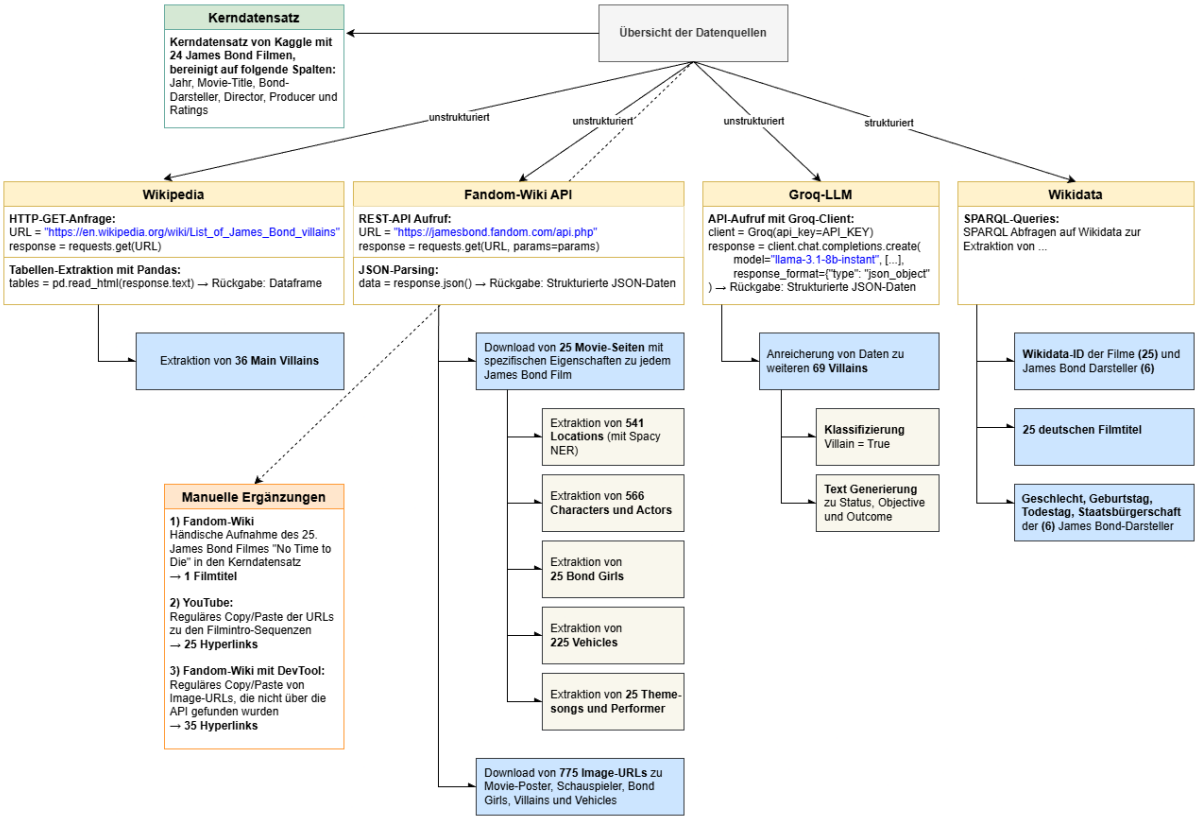


Abbildung 2: Übersicht der verwendeten Datenquellen (eigene Darstellung).

Dieses Kapitel beschreibt beide Extraktionsansätze entlang der Datenpipeline, die für den Knowledge-Extraction-Prozess in Python erstellt wurde (Abbildung 3). Dabei verweisen die Buchstaben auf die entsprechenden Skripte.

Data Pipeline

#	Script Name	Description
a	data_preparation.py	Cleans and enriches raw dataset
b	fandom_request_all_movies.py	Retrieves movie wiki pages from Fandom
c	fandom_request_movie_posters.py	Fetches movie poster URLs from Fandom
d	extract_locations_all_movies.py	Extracts and geocodes film locations using NER
e	extract_characters_all_movies.py	Extracts character and actor information
f	fandom_request_character_images.py	Retrieves character image URLs from Fandom API
g	character_image_url_completion.py	Manually corrects missing character image URL
h	fandom_request_bond_girls_with_images.py	Extracts Bond Girls data and images
i1	wikipedia_request_villains_with_images.py	Retrieves villain data from Wikipedia tables
i2	extract_additional_villains_with_LLM.py	Classifies additional villains using LLM and Groq API
i3	merge_all_villains.py	Merges Wikipedia and LLM-identified villains
i4	enrich_villains_data_with_LLM.py	Generates missing villain text fields using LLM and Groq API
j	extract_vehicles_all_movies.py	Extracts vehicle information from movie pages
k	fandom_request_vehicle_images.py	Retrieves vehicle image URLs from Fandom API
l	extract_songs_all_movies.py	Extracts song and performer information
m	extract_bond_wikidata_id_sparql.py	Queries Wikidata for Bond actor IDs
n	extract_bond_info_sparql.py	Extracts detailed Bond actor info via SPARQL
o	extract_movie_title_german_sparql.py	Fetches German movie titles from Wikidata
p	merge_all_data_to_json.py	Merges all CSV/JSON data into a single JSON file
q	merge_json_to_knowledge_graph.py	Converts JSON to RDF knowledge graph
r	run_reasoner.py	Runs OWL reasoner for inferred properties

Caption:
- Structured Data Extraction
- Unstructured Data Extraction
- Data Fusion and Reasoning

Abbildung 3: Tabellarische Auflistung der Datenpipeline zur Knowledge Extraction (eigene Darstellung).

3.1 Unstrukturierte Daten

Fliesstexte und freie Textpassagen in Filmbeschreibungen sind typische Beispiele für unstrukturierte Datenquellen. Sie folgen keiner einheitlichen Struktur und lassen sich nicht direkt in relationalen Datenbanken abbilden. Die Informationen sind in natürlicher Sprache eingebettet und erfordern komplexere Verarbeitungstechniken wie Natural Language Processing (NLP) oder Pattern Matching, um relevante Entitäten, Beziehungen

und Fakten zu identifizieren. Als primäre Datenquelle wird in diesem Projekt das James Bond Wiki-Fandom verwendet und mit Wissen aus Wikipedia und dem Groq-Client ergänzt.

3.1.1 Wiki-Fandom API

Für die Datenextraktion aus dem James Bond Fandom Wiki wird die offizielle API verwendet. Die Schnittstelle liefert filmspezifische Informationen, darunter Charaktere und Schauspieler:innen, Schauplätze, Fahrzeuge, Songs und Bilder („James Bond Wiki“, n. d.).

Abruf der 25 Filmseiten als zentrale Datenbasis

Die Grundlage der gesamten Datenextraktions-Pipeline bildet das `Skript b`, das die vollständigen Seiten aller James Bond Filme von der Fandom-Seite abruft. Das Skript durchläuft eine vordefinierte Liste von Bond-Filmen und fordert für jeden Film den vollständigen Wikitext-Inhalt über die API an. Die Schnittstelle liefert den rohen Text, der neben dem sichtbaren Inhalt auch strukturelle Elemente wie Abschnittsüberschriften und Formatierungsanweisungen enthält. Mit der Bibliothek `wikitextparser` kann der Wikitext in zwei Hauptkategorien gegliedert werden: Seitenabschnitte (engl. Sections) (Abbildung 4) und Infoboxen (Abbildung 5). Für jede Filmextraktion wird eine JSON-Datei gespeichert.

James_Bond_Universe > extract_knowledge > fandom_wiki_pages > {} Diamonds_Are_Forever_film.json > { } sections

```
{
  "title": "Diamonds Are Forever (film)",
  "sections": {
    "intro": "{{Cinematic}}\n{{Diamonds Are Forever}}\n{{Infobox film\n| image = 0
    "Synopsis": "After having seemingly killed his arch-nemesis [[Blofeld (classic
    "Plot summary": "{{Cleanup}}\n===Hunt for Blofeld===\n[[File:Creating Blofeld'
    "Hunt for Blofeld": "[[File:Creating Blofeld's duplicates (Diamonds are Foreve
    "Search for the diamond smugglers": "[[File:Diamonds are Forever - Wint and Ki
    "Arrival to Las Vegas": "[[File:MrWintAndMrKiddBurningTheCoffin.jpeg|thumb|rig
    "W Tectronics": "[[File:Bondwiki DAF71 Bond Metz Lab1.jpg|thumb|left|250px|Bor
    "Las Vegas Strip and Whyte House": "He and Tiffany return to Las Vegas, and af
    "Rescue of Whyte": "[[File:Bondwiki Bond attacked by Bambi and Thumper.jpg|thu
    "Blofeld's Oil Rig": "[[File:BlofeldExplainingHisPlansToJBond-DAF.jpeg|thumb|r
    "Epilogue": "[[File:Bondwiki DAF71 Bond Tiffany Wint Kidd Ship.jpg|thumb|left|
    "Cast & characters": "<gallery orientation=\"square\" spacing=\"small\" widths
    "Crew": "*Directed by: [[Guy Hamilton]]\n*Written by: [[Ian Fleming]]\n*Screer
```

Abbildung 4: Auszug der geparsten Seitenabschnitte der Filmseite *Diamons Are Forever* (Diamonds_Are_Forever_film.json).

```
43      "infobox": {
45        "caption": "'''Diamonds Are Forever'' theatrical poster",
46        "bond": "[[Sean Connery]]",
47        "writer": "[[Ian Fleming]] (novel)",
48        "screenplay": "[[Richard Maibaum]],<br>[[Tom Mankiewicz]]",
49        "director": "[[Guy Hamilton]]",
50        "producer": "[[Harry Saltzman]], [[Albert R. Broccoli]]",
51        "editing": "Bert Bates, John Holmes",
52        "music": "[[John Barry]]",
53        "song": "\"[[Diamonds Are Forever (song)|Diamonds Are Forever]]\"",
54        "distributor": "[[United Artists]]",
55        "released": "[[007_Timeline#1971|14 Dec 1971]] <small>(West Germany, premiere)
```


Abbildung 5: Auszug der geparsten Infoboxen der Filmseite *Diamons Are Forever* (Diamonds_Are_Forever_film.json).

Abruf von Bild-URLs

Während die primären Filmdaten umfangreiche Textinformationen enthalten, erfordern visuelle Elemente wie Filmposter und Bilder einen separaten Extraktionsprozess. Die `Skripte c, f, h und k` verwenden die `pageimages`-Eigenschaft mit dem Parameter `original`, um die Originalversion der Bilder abzurufen. Für die abgefragten Entitäten liefert die API eine URL zur Originaldatei auf Fandom, die in einem CSV gespeichert wird. Dies bildet die Basis für die spätere Zuordnung im Knowledge Graph.

Extraktion von Orten

Mit `Skript d` erfolgt die Extraktion von Schauplätzen in drei Schritten:

1. Die JSON-Datein der Filmseiten werden geladen und die Sections *Locations, Film locations* und *Shooting locations* isoliert. Die NLP-Bibliothek `SpaCy (Modell en_core_web_lg)` analysiert den Text mittels `Named Entity Recognition` und identifiziert geografische Entitäten wie Länder, Städte und Sehenswürdigkeiten.

2. Anschliessend werden die extrahierten Ortsnamen über den `Nominatim`-Geocoder, der OpenStreetMap-Daten nutzt, verarbeitet. Dabei werden Ortsnamen in geografische Koordinaten (Breiten- und Längengrad) umgewandelt.

3. Im letzten Schritt erfolgt die Datenbereinigung: Einträge ohne erfolgreiche Geocodierung werden entfernt und Duplikate eliminiert. Das Ergebnis ist eine CSV-Datei mit geocodierten Schauplätzen pro Film.

Extraktion von Charakteren und Schauspieler:innen

`Skript e` extrahiert Charakternamen und zugehörige Darsteller:innen aus den *Cast*

and characters-Abschnitten der Film-JSON-Dateien. Der Text dieser Sections wird zeilenweise verarbeitet und Charakter-Schauspieler:in-Paare mittels regulärer Ausdrücke aus dem Wiki-Markup identifiziert. Jeder Eintrag wird mit dem jeweiligen Film verknüpft, was später die Analyse wiederkehrender Figuren ermöglicht. Das Ergebnis ist eine CSV-Datei mit allen Charakteren der gesamten Filmreihe.

Diese Daten werden anschliessend durch `Skript f` mit Bild-URLs aus der API angereichert. Wo die API keine URL liefert, erfolgt mit `Skript g` eine manuelle Ergänzung der fehlenden Einträge (begrenzt auf 35 Auffälligkeiten). Hierbei werden die URLs mithilfe des Browser-Entwicklertools direkt von den Fandom-Seiten extrahiert. Die Charakterdatenbank bildet die zentrale Grundlage für die nachfolgende Analyse der Subentitäten Bond Girls und *Villains*.

Extraktion von Bond Girls

`Skript h` extrahiert Informationen zu den Bond Girls von einer eigenen Fandom-Seite, die alle Bond Girls der Filmreihe in Tabellenform auflistet. Im Gegensatz zur filmweisen Extraktion (wie bei Charakteren und Ortschaften) bietet diese zentrale Seite zwei wesentliche Vorteile: Die Fandom-Community hat bereits eine Definition vorgenommen, welche Charaktere als Bond Girls gelten und zwischen Haupt- und Nebenfiguren differenziert. Mittels der `wikitextparser`-Bibliothek wird die Tabelle *Eon series James Bond girls* ausgelesen und auf Haupt-Bond-Girls gefiltert.

Da die Tabelle keine Bilder enthält, müssen die URLs für alle 25 Bond Girls schrittweise ermittelt werden: zuerst über filmspezifische, dann über schauspielerspezifische und schliesslich über generische Fandom-Seiten. Noch fehlende Bilder werden aus der CSV-Charakterdatenbank ergänzt.

Extraktion von Fahrzeugen

`Skript j` extrahiert Fahrzeuginformationen aus dem *Major vehicles*-Abschnitt der Film-JSON-Dateien. Diese Section umfasst Fahrzeuge in einem Galerie-Format ähnlich den Charakteren, wobei jeder Eintrag den Fahrzeugnamen, einen Bilddateinamen und eine Sequenzbeschreibung enthält.

Mit `Skript k` erfolgt der API-Aufruf für die Bild-URL. Hierbei wird ein zweistufiger Ansatz verfolgt. Zunächst wird der extrahierte Bilddateiname direkt über die API abgefragt (Property `imageinfo`), um die tatsächliche URL zu erhalten. Die URL wird anschliessend bereinigt (Entfernung von Revisions-Parametern und Query-Strings). Schlägt diese Methode fehl, etwa durch fehlerhafte Dateinamen oder nicht existierende Dateien, greift eine Fallback-Strategie. Das Skript sucht nach der Fahrzeugseite selbst und ruft

das zugehörige Hauptbild ab. Dieser Ansatz sorgt dafür, dass möglichst viele URLs gefunden werden. Als Ergebnis wird eine CSV-Datei gespeichert, die die Zuordnung zu den Filmen beinhaltet.

Extraktion von Titelsongs

`Skript l` extrahiert Informationen zu den Titelsongs aus den Infoboxen der Film-JSON-Dateien. Infoboxen weisen eine *Key-Value* Datenstruktur auf, die typischerweise Felder für Titelsong, Interpret und Komponist enthält. Mittels regulärer Ausdrücke wird der tatsächliche Textinhalt extrahiert und Markup-Syntax, Anführungszeichen sowie HTML-Tags entfernt. Ergänzend werden die YouTube-Links zu den Opening-Sequenzen aller 25 Filme manuell hinzugefügt. Das Ergebnis ist eine CSV-Datei mit den Spalten Movie, Song, Performer und Composer sowie YouTube-Link.

3.1.2 Wikipedia

Obwohl Wiki-Fandom umfangreiche Informationen zum James Bond Universum liefert, bietet Wikipedia für bestimmte Entitäten noch gezieltere Daten. Ein solches Beispiel sind die Villains. Wikipedia verfügt über eine strukturierte Tabelle der Haupt-Antagonisten der James Bond Filmreihe. Mit dem `Skript i_1` wird eine GET-Anfrage an den Endpoint `https://en.wikipedia.org/wiki/List_of_James_Bond_villains` gestellt und die Tabelle *#Eon_Productions* mittels der Pandas-Funktion `read_html` direkt aus HTML geparst. Neben Identifikationsdaten wie Film, Villain, Portrayed by bietet die Tabelle auch narrative Kontextinformationen:

- **Objective:** Beschreibt den Plan des Gegners.
- **Outcome:** Erklärt, wie der Plan durchkreuzt wird.
- **Status:** Hält das Schicksal des Gegners fest.

Die Bild-URLs werden wie bei den anderen Entitäten auch über die Fandom API extrahiert. Im Post-Processing erfolgen manuelle Korrekturen bekannter Inkonsistenzen und Ergänzungen durch Querverweise mit der CSV-Charakterdatenbank.

3.1.3 Groq-Client und LLM

Während Wikipedia die Haupt-Antagonisten mit 36 Treffern gut abdeckt, fehlen sekundäre Bösewichte und weniger bekannte Handlanger. Das Skript i_2 nutzt die Groq API mit einem Large Language Modell zur automatisierten Klassifikation zusätzlicher Villains aus der CSV-Charakterdatenbank („GroqCloud - Build Fast“, n. d.). Es wird das Modell `llama-3.1-8b-instant` verwendet.

Das Skript lädt alle extrahierten Charaktere und gleicht sie mit der bestehenden Wikipedia-Villain-Liste ab. Für jeden noch nicht klassifizierten Charakter wird ein User-Prompt mit Name, Schauspieler:in, Film und Bild-URL durchlaufen (Code 1). Ein System-Prompt definiert im Hintergrund, was einen relevanten Bösewicht ausmacht (Code 2):

```
user_prompt = f"""
Character: {row['character']}
Actor: {row['actor']}
Movie: {row['movie']}
search_title: {row['search_title']}
image_url: {row['image_url']}
"""
```

Code 1: User-Prompt zur Klassifizierung weiterer Villains mittels LLM (i_2_extract_additional_villains_with_LLM.py).

Die identifizierten Villains werden im Wikipedia-Schema gespeichert (Film, Villain, Portrayed by) und mit Skript i_3 mit den Haupt-Antagonisten vereint. Mit dieser Methode können weitere 69 Villains klassifiziert werden.

Die LLM-Funktionalität wird in Skript i_4 erneut eingesetzt, um die narrativen Felder (Objective, Outcome, Status) mit generiertem Text anzureichern (Code 3 und Code 4).

```
        user_prompt = f"""
Film: {film}
Villain: {villain_name}
Portrayed by: {actor}

Please provide the Objective, Outcome, and Status for this villain.
"""
```

Code 3: User-Prompt zur Textgenerierung der Merkmale Objective, Outcome und Status (i_4_enrich_villains_data_with_LLM.py).

```
SYSTEM_PROMPT = """
You are an expert on James Bond movies.
Given the following character information:

- character name
- actor
- film
- search_title

Determine whether the character is a SIGNIFICANT and ICONIC VILLAIN who plays a major antagonistic role.

A James Bond villain worthy of inclusion is defined as:
- Main antagonist / mastermind of the film (the primary villain)
- Major henchmen who are memorable, have significant screen time, and directly engage with Bond in combat or confrontation
- Only include secondary villains if they are well-known and iconic characters from the franchise

DO NOT classify as villains:
- Minor characters with brief appearances
- Background henchmen without names or speaking roles
- Allies or neutral characters
- Characters who merely work for villains but don't actively oppose Bond
- Generic guards, soldiers, or unnamed thugs
- Bond Girls or love interests (even if they betray Bond)
- Supporting accomplices without significant antagonistic roles
- Informants, contacts, or minor associates of the main villain
- Characters who appear in only one or two scenes

IMPORTANT: Be EXTREMELY SELECTIVE. Only include villains that are memorable and iconic.
When in doubt, mark as NOT a villain (is_villain = false).

Return ONLY a JSON object with the structure:
{
  "is_villain": true/false,
  "film": "...",
  "villain": "...",
  "portrayed_by": "...",
  "image_url": "..."
}

Rules:
- If the character is not a significant villain → is_villain = false and all other fields empty strings.
- Be VERY SELECTIVE - only mark truly important and iconic villains as is_villain = true.
- Do NOT include explanations.
"""
```

Code 2: System-Prompt zur Klassifizierung weiterer Villains mittels LLM (i_2_extract_additional_villains_with_LLM.py).

```
SYSTEM_PROMPT = """
You are an expert on James Bond movies with detailed knowledge of all villains and their storylines.

Given a villain's information (Film, Villain name, and Portrayed by), provide:
1. **Objective**: A concise description of the villain's main goal or evil plan in the movie (1-2 sentences)
2. **Outcome**: What happened to their plan - was it foiled by Bond? How? (1-2 sentences)
3. **Status**: The villain's fate at the end of the movie (1 sentence, e.g., "Killed by Bond", "Arrested", "Survives", etc.)

Be specific and accurate to the James Bond film and use clear, concise language.

Return ONLY a JSON object with this exact structure:
{
  "objective": "...",
  "outcome": "...",
  "status": "..."
}

Do NOT include any explanations or additional text outside the JSON object.
"""
```

Code 4: System-Prompt zur Textgenerierung der Merkmale Objective, Outcome und Status (i_4_enrich_villains_data_with_LLM.py).

3.2 Strukturierte Daten

Neben unstrukturierten Daten werden auch strukturierte Daten zur Wissensanreicherung des James-Bond-Universums extrahiert. Diese Daten liegen in formal strukturierten Formaten wie Datenbanken und Tabellen vor. Deren Datenmodelle und Schemata sind in den jeweiligen Quellen vordefiniert und in Datenbanksystemen organisiert, in denen Tabellen mit eindeutig spezifizierten Attributen abgelegt sind. Strukturierte Daten aus Wissensdatenbanken wie Wikidata und DBpedia können überwiegend über deren öffentliche Endpunkte mittels SPARQL-Abfragen extrahiert werden. Das zugrunde liegende Datenmodell basiert auf Elementen, Eigenschaften und Aussagen und ist damit als Tripel-Struktur (Subjekt–Prädikat–Objekt) organisiert. Dies ermöglicht präzise, selektive Abfragen über die bereitgestellten Services und Endpunkte.

3.2.1 Wikidata

Im Zuge der Themenfindung wurden ausgewählte Wissensdatenbanken, insbesondere Wikidata und DBpedia, analysiert und hinsichtlich ihrer inhaltlichen Reichhaltigkeit evaluiert. Dabei zeigte sich, dass Wikidata einen grösseren Umfang an relevanten Informationen bereitstellt und somit eine geeignetere Quelle für die Extraktion von Daten zu Filmen und Schauspieler:innen darstellt. Aus diesem Grund wird die weitere Datenextraktion in dieser Arbeit auf Wikidata fokussiert.

Extraktion der Wikidata-IDs aller Filme
Die initiale Extraktion erfolgt im Rahmen der Aufbereitung des Kerndatensatzes mit `Skript a`, wie in Abschnitt 2.1 erwähnt. Für jeden Film wird dabei die zugehörige Wikidata-ID mittels einer SPARQL-Query über den Wikidata-Endpunkt ermittelt. Dieses Vorgehen erleichtert nachfolgende Extraktionen aus Wikidata erheblich, da anstelle eines potenziell fehleranfälligen und rechenintensiven "String-Matching" der Filmtitel lediglich ein Abgleich der bereits ermittelten eindeutigen Identifikatoren erforderlich ist, was zu einer deutlich höheren Effizienz und Kosteneffektivität führt.

Hierzu werden sämtliche Entitäten abgefragt, die als Teil ("part of the series", `Property:P179`) der James-Bond-Filmreihe (`Q2484680`) in Wikidata verzeichnet sind; optional werden zusätzlich deren englische Labels ausgelesen. Um orthografische Varianten zwischen britischem und amerikanischem Englisch wie beispielsweise "License" vs. "Licence" im Filmtitel *Licence to Kill* zu berücksichtigen und dadurch das Übersehen potenzieller Treffer zu vermeiden, werden neben dem offiziellen Label auch die alternativen Bezeichnungen aus dem Wikidata-Feld *Also known as* abgefragt. Die im Datensatz enthaltenen englischen Filmtitel werden anschliessend mit den abgefragten Filmen anhand zweier Matching-Varianten abgeglichen und daraufhin gefiltert: (1) ein casesensitives exaktes Matching sowie (2) ein Matching mittels `CONTAINS`, um längere Bezeichnungen in Wikidata mit einzubeziehen. Abschliessend wird das Ergebnislimit auf 1 gesetzt, um etwaige Duplikate in der Treffermenge zu verhindern.

Die eingesetzte SPARQL-Abfrage wird exemplarisch in Code 5 illustriert, indem gezielt ein einzelner Film (*Casino Royale*) selektiert wird („Wikidata", n. d.).

```
PREFIX wd:   <http://www.wikidata.org/entity/>
PREFIX wdt:  <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?film
WHERE {
  ?film wdt:P179 wd:Q2484680 .

  OPTIONAL { ?film rdfs:label ?label . FILTER(LANG(?label) = "en")}
  OPTIONAL { ?film skos:altLabel ?altLabel . FILTER(LANG(?altLabel) = "en")}

  FILTER( # fix movie = Casino Royale
    LCASE(STR(?label)) = LCASE("Casino Royale") ||
    LCASE(STR(?altLabel)) = LCASE("Casino Royale") ||
    CONTAINS(LCASE(?label), LCASE("Casino Royale")) ||
    CONTAINS(LCASE(?altLabel), LCASE("Casino Royale"))
  )
}
LIMIT 1
```

Code 5: SPARQL Abfrage auf Wikidata zur Extraktion der Wikidata-ID des Films *Casino Royale* (extract_movie_id.ttl).

Abbildung 6 zeigt exemplarisch die Label-Struktur eines James-Bond-Films in Wikidata. Neben dem offiziellen Titel (`rdfs:label`) sind zusätzliche alternative Bezeichnungen im Feld *Also known as* hinterlegt, welche in Wikidata als `skos:altLabel` modelliert sind. Diese alternativen Labels umfassen unter anderem orthografische Varianten sowie abweichende Schreibweisen und sind entscheidend, um eine robuste Identifikation der Filme zu gewährleisten. Die Abbildung verdeutlicht zudem die Mehrsprachigkeit der Filmtitel in Wikidata, welche in späteren Extraktionsschritten genutzt wird, um deutsche Titel gezielt über `FILTER(LANG(...)= "de")` zu selektieren.

Language	Label	Description	Also known as
default for all languages	No label defined	–	
English	Licence to Kill	1989 film by John Glen	License to Kill
German	James Bond 007 – Lizenz zum Töten	Film von John Glen (1989)	Lizenz zum Töten Licence to Kill License to Kill James Bond 007 - Lizenz zum T…

Abbildung 6: Darstellung der Label-Struktur des James-Bond-Films *Licence to Kill* in Wikidata.

Extraktion der Wikidata-IDs aller Bond-Schauspieler

Zusätzlich zur Extraktion der Wikidata-IDs aller Filme werden die Wikidata-IDs

sämtlicher Schauspieler erhoben, welche die Hauptfigur James Bond verkörpert haben. Dies ermöglicht ebenfalls nachfolgende Extraktionen aus Wikidata auf Basis eindeutiger Identifikatoren anstelle eines fehleranfälligeren und rechenaufwendigen "String-Matchings".

In `Skript m` werden mittels einer SPARQL-Abfrage alle Entitäten selektiert, die gemäss der Eigenschaft *occupation* (`Property:P106`) entweder als *television actor* (`Q10798782`) oder als *film actor* (`Q10800557`) klassifiziert sind. Analog zur Ermittlung der Wikidata-IDs der Filme werden auch in diesem Schritt optional das offizielle sowie ein alternatives englisches Label abgerufen. Anschliessend erfolgt ein case-sensitives, exaktes String-Matching zum Abgleich und zur Filterung der Labels. Zur Vermeidung von Duplikaten wird jeweils nur der erste Treffer zurückgegeben. Das resultierende Datenobjekt wird in einer separaten CSV-Datei abgespeichert.

Mit Code 6 wird die zugrunde liegende SPARQL-Abfrage exemplarisch für den Schauspieler *Sean Connery* dargestellt.

```
PREFIX wd:   <http://www.wikidata.org/entity/>
PREFIX wdt:  <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?actor
WHERE {
    { ?actor wdt:P106 wd:Q10798782 . }  # television actor
    UNION
    { ?actor wdt:P106 wd:Q10800557 . }  # film actor

  OPTIONAL { ?actor rdfs:label ?label . FILTER(LANG(?label) = "en")}
  OPTIONAL { ?actor skos:altLabel ?altLabel . FILTER(LANG(?altLabel) = "en")}

  FILTER( # fix actor = Sean Connery
    LCASE(STR(?label)) = LCASE("Sean Connery") ||
    LCASE(STR(?altLabel)) = LCASE("Sean Connery")
  )
}
LIMIT 1
```

Code 6: SPARQL Abfrage auf Wikidata zur Extraktion der Wikidata-ID James Bond Darsteller *Sean Connery* (extract_bond_id.ttl).

Extraktion der biografischen Attribute der Bond-Darsteller

Aufbauend auf den zuvor ermittelten Wikidata-IDs der James-Bond-Darsteller werden in einem weiteren Schritt detaillierte personenbezogene Informationen aus Wikidata

extrahiert. Ziel dieser Extraktion ist die Anreicherung der Darsteller-Entitäten mit grundlegenden biografischen Attributen, um diese anschliessend strukturiert im Knowledge Graph weiterverwenden zu können.

In `Skript n` erfolgt die Abfrage der Informationen auf Basis der eindeutig bekannten Wikidata-ID. Für jede Darsteller-Entität werden anschliessend mehrere biografische Attribute extrahiert: das Label (`rdfs:label`), das Geschlecht über die Eigenschaft *sex or gender* (`Property:P21`), die Staatsbürgerschaft mittels *country of citizenship* (`Property:P27`) sowie das Geburts- (`Property:P569`) und – sofern vorhanden – das Todesdatum (`Property:P570`).

Zur Sicherstellung einer konsistenten sprachlichen Darstellung werden für alle textuellen Attribute ausschliesslich englischsprachige Labels berücksichtigt. Die Verwendung von `OPTIONAL`-Blöcken gewährleistet eine robuste Abfrage auch bei unvollständigen Datensätzen. Die resultierenden Daten werden abschliessend in einer separaten JSON-Datei gespeichert und für die weitere Integration in den Knowledge Graph verwendet.

Mit Code 7 wird die zugrunde liegende SPARQL-Abfrage exemplarisch für den James-Bond-Darsteller *Sean Connery* mit der Wikidata-ID `Q4573` dargestellt.

```
PREFIX wd:   <http://www.wikidata.org/entity/>
PREFIX wdt:  <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?actor ?actor_label ?gender_label ?country_label ?dob ?dod
WHERE { # fix actor = wd:Q4573 (Sean Connery)
  BIND(wd:Q4573 AS ?actor)

  OPTIONAL { ?actor rdfs:label ?actor_label .
             FILTER(LANG(?actor_label) = "en")}

  OPTIONAL { ?actor wdt:P21 ?gender . ?gender rdfs:label ?gender_label .
             FILTER(LANG(?gender_label) = "en")}

  OPTIONAL { ?actor wdt:P27 ?country . ?country rdfs:label ?country_label .
             FILTER(LANG(?country_label) = "en")}

  OPTIONAL { ?actor wdt:P569 ?dob . }
  OPTIONAL { ?actor wdt:P570 ?dod . }
}
ORDER BY ?actor_label ?country_label
```

Code 7: SPARQL Abfrage auf Wikidata zur Extraktion der Informationen (Geschlecht, Geburtstag, Todestag, Staatsbürgerschaft) zum James Bond-Darsteller *Sean Connery* mit Wikidata-ID `Q4573` (extract_bond_info.ttl).

Extraktion der deutschen Filmtitel

Für eine konsistente Darstellung in der Web-Applikation werden in einem abschliessenden Extraktionsschritt die deutschen Filmtitel der 25 James-Bond-Filme aus Wikidata abgerufen. Da die eindeutigen Wikidata-IDs der Filme bereits vorliegen, kann die Abfrage direkt auf den jeweiligen Entitäten ausgeführt werden, ohne zusätzliche String-Matching-Logik zur Identifikation der Filme.

Mit einer SPARQL-Abfrage in `Skript o` werden die relevanten Film-Entitäten über `VALUES` eingebunden. Anschliessend wird das deutschsprachige Label über `rdfs:label` selektiert und mittels `FILTER(LANG(...)= "de")` auf deutsche Titel eingeschränkt. Die extrahierten deutschen Filmtitel werden anschliessend mit den bereits vorhandenen englischen Originaltiteln gematcht und gemeinsam in einer separaten CSV-Datei gespeichert.

Code 8 zeigt die Abfrage exemplarisch für den Film *Dr. No* mit der Wikidata-ID *Q102754*.

```
PREFIX wd:   <http://www.wikidata.org/entity/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?item ?title_de
WHERE { # fix film = wd:Q102754 (Dr. No)
  VALUES ?item { wd:Q102754 }

  ?item rdfs:label ?title_de . FILTER(LANG(?title_de) = "de")
}
```

Code 8: SPARQL Abfrage auf Wikidata zur Extraktion des deutschen Filmtitel zu *Dr. No* mit Wikidata-ID *Q102754* (extract_movie_title_german.ttl).

3.3 Datenqualität

Die Datenpipeline erzielt eine hohe Datenqualität mit nahezu vollständiger Abdeckung (>95% overall). Tabelle 2 liefert eine Übersicht über die erreichten Extraktionen. Zur Datenqualität lassen sich folgende zentrale Erkenntnisse festhalten:

- **Vehicles:** 7 von 25 Filmtitel liefern keine Fahrzeugdaten. Grund dafür ist, dass der Abschnitt *Major Vehicles* bei diesen Film-Seiten nicht existiert: *Live and Let Die, GoldenEye, Casino Royale, Quantum of Solace, Skyfall, Spectre, No Time to Die*.
- **Locations:** 3 Ortschaften-Film Beziehungen treten doppelt auf:

  - Netherlands in *Diamonds Are Forever*.
  - New Orleans in *Live and Let Die*.
  - Germany in *Tomorrow Never Dies*.

  Weiter kommen auch ähnliche Schauplätze vor, wie beispielsweise *San Francisco* und *San Francisco City*. Zudem liefert 1 von 25 Filmtitel gar keine Schauplätze (*No Time to Die*). Grund dafür ist, dass der Abschnitt *Locations* für diesen Filmtitel nicht existiert.

- **Filmintro Sequenzen:** Die URLs zu den YouTube Filmsequenzen werden über ein manuelles Mapping im Code integriert und nicht über einen Automatismus extrahiert.
- **Bild-URLs:** Zur Vervollständigung werden rund 6% der Charaktere-Bilder über ein manuelles Mapping im Code ergänzt.
- **Schauspieler:innen:** Rund 6% der Charaktere haben keine Information zu Darsteller:innen.

Tabelle 2: Übersicht der erzielten Datenqualität im Knowledge-Extraction-Prozess (eigene Darstellung).

Entität	Aspekte	Anzahl Treffer	
Filmtitel *Score: 100%*	Filmtitel im Kerndatensatz	+24	
	Filmtitel manuell ergänzt	1	∼1%
	Wikidata IDs	25	100%
	Filmtitel in Deutsch	25	
Movie Poster *Score: 100%*	Filme mit Poster	25	100%
Charaktere *Score: 94%*	Charaktere-Film Beziehungen	566	100%
	Charaktere ohne Schauspieler:in	33	∼6%
	Einzigartige Charaktere	435	
	Einzigartige Schauspieler:innen	414	
Charaktere-Bilder *Score: 90%*	Charakter mit Bild	511	
	Charakter ohne Bild	55	∼10%
	Charakter Bild manuell ergänzt	35	∼6%
Locations *Score: 99%*	Geocoded Locations	541	100%
	Duplikate	3	$<$ 1%
Bond Girls *Score: 88%*	Haupt Bond Girls	25	100%
	Bond Girl Schauspielerin	22	
	Bond Girl Schauspielerin manuell ergänzt	3	∼12%
Villains *Score: 100%*	Gesamt Anzahl Villains	105	100%
	Haupt Antagonisten	36	
	Zusätzliche Klassifikation über LLM	69	
Vehicles *Score: 72%*	Fahrzeuge-Film Informationen	18	
	Filme ohne Fahrzeuginformation	7	∼28%
Vehicles-Bilder *Score: 95%*	Fahrzeuge-Film Beziehungen	225	100%
	Fahrzeuge mit Bild	214	
	Fahrzeuge ohne Bild	11	∼5%
Theme Songs *Score: 100%*	Titel Song und Performer	25	100%
	Intro Sequenz von YouTube manuell ergänzt	25	
James Bond Darsteller *Score: 100%*	Wikidata IDs	6	100%
	Geschlecht (alle sind männlich)	1	
	Geburtstag (und ggf. Todestag)	6	
	Staatsbürgerschaft	4	

4 Knowledge Engineering

Während sich das vorherige Kapitel auf die Extraktion von Wissen aus strukturierten und unstrukturierten Datenquellen konzentriert, liegt der Fokus dieses Kapitels auf der formalen Modellierung und semantischen Strukturierung der extrahierten Informationen. Im Rahmen des Knowledge Engineerings werden Klassen, Eigenschaften sowie deren Relationen definiert und in einer Ontologie abgebildet. Diese konzeptuelle Modellierung ermöglicht die Abbildung domänenspezifischer Zusammenhänge im James-Bond-Universum und stellt die Grundlage für die instanzielle Repräsentation des Wissens in Form eines RDF-basierten Knowledge Graphs dar.

4.1 Idee

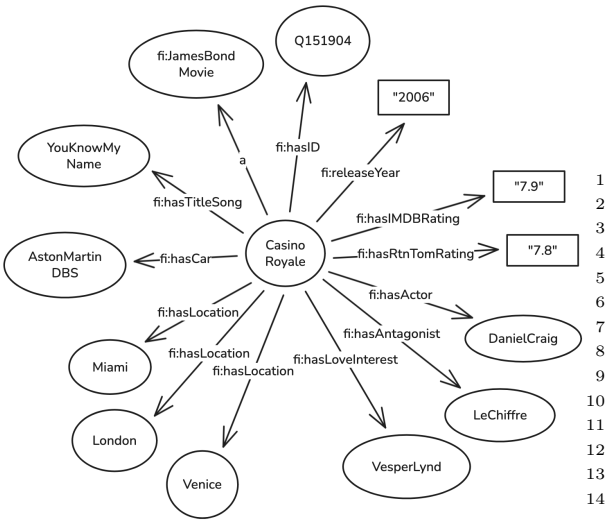
Im Rahmen der Themenfindung wurde ein erstes Konzept zur Wissensmodellierung des James-Bond-Universums entwickelt. Aufbauend auf dem Kerndatensatz sowie den zusätzlich extrahierten Informationen sollen zentrale Entitäten identifiziert und semantisch modelliert werden. Im Fokus stehen dabei insbesondere folgende Entitätstypen:

- **Film:** Titel, Erscheinungsjahr, Genre
- **Schauspieler:** Beteiligung am Film, Filmographie, biografische Daten
- **Regisseur und Producer:** Regie beim Film
- **Musik/Band:** Soundtrack, Titel, Performer, Videosequenzen
- **Locations und Vehicles:** Filmschauplätze, Fahrzeuge

Diese Entitäten werden durch semantische Relationen miteinander verknüpft, etwa über Beziehungen wie “hat Schauspieler”, “hat Antagonist”, “hat Drehort” oder “hat Titelsong”. Ziel ist die Abbildung inhaltlicher Zusammenhänge innerhalb des James-Bond-Universums in Form eines strukturierten Knowledge Graphs.

Abbildung 7 zeigt exemplarisch die konkrete Instanziierung dieser Modellierung anhand des Films *Casino Royale*. Dabei wird links eine graphische Darstellung der A-Box präsentiert, welche die instanziierten Entitäten und deren Beziehungen visualisiert. Rechts ist dieselbe Struktur als formale, triple-basierte Repräsentation in Turtle-Syntax

dargestellt. Beide Darstellungen beschreiben dieselben Sachverhalte auf unterschiedlichen Abstraktionsebenen.



(a) Graphische Modellierung.

```
:CasinoRoyale
    a                       fi:JamesBondMovie ;
    fi:hasID                :Q151904 ;
    fi:releaseYear          "2006" ;
    fi:hasIMDBRating        "7.9" ;
    fi:hasRtnTomRating      "7.8" ;
    fi:hasActor             :DanielCraig ;
    fi:hasAntagonist        :LeChiffre ;
    fi:hasLoveInterest      :VesperLynd ;
    fi:hasLocation          :Venice,
                            :London,
                            :Miami ;
    fi:hasCar               :AstonMartinDBS ;
    fi:hasTitleSong         :YouKnowMyName .
```

(b) Triple-basierte Modellierung in Turtle.

Abbildung 7: Visuelle und formale Repräsentation des Films *Casino Royale* im Knowledge Graph (eigene Darstellung).

Abbildung 8 zeigt ergänzend die T-Box der Ontologie. Sie definiert die zugrunde liegenden Klassen, Eigenschaften sowie deren Domain- und Range-Beschränkungen und bildet damit den konzeptuellen Rahmen für die instanziierten Daten der A-Box.

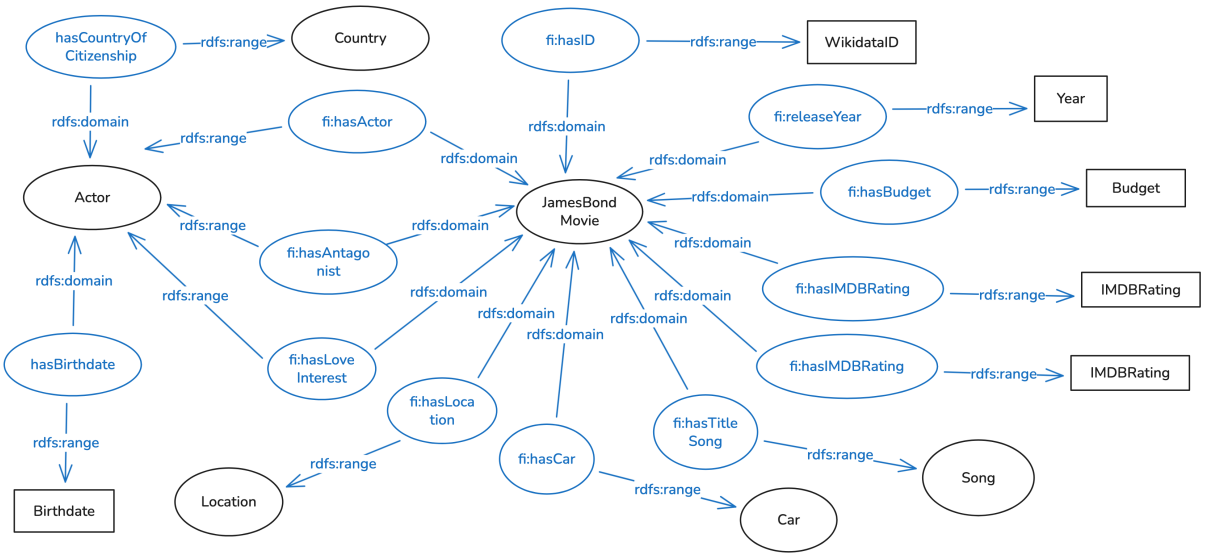


Abbildung 8: Visuelle Darstellung der T-Box der James-Bond-Ontologie mit Klassen, Properties sowie Domain- und Range-Definitionen (eigene Darstellung).

4.2 Vokabular und Namespaces

Für die Ontologie des James-Bond-Universums wird ein einheitliches Vokabular eingesetzt, das sowohl externe Ontologien als auch projektspezifische Erweiterungen umfasst. Ziel ist es, bestehende Standards möglichst wiederzuverwenden und diese gezielt um domänenspezifische Konzepte zu ergänzen. Die folgenden Namespaces werden im Rahmen der Ontologie verwendet:

- `movie: <https://triplydb.com/Triply/linkedmdb/vocab/>`

  Zur Modellierung zentraler Film- und Personenklassen auf Basis des LinkedMDB-Vokabulars. (Tripathi, 2024)

- `dbo: <http://dbpedia.org/ontology/>`

  Zur Abbildung biografischer Attribute sowie musikalischer Entitäten aus der DBpedia-Ontologie.

- `foaf: <http://xmlns.com/foaf/0.1/>`

  Zur Beschreibung grundlegender Personenattribute wie Namen und Geschlecht.

- `schema: <http://schema.org/>`

  Zur Verknüpfung externer Ressourcen und Medieninhalte (z. B. `sameAs`, Bilder und URLs).

- `time: <http://www.w3.org/2006/time#>`

  Zur Modellierung zeitlicher Informationen, insbesondere des Erscheinungsjahres von Filmen.

- `geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>`

  Zur Repräsentation geografischer Koordinaten von Drehorten.

- `bond <http://example.org/bond/>`

  Projektspezifischer Namespace zur Abbildung domänenspezifischer Konzepte und Relationen des James-Bond-Universums.

Die konkrete Verwendung der einzelnen Namespaces innerhalb der Ontologie ist in Tabelle 3 zusammengefasst.

Tabelle 3: Übersicht der verwendeten Namespaces und zugehörigen Klassen bzw. Properties (eigene Darstellung).

Namespace	Beschreibung	Verwendete Klassen / Properties
`bond:`	Domänenspezifische Erweiterungen für das James-Bond-Universum	`bond:BondActor`, `bond:BondGirl`, `bond:Villain`, `bond:Vehicle`, `bond:hasActor`, `bond:hasJamesBond`, `bond:hasAntagonist`, `bond:hasBondGirl`, `bond:hasLocation`, `bond:hasVehicle`, `bond:hasThemeSong`, `bond:imdbRating`, `bond:rtRating`
`movie:`	Generische Film- und Personenklassen aus LinkedMDB	`movie:Film`, `movie:Actor`, `movie:Director`, `movie:Producer`, `movie:FilmLocation`, `movie:MusicContributor`, `movie:Person`
`dbo:`	Biografische und musikalische Konzepte aus DBpedia	`dbo:birthDate`, `dbo:deathDate`, `dbo:citizenship`, `dbo:Song`
`foaf:`	Grundlegende Personenattribute	`foaf:name`, `foaf:gender`
`schema:`	Verlinkung externer Ressourcen und Medien	`schema:sameAs`, `schema:image`, `schema:url`
`time:`	Zeitliche Modellierung	`time:year`
`geo:`	Geografische Koordinaten	`geo:lat`, `geo:long`

4.3 Umsetzung

Zur Erstellung eines konsistenten und integrierten Knowledge Graphs werden in einem abschliessenden Schritt sämtliche extrahierten Daten zusammengeführt. Hierzu werden die Inhalte des Kerndatensatzes sowie die Ergebnisse der strukturierten und unstrukturierten Wissensextraktion in ein gemeinsames Datenformat überführt.

Mit `Skript p` werden alle vorhandenen CSV- und JSON-Dateien eingelesen und in einem zentralen, semi-strukturierten JSON-Dokument zusammengeführt. Dieses Dokument dient als Zwischenschicht zwischen der Datenextraktion und der RDF-basierten Modellierung und stellt sicher, dass alle relevanten Informationen konsistent und einheitlich vorliegen. Die resultierende JSON-Struktur ist hierarchisch aufgebaut und umfasst zwei zentrale Hauptebenen: (1) `"movies"`, welche alle Filme mit ihren Metadaten sowie Informationen zu Bond-Girls, Antagonisten, weiteren Charakteren, Drehorten und Fahrzeugen enthält, und (2) `"actors"` , welche die sechs James-Bond-Hauptdarsteller

inklusive ihrer biografischen Attribute abbildet.

Die Struktur dieser beiden Hauptebenen ist exemplarisch in Abbildung 9 dargestellt.


```
{
  "movies": [
    {
      "movie": "A View to a Kill",
      "title_en": "A View to a Kill",
      "title_de": "James Bond 007 - Im Angesicht des Todes",
      "year": "1985",
      "bond_actor": "Roger Moore",
      "bond_actor_qid": "Q134333",
      "director": "John Glen",
      "producer": "Albert R. Broccoli & Michael G. Wilson",
      "imdb_rating": "6.2",
      "rotten_tomatoes_rating": "4.7",
      "wikidata_id": "http://www.wikidata.org/entity/Q332368",
      "bond_girls": [
        {
          "name": "Stacey Sutton",
          "actress": "Tanya Roberts",
          "image_url": "https://static.wikia.nocookie.net/jamesbo
        }
      ],
      "characters": [
        {
          "name": "James Bond",
          "actor": "Roger Moore",
          "image_url": "https://static.wikia.nocookie.net/jamesbo
        },
```


(a) Ausschnitt der `"movies"`-Ebene mit Filmmetadaten und zugehörigen Entitäten.


```
  "actors": {
    "Q4547": {
      "actor_uri": "http://www.wikidata.org/entity/Q4547",
      "label": "Daniel Craig",
      "birth_date": "1968-03-02",
      "death_date": null,
      "genders": [
        {
          "uri": "http://www.wikidata.org/entity/Q6581097",
          "label": "male"
        }
      ],
      "citizenships": [
        {
          "uri": "http://www.wikidata.org/entity/Q145",
          "label": "United Kingdom"
        },
        {
          "uri": "http://www.wikidata.org/entity/Q30",
          "label": "United States"
        }
      ]
    },
```


(b) Ausschnitt der `"actors"`-Ebene mit biografischen Informationen der Bond-Darsteller.

Abbildung 9: Hierarchische Struktur des zentralen JSON-Dokuments als Integrationsschicht für den Knowledge Graph (james_bond_knowledge.json).

In `Skript q` wird die T-Box der Ontologie definiert. Diese umfasst sämtliche relevanten Klassen (Movie, Actor, Character, BondGirl, Villain, Vehicle, Location, Song) sowie die zugehörigen *Object Properties* (Relationen zwischen den Klassen) und *Data Properties* (Attribute mit Literalwerten). Auf Basis dieser T-Box sowie der extrahierten und bereits vorliegenden Daten werden anschliessend die RDF-Tripel für die beiden Hauptebenen `"movies"` und `"actors"` generiert und damit der Knowledge Graph konstruiert. Der resultierende Graph wird anschliessend sowohl im Turtle-Format (`.ttl`) als auch im OWL-Format (`.owl`) serialisiert, um nachgelagerte Reasoning-Schritte und semantische Inferenzprozesse zu ermöglichen.

In Abschnitt 5 und Abbildung 13 wird der resultierende Graph mithilfe der Streamlit-Anwendung visualisiert und dargestellt. Abbildung 10 veranschaulicht die T-Box der Ontologie und illustriert die konzeptuelle Struktur des Knowledge Graphs in Form der zugrundeliegenden Klassen, Relationen und Attribute.

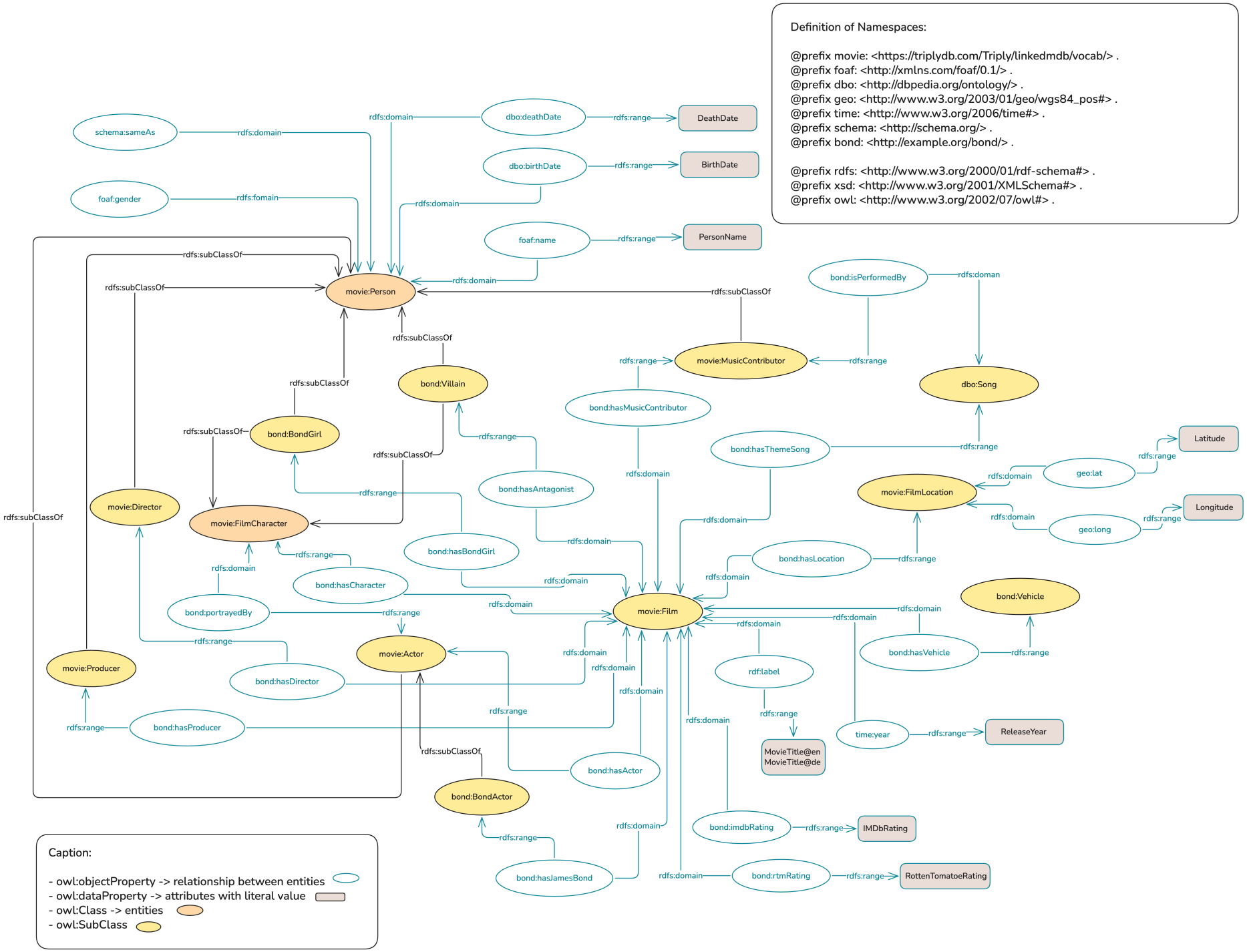


Abbildung 10: T-Box der James-Bond-Ontologie mit Klassenhierarchie sowie definierten Objekt- und Datenrelationen (eigene Darstellung).

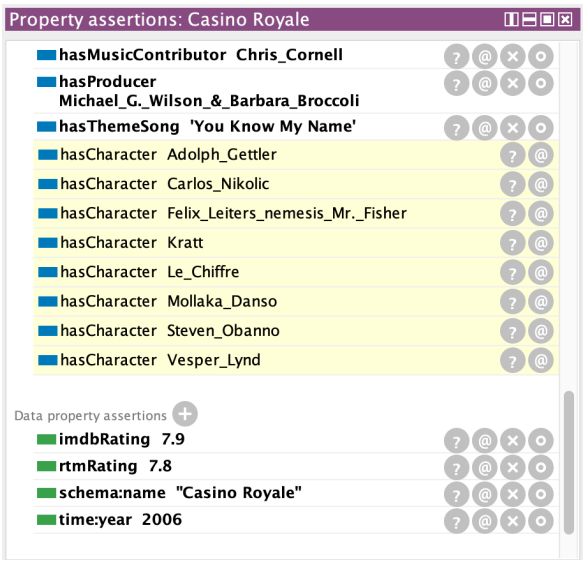
4.4 Reasoning

Abschliessend wurde ein Reasoning durchgeführt, um einerseits die Konsistenz und Korrektheit der definierten Ontologie zu überprüfen und andererseits zusätzliche, implizite Relationen abzuleiten. In einem ersten Schritt wurde das Reasoning interaktiv mit dem Tool Protégé durchgeführt. Hierzu wurde der erzeugte Knowledge Graph über das Turtle-File geladen und das integrierte Reasoning mit dem Reasoner HermiT (Version 1.4.3.456) gestartet.

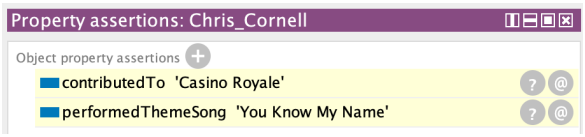
Die initialen Reasoning-Durchläufe meldeten eine inkonsistente Ontologie. Die Ursachen lagen in mehreren Datentyp-Inkonsistenzen: Die Properties `schema:image` und `schema:url` waren mit dem Range `xsd:anyURI` definiert, während die zugehörigen Werte noch als einfache Strings vorlagen. Zudem führte das Filmlabel wie beispielsweise *"Die Another Day"@en* zu einem Konflikt mit einer zu restriktiven Definition von `rdfs:label` als `xsd:string`. Schliesslich war die Property `time:year` zunächst als `xsd:date` modelliert, obwohl ausschliesslich vollständige `xsd:dateTime` Formate unterstützt werden.

Nach der Normalisierung der URLs zu `xsd:anyURI` sowie der Änderung von `time:year` auf den Datentyp `xsd:integer` konnte die Ontologie erfolgreich konsistent gemacht werden. Die ursprünglich vorgesehenen Label-Constraints wurden weggelassen, da sie nicht zwingend definiert werden müssen und sich als zu restriktiv erwiesen hatten. Die anschliessenden Reasoning-Durchläufe verliefen ohne Fehlermeldungen und bestätigten die korrekte Definition der Klassenhierarchie sowie der semantischen Relationen.

Abbildung 11 zeigt exemplarisch die durch das Reasoning gewonnenen zusätzlichen Aussagen (gelb hervorgehoben) anhand des Films *Casino Royale* im Tool Protégé.



(a) Durch Reasoning abgeleitete Relationen beim Film *Casino Royale*.



(b) Inverse Musikrelationen bei *Chris Cornell*.

(c) Inverse Beziehung für *Vesper Lynd*.

(d) Inverse Beziehung für *Carlos Nikolic*.

Abbildung 11: Ergebnisse des Ontology Reasonings in Protégé mit HermiT. Neu abgeleitete Aussagen sind gelb hervorgehoben.

In Abbildung 11 ist zu erkennen, dass dem Film *Casino Royale* durch das Reasoning zusätzliche Relationen hinzugefügt wurden, unter anderem mehrere Charakterverknüpfungen über die Property `bond:hasCharacter`. Darüber hinaus wurde für die Instanz *Chris Cornell* korrekt abgeleitet, dass er über die inverse Relation `bond:contributedTo` mit dem Film *Casino Royale* verbunden ist und den Titelsong *You Know My Name* performt hat. Vor dem Reasoning war diese Beziehung lediglich einseitig über `bond:hasMusicContributor` am Film modelliert. Ein analoges Verhalten ist bei den inversen Relationen `bond:isBondGirlIn` für *Vesper Lynd* sowie `bond:isAntagonistIn` für *Carlos Nikolic* zu beobachten. Diese automatisch abgeleiteten Aussagen bestätigen die korrekte Definition der Inversen in der Ontologie und zeigen, dass das Reasoning wie beabsichtigt zusätzliche, semantisch konsistente Beziehungen erschliesst.

Im abschliessenden Schritt der Datenpipeline wurde das Reasoning mit `Skript r` mithilfe von Python und der Bibliothek `owlready2` automatisiert durchgeführt, um den gesamten Workflow von der Datenextraktion bis zur Inferenz vollständig zu integrieren. Die daraus resultierende, inferierte Ontologie wurde anschliessend in einer separaten Datei gespeichert.

5 Visualisierung der Ergebnisse

Die Streamlit-App ist öffentlich zugänglich unter James Bond Universe. Alternativ kann das Projekt auch lokal eingerichtet und ausgeführt werden. Hierfür steht ein öffentliches GitHub Repositry zur Verfügung, dessen README.md-Datei eine Installationsanleitung enthält.

Die App gliedert sich in sechs thematische Seiten, die über die Navigationsleiste links ausgewählt werden können (Abbildung 12). Das nachfolgende Kapitel konzentriert sich auf zwei zentrale Visualisierungen zur explorativen Analyse des Knowledge Graphs. Erstens die Darstellung des *RDF-Graphs*, der die semantischen Verknüpfungen sichtbar macht und zweitens die *Analyse wiederkehrender Charaktere*, die Muster über mehrere Filme hinweg aufdeckt.

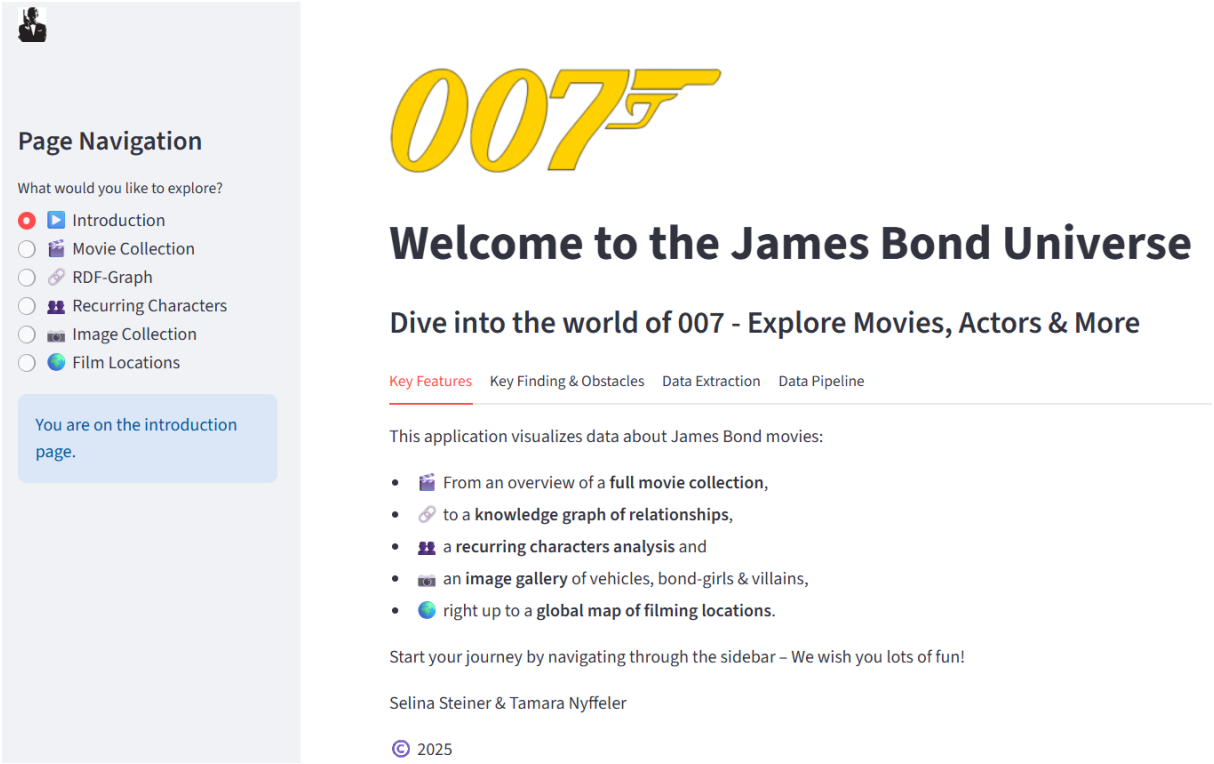


Abbildung 12: Einstiegsseite der Streamlit-App mit *Page Navigation* (app.py).

5.1 RDF-Graph

Angesichts der Vielzahl an Entitäten und Relationen im Knowledge Graph wäre eine ungefilterte Gesamtdarstellung nur schwer interpretierbar. Aus diesem Grund wurde ein zweistufiger Filtermechanismus implementiert. Zunächst kann definiert werden, welches

semantische Merkmal (z.B. Charaktere, Fahrzeuge, Schauplätze, etc.) im Fokus stehen soll. Anschliessend lassen sich über ein Dropdown-Menü ein oder mehrere Filmtitel auswählen, um die Visualisierung auf die gewünschten Instanzen zu beschränken. Abbildung 13 zeigt beispielhaft den RDF-Graphen für die Filme *Diamonds Are Forever* und *Casino Royale*. In der Ansicht „Movie Overview“ werden dabei die wichtigsten Entitäten und ihre Beziehungen zueinander dargestellt.

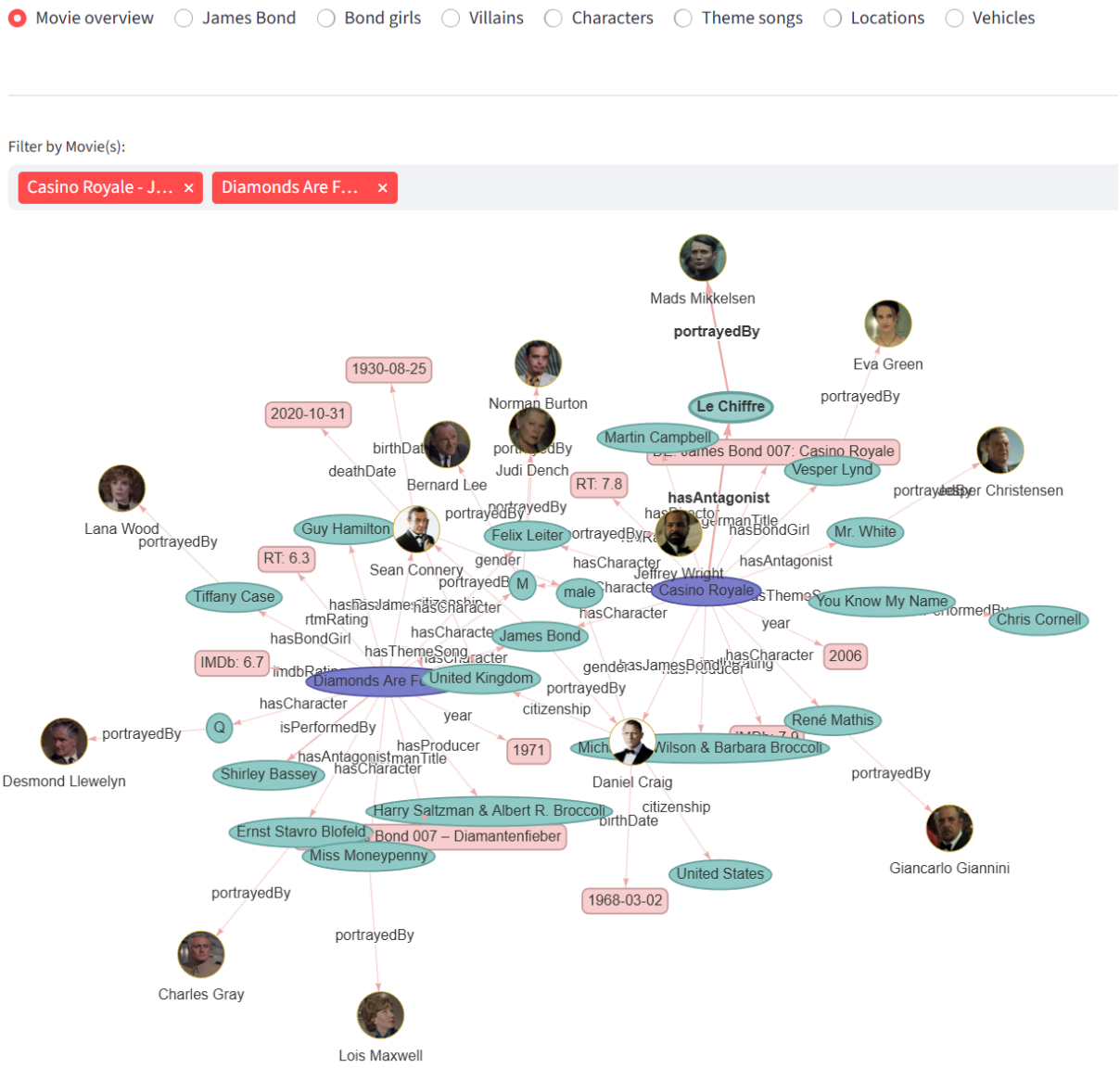


Abbildung 13: Visualisierung des RDF-Graphs für die Filme *Diamonds Are Forever* und *Casino Royale* (app.py).

Die Filterung der semantischen Merkmalskategorien wie *Bond Girls*, *Villains*, *Locations* etc. erfolgt mittels einer SPARQL-Abfrage, welche ausschliesslich den relevanten Teilgraphen aus der zugrunde liegenden Turtle-Repräsentation extrahiert. Die Resultate der Abfrage werden direkt in entsprechende Node-Objekte überführt, die anschliessend

im Graphen visualisiert werden. Auf diese Weise wird nicht der vollständige Knowledge Graph gerendert, sondern lediglich der nutzerspezifisch relevante Teilgraph in Abhängigkeit von der jeweiligen Auswahl.

In Code 9 ist dieser Filtervorgang exemplarisch für die Entitätengruppe der *Bond Girls* dargestellt. Dabei werden mittels der SPARQL-Abfrage ausschliesslich Instanzen der Klasse `bond:BondGirl` aus dem vollständigen RDF-Graphen extrahiert.

```
bondgirl_query = '''
        PREFIX bond: <http://example.org/bond/>
        PREFIX foaf: <http://xmlns.com/foaf/0.1/>
        SELECT DISTINCT ?bondgirl ?name
        WHERE {
            ?bondgirl a bond:BondGirl .
            ?bondgirl foaf:name ?name .
        }
    '''
    bondgirls = [
        Node(id=str(row[0]),
                label=str(row[1]),
                color='#7CCCC7',
                shape='ellipse',
                size=20)
        for row in g.query(bondgirl_query)
    ]
```

Code 9: SPARQL-basierte Filterung von Teilgraphen am Beispiel der Bond-Girls (rdf_graph.py).

5.2 Analyse wiederkehrender Charaktere

Zur Identifikation ikonischer Charaktere wurde ein Scatterplot entwickelt, bei dem die horizontale Achse die Charakternamen und die vertikale Achse die Filmtitel abbildet. Ein Range Slider ermöglicht die Filterung nach Mindestanzahl der Filmvorkommnisse, während über ein Dropdown-Menü gezielt einzelne Charaktere ausgewählt werden können. Abbildung 14 zeigt exemplarisch die Visualisierung bei einer Filterung auf mindestens drei Vorkommnisse.

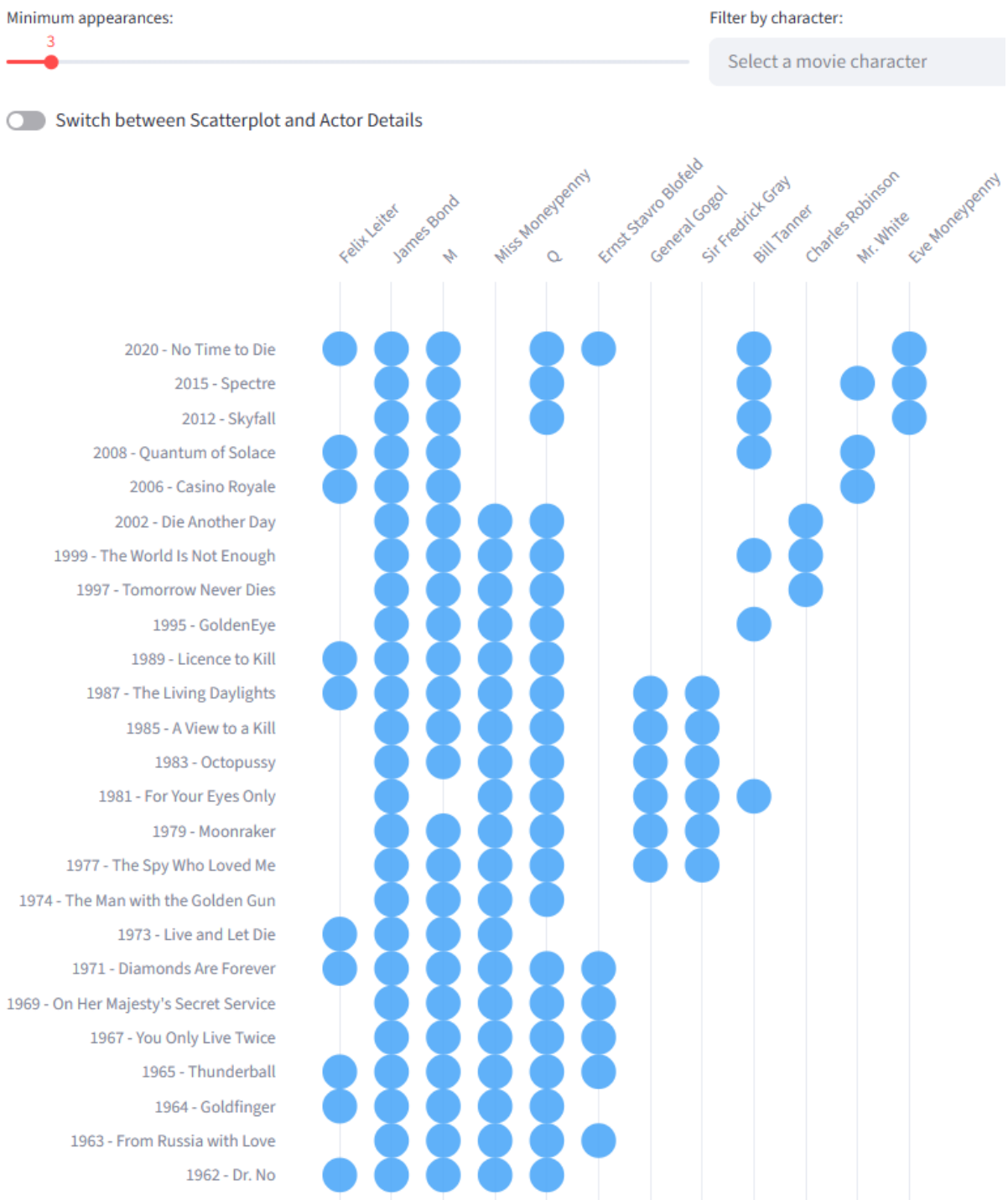


Abbildung 14: Visualisierung für wiederkehrende Charaktere, mit Filterung auf mind. 3 Filmvorkommnisse (app.py).

Die Ansicht „Switch between Scatterplot and Actor Details“ zeigt zusätzlich detaillierte Informationen zu den Darsteller:innen mit Bild. Am Beispiel der Figur „M“ wird demnach deutlich: Sie tritt in 24 von 25 Filmen auf (die Ausnahme ist der Film *For Your Eyes Only*), wurde bis 1995 von männlichen Schauspielern dargestellt, von 1995 bis 2012 von *Judi Dench* als Frau verkörpert und erscheint in den beiden neuesten Filmen (*Spectre, No*

Time to Die) mit *Ralph Fiennes* wieder in männlicher Besetzung (Abbildung 15).

M	The Man with the Golden Gun *James Bond 007 – Der Mann mit dem goldenen Colt*	Bernard Lee	1974		M	The World Is Not Enough *James Bond 007 – Die Welt ist nicht genug*	Judi Dench	1999	
M	The Spy Who Loved Me *James Bond 007 – Der Spion, der mich liebte*	Bernard Lee	1977		M	Die Another Day *James Bond 007 – Stirb an einem anderen Tag*	Judi Dench	2002	
M	Moonraker *James Bond 007 – Moonraker – Streng geheim*	Bernard Lee	1979		M	Casino Royale *James Bond 007: Casino Royale*	Judi Dench	2006	
M	Octopussy *James Bond 007 – Octopussy*	Robert Brown	1983		M	Quantum of Solace *James Bond 007: Ein Quantum Trost*	Judi Dench	2008	
M	A View to a Kill *James Bond 007 – Im Angesicht des Todes*	Robert Brown	1985		M	Skyfall *James Bond 007: Skyfall*	Judi Dench	2012	
M	The Living Daylights *James Bond 007 – Der Hauch des Todes*	Robert Brown	1987		M	Spectre *James Bond 007: Spectre*	Ralph Fiennes	2015	
M	Licence to Kill *James Bond 007 – Lizenz zum Töten*	Robert Brown	1989		M	No Time to Die *James Bond 007: Keine Zeit zu sterben*	Ralph Fiennes	2020	

Abbildung 15: Exemplarische Darstellung der Schauspieler Details für den Charakter *„M“* (app.py).

5.3 Datengrundlage

Die Darstellung des RDF-Graphs basiert auf dem serialisierten Turtle-File (wie in Abschnitt 4.3 beschrieben). Während das zugrunde liegende JSON-Dokument alle extrahierten Charaktere vollständig enthält, erfolgt bei der Serialisierung in das Turtle-Format eine Filterung. Es werden folglich nur Charaktere übernommen, die in mindestens zwei Filmen vorkommen. Diese Einschränkung dient der Übersichtlichkeit des Knowledge Graphs. Ausgenommen von dieser Filterung sind die Subklassen Bond Girls und Villains. Diese werden aufgrund ihrer zentralen Bedeutung vollständig vom JSON in das Turtle-File serialisiert.

Alle weiteren Visualisierungen in der App greifen auf die jeweiligen CSV-Dateien als Datengrundlage zurück, die im Rahmen der Knowledge Extraction erstellt wurden.

6 Diskussion

6.1 Mehrwert für Nutzerinnen und Nutzer

Die Wissensdatenbank sowie die Bereitstellung der Daten in der Streamlit-App bieten einen Zugang zum James-Bond-Universum, der über eine konventionelle Filmdatenbank hinausgeht. Bestehende online Plattformen ermöglichen primär die Betrachtung isolierter Faktendaten. Die vorliegende Lösung ermöglicht den Nutzenden eine semantische Analyse filmübergreifender Zusammenhänge.

Ein zentraler Mehrwert der Lösung besteht in der Integration heterogener Datenquellen. Strukturierte Informationen aus Wikidata werden dabei mit unstrukturierten Daten aus Wiki-Fandom, Wikipedia sowie LLM-generierten Inhalten zusammengeführt. Die interaktiven Visualisierungen ermöglichen es den Nutzenden, spezifische Fragestellungen gezielt zu explorieren.

6.2 Mehrwert der Ontologie

Die Ontologie bildet die semantische Grundlage der Wissensdatenbank und ermöglicht eine strukturierte, konsistente Modellierung zentraler Entitäten sowie ihrer Relationen im James-Bond-Universum. Durch die Harmonisierung und Integration heterogener Datenquellen sowie den Einsatz von Reasoning werden quellenübergreifende Abfragen sowie die Ableitung implizit vorhandenen Wissens ermöglicht. Sie stellt damit eine flexible und erweiterbare Basis für die Analyse und zukünftige Erweiterung des Datenbestands.

6.3 Beantwortung der Fragestellungen

- **Beziehungsanalyse:** Der RDF-Graph visualisiert die Beziehungen zwischen den zentralen Entitäten des James-Bond-Universums. Dadurch lässt sich nachvollziehen, wie James Bond mit Antagonisten, Bond Girls, Schauplätzen, Fahrzeugen oder Titelsongs verknüpft ist. Mithilfe verschiedener Filteroptionen können diese Beziehungen gezielt untersucht und die Fragestellung beantwortet werden.
- **Wiederkehrende Elemente:** Die Darstellung im Scatterplot ermöglicht eine Analyse wiederkehrender Charaktere durch eine Filterung nach Mindestanzahl an

Auftritten. Diese Visualisierung hilft dabei, ikonische Filmfiguren zu identifizieren. Neben James Bond selbst weisen insbesondere die Rollen *M, Q* und *Miss Moneypenny* die höchste Anzahl an Auftritten auf.

- **Entwicklung über die Zeit:** Die zeitliche Entwicklung einzelner Rollen kann sowohl über die Bildergalerie als auch über den RDF-Graphen über mehrere Filme hinweg untersucht werden. Anhand von Figuren wie *Blofeld, Q* oder *M* wird sichtbar, wie sich deren Darstellung im Laufe der Jahre verändert hat; etwa im Hinblick auf unterschiedliche Schauspieler:innen, Geschlecht oder Altersgruppen.

6.4 Limitationen

Trotz insgesamt guter Datenqualität bestehen weiterhin Lücken, etwa durch fehlende Fahrzeuginformationen (Abschnitt 3.3). Zudem ist die LLM-basierte Extraktion anfällig für Fehlklassifizierungen. Im Zuge der Datenvalidierung wurden auch inhaltliche Diskrepanzen in den generierten Texten festgestellt. So überlebt beispielsweise der Antagonist *Jaws* in beiden Filmen (*Moonraker* und *The Spy Who Loved Me*), wurde vom LLM jedoch fälschlicherweise mit Status *„Killed by Bond“* klassifiziert. Solche inhaltliche Fehler lassen sich nur mit spezifischem Domänenwissen erkennen.

Hinzu kommt die Frage der Robustheit und Wartbarkeit des Codes gegenüber zukünftigen Änderungen. Die Datenpipeline ist in Python modular aufgebaut, sodass die einzelnen Skripte sequenziell und weitgehend unabhängig voneinander ausgeführt werden können. Im abschliessenden Verarbeitungsschritt erfolgt eine zentrale Data Fusion mit Reasoning, wodurch neue Datenquellen grundsätzlich einfach in den bestehenden Knowledge-Extraction-Prozess integriert werden können.

Gleichzeitig stellen die in den Extraktionsskripten enthaltenen manuellen Mappings eine potenzielle Schwachstelle dar. Diese sind eng an die aktuelle Struktur der Quelldaten gekoppelt und können bei Änderungen der Datenformate oder Inhalte ihre Gültigkeit verlieren. Eine klassenorientierte und abstrahierte Code-Struktur könnte dazu beitragen, diese Abhängigkeiten besser zu kapseln und die langfristige Wartbarkeit der Pipeline zu verbessern.

6.5 Reflexion

Das Projekt war eine spannende und lehrreiche Erfahrung, die uns tiefere Einblicke in die praktische Anwendung von Knowledge Engineering und Extraction ermöglichte. Besonders wertvoll war die direkte Umsetzung theoretischer Konzepte aus dem Unterricht in ein konkretes Projekt mit einer Vielzahl semantisch modellierten Entitäten.

Anwendung der Unterrichtsinhalte
Der Umgang mit SPARQL- und API-Endpunkten, den wir im Unterricht kennengelernt hatten, erwies sich als zentral für die Extraktion strukturierter und unstrukturierter Daten. Die Entwicklung einer eigenen Ontologie unter Verwendung etablierter Vokabulare und das anschliessende Reasoning mit HermiT waren weitere Kernelemente, die wir direkt aus dem Unterricht übernehmen konnten.

Ein besonders spannender Aspekt war die Integration von Large Language Models in den Extraktionsprozess. Diese Methode erwies sich als besonders effektiv bei Klassifikationsaufgaben, wodurch wir die Anzahl extrahierter Bösewichte nahezu verdoppeln konnten. Dies zeigte uns eindrücklich, wie moderne KI-Technologien die klassischen Methoden der Wissensextraktion sinnvoll ergänzen können.

Herausforderungen und Lösungsansätze
Trotz guter theoretischer Vorbereitung traten auch ein paar technische Herausforderungen auf. Besonders die Extraktion von Bild-URLs erwies sich zu Beginn des Projekts als problematisch. Nachdem wir jedoch verschiedene Ansätze ausprobiert und uns eingehender mit dem API-Aufruf und der Zusammensetzung der URLs beschäftigt hatten, konnten wir diese Herausforderung erfolgreich meistern. Trotzdem war ein manuelles Mapping für einen geringen Prozentsatz der Entitäten notwendig.

Beim Reasoning mit HermiT in Protegé meldeten die ersten Durchläufe Inkonsistenzen aufgrund falscher Datentypen. Auch hier mussten wir uns nochmals tiefer mit dem Aufbau der Serialisierung befassen und Korrekturen vornehmen. Danach lief das Reasoning jedoch fehlerfrei durch und konnte die gewünschten inversen Relationen ableiten.

Fazit
Insgesamt vermittelte uns das Projekt wertvolle Kenntnisse für zukünftige Projekte und den Berufsalltag. Besonders in Bezug auf die Knowledge Extraction wurde dennoch auch deutlich, wie entscheidend die Wahl des Themas für den Projekterfolg ist. Der Fokus auf das James-Bond-Universum erwies sich als sehr geeignet, da zahlreiche

öffentlich zugängliche Datenquellen verfügbar sind, was uns die Recherche erleichtert und das Projekt inhaltlich bereichert hat.

Literaturverzeichnis

Block, A. (2020). James Bond Movie Dataset. Verfügbar 28. November 2025 unter https://www.kaggle.com/datasets/dreb87/jamesbond

GroqCloud - Build Fast. (n. d.). Verfügbar 15. Dezember 2025 unter https://console.groq.com

James Bond Wiki. (n. d.). Verfügbar 15. Dezember 2025 unter https://jamesbond.fandom.com/wiki/James_Bond_Wiki

Tripathi, A. (2024). Linked Internet Movie Database (IMDb). Verfügbar 15. November 2025 unter https://triplydb.com/AradhyaTripathi/linkedmdb

Wikidata. (n. d.). Verfügbar 15. Dezember 2025 unter https://www.wikidata.org/wiki/Wikidata:Main_Page