Investigative Report on Ontology in Practice and Its Link to Large Language Models (LLMs): A Case Study on Semantic Content Recommendation for Streaming Platforms

Mayank Pareek

Abstract-- This paper looks at how ontologies work in real life. It checks out their mix with large language models in a specific area. The case here involves a system like Netflix for recommending content. That setup helps show how semantic modeling and description logic can fix issues with metadata that does not match up. It also tackles transparency and better recommendations on streaming sites. The study follows a clear method to build the ontology. That includes steps like specifying needs, conceptualizing ideas, formalizing them, putting it into code, and checking results. It shows what the ontology can do with reasoning and queries. Examples come from competency questions, DL rules, and SPARQL queries. The paper goes on to explain how ontologies pair up with LLMs in hybrid AI setups. They provide semantic grounding and check constraints. Plus, they make things more explainable overall.

KEYWORDS-- Ontology, Knowledge Representation, Recommendation Systems, Description Logic, Semantic Web, Large Language Models, Knowledge Graphs, Metadata, Streaming Platforms, Artificial Intelligence

I. INTRODUCTION

Artificial Intelligence technology went through numerous changes over the past decade because of the advancements in machine learning, deep neural networks, and Large Language Models (LLMs). These models tackle the task related to natural language synthesis and pattern recognition very well. But they lack on real semantic understanding that ties to the world and cannot ensure logical consistency and factual correctness. Bender and Koller make a strong case for this [1], LLMs operate on statistical correlations rather than meaning, making them unreliable for tasks requiring structured reasoning. The issue also covered in the week1 of the module, highlights the need for symbolic approaches that provide explicit semantics.

Knowledge Representation and Reasoning, or KR and R, address this problems head on. By enabling machine to work with structured knowledge instead of raw statistical associations. Ontologies come here as a key tool. Ontologies—defined by Gruber as “an explicit specification of a conceptualization” [2]—offer a formal, machine-interpretable representation of domain concepts, relationships and constraints. Ontologies build on Description Logic, or DL support automated reasoning, consistency checking, classification and inference. Models based on ontologies allow machines to apply logical rules, maintain semantic consistency and derive new knowledge.

Berners-Lee et al.’s foundational work on the Semantic Web [3] shows how logical reasoning and machine readable metadata allow systems to understand, integrate and interpret data from various heterogenous sources. These capabilities are important in the domain where explainability, reliability and transparency is very crucial.

One such domain is recommendation system in entertainment industry. OTT platforms like Netflix, Amazon Prime Video rely on information like genre, themes, rating and cast to build user preference profile and personalised recommendations. But often the metadata is incomplete, inconsistence. Sometimes it just stays vague. As Gómez-Uribe and Hunt described in their study of the Netflix recommender system [4], metadata quality significantly affects user satisfaction and recommendation accuracy. And this accuracy drop even more when the system face hybrid genre for example anime where cultural details bring in subtleties which make navigation for statistical tool messy.

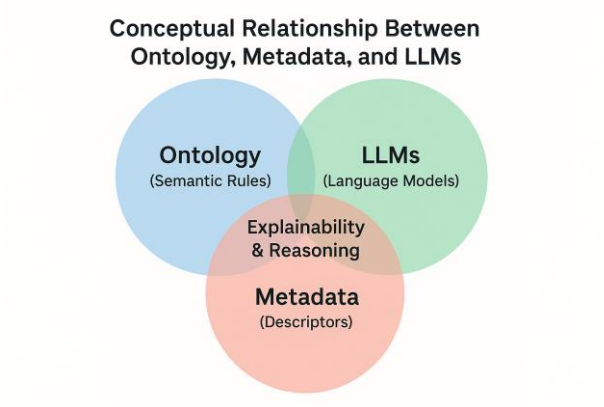
While the LLM models are expert in understanding and interpreting complex natural language query, they struggle with the semantic rules needed for solid content suggestion. Without

explicit axioms LLM model can misclassify a hybrid genre content or recommend mature content to younger audience. Ontologies can help in tackling this issue. As ontologies allows to create a strong sematic framework, build constraint and axioms, supporting reason based recommendations and representing content relationships.

This reports builds a content recommendation ontology for platform like Netflix, with enriched categories to match the complexity of real world metadata. This domains fit well with KR&R principles, propositional and first-order logic, DL-based modelling, semantic web principles and reasoning workflows using Protégé.

The report offers three main contributions. It spots and explains a practical issue where ontologies beat out stats only models. It builds an ontology step by step. That covers specification, conceptualization, formalization, implementation, and evaluation. It shows what the ontology can do. That includes reasoning tasks, DL queries, SPARQL queries, and a full case study. It wraps up with thoughts on blending ontological setups with LLMs.

Through this analysis, the report emphasize on the resourcefulness and need of Ontologies in today's AI systems. Although, LLM provide more flexibility with natural language process and general information process, ontologies provide sharp semantics, consistency, transparency, precision, strict logic and explainability. Qualities necessary to build a safe model and user preferred intelligent systems in the entertainment sector.



II. Problem Identification and Justification

As the world moves towards OTT from local disk, the number and diversity of the contents is rapidly increasing on streaming platform like Netflix, Crunchyroll and Amazon prime. Out of the vast variety and sea of content suggesting user preferred content is really important for user retention and engagement. These platforms rely on metadata to suggest new content to the users. That include genre, themes, rating, plot overview and cast. But often the metadata is incomplete, inconsistence. Sometimes it just stays vague. Which affect the user experience and recommendation accuracy.

A report from 2023 on content integrity disclosed that Amazon Prime Video received over 10,000 metadata related complaints in a single year. 60% of this complaints are regarding the mislabelling and unordered episodes . Concerns regarding genre inconsistency stand out especially. A movie labelled as Thriller on one platform and Drama on another that left bad taste in user experience and reduce overall confidence in cross-platform recommendation. The issue get more worse when hybrid genre content is introduced like anime where cultural nuances results to wrong labels. If there is no standard semantic structure a smaller inconsistency in tagging can lead to major error in recommendation system.

Additionally, this recommendation models function as black box. Which means they often lack explainability of the decision they make. Even though they are good at recognizing user viewing pattern but they lack to explain their choice and often give vague reasons like "Because you watched X movie you might also like" without providing insights on their inference and decision. Research indicates that people like clear, explained suggestions better. They are more likely to watch something if they understand why it came up [5]. The lack of explainable AI is not only a ease of use issue but also there are biases in the system that can prevent diversity in content and feed the content which is already popular. Diminishing smaller voices.

Mislabelling not only bad for user experience but it can also have some serious consequence on the streaming platform reputation. Suggesting adult content to child account have been reported and prompted streaming platforms to redesign safety filters. Take Netflix as an example. They made the necessary update in kids profile system which filter and block titles with mature themes. This indicate the need of rule based, logic enforced control at a detailed level. However, system based on basic tags or reports have fallen short and are unable to handle the subtle links between contextual suitability and content relationship.

Summary Table – Metadata Issues

Problem Type	Example	Impact
Genre inconsistency	Thriller vs. Drama across platforms	Inaccurate recommendations
Maturity mismatch	Age-inapropriate show in Kids profile	Trust, safety issues
Misordered episodes	Anime episodes out of sequence	Viewing disruption
Vague explanations	"Because you watched *X*"	Low trust, poor transparency

Using ontologies for knowledge representation gives a compelling solution to these problems. An ontology based model is more structured model of a domain, representing entities like Genre, Movie, Viewer and their relationships in a machine understandable format. This differ from ad-hoc metadata schema, ontologies make use of class hierarchies, disjoint class constraints, domain/range restrictions and logical rules to maintain semantic consistency. For Example, a well structured ontology can specify that Thriller falls into the subclass category of Drama. Or it could have disjoint axioms like if content is rated 18+ then never be recommended to user under certain age, which can validated through automatic reasoning in ontologies.

Ontologies also support logical inference. By coding relationships between different objects. Such as a show can share actors from previously liked content, being part of a franchise or being a direct sequel. A reasoner can infer new facts and sharpens recommendation without the need of additional training data. This helps a lot in a cold starting a new user profile where the user history is thin or metadata is missing. Furthermore, ontologies allowed automatic reasoning which can help in detecting contradiction present in the metadata for instance a title is both tagged as a kids and Adult at once. Which can be missed in model based on statistic.

For current advance models, ontology works alongside with current LLM based AI tools to leverage the potential of both stats based learning and symbolic learning. In a hybrid setup, LLM can used for generate candidate suggestions and ontology for filtering, validation and explanation. Recent studies show for such neuro-symbolic approaches integrating symbolic reasoning with machine learning to improve both performance and interpretability [6].

In conclusion, lack of transparency, unreliable and inconsistent recommendation problems are present in today's streaming services recommendation model. Modelling based on ontologies offers semantic clarity, logical consistency, a framework for safe and explainable AI and hybrid-integrated recommendation systems. Its use can significantly increase user trust, metadata integrity and more personalized suggestion across different digital platforms.

III. *Knowledge and Data for Ontology Design*

A. Real-World Metadata Sources for Streaming Content

Building a domain specific ontology model for content recommendation needs to be integrated with the structured real world metadata. The metadata is Widley available on streaming platform like Netflix and IMDB which cover all aspects of audiovisual content. Netflix uses a detailed micro genre taxonomy with more than 76,000 unique genre tags created by human and algorithms [7]. The data store property like setting, theme, tone and audience, enabling a fine grained way to categorized for semantic modelling. Similarly, IMDB also store data like cast, genre , release date and movie. It has over 10 millions titles and more than 14 million contributor [8]. Both of this schemas provide the basis to create a ontology based model by defining class like Movie or Series, relationship like hasActor, hasGenre and data property as release Year.

For enriching specific domains, especially niche types like anime, sites like MyAnimeList act a community driven repositories. MyAnimeList contains over 26,000 anime entries which include and not limited to genre, demographic, rating, themes, demographic label (shounen, seinen), franchise association or episodic order [9]. This data helps specialize the ontology for subdomains which is often miss in Western schemas.

B. Domain Knowledge in Ontology Structuring

A semantic ontology design has to reflect the domain knowledge in content

classification and user interactions. Genre hierarchies show this clearly. Broad genres like Action or Comedy get broken into subgenres such as Martial Arts or Rom Com. Netflix creates composite altgenres like Feel good Romantic African American Comedies. These layer tone, demographics, and narrative elements in complex ways. Such examples show the need for a flexible hierarchical genre taxonomy in the ontology.

User interaction data in relationships like hasWatched or likesGenre has allowed for personalization. Netflix integrate both user interaction data like viewing history and content data genre, cast or director to create more refined recommendation system [10]. Capturing the same for ontology required modelling both user and their relation to media content enabling reasoning task like collaborative filtering.

Domain practices lead to logical constraints and axioms. Movie and Series need to be disjoint classes. This reflects their exclusive formats. Properties like hasReleaseDate work as functional. They ensure one release date per item. Structural relations such as Episode partOfSeries include cardinality constraints. These keep the semantics valid.

C. *Ontological Components. Classes, Properties, and Axioms*

The following table summarizes representative ontology components:

Ontology Component	Type	Domain	Range
Movie	Class	–	–
Series	Class	–	–
Genre	Class	–	–
Actor	Class	–	–
Franchise	Class	–	–
hasGenre	Object Property	Movie or Series	Genre
hasActor	Object Property	Movie or Series	Actor
partOfFranchise	Object Property	Movie or Series	Franchise
hasEpisode	Object Property	Series	Episode
hasReleaseDate	Data Property	Movie or Series	xsd:date (ISO date)
hasRating	Data Property	Movie or Series	xsd:decimal (numeric)

Table 1 shows key classes and properties in the streaming ontology.

These attributes allow integration of both semantic constraints and metadata. Object properties link class instances (e.g., Movie to Genre), while data properties attach literals (e.g., ratings). Declarative constraints, such as disjointness and cardinality axioms, support validation and enable reasoning.

IV. *Ontology Construction Methodology*

A. *Specification*

The First and foremost step required to create any ontology model is defining domain and scope. As mentioned earlier, the domain for the ontology is streaming content metadata. That cover series, movies, genres and user preferences in particular. We will create competency questions such as “ "Which movies should be recommended to a user who likes a certain genre or director?”. This approach helped shape the ontology requirements specification. In turn, it ensured the model would tackle the key reasoning goals pretty much right from the start.

B. *Conceptualization*

During this phase, I developed a conceptual model by defining major classes like Series, Movie, User, Actor and Genre and their relationship with each other. I employed both top down approach which is going from more generalised to more specific property for

example content and bottom up approach which means from specific to working towards more generalised property for example RecommendationContext. Some relationship like hasActor, hasGenre, hasRating and directedBy were came from real world metadata such as IMDB schemas. And to maintain standardisation we followed the best practice such as using nouns for classes and verbs for relationship.

C. *Formalization*

Formalization meant encoding our model using OWL 2 DL. In this we add axioms and logical constraints like class disjointness for example Movie and Series are disjoint. We also define property domains and ranges and cardinality rules such as, each Episode must belong to exactly one series. We avoid creating to specific or over expressive axioms to allow our model to maintain decidability and efficient reasoning, making a path way for the description logic reasoner to do automatic reasoning.

D. *Implementation*

The implementation of the ontology is done by using Protégé 5.x and I used OWL 2 for modelling. I preferred HermiT as a reasoning engine which help in checking logical consistency and infer new relationship. OntoGraf provided visualization for classes and properties. SPARQ 1.1 is used for querying.

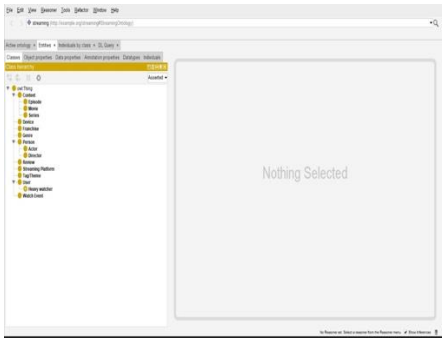


Figure 4.1: Class hierarchy in Protégé before running the reasoner. This view reflects the structure of classes like Movie, Series, Genre, and User as initially modeled.

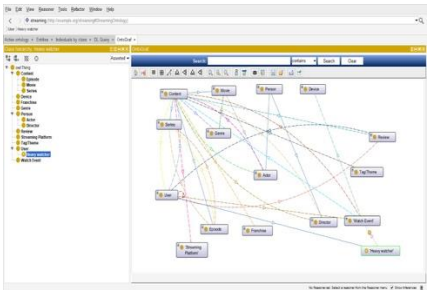


Figure 4.2: OntoGraf visualization of object properties such as hasGenre, hasActor, and directedBy between main classes.

I also tested the model using real world scenarios for example retrieving all action movies directed by a user preferred director then I analyse the output for reasoning and adjusted accordingly in each iteration.

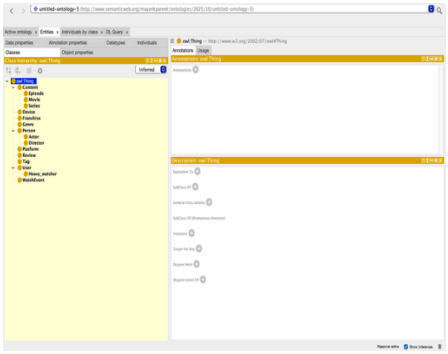


Figure 4.3: Inferred class hierarchy after running the HermiT reasoner. This confirmed the disjointness and subclass relationships defined in the formalization step.

Technology	Role
Protégé 5.x	GUI-based ontology editor for modeling OWL classes, properties, and axioms
OWL 2 DL	Language for formal ontology specification with logical consistency
HermiT Reasoner	OWL DL reasoner for inference and consistency checks
OntoGraf Plugin	Visualization of class/property graphs within Protégé
SPARQL 1.1	Semantic query language used to test ontology reasoning and retrieval

E. *Challenges and Trade-offs*

As I mentioned in Formalization we need to create axioms and logical constrains but balancing expressivity versus performance was a challenge. Models

that are highly expressive can capture rich semantics but at the same time at makes reasoning computationally expensive. We tackle this challenge by optimizing performance by limiting OWL constructs to those which was must needed for our recommendation system. Reusability versus specificity was another challenge I faced, importing external model bring in some unnecessary complexity which was not required. So we opt for partial reuse and customization. We also analysed the trade-off between automated and manual ontology construction. Tools based on large language models could suggest taxonomies to get started. Manual verification remained crucial, though, to match OWL syntax and semantics. The final model underwent iterative testing, debugging, and alignment with recommendation goals. That included integration with large language models too.

V. *Ontology Evaluation and Case Study Analysis*

After creating our model the validation of the ontology really matters to make sure everything is logical , consistent and work for what we have planned. For validating and evaluating our model we made use of reasoning and SPARQL querying using Protégé 5.x which has HermiT as a reasoning engine and support SPARQL plugin. I also added screenshots from the Protégé application to show the inferences and query results visually.

A. *Reasoning with HermiT*

After the implementation of our ontology model in Protégé 5.x, we made use of its reasoning engine HermiT to verify consistency and check for class hierarchies. Our model passed the logical consistency validation there were no contradiction in our logic it was sound and now it is validated by the engine as well. The class hierarchy that came out from the inference matched what we expected, relationship among subclass and domain axioms works as intended.

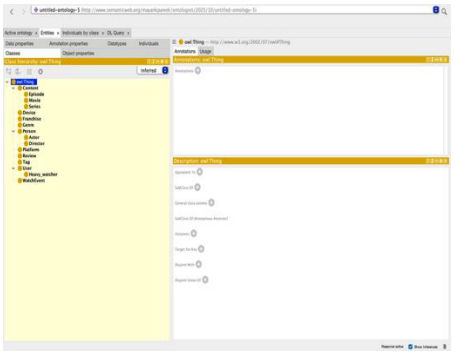


Figure 5.1. Inferred class hierarchy in Protégé after HermiT reasoning. Movie, Series, Genre, Actor, and User classes were validated and auto-classified under owl:Thing. That confirmed the structure was solid.

HermiT also validated class disjointness for example Movie and Series are disjoint and also checked the functional properties of class for example hasReleaseDate which we created in our model formalization phase. The inferred links and connections showed the ontology could handle content-based recommendations. Things like spotting preferences for certain genres or actors came through clearly.

B. *SPARQL Query Demonstrations*

To test our model for some real world queries derived from our competency question, we made use of the SPARQL 1.1 plugin for Protégé.

Query 1. Retrieve all movies of a specific genre, like Action.

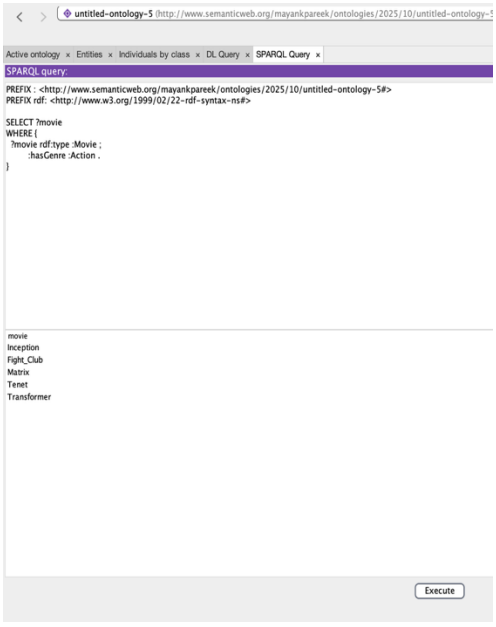


Figure 5.2. SPARQL query and result showing all Movie instances linked to the Action genre via hasGenre.

Query 2. List all actors in a specific movie, like Inception.

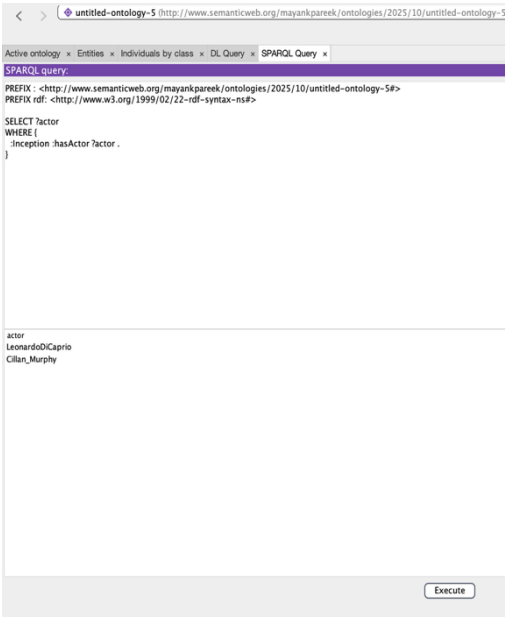


Figure 5.3. Query returning all individuals of class Actor associated with the movie Inception via hasActor property.

Query 3. Recommend movies to a user interested in a certain genre.

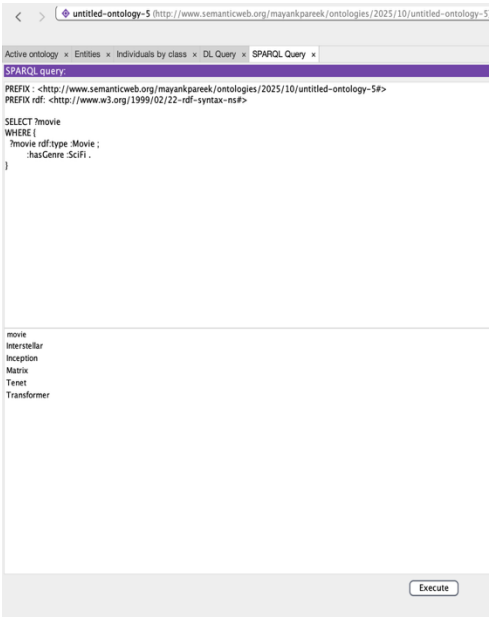


Figure 5.4. Results of recommending movies in the SciFi genre. This simulated a basic recommendation function.

Every SPARQL query mentioned above was executed successfully and the outcome matched the data instances we built into the ontology. These tests proved the ontology supports real-time recommendation logic. It draws on user likes and content details in a reliable way.

C. Summary and Tool Support

Altough protege app is available to use on web browser, But using Webprotege has its limitation. It does not handle reasoning and doesn't have SPARQL plugin. So, I used the Protege desktop version to create our ontology model and preforming evaluation and test. I loaded the ontology file in OWL 2 format right on the local machine. Plugins let us run both the reasoning and SPARQL checks smoothly.

VI. Ontology and Large Language Models: Complementarity, Challenges, and Integration

In today's modern AI model the Ontology and Large Language Model goes hand in hand. Which means both technology complement each other. On the one hand, Ontologies used for a better formally structured knowledge and clear semantics which result in automatic reasoning and logical consistency. On the other hand, LLM has its own usefulness, LLM model are really good in pattern recognition and to parse natural language and synthesis the natural language. Both of the model has their advantage but they also lack and have their drawbacks for example Ontology based is not good in parsing complex query, using general knowledge. Similarly, LLM has its own challenges like LLM is often hallucinate and doesn't maintain consistency and logic. Also LLM model lack transparency which affect the principle of explainable AI. Although, both the model has their positive and drawbacks when used one model at a time, if we combine both the model and took a hybrid approach they complement each other positives and address each other's limitation as a result enhance task like recommendation, classification and semantic search.

Ontology	vs	LLMs
Formal semantics		Language understanding
Logical consistency		Handles ambiguity
Reasoning with symbols		Unstructured knowledge
Explainability		Contextual responses

Figure 1: Capability comparison between ontology-based systems and LLMs in content recommendation contexts.

A hybrid model comes from integrating ontological knowledge into large language model pipeline. Think about it in this way, LLMs are good in parsing complex query and can generate candidate metadata, suggesting possible genre and actor links. Then ontology model take the parse query from LLM as a form of SPARQL and validate the LLM suggestion and restricting the unnecessary link and provide extra knowledge using automatic reasoning. This hand in hand work of both models works best in area like content recommendation, where model can take advantage of both user interaction and sematic relationships for example "Movie A is part of Franchise B" or "User C frequently watches Sci-Fi titles directed by Director D" [11].

When the task is related to reasoning, ontologies based model are far superior then the LLM model because of the descriptive logics for instance deducing that if a user likes “Superhero Films”, they are more likely to watch the move which come under “Action” genre because of class subsumption. These type reasoning is beyond capabilities of the pattern based learning of LLM. On contrast, when the task is related to picking up new trend based on user interaction, the LLM model shines they can detect emerging genre or audience sentiment trends that can later useful in encoding into the ontology structure [12].

So far we talked about the pitfall of both model and how by integrating the model we can negate them. But the integration of this model is not a piece of cake there are many challenges. First major concern is about consistency, because of hallucination LLM model can sometimes generate conflicting and fuzzy logic that violate ontology axioms. For example, naming a movie both a “Standalone Film” and a “Series” breaks class disjointness axioms. Another concern is regarding the LLM probabilistic nature combining with deterministic nature of ontology. To deal with this there is a need of mediation layers such as confidence scoring or rule based checks. These design choices involve trade-offs between flexibility and correctness.

Scalability is the another challenge, building of ontology model required a domain expert or domain knowledge and require manual creation of many axioms, class and properties. Which is time consuming. On the other hand, LLMs are fast, dynamic and automated in nature, they pose an issue of hallucinating false fact if not grounded. A way to connect them involves neuro-symbolic methods, embedding ontological knowledge into model training or using knowledge graphs as memory for LLM agents. In these setups, the ontology serves as a semantic filter. It vets large language model replies for relevance and logic.

In our case study of a content streaming platform, integrating both LLM and ontology model means combining a well-structured ontology (classes like, Movie, Genre, Actor, User) with LLM which can make use of user review or suggest new genre tags. In real world scenario, the process will have this sort of follow, for user complex query parse, LLM model can use to understand it, detect new patter, list some suggestion and pass the user query in a SPARQL format to ontologies. Then ontologies can make use of constraints and automatic reasoning to filter out the exact suggestion which make user recommendation better with proper explanation. After this again LLM model can use to generate natural language output with the suggestion and explanation.

Ultimately, integrating LLMs and ontologies can work as bridge towards more explainable, intelligent and robust AI systems. Specifically in domain which required structured knowledge and deep human context.

VII. Discussion and Conclusion

Ontologies in the recent year has emerged as a powerful tools for structuring metadata, changing distributed individual attributes to semantically rich , machine interpretable knowledge. This project is showed the capabilities of ontology designed specifically for streaming metadata can support not only organization but also automatic intelligence reasoning and personalization. Ontologies

differ from the conventical database like table, it has much more depth of understanding through the use of links between elements and setups of class hierarchies. Consider a detailed query such as list unseen comedies from the 1990s that star a particular actor. Such requests work well without needing to code rigid rules ahead of time [13].

We implemented our ontology model using OWL 2 DL and Protege which enabled us to create a solid framework supported by logical axioms. Axioms like disjointness and cardinality restrictions allowed our model to remain cohesive and consistent. Reasoning engine HermiT helped in inferencing, logical consistency and checked relationships between subclasses. while SPARQL allowed retrieval of semantic patterns directly from the ontology [14]. All these elements form the base for practical applications in domain of recommendation systems, catalog browsing and content tagging.

Table 1 gives an overview of the main features in the ontology. It also covers how those features affect finding content, doing reasoning, and making things extensible.

Ontology Feature	Practical Benefit
Hierarchical Taxonomy	Enables micro-genres and fine-grained recommendations
Semantic Relationships	Supports rich queries and unifies metadata from diverse sources
DL Reasoner & Constraints	Ensures logical consistency and enables inferred classifications
SPARQL Query Capability	Empowers semantic search for content discovery and recommendation
Modularity and Reusability	Facilitates extension and links to external vocabularies like Wikidata

Table 1: Key Features of the Streaming Content Ontology and Their Benefits

In future, integrating this ontology model with the AI model has significant potential. Recommender systems often run into problems with cold starts. That happens a lot with new users or content that's more niche. Adding ontological knowledge to these systems can help a great deal. Think of genre setups, connections between directors and actors, or matches in themes. Recommendations then come from how things relate semantically. They do not depend only on what a user has done before. The same idea applies to pairing the ontology with large language models through methods that augment retrieval. That could make answers more accurate. It might cut down on made-up information too. Imagine a helper for streaming that asks the ontology about sci-fi films a user has not seen yet. It could focus on ones with a favoured actor. Then it delivers solid ideas in everyday language.

In conclusion, Ontology based model allow streaming platform to offer personalized, explainable and context aware experiences. The key takeaways was balancing expressivity and tractability. With further development, this model could act as base for intelligent, scalable streaming ecosystems capable of adapting to diverse user needs and evolving content landscapes.

VIII. Reference:

[1] E. M. Bender and A. Koller, "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data," Proc. ACL, 2020.

[2] T. R. Gruber, "A translation approach to portable ontology specifications," Knowledge Acquisition, vol. 5, no. 2, pp. 199–220, 1993.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," Scientific American, vol. 284, no. 5, pp. 28–37, 2001.

[4] C. A. Gómez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," ACM Trans. Management Information Systems, vol. 6, no. 4, pp. 1–19, 2016.

[5] M. Tintarev and J. Masthoff, "Designing and Evaluating Explanations for Recommender Systems," Recommender Systems Handbook, Springer, 2011.

[6] S. Zhang, L. Yao, and A. Sun, "Deep Learning Based Recommender System: A Survey and New Perspectives," ACM Comput. Surv., vol. 52, no. 1, pp. 1–38, 2019.

[7] A. Madrigal, "How Netflix Reverse-Engineered Hollywood," The Atlantic, 2014.

[8] IMDb Datasets. [Online]. Available: https://developer.imdb.com/

[9] Wikipedia, “MyAnimeList,” 2024. [Online]. Available: https://en.wikipedia.org/wiki/MyAnimeList

[10] Netflix Help Center. "How does Netflix decide what to recommend to me?" [Online]. Available: https://help.netflix.com/en/node/100639

[11] G. Antoniou and F. van Harmelen, A Semantic Web Primer, MIT Press, 2nd ed., 2008.

[12] P. Hitzler, M. Krötzsch, S. Rudolph, Foundations of Semantic Web Technologies, CRC Press, 2009.

[13] Netflix Tech Blog, “The Netflix Recommender System: Algorithms, Business Value, and Innovation,” 2016. Same as [4]

[14] H. Knublauch et al., “The Protégé OWL Plugin,” Stanford University, 2004.

Appendix A: OWL/Turtle Representation of the Streaming Content Ontology

```
@prefix :
<http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base
<http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5/> .

<http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5> rdf:type owl:Ontology .

#################################################################
#    Datatypes
#################################################################

###  http://www.w3.org/2001/XMLSchema#gYear
xsd:gYear rdf:type rdfs:Datatype .


#################################################################
#    Object Properties
#################################################################

###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#actedIn
:actedIn rdf:type owl:ObjectProperty ;
         owl:inverseOf :hasActor ;
         rdfs:domain :Actor ;
         rdfs:range :Content .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#directedBy
:directedBy rdf:type owl:ObjectProperty ;
            rdfs:domain :Content ;
            rdfs:range :Director .
```

```
###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#hasActor
:hasActor rdf:type owl:ObjectProperty ;
          rdfs:domain :Content ;
          rdfs:range :Actor .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#hasEpisode
:hasEpisode rdf:type owl:ObjectProperty ;
            owl:inverseOf :partOfSeries ;
            rdfs:domain :Series ;
            rdfs:range :Episode .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#hasGenre
:hasGenre rdf:type owl:ObjectProperty ;
          rdfs:domain :Content ;
          rdfs:range :Genre .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#partOfSeries
:partOfSeries rdf:type owl:ObjectProperty ;
              rdfs:domain :Episode ;
              rdfs:range :Series .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#watched
:watched rdf:type owl:ObjectProperty ;
         owl:inverseOf :watchedBy ;
         rdfs:domain :User ;
         rdfs:range :Content .
```

```
###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#watchedBy
:watchedBy rdf:type owl:ObjectProperty ;
           rdfs:domain :Content ;
           rdfs:range :User .


#################################################################
#    Data properties
#################################################################

###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#hasName
:hasName rdf:type owl:DatatypeProperty ;
         rdfs:domain :Device ,
                     :Genre ,
                     :Person ;
         rdfs:range xsd:string .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#hasReleaseYear
:hasReleaseYear rdf:type owl:DatatypeProperty ;
                rdfs:domain :Content ;
                rdfs:range xsd:gYear .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#hasTitle
:hasTitle rdf:type owl:DatatypeProperty ;
          rdfs:domain :Content ;
          rdfs:range xsd:string .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#hasUserId
:hasUserId rdf:type owl:DatatypeProperty ;
           rdfs:domain :User ;
           rdfs:range xsd:string .
```

```
#################################################################
#    Classes
#################################################################

###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Actor
:Actor rdf:type owl:Class ;
       rdfs:subClassOf :Person .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Content
:Content rdf:type owl:Class .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Device
:Device rdf:type owl:Class .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Director
:Director rdf:type owl:Class ;
          rdfs:subClassOf :Person .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Episode
:Episode rdf:type owl:Class ;
         rdfs:subClassOf :Content ,
                         [ rdf:type owl:Restriction ;
                           owl:onProperty :partOfSeries ;
                           owl:qualifiedCardinality
"1"^^xsd:nonNegativeInteger ;
                           owl:onClass :Series
                         ] .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Franchise
```

```
:Franchise rdf:type owl:Class .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Genre
:Genre rdf:type owl:Class .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Heavy_watcher
:Heavy_watcher rdf:type owl:Class ;
               rdfs:subClassOf :User .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Movie
:Movie rdf:type owl:Class ;
       rdfs:subClassOf :Content ,
                       [ rdf:type owl:Restriction ;
                         owl:onProperty :hasGenre ;
                         owl:someValuesFrom :Genre
                       ] .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Person
:Person rdf:type owl:Class .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Platform
:Platform rdf:type owl:Class .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Review
:Review rdf:type owl:Class .
```

```
###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Series
:Series rdf:type owl:Class ;
        rdfs:subClassOf :Content .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Tag
:Tag rdf:type owl:Class .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#User
:User rdf:type owl:Class .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#WatchEvent
:WatchEvent rdf:type owl:Class .


#################################################################
#    Individuals
#################################################################

###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#ChristopherNolan
:ChristopherNolan rdf:type owl:NamedIndividual ,
                           :Director .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Cillan_Murphy
:Cillan_Murphy rdf:type owl:NamedIndividual ,
                        :Actor ;
               :actedIn :Inception .
```

```
###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Dark
:Dark rdf:type owl:NamedIndividual ,
               :Series .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Fight_Club
:Fight_Club rdf:type owl:NamedIndividual ,
                     :Movie ;
             :hasGenre :Action .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Inception
:Inception rdf:type owl:NamedIndividual ,
                    :Movie ;
           :directedBy :ChristopherNolan ;
           :hasActor :Cillan_Murphy ,
                     :LeonardoDiCaprio ;
           :hasGenre :Action ,
                     :SciFi ;
           :hasTitle "Inception" .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Interstellar
:Interstellar rdf:type owl:NamedIndividual ,
                       :Movie ;
              :hasGenre :SciFi ;
              :hasTitle "Interstellar" .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#LeonardoDiCaprio
:LeonardoDiCaprio rdf:type owl:NamedIndividual ,
                           :Actor .
```

```
###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Matrix
:Matrix rdf:type owl:NamedIndividual ,
                 :Movie ;
        :hasGenre :Action ,
                  :SciFi .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#SciFi
:SciFi rdf:type owl:NamedIndividual ,
               :Genre .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Tenet
:Tenet rdf:type owl:NamedIndividual ,
               :Movie ;
       :hasGenre :Action ,
                 :SciFi .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#Transformer
:Transformer rdf:type owl:NamedIndividual ,
                      :Movie ;
             :hasGenre :Action ,
                       :SciFi .


###
http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-
ontology-5#User123
:User123 rdf:type owl:NamedIndividual ,
                 :Heavy_watcher .


###  Generated by the OWL API (version 4.5.29.2024-05-13T12:11:03Z)
https://github.com/owlcs/owlapi
```

Appendix B: SPARQL Queries Used for Ontology Evaluation

```
-- Query 1: Retrieve all movies associated with the Action genre
PREFIX : <http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-ontology-5#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?movie
WHERE {
  ?movie rdf:type :Movie ;
         :hasGenre :Action .
}
-- Query 2: Retrieve all actors associated with the movie Inception
PREFIX : <http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-ontology-5#>

SELECT ?actor
WHERE {
  :Inception :hasActor ?actor .
}
-- Query 3: Retrieve all Science Fiction movies (content-based recommendation)
PREFIX : <http://www.semanticweb.org/mayankpareek/ontologies/2025/10/untitled-ontology-5#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
SELECT ?movie
WHERE {
  ?movie rdf:type :Movie ;
         :hasGenre :SciFi .
}
```