Knowledge Engineering - Final Assignment

Vlad Stan, Karl Lundfall, Wenjie Zhong

January 26, 2015

1 Overview

In this document for the course Knowledge Engineering (2014-2015) at Vrije Universiteit, Amsterdam, we will present a knowledge-based approach for creating a movie recommendation system. We will attempt a novel approach in solving a problem with which a large range of companies struggle on a massive scale.

For the majority of the presented content, the material will be presented in tables providing a clear and structured way of moving towards an implementation of a movie suggestion system.

2 Context analysis

We will start from a broad perspective, keeping the existing movie recommendation systems in mind as an organization. In particular, we have looked more closely on the service provided by Netflix[1].

[1] http://www.netflix.com/

Organization Model	**Problems and Opportunities Worksheet OM-1**
PROBLEMS AND OPPORTUNITIES	Modern recommendation systems as of today are facing a plethora of challenges. Traditional systems limit themselves to merely suggesting similar movies to what the user previously likes, but modern systems need to be able to do more than that [a][b]. Nowadays the issues can include: • **Diversity:** The system should provide the user with a variety of suggestions in order to be flexible and interesting. • **Exploratory:** The system should serve as a platform for users exploring new terrains that they aren't familiar with but would like if they tried. • **Explanatory:** The system should be able to explain the underlying rationale behind a recommendation. • **Social:** The system should be able to take into account what the user's friends like. • **Freshness:** The system should be able to make recommendations that seem fresh, which could be done by suggesting content that was recently released. • **Genre-specific:** The system should be able to handle both broad (traditional) genres and narrow genres (such as *Imaginative Time Travel Movies from the 1980s*). These challenges are still being struggled with, and the method remains the same: by means of statistical analysis or Machine Learning-based. This creates the constraint that the system has to be used extensively in order to provide adequate recommendations. The following are opportunities new recommendation systems can take advantage of: • No high cost of entry into the market. • More extensively utilize human expert knowledge in the domain. • Make use of semantic technologies and databases already existing, compiling movie knowledge and making connections and complex queries easier than in traditional databases.
ORGANIZATIONAL CONTEXT	**Mission, vision and goals of the organization** • Movie recommendation systems want to be appreciated by users, therefore adding market value to their platforms. **Important external factors the organization has to deal with** • The algorithms cannot be too slow (or computationally expensive). • The recommendation system must be able to adapt to different situations and different types of users with different expectations. • The format of the recommendations must be taken into account, for example: How many recommendations should be given? • The system should not be too intrusive on private information, or give an impression of privacy intrusion. • It is important to have a very big database of content in order to please every type of customer. **Strategy of the organization** • Providing as good watching experience as possible, when following the advice of the system. **Its value chain and the major value drivers** • The main concern of many retailers in this domain nowadays is how to improve on their recommendation system, in order to get an edge on the market competition. Netflix even announced a $1 million prize for a better suggestion algorithm in 2006.
SOLUTIONS	**A recommendation system that uses human expert knowledge as a basis for its decisions.** This will ameliorate explanations, and increase the trust in the system. Furthermore, the awareness that the recommendation is based on knowledge from a human expert will provide an exotic touch to the system. In an expert-based system, user statistics don't have to be shared among other users, hence improving on privacy issues.

[a] http://research.microsoft.com/pubs/115396/EvaluationMetrics.TR.pdf
[b] http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

Organization Model	Variant Aspects Worksheet OM-2
STRUCTURE	See figure 1.
PROCESS	Modern recommendation algorithms are typically complex and use different methods. The current system from Netflix uses the Matrix Factorization algorithm (also known as SVD, Singular Value Decomposition) and Restricted Boltzmann Machines (RBM), among other Machine Learning algorithms, in order to make good recommendations. The machine learning algorithms lie at the heart of the process, but there are other components as well. See figure 2 for an architecture over the recommendation system provided by Netflix.
PEOPLE	• The back-end developers of the system. • The front-end developers of the system. • Customers. • Content owners (e.g. copyright holders of movies).
RESOURCES	**Information systems** • User profiling system. • (Optional) Connected social networks. • (Optional) Streaming or downloading service for movies. • The recommendation algorithm. **Equipment and materials** • Server machines. **Patents** • Software licenses. • Film copyrights.
KNOWLEDGE	• User ratings history and/or previously seen films. • User genre preferences. • Friend connections between users. • Information on the popularity of content.
CULTURE & POWER	• Netflix is a content network business with highly advanced technical capabilities. The disregard and loathing existent between the content and technology industries is not present in Netflix.

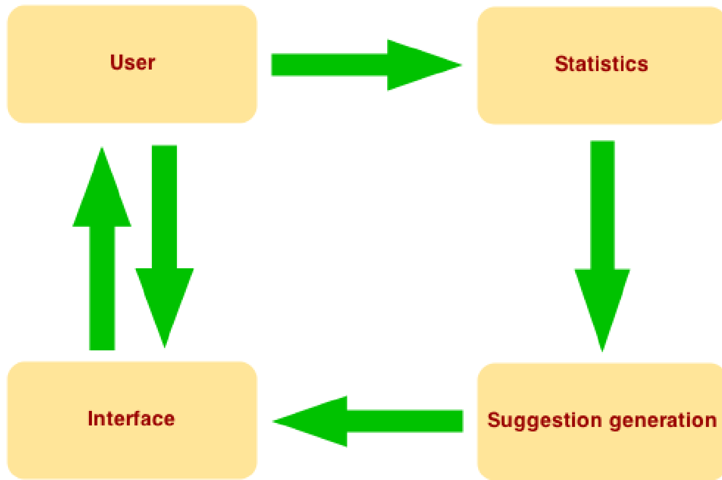

Figure 1: The structure of the organization.

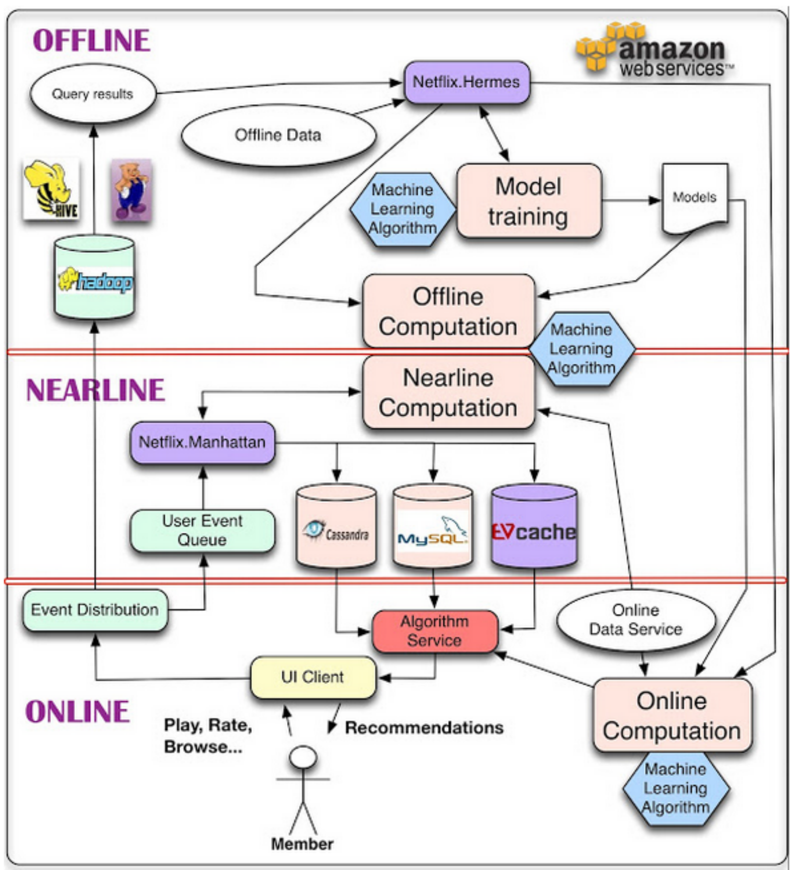

Figure 2: The architecture for the Netflix recommendation system.

Organization Model	Process Breakdown Worksheet OM-3					
NO.	TASK	PERFORMED BY	WHERE?	KNOWLEDGE ASSET	INTENSIVE?	SIGNIFICANCE
1	Research and development of the recommendation algorithm.	Back-end developer	Office	• Statistics. • Machine Learning. • Programming. • Specific knowledge of the algorithm.	Yes	5
2	Maintenance and upgrading of user interface	Front-end developer	Office	• Design patterns. • Web design knowledge. • Human computer interaction. • Basic understanding of the back-end.	No	4
3	Maintenance and upgrading of user profile system	Back-end developer	Office	• Design patterns. • Knowledge of the relevant database system. • Social networking knowledge. • Programming.	No	3
4	Maintenance and upgrading of movie database system.	Back-end developer	Office	• Design patterns. • Knowledge of the relevant database system. • General understanding of movie labeling. • Programming.	No	3

Organization Model		Knowledge Assets Worksheet OM-4				
KNOWLEDGE ASSET	POSSESSED BY	USED IN	RIGHT FORM?	RIGHT PLACE?	RIGHT TIME?	RIGHT QUALITY?
Statistics	Back-end developer	1	Yes. Statistics knowledge are usually in the back of the head of many engineers and in text-books for further reference	No. Our hypothesis is that recommendation algorithms nowadays are overemphasized in terms of statistics.	N/A.	Yes; The statistics knowledge in modern algorithms is at many times very elaborate and sophisticated.
Machine Learning	Back-end Developer	1	Yes, the state of the art Machine Learning tools are well documented.	No. We want to make a case for developing an algorithm without using Machine Learning knowledge.	N/A	N/A
Programming	Back-end Developer	1,3,4	Yes. Developers are experienced programmers and usually know about most necessary paradigms in order to make high quality code.	Yes. Programming knowledge is essential in developing software.	N/A	Yes. Modern enterprise software developers are skillful and precise.
Specific knowledge of the algorithm	Back-end developer	1	Yes. The algorithms should be documented with appropriate tools.	Yes. It is important to have sufficient understanding of the algorithm and its implementation in order to understand and maintain the code.	N/A	Yes, given that the algorithms used are sufficiently documented.
Knowledge of the relevant database system	Back-end Developer	3,4	Yes. Database systems have documentation publicly available online.	Yes. Having a good understanding of the database systems helps improve statistics quality.	N/A.	Yes. Database systems are usually well-documented.

Knowledge asset	Possessed by	Used in	Right form?	Right place?	Right time?	Right quality?
Social networking knowledge	Front-end Developer	3	Yes, documentation and history of online social networking is widely available	Yes. Social networking skills are getting increasingly important in most organizations.	N/A	Yes, very detailed information and experiments are available regarding social networking
General understanding of movie labeling.	Back-end Developer	4	No, it is currently not accounted for enough and should be provided by experts in the movie business.	No, companies use too much statistical information and not enough expert information.	N/A	No (see previous columns).
Design patterns.	Front-end developer.	2,3,4	Yes, design patterns are available online.	Yes. Design patterns are important in order to develop intuitive interfaces.	N/A.	Yes, interfaces and back-end systems of modern recommendation systems follow most design principles.
Human computer interaction	Front-end Developer	2	Yes, human computer interaction is taught in universities and documentation is widely available online.	Yes.	N/A.	Yes, due to availability in universities, quality is constantly improved through r & d.
Basic understanding of the back-end.	Front-end developer.	2	Yes.	Yes. Without an understanding of the underlying back-end it is hard to provide an adequate explanation for the recommendations.	N/A.	Yes, interfaces of modern recommendation systems follow most design principles

Task Model	**Task Analysis Worksheet TM-1**
TASK	Research and development of the standard recommendation algorithm.
ORGANIZATION	This task is carried out by the back-end developers in the research and development department.
GOAL AND VALUE	The goal of the task is to design a better solution than the available movie recommendation algorithms in the market. In this way clients can have a better experience with the product and spend more time with the software service. Once our product is proven to work, finding an appropriate buyer would be the next step.
DEPENDENCY AND FLOW	Input comes from the user profile and their viewing history, information that can be collected from their input. So it is important the visitor is logged into the service as a user. Then, the input variables have to be aggregated and presented to the algorithm, so that an appropriate movie can be recommended to the user.
OBJECTS HANDLED	There are four main input items that are needed when carrying out this procedure: • A setting wherein the user is viewing the movie. • The user's history of seen movies. • Preset rules deduced from expert knowledge. • A movie database expressed with semantic technology. The recommended movie is the output of this task, afterwards the recommended movie is presented to the user with relevant information. If the user declines the recommendation, the system will try again.
TIMING AND CONTROL	This task can only be carried out if the client using the service is known and the mentioned user has a viewing history recorded within the recommendation service. So it is important that the visitor has an existing account with recorded inputs and is logged into the recommendation service. **Preconditions**: Customer preferences and information recorded. **Postcondition**: Movie recommendation presented.
AGENTS	The back-end developer and a domain expert.
KNOWLEDGE AND COMPETENCE	• Statistics. • Semantic technology. • Programming. • Movie expertise. These are all classified as "knowledge intensive" items.
RESOURCES	To create the algorithm based on expert feedback, three back-end developers and one expert are needed for feedback processing and implementation. The developers also need access to computers and the servers where the service is active.
QUALITY AND PERFORMANCE	Some basic quality assurance and bug testing can be performed by the developers, but the main test should be performed by the users. If a majority of users leave the service without finding a fitting movie, the quality of the service is poor.

Task Model	Knowledge Item Worksheet TM-2
NAME POSSESSED BY USED IN DOMAIN	Algorithm for predicting movie preferences Back-end developer Movie prediction systems Computer Science

Nature of the knowledge		**Bottleneck / to be improved?**
Formal, rigorous	X	
Empirical, quantitative		
Heuristic, rules of thumb	X	X
Highly specialized, domain-specific	X	
Experience-based		
Action-based	X	
Incomplete		
Uncertain, may be incorrect		
Quickly changing		
Hard to verify		
Tacit, hard to transfer		
Form of the knowledge		
Mind	X	
Paper		
Electronic	X	X
Action skill		
Other		
Availability of knowledge		
Limitations in time		
Limitations in space		
Limitations in access		
Limitations in quality	X	X
Limitations in form		

Agent Model	Agent Worksheet AM-1
NAME	*Movie expert capable of recommending fitting content.*
ORGANIZATION	Research and Development department.
INVOLVED IN	Create a better solution for the movie recommendation algorithm (task 1)
COMMUNICATES WITH	Back-end developers
KNOWLEDGE	• Understanding the movie industry, different movie types and movie production related concepts. • Knowledgeable in how to recommend movies to people based on their preferences and the nature of their preferred movies.
OTHER COMPETENCES	• Communication skills, in order to communicate with the development team without using jargon.
RESPONSIBILITIES AND CONSTRAINTS	• Keep track of new developments in the movie entertainment industry or movie producing industry that can affect the recommendation service. • Should fully realize that the consequences of giving faulty advice could negatively affect the quality of the recommendation service and damage the reputation of the service.

3 Knowledge Model

3.1 Task template

For the proposed task presented in TM-1, we chose to use the *prediction* task template. Prediction belongs to the *analytic* tasks, and has the following specifications:

- **Input - System data**: The input of the system is the current user history of rated and seen films.
- **Output - System state**: The output of the system is the new movie that the user should watch after the system has applied appropriate knowledge-based reasoning.

The prediction task template was used because we are predicting what the user is going to watch and then rate; this results in a new rating, from which more suggestions can be made in a new round of prediction. There can also be a case in which the prediction failed, when the user does not approve of the suggested movie. This will be explained more in the following section. We call this method *prediction through suggest-and-evaluate*. A task decomposition can be found in figure 3.

3.2 Task Inference Model

The task inference model can be seen in figure 4. The model did not exist as part of literature, therefore we have made our own inference model. These are the inferences used:

- **Abstract:** Starting from a case where the user wants to watch a movie based on current movie setting, the system structures the data and collects previous movie history.

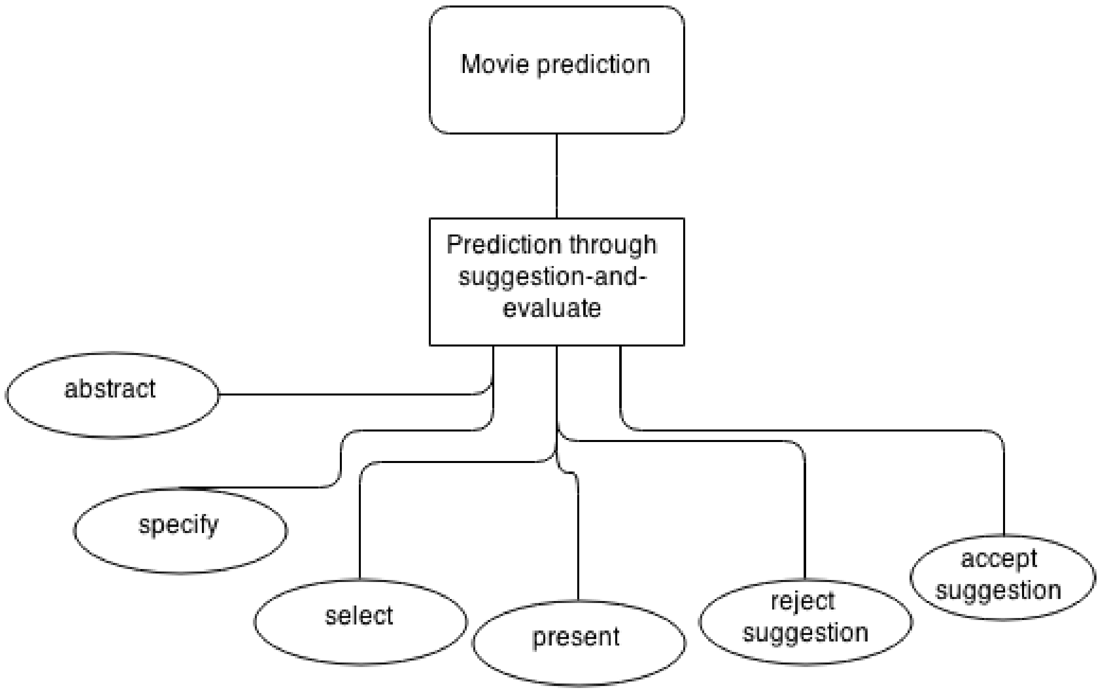

Figure 3: The task breakdown for the prediction with suggest-and-evaluate

- **Specify:** Based on the user history, the system extracts norms derived from this history, following common trends found in the existing data. Norms are preferences in terms of actors, producers, directors or genres.
- **Select:** Based on expert knowledge, these norms are weighted and used to select an appropriate recommendation for the user.
- **Present:** The candidate suggestion is presented to the user.
- **Reject suggestion:** The user rejects the new proposal, which triggers the system to re-evaluate the decision.
- **Accept suggestion:** The user accepts the new proposal, which means that the prediction has been successful.

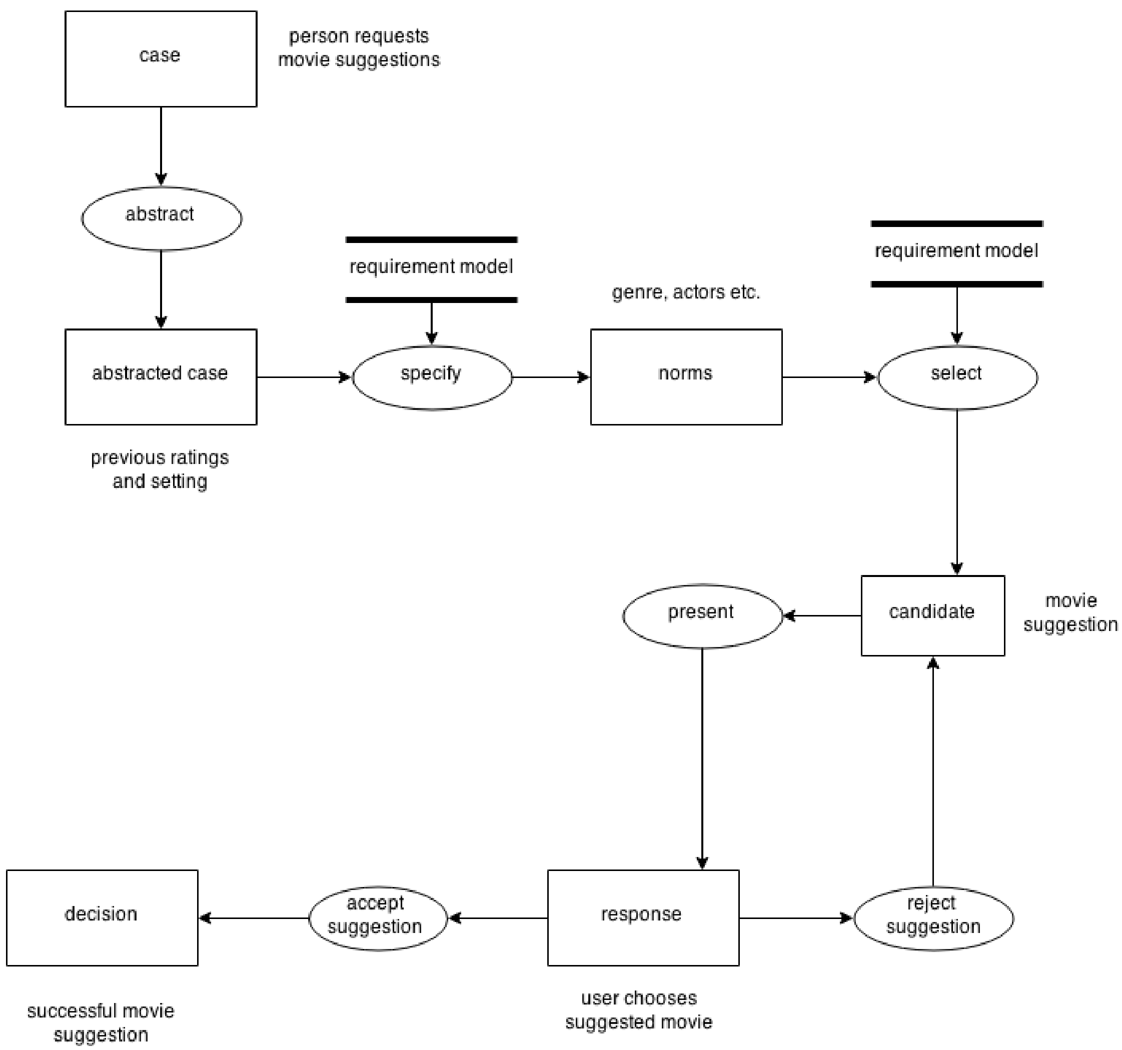

Figure 4: The proposed inference model for prediction.

3.2.1 Domain Schema

Expert

Stefan Hickert,our expert is an editor in training at the "Nederlandse Filmacademie Amsterdam". He worked on several projects, projects like "Adam & Eva", a television show broad casted in the Netherlands in 2014. One of the most important goals of the interview was to understand concepts and highly influential factors contributing to a success film. Another goal was to investigate the reasoning behind existing movie recommendation services, exploring their decisions and flaws. For these two reasons, the interview started with a general explanation of the desired system. Afterwards, a card game was presented, whereby the expert noted some important concepts on cards and organized them. For the second assignment, the expert was asked to organize the cards in another method. The first organization simply described staff concepts like actors, producers and directors. The other group were audience, budget, soundtrack and genre. This information was not extremely informational. The second organization was more helpful. The second organization displayed that the producer is linked with the audience. The expert elaborated that the audience does not really choose the producer, the producer chooses the audience. This is because the producer always makes the same style of movies and therefor attracts the same style of people. Finally, the last part of the interview was the elaboration of Netflix suggestions. There it became apparent that actors and genres play a very important role, but in the eyes of the expert, a director plays the most important role. The director decides how the story is told, perceived and influences how the actor perform. The setting and the style of the movie is greatly dependent on the director. However, genre should be on the first place when presenting a movie recommendation. In descending order, the following jobs are also important: director, actor and producer. The soundtrack variable is not included, because only one name in the industry can make a difference in a movie. The reach of influence of the music composer is too small.

A final note from the expert, during the end of the interview. The expert asked if the user will see and notice all these kind of rules and details of director and producer. He asked the question followed with the advice, that we should not show these intricacies to the end user. Simply because the user does not care about these intricacies. Otherwise the industry would not pay editors that much for movie trailers. The average user can only remember the name of the movie and some famous actors. In the vision of the expert, the end product should be seamless and work automatically, just by observing the watching behavior of the user.

Domain Structure

We are going to use a solution based on semantic web technology. Therefore we present the domain schema as an architecture based on the existing movie database which is called *LinkedMovieDataBase*[2]. The schema can be found in figure 6. Here, there is a divide between the existing relevant structures belonging to this database to the left and our own database to the right.

In the domain of different movie types, the class movie has several sub-classes. For example, it has the subclass of genre and within genre there are even more types. Under the subclass genre, horror and fantasy are two examples. Furthermore, a movie is identified by an ID, and in our case we use the same identification numbers from LMDB. A movie also has staff that worked on the movie or acted in the film. Numerous jobs are needed for creating a movie, such as actors, producers, directors, etc. From our expert interview, some important concepts came up. When recommending a movie to watch, one does not only need to pay attention to the genre and user preferences, but also the setting is relevant when watching the movie. Just like every party has a theme, every occasion to watch a movie also has a theme. This rule is one of the most important rules; for example, one cannot show a horror movie on a child's birthday. Apart from that, genre also plays a big role in recommending the right movie, and so do the director and the producer. The latter two concepts have an important role because they have their own specific audience.

To start the movie recommendation, some information is needed from the user. The system needs a profile of the user with known preferences for actors, genres, producers and directors. Information that can be gleaned from previous liked movies and ratings. From this information, norms can be specified, whereupon a likely candidate movie will be selected. The rule types for norm specification and candidate selection are shown in the following structures and a graphical representation is available in figure 5.

[2]http://www.linkedmdb.org/

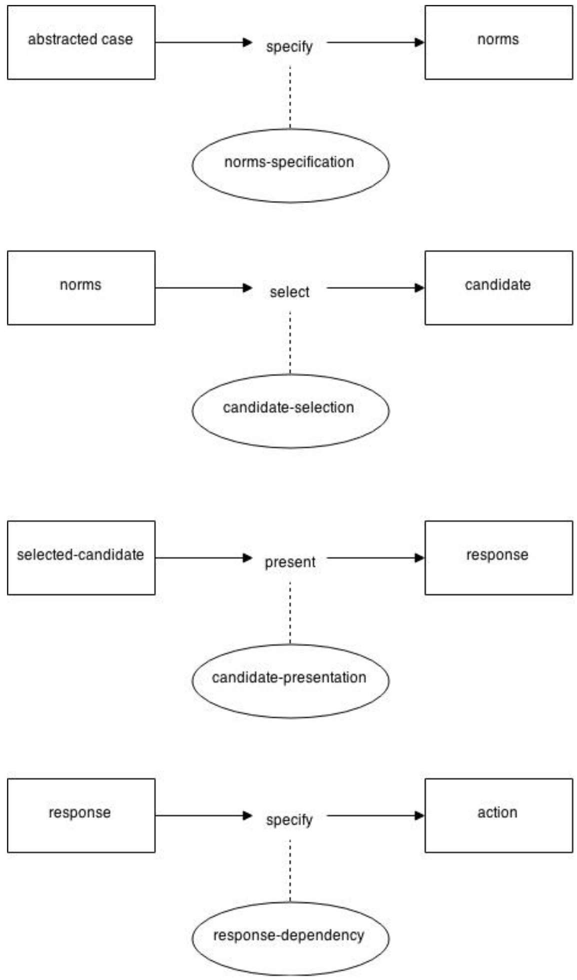

Figure 5: Graphical representation of the different rule-types.

```
RULE-TYPE norms-specification;
     ANTECEDENT:
          abstracted-case;
     CONSEQUENT:
          norms;
     CONNECTION-SYMBOL:
          specify:
END RULE-TYPE norms-specification;

RULE-TYPE candidate-selection;
     ANTECEDENT:
          norms;
     CONSEQUENT:
          candidate;
     CONNECTION-SYMBOL:
          select:
END RULE-TYPE candidate-selection;
```

When a suitable candidate movie is found, the next step of the system is to present it to the user. The next step depends on the user's response. If the user rejects the movie, an important design decision comes into play. The expert states that it is very important to present multiple options, but not all of them at once, since the user could reach a state of indecisiveness. When the other situation occurs, and the user accepts the movie suggestion, another action will be activated. The following rule-types will describe these steps:

```
RULE-TYPE candidate-presentation;
     ANTECEDENT:
          selected-candidate;
     CONSEQUENT:
          response;
     CONNECTION-SYMBOL:
          present:
END RULE-TYPE candidate-presentation;

RULE-TYPE response-dependency;
     ANTECEDENT:
          response;
     CONSEQUENT:
          action;
     CONNECTION-SYMBOL:
          specify:
END RULE-TYPE response-dependency;
```

3.2.2 Knowledge Base

```
KNOWLEDGE-BASE movie-recommendation
     USES:
          norms-specification FROM prediction schema;
          candidate-selection FROM prediction schema;
          candidate-presentation FROM prediction schema;
          response-dependency FROM prediction schema;
EXPRESSIONS:
     /* norms-specification rules */
     movie.rating = rating
     movie.genre = genre
     movie.director = director
     movie.actors = actor
     movie.producer = producer
          CAUSES
     preferences.genre += rating
     preferences.director += rating
     preferences.actors += rating
     preferences.producer += rating

     setting = date
          CAUSES
     preference.genre.documentary = 0
     preference.genre.parody = 0
```

```
    setting = children
        CAUSES
    preference.genre.documentary = 0
    preference.genre.romance = 0

    setting = friends
        CAUSES
    preference.genre.romance = 0

    /* candidate-selection rules */
    candidate.watched-before = true
        CAUSES
    candidate.dismissed = true

    candidate.watched-before = false
        CAUSES
    candidate.dismissed = false

    candidate.genre != preference.genre
        CAUSES
    candidate.dismissed = true

    candidate.dismissed = false
    candidate.genre = genre
    candidate.director = director
    candidate.actors = actor
    candidate.producer = producer
        CAUSES
    candidate.rating = preferences.genre * weight.genre + preferences.director
             * weight.director + preferences.actors * weight.actors +
             preferences.producer * weight.producer

    candidate.highest-rating = true
        CAUSES
    candidate.selected = true

    /* candidate-presentation rules */
    candidate.is-selected = true
        CAUSES
    candidate.presented = true

    /* response-dependency rules */
    candidate.rejected = true
        CAUSES
    new-candidate.selection = true

    candidate = accepted
        CAUSES
    new-candidate.selection = false
END KNOWLEDGE-BASE movie-recommendation
```

3.3 Process Model

A psuedo-code specification of the process model looks as follows.

```
TASK Determine-right-movie:
     GOAL: "To determine the right movie that goes well with the setting and
          the user";
     ROLES:
          INPUT:
               setting:  "The mood and setting the movie watching is taking
                    place in.  For example, romantic date or with friends";
               user:  "The selected movie must fit to the preferences and
                    likings of the user, for example right genre and favorite
                     actors";
          OUTPUT:
               right-movie:  "The right movie that satisfies the user
                    and increases the chance that the customer keeps using
                    the recommendation service";
          SPECIFICATION: "Find the right movie fitting the
               setting and the user";
END-TASK determine-right-movie;
```

```
TASK METHOD best-movie-determination-method:
     REALIZES: determine-right-movie;
     DECOMPOSITION:
          INFERENCES: specify, select, present;
          TRANSFER-FUNCTIONS: present;
     CONTROL-STRUCTURE:
          WHILE results == false DO
               specify(setting & user -> norm);
               select(norm -> candidate);
               present(candidate + user_feedback -> result)
          END WHILE
END TASK_METHOD
```

3.4 Scenarios

Here are some scenarios along with the corresponding information of what is happening in the knowledge base.

3.4.1 Science fiction lover watching a movie with a date

Domain	**Model**	**Explanation**
A user is logged into the system and wants a new movie suggestion. The user is going to watch the movie with a date.	PREDICT: **input**:{User history, Current setting="Date"}	A user enters the setting into the system after already having a history recorded in the system.
The user gave the movie *Interstellar* five stars.	CAUSES: preferences."Christopher Nolan" += 5 preferences."Matthew McConaughey" += 5 preferences."Science fiction" += 5.	Because Christopher Nolan is the director of *Interstellar*, he receives five points in the user's ranking points. The same thing happens to the actor Matthew McConaughey and all other actors involved in the movie, as well as to the genre score.
The user gave the movie *The Lego Movie* one star.	CAUSES: preferences."Phil Lord" += 1 preferences."Chris Pratt" += 1 preferences."Adventure" += 1	Because Phil Lord is the director of *The Lego Movie*, he receives one point in the user's ranking points. The same thing happens to the actor Chris Pratt and all other actors involved in the movie, as well as to the genre preference.
The user's choice of watching the movie with friends influences what genre is going to be watched.	CAUSES: preferences."Documentary" = 0 preferences."Parody" = 0	Because the user is going to watch the movie with a date, some genres are considered inappropriate for the setting and their score is therefore set to 0.
The science fiction film *Inception*, directed by Christopher Nolan, will be highly rated and suggested.	CAUSES: "Inception".rating = $5 * 4 + 5 * 3 = 35$ "Inception".selected = true	Because the genre aspect has a high weight in the system, the suggestion will be a science fiction and will be starring a highly rated director, which is the second-most important criterion according to the weight settings.
The user rejects the suggestion of the film *Inception*, because he has heard that it was not so good, and, as a consequence, the system suggests the movie *Contact* instead.	CAUSES: new-candidate.selection = true "Contact".rating = $5 * 4 + 5 * 2 = 30$ "Contact".selected = true	The third-most weighted aspect is the actor of the movie, which makes the system select a science-fiction movie starring Matthew McConaughey as an actor (which constitutes the part $5 * 2$)

3.4.2 Horror movie fan watching a movie alone

Domain	**Model**	**Explanation**
A user is logged into the system and wants a new movie suggestion. The user is going to watch the movie alone.	PREDICT: **input**:{User history, Current setting="Alone"}	A user enters the setting into the system after already having a history recorded in the system.
The user gave the movie *The Conjuring* four stars.	CAUSES: preferences."James Wan" += 4 preferences."Leonardo DiCaprio" += 4 preferences."Horror" += 4.	Because James Wan is the director of *The Conjuring*, he receives four points in the user's ranking points. The same thing happens to the actor James Wan and all other actors involved in the movie, as well as to the genre score.
The user gave the movie *Insidious* three stars.	CAUSES: preferences."James Wan" += 3 preferences."Patrick Wilson" += 3 preferences."Horror" += 3	The user likes another movie with the same genre and director, causing the score of the two to go up even more. We also get some new actor scores.
The user is alone when watching the movie, so only the user's history matters when watching the movie.	-	The situation of watching movie alone does not change anything in the users preference, so the system simply bypasses the setting parameter.
The science fiction film *Insidious: Chapter 2*, directed by James Wan, will be highly rated and suggested.	CAUSES: "Insidious: Chapter 2".rating = $7*4+7*3+3*2=45$ "Insidious: Chapter 2".selected = true	This movie has three attributes that the user has appreciated before: being a horror movie, starring Patrick Wilson, and being directed by James Wan.
The user accepts the suggested movie	CAUSES: "Insidious: Chapter 2".accepted = true new-candidate.selection = false	The movie was accepted by the user.

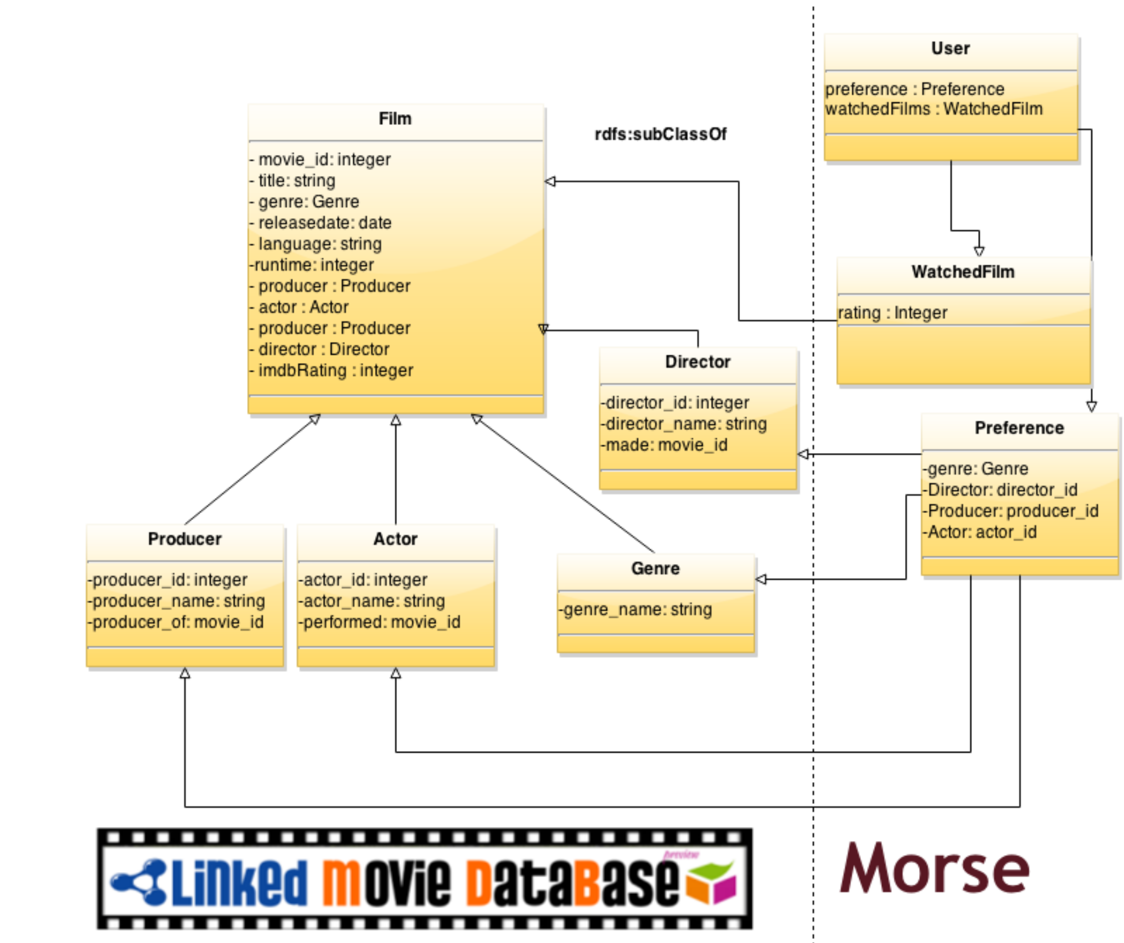

Figure 6: The domain schema representing the semantic solution for the movie recommendation system.

4 Communication Model

Here we will present how key elements in the architecture communicate with one another.

4.1 Communication Plan

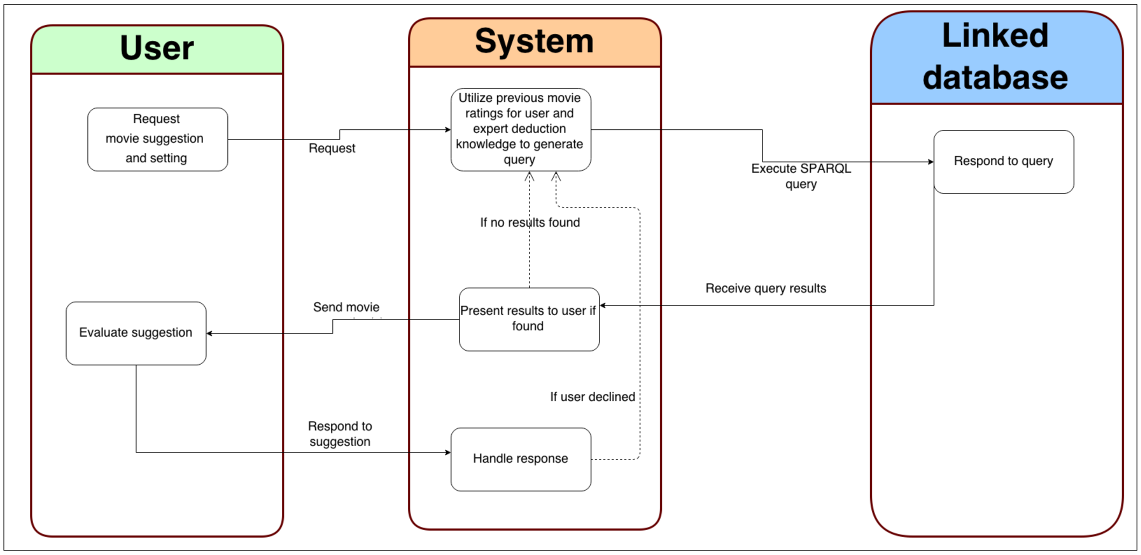

Figure 7: The communication plan

4.2 Transactions

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	Request.
INFORMATION OBJECT	Transaction between user movie suggestion request and system query generation.
AGENTS INVOLVED	The user and the system.
COMMUNICATION PLAN	P-1.
CONSTRAINTS	Pre-condition: The user has a request and knows the setting.

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	Execute SPARQL query.
INFORMATION OBJECT	A SPARQL query retrieving a movie according to extracted norms.
AGENTS INVOLVED	The system and the database.
COMMUNICATION PLAN	P-1.
CONSTRAINTS	Pre-condition: The system knows enough to form a query to the database.

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	Receive query results.
INFORMATION OBJECT	The results of the executed query expressed in triples.
AGENTS INVOLVED	The database and the system.
COMMUNICATION PLAN	P-1.
CONSTRAINTS	Pre-condition: The database must be online and be able to respond to the system.

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	Send movie suggestion.
INFORMATION OBJECT	A movie suggestion, with corresponding title, picture and other information.
AGENTS INVOLVED	The system and the user.
COMMUNICATION PLAN	P-1
CONSTRAINTS	Pre-condition: The system has been able to find an appropriate suggestion for the user.

Communication model	**Transaction Description Worksheet CM-1**
TRANSACTION IDENTIFIER/NAME	Respond to suggestion.
INFORMATION OBJECT	A response can be either accepting or rejecting the proposed suggestion
AGENTS INVOLVED	The system and the user.
COMMUNICATION PLAN	P-1
CONSTRAINTS	Pre-condition: The user is able to evaluate whether the suggestion is good or not.

Communication model	**Information Exchange Specification Worksheet CM-2**
TRANSACTION	Request.
AGENTS INVOLVED	1. **Sender**: User 2. **Receiver**: System
INFORMATION ITEMS	1. **Role**: Core 2. **Form**: Boolean and enumerable (Representing the setting) 3. **Medium**: Online graphical user interface.
MESSAGE SPECIFICATIONS	Describe all messages that make up the transaction. For each individual message describe: 1. **Communication type**: Request/propose. 2. **Content**: User requires a movie suggestion from the service. 3. **Reference**: N/A.
CONTROL OVER MESSAGES	Setting.

Communication model	**Information Exchange Specification Worksheet CM-2**
TRANSACTION	Execute SPARQL query.
AGENTS INVOLVED	1. **Sender**: System 2. **Receiver**: Database
INFORMATION ITEMS	1. **Role**: Core. 2. **Form**: A text string. 3. **Medium**: A TCP connection.
MESSAGE SPECIFICATIONS	Ask/reply. 1. **Communication type**: Ask/Reply. 2. **Content**: A SPARQL query asking for information about movies. 3. **Reference**: Knowledge about semantic web technology.
CONTROL OVER MESSAGES	According to SPARQL specification.[a]

[a] http:www.w3.orgTRrdf-sparql-query

Communication model	Information Exchange Specification Worksheet CM-2
TRANSACTION	Receive query results.
AGENTS INVOLVED	1. **Sender**; Database. 2. **Receiver**: System.
INFORMATION ITEMS	1. **Role**: Core 2. **Form**: RDF Triplets. 3. **Medium**: TCP connection.
MESSAGE SPECIFICATIONS	1. **Communication type**: Ask/reply. 2. **Content**: Information about the movies that confirms to specified criteria. 3. **Reference**: Knowledge about semantic web technology.
CONTROL OVER MESSAGES	According to the SPARQL specification.[a]

Communication model	Information Exchange Specification Worksheet CM-2
TRANSACTION	Send movie suggestion.
AGENTS INVOLVED	1. **Sender**: System 2. **Receiver**: User
INFORMATION ITEMS	1. **Role**: Core 2. **Form**: Picture, name, and other information about the movie. 3. **Medium**: Online graphical user interface.
MESSAGE SPECIFICATIONS	1. **Communication type**: Request/propose. 2. **Content**: Proposal of movie to watch next. 3. **Reference**: N/A.
CONTROL OVER MESSAGES	The vastness of the linked database is going to influence the versatility of this transaction.

Communication model	Information Exchange Specification Worksheet CM-2
TRANSACTION	Respond to suggestion.
AGENTS INVOLVED	1. **Sender**: User 2. **Receiver**: System
INFORMATION ITEMS	1. **Role**: Core 2. **Form**: Boolean. 3. **Medium**: Online graphical user interface.
MESSAGE SPECIFICATIONS	1. **Communication type**: Report. 2. **Content**: Response to whether the user likes the advice given and is going to follow it or not. 3. **Reference**: N/A.
CONTROL OVER MESSAGES	User's impression of the movie.

[a] http:www.w3.orgTRrdf-sparql-query

5 Design Model

Design Model	Worksheet DM-1: System Architecture
ARCHITECTURE DECISION	**Format**
SUBSYSTEM STRUCTURE	Our system uses a variant of the Model View Controller (MVC) design pattern. This fits well with the intended tools that we are going to use, being a web application: Python with flask, Tomcat, and Sesame workbench. The tools are designed to use this architecture. See figure 8 for an overview.
CONTROL MODEL	Rule-driven.
SUB-SYSTEM DECOMPOSITION	The subsystem decomposed into modules as seen in figure 8.

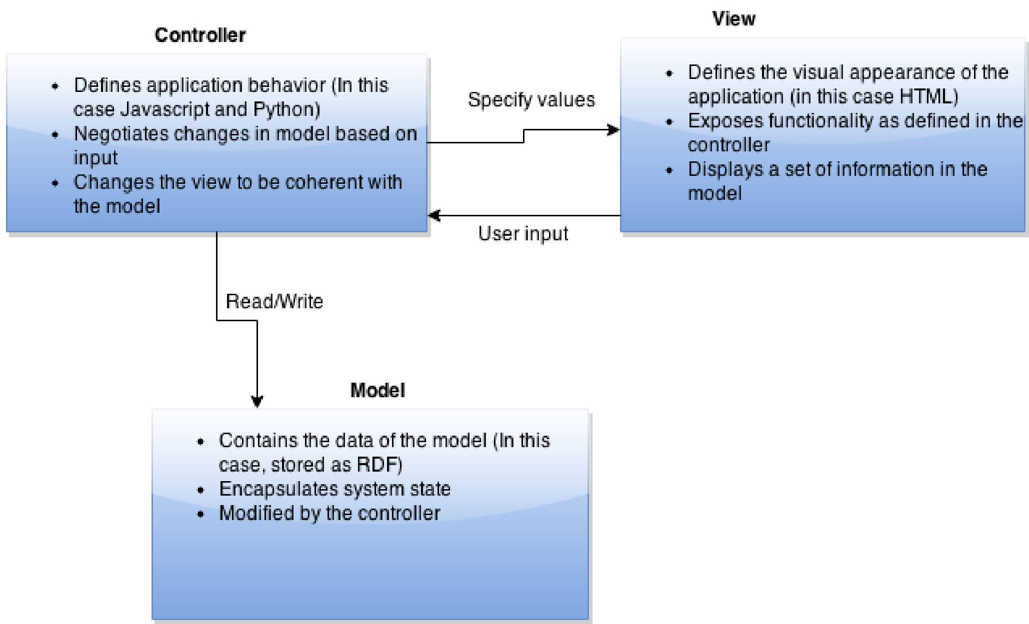

Figure 8: The model view controller architecture as implemented by the system.

Design Model	Worksheet DM-2: Target Implementation Platform
SOFTWARE PACKAGE	JavaScript, Python, HTML, Tomcat, Flask.
POTENTIAL HARDWARE	Server-hall machines.
TARGET HARDWARE	Personal computers running Windows, Mac OSX, UNIX and smart phones with Android, iOS and Windows Phone.
VISUALIZATION LIBRARY	Jinja2, CSS (Bootstrap), HTML.
LANGUAGE TYPING	Javascript and Python are untyped.
KNOWLEDGE REPRESENTATION	The RDF semantic web technology offers a basis on which to perform reasoning tasks.
INTERACTION PROTOCOLS	SPARQL
CONTROL FLOW	Cloud-based server solution.
COMMONKADS SUPPORT.	-

6 Reflection

Process
In hindsight, there are some things that could be done differently. If we had taken more time with the expert, we could probably have come to think about more clever rules for the reasoning part. However, we managed to achieve a situation-dependent expert-based suggestion tool that we haven't yet seen elsewhere. The application brought up some implementation-related challenges and we had to brush up our semantic web skills, which took some time and constrained us from improving on the quality of the algorithm itself.

Results
The movie database we made use of was a considerable hindrance in achieving high quality results from our algorithm. The amount of movies it contained was too small compared to the wide array of movie options available today, and many of the links in the database were missing, causing some of the movie posters to not be displayed. We predict that replacing this database with a more comprehensive one (and costly), such as IMDB, would greatly enhance the suggestion quality of our system. Secondly, for the first try we did not implement a system to record watching behavior of the user. There is not yet a graphical user interface that the user can use to rate movies and store these ratings. For now the demo is done by manually entering the information in the database, in triples. Finally, we delayed the implementation of the live version, because we underestimated it. The live implementation of a triple store was harder than just replacing the local server address with a online one. If we began the investigation of a live version earlier, an online version of the app could be a reality. For now we have to do with a video introduction of the product.[3] Alternatively one can also setup a triple store locally on the computer, to test this application.[4]

[3] https://www.youtube.com/watch?v=D6JcLGN24eo
[4] https://github.com/RinkeHoekstra/iwa-tutorial