**TARS Chatbot Report**

By: Sameer Haider and Priyesh Patel

Introduction

Our chatbot is known as TARS. The name is inspired from the actual artificial robot that was portrayed in Interstellar. TARS is able to answer general questions about the movie *Interstellar*. It can answer specific questions within the knowledge domain around the movie, as well as some questions about the cast. Possible questions consist of “what is interstellar”, “who’s the director”, “who’s in the cast” etc… We were able to make the chatbot flexible, allowing multiple variations of input to the same intent. This allows more of a dynamic feel.

As for the development of our chatout we decided to use AIML, an XML-based language that was developed by the A.L.I.C.E AI Foundation and Richard Wallace. AIML provides a simple way to develop responses to user info in order to build a chatbot interface. We used the PyAIML library in python to interact with the AIML file. Through our chatbot creation we started to see more and more issues with AIML’s pattern matching. The pattern matching was so strict that any slight spelling mistakes would cause the chatbot to not understand the user's request. We were able to solve these issues using some NLP techniques that we learned from class.

Instructions for setting up Chatbot

1. First clone the repo:

**git clone** [**https://github.com/sameer-haider/HLT-chatbot-project.git**](https://github.com/sameer-haider/HLT-chatbot-project.git)

1. Open inside an IDE, then create a virtual environment with python version 3.7 using the following command: (install [python-3.7](https://www.python.org/downloads/release/python-370/) first if you don’t have it)   
   **python3.7 -m venv <venv-name>**
2. Source it with the following command. You can also do it through your IDE

Mac: **source <venv-name>/bin/activate** or Windows: **<venv-name>\Scripts\activate.bat**

1. Install all the required packages & respective versions with the following:  
   **pip install -r requirements.txt**
2. Run main.py

**\*NOTES\*** if an error related to en\_core\_web\_md occurs, remove the first line (starts with “https”) in requirements.txt and run this command in the terminal. Then do step 4 again:

**python -m spacy download en\_core\_web\_md**

System Description

NLP Techniques

We used multiple NLP techniques to improve the functionality of our chatbot.

*Libraries used: nltk, spaCy, and numpy.*

### 1. Similarity Matching

We used a combination of nlp techniques in order to match user input to the patterns.

#### Remove extraneous characters

Remove punctuation and other characters, only keeping alphanumeric characters.

#### Tokenization

Tokenize the user input as well as all the intents in the base.

#### Lemmatization

Lemmatize each token, turning it into its base form

#### Calculate Tf-vectors

Calculate the term frequencies for the user input and each pattern/intent

#### Calculate Cosine Similarity scores

Calculate the cosine similarity between the user\_input and each pattern. We store the highest one.

#### Thresholding

Return the highest-score pattern if the cosine similarity score is above the threshold. We picked the threshold value after testing and tweaking our chatbot.

### 2. Named Entity Recognition

We used the powerful spaCy library to implement Named Entity Recognition

#### Download database

Load the *en\_core\_web\_md* database from spaCy

#### Process message

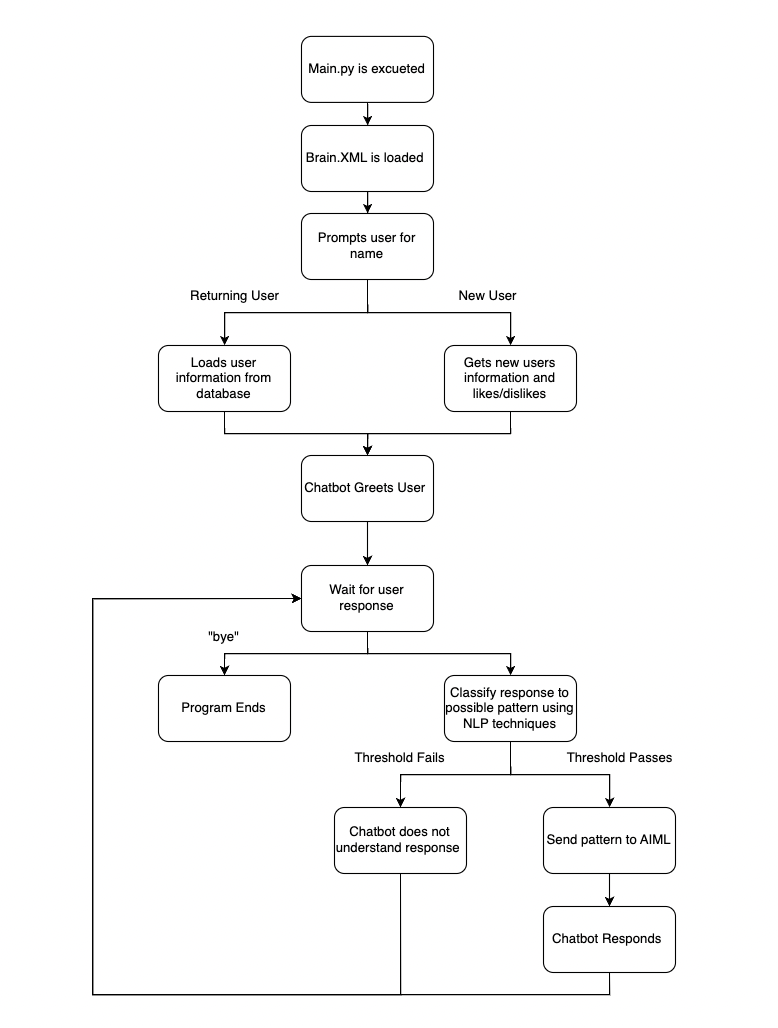
Process the message with the spaCy module, which does the main NER computation

#### Retrieve named entities

Retrieve the named entities + type and return. Return None if no named entities.

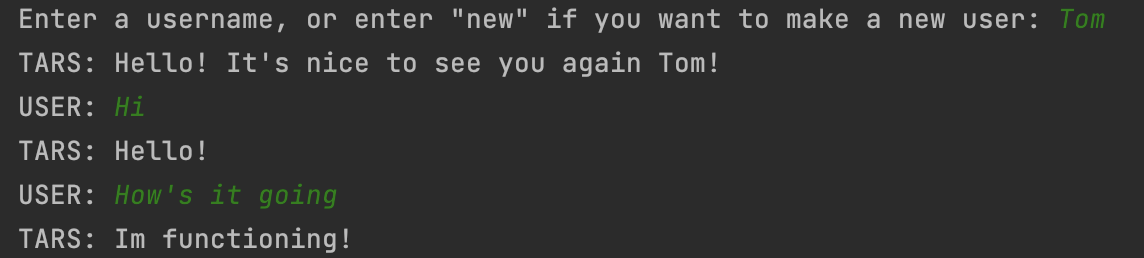
After retrieving the named entities, we use them to customize certain responses like *“What character did [actor] play?”*

Depending on the actor (which is identified with NER), we return a specific response.

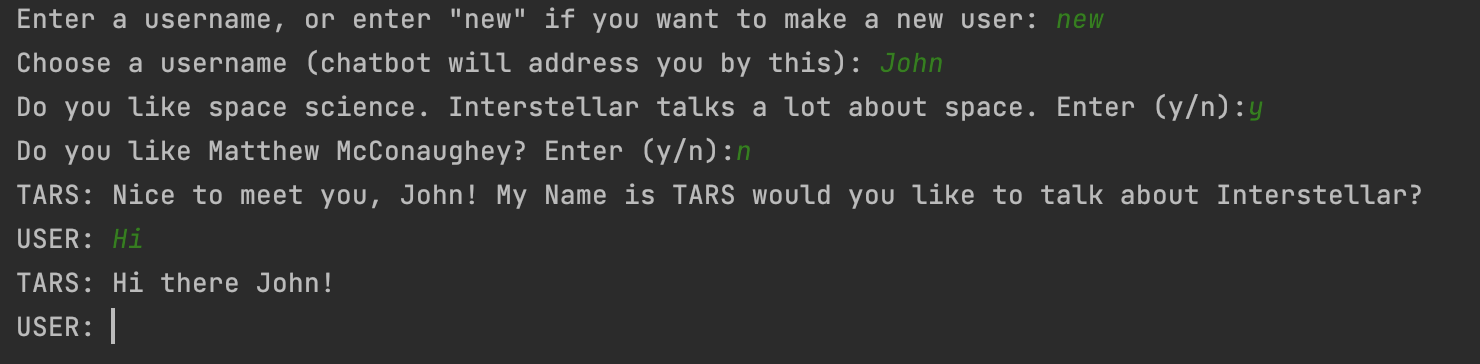
Diagram of Dialog & Logic

Sample Dialog Interactions

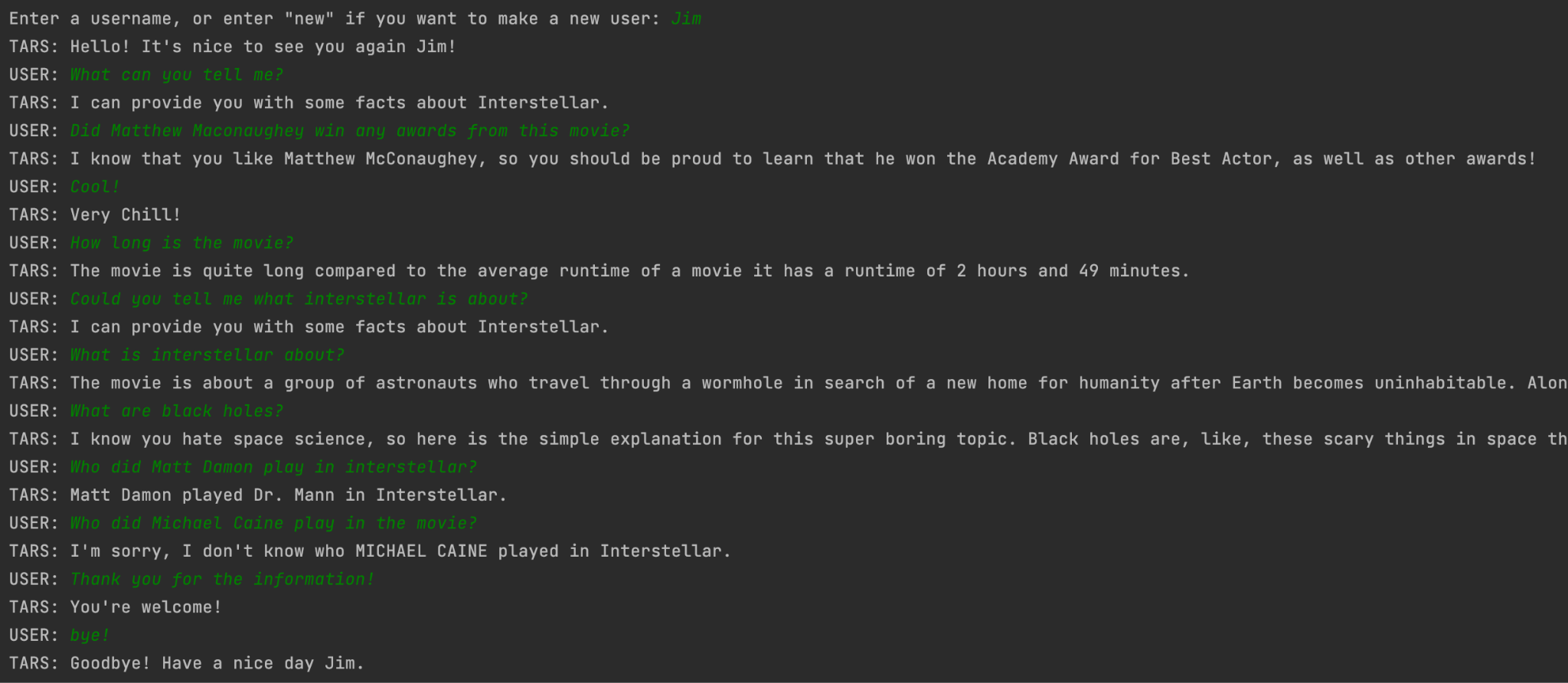
1. Existing User



1. New User



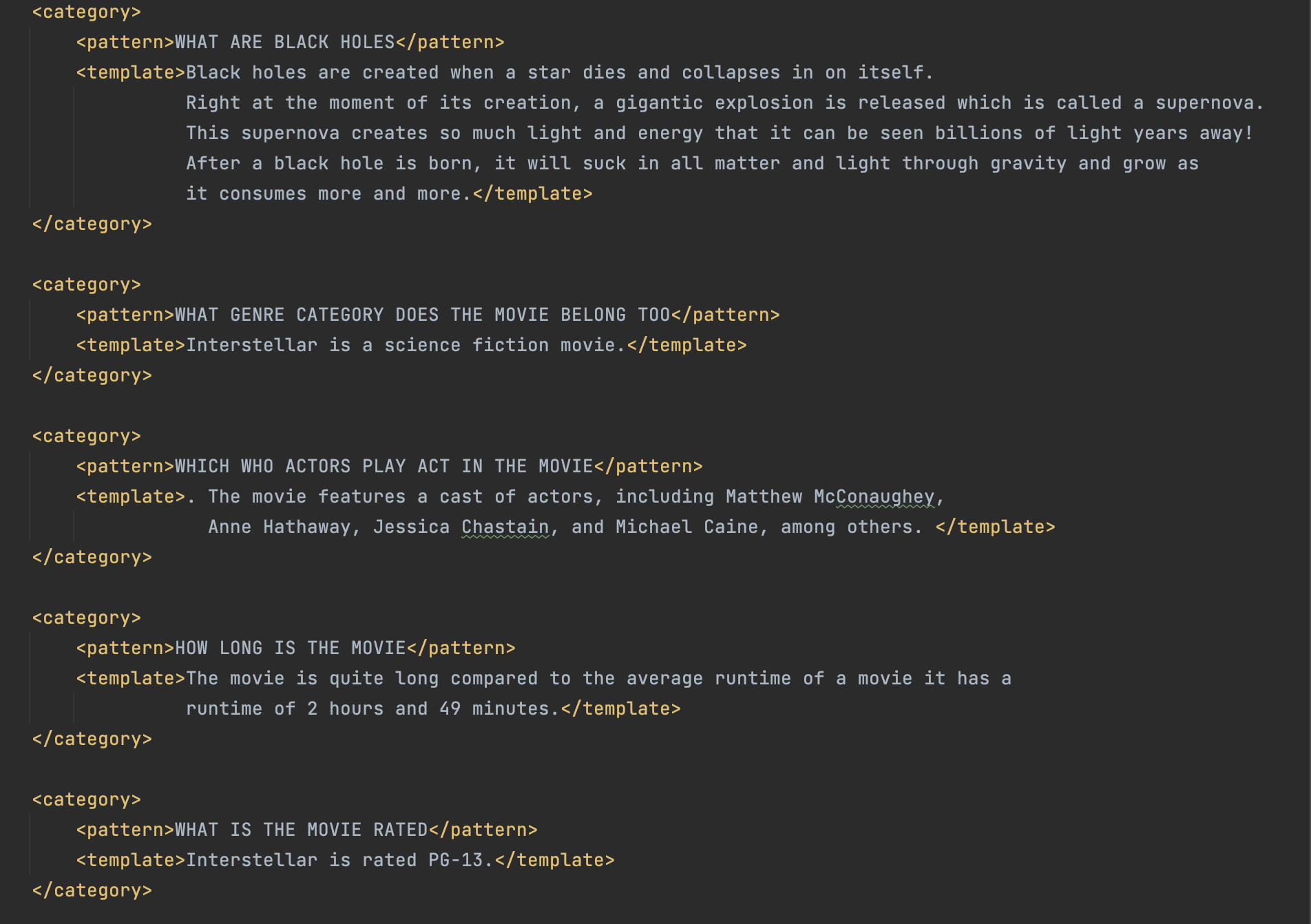
1. Full Conversation involving NER, Interest based reponses, and ending conversation

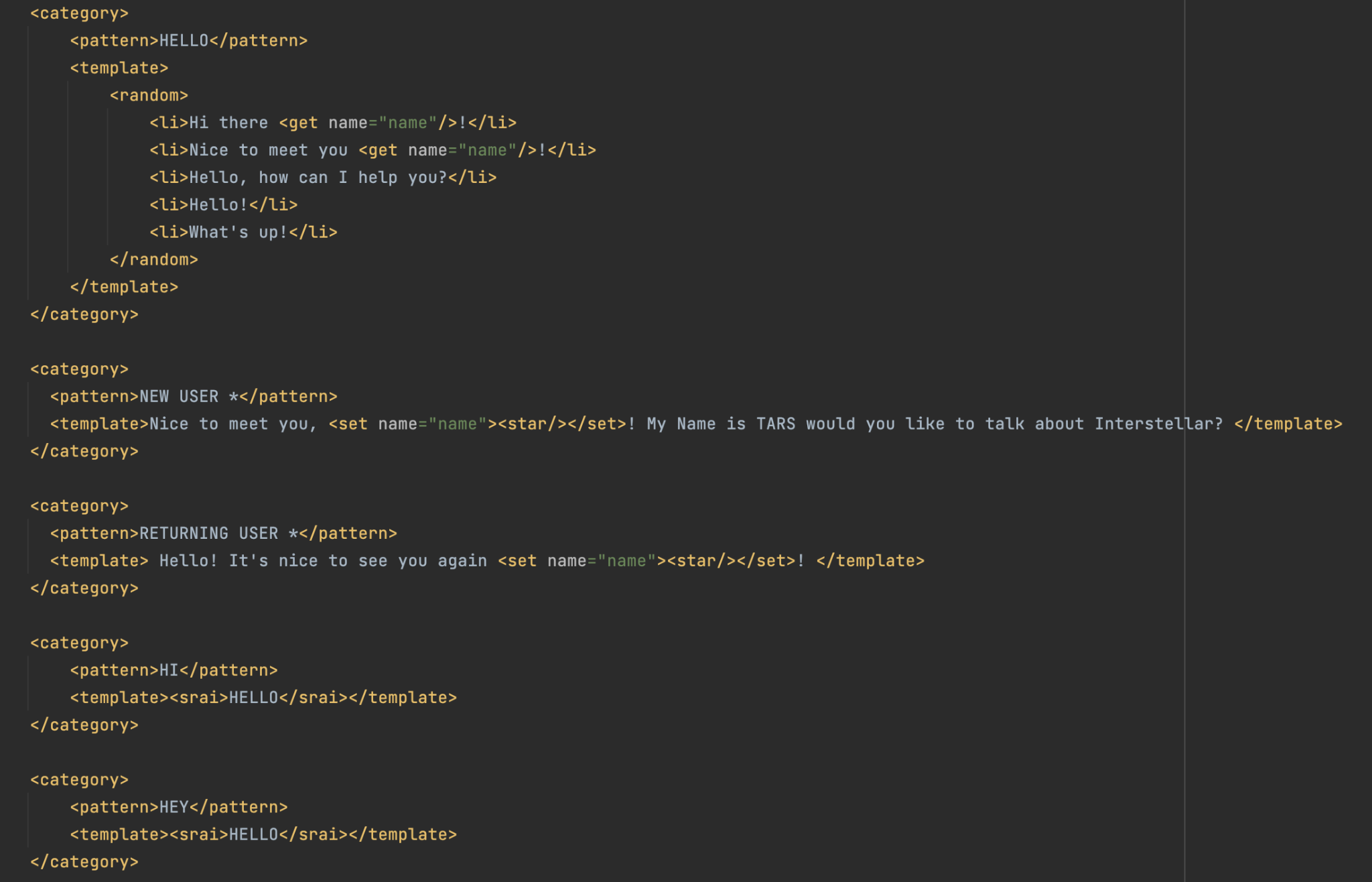


Appendix A - Knowledge Base

Since we decided to work with AIML we needed a brain or knowledge base XML file. This file is used to match patterns and return appropriate responses. We were able to use information gained from the web crawler assignment to create responses to the user’s questions. One drawback to our knowledge bases is that it is fairly small mainly due to our topic.

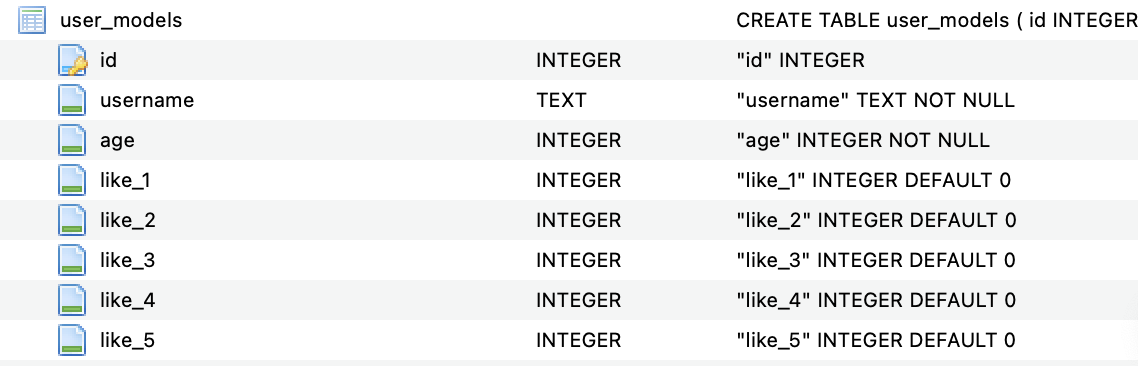
Below is snippets from the Brain.XML file





Appendix B - User Models

We store the user information in the user\_models SQL table. The table is stored locally in the user\_models.db file, with test users already inserted. The table looks like the following:

  
All of the like\_n store binary 0 or 1, to show if the user likes or dislikes a certain topic as determined in the code. We use the SQLite3 library in python to add new users to the table and also retrieve information to use in the chatbot.

Evaluation and Analysis

TARS is a simple chatbot we created that can provide a user with limited information and facts relating to the movie Interstellar. We plan on refining dialog and building a complex chatbot that can pick up on a user interest through dialog alone. Currently, TARS can provide limited information about Interstellar such as plot, cast, and awards. Due to the limited amount of information one movie can provide we would like to expand the chatbot to work with all movies instead of just one in the future. This would provide us with a larger knowledge base. A limited knowledge base was very hard to work with so a larger one would help.

We were able to create the TARs chatbot almost completely from scratch with the help from AIML. TARs has some strengths which include a clean user interface, named entity recognition to identify actor names, and a cosine similarity score to classify the users response. In addition, based on the user's likes or dislikes TARs can display specific responses that would align with those likes or dislikes. As for weaknesses, TARs has a very limited knowledge bases so the responses are also very limited. Another weakness we faced was using AIML. Initially, using the AIML pattern match was very useful. However, as our responses got more and more complex, AIML was very strict and couldn’t determine the appropriate response. Our solution was to use NLP techniques such as cosine similarity, TF-vectors, and threshold scores to identify the closest related response.

As for future improvements, we would like to expand our current chatbot to have conversations about movies in general instead of only Interstellar. For this we would need a larger knowledge base. We would try to implement a web live look-up compared to a static knowledge base. With the issues faced using AIML we would also try a different form of pattern matching and response. Another improvement we want to make is to update the user models with more information and have the chatbot respond uniquely based on the users current mood. Overall, there are a lot of optimization and functionalities we could incorporate. We could also incorporate machine learning to the chatbot so that it can learn based on each user.