Doug Marcum

DSC 680 – Project 2 Draft

Movie Recommendation Engine Chatbot

Abstract

Chatbots are an excellent tool for communication being utilized by nearly every industry. It is estimated that chatbot chat sessions can cut customer service costs by more than 30%, and it is estimated that chatbots will help reduce business costs by approximately $8 billion by the year 2022. On any given day, we can expect to interact with a chatbot in some form. From contact customer service to booking a flight or asking for the weather, chatbots are becoming engrained in our society. With this, why is there still resistance to their full utilization? Most surveys show that the average consumer would still prefer to have direct contact with a human instead of working through a chatbot. Is this related to poor chatbot design? Is it a generational issue? Before being able to explain complex questions like these, the understanding of chatbots and the elements they are composed of must be examined. This project is designed in a manner to allow for exploration into two arenas, recommendation engines and conversational chatbots. A functioning movie recommendation engine will be integrated into a Rasa generated chatbot to highlight and showcase elements from each.

Introduction

As technology has become a necessity and not a novelty over the past twenty years, it is fascinating to explore the areas that are adapting and impacting our lives the most. With data and communication being a focal point for nearly everyone today, the relevancy of chatbots has become a central theme.

Chatbots are not a new idea, as they actually predate the internet with the creation of ELIZA[1], yet they still feeling modern. The idea of speaking with a "robot" causes hesitation with some, as they are uncertain as to if the results will be valid or if their information will be safe. As a direct result of the global pandemic caused by Covid-19, user acceptance of chatbots has risen. Being forced to isolate and have little direct human interaction has led to a direct increase in the need for better chatbots in terms of deign and efficiency.

With this in mind, the purpose of this project and paper was born. Three main elements are to be explored:

1. Since many users are comfortable looking for answers to questions with Google or Wikipedia, it is important to first design a subject matter that is more than just a simple fact. For this, a movie recommendation engine will be created. This engine will allow the user to have movies recommended by genre or based on the similarity to another film.
2. A conversational chatbot will be constructed to move past a command line input output form of response. As chatbots have evolved, the line between a bot and a human has blurred a bit in a chatting forum. Even as users interact with chatbots like Amazon's Alexa or Apple's Siri, the idea of solely speaking with a robot has softened. With this, it is important to see how a chatbot can be programmed to have a conversation and not just process information.
3. The final element of this project falls into the arena of identifying why users are still reluctant to fully utilize bot-centric technology. Only 23% of customer service organizations are using AI chatbots, and approximately only 55% of consumers are willing to interact with a business via

[1] Edwards, N. (2019, March 4). A Brief History of Chatbots. Retrieved from https://mhrglobal.com/us/en/blog/a-brief-history-of-chatbots.

messaging apps to solve a problem.[2] Why are chatbots not being utilized to a greater extent, and what is causing hesitate on the side of the consumer to engage?

Data

The basis of the movie recommendation engine was built utilizing a collection of movie data obtained from IMDB. This data can be found via Kaggle. https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset?select=IMDb+title_principals.csv. The datasets include the following:

- The movies dataset includes 85,855 movies with attributes such as movie description, average rating, number of votes, genre, etc.
- The ratings dataset includes 85,855 rating details from demographic perspective.
- The names dataset includes 297,705 cast members with personal attributes such as birth details, death details, height, spouses, children, etc.
- The title principals dataset includes 835,513 cast members roles in movies with attributes such as IMDb title id, IMDb name id, order of importance in the movie, role, and characters played.

Data pertaining to the chatbot was partially collected from an intents and stories dataset originating from Kaggle. By utilizing this initial data, additional aspects were easily added to further Rasa Open Source's Natural Language Understanding (NLU) model. This intent dataset can be found here: https://www.kaggle.com/swapnilpote/movie-chatbot-dataset/metadata.

Data Preparation

Preparing the data for each model was interesting, as each model requires different interactions, respectively. First, let us review the movies dataset, as it has a direct impact on both models. The data in the move dataset is related to general information one might obtain about a movie (title, director, cast, rating, budget, gross income, and so on) and consisted of 85,855 rows. The first pass did show a number of null values for multiple features. Fortunately, the only feature directly impacted by this was the description field, and only 2115 (approximately 2.4%) did not include a description. Since the description is an important feature for the recommendation engine and these fields could not be filled with alternative information, they were dropped from the dataset.

After reviewing the remaining features with null values, they were determined to have zero impact on the recommendation engine. However, these features are important, as the information or lack of information, may be needed for completing a request to the chatbot. Features 'metascore', 'reviews_from_users', and 'reviews_from_critics" were dropped, as they were missing from multiple rows, information could not be filled in their place, and the numeric values for the two reviews columns had little value, as the fields only displayed the number of reviews without any context. Features for 'avg_vote' and 'votes' were dropped, as this information would later be added from the ratings dataset. All remaining null fields were filled with 'unknown', as this information is necessary for the chatbot. The final step in data cleaning/preprocessing came in only selecting movies that were available in English in some form. This is due to two factors:

[2] Zaboj, D. (2020, May 6). Key Chatbot Statistics You Should Follow in 2021. Retrieved from https://www.chatbot.com/blog/chatbot-statistics/.

1. By eliminating movies not available in English in some form, the elimination of multi-language processing was no longer an issue.
2. Due to memory constraints, this allowed for a robust dataset to be utilized, without too much memory consumption.

Working with the names dataset was initially confusing. None of the information in the dataset is necessary for the recommendation engine but is vital for constructing answers with the chatbot. The first review of the data revealed numerous null fields, but this was expected as those fields include information relating to death. With many of those listed in the data still alive, conditional arguments were needed to separate those that were dead, yet missing information, from those who are alive. The steps to complete this process are listed below.

```
# conditions for filling NaN fields
miss_reason = (pd.notnull(names.date_of_death) == True) & (pd.notnull(names.reason_of_death)== False)
miss_place = (pd.notnull(names.date_of_death) == True) & (pd.notnull(names.place_of_death) == False)
alive = (pd.notnull(names.date_of_birth) == True) & (pd.notnull(names.date_of_death) == False)

# make copy of names data
people = names.copy()

# clean/process people data
for i in range(len(names)):
    if miss_reason[i] == True:
        people.reason_of_death[i] = 'unknown'
for i in range(len(names)):
    if miss_place[i] == True:
        people.place_of_death[i] = 'unknown'
for i in range(len(names)):
    if alive[i] == True:
        people.place_of_death[i] = 'none - alive'
        people.death_details[i] = 'none - alive'
        people.reason_of_death[i] = 'none - alive'
        people.date_of_death[i] = 'none - alive'
for i in range(len(names)):
    if people.spouses[i] == 0:
        people.spouses_string[i] = 'none'

people.fillna('unknown', inplace = True)
```

The remaining datasets required less attention, as the data was well processed for both. With the principals dataset, only the ‘job’ feature needed to be dropped, as it appeared to be filled with misaligned, random information. When working with the ratings data, it was easy to see that while a great deal of demographic information was shared, it had little value. The numerical counts for features like ‘males_18age_votes’ and ‘females_45age_avg_vote’ provided information that was too vague for recommendations. It was determined that the features (‘imdb_title_id’, ‘weighted_average_vote’, ‘total_votes’, ‘mean_vote’ and ‘median_vote’) would be retained and merged with the movies data, thus creating the rated_movies dataset.

Reviewing the rated_movies data gave insight into the composition of the dataset, and how reviewers look at movies in general. By taking advantage of the ‘weighted_average_vote’ feature, we could compare film to film, without the number of votes overly biasing the outcomes. IMDb defines the weighted average vote as:

IMDb publishes weighted vote averages rather than raw data averages. The simplest way to explain it is that although we accept and consider all votes received by users, not all votes have the same impact (or ‘weight’) on the final rating. When unusual voting activity is detected, an alternate weighting calculation may be applied in order to

preserve the reliability of our system. To ensure that our rating mechanism remains effective, we do not disclose the exact method used to generate the rating.[3]

This feature shared the most interesting data to this point. The interquartile range was 4.9 – 6.6 for 47,237 films. Initially, it was proposed that this range would be much larger, but it appears that most films produced are viewed as average by most consumers. The mean rating was 5.67. Since a recommendation model does not want to recommend poorly rated films, the bottom 25% were dropped from consideration.

The next step in preprocessing the data come in the creation of lists of words features. This was necessary to complete the transformation of selected text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. By utilizing Term Frequency Inverse Document Frequency (TF-IDF) Vectorizer, we are able to consider the overall weightage of a word. It helps us in dealing with most frequent words. TfidfVectorizer weights the word counts by a measure of how often they appear in the documents. Here we ran this against both sets of selected words ('words_plots' and 'words_features') by fitting and transforming the data to create TF-IDF matrices.

Finally, cosine similarity could be determined between each of the films. Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between the two vectors. This produced a score ranging from -1 (opposite similarity) to 1 (exact match in similarity), which in turn allowed for a value to be established for each film and the relationship it has with each film in our dataset.

Models

Before delving into the specifics of each model, it is important to understand the types of recommendation engines and chatbots. First, let us discuss recommendation engines.

Recommendation engines are not a new concept, as humans have been de facto recommendation engines themselves. If person A is curious as to what books they should read to better perform in their studies, they may seek the recommendations from other students are professors. While this is may seem simple, recommendation engines take this concept and amplify the decision making by processing vast quantities of data to determine the appropriate recommendation. This could be a product Amazon believes you may have interest in purchasing based on previous purchases to job recommendations from Monster based one's profile creation to the movie recommendation constructed in this project. With this, there are two main types of recommendation systems: Collaborative Filtering and Content-Based Filtering.

Collaborative filtering filters information by using the interactions and data collected by the system from other users. It is based on the idea that people who agreed in their evaluation of certain items are likely to agree again in the future. A content-based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.[4] With the data presented for this

[3] IMDb. Weight Average Votes. Retrieved from https://help.imdb.com/article/imdb/track-movies-tv/weighted-average-ratings/GWT2DSBYVT2F25SK#.

[4] Das, S. (2015, Aug. 11). Beginners Guide to Learn About Content Based Recommender Engines. Retrieved from https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/.

project, multiple, specific user profiles were not made available. This limited the creation of the recommendation engine to one of based on content filtering.

The recommendation engine constructed for this project works as so:

1. The array of cosine similarities is computed.
2. A series is created that is a reverse map of indices and movie titles from the rated_movies DataFrame.
3. An enumerated list of pairwise cosine similarity scores is created based off the item for recommendation.
4. Scores are then sorted with the highest score being eliminated as it is the same item as the basis of the recommendation.
5. For purposes of this model, the top five recommendations are returned.

A test of this recommender returns Star Wars: Episode VI - Return of the Jedi, Star Wars: Episode V - The Empire Strikes Back, Star Wars: Episode VII - The Force Awakens, Star Wars: Episode I - The Phantom Menace, Star Wars: Episode VIII - The Last Jedi when asked for recommendations based on Star Wars.[5]

In constructing the chatbot, Rasa Open Source was the obvious choice for this challenge. Rasa Open Source uses includes natural language understanding and open-source natural language processing to turn messages from users into intents and entities that chatbots understand. Based on lower-level machine learning libraries like Tensorflow and spaCy, Rasa Open Source provides natural language processing software that is approachable and customizable.[6]

By working with the full toolset for extracting the important keywords, or entities, from user messages, as well as the meaning or intent behind those messages, the output is a standardized, machine-readable version of the user's message, which is used to determine the chatbot's next action. This allows for a conversation drive approach that takes those insights to train and teach the chatbot to improve over time.

With the models behind the chatbot primarily coming preconstructed by Rasa, most of the time and focus falls into the defining of intents, entities, and interacting with the chatbot. This will be further discussed in the results section.

Results

The results of the two stages of the project where interesting. Since the recommendation engine was constructed using cosine similarity between film based on bag of words constructed from writers, directors, actors, and genres, results were difficult to quantify. This led to more of an understanding of the films being selected for recommendations and determining if the results were correct. Overall, the recommendations were reliable. With the small user group test conducted, the respondents overwhelmingly found the recommendations to be inline with movies they would watch based off of the movie they entered. They did find the recommendations to be limiting to a degree, but this was due to the basis of the data utilized for the recommendations. For the recommendation engine to expand beyond the scope generated in this project, a substantial increase in RAM would be needed to calculate the cosine similarity matrix.

[5] Results can be found in Appendix A

[6] Rasa Technologies, Inc. (2021). Retrieved from https://rasa.com/solutions/open-source-nlu-nlp/.

The results for the chatbot were quite interesting. As interaction with the chatbot increased, its ability to correctly determine the correct intent by the user was substantial. Often have a confidence (accuracy) score in excess of 98% when making this determination. The same can be said for the chatbots ability to learn the appropriate response to answering the questions asked by the user. The chatbot was able to identify when the user was asking for a recommendation on a film based on a genre as opposed to a specific title. Additionally, the bot was able to provide information, such as who directed and starred in the film, when prompted.

The main drawback of the chatbot came from the bot's ability to identify proper titles in entirety. This could be for example only seeing the word War as the movie title instead of Star Wars or not being able to fully distinguish the number of genres being requested. Often a romantic comedy would return results for only a comedy or an action thriller would only return recommendations for an action film.

Even this small shortcoming can be addressed with time. The number of training examples for this model was under 1000, which was due to time constraints. With continued interactions, the chatbot should continue to grow and learn from the conversations. In the coming weeks, the bot will be hosted on a website, so it can have daily interactions. After a period of six months, the results will be reexamined.

Conclusion

In conclusion, this challenge provided insights and analysis that was not expected. The recommendation engine is a tool most are familiar with and comfortable using. Nearly every commerce and entertainment platform have a recommendation engine embedded somewhere within the fabric of the company. In constructing this tool, it became obvious how necessary they have become. Most consumers are not certain as to what they want, until it is shown to them (to paraphrase Steve Jobs). By allowing ourselves to share our actions (likes, dislikes, interest, etc.), we can actually increase our exposure to the world in a positive manner. This is not to say we should be an open book, but with proper sensitive data protocols in place, recommendation engines provide a wonderful service.

To circle back to the third element of this challenge, *why are chatbots not being utilized to a greater extent, and what is causing hesitate on the side of the consumer to engage*, the construction of the chatbot put a spotlight on why we are still hesitant to fully utilize them.

1. The construction requires either a very simple chatbot or a very elaborate bot. Often bots are too simple in nature, and only provide limited information to a user before they are pushed to a human. This is a difficult area of chatbots to over come as the knowledge base required to construct an interactive bot that goes beyond a singular subject is difficult and time consuming. This is why chatbots are often used to gather information or resolve a simple issue, as they have not grown to the level of comfortably tackling more complex issues.
2. The nuances of human language are incredibly complex. Something as simple as affirming a Yes/No question can create up to 50+ responses from a user that all mean Yes or No. So as humans interact with chatbots, the expectation of past bad experiences are hard to overcome. Bots from a few years ago did not have the breadth of knowledge as they do now, and yet they are still falling short in certain areas. Users quickly grow frustrated when a chatbot cannot quickly ascertain the question that is being ask. This often leads to a user preferring to speak with a human, in order to shorten their path to a solution.

Overall, chatbots are evolving and their utilization will only increase. They have already become second nature to members of Generation Z, and even Baby Boomers have grown to respect the technology and

not fear it. So, for those that still prefer to have their complaints and questions addressed by a human, you better act fast because that will soon be a luxury and not the norm.

References

Ehiorobo, E. (January 15, 2021). How to Create and Intelligent Chatbot in Python Using spaCy NLP Library. Retrieved from https://www.digitalocean.com/community/tutorials/how-to-create-an-intelligent-chatbot-in-python-using-the-spacy-nlp-library.

Martin, J. (February 23, 2017). Design Framework for Chatbots. Retrieved from https://chatbotsmagazine.com/design-framework-for-chatbots-aa27060c4ea3.

Maruti Techlabs (May 9, 2017) How to Develop a Chatbot From Scratch. Retrieved from https://chatbotsmagazine.com/how-to-develop-a-chatbot-from-scratch-62bed1adab8c.

Moy, L. (December 12, 2020). Building a Netflix Recommendation SMS Bot with Python and Twilio Autopilot for Gift of Code. Retrieved from https://www.twilio.com/blog/netflix-bot-python-twilio-autopilot-giftofcode.

Raj, S. *Building Chatbots with Python: Using Natural Language Processing and Machine Learning.* Apress. (2018).

Rehan, A. (November 24, 2020). 10 Best Chatbot Development Frameworks to Build Powerful Bots. Retrieved from https://geekflare.com/chatbot-development-frameworks/.

Sens, R. (May 23, 2016). Designing a Chatbot Conversation. Retrieved from https://www.behance.net/gallery/37453869/Designing-a-Chatbot-UX-Design-Process-Case-Study.

Shawar, Bayan & Atwell, Eric. (2007). Chatbots: Are they Really Useful? LDV Forum. 22. 29-49. Retrieved from https://www.researchgate.net/publication/220046725_Chatbots_Are_they_Really_Useful.

Shevat, A. *Designing Bots: Creating Conversational Experiences*. O'Reilly Media. (2017).

Scott, K. (October 18, 2016). Popular Use Cases for Chatbots. Retrieved from https://chatbotsmagazine.com/popular-use-cases-for-chatbots-925ef8f2b48b.

Appendix A

```
# function that recommends similar movies based of movie title input
def get_recommendations(title):
    if title in indices:
        # index of the movie that matches the title
        idx = indices[title]
        idx_check(idx, title)
    else:
        suggested = is_this_your_movie(title)
        print('I am sorry, but I could not find {}. Did you mean {}?'.format(title, suggested[0][0]))
        answer = input('Y or N: ')
        if answer == 'Y':
            print(str(suggested[0][0]))
            get_recommendations(str(suggested[0][0]))
        elif answer == 'N':
            print('Maybe it is this one, {}?'.format(suggested[1][0]))
            answer2 = input('Y or N: ')
            if answer2 == 'Y':
                get_recommendations(suggested[1][0])
            else:
                 print("I guess we'll never know")
```

```
# idx check to account more multiple titles with same name
def idx_check(idx, title, cosine_sim = sim_feat):
    if idx.shape == ():
        # pairwise similarity scores of all movies with that movie
        sim_scores = list(enumerate(cosine_sim[idx]))
        sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
        sim_scores = sim_scores[1:6]
        movie_indices = [i[0] for i in sim_scores]
        # the top 5 most similar movies
        print('Here are some recommendations:', ", ".join(list(rated_movies['original_title'].iloc[movie_indices])))
    if idx.shape > (1, ):
        movie = rated_movies[rated_movies.original_title == title]
        print('There are multiple movies by that name. Can you help me narrow it down?')
        print('Here are your choices:')
        print(movie[['original_title', 'year', 'director', 'description']])
        print('')
        yr = input('What year did the movie come out that you are looking for: ')
        new_idx = movie[movie.year == yr]
        new_idx = np.int64(int(str(new_idx.index.values).strip('[').strip(']')))
        idx_check(new_idx, title)
```

```
get_recommendations('Star Wars')
```

```
Here are some recommendations: Star Wars: Episode VI - Return of the Jedi, Star Wars: Episode V - The Empire Strikes Bac
k, Star Wars: Episode VII - The Force Awakens, Star Wars: Episode I - The Phantom Menace, Star Wars: Episode VIII - The L
ast Jedi
```