Human-Machine Dialogue Project Report

Enhancing User Experience through Conversational Agents: The Popcorn Movie Assistant

June 6, 2023

[1]Matteo Brugnera (mat. 232670), University of Trento.
matteo.brugnera@studenti.unitn.it

1. Introduction

This report explores the realm of conversational agents and their potential to improve user experience in the domain of movies. Our focus is on a specific conversational agent named *Popcorn*[6], which serves as a movie assistant. *Popcorn* aims to provide users with a seamless and interactive way to access movie-related information, recommendations, and more. This report delves into the design, implementation, and evaluation of *Popcorn*, showcasing its capabilities and discussing its impact on enhancing user engagement and satisfaction. Through rigorous testing and analysis, we highlight the advantages, challenges, and future directions of conversational agents like *Popcorn* in the realm of movie assistance. The system employs a **multitask-based model**, enabling us to leverage the inherent relationships and correlations among different aspects of the conversational agent's functionality. This approach yields improved generalization, increased accuracy, and enhanced performance across diverse tasks. In fact, the system exhibits the capability to perform the following tasks:

1.1. TASK #1: Information Retrieval Task

This task empowers users to obtain a wide range of movie-related information effortlessly. Users have the freedom to inquire about various aspects such as *genre*, *budget*, *runtime*, *cast*, and more. This task can be activated by users by simply asking specific questions such as *"Can you tell me the genre of Avatar?"*. Furthermore, our system is designed to handle multiple inquiries within the same sentence, allowing users to efficiently request information on various aspects simultaneously. For example, they can inquire about the *budget* and *runtime* of a movie all in one sentence.

1.2. TASK #2: Movie Recommendation

The movie recommendation task can be invoked by users in two distinct ways:

- the first approach allows users to initiate the task by simply asking *"Can you recommend me a movie?"*. In response, the system engages in a dynamic conversation, prompting the user to specify their preferences such as *genre*, *rating*, *cast*, and more. By gathering these preferences, the system generates a personalized recommendation that encompasses all the requested elements, ensuring a tailored movie suggestion aligned with the user's tastes.
- Alternatively, users can trigger the recommendation task by directly specifying a similar movie that they enjoyed, for instance, by asking, *"Can you recommend me a movie that is similar to Avatar?"*. In this scenario, the system leverages the provided reference and retrieves movies that share similarities in terms of *genre*, theme, or other relevant factors, offering a recommendation that aligns with the user's indicated preference.

1.3. TASK #3: Plot-based Movie Recommendation Task

In the third task, users have the convenience of simply specifying a plot or theme that they would like to explore, and the system responds by offering thoughtful guidance. By leveraging advanced algorithms and analyzing the provided plot or theme, the system suggests five movies that exhibit similarities in terms of storyline or thematic elements.

Finally, the **target user group** for this conversational agent encompasses a diverse range of individuals, catering to both cinema enthusiasts seeking to deepen their knowledge and discover fascinating aspects of their favorite movies, as well as individuals who may have limited knowledge and seek recommendations based on their personal preferences.

2. Conversation Design

Conversation design plays a pivotal role in creating engaging and dynamic interactions between users and virtual agents. Understanding how this changes depending on the task is key to designing *effective* and *contextually-aware* conversational experiences.
The system can be initiated with a **greeting**. For users who are interacting with the system for the first time, they may inquire about the available options by asking questions such as *"What can I do?"*. In response, the agent will provide a comprehensive list of all the available possibilities and functionalities. From this point onward, the user may embark on various paths based on the specific task they wish to accomplish.

2.1. TASK #1

The conversation in this specific task is initiated by the user, which is commonly known as **"user-initiative"**.
This initiative follows a **"rule-based"** approach, where specific conditions or events trigger predefined actions or responses. This design choice was made for several reasons. Firstly, the nature of the questions asked in this task is commonly encountered, allowing us to define rules to handle these common user queries. Additionally, rule-based initiatives provide control over the flow of the conversation and enable predefined responses and actions.

With this design, users can smoothly ask multiple questions one after the other, maintaining a coherent conversation. Furthermore, these rules can handle *multi-intent requests*, where users ask for multiple things in a single question. A really important part is played by the **Error Handling and Recovery**. It refers to the techniques and mechanisms employed to handle and recover from errors or unexpected inputs during a conversation with the virtual assistant. This is not done only through the use of **Fallback actions** or **out-of-scope handling**, but also thanks to the use of custom actions. For example, if a user asks a question like *"What is its genre"* without providing the name of a movie or if it's not the appropriate stage in the conversation (i.e., not after the completion of *TASK #2*), the system responds with an error message such as: *"I'm sorry, but there has been an error. Could you please specify the name of the movie in your request?"* Similarly, if a query fails to return any results due to a typo in the movie's name or if the movie doesn't exist, the system acknowledges the error and prompts the user to provide the title again. To ensure a robust and error-free system, all **slots** are reset once they have served their purpose. This precautionary step helps avoid any unintended events and maintains the system's stability.

2.2. TASK #2

As mentioned earlier, the movie recommendation task involves two different approaches for initiating the conversation, depending on the user's interaction with the agent. In this report, we will primarily focus on the first approach, as it is more complex compared to the second one.
Furthermore, the system for the first approach has been developed in two different ways based on how the user interacts with it. These two approaches have distinct design considerations and implementation strategies:

- the first approach is related to a **system-initiative system**, where the virtual assistant takes the lead by proactively providing information, asking questions, and guiding the user through a predefined flow. This approach is particularly useful when the user is interacting with the system for the first time and is unsure about what to expect. In such cases, the system assists the user by prompting them with questions about their preferences, such as the desired *genre* or *rating*.
- On the other hand, when the user is already familiar with the system or tends to provide more information than requested, the conversation adopts a **mixed-initiative** approach. This means that both the user and the virtual assistant can take turns driving the conversation forward. In this interactive dialogue, the user has the opportunity to not only respond to specific prompts but also provide additional information that may not have been explicitly asked for. For instance, in response to the question *"What kind of genre would you like?"*, the user can go beyond specifying the *genre* and also mention specific *actors* or a preferred *release date* (i,e. *"horror. I would also like to have Brad Pitt as an actor, and I prefer the movie to be released in 2012."*). This *mixed-initiative* approach allows for a more dynamic and interactive conversation, accommodating the user's preferences and providing a personalized movie recommendation experience.
  Furthermore, the system is designed to effectively handle an **over-informative** user right from the start of the conversation. It can handle scenarios where the user provides detailed information and preferences in their initial request. For instance, the user may ask a question like: *"Can you recommend me a movie? I would like it to be a horror movie with Johnny Depp as an actor, and I prefer it to be released in 2020."*.

When it comes to *Error Handling and Recovery*, at the end of the iteration we reset all the *slots* that were used aside from the *movie name*, so that we are able to ask any sort of question related to it through *TASK #1* (even without specifying the actual movie name in the query).

2.3. TASK #3

The interaction with *TASK #3*, is designed to be straightforward and efficient. This task is implemented using a **form**, which guides the conversation and ensures a structured flow of information. The interaction begins with the virtual assistant asking the user to provide the desired *plot* or theme. Based on this input, the system retrieves a list of movies that are most similar to the given *plot* or theme. Once the movie recommendations are presented, the virtual assistant asks the user whether they would like to check another *plot*. To handle the user's response and manage the conversation flow, the *form* utilizes validation techniques. If the user decides to check another *plot*, the system sets the *slot* **"is_plot_received"** to *None* and triggers the *form* again, allowing the user to provide a new *plot*. This ensures a seamless transition to the next iteration of the *form* and enables the user to explore additional movie recommendations. On the other hand, if the user is satisfied with the information received, the *form* concludes. At this point, the user can freely interact with the system by asking questions related to the recommended movies or exploring other topics of interest.

On certain occasions, users may express requests or provide input that the system has not encountered before. To mitigate potential issues arising from such situations, our agent incorporates a set of **Fall-back techniques**. These techniques are designed to address system limitations and ensure smoother interactions. Notably, we have integrated an **out-of-scope intent** to handle user inquiries that fall outside the conversational agent's knowledge domain. Additionally, a **Fallback classifier** has been incorporated into the *NLU* pipeline, which predicts the intent **"nlu fallback"** when the confidence scores of all other intents fall below the configured threshold. These measures contribute to enhanced system robustness and adaptability in handling unanticipated user interactions.

Acknowledgement and Ground-taking are also fundamental concepts in conversation design and play a pivotal role in facilitating effective communication between the virtual assistant and the user. *Acknowledgement* involves the virtual assistant's ability to recognize and validate the user's input or request, creating a sense of understanding and rapport. In this system, we have implemented acknowledgement techniques in various ways to enhance the user experience. One approach to acknowledgement is through the virtual assistant's responses, which include affirmations and clarifications such as *"OK!"* or other appropriate acknowledgements. This can also be done via the so-called *grounding by acknowledgment*, which is often achieved through the use of transitional phrases like *"And,"* which not only provides affirmation and validation to the user's input, but it also contributes to the continuity and flow of the dialogue. In *Task #2*, we have taken acknowledgement a step further by executing a custom action. This action summarizes

all the user's requests made during the conversation and seeks confirmation from the user, asking if the inquiry is satisfactory or if any changes are required.

3. Data Description & Analysis

3.1. Dialogue data for training

Dialogue data forms the foundation for training and improving the conversational agent's understanding and response generation abilities. The dialogue data, in our case, was generated through manual annotation, where I personally scripted both user inputs and corresponding agent responses. This hands-on approach allowed me to have direct control over the dialogue flow and ensured that the annotations accurately reflected the intents, entities, and dialogue context. All the dialogues have been divided between the **stories** and the **rules**. Moreover, we have a total of 43 *stories* and 71 *rules* with a average number of *steps* of 4. Also we have 21 *utterances* and 36 *custom actions*.

3.2. NLU data for training

NLU data refers to the training data that helps the *NLU* model understand and extract structured information from user inputs. NLU data typically consists of two main components: **intents** and **entities**. In fact, all the examples found in the file serve the purpose of instructing the model on how to effectively extract this valuable insights from user messages. Also, in order to enhance the model's accuracy and increase its resilience, we have included supplementary details, such as *synonyms*, to assist in the accurate identification of *entities*. *Synonyms* come in handy when users employ various expressions to refer to the same object or concept. Overall, our dataset consists of 1627 sentences. Additionally, we have documented all the *intents* and *entities*, which can be found in the appendix A.

3.3. Database, API and ML algorithms

To accomplish *TASK #1* and *#2*, we utilized the following **API**[4], which is offered by *The Movie Database (TMDb)*. This tool enables developers to access and extract movie and TV show-related data. Through this *API*, we successfully retrieved a wide range of information including movie details, cast and crew details, release dates, ratings, trailers, and more. The *API* provides an extensive selection of endpoints and parameters that facilitate querying the database and retrieving specific data based on diverse criteria. Additionally, we were able to obtain movie recommendations using this *API*. When it comes to *TASK #3*, we used the following database [3]. The dataset contains descriptions of 34.886 movies from around the world together with the release year, title, origin/ethnicity and many more information. To accomplish this task, we implemented a *machine learning framework* that takes a plot or theme as input and identifies the five most similar movies to it. We achieved this by using a **SBERT** model [1], which allows us to perform **Semantic Search**. *Semantic search* is a technique used for data searching that goes beyond simple keyword matching. It aims to understand the intent and context behind the words used in a search query. Rather than relying solely on lexical matches, *semantic search* strives to improve search accuracy by comprehending the content of the query. One crucial aspect to consider is the nature of the problem we are addressing, which is an **asymmetric semantic search problem**. In this context, we typically encounter a scenario where the user provides a short query, such as a question or a set of keywords, and our objective is to find a longer paragraph or text that effectively answers the query. In the case of our movie recommendation task, users typically input a few words to describe the desired plot or theme they want to watch and the system has to check data that is considerably longer, as the movie plots in our dataset. We achieved all of this by coupling the *SBERT*[2] model together with **FAISS** (a library that utilizes advanced **indexing** and **search techniques** to enable fast and accurate **similarity search** over large collections of vectors [5]) and **fine-tuning** it through the use of **BEIR**, a framework that enables us to **generate synthetic queries** by leveraging the plots from our dataset. The detailed description of this process can be found in my code [6]. Moreover, here you can find an image that represents the overall structure [1]

4. Conversation Model

The development of our conversational agent is built upon the powerful framework called **Rasa** which is an open-source framework that empowers developers to build and deploy conversational agents with natural language understanding (**NLU**) and dialogue management (**DM**) capabilities.
The *NLU* **pipeline** that we used consists of several components that sequentially process and extract meaning from user inputs. First of all we have *SpacyNLP*, a pre-trained model (*en_core_web_md*) for natural language processing tasks such as *tokenization*, *part-of-speech tagging*, and *dependency parsing*. This is used together with *SpacyTokenizer*, which splits user *intents* using the specified intent split symbol, enabling the identification of multiple *intents* in a single user input, and *SpacyFeaturizer* which generates features based on the *Spacy* word embeddings, capturing semantic information from the input text. The latter *featurizer* is coupled with *RegexFeaturizer*, *LexicalSyntacticFeaturizer* and *CountVectorsFeaturizer* in order to extract features using regular expressions, allowing for pattern matching and rule-based feature generation, capturing linguistic information and converting text into a numerical representation using count-based vectorization. After this we have the *DietClassifier*, a component that is responsible for intent classification and entity recognition using the *Dual Intent Entity Transformer* (*DIET*) algorithm (we also used **SklearnIntentClassifier** on a second configuration). It is trained for a specified number of epochs (50 in this case) and constrains similarities to enhance performance. At the end we have the *EntitySynonymMapper* that maps synonymous entities to a single canonical form, providing consistency and reducing ambiguity, and *ResponseSelector* which predicts the best response from a predefined set of responses based on the user input. As a fallback mechanism we used *FallbackClassifier*. It determines whether to trigger a fallback action based on a defined threshold.

The **policy configuration** that we opted for includes:

- **TEDPolicy**, which manages dialogue history, with a maximum of 7 turns and training for 50 epochs. It promotes diverse responses by constraining similarities.
- **AugmentedMemoizationPolicy**, utilizes augmented memoization for memory-based training, considering a maximum of 4 turns.
- **RulePolicy**, allows the definition of explicit rules for guiding the agent's behavior.
- **UnexpecTEDIntentPolicy**, handles unexpected intents, trained for 20 epochs with threshold-based intent distinction in *NLU* and core policy predictions.

While Rasa handles natural language understanding, dialogue management and **natural language generation** (**NLG**), we used **Alexa** for **automatic speech recognition** (**ASR**) and **text-to-speech** (**TTS**). This was done through the use of *Amazon Developer Console*.

5. Evaluation

In this section of the report, we will focus on evaluating the performance and effectiveness of the *NLU* and *DM* components of our conversational AI system. Additionally, we will conduct an **extrinsic evaluation** by gathering feedback and observing how other users interacted with the system in real-world scenarios.

5.1. NLU Evaluation

The *NLU* evaluation involved performing **cross-validation**, which automatically generates multiple *train/test* splits and calculates average evaluation results for each split. This comprehensive approach ensures that all the data is thoroughly evaluated, making *cross-validation* the most effective method for testing the *NLU* model automatically. The evaluation was conducted with a specified number of folds, which in this case was set to 5. Two different configurations were tested individually: the first configuration as explained previously [chap. 4], and the second configuration, which replaced the *DIET* classifier with *SKlearnIntentClassifier* and *CRFEntityExtractor*. The evaluation results are displayed in the provided tables [1-2]. Upon reviewing the report values, it is evident that the first configuration demonstrates superior performance in the *intent evaluation task*, exhibiting a noteworthy 4% improvement across all scales. Conversely, the second configuration shows a slight advantage in the entity evaluation task compared to the former configuration.

In both configurations, a significant challenge arises in *entity extraction*, particularly with classifying the names of *actors*, *directors*, and *movies*. This issue is evident from the *confusion matrices* presented here [2-3]. The system struggles to accurately classify these specific names, often failing to identify complete names (e.g., considering only the first name instead of the full name of a person) or incorrectly determining whether a person is part of the cast or a movie director. Notably, the system only performs well when the name is already included in the *NLU* data. To address this specific challenge, there are two potential solutions. First, increasing the number of names in the *NLU* data can help improve the system's ability to correctly classify actors, directors, and movie names. By including a broader range of names, the system can learn to recognize and classify them more accurately.
Secondly, introducing validation of names using an external *API* or ad-hoc code can be beneficial. This approach involves leveraging a system that specializes in distinguishing between actor names, director names, and movie names. To fully leverage our training data, we conducted training and evaluation on different pipelines simultaneously to enable a comprehensive comparison. The primary objective was to assess the performance of the two configurations. In particular, we plotted a graph that illustrates the mean and standard deviations of f1-scores across multiple runs (refer to image [4]). Analyzing the *f1-score* graph provides valuable insights into the adequacy of the training data for our NLU model. In the graph, we observe a continuous improvement in the *f1-score* as more training data is utilized. This indicates that incorporating additional data has the potential to further enhance the model's performance. Furthermore, the graph highlights the superior performance of the first configuration as the number of intent examples increases. By considering the *f1-score* graph and observing the trends, we can gauge the impact of training data volume on the NLU model's performance and make informed decisions regarding data augmentation or collection strategies to optimize the system's effectiveness.

5.2. DM Evaluation

To evaluate the *Dialogue Manager*, we conducted tests using both the predefined stories and additional ad-hoc stories specifically designed to challenge the system. These ad-hoc stories aimed to put the agent in difficult situations to assess its performance and problem-solving abilities. The evaluation results were highly satisfactory, as the system successfully resolved the challenges presented in the ad-hoc stories. The system demonstrated its capability to handle complex dialogues and provide accurate responses. Detailed results and performance metrics can be found on the images in the code [6]. The positive outcomes of the evaluation indicate the effectiveness and robustness of the *Dialogue Manager* in navigating diverse dialogues and adapting to different scenarios. The system's ability to handle challenging situations is a testament to its competence and suitability for real-world applications.

5.3. Extrinsic Evaluation

The system underwent testing by four independent users who were unfamiliar with its functioning. These users were asked to provide feedback on their experience and overall impression of the system. The results indicate that the users were able to successfully complete all the tasks without encountering major difficulties. However, they emphasized the specific issues highlighted throughout the report. One prominent concern raised by the users was the requirement to input sentences precisely as specified in the *NLU* in order to execute *TASK #1* through the rules. This precise requirement posed a challenge and limited the system's flexibility in understanding variations of user input. Additionally, the users expressed dissatisfaction with the system's ability to accurately recognize and classify the names of actors and directors. They found that the system often failed to capture complete names, considering only the first name or omitting surnames. Furthermore, the distinction between individuals being part of the cast or functioning as movie directors was not consistently discerned.

6. Conclusion

The development of our conversational agent *Popcorn* is built upon the powerful framework called Rasa. Rasa serves as the foundation for creating an interactive and intelligent movie assistant that caters to users' movie-related needs. Leveraging the capabilities of Rasa, we have crafted a robust and user-centric conversational agent capable of understanding natural language, engaging in dynamic conversations, and providing personalized recommendations and information. During the system development process, we conducted tests and evaluations on two different *NLU* pipelines. Our findings revealed that the performance of the **DIET** Classifier in classifying intents was more robust compared to **SVM**. However, the behavior of **DIET** Classifier in entity recognition was found to be less effective with respect to the **CRFEntityExtractor**. Additionally, during our discussions, we identified certain issues that could be addressed

and resolved in future developments.

7. References

[1] Nils Reimers and Iryna Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” Ubiquitous Knowledge Processing Lab (UKP-TUDA) Department of Computer Science, Technische Universitat Darmstadt.

[2] SBERT documentation: https://www.sbert.net/docs/publications.html, SBERT pretrained models: https://www.sbert.net/docs/pretrained_models.html

[3] Movie plot dataset: https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots

[4] API used: TMDB https://www.themoviedb.org/settings/api

[5] FAISS: (Facebook AI Similarity Search): add Similarity Search to DynamoDB with Faiss https://medium.com/swlh/add-similarity-search-to-dynamodb-with-faiss-c68eb6a48b08 For more information check: https://medium.com/mlearning-ai/semantic-search-with-s-bert-is-all-you-need-951bc710e160

[6] My code: https://github.com/Bru0x11/rasa_project

A. List of INTENTS and ENTITIES

List of intents:

- greetings
- goodbye
- affirm
- deny
- out_of_scope
- ask_movie_question
- movie_genre
- movie_cast
- movie_budget
- movie_rating
- movie_release_date
- movie_composer
- movie_director
- movie_plot
- movie_producer
- movie_runtime
- movie_revenue
- ask_recommendation_without_movie
- recommendation_without_movie_over_informative
- is_after
- is_before
- is_exactly
- just_cast
- just_director
- just_genre
- just_rating
- just_year
- just_rating+just_director
- just_genre+just_cast+just_year
- dummy_genre
- dummy_cast
- dummy_rating
- dummy_release_date
- dummy_director_name
- movie_budget+movie_rating
- movie_budget+movie_revenue
- movie_budget+movie_runtime
- movie_cast+movie_budget
- movie_cast+movie_plot
- movie_cast+movie_rating
- movie_cast+movie_release_date
- movie_cast+movie_revenue
- movie_cast+movie_runtime
- movie_composer+movie_budget
- movie_composer+movie_cast
- movie_composer+movie_plot
- movie_composer+movie_rating
- movie_composer+movie_release_date
- movie_composer+movie_revenue
- movie_composer+movie_runtime
- movie_director+movie_budget
- movie_director+movie_cast
- movie_director+movie_composer
- movie_director+movie_plot
- movie_director+movie_rating
- movie_director+movie_release_date
- movie_director+movie_revenue
- movie_director+movie_runtime
- movie_genre+movie_budget
- movie_genre+movie_cast
- movie_genre+movie_composer
- movie_genre+movie_director
- movie_genre+movie_plot
- movie_genre+movie_producer
- movie_genre+movie_rating
- movie_genre+movie_release_date
- movie_genre+movie_revenue
- movie_genre+movie_runtime
- movie_plot+movie_budget
- movie_plot+movie_rating
- movie_plot+movie_release_date
- movie_plot+movie_revenue
- movie_plot+movie_runtime
- movie_producer+movie_budget
- movie_producer+movie_cast
- movie_producer+movie_composer
- movie_producer+movie_director
- movie_producer+movie_plot
- movie_producer+movie_rating
- movie_producer+movie_release_date
- movie_producer+movie_revenue
- movie_producer+movie_runtime
- movie_release_date+movie_budget
- movie_release_date+movie_rating
- movie_release_date+movie_revenue
- movie_release_date+movie_runtime
- movie_revenue+movie_rating
- movie_runtime+movie_rating
- movie_runtime+movie_revenue
- ask_recommendation_with_movie
- ask_movie_with_similar_plot

List of entities:

- movie_name
- genre
- rating
- composer_name
- director_name
- producer_name
- cast
- release_date
- budget
- runtime
- revenue
- number_of_actors
- movie_period

A. Images & Tables

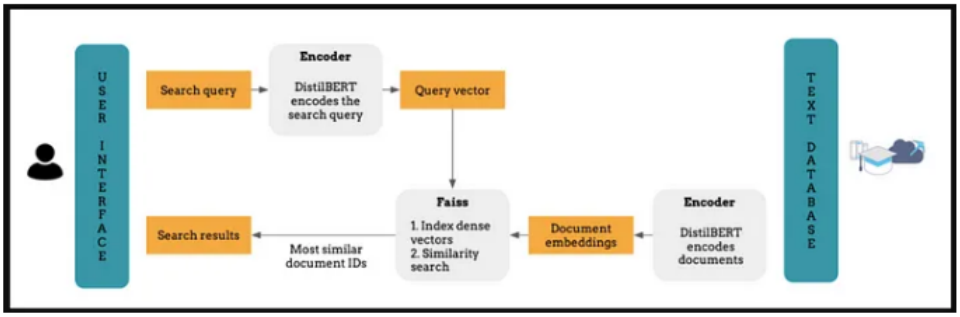

Figure 1: *Overall structure used to perform TASK #3.*

Table 1: *Intent Evaluation*

CONFIG	TRAIN ACCURACY	TRAIN F1-SCORE	TRAIN PRECISION	TEST ACCURACY	TEST F1-SCORE	TEST PRECISION
Config1	99.9%	99.9%	99.9%	88.8%	88.5%	89.5%
Config2	99.5%	99.5%	99.5%	84.0%	83.8%	85.5%

Table 2: *Entity Evaluation*

CONFIG	TRAIN ACCURACY	TRAIN F1-SCORE	TRAIN PRECISION	TEST ACCURACY	TEST F1-SCORE	TEST PRECISION
Config1	99.3%	97.8%	97.0%	96.3%	86.0%	86.8%
Config2	99.7%	99.0%	99.2%	97.1%	88.3%	89.5%

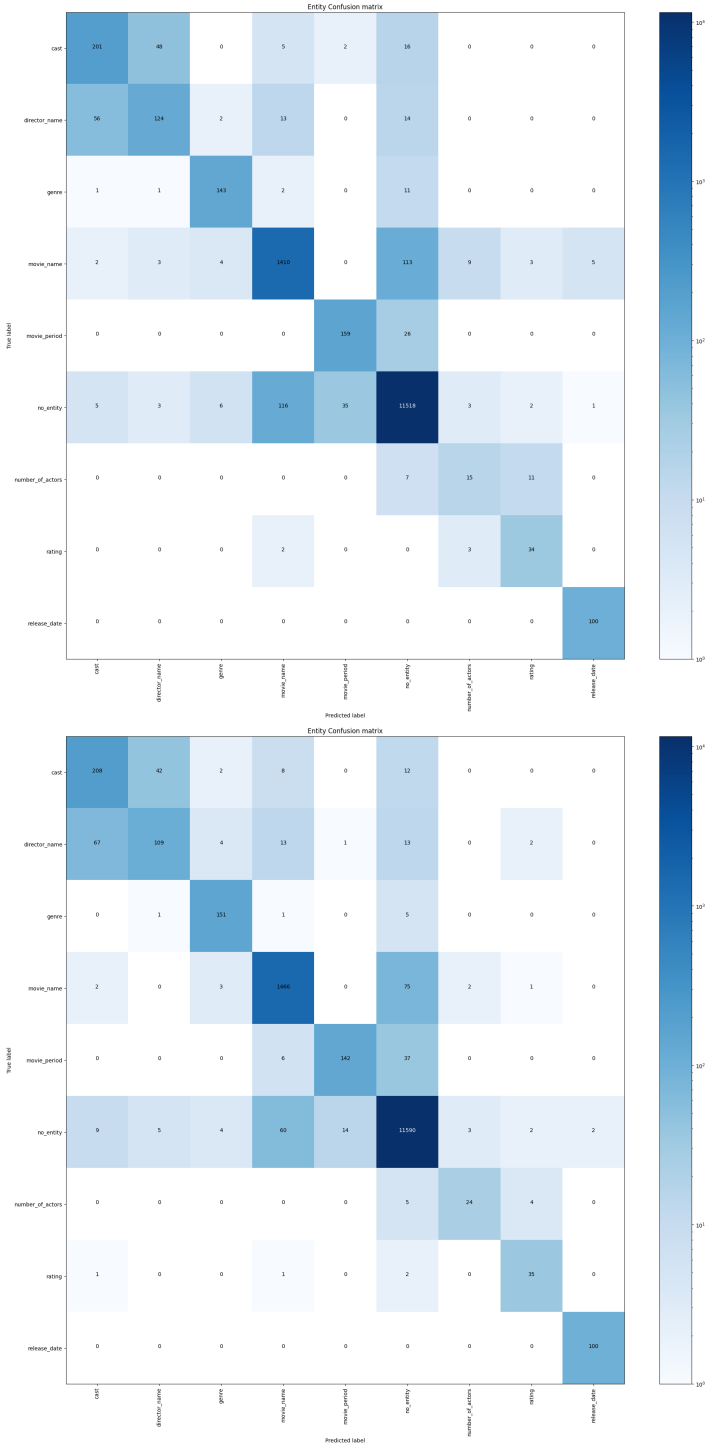

Figure 2: *Confusion matrix for the DIET classifier and CRFEntityExtractor for entities*

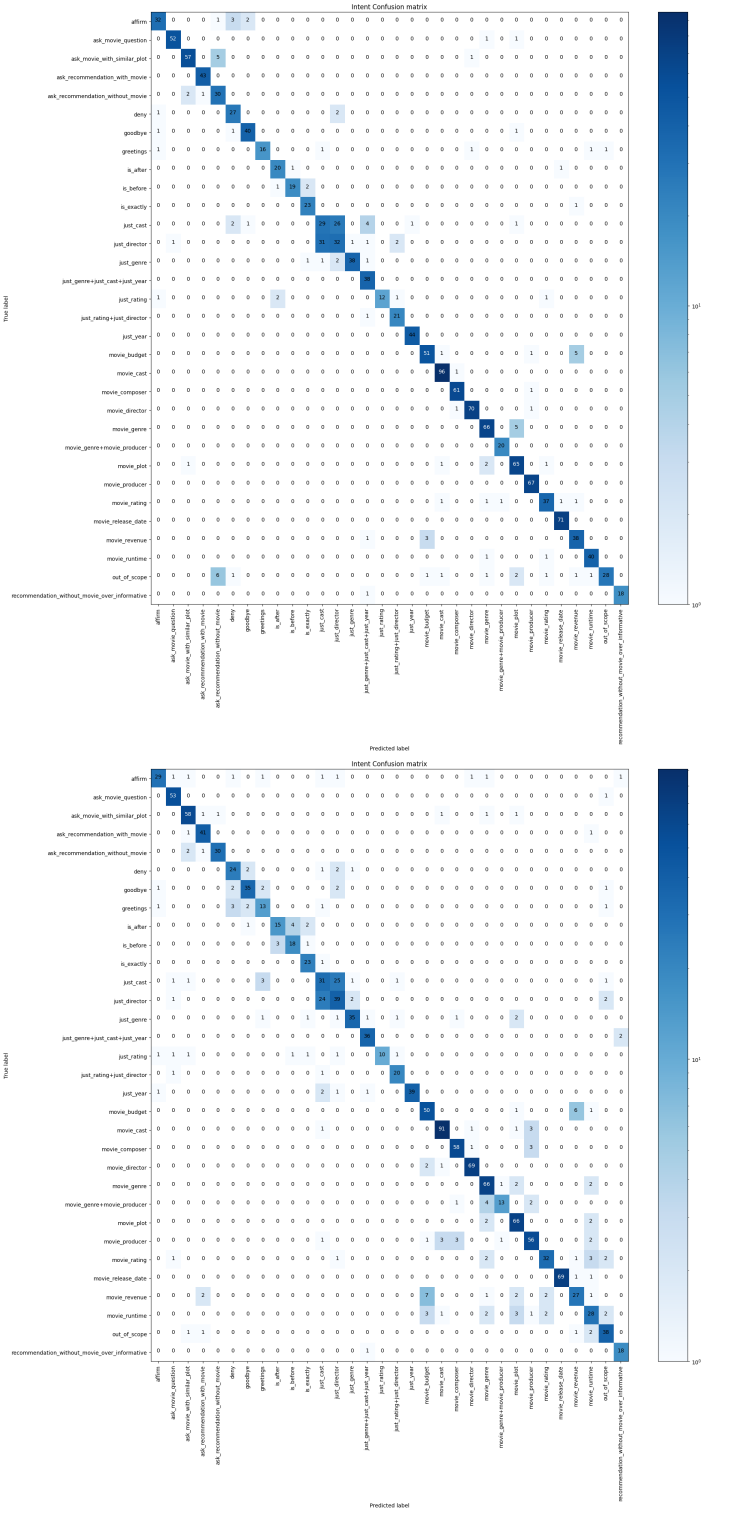

Figure 3: *Confusion matrix for the DIET classifier and CRFEntityExtractor for intents*

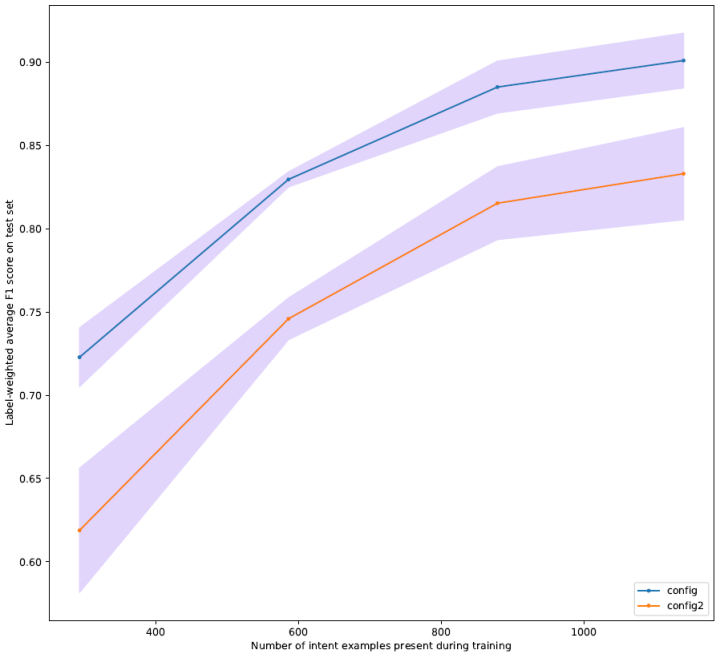

Figure 4: *Confusion matrix for the DIET classifier and CRFEntityExtractor for entities and intents.*