E-Commerce conversational AI

Moreno D'Incà [223820] and Francesco Ferrini [224010]

Human Machine Dialogue Course
University of Trento

June 5, 2022

1 Introduction

The aim of this project is to build an example of e-commerce conversational AI which is able to deal with search of products, purchases and users. The user can ask to visualize the available products with a key word (e.g. t-shirt, jacket, dress...), the AI will display the matched products alongside with their available colors and sizes. The user can also purchase new products, visualize the already purchased ones and manage the cart.
The system deployed is task based, in particular it can handle one task at a time (purchase, search, management etc...) with the ability of switching between tasks.
We leverage a GUI and Amazon Alexa skill in order to improve the user experience showing the user products via specific cards. The report proceeds explaining the tasks the model is able to carry out. Section 2 explains the dialogues the system is able to handle, section 3 describes the data used, section 4 explains the model and the development choices made, section 5 presents the evaluation and performances and, finally, we conclude with section 6.

Domain

Since the system's domain is e-commerce it can handle domain specific requests, more in details we have five areas that the user can work with:

- Products
- Cart management
- Purchase history
- Account management
- Chitchat

Products

This is the main topic, here the user can search products using specific key-words which can be retrieved querying the system which kind of products are available. Here a possible input sentence may be *"I would like to buy a jacket"*. The model will display the products with the specific images and information (such as available colors and sizes). The user can also ask further details for a specific product by saying, for example, *"May I see the details of the first product?"*. The system will display the product in all the available colors along with its description. Moreover during the purchase the system will drive the user in filling the slots which are needed for the specified product, for instance if a product comes with a predefined size or color the system will not ask for them. At the end of the slot filling the AI will ask to login using a registered email, if the given email is not in the record the user can proceed with the registration. After the login the user can choose to put the product in the cart or to directly order it.

Cart management

The cart is a simple list of previously selected items. The user can ask to visualize it by saying *"Can I see the cart?"*, the AI will reply with the list of the products currently in the cart, here the user can decide to delete a product by saying *"Delete the second one please"* or to buy everything.

Purchase history

The user may want to see its history of purchases along with the prices and dates. Following this idea the user can ask *"Can I see my purchases?"* and the system will reply with the list of bought items ordered by date. Since this is just a visualization task, the user is not supposed to make other inquires.

Account management

We register name, birth date, email and address for each account. The account management includes the creation, visualization and update. The user can create a new account by following the directives of the system after saying, for example, *"Can I create a new account?"*. The system can also ask the user to register when a new order is being placed and the email inputted at login time does not exist in the record, in this way the user will register and will be able to place the order. The account details visualization includes the name, the birth date and the address. The user can modify its stored information by asking it to the system. The modifiable data are name, birth date and address.
Note: for complexity reasons the system acknowledge USA addresses only, this can be changed by adding new data to the training set. We preferred to focus our efforts on more important improvements.

Chitchat

We add this out of domain task in order to reply to simple chitchat inquires such as *"How are you?"*, *"Tell me a joke"*, *"Do you love me?"*, *"Will you marry me?"* and *"Do you have an hobby?"*. This gives the system the possibility to answer well known queries giving the user a better overall experience. We stick with the above simple inquires, however, since the chitchat is implemented using rules, it is quite simple to add new ones.

Repository with code: Repository. Project folder containing zip of the project and videos.

Target users

The target users are possible customers of a sale company, thus the audience can be identified as male and female in the range of 16-65 years old with no specific social attributes.

2 Conversation Design

The system is task based and the initiative is mixed. The user can drive the conversation through the various tasks, then the system drives the conversation for filling the needed slots for the selected task. For instance, as stated before, when the user asks for a specific product the system will proceed on asking the desired properties of the selected item (e.g. size, color etc...). During the conversations the system will employ an implicit confirmation for acknowledgement, suppose that the user has chosen beige as color for a t-shirt, at this point the system will confirm this fact with the following possible reply: *"Beige, good choice. Which size do you want?"*, in this way the user can ask the system to change the color if wrong, then the conversation will proceed normally. Since this system is a simple exchange of information and directives the turn taking is as well simple with a mutual response among the user and the system (based on the current initiative).
The system is able to understand different type of queries when it comes to buy a product. The main entity which has to be always present is the key word of the desired product (e.g. dress), the user can ask the model to provide such list of words. Optional words are the color and size, for example the user may say *"Can I buy a beige tshirt?"*, *"I would like to see a tshirt with size XL"* or *"May I buy a beige tshirt with size XL?"*. In these cases the model will proceed by providing the list of products which match with the desired query. The system will then remember which options the user already chose and it will not ask for already filled slots, for example after saying that the desired product is a beige t-shirt the system will just ask for its size if the product needs one. This situation also apply if the user asks for further details. Sticking with the same example, if the user asks for further details of the beige t-shirt and then buys it, the system will not ask the color since the user already stated it. Since these conversations may be too long to attach here we suggest to have a look at the video demos to better understand these capabilities.
In case the model is not able to understand the user or there is lack of data the system can trigger a reset policy as fallback policy, in this way the user can start over a new conversation. The fall-back policy is implemented via the FallbackClassifier with a specific rule for the *nlu_fallback* intent, the response assigned to this intent is the *utter_please_rephrase* where the system asks the user to rephrase its query, please for further details on the fallback policy see section 4. Nevertheless in a situation in which the system is not performing well the user can always start over a new conversation by asking the system a new question (for instance *"Can I buy a tshirt?"*).
The type of conversations that this system has to handle (product search, property selection etc...) entails a close world of choices during the dialogues, for instance a specific product may have two available colors thus the user choice is bounded, this leads to a need of control over the user selection, if one choice is out of the available ones the system has to recover driving the user through an available option. One example can be seen in Fig. 1.

Figure 1: Color recovery example

3 Data Description and Analysis

The data used during the development of this project are collected using self-made conversations. We began from small conversations (15-20 exchanges), one for each task, allowing us to understand better the problem and the data we were in need. We then moved to more complex dialogues adding stories and nlu data. During the development, due to the better understanding of Rasa, we shifted our initial idea, simplifying the tasks and what the AI can handle leading us to the current setting. Moreover we leveraged Rasa Interaction which gave us a great help in understanding the errors the model was undertaking thus helping us in the creation of multiple stories in a relative short amount of time. We also used Rasa x for the overall analysis and improvements.
The nlu data are composed of 41 intents with an average of 22.76 examples per intent, with the smallest being *say_address* with 7 examples and the biggest being *choice_createNewAccount* with 55 examples. We also use lookups and synonyms in order to increase the diversity and have a better management of the nlu data. The lookups allow us to add possible values to the entities leading to less amount of examples per intent since there is no need to repeat each entry with all its possible values. We have 9 lookups ranging from 3 examples (*CITY*) to 61 examples (*STATE*). Similarly, one value may have different synonyms which are captured by the homonym component. Here we have 17 entries ranging from 2 examples (*cologne*) to 21 (*hat*).
The domain data are collected through the web-sites of different famous brands (Dior, Armani, Hugo Boss etc...), from these sources we collected general information such as products' names, images, descriptions and prices. We use SQLite as relational database storing there the data. This database includes 42 products of 10 brands. We have 28 available accounts with the possible registration of new ones using the conversational AI. The tables cart and purchases are updated as soon as new products are put in the cart or purchased. You may access the database through the path *db/database.db* of the repository.
Fig. 2 shows the database schema.
Note: we host the images using gifyu.

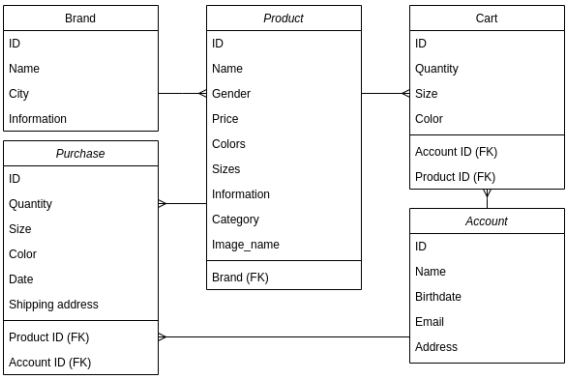

Figure 2: DB Schema

Overall system:

- Number of intents: 41
- Number of default responses: 100
- Number of stories: 169
- Number of entities: 22
- Number of slots: 32
- Number of rules: 2

4 Conversation Model

The conversation model is based on the Rasa framework with the following setting: the response policy leverages both TEDPolicy and MemoizationPolicy, the former is the Transformer Dialogue Model capable of next action prediction and entity recognition [1], the latter memorizes the stories from the training set and if the current conversation matches a story it will predict the next action based on the specified story [2]. Rasa employs a policy priority in case the predicted actions from multiple policies are different, in our case this policy gives more priority to the memoization policy. We stick with the default policy priority as suggested by Rasa [3]. Furthermore we leverage the Rule Policy to menage the cases of fall-back and chitchat [4].
The fall-back policy is implemented via the FallbackClassifier using a specific rule which maps the *nlu_fallback* intent predicted by the classifier to the *utter_please_rephrase* sentence, after some experiments we set the fall-back threshold to 0.05 [5]. This sentence will ask the user to paraphrase the query, bear in mind that in any time the user may ask a primary question (for example *"Can I buy a jacket?"*) to start over a new conversation. The rule policy also menage the chitchat queries and replies. We have 7 chitchat intents capable of understanding the sentences described in section 1. We leverage the ResponseSelector for this specific task.
We use the DIETClassifier for entity extraction and intent classification, this architecture is based on a shared transformer for both tasks [6]. We opted for the usage of Spacy for the NLU, tokenization, featurization and person name entity extraction, this choice was made following some experiments which showed better performances from Spacy (compared to other components such us WhitespaceTokenizer and others). We employed the *en_core_web_md* as NLU spacy model. In addition we use the EntitySynonymMapper component to capture the synonyms defined in the NLU data, namely for multiple entities we paraphrase them in order to diversify the possible user inputs, moreover this setting allows us to contain the size of the nlu entries since without the synonym component we should have repeated the same entry multiple times changing its entity with all its synonyms [7]. Finally we use the DucklingEntityExtractor for the extraction of the entities: time, email and quantity, this choice is driven by the complexity of these entities, duckling is already trained to recognize them leading us to its usage [8].
The interaction between the user and our model can be fulfilled using two different methods: a GUI and the voice through a custom amazon Alexa skill. The first option is a simple GUI that we found online [9], this GUI allows us to have a better interaction using cards to display the products (during purchases, cart and purchases visualization) along with buttons to choose the next action. Fig. 3 shows how the GUI displays the products, the button buy is used to buy the specified product while the details button is used to ask further details; obviously the user can still choose to type.

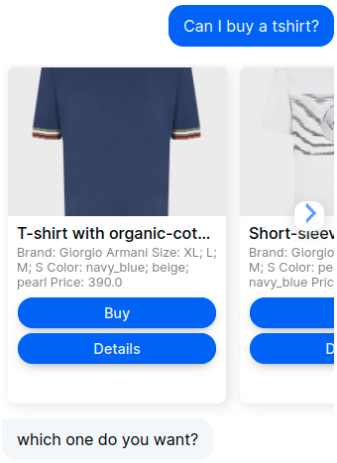

Figure 3: The GUI displaying t-shirts

Since one of the goals of this project is to use the voice we also add an amazon Alexa custom skill following the explanation during the course [10]. We customize it in order to support the Alexa Presentation Language to show the products leading to a better human-machine interaction. The Fig. 4 shows a possible interpretation of the document displayed, in the example the ratings and reviews are just random numbers to show possible features that the system may have. During testing we realized that the speech to text on Alexa Skill does not always work properly, thus we add only one video with audio.

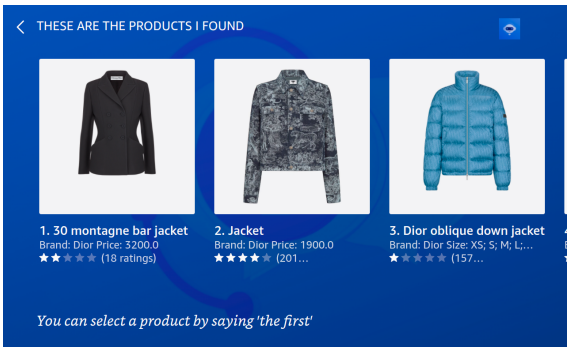

Figure 4: Alexa display example

5 Evaluation

We leverage intrinsic and extrinsic evaluations for assessing our model along side the attribute-value matrix for a simple task-based evaluation.

5.1 Intrinsic evaluation

Rasa integrates the intrinsic evaluation calculating some metrics regarding intents, entity extraction and response selection, in particular it is possible to calculate the accuracy, the F1-Score and the Precision. Table 1, 2, 3 show the results obtained from this evaluation. We can observe high quality results despite the difficulty of the tasks, for example the intent classification has interesting results even with a solit amount of intents. Fig. 5, 6 and 7 visualize the confusin matrices from the three tasks highlighting the results specified form the tables.

	Accuracy	F1-Score	Precision
Train	0.996	0.996	0.996
Test	0.943	0.943	0.950

Table 1: Intent evaluation results

	Accuracy	F1-Score	Precision
Train	0.999	0.995	0.995
Test	0.995	0.976	0.974

Table 2: Entity Extractor evaluation results

	Accuracy	F1-Score	Precision
Train	0.981	0.981	0.981
Test	0.878	0.867	0.876

Table 3: Response Selection evaluation results (chitchat)

5.1.1 Task-based evaluation (AVM)

This model is task-based therefore we perform a task based evaluation following the lectures and [11]. We focus on the purchase task since it is the core task of this model. This task has a great amount of possible values therefore we stick with a subset of them performing 58 dialogues. Table 5 summarizes the results. The model is always able to recognize the correct value, therefore the k value is 1 showing that the model has great understanding of the task it is performing (which is the goal of this evaluation). Despite the good results, sometimes the model mistakes predicting wrong NLU intents changing the task it is performing, this behavior can be improved significantly by adding and diversifying more the NLU training data.

5.2 Extrinsic evaluation

The human evaluation is a fundamental part of the evaluation pipeline to better understand the conversational AI behavior driving its development through a more inclusive human experience. We leverage a group of 10 people leaving them with the system for 10 minutes, we ask them to rate from 1 to 10 six queries reflecting the model behavior.

Table 4 reports the questions and the average rate. We can deduce that the system is not performing as well as the intrinsic evaluation meaning that there is a need of improving how the model behaves collecting more data from people. This evaluation was helpful in understanding what the users were expecting, for instance we added two new intents one for asking the system which products the AI offers and the other for asking if the products displayed are all there.

Question	Rate
Rate the ability of the model at solving the task you are asking for.	6.3
Rate the ability of the model in doing chitchat conversation.	5
Rate the ability of the system in helping to create an account.	7.8
Rate the ability of the system in ordering an item.	7
Rate how close is this conversation to a human like conversation.	6.5
Rate the velocity of the system in completing your task with respect to a human	7.2

Table 4: Human evaluation results

6 Conclusion

In conclusion we develop a Rasa based conversational AI model capable of searching products from a database providing further details when asked, following the user on buying them or adding them to its cart, registering new user as well as logging them in, managing the users' accounts and their carts and answering simple chitchat queries. The system can capture different entities during the initial phase of the ordering where the user can ask for a specific product using a key-word adding some optional details such us color and size. The system will then remember these options without the need of bothering the user again (this applies also if the user asks for details of a product during the conversation). The cart can be managed deleting products or buying everything. New users can register following the system's directives in order to be able to order products. This project gave us the possibility of understanding and working with Rasa. Initially we struggled in realizing how Rasa was learning but after several experiments we understood that the main point is adding a lot of valuable training data and conversations. The performances of the system are overall good if the user sticks with the conversations we modelled, otherwise the system struggles on handling unseen conversations. One of the main issues we encountered during the development has been the training time of the system which takes around 20 minutes with the current final setting. This issue slow us down on the expansion of the project leading us to give up in adding more diversity to the system (for example adding new ways for ordering products). We think this is an adequate final stage for a university project and an acceptable starting point for a more complex system.

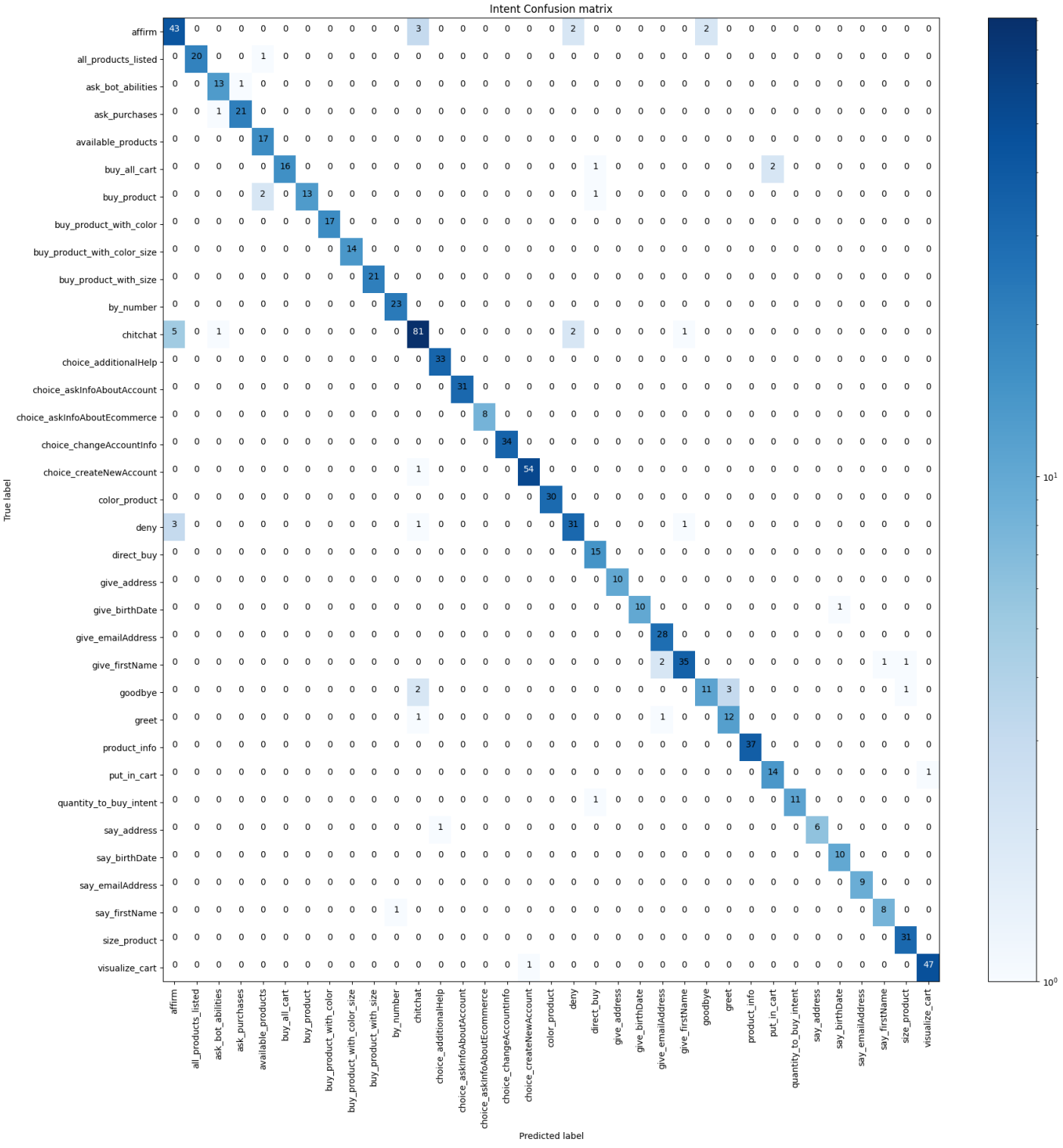

Figure 5: Intent confusion matrix

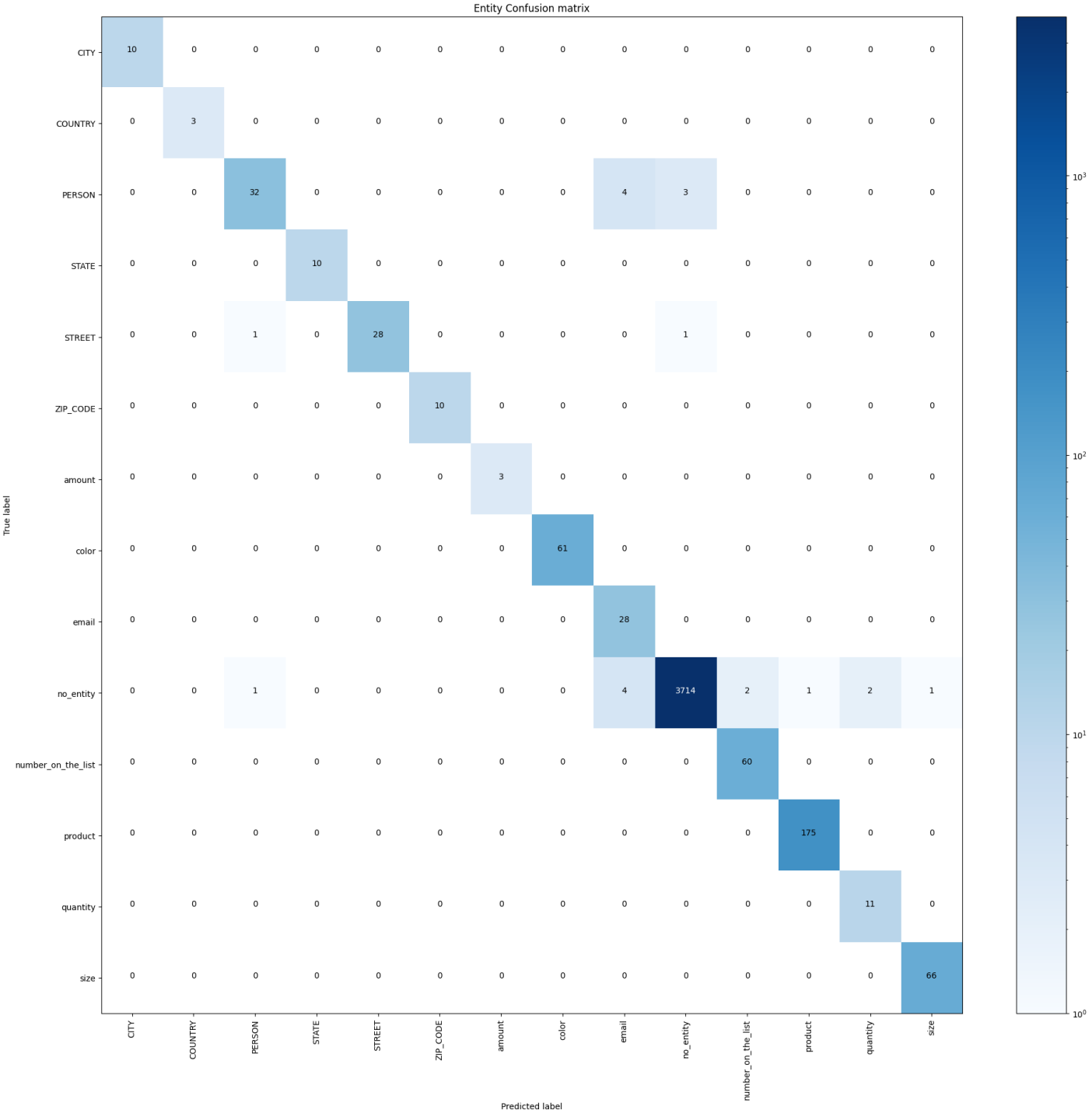

Figure 6: DIETClassifier confusion matrix

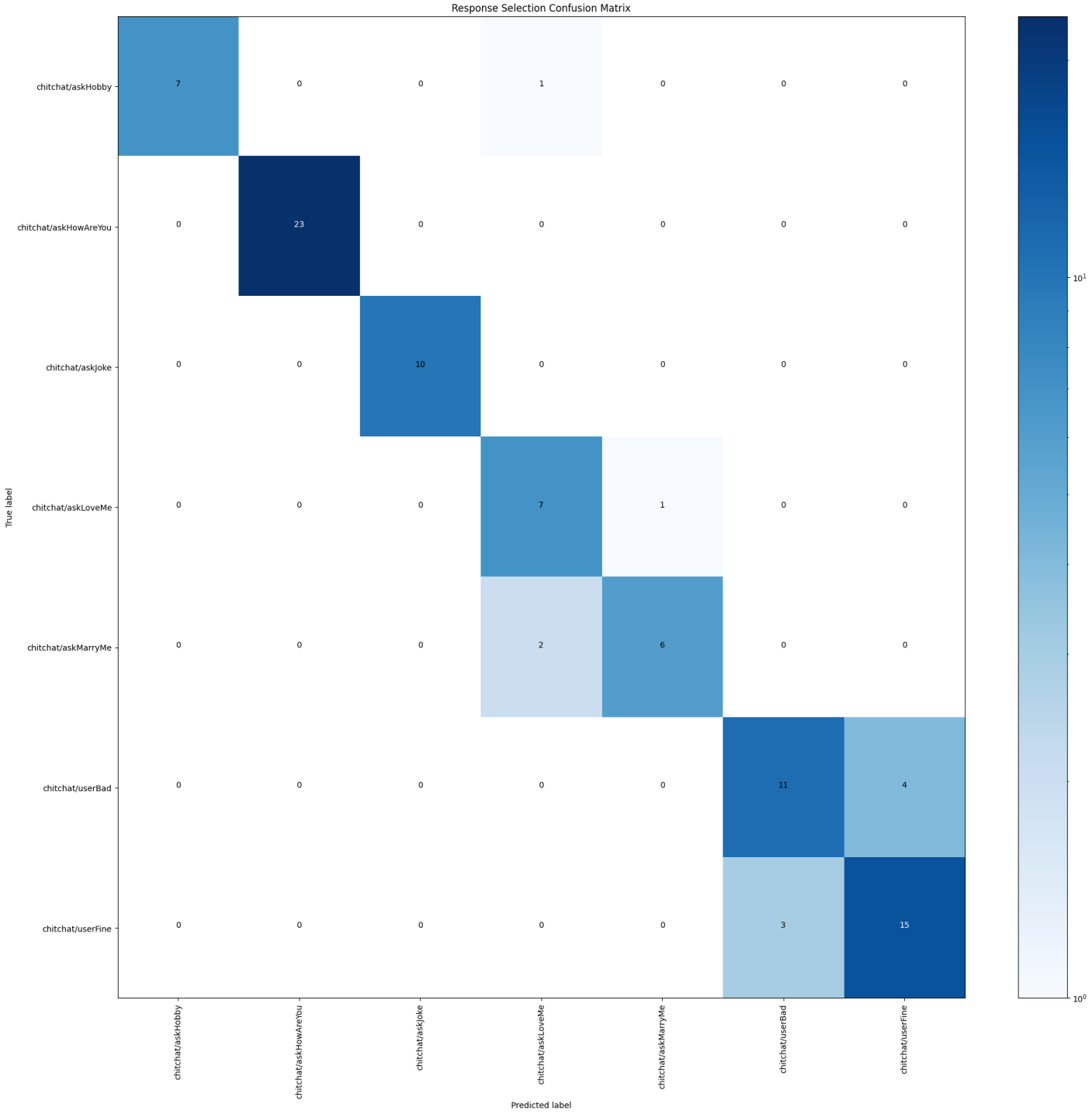

Figure 7: Response selection confusion matrix

			Key																
			Product				Color			Size									
			jacket	skirt	t-shirt	watch	pearl	white	beige	XXL	XL	L	M	S	40	42	44	46	48
System	Product	jacket	15																
		skirt		16															
		t-shirt			12														
		watch				15													
	Color	pearl					3												
		white						2											
		beige							2										
	Size	XXL								3									
		XL									4								
		L										3							
		M											4						
		S												4					
		40													2				
		42														2			
		44															2		
		46																2	
		48																	3

Table 5: Attribute-value matrix for purchase task

References

[1] Rasa ted policy. https://rasa.com/docs/rasa/policies/#ted-policy.

[2] Rasa memoization policy. https://rasa.com/docs/rasa/policies/#memoization-policy.

[3] Rasa policy priority. https://rasa.com/docs/rasa/policies/#policy-priority.

[4] Rasa rule. https://rasa.com/docs/rasa/rules/.

[5] Rasa fall-back classifier. https://rasa.com/docs/rasa/components/#fallbackclassifier.

[6] Rasa dietclassifier. https://rasa.com/docs/rasa/components#dietclassifier.

[7] Rasa synonyms. https://rasa.com/docs/rasa/nlu-training-data/#synonyms.

[8] Rasa duckling entity extractor. https://rasa.com/docs/rasa/components/#ducklingentityextractor.

[9] Chat gui. https://github.com/botfront/rasa-webchat.

[10] Rasa alexa skill class. https://tinyurl.com/rasa-alexa.

[11] Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. PARADISE: A framework for evaluating spoken dialogue agents. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 271–280, Madrid, Spain, July 1997. Association for Computational Linguistics.

Appendix

Work division

Moreno:

- Data collection (database)
- Product search
- Product order
- Product details
- Cart management
- Purchase history
- Alexa Skill and Alexa Presentation Language
- GUI
- Human evaluation
- AVM Eval

Francesco:

- Account creation
- Account visualization
- Account update
- Login and registration
- Chitchat
- Duckling
- Evaluation
- Human evaluation

Obviously we both run experiments on understanding the best overall configuration of Rasa along side with testing for general system improvement.