"Jibot": design & development of a vocal pizza ordering assistant

Jihed Dachraoui [236842]
jihed.dachraoui@studenti.unitn.it

1 INTRODUCTION

An interactive vocal pizza ordering assistant is a chatbot that allows customers to place orders for pizzas using voice commands. The chatbot is designed to handle the entire process of ordering a pizza, from selecting the size and toppings to providing delivery instructions and collecting customer feedback. It is built using the open-source Rasa [2] framework and is trained on a dataset of pizza-ordering conversations.
The purpose of this assistant, known as Jibot, is to determine if vocal AI technology, specifically an Amazon Alexa Skill [1], can effectively replace traditional ordering mechanisms for pizza shops.
Jibot is equipped to handle both task-based and question-answering requests, whether you have a straightforward question about available pizza toppings or you need assistance with more complex tasks like updating orders and creating new order lists, Jibot is here to help. Additionally, he is designed to collect customer feedback, allowing us to continually improve our services. By using Jibot, we hope to improve the convenience and efficiency of the ordering process for our customers. You can see a demo of Jibot in action at the provided link.

2 CONVERSATION DESIGN

2.1 General view

The conversation design determines the flow and structure of the conversation between the user and the chatbot. It helps to ensure that the chatbot is able to understand and respond to user input in a logical and meaningful way.
A well-designed conversation can also make the user experience more seamless and natural, which can improve the overall effectiveness of the chatbot and help to reduce confusion and frustration on the part of the user, which can lead to a more positive overall interaction.
In order to create a chatbot that is able to handle a wide range of different orders and requests, it is important to know the user's needs.
Jibot focuses on three main tasks:

1) Giving information and placing an order: This includes requests for specific sizes and toppings, such as "I'd like a large pepperoni pizza" or " I want two small Funghi)pizza" It also includes more general questions about the menu or ordering process, such as "What sizes do you have available?" or " what do you have for today?
2) Changing an existing order: This includes requests for modifications to an order, such as " I want my pizza in small size instead?" or " I changed my mind.
   Can I get a Funghi instead? " It is essential for the chatbot to understand and make changes to an existing order in a seamless and intuitive way.
3) Giving feedback: It is important for the chatbot to be able to collect customer

feedback in order to continually improve the service.
This includes asking for ratings and reviews at the end of the conversation, as well as gathering more specific feedback about the ordering process or the quality of the food.
For example, the chatbot may ask questions such as ″ On a scale from very satisfied to very unsatisfied, how would you rate your experience with our chatbot? ″ or ″ We'd love to hear your thoughts on your experience with our chatbot. what do you think about our chatbot?″ By collecting this feedback, the chatbot can help identify areas for improvement and ensure that the service is meeting the needs of customers. Feedback is saved in "results/feedback.txt".

During a conversation, it is common for a user to ask questions from different paths. If Jibot is able to answer these questions, it will do so.
Some of these questions are controlled by rules for example the rule ″exists″, says that if the user expresses the intent of ″item_type_start_negative″ (meaning that they are asking about a specific item that may or may not be available), the action ″utter_item_type_start_negative″ (i.e. "sadly, we don't offer that") should be triggered.
However, If the user provides commands that are unrelated to the current conversation, it may cause Jibot to lose track and disrupt the flow so the bot will trigger the fallback_action provided by UnexpecTEDIntentPolicy and re-ask the last question. The 3 forms that we have, pizza_order_form, person_form, and feedback_form are also controlled by rules. These forms are useful in guiding the conversation toward obtaining necessary information from the user.
The required slots ensure that all necessary information is collected before moving on to the next step in the conversation.
The rules controlling the use of these forms (i.e. activate form, submit form) help to maintain a structured and efficient conversation flow, ensuring that all relevant information is gathered in a timely manner.

2.2 Conversations

The conversation between Jibot and the user can be divided into many paths.
The first conversation flow deals with greeting the customer and initiating the conversation.
The customer is greeted, and the conversation is directed to a checkpoint where the chatbot can ask questions about pizza preferences. In the second story, the customer is also greeted, but the conversation is directed to a checkpoint where the chatbot can start taking the order.
Then Jibot deals with the actual process of ordering a pizza. first, the chatbot asks the customer to initiate their order request and then presents a pizza ordering form to the customer. The customer can then specify their desired pizza size, type, and amount through the form. The chatbot will confirm the order and proceed to the next checkpoint. Where he first presents the pizza ordering form results and then asks the customer to confirm the order. If the customer indicates that they would like to change the order, the chatbot will present the form again and allow the customer to make changes.
The following set of conversations are mixed-initiative they deal with modifying the order and proceeding with the final stages of the order process. the chatbot summarizes the order and the customer can take the initiative and make changes, the chatbot presents the pizza ordering form again, and allows the customer to make changes. On the second hand, the chatbot first asks the customer to confirm their order and if the customer indicates that they do not want to proceed with the order, the chatbot offers the option to add another order or start a new order. If the customer decides to proceed with the order, the chatbot will present a form to collect the customer's name, phone number, and delivery address, and then ask the customer if they would like to have the pizza delivered or if they will pick it up. The chatbot will then complete the order and end the conversation.

The final dialog flow deals with collecting customer feedback. The chatbot will ask the customer for feedback after the order is complete and will present a form for the customer to fill out. If the customer indicates that they do not want to provide feedback, the chatbot will thank them and end the conversation. If the customer agrees to provide feedback, the chatbot will present a form for the customer to fill out and then thank them and end the conversation.
In the pizza ordering process, the chatbot uses active loops to present forms to the customer for specifying their order and personal information. These forms allow the chatbot to gather the necessary information from the customer in a structured way. The chatbot also uses response_positive and response_negative intents to understand the customer's response to prompts and offers. For example, when the chatbot asks the customer to confirm their order, the customer can either respond positively or negatively, indicating whether they want to proceed with the order or make changes. Similarly, when the chatbot asks the customer for feedback, the customer can either agree to provide feedback or decline to do so. These intents and forms enable the chatbot to have a smooth and interactive conversation with the customer while facilitating the process of ordering a pizza.
you can find some conversations examples in the appendix Fig.1.

3 Dataset

To ensure that the chatbot is able to handle a wide range of customer requests and provide an effective and intuitive ordering experience, it is important to include a diverse range of examples in the training dataset.
The dataset used for training the chatbot consists of a collection of customer intents and bot utterances. The customer intents represent the actions or requests made by the customer, such as initiating an order or requesting more information about a particular pizza topping. The bot utterances represent the responses or actions taken by the chatbot, such as presenting a form for the customer to specify their order or providing more information about a particular topping. The dataset includes a variety of intents and utterances, covering a wide range of topics related to ordering pizzas and collecting customer feedback.
The dataset includes labels or tags indicating the intent and action associated with each customer utterance and chatbot response. This annotation allows the chatbot to learn to recognize and respond to different types of customer requests and to take appropriate actions in response.
The training NLU data is provided in the "data/nlu" path in 9 files. They contain 28 intents with 392 examples I also use rasa interactive to augment the data and add new different examples.
Training dialogues are contained in 2 yml files within the directory data/ in the project repo. we have 13 stories with 9 checkpoints and with the use of 3 forms so it Generates 249 full training stories when connected to Rasa's training engine.

4 Model

Jibot was built using version 3.1 of the Rasa library and the custom actions code was written in Python 3.8. The project file also includes a connector and JSON schema for Amazon Alexa, which hosts the Skill for voice interaction. Rasa is connected to Alexa with a ngrok [3] endpoint
In this project, I tried two different pipelines. The first pipeline uses the SpacyNLP [4] component to process the text and extract linguistic features such as lemmas, part-of-speech tags, and dependencies.
It then uses EntitySynonymMapper to map entity synonyms to a common representation and the SpacyTokenizer to split the text into tokens.
The SpacyFeaturizer is used to extract additional features from the tokens and the RegexFeaturizer is used to extract features based on regular expressions. The LexicalSyntacticFeaturizer is used to extract lexical and syntactic features, and

the CountVectorsFeaturizer is used to generate count vectors based on the n-grams of the tokens. The second instance of CountVectorsFeaturizer generates count vectors based on character n-grams.
Finally, the DIETClassifier is used to classify the intent of the text and the ResponseSelector is used to select an appropriate response.

The second pipeline has the same components as the first one, except for the removal of the SpacyFeaturizer and the replacement of the SpacyTokenizer with the WhitespaceTokenizer. The WhitespaceTokenizer divides the text into tokens by considering the whitespace between them.
In both pipelines, DIETClassifier and ResponseSelector are trained for 200 epochs.
Jibot uses four different policies for dialogue management: TEDPolicy, AugmentedMemoizationPolicy, RulePolicy, and UnexpecTEDIntentPolicy. The TEDPolicy is a supervised learning-based policy that uses the DIET model for training and considers the last 7 user messages in its prediction of the next action. The AugmentedMemoizationPolicy is a rule-based policy that searches its history for a matching user message and returns the associated action if found, only considering the last 4 user messages. The RulePolicy is also a rule-based policy, executing actions based on predefined rules that match user messages to specific actions. Finally, the UnexpecTEDIntentPolicy is a hybrid policy that combines the TEDPolicy and the RulePolicy, using a combination of the DIET model and predefined rules to determine the next action to take. It has various threshold parameters set and the fallback_action_name parameter specifies the action to be taken if the policy is unable to determine a valid action. TEDPolicy and UnexpecTEDIntentPolicy are both trained for 70 epochs.

5 Evaluation

5.1 System internal evaluation

To evaluate the performance of the chatbot, I used Rasa's built-in testing command. This allowed me to test both the NLU (intent prediction) and the dialogue management components. The metrics calculated by Rasa include accuracy, precision, recall, and f1-score. I performed this evaluation for both the first and second NLU pipelines described earlier. The results of this evaluation can be found in the "results pipeline 1" and "results pipeline 2" folder of the chatbot's repository.
Using the rasa test nlu command with specifying both the testing and the training paths (the split is performed with the rasa data split command and the .yml files are stored in "tests/").
The results are displayed in table 1, you can find also in the appendix the confusion matrices.

METRIC	PIPELINE 1	PIPELINE 2
TRAINING SET		
ACCURACY	0.972973	0.972973
PRECISION	0.967905	0.959459
RECALL	0.972973	0.972973
F1-SCORE	0.972973	0.963964
TESTING SET		
ACCURACY	0.969283	0.955631
PRECISION	0.961626	0.952043
RECALL	0.969283	0.952043
F1-SCORE	0.963846	0.952043

Table 1 intent evaluation results for both pipelines

Also, I evaluated Jibot's dialogue management components using the command rasa test core. The stories test set contains 7 stories exported from rasa interactive command.
The results for the test set are shown in Table 2 and the story confusion matrix can be found in the Appendix. The results are good, with the next action being correctly predicted in 85% of cases (6 out of 7). It should be noted that the results for the response selection components are the same for both tested pipelines because the response selection components are the same.

METRIC	BOTH PIPELINES
ACCURACY	0.984293
PRECISION	0.976828
RECALL	0.984293
F1-SCORE	0.979875

Table 2 stories evaluation results

5.2 Real life evaluation

In evaluating the performance of the Rasa Pizza Bot, we conducted user testing with 4 friends. The results of the user testing showed that the Rasa Pizza Bot was able to effectively facilitate the ordering of pizzas through natural language conversation.
The slot-filling feature was particularly useful in gathering the necessary information to complete the order.
User testing identified the need for the ability to place multiple orders as a potential improvement to the current system, which currently only allows for the placement of a single order at a time.

6 Conclusion

In conclusion, the pizza ordering vocal assistant developed in this report demonstrates promising results in accurately understanding and fulfilling customer orders. The use of multiple dialogue management policies allows the assistant to effectively handle a variety of user inputs and effectively complete the ordering process. The trained model performs well on the test set, with high scores in both intent and next action prediction. While there are a few areas for improvement, Jibot shows potential as a convenient and efficient means of ordering pizza through voice commands.

Appendix

Extra content

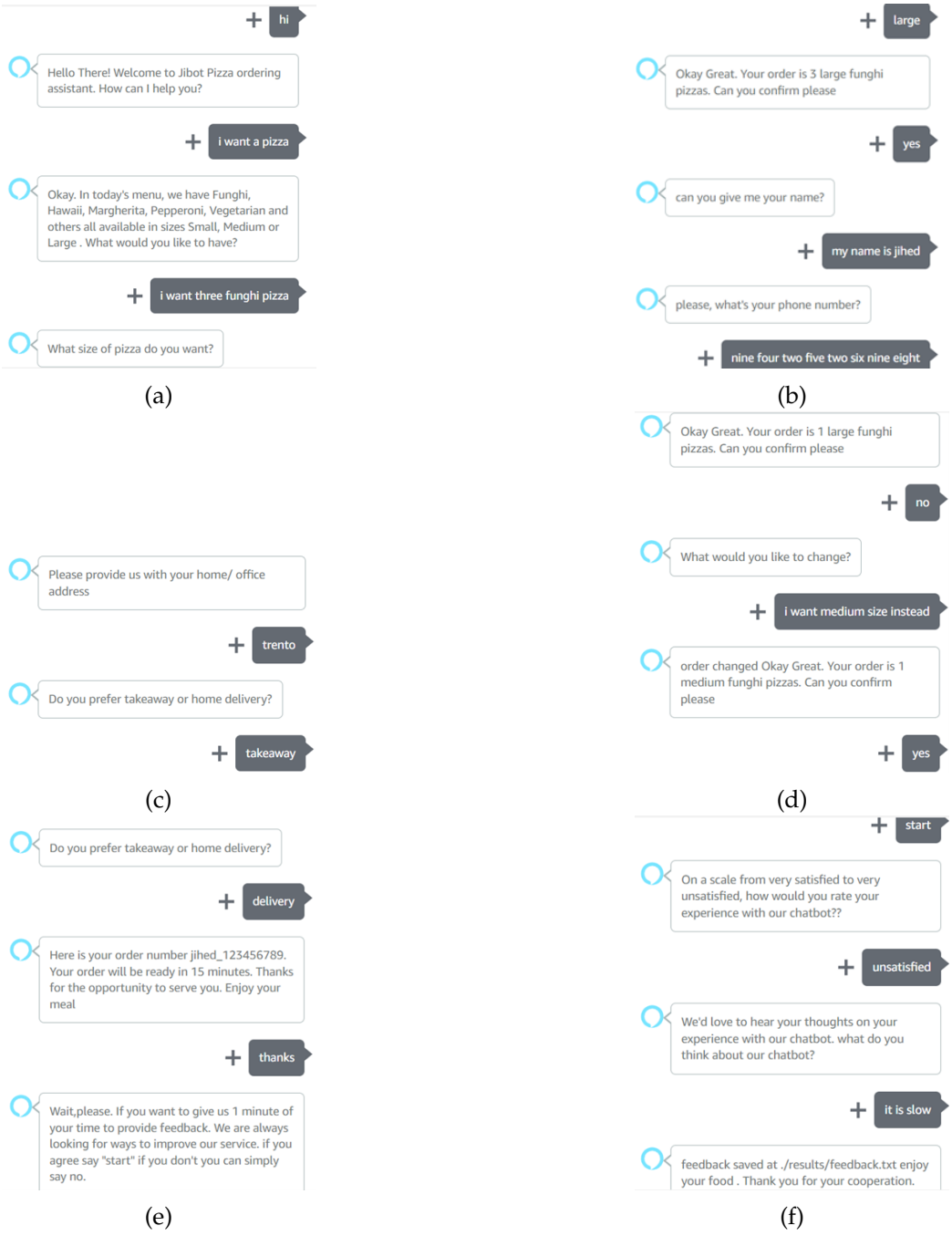

Fig. 1: Some screenshots of conversations in Alexa interface

a Greeting the customer and initiating the conversation
b Confirming the order and starting person_form
c Choosing order picking method
d Starting changes in the order
e Thanking the user and asking for feedback
f Collecting user feedback and saving it using feedback_form

Intent confusion matrices

The following figures show the intent confusion matrices for both tested pipelines. The main pipeline (pipeline 1), shown in Figure 2, has fewer errors than the second pipeline matrix presented in Figure 3.

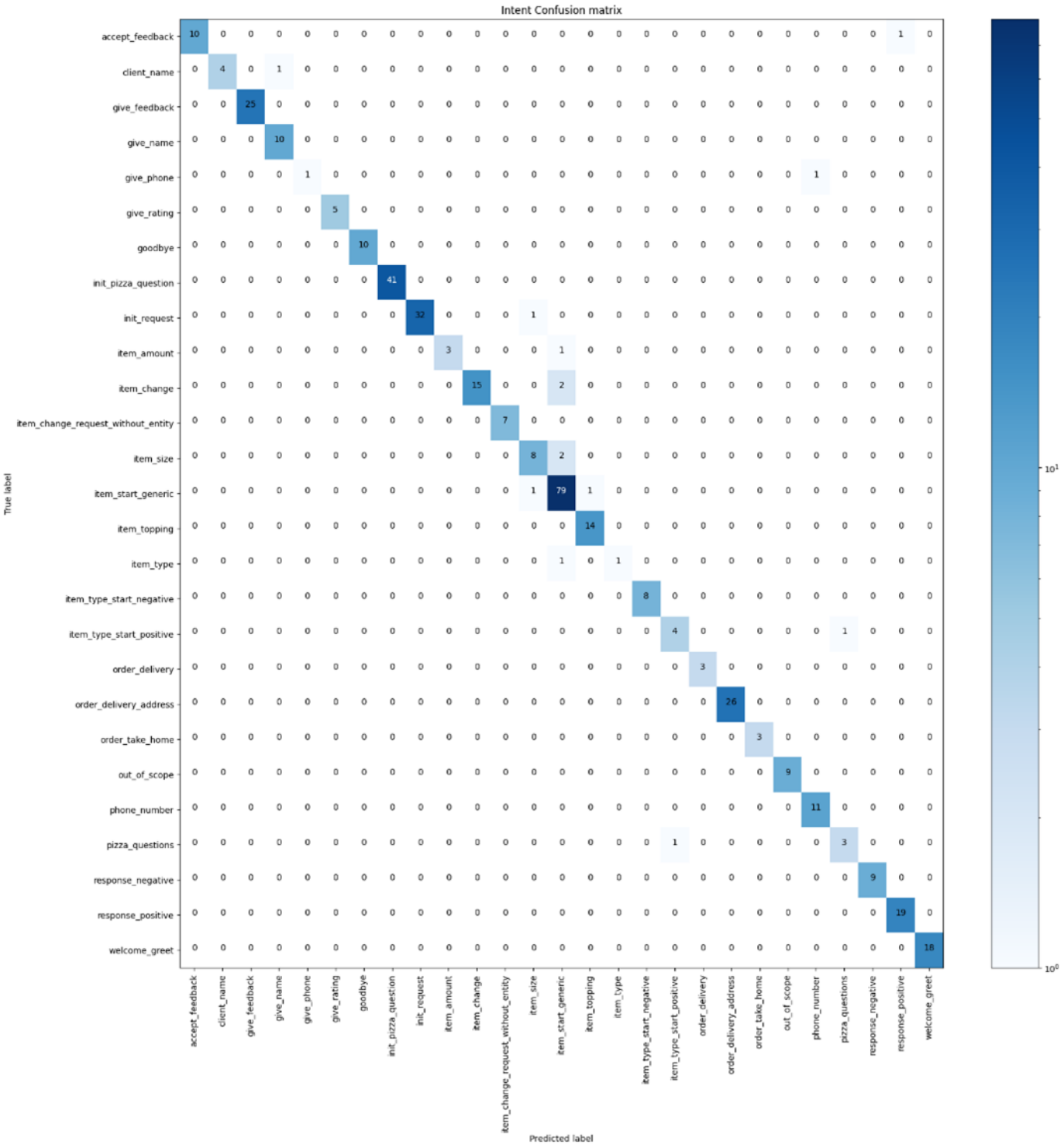

Fig. 2: Intent confusion matrix of pipeline 1

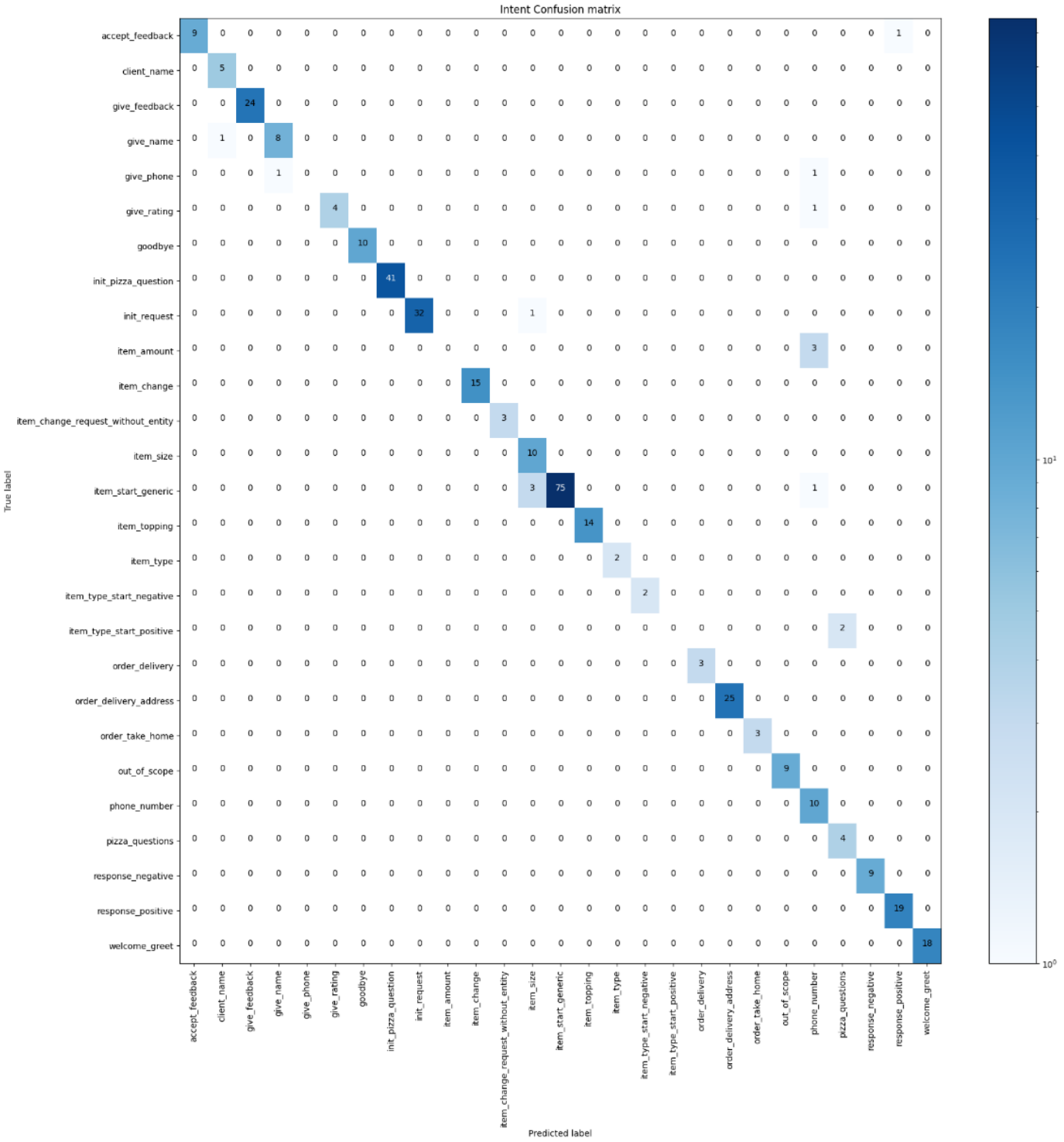

Fig. 3: Intent confusion matrix of pipeline 2

Story confusion matrices

The action confusion matrix in Figure 4, generated by the rasa test core on the test stories set, is shown below.

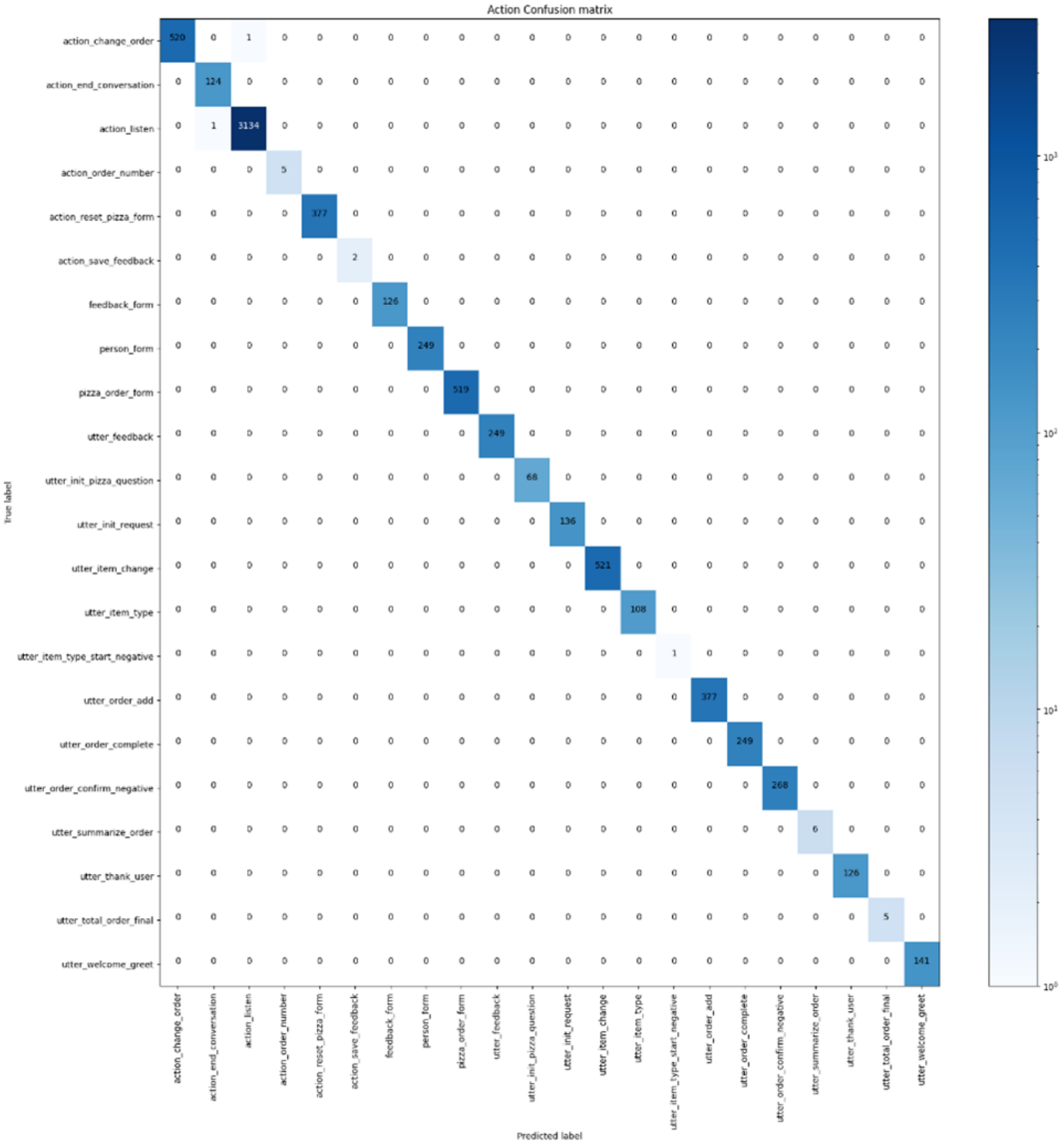

Fig. 4: Story confusion matrix

REFERENCES

[1] Amazon, “Alexa Skills Kit,” [Online]. Available: https://developer.amazon.com/en-US/alexa/alexa-skills-kit.
[2] Rasa, “Homepage - Open source Conversational AI,”. Available: https://rasa.com/..
[3] “Homepage,”. Available: https://ngrok.com/.
[4] “Homepage,”. Available: https://spacy.io/.