Figure 1: Herbie's logo.

"Herbie": design & development of a vocal Inventory Management System for a small Herbalist shop

Francesco Trono (221723)
University of Trento
A.Y. 2021/22 – Human-Machine Dialogue course
francesco.trono@studenti.unitn.it

1. Introduction

Through this project, an experimental use case for vocal AI technology is tested: could an Amazon Alexa Skill [1] replace an inventory management software for a small local shop? In this report, I propose a design for an interactive vocal assistant based on Rasa [2] and able to perform multiple tasks in the *inventory management* domain through simple conversations, among them: managing inventory stocks, giving information on products inventory, creating and updating order lists, registering orders delivery, producing printable warehouse views. For the tasks that cannot be performed by voice (i.e. insert product data, receive printable files), I developed a companion Telegram Bot [3] [4], hosted in Heroku [5] (details in *Appendix A2*). The result is an interactive and – especially – proactive shop assistant I named *Herbie.* Being Alexa's voice feminine, I gave Herbie a female characterization (see logo in Figure 1), therefore I will refer to it hereinafter using the pronouns "she" or "her". Herbie's full service combo is detailed in Figure 2 below.

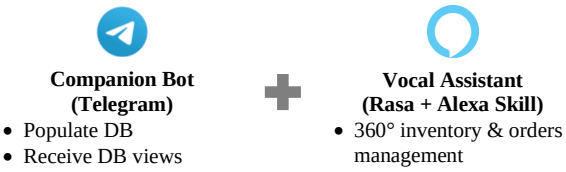


Figure 2: the full Herbie experience combo.

Herbie has been designed for and tested in a real use-case scenario: a small herbalist shop located in the town of Copertino (Lecce, Italy) [6]. For this reason, her conversation language is Italian. Her goal is to work as an intermediary between the shop manager and the shop's warehouse, acting like a vocal interface for the former to talk to the warehouse or receive suggestions by it. Therefore, her target user is the shop manager.

Specifically, Herbie's domain is inventory management with a focus on the herbalist shop world: based on some custom DB fields, she is able to answer to specific questions about products (i.e. "is it a medical device?" or "is it vegan?"). Herbie is not supposed, anyway, to take the place of the shop manager or interact with the final customer, so she has *not* been trained on any technical data that is specific to the herbalist sector.

Depending on the specific question that is asked, Herbie is both a *task-based* and *question-answering* assistant: she can answer to direct standalone questions (i.e. product/stock info), but she has a specific scheme to follow to fulfil other requests (i.e. update warehouse, create order lists, register order delivery).

2. Conversation Design

Herbie has been designed on 3 main principles: *helpfulness*, *ease of use* and *proactivity*. In a real use scenario, it was essential that she could handle different kinds of conversations, modelled on specific everyday shop management situations. With this value-targeted design in mind, she does not offer chitchat functionalities: some common questions like "come va?" or "come stai" are mapped directly to the *ask_guidance* intent.

2.1. General lookup scheme: product & supplier names

Since Herbie has been designed to work in a real shop context, I could not provide her with a fixed list of inventory to learn: the list of products normally changes month by month, with new products launched, new suppliers from which the shop is refurnished and old products being replaced over time.

Due to the need of keeping Herbie "data agnostic" and the DB open to new additions, I decided *not* to train her to recognize product names and supplier names as learnable *entities*. Instead, I trained her to know exactly *when* to listen for a product name or supplier name. Herbie always asks *first* the user to say the requested product or supplier name, *then* she will take the full user reply text (a generic intent is provided for this turn, named *"inform_text"*) and look up for a match in the DB using the Levenshtein distance [7], integrated in the Python library FuzzyWuzzy [8]. If more similar matches are found with a confidence delta lower than 10%, Herbie reads the best 2-3 matches to the user, asking him either to pick one of the three options (i.e. "il primo") or say the name again. This whole process is managed via a Rasa form: Herbie will keep requesting the name until either (a) a single match is found, (b) the user picks one of the matches suggested, or (c) the user terminates the conversation.

2.2. Terminating loops and conversations

Conversations and loops can be stopped through one of two intents:

- *Stop*: managed by Alexa, it terminates the entire Skill (i.e. "basta", "stop", "chiudi");

- *Completed*: managed by Rasa, it ends only the current loop or conversation, depending on the case (i.e. "ho completato", "finito", "fatto").

Herbie proactively guides the user on what to say time by time.

2.3. Conversations

The conversations can be grouped into 3 main areas:

1. Find a *product*
2. Find a *supplier* + prepare an *order*
3. Update *warehouse* + ask for DB views

The *initiative* alternates between Herbie and the user in all areas. The first 2 areas require specific entry points ("trova un prodotto" or "trova un fornitore", with their possible variants) that trigger the general lookup scheme; once a product or supplier has been identified, the user has a variety of alternative options to ask for any specific service he needs. Please note that, when Alexa is idle, the Herbie Skill must *first* be activated using the wake up words *"assistente negozio"* (i.e. "Alexa, apri assistente negozio").

The full conversations schemes and the detailed description of the possible flows for the 3 areas can be found in *Appendix A3*. Here follows a short but complete overview.

Area 1 (*find a product*) is the most basic of the three areas: it is a very simple *Q&A* session that is mainly *user-driven*, with the user asking various questions on the product and Herbie answering. The idea here is to make Herbie a valid replacement for the times in which the shop manager must go read the product info, price, details etc from the supplier booklet. The user can also ask Herbie which is the available stock quantity: in this case, Herbie takes the initiative and proactively asks the user if he wants her to add the product to the next order list.

Area 2 (*find a supplier*) is the most complex of the three areas. It is strongly *mixed-initiative* and follows 3 separate paths: (i) order preparation, (ii) supplier & order requests, (iii) register a delivery. Path (i) is an ordered sequence of steps (read the list, suggest products running out of stock, dictate a list, mark a list as closed), with Herbie proposing each step to the user and the latter deciding to enter it or fork the conversation according to what he needs. Path (ii) contains general requests to Herbie regarding the supplier and the order (i.e. receive an order list or a warehouse view via Telegram Bot, mark an order list as closed). Path (iii) allows to register the delivery of an order previously created via Herbie, with the latter asking confirmation to the user before automatically updating the DB following the closed list.

Area 3 (*warehouse area*) has no specific entry point. The user can ask either to (i) receive a warehouse view via Telegram Bot, or to (ii) update the stock quantities in the warehouse (i.e. at the end of a day in the shop). Path (ii) is mainly *system-driven*, with Herbie asking to the user the list of products and the quantities sold/received. But if Herbie finds a closed order list in the DB, she will *first* proactively ask to the user if he just received the delivery of an order (i.e. "Ti è stato consegnato un ordine?"): if he affirms, she will proceed with the registration of the delivery as seen for area 2, path (iii).

Please refer to *Appendix A3* for the details on all 3 areas.

2.4. Interaction properties

Due to the complex nature of the interactions that Herbie can manage (the initiative is given to Herbie, to the user or to both depending on the specific point of the conversation), I prioritized for Herbie the need to *explain*, to *guide* the user on each step. This not only helps, but it gives the user the idea of a very collaborative and friendly assistant (section 5.2). The user can ask for guidance in *every* conversation path (i.e. "Come funzioni?"), she will reply differently according to the path she is in.

The majority of Herbie's replies start with "ok", "va benissimo" or other acknowledgement words or sentences that communicate to the user that Herbie *understood the command*. In order to make Herbie more "human-like", she never directly repeats what the command was: she instead provides context on what she understood via brief sentences that introduce the user to the conversation (i.e. "Ok, sono pronta a preparare l'ordine!"), or through guidance (i.e. "Va benissimo: dimmi il nome del prodotto.").

Ground-taking is also achieved in different ways depending on the importance of the information: for the general lookup scheme, the name of the supplier and/or product Herbie found is read aloud (i.e. "Ok, di Aboca, Pappa Reale bustine"), while other information obtained within a loop are generally integrated into a recap of what she is doing (i.e. "5 pezzi aggiunti! Continuiamo.", or "Ok, ignoro. Andiamo avanti.").

The majority of the cases in which Herbie *must* get an information to proceed are managed via Rasa Forms: Herbie will keep asking the question to the user until she gets the right kind of data to populate the requested slot.

Regarding over-informative cases, NLU entities are not managed by Rasa, therefore Herbie will simply ignore names of products / suppliers and numbers that are placed out of context. But there is an exception: when Herbie is reading an order list (via the *read_order_form*), Herbie will ask the user if he wants to keep each single product on the list: the user may confirm, deny or may even directly ask a different quantity of pieces to order (i.e. "Facciamo 10 pezzi"). Herbie will update the order list accordingly.

Ambiguous input is managed in two ways: (a) with the general lookup scheme (section 2.1); (b) using confidence thresholds in the NLU configuration. Confidence thresholds are also used to predict a single intent when it contains multiple requests. When Herbie does not understand, or does not map an intent with enough confidence, or does not find what the user is looking for, she asks the user to repeat. The lack of user input is not managed by Rasa, but it is left to the default management setting provided by Alexa (i.e. repeat last Herbie's utterance twice, then terminate the skill).

In many cases, the user can ask questions from other paths in the middle of a conversation – if Herbie can answer, it will. When the user gives commands out of context, though, the conversation flow might be altered and Herbie could lose track of what was happening. In these cases, I prevented Herbie from writing to the DB or giving answers when she lacks key slots: she will reply by guiding the user on the right way to ask for that command (i.e. "Chiedimi di trovare un prodotto, potrò risponderti subito dopo").

The key entry points Herbie mentions (in particular, "Trova un prodotto", "Trova un fornitore" or "Aggiorniamo le giacenze", with all their variants) erase all slots in Rasa's memory and terminate all loops, so that, if the user uses them, the conversation can start over without interferences from the previous flow. Furthermore, within a Form, the slots requested in each turn are erased before starting the next turn, in order to avoid interferences in the loops.

When these prevention measures are not enough or Herbie understood something wrong, the user can instinctively use the *stop* intent to terminate the skill or directly ask Herbie to *restart* (i.e. "Ricominciamo", "Riavviati). The latter intent calls Rasa's default action *"/restart"*, which instantly erases all slots and terminates the current conversation; Herbie will say to the user that she is ready to start over.

Within the *write_order_form* and the *variations_form* the user is requested to dictate the names of products: it may happen that Herbie picks up the wrong product. The user has the possibility to reply by denying (i.e. "no, questo no" or "no, sbagliato") and Herbie will ask the user to dictate a new product.

When some operation fails due to an error, Herbie instantly tells the user that there was a problem (i.e. "C'è stato un problema, ti chiedo scusa."). In these cases, if the error happens within a loop, Herbie goes to the next loop without interrupting the conversation, otherwise the conversation is generally automatically terminated by a *rule*.

3. Data Description & Analysis

3.1. Training data

Herbie has been trained on conversational data that is based on everyday situations I observed in Copertino's shop: the dialogues written in the Stories are a mixture of (i) adaptation to Rasa's capabilities of the human dialogues I recorded, and (ii) formalization of dialogues with an imaginary agent that intervenes to give information to the shop manager when I observed that he needs it.

Training dialogues are contained in 4 yml files within the directory *data/stories* in the project repo. The files contain 38 pieces of Stories that builds together like Lego bricks, branched through 9 *checkpoints* that, when reconnected by Rasa's training engine, create 311 full training stories. The 38 pieces contain approximately 93 exchanges and make use of 6 Forms.

The training NLU data is provided in the *data/nlu.yml* file: the file contains 33 intents, with 419 samples in total (12.7 samples per intent on average, from a minimum of 5 to a maximum of 39), 344 words and no learnable entities, since I decided not to manage them through Rasa (paragraph 2.1). This happens not only for product and supplier names, but also for the number of pieces: unfortunately, the Italian speech-to-text engine built in Amazon Alexa translates by default into numbers any article (i.e. "un", "uno") or verb (i.e. "sei") that sounds like a number. For this reason, the intent "inform_pieces" was triggered in random points of the conversation, misleading the whole flow. I therefore decided to build custom-made extractors inside the slot validation actions of the Forms that needed them.

Herbie uses 31 slots to keep track of what happens within a conversation, 7 of which extracted automatically by Rasa (i.e. using *from_intent* or *from_text*) and 24 via custom actions in Python. 5 slots are used to fork conversations in Stories (*influence_conversation = True*).

The *domain.yml* file contains 30 default responses, each present in 2-6 alternatives. Other possible responses are written in the Python code of the custom actions and include error messages, information reading templates or answers that depend on variables.

Herbie uses 27 custom actions and 6 Forms (each with its own validation actions). All actions are declared into the *actions/actions.py* file and call support functions that are written in the other Python files present in the *actions* directory.

3.2. Database

Herbie's knowledge base is the actual warehouse of the shop, represented by a relational database hosted in Heroku's PostgreSQL instance [9]. The DB is shared with Herbie's Telegram Bot and can be populated and updated through the latter depending on the shop's needs. The DB design scheme is reported in *Appendix A1*: it contains tables for Suppliers, Categories, Products, Order History and Order Lists. This design can be repeated for many shops, each listed in the global table Schemi: for testing purposes, one Schema only has been created, with the name "Test", which contains 22 actual products sold by the Copertino shop, belonging to 4 suppliers and mapped into 3 categories.

4. Conversational model

Herbie has been built using version 3.0 of Rasa library [2], with the custom actions code written in Python (version 3.8) [10]. Herbie's repo contains also the connector and the *json* schema for Amazon Alexa, which hosts the Skill, for voice interaction. All tests in Copertino's shop have been done by connecting Rasa to Alexa using a *ngrok* endpoint [11].

During the development of Herbie, I tested two alternative NLU pipelines using different components of the spaCy library [12]. Both of them load the pretrained model for the Italian language ("*it_core_news_sm*"), then the first pipeline ("*main"*) makes use of SpacyTokenizer only, the second pipeline (*"alternative"*) of both SpacyTokenizer and SpacyFeaturizer. Both pipelines are then followed by Rasa's default featurizers: the RegexFeaturizer, the LexicalSyntacticFeaturizer and the CountVectorsFeaturizer. The chosen classifier is the DIET Classifier and it is used by Herbie only for intent classification, since there are no learnable entities. The default ResponseSelector is used for response prediction. Herbie uses the FallbackClassifier for the cases in which the NLU pipeline predicts an intent with low confidence (threshold: 0.3). In this case, a specific rule is provided in the *data/rules.yml* file to make Herbie ask the user to rephrase. After testing both pipelines, I decided to keep the one using the SpacyTokenizer only (*main* pipeline), considering the better intent prediction capabilities it demonstrated during live tests in Copertino's shop.

The chosen Policies configuration makes use of: (i) the MemoizationPolicy, to learn the Stories provided; (ii) the RulePolicy, to exploit the 8 rules for fixed behaviour I specified in the *data/rules.yml* file; (iii) the TEDPolicy (Transformer Embedding Dialogue), for next action prediction. For TEDPolicy, it became apparent from the first tests with longer Stories that Herbie needed to have more "memory", meaning that it had to keep track of more dialogue turns. After testing various values for TEDPolicy's hyperparameters, I achieved the best results with 10 turns cached in *max_history* and 50 training epochs.

5. Evaluation

5.1. Intrinsic evaluation

For intrinsic evaluation, I used Rasa's built-in *test* command, which allows to evaluate and test the NLU (i.e. intent prediction – no entity extraction in this case) and dialogue management components. The metrics calculated by Rasa are accuracy, precision [13], recall [14] and f1-score [15]. I have performed the intrinsic evaluation for both the *main* and *alternative* NLU pipelines introduced in section 4. Results are provided in Herbie's repo in the "*results"* folder.

To test Herbie's NLU components, I used the *rasa test nlu* command on both the training set and a test set (*tests/nlu_test.yml*) containing 3 new samples per intent (99 in total) that are the most ambiguous ones I collected during the tests in Copertino's shop. For both the *main* and *alternative* pipelines, the resulting evaluation metrics are reported in Exhibit 1, while in *Appendix A4* their intent confusion matrices for the test nlu set are reported, showing the better prediction capabilities of the *main* pipeline.

Metric	***main*** **pipeline**	***alternative*** **pipeline**
Training set		
Accuracy	0.9952	0.9952
Precision	0.9958	0.9958
Recall	0.9952	0.9952
F1-score	0.9953	0.9953
Test set		
Accuracy	0.9091	0.8586
Precision	0.9192	0.8818
Recall	0.9091	0.8586
F1-score	0.9079	0.8543

Exhibit 1: intent evaluation results for both the *main* and *alternative* pipelines.

I then evaluated Herbie's dialogue management components on a series of test stories I collected during the tests in Copertino's shop, using the command *rasa test core*. I provided 6 test stories, with 58 exchanges in total. The results on the test set are reported in Exhibit 2, while in *Appendix A5* the action confusion matrix is reported. We can see that also in this case the results are very good and the next action is correctly predicted in the 95% of cases.

For the evaluation at conversational level, unfortunately, the *rasa test core* command does *not* execute custom actions: Herbie relies heavy on the outcome of many custom actions, so I had to report manually the value of each single slot populated by any action in the test stories I was writing.

Despite ensuring the accuracy of the test stories and slot values I reported (I helped myself with throwaway runs of *rasa interactive*), some small issues remained: *rasa test core* reports only 3 test conversations out of 6 completed without divergences (0.5 accuracy). If we actually look at the provided *failed_test_stories.yml* file, though, we can see that these "fails" are only related to *action_listen*, *action_default_fallback* or *action_ask_more*. These fails do *not* happen if I run the very same test conversations via *rasa shell* or the Alexa endpoint: I reported an example in the file *shell_test_suppl_order_2.log* present in the folder *results/core*, which shows no divergence in predictions with the corresponding test story file (*test_suppl_order_2.yml*).

Metric	Both pipelines
Action level (test set)	
Accuracy	0.9730
Precision	0.9684
Recall	0.9730
F1-score	0.9698

Exhibit 2: dialogue management evaluation results (same for both the *main* and *alternative* pipelines).

5.2. Extrinsic evaluation

Herbie has been designed specifically on the real needs of a shop, so I decided to directly involve 3 end-users (the shop owner/manager and 2 collaborators) and 5 external observers (all of them are nearby shop managers) to conduct a human evaluation survey. The 3 end-users sustained 3 entire conversations (one each) for each area, using various ways to ask for the same task, in front of the 5 observers. Then, I asked all 8 interviewees to give a score from 1 (lowest) to 5 (highest) to specific Herbie features. Results are provided in *Appendix A6*.

The overall feedback is very positive: *proactivity*, *exhaustiveness* and the presence of *alternative ways* to ask for the same task are the features that have been valued the most, together with the *clarity of guidance* that Herbie provides on each turn. The users found that they may need a bit of trials to fully understand and remember all that Herbie can do (*learning curve*) and thought that the schematized discourse flow, organized into separate areas and lookup schemes, might impact in this: considering that Herbie must be data-agnostic and that she had to learn 311 very long stories, this turned out to be the only feasible choice. Overall, they found speaking with Herbie incredibly *natural* – they all said there is an abyss with respect to the "dumb" chatbots they are used to speak to via telephone for utilities services.

6. Conclusion

According to the people who have been testing Herbie since May and participated to the survey, the proposed design for a vocal inventory manager works – they are *all* interested in her. They never had the possibility to speak with their warehouse like it was their assistant and said that simple things like dictating a product list, receive proactive suggestions and even receive Telegram messages after asking her via voice make Herbie feels like a real human assistant. Now Herbie has been officially hired by the Copertino herbalist shop and they are waiting for her to get to the cloud.

Appendix

Table of extra content

A1. Database scheme

Figure 3 below shows the design scheme of the DB that is shared between Herbie Vocal and Herbie Telegram Bot and hosted in Heroku's PostgreSQL instance [9]. It contains tables for *Suppliers*, *Categories*, *Products*, *Order History* and *Order Lists*. This design can be repeated for many shops, each listed in the global table Schemi. For testing purposes, one Schema only has been created, with the name "Test": Herbie defaults to it through a global variable stored in the *globals.py* file.

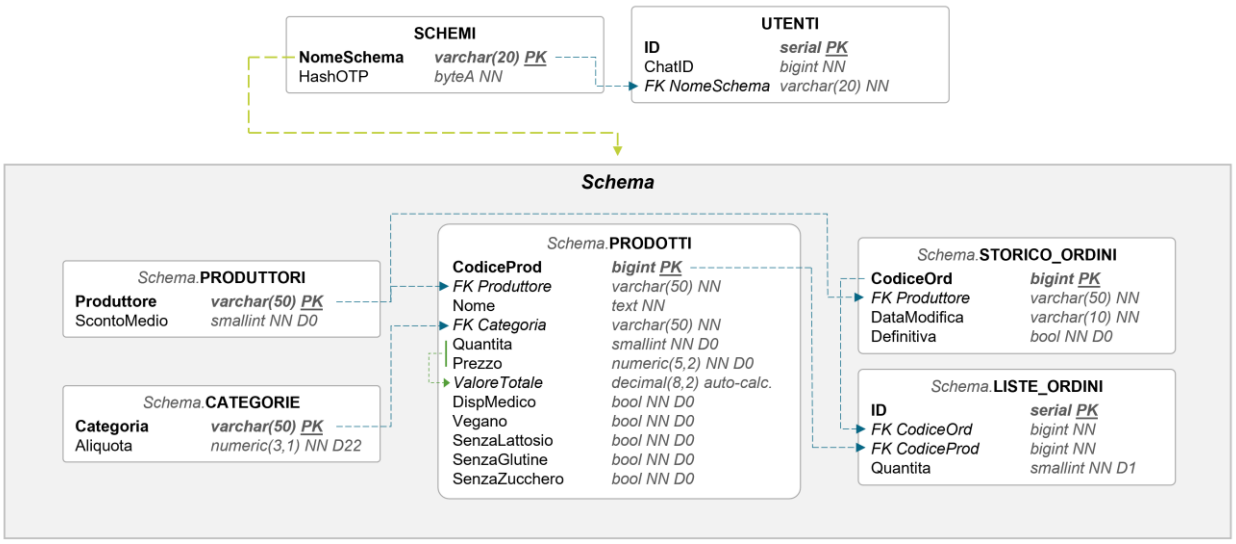


Figure 3: Herbie's DB design scheme.

A2. Herbie Telegram Bot

Herbie is live on Telegram at the following link: https://t.me/herbie_tbot Her Bot is hosted on Heroku free tier [5]: she is available 24h, but she may initially answer after 1 minute when contacted (due to the virtual "Dyno" turning back on). In Figure 4 (next page), a few screenshots of Herbie Telegram Bot in action are provided.

The Bot contains 3 main commands that can be launched via the Menu (Figure 4 (b)) or via text:

- **Command *"/aggiorna"*** (Figure 4 (c)):
  - Register a new Product into the database:
    - Base info (*barcode*, *name*, *supplier*, *category*, *quantity*, *price*): these data don't have a default value and must be populated for each product.
    - Extra info (booleans: *medical device*, *vegan*, *lactose-free*, *glutine-free*, *sugar-free*): these data have a default value of "False", so they can be skipped if not needed.
  - Update information of a Product already registered:
    - Update base info
    - Update extra info
  - Register a Supplier or update its information:
    - Supplier name
    - Typical discount on purchases
  - Register a Category or update its information:
    - Category name
    - Applicable VAT rate
  - DB cleaning (delete products, suppliers and categories with 0 pieces)

- **Command *"/vista"*** (Figure 4 (d)):
  - Manually download printable views of the warehouse (Excel format):
    - Warehouse inventory view (all or one supplier only)
    - Recap view with the total inventory quantity and value by supplier / category
    - An order list (in progress or definitive)

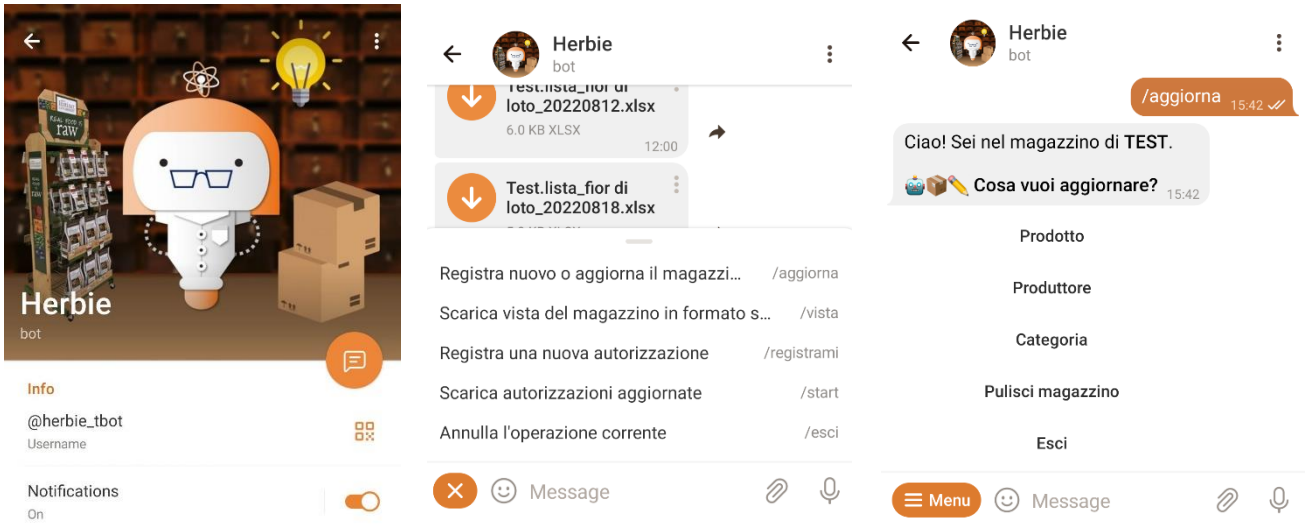
- **Command *"/registrami":***
  - Needed only once, to authorize the user to the use of the Herbie Telegram Bot. The user needs a *passcode* ("OTP") that must be given to him by the Bot admin. Once registered, the user's *chat_id* is stored to the DB in the table "Utenti".

Furthermore, Herbie Telegram Bot works as a **receiver** for any printable warehouse view that is requested via Alexa to Herbie Vocal (Figure 4 (b)). At the moment, any view requested vocally is delivered to all registered users of Herbie Telegram Bot, but this will be improved in the future.

To identify a Product, Herbie Telegram Bot integrates a **barcode scanner** from PyZbar [16] and a **search algorithm** for the product name. When registering a new product, the user must send Herbie a photo of the barcode or type the barcode numbers in the chat. When the user wants to update a product already registered, he can identify it either via barcode (sent via photo or via text) or by typing the keywords of the product name in the chat: Herbie will return either the best match or the best 3 matches (Figure 4 (e)), similarly to her Vocal counterpart.

Furthermore, to ensure consistency in the data typed by the user, any operation requiring to write the name of a supplier or a category will show a series of **clickable buttons** with the suppliers / categories already present in the DB, so that the user can directly pick one of them (Figure 4 (f)).

(a) (b) (c)



(d) (e) (f)

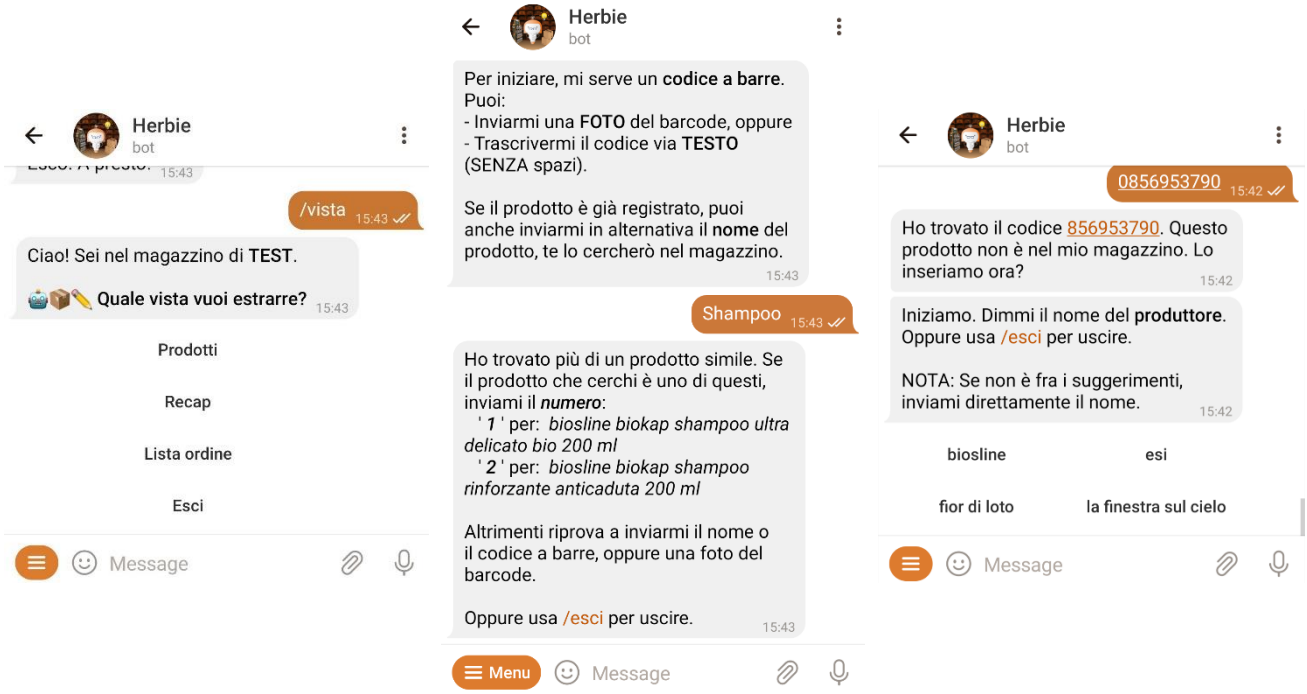


Figure 4: Some screenshots of Herbie Telegram Bot in action. The figure shows:
(a) Herbie's profile page;
(b) The list of Herbie's commands in the Menu. In the background, some Excel views previously requested via voice are shown;
(c) The menu of the ***/aggiorna*** command, with the clickable buttons corresponding to the target functions;
(d) The menu of the ***/vista*** command, with the clickable buttons corresponding to the downloadable views;
(e) The built-in search functionality by product name;
(f) The clickable buttons with the list of suppliers already stored to the DB. These buttons help in reducing the risk of creating duplicate, mistyped instances of existing records.
Note: all clickable buttons disappear after a new input or command is given, in order not to interfere with the next conversational states.

A3. Conversations details

Herbie's conversations can be grouped into 3 main areas:

1. Find a *product*
2. Find a *supplier* + prepare an *order*
3. Update *warehouse* + ask for DB views

Conversations in the first 2 areas must start using specific triggers, which are "trova un prodotto" or "trova un fornitore" (with their possible variants).

Conversations schemes

Here follows an overview of the various tasks and opportunities provided by each area:

- **Area 1) *"Trova un prodotto":***
  - Ask the available quantity: *"Quanti pezzi ho?"*
    - Proactive Herbie: *"Vuoi aggiungerlo al prossimo ordine?"*
  - Q&A: ask for product information (separate questions or "*leggi tutte le informazioni*"):
    - Supplier
    - Category + applicable VAT rate
    - Medical device (y/n)
    - Retail price
    - Vegan (y/n)
    - Lactose, gluten, sugar (y/n)
  - Give user guidance: *"Cosa puoi fare?"*

- **Area 2) *"Trova un fornitore":***
  - Ask DB view: *"Invia vista giacenze su Telegram"* (for the selected supplier only)
  - Ask products running out of stock (less than 5 pieces): *"Trova prodotti in esaurimento"*.
    - This will trigger the order creation conversation, starting directly from the *Suggest* step instead of the *Read* step.
  - Guided order creation or continue an order:
    - Commands:
      - *"Prepariamo / Continuiamo ordine"*
      - *"Trova ultima lista ordine"* (will *first* ask the user what to do with the list)
    - Proactive steps by Herbie:
      1. *Read* step: Herbie can read the open list (only if the list is *not* newly created)
      2. *Suggest* step: Herbie can suggest products that are running out of stocks
      3. *Write* step: Herbie lets the user dictate the products to order
    - Total order cost
    - Ask send list: *"Invia lista ordine su Telegram"*
    - Definitely mark an order list as closed (definitive)
  - Register the delivery of an order: *"Mi è stato consegnato l'ultimo ordine"*
    - Automatic update of the inventory quantities for the products in the closed list
  - Give user guidance: *"Cosa puoi fare?"*

- **Area 3) Warehouse area:**
  - Ask DB view: *"Invia vista giacenze su Telegram"* (all suppliers)
  - Update the warehouse quantities: *"Aggiorniamo le giacenze":*
    - Proactive Herbie: *"Ti è stato consegnato un ordine?"*
      - Herbie will ask this only if it finds a closed list in the DB (table OrderHistory)
      - Automatic update of the inventory quantities for the products in the closed list
    - Let the user dictate the products received or sold and their quantities
  - Give user guidance: *"Cosa puoi fare?"*

Detailed description of each area

In the following paragraphs, each conversational area is described in detail, with the possible turns, alternative paths and full flows.

The *initiative* alternates between Herbie and the user in all areas.

Area 1: *find a product*

This is the most basic of the three areas: it is a simple *Q&A* session. The idea here is to make Herbie a valid replacement for the times in which the shop manager must go read the product info, price, details etc from the supplier booklet, with some added functionalities too.

The user must start the conversation (entry point) asking Herbie to *find a product* (i.e. "Trova un prodotto" and similar variants). Herbie asks the user to say the name of the product (it may include the supplier too), then activates the general lookup scheme. When a match is chosen, Herbie asks the user what he wants to do.

Here the conversation becomes mainly *user-driven*: the user can ask questions on specific product information (single info or read all, i.e. price, product details, …), Herbie answers and then asks the user if he needs more. The Q&A can go on infinitely until the user uses either the *stop* or *completed* intent.

During the Q&A, the user can ask Herbie which is the number of items of the product in the warehouse. In this case, the conversation becomes mainly *system-driven:* Herbie checks the warehouse, may warn the user that the product is going to run out soon and proactively asks the user if he wants Herbie to add the product to the next order list (i.e. "Vuoi che te lo aggiunga in lista per il prossimo ordine?"). If yes, Herbie adds it to the latest open order list (or to a new list, if no open list is found).

Area 2: *find a supplier & order preparation*

This second area is the most complex of the three. The entry point is always the request by the user to *find a supplier* (i.e. "Trova un produttore" and similar variants), with Herbie activating the general lookup scheme. After a match is chosen, Herbie asks the user what he wants to do. From here, the conversation forks, following different paths.

a. Order preparation

The biggest path is *order preparation*. The idea here is to support the shop manager in the preparation of an order list: Herbie can offer proactive suggestions of pieces running out of stock and provide information on the number of pieces available. When the order list is ready, Herbie can send it via Telegram Bot to the shop manager, so that he can forward it to the supplier. Please note that, in the case of Copertino's small herbalist shop, *all suppliers are the manufacturers* of the products (they use no resellers/distributors), therefore each order list must be made in reference to *one* manufacturer and include only products made by it.

The *order preparation* path is an ordered sequence of tasks to which Herbie proactively involves the user. It is *mixed-initiative*: Herbie proposes each step to the user, but the latter can decide to enter it or fork the conversation according to what he needs. Each step is managed by a separate Rasa form. Herbie always guides the user time by time on what he can do and what to say.

Herbie asks the user *in sequence* if he wants her to:

1. *Read the open list:*
   If the user accepts, Herbie will read through the list one product at a time. For each item, the user must either (i) keep it (i.e. "va bene"), (ii) state a different number of pieces (i.e. "no, segna 10 pezzi"), or (iii) ask to remove it from the list (i.e. "no, questo toglilo").
2. *Suggest products running out of stock:*
   If the user accepts, Herbie will prepare a shortlist with the products of the current supplier that have less than 5 pieces in the DB, then will read through it one product at a time. For each item, the user must either (i) state the number of pieces to order, or (ii) ask to skip it (i.e. "no, salta").
3. *Let the user dictate the products he wants for the current supplier:*
   If the user accepts, Herbie will enter a form which will ask the user, in a loop, (i) the name of a product, (ii) the number of pieces to order.

The *order preparation* path can be triggered in 3 different ways:

- "Prepariamo un ordine" (or similar variants)
- "Trovami l'ultima lista ordini" (or similar variants)
- "Trova prodotti in esaurimento" (or similar variants) – note: this entry point will skip the "*read the open list"* step.

When all steps are completed (or skipped), Herbie will read the total order cost and ask the user to mark the order as *closed* (definitive). If the user accepts, the list cannot be modified anymore and will be sent by Herbie as xslx file via the Telegram Bot. If the user denies, instead, Herbie is available for further questions involving the supplier and the order (see below).

b. Supplier & order requests

This path contains general requests to Herbie regarding the supplier and the order, like (i) sending the DB view with the inventory of the supplier via Telegram Bot, (ii) sending the current order list via Telegram Bot, (iii) asking the total cost of order, (iv) marking the open order list as closed, (v) asking to *register the delivery* of a closed order (below).

c. Register a delivery

When a delivery is received for an order prepared via Herbie and marked as closed, Herbie takes care of automatically updating the available quantities in the warehouse for all products in the list. This part is mainly *system-driven*, with Herbie asking confirmation to the user before updating the DB. Herbie will also send the closed list via Telegram Bot as a reference for the update she makes to the warehouse and permanently *delete* the closed list from the order history.

Area 3: *warehouse update & views*

This third area has no specific entry point. The user can ask either to (i) send the DB view of the whole warehouse via Telegram Bot, or to (ii) update the warehouse (i.e. “aggiorniamo le giacenze”).

When the user asks to update the warehouse, Herbie *proactively* performs a quick check, looking if there is any order list in the DB that has been marked as closed and not yet registered as delivered: in this case, she immediately asks to the user if he just received the delivery of an order (i.e. “Ti è stato consegnato un ordine?”). If the user confirms, Herbie asks for the name of the supplier via the general lookup scheme and performs the delivery registration task as explained in area 2, point ‘c’. If the user denies that a delivery was received (or if no closed lists are found), Herbie skips this part and goes directly to the normal warehouse update path.

The normal warehouse update path is mainly *system-driven* and it is performed via a Rasa form: Herbie first guides the user on what to do, then starts a loop in which she asks first to say the name of a product (or supplier + products) via the general lookup scheme and then to state the inventory variation (add/subtract *n* pieces, i.e. “aggiungi 5 pezzi”, or “togli un pezzo”). Herbie updates live the DB on each turn. The loop proceeds until the user terminates the conversation via either the *stop* or *completed* intent.

A4. NLU evaluation: intent confusion matrices

The following Figures 5 and 6 show the 2 intent confusion matrices calculated for both the tested pipelines by *rasa test nlu* on the *test* nlu set (*tests/nlu_test.yml*). It can be easily seen that the chosen pipeline (the *main* pipeline, Figure 5) reports a lower number of mistakes.

Chosen pipeline ("*main*"):

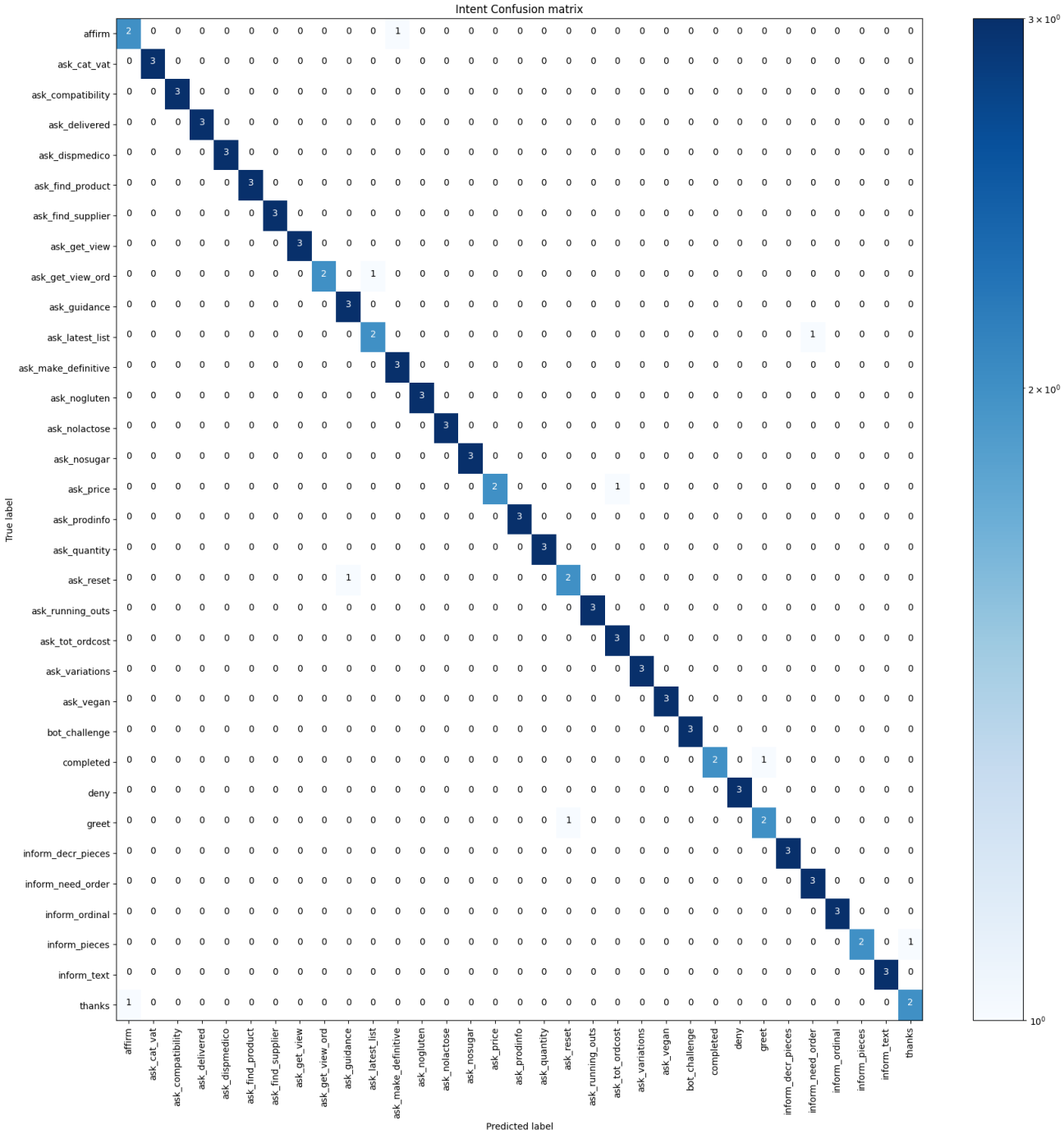


Figure 5: Intent confusion matrix on the test nlu set for the *main* pipeline.

Alternative pipeline:

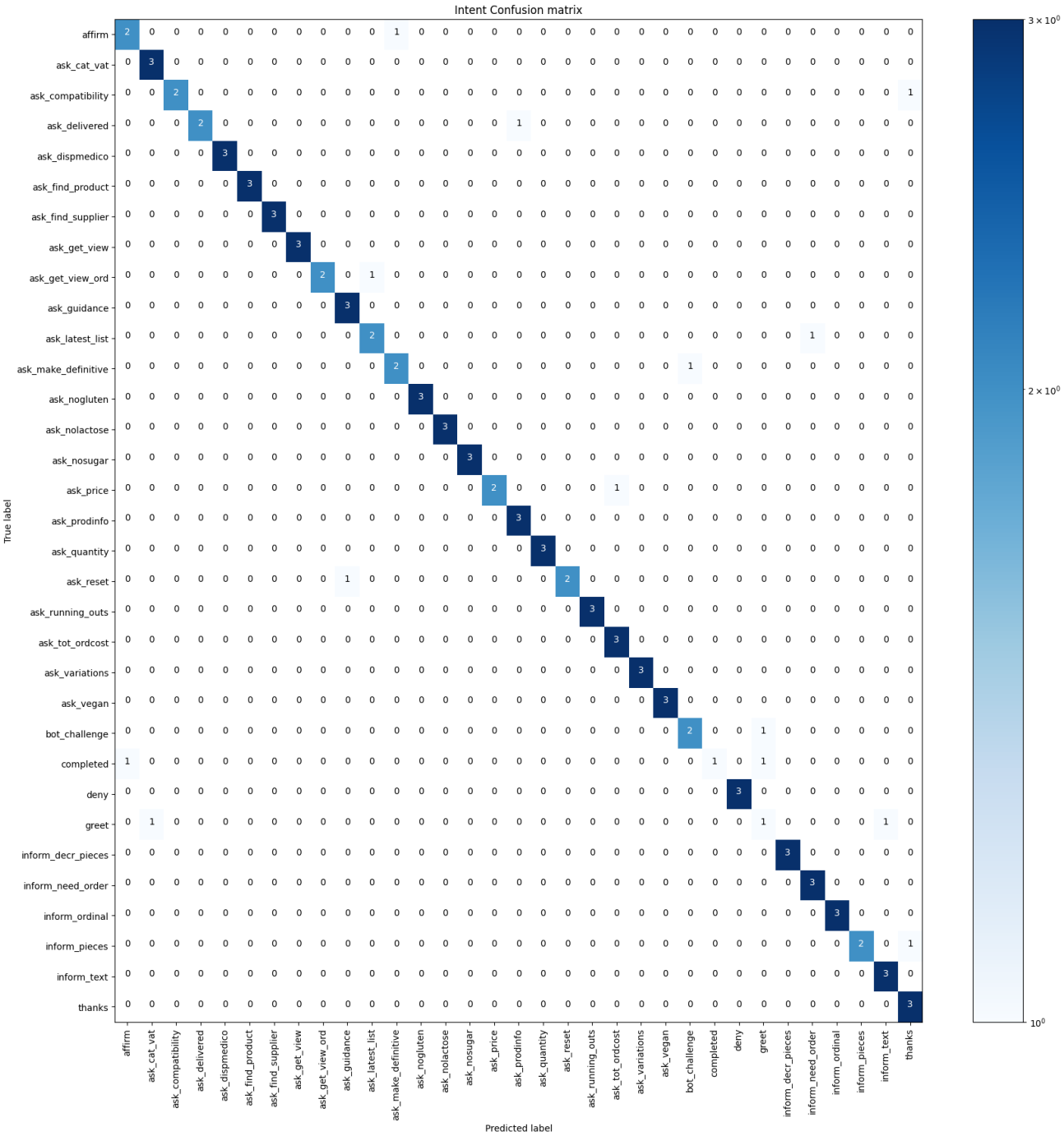


Figure 6: Intent confusion matrix on the test nlu set for the *alternative* pipeline.

A5. Dialogue management evaluation: actions confusion matrix

Figure 7 below shows the action confusion matrix calculated by *rasa test core* on the *tests* stories set. Since the response selection components are the same for both the tested pipelines, the results don't change.

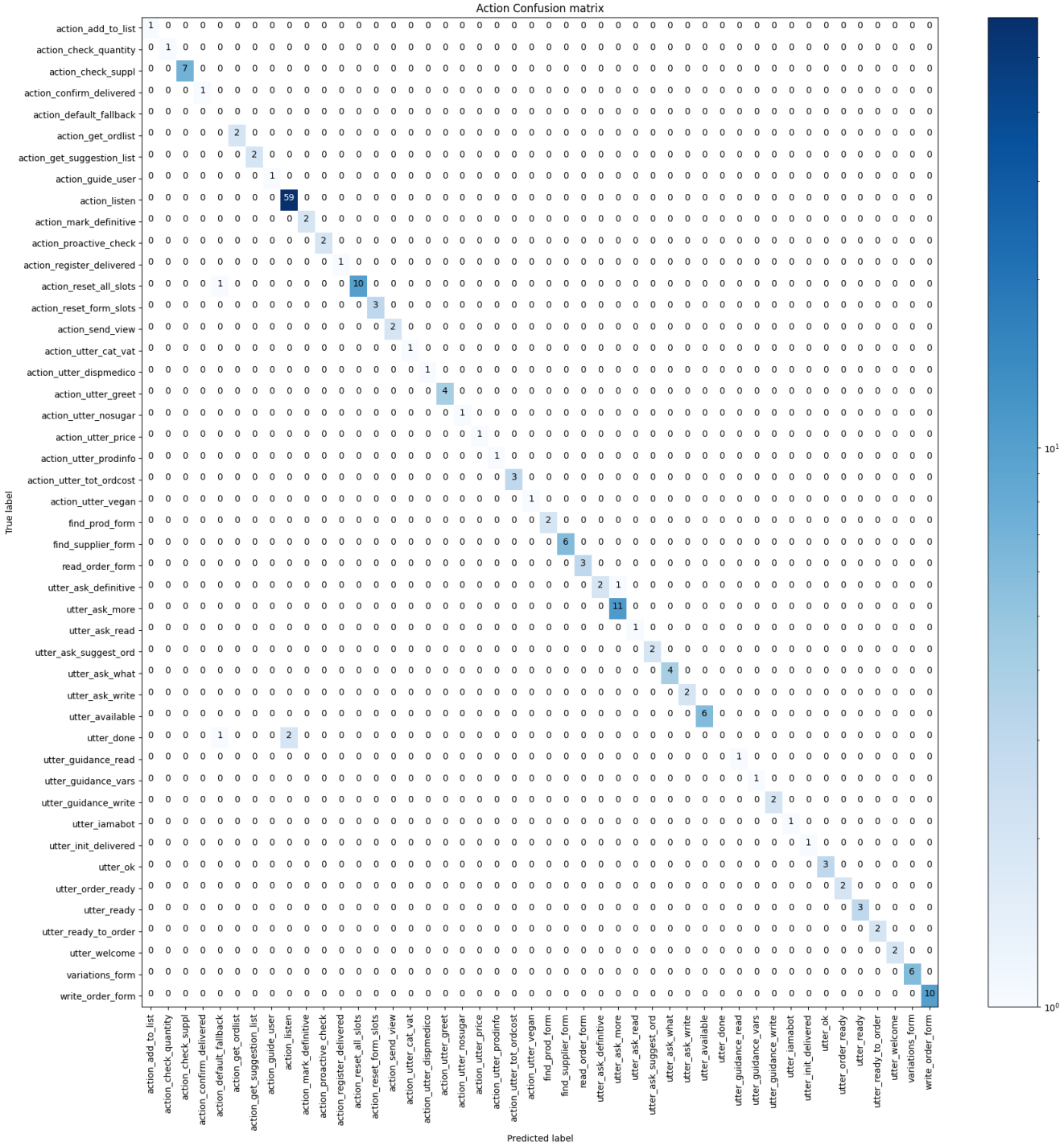


Figure 7: Action confusion matrix on the test stories set.

A6. Human evaluation survey

In Exhibit 3 below, the list of questions in the survey and the scores obtained by each of the 8 interviewees (3 testers and 5 observers) are reported. The assignable score goes from 1 (lowest) to 5 (highest).

Questions	User 1	User 2	User 3	Avg users	Obs. 1	Obs. 2	Obs. 3	Obs. 4	Obs. 5	Avg obs.	Avg
Helpfulness: do you think Herbie is providing useful help to the shop manager in performing each task?											
- *Area 1) Product information*	*5*	*5*	*5*	**5.0**	*5*	*5*	*4*	*4*	*5*	**4.6**	**4.8**
- *Area 2) Order making*	*5*	*5*	*4*	**4.7**	*5*	*5*	*4*	*5*	*5*	**4.8**	**4.7**
- *Area 3) Warehouse update*	*5*	*5*	*5*	**5.0**	*5*	*5*	*5*	*5*	*5*	**5.0**	**5.0**
Proactivity: do you think Herbie's proactive questions & suggestions create useful value to the shop manager?	*5*	*5*	*5*	**5.0**	*5*	*5*	*5*	*5*	*5*	**5.0**	**5.0**
Guidance:											
- **Quantity**: do you think that Herbie provides the right quantity of guidance on how to do things? (i.e. not too few, not too much)	*5*	*4*	*5*	**4.7**	*5*	*5*	*5*	*5*	*5*	**5.0**	**4.8**
- **Clarity**: do you find Herbie's instruction clear to follow and to understand?	*5*	*5*	*5*	**5.0**	*5*	*5*	*5*	*5*	*5*	**5.0**	**5.0**
Ease of use:											
- **Learning curve**: do you think it takes a small time and effort to learn to use Herbie?	*3*	*4*	*4*	**3.7**	*4*	*5*	*3*	*3*	*4*	**3.8**	**3.7**
- **Intuitiveness of structure**: do you find the organization of conversations into areas, paths and schemes intuitive and easy to remember?	*3*	*3*	*4*	**3.3**	*3*	*4*	*3*	*3*	*4*	**3.4**	**3.4**
- **Alternative ways to ask**: do you find the presence of alternative ways to ask for the same task intuitive?	*5*	*5*	*5*	**5.0**	*5*	*5*	*5*	*5*	*5*	**5.0**	**5.0**
- **Overall ease of use**: considering what discussed before, do you find Herbie easy to use?	*5*	*5*	*4*	**4.7**	*5*	*5*	*4*	*5*	*5*	**4.8**	**4.7**
Exhaustiveness: do you think that the functionalities provided by Herbie are exhaustive for the 3 areas in a real use scenario?	*5*	*5*	*5*	**5.0**	*5*	*5*	*5*	*5*	*5*	**5.0**	**5.0**
Speed: do you think Herbie's interactions are quick enough for a real use scenario?	*4*	*5*	*5*	**4.7**	*5*	*5*	*4*	*5*	*5*	**4.8**	**4.7**
Naturality: how natural does talking to Herbie feel to you?	*5*	*5*	*4*	**4.7**	*5*	*5*	*5*	*4*	*5*	**4.8**	**4.7**

Exhibit 3: questions and results of the human evaluation survey.

References

[1] Amazon, “Alexa Skills Kit,” [Online]. Available: https://developer.amazon.com/en-US/alexa/alexa-skills-kit. [Accessed August 2022].

[2] Rasa, “Homepage - Open source Conversational AI,” [Online]. Available: https://rasa.com/. [Accessed August 2022].

[3] Telegram, “Telegram Bot API,” [Online]. Available: https://core.telegram.org/bots/api. [Accessed August 2022].

[4] Herbie, “Telegram Bot,” [Online]. Available: https://t.me/herbie_tbot. [Accessed August 2022].

[5] Heroku, “Homepage,” [Online]. Available: https://www.heroku.com/home. [Accessed August 2022].

[6] Google, “L'Erboristeria di Anna Nadia Pignatelli on Google Maps,” [Online]. Available: https://g.page/lerboristeriapignatelli. [Accessed August 2022].

[7] A. Fawzy Gad, “Paperspace Blog - Measuring Text Similarity Using the Levenshtein Distance,” [Online]. Available: https://blog.paperspace.com/measuring-text-similarity-using-levenshtein-distance. [Accessed August 2022].

[8] PiPI, “FuzzyWuzzy Project description,” [Online]. Available: https://pypi.org/project/fuzzywuzzy/. [Accessed August 2022].

[9] Heroku, “Heroku Elements: PostgreSQL Add-On,” [Online]. Available: https://elements.heroku.com/addons/heroku-postgresql. [Accessed August 2022].

[10] Python, “Downloads,” [Online]. Available: https://www.python.org/downloads/. [Accessed August 2022].

[11] Ngrok, “Homepage,” [Online]. Available: https://ngrok.com/. [Accessed August 2022].

[12] spaCy, “Homepage,” [Online]. Available: https://spacy.io/. [Accessed August 2022].

[13] Scikit-Learn, “Metrics - Precision Score,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html. [Accessed August 2022].

[14] Scikit-Learn, “Metrics - Recall Score,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html. [Accessed August 2022].

[15] Scikit-Learn, “Metrics - F1 Score,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html. [Accessed August 2022].

[16] PyPI, “PyZbar Project description,” [Online]. Available: https://pypi.org/project/pyzbar/. [Accessed August 2022].