Pre-print of full paper presented at CONVERSATIONS 2022 – the 6th International Workshop on Chatbot Research, Applications and Design, hosted by the University of Amsterdam, Netherlands, November 22-23. The final version of the paper will be published in the forthcoming Springer LNCS post-workshop proceedings.

Increasing the Coverage of Clarification Responses for a Cooking Assistant

Gina E.M. Stolwijk[1] and Florian A. Kunneman[1][0000−0002−1932−3200]

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081HV Amsterdam, Neteherlands
gina.stolwijk@gmail.com, f.a.kunneman@vu.nl

Abstract. In conversation genres like instruction, clarification questions asked by a user may either relate to the task at hand or to common-sense knowledge about the task domain, whereas most conversational agents focus on only one of these types. To learn more about the best approach and feasibility of integrating both types of questions, we experimented with different approaches for modelling and distinguishing between task-specific and common sense questions in the context of a cooking assistant. We subsequently integrated the best ones in a conversational agent, which we tested in a study with six users cooking a recipe. Even though the three elements functioned well on their own and all participants completed the recipe, question-answering accuracy was relatively low (66%). We conclude with a discussion of the aspects that need to be improved upon to cope with the diverse information need in task-based conversational agents.

Keywords: Clarification · Cooking Assistant · Natural Language Processing · User study.

1 Introduction

Today's task-based conversational agents have been mainly built to generate responses to direct knowledge questions, where the context is represented by the relevant knowledge underlying these questions and is leveraged to determine the user goal, perform slot-filling and answer follow-up questions [25,27]. Typical examples of such task-domains are restaurant reservations and service agents. In contrast, in conversational genres like instruction-giving, information requests are typically related to the task at hand, where both the task-related concepts and status are of importance for a proper interpretation of the request. An agent giving directions in a virtual environment, for example, would need to consider the current location and view of the user, as well as types and features of objects along with their spatial relation [13].

In this paper we study the challenges of modelling information seeking dialogue as part of cooking instruction, a genre where both common-sense knowledge and task-related knowledge are required by the agent to best assist the user in its endeavour. In order to handle the diverse questions that may be asked during cooking instruction dialogue, the agent would need to incorporate a vast body of recipe-related and cooking-related knowledge, recognise which of

the two needs to be drawn upon when faced with an information request, and know whether a proper answer to the question is available. Such an application hence comes at a larger risk of confusing a request with a different one and giving the wrong response to a question, which is why most task-based conversational agents limit the scope of requests that can be addressed to only task-related questions (e.g.: [23,12]) or only questions about general domains (e.g.: [25]).

The studies that did apply conversational agents with both task-specific knowledge and domain-specific knowledge did not obtain insight into the quality of responses to clarification questions when such a system converses with a user. [29] created a dataset consisting of cooking recipes and annotated cooking instruction dialogue grounded in these documents, and limit evaluation to performance on this dataset. Participants in the recent Alexa Taskbot challenge [1] who integrate a diverse set of knowledge sources, like Howdy Y'all [2], Grillbot [14] and Miutsu [20], only focused on user satisfaction in their evaluation. In contrast, in our study we propose detailed heuristics to draw upon a particular information source during conversation, and conduct a user study where we evaluate the conversations on the answer accuracy and can pinpoint the nature of the mistakes that are made. We address the following research questions:

RQ1) How can a task-based conversational agent distinguish task-specific from general domain questions?

1. We deployed and evaluated a set of approaches to model general domain knowledge and task-specific knowledge in the cooking domain, divided into:
   (a) Cooking-related question answering based on question-answer pairs in a community question answering platform
   (b) Extracting knowledge from cooking recipes based on heuristics and segmenting the recipe procedure into conversational steps
   (c) Classifying given questions into task-related or common sense
2. We integrated the approaches in a conversational agent, and formulate a set of heuristics to enable the agent to draw upon the right knowledge module during the conversation

RQ2) To what extent can a task-based conversational agent distinguish task-specific from general domain questions in a real-world setting?

To answer this question, we evaluated the conversational agent through a user study where the users are actually cooking when talking with the agent. All conversations were specifically analyzed for the performance of the agent when addressing user questions.

2 Related Work

2.1 Modelling knowledge for conversational agents

Disclosure of the right knowledge at the right time is a key aspect to the success of many task-based conversational agents, where the nature of the task defines

the requirement of its knowledge-based capabilities. For tasks that are targeted at fulfilling a request (e.g.: transfers in a banking context, or booking a restaurant), the most common approach is to train a model on dialogues that are annotated for a pre-defined set of slots and values [25]. Another common task is conversational search, where the right answer to a user's query is coordinated with the user through conversation. Since such agents need to accommodate a wide range of questions, large datasets are typically used [33], and retrieved based on question-question similarity (i.e., example-based) [32]. For tasks that require the user or agent to perform a sequence of steps, like navigation [13], document inquiry [12] or cooking [23,29], the agent needs to have a thorough understanding of the important concepts and their relation in separate documents. Enabling this required knowledge may be approached as a reading comprehension task [6], or by transforming documents into a dedicated meaning representation for the task at hand [21,7]. In our approach, we adopt an example-based approach to model general domain knowledge, and parse recipes for a particular set of information units to accommodate task-based knowledge.

2.2 Distinguishing between different knowledge sources

The main challenge for conversational systems that need to accommodate a wide range of questions, is the large search space that increases the chance for confusing a posed question with a similar one. A common approach to tackle this is to deploy a module to first classify a question by its domain [2] or type [31]. A different approach is taken by [28,29], who limited the commonsense-knowledge in their cooking agent to a set of predefined topics (e.g., replacing ingredients, use of cooking utensils, etc.) by creating databases with background information for each topic, and deployed a set of rules and custom actions to select the right knowledge source to address a question. We adopt both question type classification and a set of heuristics to query the right knowledge-module (task-specific or common sense) in our system. In contrast to [29], the commonsense database that we make use of covers a broader range of questions in the cooking domain. In addition, we conduct a user study where we zoom in on the quality of question answering, which has not been done in the studies cited above.

3 Increasing the coverage of clarification responses

We distinguish two broad types of clarification questions that may be asked during cooking recipe instruction: commonsense questions and task-specific questions. For both types we set out to ensure a broad coverage in a data-driven way, and additionally studied how well the two can be distinguished to reduce the chance for confusion during conversation. In the following, for each of these three sub tasks we will describe experimentation to inquire into the best performing method. We focus our study on the Dutch language sphere, but the methods we apply are mostly applicable to other languages as well.

3.1 Commonsense question answering

The common-sense question-answering task is formulated as finding the best answer to a general user question in the cooking domain from a large database with QA-pairs. We experimented with two approaches to model question similarity.

Approach 1) Word2vec The first algorithm aimed to find the sentence(s) from a database with the largest similarity to the user's query using a word2vec model [22]. For this, SpaCy's [16] Dutch pipeline nl_core_news_lg was implemented,[1]. For each sentence in the database, using word2vec, separate token embeddings were computed, which were averaged to obtain a sentence embedding. An incoming user query was represented in the same way and compared to each query in the database using cosine similarity, selecting the sentence with the highest similarity as the best match.

Approach 2) Sentence-BERT The second algorithm consisted of finding the sentence(s) from the database with the smallest distance to the user's query, using the context-dependent sentence-BERT trained on Dutch data.[2] BERT (Bidirectional Encoder Representations from Transformers) has shown to allow for state-of-the-art performance in a wide range of tasks [10]. Sentence-BERT computes the embeddings for each sentence separately, and then compares them using a similarity metric [26]. Each sentence was mapped to a vector space of 768 dimensions, using mean pooling of the context-dependent token embeddings with an attention mask. The sentence embedding of the user query was compared to the embedding of each query in the database using cosine distance, selecting the sentence with the lowest distance as the best match.

Dataset and pre-processing We chose to make use of a general community question answering (CQA) platform, which aligned with our purpose of common sense cooking knowledge. We downloaded 10,000 questions from the Dutch Community Question Answering platform `goeievraag.nl`[3], categorised with the Food and Drinks label. Each QA-pair consisted of a user query and the most popular, or first given, answer. We pre-processed the queries in the database by applying a CNN-based part-of-speech tagger by means of SpaCy.[4] Afterwards, only (proper) nouns, verbs and adjectives were maintained, as they represent the central information to most cooking-related questions. Stop words were also removed,[5] after which each query was vectorised by either of the two approaches.

Experimental procedure We performed a controlled experiment by manually selecting seventy queries from the database and testing how well the two approaches perform in retrieving any of these queries when presented with a differently worded version of it. We set the size of the database to 10,000. The manual

[1] `https://spacy.io/model/nl#nl_core_news_lg`
[2] jegorkitskerkin/bert-base-dutch-cased-snli
[3] `https://www.startpagina.nl/v/eten-drinken/`
[4] `https://spacy.io/api/tagger`
[5] Using the following stopwordlist: `https://github.com/explosion/spaCy/blob/master/spacy/lang/nl/stop_words.py`

Table 1: Proportion of correct answers per algorithm and ranking on retrieving the right common-sense cooking-related question-answer pair.

	Algorithm	
Rank	Word2vec	sentence-BERT
1	0.47	0.66
2	0.09 (of 53% = 0.05)	0.24 (of 34% = 0.08)
3	0.06 (of 48% = 0.03)	0.20 (of 26% = 0.05)
Total	0.55	0.79

selection was done by one of the authors. So as to ensure a variety of queries to evaluate on, the selected queries were evenly distributed across seven question categories (see Appendix A for details).

To generate seed queries for each of the 70 selected queries, we used a combination of two techniques: backtranslation [11,18] with deep-translator [3] and paraphrasing [4] with the parrot [9] library. The augmenter aimed to produce a maximum of 10 paraphrases, by collecting the input utterance and annotations (intents, slots, slot types), and augmenting these [9]. To make sure that the generated sentences covered the same meaning as the original query, but differently phrased, one of the authors manually checked each generated sentence and removed incorrect paraphrases. In the end, an average of almost four sentences per original query remained (M = 3.97, SD = 1.91), for a total of 278. Grammatically incorrect sentences were not removed, since end users could also pose incorrectly formulated queries. The seed queries were pre-processed in the same way as the queries in the database.

For the two approaches (Word2vec and Sentence-BERT) we tested how well they could retrieve the right query from the database at different database sizes. The approaches were presented with each of the 278 reformulated variants of the queries. As evaluation, the average number of correct answers (i.e., belonging to the exact original [non-paraphrased] query) was computed. To measure how close an approach was, this number was computed for the correct answer at rank 1, 2 or 3. Significant effects were measured by performing an ANOVA with a dichotomous dependent variable (correct/incorrect). The main effects as well as the interaction between the independent variables 'database size' and 'algorithm' were assessed. This was done combining all three highest ranks: if the best match was incorrect, the second best was also taken into account, if the second best match was incorrect, the third best was also taken into account.

Results The results per approach, database size and ranking are given in Table 1. There was no significant interaction between database size and algorithm on answer correctness ($F(2) = 1.93, p = 0.15$). A second ANOVA was performed looking only at the main effects, showing that the best performing algorithm was sentence-BERT $F(1) = 177.56, p < 0.001$), and changing the size of the database did not significantly affect performance ($F(2) = 0.62, p = 0.54$).

The sentence-BERT approach manages to retrieve a correct question for around 80% of the queries when considering all three ranks. If the first retrieved question-answer pair is not correct, the chance of finding a correct answer at

the second rank drops considerably to 0.24. In a conversational setting, these outcomes are not trustworthy enough for presenting the user with an alternate answer (e.g.: the second-ranked QA pair) when an initially retrieved result is not satisfactory. Sentence-BERT gave the best results, for which scores higher than 0.28 were associated with mostly incorrect answers. This approach will be implemented in the cooking assistant for answering common-sense questions in the cooking domain, using a threshold of 0.28 below which an outcome is presented.

3.2 Task-specific question answering

A cooking assistant should have a sufficient number of recipes to instruct to the user, which are abundantly available on web-based cooking platforms. The approach to modelling recipe-specific knowledge is strongly related to the particular recipe to model, and more specifically to the way the recipe is formatted. We identify the following as the most important elements of a recipe: the recipe name, the number of people for whom the recipe is meant, the expected cooking time, the ingredients and quantities, the cooking utensils and the recipe procedure.

Part of what we present below incorporates heuristics that are specific to the website from which we extract recipes and their constituents (Smulweb[6]). The extraction of other types of information is more generalisable to any website with cooking recipes, namely separating quantity, unit and ingredient and dividing the recipe procedure into steps that are suitable to a conversational interface.

Heuristics Part of the heuristics were based on numerals and grammatical information, for which we made use of SpaCy's POS-tagger and lemmatizer, trained on the Dutch pipeline nl_core_news_lg.[7]. An example of the recipe lay-out and type of information that was extracted is given in Figure 1.

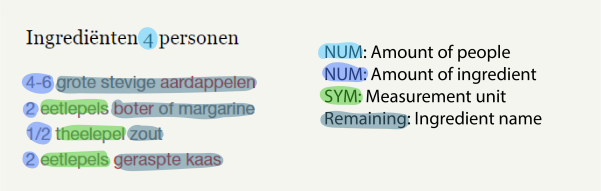

Fig. 1: Recipe ingredients split up using part-of-speech tags and heuristics.

Extracting recipe steps The task of extracting recipe steps can be formulated as a segmentation task, where each segment should be suitable as a single unit of instruction in a conversational setting. In line with [17], we defined single

[6] smulweb.nl
[7] https://spacy.io/models/nl#nl_core_news_lg

instruction steps as instructions where a single action is performed. A difficulty is that a single sentence may include multiple actions.

To identify cooking actions and their accompanying information from a recipe instruction, it was first split up into sentences. [8] We then generated a dependency tree for each sentence using the NLTK library.[9] In parallel, the full sentence was tagged using SpaCy's POS-tagger. The assumption was made that each step (sub-tree), should contain at least a verb (i.e., cooking action) as its root, of which the children down to the leaves represented the context (ingredients, quantities, utensils, etc.). Whenever a token was indicated to be the root of a subtree, it was assessed whether this token was considered a verb as indicated by the tagged sentence. If both of these conditions were met (i.e., token is the root of a subtree, and token is a verb), the subtree was treated as a step. The root of the main tree and the remaining tokens served as a step on its own.

Since POS-tagging models are usually not trained on imperative sentences [7], which are common in cooking instructions, we retrained the tagger using a Random Forest classifier. The classifier used different features in order to find the best fitting tag for a token: the token itself, its precedent, its subsequent token, its prefix (3 first letters), its suffix (3 last letters), whether it is the beginning or the end of a sentence, whether it is a number, and whether it has capitals. For training purposes, a dataset of 284 sentences (3,851 tokens) was extracted from a variety of Dutch cooking instructions posted on Smulweb. For this dataset, the SpaCy tagger's results were used as a reference, after which the tags were manually corrected where necessary by one of the authors. The focus of this correction was on tokens that were wrongfully (not) tagged as verbs, since the verbs were used to split up the procedure.

Experimental procedure To evaluate the quality of recipe segmentation with and without a re-trained Pos-tagger, we extracted ten cooking recipes from Smulweb. The ten selected recipes were annotated manually by one of the authors to set a ground truth for evaluation. They were then fed into the different algorithms.[10] Apart from recipe segmentation, the information units of the recipes were also annotated. We found that this information could be identified by means of the heuristics at near-perfect accuracy for the ten recipes, which can be explained by their accommodation to the platform.

We compared the two segmentation approaches to two baselines: a baseline that consisted of selecting each sentence as a single step, and a semi-random baseline which segmented a sequence at each position with probability $1/k$ (where k is the average segment length of the ground truth). We represented the output of the four approaches by marking each token with a 0 or 1, where the latter indicated the end of a step.

Results. The best performing algorithm was the dependency tree-based segmentation using POS-tagging, closely followed by the dependency tree-based

[8] `https://spacy.io/api/sentencizer`

[9] `https://www.nltk.org/_modules/nltk/tree.html`

[10] See (Appendix C) for the annotation guidelines.

segmentation using the re-trained tagger (Table 2). The Sentencizer errors were almost all false negatives (i.e., predicted not to be the end of a step, while it actually was), while the tree-based segmentations yielded some false positives as well (i.e., predicted to be the end of the step, while it actually was not).

Table 2: Confusion matrices between each approach and the ground truth segmentation.

Ground Truth	Random Baseline		Sentencizer Baseline		Tree-based with regular POS		Tree-based with fine-tuned POS	
	0	1	0	1	0	1	0	1
0	1,058	159	1,216	1	1,200	17	1,195	22
1	152	18	63	107	25	145	27	143

3.3 Combining question-answering methods

To assist a conversational agent to distinguish between task-specific and commonsense questions in the cooking domain, we set out to train a machine learning classifier to differentiate between the two.

Question type classification We approached question type classification as a supervised machine learning task , aiming to detect features that distinguished both types of questions, which served as a basis for classifying new instances. Two features were created: the number of tokens and the number of characters in a query. Then, the queries were preprocessed by lemmatising, removing stop words and lower-casing. Sentence embeddings were again computed using Sentence-BERT [26], returning a total of 768 features. The five embedding positions with the highest scores on the training data were selected by using the ANOVA F-value between each feature and the label (i.e., indices 2, 112, 284, 320, 420). This resulted in seven features, when added to the two length-based features. A random forest classifier with 100 estimators was used to classify sentences as belonging to the general question type or to the recipe-specific question type.

A total of 359 queries (70% training and 30% evaluation) were selected and written to train and validate the classifier on. These consisted of 223 general cooking questions selected from the previously described Dutch food and drinks questions database, and 136 recipe-specific questions which were manually written based on the types of questions that the cooking assistant was able to answer. Ten splits were created, so that the average performance of the classifier over these different train/evaluation sets could be computed. The classifier's performance was evaluated using precision, recall and F1-score. We compared the Random Forest classifier to a majority baseline.

Results. The random forest classifier performed better on the evaluation set than the majority baseline (Table 3). The accuracy of the classifier was approximately 85% ($M = 0.85, SD = 0.01$, for ten evaluation/test splits).

Table 3: Classifier performance based on 108 test queries

	General			Specific			
	Precision	Recall	F1-score	Precision	Recall	F1-score	Macro-F1
Baseline	0.60	1.00	0.75	0.00	0.00	0.00	0.37
Random Forest	0.87	0.91	0.89	0.85	0.79	0.82	0.86

4 Analyzing the quality of clarification responses in real-world cooking assistant conversations

4.1 Conversational agent architecture

To test our question-answering models in action, we developed a conversational agent with a dialogue management component based on the Information State Update paradigm [30], connected to the Google Dialogflow conversational design interface to handle natural language understanding and interface with the user (Figure 2). We developed Chefbot[11] to plan and manage the cooking instruction dialogue and draw upon the agent's knowledge about recipes and the cooking domain. A Django[12] application was developed to connect Chefbot to Dialogflow.[13] In Chefbot, moves of the agent are specified and linked to preconditions and effects, such as the previous intent of the user and the position in the recipe. The information state is updated based on a move's effects.

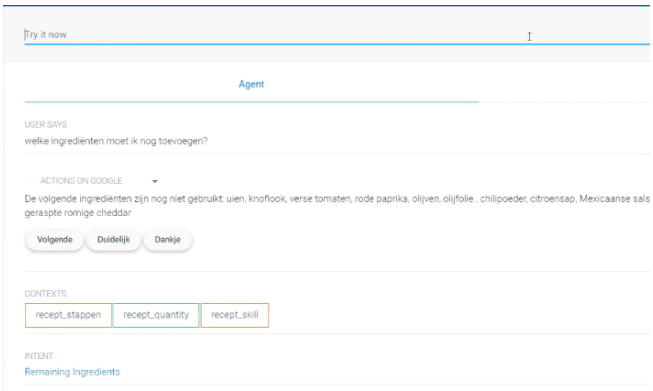

Fig. 2: Google Dialogflow interface.

The general conversation flow was modelled as follows: After an initial greeting, the agent asks the user what recipe s/he would like to cook. When a decision is made, the agent presents the ingredients and utensils on the screen, for the

[11] https://github.com/fkunneman/Chefbot_NCF.git
[12] `https://www.djangoproject.com/`
[13] https://github.com/fkunneman/smoothbot

user to confirm that all preparations have been made. Then the recipe instruction starts, where at each instructed step the user has the option to ask for details or clarification, as well as to ask commonsense cooking questions. When the recipe is completed, the agent does a last topic check and closes the conversation.

4.2 Selecting clarification responses

Whereas answers to common-sense questions could directly be extracted from the database, recipe-specific information was stored as part of the recipe in a Json-file to be retrieved by Chefbot. Based on the heuristics and segmentation approach described in section 3.2, recipes were parsed and transformed into a json-file with the following elements: Recipe title, Number of people, Cooking duration, Ingredients (including unit and amount), Cooking utensils and Recipe steps (consisting of Step description text, Ingredients used in step, Quantities of ingredients used in step, Image and More extensive step description texts).

When a user query was classified task-specific, it was matched to one of the predefined intents (Appendix B). Each of the intents required specific bits of information, which were extracted from the json-file and/or from the context of the conversation (e.g., previous steps, already used ingredients...).

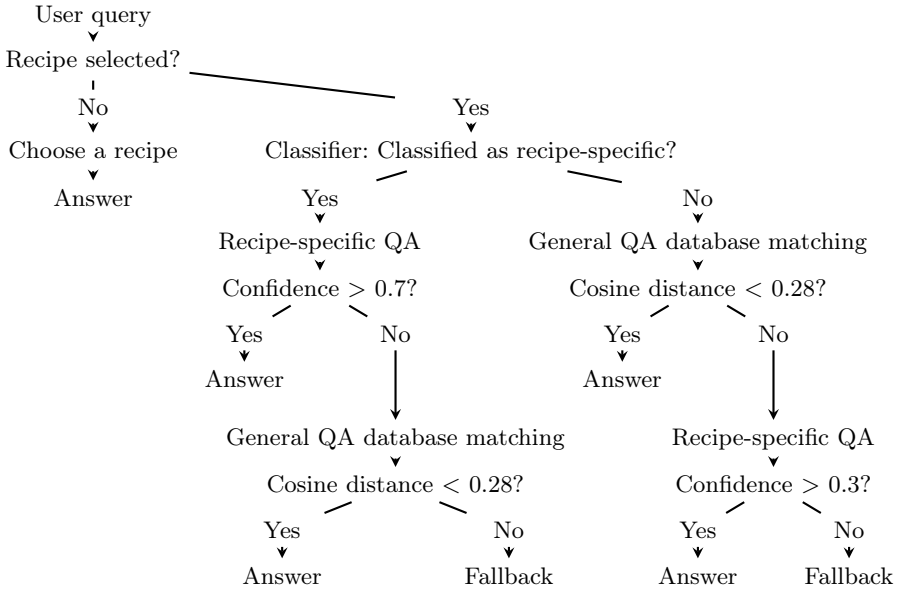

Fig. 3: Decision tree for selecting a response to the user query based on the conversation context, question type classification and answer confidence.

We defined a decision-tree by which the agent decides what knowledge source (task-specific or common-sense) to draw upon to answer a query (Figure 3).

Questions asked before selecting a recipe were answered by drawing upon the knowledge that the agent has of the available recipes and the agent's capabilities. Once a recipe had been selected, the user's query was assessed by the question type classifier (see section 3.3). If the query was classified as general and the distance score was below 0.28 (which was found to be the optimal threshold in Section 3.1), the answer to the best matching question (based on the best-performing BERT with cosine distance approach) in the general QA database was returned to the user. If the user's query was classified as recipe-specific and the confidence of the best matched Dialogflow intent was above 0.7 (empirically determined), the best match's answer was returned to the user. If the confidence was too low, or the distance score too high, the other path was tried. If the threshold was not met, it was checked for the other response-type, now checking for a 0.3 threshold. This lower threshold than the 0.7 described previously was used since it was the value that was set for the first version of the implemented cooking assistant, and allowed to reduce the number of fallback intents. If in this case the confidence was again too low or the distance score too high, a fallback intent was triggered. Otherwise, the best matching answer was returned.

4.3 Experiment design and data analysis

We conducted a user study to evaluate the quality of question answering during cooking instruction.

Participants Six participants cooked a recipe using the cooking assistant. Their average self-reported cooking skills score was 63.5 ($M = 63.50, SD = 17.40$, [19]) out of a maximum score of 98. The participants had some prior knowledge about the study's goal: to assess the performance of a smart cooking assistant. Since the number of participants was relatively small due to the limited resources, they were encouraged to ask as many questions as they could come up with. This allowed the small number of participants to still give the researchers relevant quantitative and qualitative insights, by assessing the types of questions that might arise at the time of cooking a recipe (i.e., in a natural setting), concerning both task-specific knowledge as well as domain knowledge.

Procedure Before using the cooking assistant, participants completed the self-reported cooking skills questionnaire, were instructed about the main goal of the study and received some information about the ingredients and cooking time needed to complete the dish. All participants cooked a spicy Mexican casserole, which was selected based on its linearity (i.e., no temporally overlapping steps) and limited complexity. Each participant cooked the dish in their own kitchen.

The experiment was done by means of a video-call, during which participants were instructed to ask as many questions as they could come up with, while preparing the given dish. The experimenter shared their screen, and entered all user queries, such that it was not necessary for participants to download any of the required software. Then, the answer given by the cooking assistant was returned to the participants. Shortly after finishing the recipe, the participant was asked to fill in the System Usability Scale [5].

Evaluation The quality of clarification responses during the six conversations was measured by manually checking their correctness, based on which precision, recall and F1-score were computed.

Table 4: Classifier performance the 293 queries asked during recipe instruction.

Label	Precision	Recall	F1-score
General	0.53	0.70	0.61
Specific	0.92	0.85	0.88
Weighted avg.	0.85	0.82	0.83

4.4 Results

A total of 297 queries (M =49.50, SD =16.93) were asked. 6 of these were manually labeled as "Before classification" (i.e., before selecting a recipe), 57 as "General" and 234 as "Specific". This is a relatively small dataset. However, it serves as a first indication of the types of questions that users ask in a natural setting, on top of the more artificial setting explained above.

The results of the random forest classifier on distinguishing between general-domain and task-specific questions are reported on in Table 4. The weighted average F1-score was 0.83 for the random forest classifier. General domain questions were more difficult to identify than task-specific questions. Out of the incorrectly classified queries (18%), the heuristics (i.e., maximum distance score or minimum confidence score) corrected 29%. This means that a total of 87% of the queries ended up being correctly classified.

The influence of different components in the decision tree on returning the right answer is presented in Table 5. 66% of all questions were correctly answered, while 87% were correctly classified. This could be partly attributed to directing the wrong answer to a question answering component (21% of the questions were not correctly recognised or failed to pass the threshold-based heuristics), and for the other part to a wrongly retrieved answer by the algorithm or the absence of an answer in the database. The latter applies for a large part to the general QA database, where only for 56% of the questions an answer would be available in the database (had they correctly passed through the decision tree to be retrieved by the Sentence-BERT approach). The threshold-based heuristics accounted for 13 queries that wrongfully did not pass the threshold and 5 queries that were passed to the right question-answering module after a wrong classification.

Out of the incorrect responses, only 6% was still relevant to another part of the recipe, and used by the participant at some point later in time. In 19% of the cases where the agent could not give a correct response, a fallback intent was triggered (e.g., "Can you rephrase your question?"). The remaining 74% of non-correct answers led to confusion amongst participants, since an unexpected answer was returned. Only 25% of the general cooking questions was answered correctly. Of the remaining 75%, most mistakes were made because the posed query was not present in the database. Of the recipe-specific questions, a much higher percentage of 76% was answered correctly.

Table 5: Influence of different components in the conversational agent on returning a wrong answer.

	General		Recipe-Specific		Total	
	N	P	N	P	N	P
# Questions asked	57	1.0	234	1.0	291	1.0
# Questions available in database	32	0.56	205	0.88	237	0.81
Remaining questions after question type classification	40	0.70	199	0.85	239	0.82
Remaining questions after decision tree heuristics	37	0.65	194	0.83	231	0.79
Correct answers returned by QA component	14	0.25	178	0.76	192	0.66

System usability The average score on the system usability scale, from 0 to 100, was 82.08 ($M = 82.08$, $SD = 4.85$). Three participants gave a score above 80, showing that they liked it and would probably recommend it to other people. The remaining participants gave a score between 68 and 79, showing that they thought the agent performed okay, but still needs some improvement.

5 Discussion

The main take-away from our study is that the approaches we took to answering general domain questions and task-specific questions show good performance in a controlled experiment, while this drops considerably when they are posed with questions asked by a user that is involved with the actual task. Our heuristics for handling user requests in combination with our detailed analysis of the conversations permits to pinpoint the two main causes for this performance gap: performance of general QA and confusion between common-sense and task-specific questions. In the following, we will provide explanations for these causes and discuss their implications for an improvement of conversational systems covering a knowledge space of similar width.

As for the first cause, a significant difference was seen in the performance on general cooking-related questions, where the F1 score dropped from 0.89 in the controlled experiment to 0.61 in the user study. This can be foremost ascribed to a mismatch between the used source for general domain questions, a CQA platform where at least part of the queries are posed out of curiosity, and the application area of users that are involved in the act of cooking when posing their questions. Improvement can be made by filtering types of queries from the database that are typically not posed during cooking instructions, and by iteratively adding queries that are not present in the database but have been posed by users. Alternatively, the use of a CQA could be discarded altogether and replaced by a dataset grounded in dialog as [29] have done, which does limit the number of questions that can be addressed.

The second cause, the confusion between common-sense and task-specific questions during the cooking instruction conversations, has led to a considerable portion (21%) of wrong answers given. Part of this can be attributed to the

question type classifier, although it did yield a consistent performance on the controlled evaluation (macro F1 = 0.86) and user study (weighted F1 = 0.83). The heuristics (cosine distance < 0.28 or confidence > 0.7) were another factor that led to a failed match in some cases. These failed question type categorizations were partly due to incomplete formulations of the user, that could be dealt with by using slot-filling. In addition, recipe-specific questions about the quantity of an ingredient were often mistaken for general questions, which could be improved upon by either omitting ingredient names as a feature for question type classification, or by training dedicated classifiers on particularly distinguishing different requests that include a recipe ingredient. Finally, the conversational interface itself could be used, by asking the user for a confirmation when there is confusion between particular question types, or relying on the user to perform conversational repair when a wrong response has been given. [8] show that wrong answers by a home assistant can be instrumental to move forward in a conversation. The system itself should of course be properly equipped to interpret user repair after a wrong response. A final point of discussion is the consideration to refrain from giving an answer when a question is out of scope. This option is currently only triggered in the heuristics when a threshold has not been met, but not integrated as central feature. Arguably, a system that is aimed at covering a wider knowledge space in a domain should also be knowledgeable of what it does not have the answer to. One way to identify questions that can not or should not be answered by the agent is by training a classifier on a set of unanswerable questions, as has been done in [24].

A central limitation to our study is that we only tested the system with six participants who were encouraged to ask many questions to the agent. This has been suitable as a first exploration, but in future work this number should be higher to draw strong conclusions, and participants should not be tasked with anything else than making the recipe to learn more about system performance and common questions that are asked. Apart from that, a strength of the current set-up is that participants were actually in their kitchen when talking to the assistant, and the thorough analysis of the conversations has given a clear indication of system requirements and challenges when expanding on the agent's knowledge. A question that the outcomes raise is whether a task-based conversational system should cover a long tail of questions that may someday be asked by a user, at the cost of confusing more questions with different ones. The current study shows that this cost is too significant, but the gain may be worthwhile when the rate of wrong answers can be reduced. Our study highlights a number of directions to explore further to this end.

6 Conclusion

We set out to increase the coverage of clarification responses for a cooking assistant, by drawing upon a community question answering platform for answering common sense questions related to cooking and a recipe platform for modelling recipe-specific knowledge. The approaches to answer cooking-related questions based on these data were tested in a controlled set-up and in a user study as part

of a cooking assistant. The outcomes of the user study supported that having to distinguish between commonsense and task-specific questions can lead to a considerable proportion of questions that are not answered by the right module. In addition, much improvement can be made by increasing the coverage of common sense questions. Testing the quality of clarification responses in a user study has been vital to gain empirical insights into the information-seeking challenges of an instruction conversation in a broad knowledge domain.

References

1. Agichtein, E., Maarek, Y., Rokhlenko, O.: Alexa prize taskbot challenge (2022)
2. Alfifi, M., Dong, X., Feldman, T., Lin, A., Madanagopal, K., Pethe, A., Teleki, M., Wang, Z., Zhu, Z., Caverlee, J.: Howdy y'all: An alexa taskbot (2022)
3. Baccouri, N.: deep-translator. `https://github.com/nidhaloff/deep-translator` (2020)
4. Beddiar, D.R., Jahan, M.S., Oussalah, M.: Data expansion using back translation and paraphrasing for hate speech detection. Online Social Networks and Media **24**, 100153 (2021). https://doi.org/https://doi.org/10.1016/j.osnem.2021.100153, `https://www.sciencedirect.com/science/article/pii/S2468696421000355`
5. Brooke, J.: Sus: A quick and dirty usability scale. Usability Eval. Ind. **189** (11 1995)
6. Burges, C.J.C.: Towards the machine comprehension of text: An essay. (2013)
7. Chang, M., Guillai, L.V., Jung, H., Hare, V., Kim, J., Agrawala, M.: Recipescape: An interactive tool for analyzing cooking instructions at scale. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. ACM (2018)
8. Cho, J., Rader, E.: The role of conversational grounding in supporting symbiosis between people and digital assistants. Proceedings of the ACM on Human-Computer Interaction **4**(CSCW1), 1–28 (2020)
9. Damodaran, P.: Parrot: Paraphrase generation for nlu. (2021)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
11. Edunov, S., Ott, M., Auli, M., Grangier, D.: Understanding back-translation at scale. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing p. 489–500 (2018). https://doi.org/10.18653/v1/D18-1045
12. Feng, S., Wan, H., Gunasekara, C., Patel, S., Joshi, S., Lastras, L.: doc2dial: A goal-oriented document-grounded dialogue dataset. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 8118–8128. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.emnlp-main.652, `https://aclanthology.org/2020.emnlp-main.652`
13. Gargett, A., Garoufi, K., Koller, A., Striegnitz, K.: The give-2 corpus of giving instructions in virtual environments. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10) (2010)
14. Gemmell, C., Fischer, S., Mackie, I., Owoicho, P., Rossetto, F., Dalton, J.: Grillbot: A flexible conversational agent for solving complex real-world tasks. 1st Proceedings of the Alexa Prize Taskbot (2022)
15. Groep, J.: Gelegenheid recepten, data retrieved from smulweb.nl, on 10/12/2021, `smulweb.nl/recepten/gelegenheid`

16. Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spaCy: Industrial-strength Natural Language Processing in Python (2020). https://doi.org/10.5281/zenodo.1212303, https://doi.org/10.5281/zenodo.1212303
17. Jian, Y., Zaporojets, K., Deleu, J., Demeester, T., Develder, C.: Extracting structured data from recipes using conditional random fields. Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing pp. 821–826 (2020)
18. Lample, G., Conneau, A.: Cross-lingual language model pretraining. 33rd Conference on Neural Information Processing Systems (2019)
19. Lavelle, F., McGowan, L., Hollywood, L., Surgenor, D., McCloat, A., Mooney, E., Caraher, M., Raats, M., Dean, M.: The development and validation of measures to assess cooking skills and food skills. International Journal of Behavioral Nutrition and Physical Activity **14**(1), 118 (2017)
20. Lin, Y.T., Kuo, H.C., Xu, Z.S., Chiu, S., Hung, C.C., Chen, Y.C., Huang, C.W., Chen, Y.N.: Miutsu: Ntu's taskbot for the alexa prize. arXiv preprint arXiv:2205.07446 (2022)
21. Maeta, H., Mori, S., Sasada, T.: A framework for recipe text interpretation. Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing pp. 553–558 (2014)
22. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
23. Neumann, N., Wachsmuth, S.: Recipe enrichment: Knowledge required for a cooking assistant (2021). https://doi.org/10.5220/0010250908220829
24. Rajpurkar, P., Jia, R., Liang, P.: Know what you don't know: Unanswerable questions for squad. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 784–789 (2018)
25. Rastogi, A., Zang, X., Sunkara, S., Gupta, R., Khaitan, P.: Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. Proceedings of the AAAI Conference on Artificial Intelligence **34**(5), 8689–8696 (2020)
26. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. EMNLP (2019)
27. Stoyanchev, S., Keizer, S., Doddipatla, R.: Action state update approach to dialogue management. ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) pp. 7398–7402 (2021)
28. Strathearn, C., Gkatzia, D.: Chefbot: A novel framework for the generation of commonsense-enhanced responses for task-based dialogue systems. Journal of the ACM pp. 46–47 (08 2021)
29. Strathearn, C., Gkatzia, D.: Task2dial: A novel task and dataset for commonsense enhanced task-based dialogue grounded in documents (04 2022)
30. Traum, D.R., Larsson, S.: The information state approach to dialogue management. Springer Netherlands, Dordrecht **6**(3), 325–353 (2003)
31. Van-Tu, N., Anh-Cuong, L.: Improving question classification by feature extraction and selection. Indian Journal of Science and Technology **9** (2016)
32. Xue, X., Jeon, J., Croft, W.: Retrieval models for question and answer archives. pp. 475–482 (01 2008). https://doi.org/10.1145/1390334.1390416
33. Zaib, M., Zhang, W.E., Sheng, Q.Z., Mahmood, A., Zhang, Y.: Conversational question answering: A survey. Knowledge and Information Systems pp. 1–45 (2022)

A General question-answering: question types

Questions in seven categories were selected:

1. Replacing ingredients: with what a certain ingredient can be replaced (e.g., with what can I substitute golden caster sugar?).
2. Meaning of ingredients: the question might rise what a certain ingredient is (e.g., what are goji berries?).
3. Difference between ingredients: what is the difference between two (similar) cooking ingredients (e.g., what is baking soda?).
4. Ingredient sustainability: how long an ingredient can be kept before it goes bad (e.g., how long is opened, organic coconut oil sustainable??).
5. Cooking time of ingredients: how long it takes to bake/cook a certain ingredient (e.g., for how long does sweet potato need to bake in the oven?)
6. Health: whether some ingredient or its usage is healthy (e.g., is it unhealthy to re-use a tea bag?)
7. Performing cooking techniques: how a certain cooking technique is performed (e.g., how do I boil water quickly in a pan?).

Language	Units
NL	gram, gr., gr, g., g, kilogram, kilo, kg., kg, liter, l., l, milliliter, ml., ml, centiliter, cl., cl, deciliter, dl., dl, eetlepel, eetlepels, el., el, theelepel, theelepels, tl., tl, kop, bos, zak, beetje, plak, plakken, scheut, handje, snuf, pond, ons, pint, tak, teen
EN	gram, gr., gr, g., g, kilogram, kilo, kg., kg, liter, l., l, millilitre, ml., ml, centilitre, cl., cl, decilitre, dl., dl, tablespoon, tablespoons, tbsp, tbsp, teaspoon, teaspoons, tsp, tsp, cup, bunch, bag, bit, slice, paste, dash, hand, snuff, pound, ounce, pint, branch, clove

Table 6: Cooking domain predefined measurement units and English equivalents.

B Recipe-specific question-answering: intent types

Which recipes. Ask which recipes are available for the user to cook.

Cooking recipe. Choose which recipe to cook, out of the available ones from Chefbot's database.

Confirm recipe. Confirm that they want to make the chosen recipe.

Number of people that the recipe is meant for. The user can ask the agent for how many people the recipe is intended. Users could subsequently use this information in order to adapt the quantities of the ingredients to the number of people for whom they want to cook.

Estimated preparation time. The estimated time for preparing the recipe has been added to the recipe's context. This allows the user to ask the agent how long it should take them to prepare a certain recipe.

Recipe name. Allows the user to ask what the name of the current recipe was.

Time or steps passed/left. The user can ask how far along they are with the recipe, or how much is left, in terms of time and/or in terms of steps. Such a question would lead to the answer "According to the recipe, the preparation should take around 30-60 minutes. You have already been cooking for 27 minutes. You have executed 14/20 steps.".

Ingredients not used yet. The user can ask the agent about the recipe that have not been used yet in previous steps. This allows them to check whether they are still on the right track, or whether there are any ingredients that should have been used already, but have not been used yet. Additionally, this functionality can be helpful whenever a recipe step states to "add all the remaining ingredients". The user could then ask which are the remaining ingredients.

Continuation. Go to the next step after completing the current one.

Repeat. Repeat the current step.

Previous step. Go back to the previous step.

Update. Let the agent know that the current step has been executed

Elicit. Ask for some clarification whenever a step is unclear. The agent can only respond to this when a step has been written in a more detailed and a more basic way.

Accept repair: show gratitude. Thank the agent for clarifying.

Accept repair: understood. Let the agent know that the clarification is understood.

How much. Ask how much of the ingredient in the current step is needed.

How to. Ask to explain how a specific cooking technique needs to be executed. For this, the technique needs to be explicitly explained within the recipe.

Motivate. Ask why a certain step is needed.

Close recipe. Finish the recipe.

C Recipe annotation guidelines: ground truth

Recipe name. The recipe name was the title of the recipe at the top of the page.

Number of people. The number of people for whom the recipe is intended, was indicated at the beginning of the ingredients list as follows: "Ingredients for N people".

Category. Found below the title of an individual recipe on smulweb.nl [15].

Cooking time. Found below the categories of an individual recipe on smulweb.nl [15].

Ingredients list. The ingredients were split up in three parts (if present): the amount, the measurement unit, and the name of the ingredient.

Utensils. For the cooking utensils, the list was split up depending on its format. The most frequent formatting was a list of comma-separated utensils names, however, other options also occurred, such as bullet-point lists.

Procedure. In order to split up the procedure, first of all, it was split up into sentences. If one sentence contained more than one cooking action (e.g., boil the water and put the pasta in it), this was again split up into two separate steps. Whenever there are multiple ingredients, for all of which the same action needs to be performed (e.g., add the onion and the garlic), they were kept together in one same step.