ISSN 2499-4553

IJCoL

Italian Journal of Computational Linguistics

Rivista Italiana di Linguistica Computazionale

Volume 8, Number 1
june 2022

aA Accademia university press

Registrazione presso il Tribunale di Trento n. 14/16 del 6 luglio 2016

Rivista Semestrale dell'Associazione Italiana di Linguistica Computazionale (AILC)

Associazione Italiana di Linguistica Computazionale

direttore responsabile
Michele Arnese

isbn 9791255000167

Accademia University Press
via Carlo Alberto 55
I-10123 Torino
info@aAccademia.it
www.aAccademia.it/IJCoL_8_1

IJCoL

Volume 8, Number 1
june 2022

CONTENTS

Graph-based representations of clarification strategies supporting automatic dialogue management*

Valentina Russo**
Logogramma S.r.l.

Azzurra Mancini†
Logogramma S.r.l.

Marco Grazioso‡
Università degli studi di Napoli
"Federico II"

Martina Di Bratto§
Università degli studi di Napoli
"Federico II"

This paper aims at presenting a dialogue-oriented approach to the construction of a graph knowledge base (KB) supporting task-oriented human-machine interactions. In particular, we focus on different pragmatic scenarios, facing the Common Ground issue and arguing that knowledge bases (in the form of graphs) are needed to make a clarification and recover pieces of information when inconsistencies occur during the communicative exchange.

The main contributions of this work are: 1) a flexible dialog system architecture designed to be plugged into existing service infrastructures, 2) a graph-based knowledge representation protocol to manage both dialog domain and dialog management, 3) a detailed investigation of clarification requests forms with respect to their functions. After a brief introduction (see Section 1), we present: the theoretical underpinnings of the paper and the background work (see Section 2) our system architecture (see Sections 3 and 4) and the clarification requests (CRs) issue (see Section 5); our CRs classification, and some examples in context (see Section 6).

1. Introduction

In this paper, we present a way to handle some of the main pragmatic issues that a task-oriented Dialogue System (DS) should manage to achieve the communicative goal (and the task completion), arguing that the construction of a back-end Knowledge Base (KB) should not only be concerned with domain representation but also with dialogues history tracking and specialised recovery strategies.

∗ The present study is the result of the collaborative work of all the authors and the research group of Logogramma S.r.l. working on the AI.CODIUM® Project. Paragraphs 1, 6, 6.1, 6.2 have been written by Valentina Russo, 4, 4.2 by Azzurra Mancini, 5, 5.1 and 5.2 by Martina Di Bratto and 2, 3, 4.1, 7 by Marco Grazioso.

∗∗ Logogramma S.r.l. (Naples, Italy) - Urban/Eco Research Center, University of Naples "Federico II" (Naples, Italy) E-mail: vrusso@logogramma.com

† Logogramma S.r.l. (Naples, Italy) - Università degli studi di Napoli "L'Orientale" - Urban/Eco Research Center, University of Naples "Federico II" (Naples, Italy) E-mail: amancini@logogramma.com

‡ Department of Electrical Engineering and Information Technologies (Naples, Italy) - Logogramma S.r.l. (Naples, Italy) E-mail: marco.grazioso@unina.it

§ Department of Humanities (Naples, Italy) - Logogramma S.r.l. (Naples, Italy)
E-mail: martina.dibratto@unina.it

The importance of pragmatics in dialogue management has increased, in recent years, following the significant breakthroughs concerned with semantics. Multi-turn interaction, in particular, greatly benefits from considerations that go beyond one-shot interpretations and that rely on previously collected knowledge. This is not limited to the context of a single dialogue but can span over previous interactions as well. In this context, ambiguities, belief conflicts, and misunderstandings require adequate strategies that should be explicitly represented in software modules providing support to dialogue managers. Pragmatic issues are mostly not handled in commercial dialogue systems. Our proposal integrates a detailed analysis of clarification requests and their representation in the knowledge graph to support the development of disambiguation strategies.

We will thus present an approach to the construction of a closed-domain Graph KB that we define "dialogue-oriented" since it stores information about the domain as well as information on how to manage dialogue to overcome uncertainty or ambiguity occurring during slot-filling or intent and entity recognition. Our KB is designed to help handle a problematic conversation, enabling a Dialogue Management (DM) module to pose the most correct Clarification Request (CR) in different pragmatic scenarios to correctly ground information.

After a brief introduction to the background work (see Section 2), in Section 3 we will present our infrastructure for the development of task-oriented dialogue systems; in Section 4 we will focus on the KB architecture, showing how data are represented in our graph model, through a real use case: a prototype in a commercial domain.

Then we will explore the grounding issue, presenting the theoretical background we refer to in human-human as well as in human-machine communication (see Section 5).

Finally, we will propose a further classification of Clarification Requests (see Section 6) and we will present some cases of miscommunication that the machine can solve by relying on our graph KB model, exemplifying them through some use cases.

We will then draft our conclusions (see Section 7).

2. Related work

Following the classification provided in Deriu et al. (2021), dialogue systems can be classified into three main categories:

- Task-oriented systems: developed to help the user solve a specific task as efficiently as possible;
- Conversational agents: display a more unstructured conversation, as their purpose is to have open-domain dialogues with no specific task to solve;
- Question answering systems: built for the specific task of answering questions concerning a specific domain.

A further classification concerns the restrictiveness of the domain which can be open, without strict boundaries, or closed. While in open-domain systems a single infrastructure may be enough to develop many dialogue systems, closed-domain systems do not allow to do the same and often require a single system for each target domain (Nakano and Komatani 2020). In this work, we present a domain-independent infrastructure based on the use of knowledge graphs for the development of task-oriented dialogue systems.

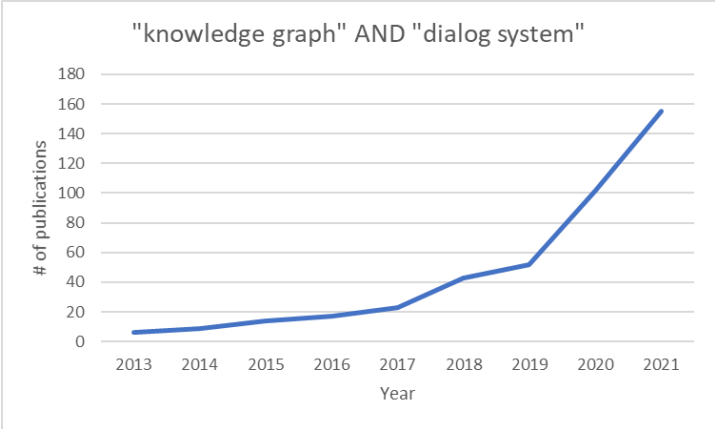

Figure 1
Publications on the use of knowledge graphs and dialog systems (source: Google Scholar).

In the literature, several works represented knowledge through graph databases in different domains of application. Origlia et al. (2019) considered them very important for the realisation of interactive experiences, so much to be included in their framework *FANTASIA*. KBot (see Ait-Mlouk and Jiang (2020)) is a chatbot system based on knowledge graph database that leverages the technologies of machine learning and natural language understanding, including named entity recognition, factoid, and recurrent questions, as well as dialogue management, which improved the end-to-end user experience in terms of interactive question answering and performance. In Bao, Ni and Liu (2020) a chatbot framework adopts a hybrid model which consists of a knowledge graph and a text similarity model for answering complex medical questions. Miliani et al. (2019) designed a system called Text Frame Detector which populates a frame-based ontology stored in a graph database.

Nowadays, graph databases are largely used in the field of dialogue systems to both represent the domain knowledge and support the dialogue management, since they provide a natural way of showing concepts and relations among them. Several approaches have been developed, each one conceived for a specific task. As shown in Figure 1, studies on the usage of knowledge graphs for dialogue management in dialogue systems have increased during the last years.

Most of the works implement frame-based dialogue management strategies (Jokinen and McTear 2009), where the user intention is mapped to a structure identifying the slots to be filled. Therefore, a dialogue system must be able to identify the frame and retrieve the entities from the userâĂŹs utterance associating them to the corresponding slots (see Section 4). In Zhao et al. (2021), the authors presented a graph-based dialogue system for disease diagnosis. Specifically, they built a weighted heterogeneous graph based on symptom co-occurrence and the proposed symptom frequency-inverse disease frequency. This kind of implementation works well for disease diagnosis (since the conversation is led by the DS) but it lacks portability to other domains. In Nakano and Komatani (2020), the authors presented a framework for developing closed-domain chat dialogue systems and employed a graph database containing food and restaurant information as a case study. The proposed system goes in the direction of our idea of domain-independent and adaptable representation but it does not provide a fine-grained response classification for clarification needs. Also, it does not store dialogue

management information (functions, responses, etc.) in the graph database. This is an important issue for dialog systems generalisation: by moving management aspects that are domain-specific from the dialog manager to the knowledge base, it is possible to design engines that dynamically request domain-specific information to support high-level strategies. For example, posing a question to fill a slot in a frame-based system is a task pertaining to the dialogue manager but which slot is mandatory and what is the actual question to pose are data to be retrieved from the database. In principle, as long as domain-specific dialogue management strategies are consistently represented, in relationship with the knowledge domain, there is no need to modify the dialogue manager, which becomes a very portable module.

To overcome the problem of managing dialogue inconsistencies, we developed an infrastructure capable of retrieving both the most appropriate CR to find missing information and also using them to produce strategies to address grounding conflicts. Retrieving the most appropriate CR given the user's utterance is an open problem that has been widely investigated in human-human conversations (see Section 5) and that is currently investigated in human-machine communication. In an attempt to face it, the ClariQ challenge was organised as part of the Conversational AI challenge series (ConvAI3) at Search-oriented Conversational AI (SCAI) EMNLP workshop in 2020. In particular, they built an open-domain dialogue corpus with manually annotated clarifying questions (Aliannejadi et al. 2021). All the submitted solutions tried to solve the problem by using neural-based approaches but none combined it with a knowledge base support. Di Maro et al. (2021a) presented a way of using graphs to find Common Ground (CG) inconsistencies in dialogue but the focus was not on the solutions needed to overcome disambiguation through dialogue strategies.

3. System architecture

The dialogue-oriented KB we propose is part of a broader project called AI.CODIUM®. There are three main goals behind the infrastructure design:

- enabling domain-specific language understanding;
- enabling the management of different pragmatic scenarios through the use of appropriate CRs;
- enabling domain-independent dialogue management.

In order to test our infrastructure, we developed a prototype of a task-oriented DS in the commercial domain, capable of recognising 54 speakers' intentions and of managing dialogue to collect the information needed to pursue a task.

In Figure 2, we show the general architecture of the prototype to better represent the concepts behind our idea. The knowledge graph is the main module and it is in charge of defining:

- the available intents to be recognised and the related slots to be filled;
- the slot-filling rules and the entity validation functions (as hooks to the user's defined functions);
- the questions to be prompted by the system.

The user request is processed by the Natural Language Understanding (NLU) module which could be implemented using different strategies, from rule-based approaches to neural ones (as we did in our prototype), possibly using data stored in the graph KB. Then, the Dialogue Management module defines both the filling/validation functions and the dialogue flow policy (state tracking/update and next action selection) by querying the graph KB. In particular, since functions and parameters are not hard-coded but are stored in the graph as nodes, it is easy to modify the system behaviour by changing relations and/or properties without rewriting the system source code. As described in the next sections, the infrastructure aims at making the dialogue efficient, by retrieving, where possible, all the information required from internal (KB) or external sources (web services), without asking for continuous user confirmation.

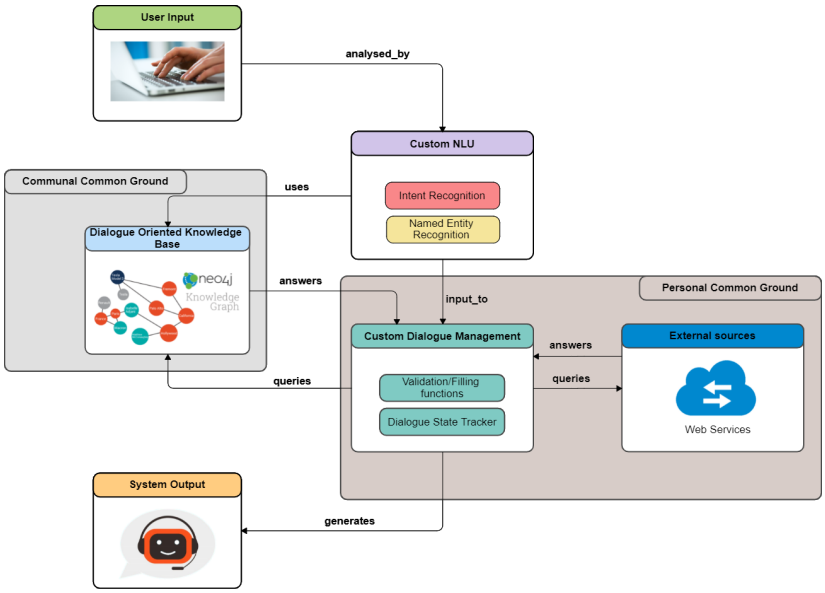

Figure 2
Overall architecture schema showing the modules communication and the pragmatic area they belong to.

Our aim is to focus on the pragmatic issue of CG, going beyond the intent-entity-slot issue and showing how important a CR management is, not only for the sake of the task completion but also in terms of naturalness of the dialogue. We will thus leave out all the technical details regarding the software as well as details concerning the NLU module. In fact, one of the advantages of our infrastructure is that it is possible to use it with different dialogue strategies.

Note that we have not compared our system with another framework. This is because we have not yet found any existing framework for selecting the most appropriate CR that we can compare with AI.CODIUM®. Nonetheless, we are planning an evaluation phase of our prototype, which will be discussed in the conclusions (see Section 7).

4. Concepts and definitions

> "All modern task-based dialogue systems, whether the simple GUS architecture [...] or the more sophisticated dialogue state architectures [...] are based around frames. A frame is a kind of knowledge structure representing the kinds of intentions the system can extract from user sentences, and consists of a collection of slots, each of which can take a set of possible values. Together this set of frames is sometimes called a domain ontology." (Jurafsky and Martin 2019)

Frames are one of the most used strategies to represent information. Most of the approaches based on frames trace back to Fillmore's (1976) definition, with FrameNET focusing on semantic frames triggered by Lexical Units (Fillmore, Lee-Goldman, and Rhodes 2012), while others show slightly different approaches to the definition of the frame, as in Miliani et al. (2019) where each frame is considered as a "domain entity".

In our approach, a frame is the representation of a communicative situation (or scenario) considered from an ontological and pragmatic point of view. Our frames were designed according to three criteria: a) retrieved communicative intention, b) task to be accomplished and c) focused entities. This structure, together with the annotated data set, was stored in a graph database (i.e. Neo4j, see Section 4.1)[1].

We defined the following elements:

Frame âĂŞ A frame is the highest level of representation in our hierarchy, consisting of a collection of all the elements involved in a specific communicative intention. The frame concept is a high-level abstract unit that consists of the low-level representation of intents and their relations, and which is not directly represented through the definition of a specific frame node.

Intent âĂŞ Each intent represents a specific communicative intention and it is designed to encompass both slots and entities. Each intent may have zero, one or more slots in a 1:*n* relation. Data are organised through the definition of micro- and macro-intents, where micro-intents are related to the respective macro-intents through parent relations, and to other intents through peer and preparatory relations (see Figure 3). Moreover, each micro-intent can have one or more slots to fill, representing the data required to accomplish a specific task.

Slot - A slot is an element that is necessary or optional to fulfil tasks. Each slot has a 1:1 relationship with entity types that may be assigned to it. Each slot has properties defining its behaviour depending on the related intent (e.g.: mandatory *vs* optional, priority, etc.).

Entity - Entity types define the objects involved in each task and are related to their linguistic cues. They can be of different types (named

1 In a graph database based on the labelled property graph model, data are represented following graph theory by defining nodes, corresponding to *vertices*, and relations, corresponding to *edges*. Nodes are often used to represent entities, may contain properties, in the form of key-value pairs, and may be associated with one or more labels. Relationships between nodes must have a single label and could contain properties as well as the nodes do.

entities but also domain-specific entities) and can have different values, i.e. their linguistic cues, their reference in the text.

Entity Dictionary - Each entity type can have n linguistic cues that are salient for the slot filling and that can be retrieved in specific lists or external sources (the ones stored in the Graph KB are labeled as "Dictionary entries"). Furthermore, the dictionaries can be used for Named Entity Recognition (NER) training and internal validation processes (see Section 4.1).

The proposed architecture reveals to be very effective to clarify ambiguity and non-understanding, during both intent and entity recognition tasks and the dialogue management, thanks to the representation of all of their relations in a graph.

From the point of view of Dialogue Management, we can define the following further elements:

Filling functions - In case of missing information, filling functions rule how the object of a specific slot should be elicited through CRs. Such data are stored in the KB instead of being treated in specific (external) Dialogue Management modules.

Validation functions - Once the slot is filled, a validation function rules how to verify if the retrieved entity can be properly grounded or needs further clarification (disambiguation).

Actions - Actions are mapped onto concrete system procedures which could instantiate several behaviours, like redirecting to a specific URL, calling a web service to retrieve information, or any other custom procedure. Actions that require parameters can be defined by relating them to one or more Action Parameter nodes.

Questions - Types of requests to be used under particular conditions during the dialogue flow (see Section 6).

All of the mentioned elements, both the ones pertaining to the specific domain (as frames, intents, slots, entities and entities dictionaries) and the ones pertaining to the dialogue management level (i.e. information on how to use Domain Knowledge while managing the dialogue, as filling and validation functions and action nodes) are connected by means of *n*-ary relations and convey their own attributes to further define data and metadata (e.g. descriptions, types, sources, etc.). We will refer to the first ones as knowledge nodes representing ontological knowledge and to the second ones as dialogue nodes representing dialogic knowledge (see also Jannach (2020) who goes in the same direction, talking about CRS, Conversational Recommender Systems)[2].

2 "Various types of knowledge are used in CRS. [...] different types of Domain and Background Knowledge are often leveraged by CRS. Many approaches explicitly encode dialogue knowledge in different ways, e.g., in the form of pre-defined dialogue states, supported user intents, and the possible transitions between the states" (Jannach et al. 2020, p. 105:4).

4.1 The Graph representation

To represent our data we defined a set of nodes, associated with meaningful labels, connected by directed relations.

In Figure 3, we show our graph structure organised into two subcategories: *knowledge nodes* and *dialogue nodes* (see Table 1).

Table 1
Table showing node types.

Node label	Category
MACRO_INTENT	Knowledge
MICRO_INTENT	Knowledge
SLOT	Knowledge
ENTITY	Knowledge
ENTITY_DICTIONARY	Knowledge
ENTRY	Knowledge
FILLING_FUNCTION	Dialogue
VALIDATION_FUNCTION	Dialogue
ACTION	Dialogue
UTTERANCE	Dialogue
QUESTION	Dialogue

The first category includes all the nodes related to the domain knowledge:

- MACRO_INTENTs represent the user intentions and they are connected to MICRO_INTENTs through HAS_MICRO_INTENT relations.
- MICRO_INTENTs are then connected to SLOT nodes representing the needed information for handling the task, with HAS_SLOT relations.
- SLOTs refer to the corresponding ENTITY nodes through HAS_ENTITY relations.
- ENTITY nodes, then, are connected through HAS_DICTIONARY relations to ENTITY_DICTIONARY nodes which contain information about the possible data associated with an entity. Regarding the ENTRY nodes, they have been conceived to be populated in different ways. If an entity can be represented by a finite set of items, it can be manually added to the KB and connected to the ENTITY_DICTIONARY node through HAS_ENTRY relations, or it can be imported by external sources like supplier's databases, linked open data (e.g. Wikidata[3]), as described in Grazioso et al. (2018), and more. Otherwise, the entries can be defined by a set of rules or patterns identifying a specific entity type (e.g., tax code, dates or product ID).
- Utterances contained in our collected data-set have also been included in the graph database by defining UTTERANCE nodes and connecting them

3 https://www.wikidata.org/wiki/Wikidata:Main_Page

with the corresponding MICRO_INTENT through HAS_INTENT relations.

All the nodes and properties strictly related to dialogue management belong to the second category. We defined four kinds of nodes handling information useful to support a human-agent dialogue: FILLING_FUNCTION, VALIDATION_FUNCTION, QUESTION and ACTION.

- Each node has a specific task, FILLING_FUNCTION provides information about the slot filling and it is conceived to support the development of an event-driven interaction. Its properties define the behaviour associated with a SLOT, providing, among others, a *fire_on* property, having a value in the set *filled, not_filled*, and a *type* property which defines, through the corresponding ACTION nodes, if the piece of information to fill the slots has to be retrieved locally, by querying the graph database or remotely, by querying external services, and which CR is the most appropriate to elicit such data.
- VALIDATION_FUNCTION stores information needed to validate an entity for a slot either locally, by querying the graph, or externally, by querying an external service. When the validation fails, the system can retrieve a CR (QUESTION node) to update the acquired piece of information before grounding it, through HAS_QUESTION relations.
- QUESTION nodes provide all the possible requests to be prompted based on the situation, they are identified through their syntactic form and pragmatic function (validation, disambiguation, etc.) and can be triggered via their type (see Section 6)[4].
- ACTION nodes define the operations to be performed at different stages of the task completion: 1) validating information (both formal or existence validations, e.g. a product ID is formally correct and/or it exists in the pointed source), 2) triggering the right questions in case of missing information for the slot-filling, 3) instantiating the output in the final stage of the task fulfilment, i.e. the answer to the user's request (giving a piece of information, showing a result, redirecting to a web page, forward the request to a human operator, etc.).

Finally, since interrelations between intents could occur, we also defined the following relations: parents, peers and preparatory. They are important for dialogue management and will be described in detail in the Sections 6.1.1 and 6.2. The above defined nodes and relations (see Table 1) are conceived to be general and domain-independent in order to provide a set of standard procedures capable of retrieving the information required to drive the dialogue towards the task completion.

4 How the question nodes are populated depends on the single project. Once defined the types, they can be manually added or generated by means of NLG techniques.

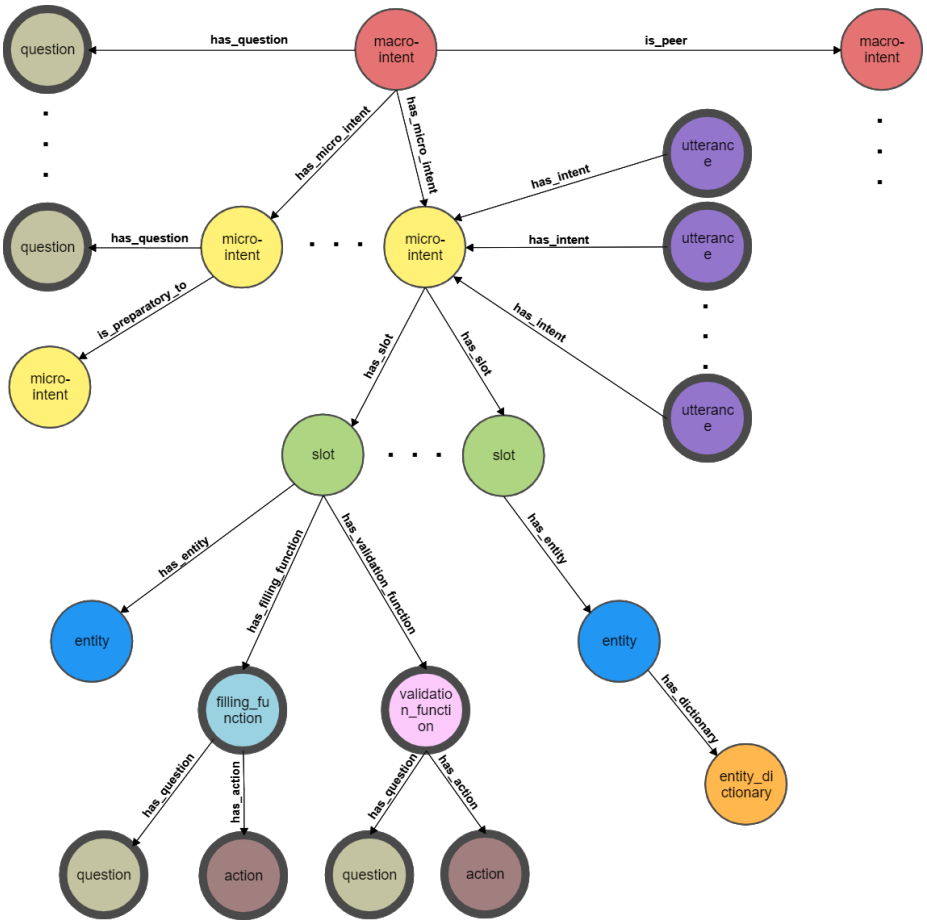

Figure 3
Graph representation of the knowledge base. It contains dialogic nodes (FILLING_FUNCTIONs, VALIDATION_FUNCTIONs, ACTIONs, and QUESTIONs) which could be used to manage the dialogue flow, and ontological nodes (MACRO_INTENTs and MICRO_INTENTs, SLOTs, and ENTITIES) representing the domain specific knowledge. Moreover, utterance nodes are connected to the intents and could be used to train intent recognition models. Knowledge and Dialogue nodes are identified respectively by thin and thick borders.

4.2 A case study

To better illustrate our infrastructure design, we set up a proof of concept by building a prototype of a task-oriented DS in the commercial domain.

We collected a corpus of conversations selected among Customer Assistance interactions in the field of commerce transactions (see Mancini and Russo (2022)). Our first aim was to create a detailed tag-set to minimise ambiguity since the very beginning. The corpus was built starting from human-human commerce interactions in Italian. To collect the widest set of linguistic expressions of the communicative intentions we focused on, we selected all the speakers' intents belonging to the Dialogue Act (DA) "Info-requests", following the latest accepted version of multi-functional ISO standard 24617-2 for DA annotation (see Roccabruna et al. (2020)). We thus selected data among

the diamesic varieties of spoken and written human-human interactions, ranging over more diatechnic varieties. In particular, we selected our data from:

- spoken conversations transcriptions extracted by phone calls in a business-to-business domain,
- written conversations extracted by a help-desk chat in a business-to-consumer domain,
- written conversations extracted by help-desk chats and emails in a business-to-business domain.

Once the corpus was built, we annotated a large set of micro-intents, grouping them together into few macro-intents. The corpus consists of about 10.000 sentences annotated as intents, after a strict annotation agreement check[5]. We thus represented all the retrieved macro- and micro-intents in a set of hierarchically related frames, each one carrying information about slots, related entity types, and more.
Although several classifications of business and commerce transactions already exist âĂŞ e.g. the commerce scenario in FrameNet[6]âĂŞ we designed a more detailed and fine-grained hierarchy to tackle ambiguity and distinguish among different intents that are not similar from a pragmatic point of view but refer to the same objects in the external world and thus can be easily misunderstood by the machine (and by humans too). The tag-set contained a wide set of micro-intents that have been further enriched in their architecture depth with another layer of macro-intents, using a bottom up approach.
In Figure 4 we show an example in context. Consider a frame - in our domain of e-commerce transactions - that encompasses the scenario of a complaint. This frame, with a macro-intent called 'Complaint' (i.e. a complaint regarding something bought that did not satisfy the customer), can encompass different pragmatic intentions that involve the same objects or entities (i.e. an order or a product).

These communicative needs can be represented by the three micro-intents through which the user may ask for: 'Refund', 'Replacement' or 'Missing item'. All of these micro-intents involve the same entities in a slightly different way having different slot filling rules and functional properties.

The macro-intent encompasses 4 slot types (Order ID, Product ID, Original product ID, Updated product ID) that can be filled by two entity types (Order ID and Product ID) and each slot has different properties depending on the micro-intents it is attached to. In fact, during a 'Refund' the slots Order ID and Product ID are not in a hierarchical relationship so that either of the two can be used to trigger the task. On the contrary, during a request for 'Missing item' both the slots are mandatory (see Fig. 4). As for entities, while in 'Refund' and 'Missing item' requests we just have one occurrence of each entity type, during the 'Replacement' we could need to differentiate among two occurrences of the same entity type (Product ID) filling two different slots (Original product ID *vs* Updated Product ID) when, for example, the customer requests a replacement with a different product. Considering entity validation, while during 'Refund' and 'Missing item' we already have all of the needed data stored in the (external) KB of the customer's history, being thus able to double-check information collected during

5 We will not go into details here since our focus is on the knowledge representation coming out from our annotation rather then on tagging itself.
6 See: https://framenet.icsi.berkeley.edu/fndrupal/frameIndex.

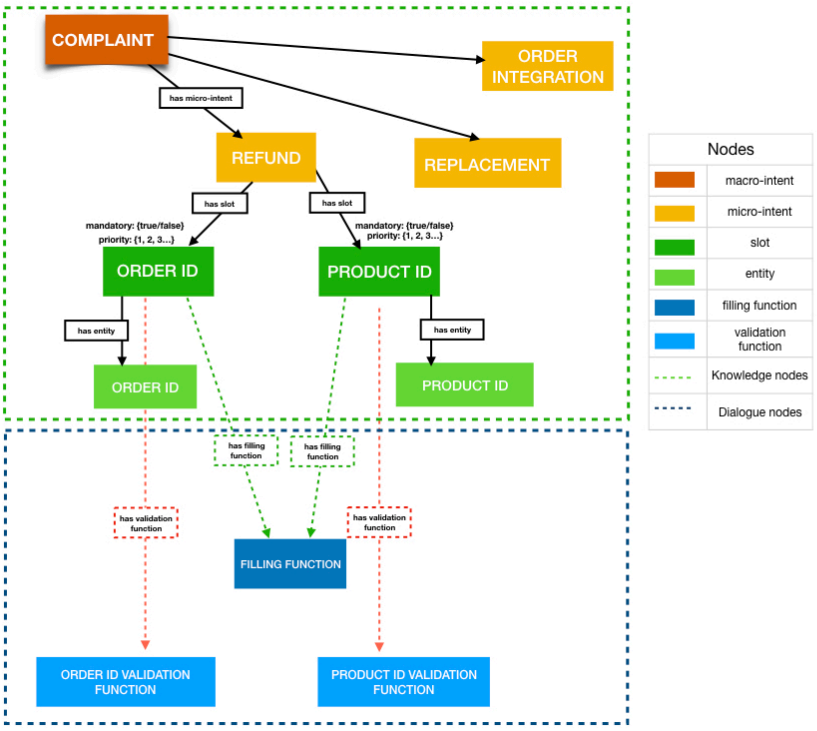

Figure 4
Architecture of the communicative scenario of REFUND, with filling and validation function nodes. Above each slot we can see some of the slot attributes (e.g. priority, mandatory) and their possible values, ruling the slot filling process during dialogue.

the dialogue, during a 'Replacement' we may not have them. In such cases, we have a missing information scenario where the pieces of information needed for the task must be retrieved by the machine. If slot-filling or validation fail for any reason, the system will trigger the most appropriate CR to ground the information it needs to go ahead with its task.

5. Grounding through clarification requests

In this section, we will argue that clarification is needed for a task-oriented DS to be sure to achieve the communicative goal and succeed in the task completion. Moreover, a more fluent and natural dialogue flow can be reached relying on a graph based ontology where different types of CRs are stored.

5.1 Grounding theory

In human-human conversation participants are constantly monitoring each other to seek evidences whether the last utterance has been correctly understood. If the exchange succeeds, the information is grounded and becomes part of shared information of the speakers; if it is not, then clarifying previous information is required to fully achieve

their informational goals (Ginzburg 1998). De Boni and Manandhar (2005) refer to such exchanges as clarification dialogues, âĂIJas the questions that constitute them either clarify previous questions or answers, or clarify the mental picture the user is trying to build by elaborating on previously asked or given informationâĂİ. Communication, indeed, is a joint activity in which âĂIJtwo speakers must assume a vast amount of shared information or common ground, that is mutual knowledge, mutual beliefs, and mutual assumptionsâĂİ (Clark and Brennan 1991). To coordinate on this process, speakers need to update their common ground from time to time.

Also in human-machine interaction, pieces of information provided in the previous exchanges need to be recovered or stabilised in order to build the so-called Common Ground (CG). In this paper we will refer to the four main types of Common Ground acknowledged by Clark (2015): personal, local, communal and specialised.

Personal common ground is "a record of the common ground accumulated over time and over repeated exchanges" with an individual (Clark 2015, p. 329). We could compare it with a customer's history kept in a DB, a sort of "diary kept in relation to that person", though this may regard the customers' commercial history and not always their dialogues with a machine. A part of personal common ground is local common ground that is tied to the information obtained from a single exchange with an unknown or known interlocutor. According to Clark (2015), information of this type can be the opening hours for a shop, what time a train leaves, and so on, i.e. typical questions managed by a task-oriented chatbot. In our infrastructure, Personal Common Ground is stored in the Custom Dialogue Management (see Figure 2).

Communal common ground refers to information shared with people that belong to an entire community and it includes general knowledge, knowledge about the same social background, and so on. Within these communities, restricted groups of people sharing areas of expertise in some domain of knowledge (medicine, law, commerce, and so on), where specialised vocabulary is extremely close to that specific domain, can be found. Clark refers to this type as specialised common ground. In our infrastructure, this type of CG is stored in Dialogue-oriented KB (see Figure 2).

All collective actions are built on common ground and its accumulation through a grounding process: an important part of natural communication, as well as of human-machine conversation. During communication, human speakers collaborate to make the message clear and store it correctly, to reach the so-called Grounding criterion:

> âĂIJThe contributor and the partners mutually believe that the partners have understood what the contributor meant to a criterion sufficient for current purposesâĂİ (Clark and Schaefer 1989).

In order to consider the communication to be successful, speakers must assume that the grounding criterion has been entirely satisfied. To do so, contributing can be divided into two phases: the presentation phase and the acceptance phase (Clark and Schaefer 1989). The latter is where difficulties occur since it is generally initiated by the interlocutor indicating his state of understanding as successful or troubling. In the first case, there is no misunderstanding in the comprehension of the utterances, that can be expressed by different types of understanding evidence. In the second case, when inconsistencies occur during communication, the interlocutor should initiate the acceptance phase by giving evidence of that problem.

In human-machine interaction, providing evidence for grounding information can be very difficult and can require a major effort from the user. This is what Brennan (1998) calls the grounding problem in human-computer interaction, principally due to

inadequate feedback and impoverished context. While in human-human conversation, feedback is naturally provided by the speakers, during an interaction with a dialogue system, the success of the communicative goal falls on the human speaker responsibility, since the machine could encounter difficulties in presenting feedback that is adequate to the communicative situation.

Feedback, indeed, is the type of evidence we are looking for during communication and it can be presented as negative or positive according to our partner's level of understanding. A milestone in this field is Allwood et al. (1992) who propose a classification in four basic communicative functions: contact, perception, understanding, and attitudinal reaction[7].

5.2 Clarification Requests

As Gabsdil (2003) puts it, in human-human communication, when speakers do not (fully) understand or are uncertain about previous information provided by their partner, a clarification request (CR) is initiated. CRs are a fundamental part of the clarification dialogue, they represent a pragmatic tool that plays a major role in the grounding process and through which the interlocutors can maintain mutual understanding of the communicated message.

Clarifications are usually uttered in the context of miscommunication. Following Hirst et al. (1994), miscommunication can be partitioned into three different types: misunderstanding, non-understanding, and misconception. The type of CRs we deal with in this paper falls under the "non-understanding" category since "misunderstanding" typically leads to correction rather than clarification[8] and "misconception" leads to a CR characterised by a surprised intonation due to misalignment between the hearer and speaker's belief about the world. Regarding the non-understanding category, Gabsdil (2003) suggests three different possible levels of interpretation: no interpretation at all, uncertain interpretation, and ambiguous interpretation. Since in our case study, the machine always interprets something, what it should be able to manage is the uncertain and/or the ambiguous interpretation; that means that even if it finds more than one possible interpretation of an utterance, the system should handle the conversation, posing the most correct CR to disambiguate and ground information correctly.

For humans, managing the grounding process is not a big deal since they can recover the implied information through simple feedback or back-channels producing the most effective CR to achieve mutual understanding. Dialogue systems, on the other hand, present different problems when they have to deal with imperfect (speech) recognition. To solve this problem, one strategy is cautious grounding, which requires the user to explicitly confirm all information provided to the system. The problem with cautious grounding is that it leads to very unnatural and inefficient dialogues. In extreme cases, systems pose much more generic clarification questions than task-related questions (Gabsdil 2003). In their study on CRs in Dialogue Systems, Stoyanchev

7 Feedback, according to Allwood (1992), is a linguistic mechanism that allows the exchange of information between two or more participants about four basic communicative functions which are: 1) contact (i.e., whether the interlocutor is willing and able to continue the interaction); 2) perception (i.e., whether the interlocutor is willing and able to perceive the message); 3) understanding i.e., whether the interlocutor is willing and able to understand the message; 4) attitudinal reactions (i.e. whether the interlocutor is willing and able to react and (adequately) respond to the message, specifically whether he/she accepts or rejects it).

8 In our case study misunderstanding could occur when new information given by the user clashes with previously given information.

et al. (2014) argue that nowadays Dialogue Systems tend to use generic clarification questions to clarify unclear sentences, i.e. requests for repetition or rephrasing of the previous utterance instead of more functional target clarification questions, i.e. requests that signal and report just the unclear portion of the utterance. The forms and functions in which CRs occur have been investigated by many scholars, one of them is Purver (2004) who, starting from Ginzburg and CooperâĂŹs investigation (2001), classifies the CRs according to their form and reading, naming generic questions as non-reprise questions and target questions as types of reprise questions. A finer-grained and multi-dimensional classification has been provided by Rodriguez and Schlangen (2004) that has been taken into consideration in the extended classification proposed by Di Maro et al. (2021). Di Maro et al. start from Allwood and colleagues (1992) and define four basic communicative functions: contact, perception, understanding, and intention. For each stage, their classification defines the problem that could have led to a misunderstanding, the phenomenon that could have triggered it, the distinction between CR form and function, and, at last, the compromised item during the conversation. Our work focuses on some of the issues that in the mentioned framework the authors define as pertaining to the communication levels of intention and understanding. In the next paragraph, we will go deeper into the CR issue and propose a more detailed classification, particularly suited for task-oriented dialogue systems, showing how data and relations stored in our graph KB can support the selection of the most correct CR type for each investigated pragmatic scenario.

6. CRs and pragmatic scenarios

As seen in the previous paragraphs, we designed a KB capable of storing information about the conceptual and linguistic domain of a task-oriented chatbot (see among others (Hussain, Ameri Sianaki, and Ababneh 2019)) as well as information regarding the actions the machine could or should do to manage the Common Ground and the dialogue, achieving the tasks related to a specific intent. In this section, we will present some cases of miscommunication in different pragmatic scenarios (see among others (McShane 2017)) which could be solved by the machine by relying on our graph KB model, and in particular, the properties and relations among the different nodes we stored in it (see Section 4). As already mentioned, the main aim of our infrastructure is to make it possible to avoid as much as possible the use of generic clarification questions (Stoyanchev, Liu, and Hirschberg 2014) in dialogue managing when, for example, an NLU model does not achieve a sufficient score that guarantees it has correctly understood the human input or it needs to acquire new information to complete its task. In order to reach this goal, we further enriched the classification proposed by Di Maro et. al (2021), identifying different types of CRs for different pragmatic scenarios that a task-oriented DS may need to handle when dealing with intents, slots, and entities. As summarised in Table 2, we identified some of the main pragmatic issues such as intent and entity disambiguation, slot-filling, and entity grounding. In the second column we show which resources of our infrastructure should be explored to retrieve the most appropriate CR among the types we identified in the third column (i.e. positive polar questions, alternative questions, high negative polar questions, wh-questions, non-reprise questions), and we suggest if the requests must be posed in a precise order or they can be used as alternatives, depending on the dialogue strategy the DS's developers want to set out. Finally, in the fourth column, we present the functions each CR type fulfils for each pragmatic issue.

In addition to nodes and their properties, we conceived three kinds of relations among the intents, on the basis of their status as "parents" or "peers"[9] and of their capability to be "preparatory" for another intent:

1. **parent** relations, labelled as HAS_MICRO_INTENT, connect MICRO_INTENTs to MACRO_INTENTs in a hierarchic structure (MICRO_INTENTs sharing part of their formal properties with the related MACRO_INTENT);
2. **peer** relations, labelled as IS_PEER, are instantiated among intents that are near from a semantic point of view but differ in their pragmatic functions. Peer relations connect intents that are not "relatives" (i.e. which do not belong to the same frame) but are mainly used to connect very similar intents (based e.g. on the same entity, or intent object) that nonetheless differ from an "operational" point of view (e.g. operational vs. informational tasks).
3. **preparatory** relations, labelled as IS_PREPARATORY_TO, signal that an intent may be useful to trigger pieces of information needed for the slot-filling of another intent when such data are not at the userâĂŹs disposal.

In the following paragraphs, we will better explain how the table is meant to be used, by exemplifying it through real communicative situations.

Table 2
Schema of CR types. For each pragmatic issue, the table shows the related source to be explored in the KB, the CRs type, and the function they perform. CR types in the same cell are intended as consequential, while the ones in separated cells are meant as alternatives.

Pragmatic Issues	KB source	CR types	Function
Intent Disambiguation	Parent relation	Positive Polar question Alternative question	Confirmation Selection
	Peer relation	Alternative question	Disambiguation
		Polar question	Confirmation
Entity Disambiguation	Dictionary relation	Alternative question	Disambiguation
	Validation function	Non-reprise question	Understanding
Entity Grounding	Dialogue state tracking and/or external DB	High Negative Polar question	Conflict resolution
Slot-filling	Filling function	Alternative question	Slot-filling
		Wh-Question	
		Positive Polar question	
		Explicit request	
Entity Retrieval	Intent preparatory relation	(Positive Polar question) Wh-Question	Selection Information seeking

6.1 Common Ground and Clarification Requests

CRs are needed when the communicative intention of the human speaker recognised by the machine needs a double-check before being stored in the Common Ground of the latter, or when the received information clashes with what the machine has already

9 The concepts of parent and peer relations have already been used in ontology research (s., among other, Ram and Park (2004)) as well as in the context of Construction Grammar (s. Norde (2014) and Russo (2016)).

stored in the so-called Personal Common Ground (Clark 2015)[10]. As to the first point, we identify two levels of disambiguation concerning, respectively, intents and entities.

6.1.1 Intent Disambiguation Requests

One of the main problems with intent recognition is what to do when the machine is not sure about the users' intention. As already mentioned, some systems rely on generic CRs while others adopt a cautious grounding strategy. In some cases, it could be possible to disambiguate on the basis of the customer's history while in other cases CRs may be instantiated on the basis of the confidence score returned by a machine learning module. In many other cases, though, such data-driven strategies might not be sufficient because the customer has never interacted with the chatbot before or the confidence score is very similar among more than two intents. In such cases, we propose to refer to the relations we instantiated in the graph KB among the intents, to find the right question to ask the users. Using a cognitive model for the representation of the communicative scenarios, as seen in Section 4, we conceived and designed them as frames, i.e. as schemes or constructions, each with its configuration, in terms of properties and relations toward other node types. As to the intent-intent relationship, parent relations can be explored in cases of uncertain or ambiguous interpretation (Gabsdil 2003) to determine at least the macro-intent of the human user. The machine could thus ask for confirmation through a positive polar question to establish the common ground and then make an alternative question, e.g. proposing the set of micro-intents related to the macro-intent. As to the frame 'Complaint' exemplified above (see Fig. 4), when the machine is not sure if the user wants a refund or a replacement of a bought article (and it has no other means to decide which is the right intent) it could ask for confirmation of the main intention: complaining about an order, and then request the user to select the right micro-intent.

On the other side, peer relations can be instantiated between (macro- or micro-) intents that may focus on the same entities, though not always filling the same slot. In our customer support domain, this strategy was applied to intents having the same object but triggering an operational *vs.* informational task. In the 'Refund' example cited above, our data show that people may ask for a refund of an order they already made (so the machine has to handle an operational task) or simply ask, before instantiating an order, if the shop has a refund policy or how it works. Depending on the adopted NLU technique this problem may have more or less impact on the dialogue flow (e.g. it could be especially relevant when frames are retrieved by means of their slot or entities since the operational and the informational intent do treat the same object). In our case study, we built a fine-tuned transformer-based model which reaches an accuracy of 0.9 (on a model trained on 54 intents) but, still, not all the requests by the users are simply to classify. In some cases, indeed, there is no contextual linguistic evidence that leads the machine (or the human annotator) to the right interpretation of the utterance, e.g. when the speaker utters something like "Hi, would it be possible to request a refund?". So, having a peer relation stored in the KB, the machine can ask for confirmation by formulating either an Alternative Question (e.g."do you want a refund of an order you made or simply want information about our refund policy?", i.e.: "are you asking if it's possible or are you asking me to do that?") or a Polar Question if, e.g., the dialogue flow has been set to automatically lean toward one of the peer intents (e.g. "so you

10 As already mentioned, in our prototype (see Figure 2) the so-called Personal Common Ground is stored in the Custom Dialogue Management Module which gets updated at each dialogue exchange, while the Communal Common Ground is stored in graph KB itself.

want to ask for a refund, don't you?"). In this example, the micro-intent performing an operational task pertains to the macro-intent 'Complaint' and has some slots to fill, while the one outputting an informational activity is connected to the macro-intent 'Generic info' of the frame FAQ, which has no slots to fill, that is, they instantiate two different actions and, thus, two different dialogue flows.

6.1.2 Entity Disambiguation Requests

Disambiguation might be needed when dealing with entities too. Scholars and developers have faced this problem from different perspectives, finding multiple solutions to the entity detection task (Shen, Wang, and Han 2015). We will not go into details here, since the aim of this Section is to show how our dialogue-oriented KB may enhance disambiguation by using some target CRs in those cases which fall under the issues of entity linking and entity validation.

In the first case, we have one signifier (or entity) related to two (or more) different signifieds (or entity types) mapped through a unique relation. How the relations are mapped in the KB depends on the single application - and we know that a graph KB can be easily used to store and manage linked open data too (4.1). The point here is that the KB is designed to overcome uncertain or misleading entity interpretation by triggering the right disambiguation request based on the involved entities. In some domains, such as the one we are dealing with, i.e. a commercial industrial domain[11], entity linking may not be so easy[12]. That is the case, for example, of entities representing the same object (thus the same signifier for the same signified) which may nonetheless fall under different categories depending on the adopted taxonomy. In the so-called ETIM classification[13] a "circuit breaker" is classified as "class" (i.e. as product type) as well as "feature" of a different class (i.e. as product attribute), so in a sentence like "I'd like to buy a circuit breaker" the machine should tag "circuit breaker" as product type, while in an utterance like "I'm looking for a CEE socket with circuit breaker" the latter should be tagged as product feature. In an intent like 'Product search', filling the right slot (i.e. product type *vs.* feature) may improve the search engine capabilities and let it return the right output values. In case of uncertainty, the machine could either instantiate a different (more general) query on the target DB or produce an alternative question, either asking the customer if he/she is looking for a product (i.e. circuit breaker) or the other item tagged as a product (i.e. CEE socket) containing a circuit breaker (as a feature) or making a multiple-choice question (this choice depending on the technique used for question/language generation).

As to the entity validation, CRs might be necessary when one word gets tagged as an entity (in our case using a machine learning algorithm) but then the form can not be found in the corresponding source we use to validate it (e.g. order or product IDs, brands, and so on). In such cases the machine could use a more generic CR, asking for rephrasing the single item it was not able to validate or it may instantiate a Wh-Question, asking the user to choose from among a list of items retrieved from the KB. As seen above, validation strategies are also stored in our KB and are mapped N-N to

11 On the same domain see also Russo et al. (2019).
12 When dealing with named entities, entity linking is mainly based on linked open data and, in particular, on Wikipedia.
13 ETIM is an international classification standard for technical products used in Europe. See https://www.etim-international.com/.

each slot, referring to the entity the slot is related to. In such nodes, we store information on which CR type should be selected for that particular validation[14].

6.1.3 Entity grounding requests

An interesting case that is more frequent in human-human than in human-machine communication is the one concerning the grounding process when information stored in the personal common ground of one participant clashes with the one evoked by the other. That can occur, in a commercial domain like the one we are focusing on, when the task is to edit an order by changing some data (the number of bought items, the shipping address, etc.) or, in general, where an already given piece of information must be updated with a new one. The original piece of information has already been stored by the machine but the one brought by the user might be different. Let's analyse the following example from the intent 'Order update'. The human user asks: "Hi, I ordered 5 led lights but I need to buy 7 of them". In such a case, the machine should be able to fill two different slots, i.e. "original quantity" and "updated quantity", with the same entity type "quantity". When dealing with task-oriented DS in very narrow domains (such as "booking a flight"), such issues may be solved with a rule-based approach. In our case, instead, the best solution would be to fill the "original quantity" slot by checking information on the Personal Common Ground, i.e. by comparing the two quantity values recognised by our algorithm with the data already stored in the DB for the corresponding order, and fill the "updated quantity" slot with the other quantity found in the utterance (possibly asking for confirmation). The need for a grounding question arises when the information given by the user clashes with the one stored in the Common Ground, as in the present example where the original quantity of the order was 4 and not 5, as told by the human user.

Such a situation may be compared to what Domaneschi et al. (2017) call "original speaker belief" where the bias of the machine is based on an already given piece of information that is not confirmed by the newly-acquired bias arising from contextual evidence, leading to the use of a so-called High Negative Polar Question, e.g. "Wasn't it 5?"[15].

6.2 Missing Information and Clarification Requests

CRs are also produced to recover information not fully provided by the user or not validated by the machine.

6.2.1 Slot-filling Clarification Requests

In task-oriented chatbots, CRs may be necessary for the correct slot-filling. They can be presented in form of:

1. **A**lternative or Wh-Questions when the user must choose between two or multiple choices to instantiate an action (e.g. in intents involving a search that can be defined with different filters, i.e. by more than one item, like

14 Our infrastructure provides three possible validation types: one that verifies if the retrieved entity exists on the DB that has to be queried; another that makes a formal control on a specific data type; another based on dictionaries, which can be stored in the knowledge graph or in an external source.

15 For Italian we refer to Di Maro et al. (2021b) who, starting from Domaneschi's data, propose an extended version of their experiments whose results exhibit that also in Italian, in such pragmatic conditions, the preferred CR form is High Negative Polar Question, mainly expressed in the past tense.

Order Search in our case study, where users can search by date, product, or other criteria);

2. **P**olar Question when the system is not sure that the piece of information to be retrieved is at the user's disposal but this would be useful to avoid other cascade questions (e.g. when a slot has a priority over other slots of the same intent and a positive response would make other slots not necessary anymore);

3. **E**xplicit requests when the required information surely is at the user's disposal (e.g. in a complaint of an order, where the user must have the order ID).

6.2.2 Entity retrieval Requests

A particular task our graph KB was designed to support is the ability to infer or retrieve information when a slot can not be filled because the human speaker does not have such a piece of information. Indeed, in cases like the request for product availability, the slot to be filled would be the product ID. But what is to be done when the user does not know it? The machine could rely on the graph KB, by exploring what we call a "preparatory relation" and either ask the user if he/she first wants to find the right product ID through a polar question (instantiating a product search) or autonomously look for it on the basis of the given information and then ask the user to select or confirm the desired product ID through a Wh-Question. In such cases, our decision to conceive entities and slots separately proves to be effective since the machine could use the entities found in the original utterance to fill the slots of the preparatory intent[16]. In a sentence like "I'd like to know if you have 10 Daikin air conditioners available", our intent recognition model would recognise the intent 'Availability' and the machine would ask the speaker for the product ID, since this would be the necessary slot to go ahead and instantiate the requested action[17]. When the machine understands that the required piece of information is not at the speaker's disposal, it can refer to the preparatory intent 'Product search' (whose relation with the 'product ID' slot of the 'Availability' intent had been previously stored in the KB) where the lexemes "Daikin" and "air conditioners", tagged as "brand" and "product type" entities respectively, will now fill the corresponding slots which are necessary for the task of product search. It could thus display the results of such a query and then ask the user to select the correct product ID (typically with a Wh-Question) to go ahead with the original task completion, i.e. giving information about the product availability.

7. Conclusion

In this work, we presented an approach to the construction of a knowledge base supporting the dialogue from different points of view. Starting from the problems arisen in the literature on human-human communication, we contextualised them in the framework of human-machine communication, focusing on the communicative

16 Many studies, indeed, talk about slot-filling but what they actually do is slot labelling (s. Weld et al. (2021)). We believe, instead, that the difference between entities and slots should be maintained for the convenience of the ontology and its querability.

17 This could be done with a simple question like "can you give me the product ID?" or a polar question like "do you know the product ID?".

function of understanding and, in particular, on the CRs that should be triggered during the information processing.

We assumed that the fine-grained design of the relations between all of the elements in our architecture can foster a correct interpretation of different intents that refer to the same objects in the external world but are nonetheless different from a pragmatic and operational point of view.

Based on both theoretical and practical points of view, the obtained frames, intents, slots, entities, and related structures have been mapped into a graph database structure stored in Neo4j, capable of supporting the dialogue management. Such information and its representation constitute the grounding architecture of the KB.

We then presented some cases of miscommunication that could occur during the dialogue, showing how our graph KB model should help solve them by navigating its relations and reading its properties, thus enabling the DM module to signal the problem and repair it properly with the most appropriate clarification request. Indeed, our infrastructure is conceived to avoid as much as possible the use of non-reprise questions in dialogue managing (Ginzburg and Cooper 2001) when situations of uncertainty occur and it is necessary to acquire new information to achieve the task. Given that, we propose to use different kinds of target CRs, depending on the pragmatic issue to be faced, and show some possible strategies to be applied when information is incomplete or when it is necessary to build or update the common ground. In Table 2, CRs with respective functions and associated issues are shown. Furthermore, the presented structure is flexible, scalable, and domain-independent, since the identified nodes and relations represent high-level concepts (intents, slots, entities, texts, etc.) that can be used through different domains. Moreover, our KB also contains dialogic information in the form of relations, properties, and nodes like filling functions, validation functions, and actions. These are used to define node-level custom behaviour to be instantiated when a slot is (or is not) filled or it needs to be validated or when a CR type must be triggered to go ahead with the communicative exchange. The integration of what we called ontological nodes and dialogic nodes allows us to refer to our KB as dialogue-oriented. The defined structure could be easily adapted and implemented to support any kind of dialogue-enabled application. Furthermore, the KB allows the developers to manage most of the dialogue configuration in the graph and this results in a flexible system that has no need to be rewritten in order to change its behaviour but rather generates it by exploring the dialogic nodes. Since domain and dialogue data are stored in the graph, it is also possible to generate data-sets to train machine learning models for entity recognition and for both micro and macro-intents, designing a multi-level classification architecture. Implementations are not limited to data-driven approaches but may also support rule-driven approaches, by adding new properties and using the existing relations. In our case study, we also implemented a Dialogue State Tracking module using a Bayesian inference engine[18] but it could be easily substituted by any other kind of implementation. In conclusion, we have shown a modular system based on the flexibility provided by graph databases to adapt dialog management services to low-level services offered by multiple clients in multiple domains. The discussed graph-based protocol provides a normalised view of the interactions between dialog policies, reference corpora and domain elements. From the industrial research point of view, the graph database provides a tool to easily compare dialog management policies in multiple domains, extract common issues, optimise procedures and abstract

18 OpenDial framework (Lison and Kennington 2016).

new general policies that can be rapidly shared with all clients. Finally, considering the problems arising in human-machine interaction due to inadequate feedback and impoverished context (Brennan 1998), we assume that a dialogue-oriented approach to the construction of KBs supporting the dialogue management plays a pivotal role that needs to be further investigated, moreover because it could easily be adapted to new contexts and domains.

7.1 Future work

At the current state of the work, frames and some of the related structures need to be manually instantiated, requiring a significant effort during the knowledge design phase (also due to the specific domain we have dealt with). In future works, the problem could be faced by investigating possible strategies for the automatic creation of the graph. Domain-related text sources contain a lot of information that could be used to define relations between nodes and could help dialogue management. Moreover, by parsing the aforementioned texts and analysing the dependencies occurring between verbs and nouns, we could be able to automatically define new relations. Keeping in mind our domain, we could identify relations like COMPOSITION, when an element is composed of other ones, or USED_TOGETHER, when two elements are frequently mentioned together, and more. We are already working on that, on the one hand, retrieving such relations from existing taxonomies (domain-specific ones but also open linked data like Wikidata), on the other hand by extracting them from domain-specific texts (such as handbooks, instructions but also Wikipedia texts). Added relations could therefore be used to improve the KB and the dialogue experience. Although tools like Google Dialogflow and Amazon Alexa Conversation offer NLU capabilities in a simple and accessible way, they do not explicitly face Common Ground issues in communication. Given that, by making the infrastructure accessible through web services, it could be easily integrated into the above-mentioned tools (as well as in any other tool for DM) combining their capabilities with our architectural ones, giving the possibility to carry out complex communication exchanges and thus improving the interaction quality.

As Di Maro et al. (2021a) acknowledge, the use of a graph database for the detection of common ground dialogue inconsistencies could be fundamental for the implementation of a dialogue system with argumentation capabilities. Indeed, this study can be framed in the field of formal argumentation, particularly concerning the study of *Argumentation-based dialogue* which is an area that still needs a substantial theoretical framework of reference (Prakken 2018).

Finally, since our theoretical approach needs to be validated, our prototype should be tested in a real environment measuring the quality of the interaction and the success rate in proposing the right CRs. Concerning task-oriented dialogue systems, two main metrics of evaluation methods have been proposed: user satisfaction and user simulation (Deriu et al. 2021). In the first case, the usability of the system can be approximated by the satisfaction of its users, which can be measured by questionnaires (see among others (Lund 2001)). In the second case, the idea is to simulate the behaviour of the users, e.g. by using the simulation as an environment to train a reinforcement-learning based system thus evaluating the system on the basis of the reward achieved by the dialogue manager under the user simulation. Under these two methods, several measures (task-success rate, dialogue efficiency, interaction quality, etc.) and frameworks (see for example PARADISE framework in Walker et al. (1997)) have been proposed in the literature and could be used to evaluate our system.

Acknowledgments
The authors would like to thank Maria Di Maro and Antonio Origlia for their suggestions and for all of the productive reasoning shared together about our AI.CODIUM® project.

References

Ait-Mlouk, Addi and Lili Jiang. 2020. KBot: A Knowledge Graph Based ChatBot for Natural Language Understanding Over Linked Data. *IEEE Access*, PP:1–1, 08.

Aliannejadi, Mohammad, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2021. Building and Evaluating Open-Domain Dialogue Corpora with Clarifying Questions. In *EMNLP*.

Allwood, Jens, Joakim Nivre, and Elisabeth Ahlsén. 1992. On the semantics and pragmatics of linguistic feedback. *Journal of semantics*, 9(1):1–26.

Bao, Qiming, Lin Ni, and Jiamou Liu. 2020. HHH: An Online Medical Chatbot System Based on Knowledge Graph and Hierarchical Bi-Directional Attention. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '20, New York, NY, USA. Association for Computing Machinery.

Brennan, Susan E. 1998. The grounding problem in conversations with and through computers. *Social and cognitive approaches to interpersonal communication*, pages 201–225.

Clark, Evev. 2015. *15 Common Ground*, volume 87. Wiley Online Library.

Clark, Herbert H. and Susan E. Brennan. 1991. Grounding in communication. In Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley, editors, *Perspectives on Socially Shared Cognition*. American Psychological Association, pages 127–149.

Clark, Herbert H. and Edward F. Schaefer. 1989. Contributing to discourse. *Cognitive science*, 13(2):259–294.

De Boni, Marco and Suresh Manandhar. 2005. Implementing clarification dialogues in open domain question answering. *Natural Language Engineering*, 11(4):343–362.

Deriu, Jan, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2021. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54(1):755–810.

Di Maro, Maria, Hendrik Buschmeier, Stefan Kopp, and Francesco Cutugno. 2021. Clarification requests negotiating personal common ground. In *Proceeding of the 4th Experimental Pragmatics in Italy Conference (XPRAG.it 2020)*, Online conference, July.

Di Maro, Maria, Antonio Origlia, and Francesco Cutugno. 2021a. Conflict Search Graph for Common Ground Consistency checks in Dialogue Systems. In *Proceedings of the 25th Workshop on the Semantics and Pragmatics of Dialogue*, University of Potsdam, Potsdam, Germany, September.

Di Maro, Maria, Antonio Origlia, and Francesco Cutugno. 2021b. PolarExpress: Polar question forms expressing bias-evidence conflicts in Italian. *International Journal of linguistics*. forthcoming.

Domaneschi, Filippo, Maribel Romero, and Bettina Braun. 2017. Bias in polar questions: Evidence from English and German production experiments. *Glossa: a journal of general linguistics*, 2(1).

Fillmore, Charles J. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32.

Fillmore, Charles J., Russell Lee-Goldman, and Russell Rhodes. 2012. The framenet constructicon. *Sign-based construction grammar*, (193):309–372.

Gabsdil, Malte. 2003. Clarification in spoken dialogue systems. In *Proceedings of the 2003 AAAI Spring Symposium. Workshop on Natural Language Generation in Spoken and Written Dialogue*, pages 28–35, Stanford University, Palo Alto, California, March.

Ginzburg, Jonathan. 1998. Clarifying utterances. In *Proceedings of the Twente workshop on the formal semantics and pragmatics of dialogues. Universiteit Twente, Faculteit Informatica, Enschede*, pages 11–30, Universiteit Twente, Twente, The Netherlands, May. Citeseer.

Ginzburg, Jonathan and Robin Cooper. 2001. Resolving ellipsis in clarification. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 236–243, Toulouse, France, July.

Grazioso, Marco, Valeria Cera, Maria Di Maro, Antonio Origlia, and Francesco Cutugno. 2018. From linguistic linked open data to multimodal natural interaction: A case study. In *2018 22nd International Conference Information Visualisation (IV)*, pages 315–320, Fisciano, Italy, July. IEEE.

Hirst, Graeme, Susan McRoy, Peter Heeman, Philip Edmonds, and Diane Horton. 1994. Repairing conversational misunderstandings and non-understandings. *Speech communication*, 15(3-4):213–229.

Hussain, Shafquat, Omid Ameri Sianaki, and Nedal Ababneh. 2019. A survey on conversational agents/chatbots classification and design techniques. In Leonard Barolli, Makoto Takizawa, Fatos Xhafa, and Tomoya Enokido, editors, *Web, Artificial Intelligence and Network Applications*, pages 946–956, Cham. Springer International Publishing.

Jannach, Dietmar, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2020. A survey on conversational recommender systems. *arXiv preprint arXiv:2004.00646*.

Jokinen, Kristiina and Michael McTear. 2009. Spoken dialogue systems. *Synthesis Lectures on Human Language Technologies*, 2(1):1–151.

Jurafsky, Daniel and James H. Martin. 2019. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Els autors.

Lison, Pierre and Casey Kennington. 2016. Opendial: A toolkit for developing spoken dialogue systems with probabilistic rules. In *Proceedings of the Annual meeting of the Association for Computational Linguistics (ACL 2016) system demonstrations*, pages 67–72, Berlin, Germany, August.

Lund, Arnold M. 2001. Measuring usability with the use questionnaire12. *Usability interface*, 8(2):3–6.

Mancini, Azzurra and Valentina Russo. 2022. Dalla teoria delle lingue e dei linguaggi alla costruzione di applicazioni IA per l'interfaccia uomo-macchina. In *La trasformazione digitale e le sue competenze per la Network Society. Contesti, saperi e professioni emergenti nelle Scienze Umane e Sociali*. Franco Angeli.

McShane, Marjorie. 2017. Natural language understanding (NLU, not NLP) in cognitive systems. *AI Magazine*, 38(4):43–56.

Miliani, Martina, Lucia C. Passaro, and Alessandro Lenci. 2019. Text Frame Detector: Slot Filling Based On Domain Knowledge Bases. In *Proceedings of the Sixth Italian Conference on Computational Linguistics (CLiC-it 2019)*, Bari, Italy, November.

Nakano, Mikio and Kazunori Komatani. 2020. A framework for building closed-domain chat dialogue systems. *Knowledge-Based Systems*, 204:106212.

Norde, Muriel. 2014. On parents and peers in constructional networks. Coglingdays 2014.

Origlia, Antonio, Francesco Cutugno, Antonio Rodà, Piero Cosi, and Claudio Zmarich. 2019. FANTASIA: a framework for advanced natural tools and applications in social, interactive approaches. *Multimedia Tools and Applications*, 78(10):13613–13648.

Prakken, Henry. 2018. *Historical overview of formal argumentation*, volume 1. College Publications.

Purver, Matthew and Richard John. 2004. *The theory and use of clarification requests in dialogue*. Ph.D. thesis, University of London.

Ram, Sudha and Jinsoo Park. 2004. Semantic Conflict Resolution Ontology (SCROL): An ontology for detecting and resolving data and schema-level semantic conflicts. *IEEE Transactions on Knowledge and Data engineering*, 16(2):189–202.

Roccabruna, Gabriel, Alessandra Cervone, and Giuseppe Riccardi. 2020. Multifunctional ISO standard Dialogue Act tagging in Italian. In *Proceedings of the Seventh Italian Conference on Computational Linguistics (CLIC-IT 2020)*, Bologna, Italy, March.

Rodríguez, Kepa Joseba and David Schlangen. 2004. Form, intonation and function of clarification requests in German task-oriented spoken dialogues. In *Proceedings of Catalog (the 8th workshop on the semantics and pragmatics of dialogue; SemDial04)*, Barcelona, Catalonia, July.

Russo, Valentina. 2016. Expressing large quantities in a rhetorical way. The epistemic large-quantity construction in Italian: [non so quant_ + N/V] (I dont't know how many/much + N/V). Societas Linguistica Europaea.

Russo, Valentina, Azzurra Mancini, and Gaetano Giancaspro. 2019. Linguistica computazionale e intelligenze artificiali, un sogno (ancora lontanto) che muove l'economia. In E.M. Piccirilli and V. Russo, editors, *Linguistica ed Economia. Un connubio tra due discipline come ricerca filosofica nell'economia degli scambi*, volume 2. Academy School, Naples, Italy, pages 161–176.

Shen, Wei, Jianyong Wang, and Jiawei Han. 2015. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Stoyanchev, Svetlana, Alex Liu, and Julia Hirschberg. 2014. Towards natural clarification questions in dialogue systems. In *AISB symposium on questions, discourse and dialogue*, volume 20.

Walker, Marilyn A., Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98/EACL '98, pages 271–280, USA, July. Association for Computational Linguistics.

Weld, Henry, Xiaoqi Huang, Siqu Long, Josiah Poon, and Soyeon Caren Han. 2021. A survey of joint intent detection and slot-filling models in natural language understanding. *arXiv preprint arXiv:2101.08091*.

Zhao, Xinyan, Liangwei Chen, and Huanhuan Chen. 2021. A weighted heterogeneous graph-based dialog system. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–6.