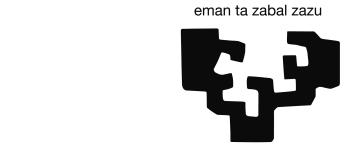
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Generic semantics-based task-oriented dialogue system framework for human-machine interaction in industrial scenarios

Cristina Aceta Moreno

Supervised by:
Aitor Soroa Etxabe
Izaskun Fernández González

2022



Generic semantics-based task-oriented dialogue system framework for human-machine interaction in industrial scenarios

Cristina Aceta Moreno

Supervised by:
Aitor Soroa Etxabe
Izaskun Fernández González

A dissertation submitted to the Department of Computation Sciences and Artificial Intelligence of the University of the Basque Country (UPV/EHU) for the degree of Doctor of Philosophy (Ph.D.) with Industrial Mention in Language Analysis and Processing

Donostia-San Sebastián, June 2022

Mi leal traidora inspiración,
de intermitente aparición
como un ángel hallado en un ascensor,
qué bien funcionas como recuerdo.

El poeta Halley, Love of Lesbian

***

Como un día dijo el Poeta Halley:
"Si las palabras se atraen
que se unan entre ellas,
y a brillar, ¡que son dos sílabas!"

Palabrera, Santi Balmes
(*Canción de Bruma*, 2017)

Acknowledgements

Quién iba a decirme que por fin llegaría el momento de escribir esta parte. Esta tesis ha sido un camino largo y duro, y nunca habría llegado hasta aquí sin aquellas personas que me han acompañado durante estos años. Aviso que me voy a enrollar como las persianas, para variar.

En primer lugar, me gustaría dar las gracias a mis directores. A Izaskun, por ser la persona que me introdujo en el apasionante mundo de la semántica y confiar en mí cuando era un pollito recién salido del máster. Y, por supuesto, por "rizar el rizo" constantemente para que esta tesis saliera adelante de la mejor forma posible. A Aitor, por recordarme que el procesamiento del lenguaje natural es una disciplina maravillosa a la par que compleja, pero aun así guiarme de la mejor forma posible a lo largo de esta tesis. A ambos, por las largas jornadas de revisiones, y por la manía que le habéis cogido a los "in order to", "regarding" y demás ;-).

Gracias también a Tekniker por darme la oportunidad de desarrollar mi tesis y poder aprovechar al máximo lo que ello implica. Pero no todo es trabajo. También quería agradecer a Álvaro, Iñaki, Egoitz (el próximo doctor del grupo), Susana, Edu y Kerman esos cafés que me han servido para evadirme, reírme, enfadarme, indignarme y, en fin, sobrellevar esta experiencia. También gracias a mis compañeros de Sistemas de Información Inteligentes por todos los momentos que hemos compartido a lo largo de estos años. Especial mención a Iker, que ha sido casi el tercer director de esta tesis, por todas sus enseñanzas y su paciencia. También a Laki, por los momentos de Lakipedia y por chincharnos constantemente, algo que hace que el día a día sea mucho más llevadero sin duda. Y porque sé que en el fondo me quiere un poquito. Y, en general, gracias a todos los compañeros con los que, entre pasillos, hemos compartido

experiencias, conversaciones y mensajes de ánimo.

Esta tesis tampoco habría podido salir adelante sin el apoyo de mis amigos, que han estado al pie del cañón en todos los pasos que he dado. A mi gran amigo Shu, el Dr. Cao (pronto Dr.Dr. Cao), por ser uno de los grandes pilares del máster y por nuestra bonita amistad. Por todos tus mensajes de cariño y apoyo, y por los momentos de comidas y quedadas que nos quedan. Gracias por ser de esa gente bonita que no se encuentra tan a menudo. A Leyre e Iñaki, por ser un apoyo incondicional en los peores y en los mejores momentos y convertirse en personas clave en mi vida. Por todas las noches de Overcooked y de dibujos con el móvil tan terapéuticas, y por los mendipaseos que por fin nos vamos a poder echar (sí, los he empezado a echar de menos y todo). A Jordina, mi bebebé y compañera de profesión, por todos los mensajes de ánimo, nuestras divertidas conversaciones y por estar siempre ahí desde la distancia. A las integrantes del akelarre, por acogerme tan bien, y por estar presentes en la recta final con grandes momentos, muchas risas y mucho ciclo indoor, que han sido geniales a la par que terapéuticos. A María, la Dra. García, por ser un modelo a seguir y confiar plenamente en mí desde que nos conocimos hace la friolera de 10 años. Por animarme a ser siempre mejor, a esforzarme, y por hacer que me enamorara perdidamente de la filología. Y, cómo no, por nuestras reuniones para ponernos al día, que a partir de ahora serán entre doctoras. Cualquier palabra que pudiera dedicarte se me quedaría corta. Y como me va a dar para otra tesis como siga así, muchas, MUCHAS gracias al resto de amigos que han estado conmigo desde siempre: Ibra, Ariana, Laura, Andrea y Brenda.

También quería agradecer el apoyo de mi familia. A mis padres que, a pesar de no saber muy bien en qué lío me había metido, han estado apoyándome, dándome ánimos y confiando en mí desde el minuto 0. A mis abuelos, por seguir regalándome momentos a su lado. A mis tíos y primos, por darme el último empujón, tan necesario.

No podía terminar este escrito sin darle las gracias a mi compañero de vida, y qué suerte he tenido. Gràcies, Haritz, per ser-hi i estar, simplement. Gràcies per recordar-me dia rere dia que sóc capaç de fer tot el que em proposi, per assecar-me les llàgrimes quan no podia més i per alegrar-te com el que més amb cadascun dels meus triomfs. Per la teva infinita paciència i sempre voler ajudar-me, encara que molts cops no m'ho he merescut. Per ser el meu betatester i pel teu

valuós *input*. Aquesta tesi també és teva. Ara que hem arribat fins el final d'aquesta etapa, tinc moltes ganes de saber què ens depararan la resta de les parades del nostre viatge. Maite zaitut.

Finalmente, me gustaría hacer una especial mención a la persona que más ha dudado de mí, y la que peor me lo ha hecho pasar durante toda esta larga experiencia. Esa persona soy yo. Y sí, es feo dedicarse algo a uno mismo, pero la ocasión lo merece, porque no hay nada más duro que luchar contra uno mismo día tras día y aun así llegar a la meta más o menos entera. Lo he conseguido, y eso ya nadie podrá quitármelo.

Si has llegado hasta aquí, gracias a ti también, que eso quiere decir que tienes mi tesis en tus manos y la estás leyendo.

Summary

In Industry 5.0, in which automatization has an increasingly important role, human workers and their well-being are placed at the centre of the production process. In this context, task-oriented dialogue systems allow workers to delegate simple tasks to industrial assets while working on other, more complex ones. Also, the possibility of naturally interacting with these systems reduces the cognitive demand to use them and triggers acceptation. However, most current solutions do not allow a natural communication, and modern techniques to obtain such systems require large amounts of data to be trained, which is scarce in these scenarios. This causes industrial task-oriented systems to be highly specific, which limits their capacity to be modified or reused in other use cases, which is bound to high development time and costs.

To overcome these challenges, this thesis leverages Semantic Web Technologies and Natural Language Processing (NLP) techniques to develop KIDE4I, a semantics-based task-oriented dialogue system for industrial scenarios that allows a natural communication between human workers and industrial systems. The modules in KIDE4I have been designed from a generic perspective, with the objective of simplifying the process of adaptation to new use cases. In this line, different methodologies and resources have been proposed to promote reuse and encourage the continuous improvement of the dialogue system through new interactions.

Among these resources, the TODO (Task-Oriented Dialogue management Ontology) is the core of KIDE4I. This modular ontology is in charge not only of modelling the domain, but also the dialogue process and the storage of traces. It has been developed by following a well-known methodology, LOT, which is reflected in its high quality.

KIDE4I has been implemented and adapted for four industrial use cases, proving that the adaptation process is not complex and it benefits from the reuse of resources. Three of these have been evaluated through user studies, and the results obtained are reported.

Resumen

En Industria 5.0, en la cual la automatización tiene un papel cada vez más importante, los trabajadores y su bienestar son cruciales en el proceso de producción. En este contexto, los sistemas de diálogo orientados a tareas permiten que los operarios deleguen las tareas más sencillas a los sistemas industriales mientras trabajan en otras más complejas. Además, la posibilidad de interactuar de forma natural con estos sistemas reduce la carga cognitiva para usarlos y genera aceptación por parte de los usuarios. Sin embargo, la mayoría de las soluciones existentes no permiten una comunicación natural, y las técnicas actuales para obtener dichos sistemas necesitan grandes cantidades de datos para ser entrenados, que son escasos en este tipo de escenarios. Esto provoca que los sistemas de diálogo orientados a tareas en el ámbito industrial sean muy específicos, lo que limita su capacidad de ser modificados o reutilizados en otros escenarios, tareas que están ligadas a un gran esfuerzo en términos de tiempo y costes.

Dados estos retos, en esta tesis se combinan Tecnologías de la Web Semántica con técnicas de Procesamiento del Lenguaje Natural para desarrollar KIDE4I, un sistema de diálogo orientado a tareas semántico para entornos industriales que permite una comunicación natural entre humanos y sistemas industriales. Los módulos de KIDE4I están diseñados para ser genéricos para una sencilla adaptación a nuevos casos de uso. En ese sentido, se han propuesto distintas metodologías y recursos para fomentar el reuso y la mejora continua del sistema de diálogo.

Entre estos recursos, TODO es el núcleo de KIDE4I. Esta ontología modular se encarga de modelar el dominio y el proceso de diálogo, además de almacenar las trazas generadas. Se ha desarrollado siguiendo una conocida metodología, LOT, lo que se refleja en su buena calidad.

KIDE4I se ha implementado y adaptado para su uso en cuatro casos de uso industriales, demostrando que el proceso de adaptación para ello no es complejo y se beneficia del uso de recursos. Tres de ellos se han evaluado a través de estudios de usuario, reportándose los resultados obtenidos.

Laburpena

Industria 5.0n, automatizazioak geroz eta paper garrantzitsuagoa du, eta langileak produkzio-prozesuaren erdigunean kokatzen ditu. Testuinguru honetan, atazetara bideratutako elkarrizketa-sistemei esker, langileek ataza errazenak industria-sistemen esku uzten dituzte, konplexuagoak diren beste batzuetan lan egiten duten bitartean. Gainera, sistema hauekin elkarrizketa modu naturalean gauzatzeak langileek behar duten karga kognitiboa murrizten du, sistema hauen onespena ahalbidetuz. Hala ere, elkarrizketa-sistema gehienek ez dute komunikazio naturalik ahalbidetzen, eta sistema horiek lortzeko gaur egungo teknikek datu kopuru handiak behar dituzte, horrelako testuinguruetan urriak direnak. Horren ondorioz, industrian atazetara bideratutako elkarrizketa-sistemak oso espezifikoak dira eta hauek aldatzea edo berrerabiltzea oso zaila da, denbora eta ahalegin handia eskatzen baitute.

Erronka hauek gainditzeko, tesi honetan Web Semantikoaren Teknologiak eta Lengoaia Naturalaren Prozesamenduko teknikak uztartzen dira eta KIDE4I garatu da. Atazetara bideratutako elkarrizketa-sistema semantiko bat da industrian erabiltzeko, gizakien eta industria-sistemen arteko komunikazio naturala ahalbidetzen duena. Sistema osatzen duten moduluak generikoak dira, eta erraz molda daitezke erabilpen kasu batetik bestera. Ildo horretan, hainbat metodologia eta baliabide proposatu dira elkarrizketa-sistemaren berrerabilera eta etengabeko hobekuntza sustatzeko. Baliabide horien artean dago TODO ontologia modularra, LOT metodologia jarraituz garatu dena. Domeinua eta elkarrizketa-prozesua modelatzen ditu, eta sortutako trazak biltegiratu.

KIDE4I industriako lau erabilpen kasutan inplementatu eta egokitu da. Horietako hiruretan ebaluazio sakona burutu da, giza erabiltzaileak erabiliz, eta tesiak lortutako emaitzen berri ematen du. Moldaketa prozesua erraza izan dela dela frogatu da, baita baliabideen erabilerari etekina atera dela ere.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

ASR Automatic Speech Recognition

BCQ Basic Competency Question

CFG Context-Free Grammars

CMC Collaborative Manipulation Corpus

CMMS Computerized Maintenance Management System

CNN Convolutional Neural Networks

CQ Competency Question

CRF Conditional Random Fields

DL Deep Learning

DLog Description Logic

DM Dialogue Manager

FSAs finite-state automata

FSTs finite-state transducers

HuRIC Human Robot Interaction Corpus

IRI Internationalized Resource Identifier

KEE Key Element Extraction

KIDE4I Knowledge-drIven Dialogue framEwork for Industry

LD Linked Data

LL Lifelong Learning

LOV Linked Open Vocabularies

LSTM Long Short-Term Memory

LU lexical unit

MCR Multilingual Central Repository

ML Machine Learning

NAF NLP Annotation Format

NER Named Entity Recognition

NERC Named Entity Recognition and Classification

NLG Natural Language Generation

NLP Natural Language Processing

NLU Natural Language Understanding

ORSD Ontology Requirement Specification Document

OWL Web Ontology Language

PM Predicate Matrix

POS Part-of-Speech

QA Question Answering

RDF Resource Description Language

RDFS RDF Schema

SPARQL SPARQL Protocol and RDF Query Language

SUMO Suggested Upper Merged Ontology

SVM Support Vector Machine

SW Semantic Web

TODO Task-Oriented Dialogue management Ontology

TTS Text-To-Speech

URI Uniform Resource Identifier

W3C World Wide Web Consortium

XFST Xerox Finite State Tool

Chapter 1

Introduction

Recent technological advances in last decades have caused a great revolution in industrial settings. Is that so, that terms such as Industry 4.0 –and even more recently, Industry 5.0–, that define sustainable, advanced and human-centred industrial environments, are now essential in modern industry.

The research area of this thesis is the human-centred component of Industry 5.0, which situates human workers as one of the fundamental pillars in the production process. In this setting, the main function of digitalisation, rather than replacing workers, is to support them, and ensuring their security and, most importantly, their well-being –both physical and mental– when performing their everyday tasks is of utmost importance. So as to put technological advances at the service of humans, and not the other way around (Commission et al., 2021), the use of technologies such as Artificial Intelligence (AI) to simplify human work in the production process is becoming widely extended nowadays.

The result is that industrial environments are becoming more automatised over time, and workers require intuitive and powerful interaction techniques so as to successfully perform their assigned tasks in collaboration with automatisms (Oborski, 2004). In this sense, workers interact with a wide range of systems at a daily basis, such as intelligent information systems or advanced collaborative robots, and it is key to facilitate this interaction to guarantee optimal work conditions. To cover this necessity, Human-Machine Interfaces (HMI) have increasingly evolved in last years with the development of new mobile

techniques and new gadgets such as smartphones, tablets or Augmented Reality (AR) glasses. A big number of solutions have been developed, especially in collaborative robotics, with human-machine interaction capabilities in different degrees, which allow a more intuitive communication with industrial systems, like interaction through gestures or programming by demonstration (Villani, Pini, Leali, & Secchi, 2018). Another example are dialogue systems, which allow workers to interact with industrial systems in a similar way as they would do with their fellows.

Among dialogue systems, task-oriented dialogue systems –as opposed to conversational dialogue systems, which try to emulate regular conversations– are designed so a target system performs specific actions upon a user request and, for this, are specially relevant in industrial contexts. In this sense, task-oriented dialogue systems are powerful technologies that allow workers to work on multiple tasks at once by delegating secondary assignments by communicating with the target system, usually through voice commands. The use of voice instructions to interact with these systems allow workers to use them from a safe distance, if necessary, and in a way that they do not need to interrupt their current tasks, leaving the quality of their work unaffected. Furthermore, enhancing these systems with the capability of interacting with users in natural language release workers from having to learn specific words of commands to use them.

Taking this into account, this thesis is focused on supporting human workers and improve their work conditions when naturally interacting with industrial systems through the use of task-oriented dialogue systems.

1.1 Problem and Motivation

Although the implementation of complex and innovative technologies in industrial scenarios has reduced the physical workload of workers, an increase of the cognitive load to control and manage such technologies has been observed as a counterpart (Madonna, Monica, Anastasi, & Di Nardo, 2019). The possibility of communicating to industrial systems through natural language is highly encouraged, since it reduces workers' mental stress and triggers acceptance (Kildal et al., 2019), as users do not have to memorise specific words or

sequences to interact with them. However, to develop systems that facilitate a natural interaction between humans and industrial systems –being task-oriented dialogue systems the most common– by using current state-of-the art technologies, such as Deep Learning (DL) techniques, is a difficult task, and leads to a series of challenges in their design process. The main issues to develop task-oriented dialogue systems for industrial environments that allow communication in natural language can be summarised in the following points:

1. **Lack of training data**. The main challenge regarding the use of DL to develop task-oriented dialogue systems capable of interacting through natural language is that great amounts of data for training are needed, and currently available data is usually bound to specific domains and is also scarce (Budzianowski et al., 2018), especially in industrial scenarios (Luckow et al., 2016) and languages other than English.

2. **Restricted natural language communication**. Since using modern technologies is conditioned by the lack of data described above, most dialogue solutions designed for industrial settings are highly specific for the application they are intended to. These solutions usually make use of static structures and rigid language and, thus, communication through natural language is quite restricted (Bugmann & Pires, 2005; Veiga, Pires, & Nilsson, 2009).

3. **Difficult-to-adapt solutions**. Due to the high specificity of dialogue systems designed for industrial environments, their capacity to be reused in other scenarios is very limited and usually bound to expert manual work and high development time and costs (Jurafsky & Martin, 2020).

One possible approach to overcome the challenges described above is to make use of Semantic Web Technologies to develop a task-oriented dialogue framework that allows a natural communication between workers and industrial systems. These technologies are a powerful asset that allows to define in detail the domain of application, reduce ambiguity between agents (Antonelli & Bruno, 2017) and to easily model the dialogue process through relationships between individuals and their properties. Moreover, the use of semantics to develop such a dialogue system framework benefits from one of the

main premises of these technologies: reuse, which enables the development of a generic framework that can reused in different use cases.

As a result, and motivated by the remarks above, this thesis presents KIDE4I, a generic semantics-based task-oriented dialogue system framework for industrial scenarios which has as main objective to enable workers to naturally interact with industrial systems. To achieve that, semantic technologies are used as its core component, and its design is generic enough to allow an easy adaptation to different industrial applications –such as collaborative tasks, guidance, information systems, assistance, etc.– and languages without requiring great amounts of training data to be constructed. Furthermore, its architecture is designed so each of its components reuse existing ontologies and resources and technologies from the NLP field, which considerably reduce their adaptation time and effort.

To prove KIDE4I's easy adaptability to different scenarios, it has been initially implemented for a guide robot, with capabilities that are analogous to logistics robots. This initial, basic implementation has been performed with a generic perspective, so as to serve as the starting point for future adaptations. With this initial implementation, KIDE4I has been adapted to three different use cases that are relevant to industrial settings, for the Spanish language: a bin-picking robot, a computerised maintenance management software (CMMS) and an assistant for maintenance procedure execution. The guide, bin-picking and assistant adaptations have been validated and evaluated with three user studies, which are described and reported in this thesis.

1.2 Thesis objectives and contributions

The main objective of this thesis is to develop a system that allows a natural communication between users and industrial systems, characterised by being easily adaptable to new use cases and languages and that is able to learn from new interactions. To achieve this, a generic framework for task-oriented dialogue systems based on semantic technologies is proposed.

To achieve this goal, the following actions are devised:

- To develop a core, easy-to-adapt ontology that provides task-oriented dialogue systems with the necessary means to be capable of naturally interacting with workers (both at understanding and at communication level), that can be easily adapted to different industrial scenarios and languages –reducing adaptation time and costs–, and allows to store and reproduce the dialogue process.
- To develop a generic task-oriented dialogue system that uses the ontology described above as its core to enable a natural interaction between users and industrial systems.
- To obtain an initial implementation of the generic task-oriented dialogue system that sets the foundations for new adaptations, oriented to make the resources developed reusable.
- To validate the proposed generic semantic task-oriented dialogue system through its adaptation to four use cases that are of relevance in industrial scenarios.
- To evaluate three of the use case adaptations through user studies.

Given these objectives, this thesis makes the following contributions:

- The Task-Oriented Dialogue management Ontology (TODO), a core, modular ontology for task-oriented dialogue systems. This contribution is addressed in Chapter 4.
- The Knowledge-drIven Dialogue framEwork for Industry (KIDE4I), a generic semantic-based task-oriented dialogue system that uses TODO as its core and allows a natural interaction between users and industrial systems. This contribution is addressed in Chapter 5.
- A strategy that leverages existing multilingual lexical, semantic and syntactic resources to semi-automatically populate domain ontologies with intent-related information. This contribution is addressed in Section 4.6.
- A strategy to perform domain data augmentation with data from different but similar domains for semi-supervised key element extraction. This contribution is addressed in Section 5.1.

- An evaluation framework for task-oriented dialogue systems in industrial scenarios. This contribution is addressed in Chapter 6.3.

1.3 Thesis structure

So as to achieve the objectives and report the contributions described above, the rest of this thesis is divided in the following chapters:

- Chapter 2: Fundamental Technologies, Tools and Resources. It describes the fundamental technologies that take part in this thesis, focusing in task-oriented dialogue systems, Semantic Web Technologies and Natural Language Processing.
- Chapter 3: Research Framework. It provides state-of-the-art information on task-oriented dialogue systems, their use in industrial environments and the use of semantic technologies in their development.
- Chapter 4: The Task-Oriented Dialogue management Ontology (TODO). This chapter thoroughly describes TODO. In here, the process to develop the ontology is reported in terms of the methodology to do so, and details on its conceptualisation, implementation, evaluation and publication and implementation are provided. Also, the strategy to populate TODO semi-automatically using existing resources is reported.
- Chapter 5: KIDE4I: Generic Semantic Task-Oriented Dialogue System. This chapter describes the main architecture of KIDE4I, thoroughly describing its modules and their function. The initial implementation for the guide/logistics robot is provided for each of the domain-dependent modules.
- Chapter 6: KIDE4I in Use. In this chapter, the adaptation process of KIDE4I in three industry-relevant use cases –a bin-picking robot, a CMMS (Computerized Maintenance Management System) and a maintenance procedure execution assistant– is reported. These adaptations, along with the guide/logistics robot implementation, are validated and evaluated.

- Chapter 7: Conclusions and Further Work. This chapter summarises the main contributions of this thesis and future directions of its achievements are discussed.

1.4 Publications

Part of the work in this thesis has been presented at conferences and published at scientific journals of impact. This section lists all of them, along with the chapters in this thesis their content is related to.

1.4.1 Conference Papers

- Kildal, J., Fernández, I., Lluvia, I., Lázaro, I., Aceta, C., Vidal, N., & Susperregi, L. (2019). Evaluating the UX obtained from a service robot that provides ancillary way-finding support in an industrial environment. In *Advances in Manufacturing Technology XXXIII: Proceedings of the 17th International Conference on Manufacturing Research, Incorporating the 34th National Conference on Manufacturing Research* (pp. 61-66). - **Chapter 5.1 – Key Element Extraction**.

- Aceta, C., Fernández, I., & Soroa, A. (2021). Ontology Population Reusing Resources for Dialogue Intent Detection: Generic and Multilingual Approach. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)* (pp. 10-18). - **Chapter 4.6 – Semi-Automatic Population of Intent-Relevant Information**.

- Aceta, C., Fernández, I., & Soroa, A. (2021). TODO: A Core Ontology for Task-Oriented Dialogue Systems in Industry 4.0. In *Further with Knowledge Graphs* (pp. 1-15). IOS Press. (SEMANTiCS EU Conference 2021) - **Chapter 4 – The Task-Oriented Dialogue management Ontology (TODO)**.

- del Pozo, A., García-Sardiña, L., Serras, M., González-Docasal, A., Torres, M. I., Ruiz, E., Fernández, I., Aceta, C., Konde, E., Aguinaga, D., de la Cruz, M., Altuna, I., Agirre, J., Etxebeste, I. (2021). EKIN: Towards Natural Language Interaction with

Industrial Production Machines. In *Annual Conference of the Spanish Association for Natural Language Processing* (pp. 5-8). - **Chapter 6 – KIDE4I in Use**

- Aceta, C., Casla, P., Fernández, I., & Soroa, A. (2022). KIDE4-Assistant: an Ontology-Driven Dialogue System Adaptation for Assistance in Maintenance Procedures. FOMI'22: Formal Ontologies Meet Industry. (submitted) - **Chapter 6 – KIDE4I in Use**.

1.4.2 Journal Publications

- Aceta, C., Kildal, J., Fernández, I., & Soroa, A. (2021). Towards an optimal design of natural human interaction mechanisms for a service robot with ancillary way-finding capabilities in industrial environments. *Production & Manufacturing Research*, 9(1), 1-32. - **Chapter 5.1 – Key Element Extraction**.

- Aceta, C., Fernández, I., & Soroa, A. (2022). KIDE4I: A Generic Semantics-Based Task-Oriented Dialogue System for Human-Machine Interaction in Industry 5.0. *Applied Sciences*, 12(3), 1192. - **Chapters 5 –KIDE4I: Generic Semantic Task-Oriented Dialogue System– and 6 –KIDE4I in Use**.

Chapter 2

Fundamental Technologies, Tools and Resources

This chapter aims to provide basic notions regarding the core technologies of the research performed in the context of this thesis: on the one hand, **task-oriented dialogue systems** –more precisely, the pipeline-based ones– and, on the other hand, **Semantic Web Technologies**.

Furthermore, specific parts of this thesis make use of existing tools and resources in the **Natural Language Processing (NLP)** field. This chapter also reports them and their characteristics.

2.1 Dialogue Systems

Dialogue systems are pieces of software designed to have the capacity of interacting with human users through natural language (such as text or speech) or other natural modes of communication (such as gestures).

The literature distinguishes between two types of dialogue systems: task-oriented and conversational. On the one hand, task-oriented dialogue systems are expected to trigger an action to be performed according to a user request, such as booking a hotel, making a reservation at a restaurant or making phone calls. Modern examples of task-oriented dialogue systems that can be found in ev-

eryday settings are virtual assistants, such as Alexa or Siri[1]. On the other hand, conversational dialogue systems do not have a specific goal associated to a command but to keep a conversation with the user about one or different topics.

In this thesis, the focus will be put on task-oriented dialogue systems, which will be described in the following lines.

2.1.1 Task-Oriented Dialogue Systems

As noted above, the main goal of task-oriented dialogue systems is that, from a user request, a specific action that supports the user in carrying out some task is performed.

Traditionally, task-oriented dialogue systems follow a pipeline-based architecture, in which a series of independent modules intervene in the dialogue process by performing specific actions given a user input, from its utterance to the response generated by the system. This architecture is opposed to the one for end-to-end task-oriented dialogue systems (Yang, Li, & Quan, 2020), which encapsulate all the input-output process in a single step.

Come as it may, according to Jurafsky and Martin (2020), most task-oriented dialogue systems nowadays are based on frame structures, in which the action to perform by the system requires a set of pieces of information to be supplied by the user in its input. If any information is missing, the dialogue system will engage with the user to obtain all the necessary information for the requested task to be performed.

Considering this, this section will, first, provide an overview of the foundations for frame-based dialogue and, finally, a basic description of the typical modules in the architecture of pipeline-based task-oriented dialogue systems, which are the type of task-oriented dialogue systems that this thesis will deal with.

[1]These also have conversational capabilities, but in here the focus is put in them as virtual assistants to perform actions such as setting an alarm or sending a message.

2.1.1.1 Foundations of Frame-Based Dialogue: the GUS Architecture

Nowadays, a wide range of modern commercial task-oriented dialogue systems are to some extent inspired in the Genial Understander System (GUS) architecture in Bobrow et al. (1977), which is a task-oriented dialogue system framework developed for the travelling domain (Jurafsky & Martin, 2020). This architecture conceives task-oriented interactions as a slot-filling process based on *frame* structures. These structures model the user intentions when directing a command to the target system –the *intents*– and the elements that take part in them in a slot form, to be filled with information provided by the user. For example, for a traditional assistant for travel booking, '*book a plane*', would be an intent, and some of its slots would be the *origin city*, the *destination city*, the *departure time* and the *departure date*. Furthermore, the fillers for these slots are restricted to a series of specific values (e.g., for a slot '*departure date*', the fillers can only be dates and not city names).

The process to obtain a system output from an user input follows a simple logic in GUS. First of all, three main pieces of information are extracted from the user command: the **domain**, the user **intent** and the **slot fillers**. For example, given the user request "I want to book a seat for the last train on Monday from Barcelona to Madrid", GUS would extract the following information:

- **Domain**: train-travelling
- **Intent**: book-train
- **Slot - Origin-City**: Barcelona
- **Slot - Destination-City**: Madrid
- **Slot - DepartureDate-Day**: Monday
- **Slot - DepartureDate-Time**: last

To obtain the domain, the intent and the slot fillers, GUS relies on static hand-written rules that determine this information based on the structure of the user input. Example (1) shows a rule to determine the *set-alarm* intent from the *alarm-clock* domain (extracted from

Jurafsky and Martin (2020)), and Example (2) shows a possible rule to determine the fillers for the *Origin-City* and the *Destination-City* slots, respectively, in GUS.

(1) a. `wake me (up) | set (the | an) alarm | get me up`

(2) a. '`from {ORIGIN-CITY}`'
b. '`to {DESTINATION-CITY}`'

Once all the information has been extracted from the user command, GUS determines whether all the necessary slots for the requested intent have been filled. If there is missing information, it will generate an output requesting the user for the values for the slots that do not have a filler. In GUS, this output is generated according to a set of templates created for each slot type and considering the slots filled. For instance, the output in Example (3) would be used by GUS if the information for the *Destination-City* slot was missing and the *Origin-City* slot was filled:

(3) a. 'Where do you want to go to from {Origin-City}?
b. 'Where do you want to go to from {Barcelona}?

In this dialogue conceptualization, the dialogue is considered as finished when all the necessary slots have been filled.

2.1.1.2 Pipeline-Based Task Oriented Dialogue Systems

Task-oriented dialogue systems, as mentioned above, are usually based on pipeline architectures inspired in the one in GUS, although they are more complex and modern. These architectures, as Gao, Xiong, Bennett, and Craswell (2022) note, are inspired in the principles of human cognition. In this sense, cognition is defined as a process in which the information in the environment and the internal conception of this environment (the properties and characteristics of the elements that are present based on one's perception) are obtained to then decide which action to take to modify that environment.

Considering this, pipeline-based task-oriented dialogue systems consist of a set of components that recreate to some extent the cognition process by supplying the following functions:

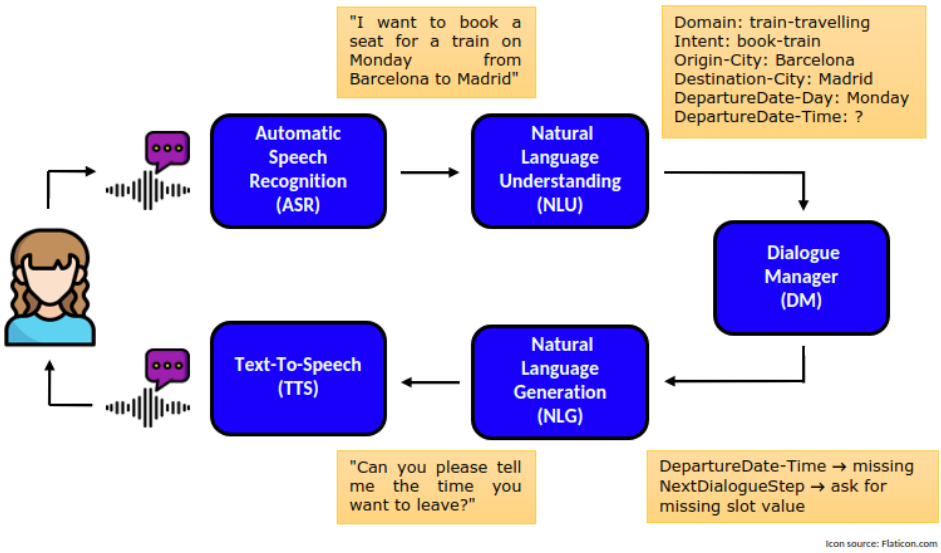


Figure 2.1: Typical architecture of a pipeline-based task-oriented dialogue system.

- To obtain an uttered user request in a text form: **Automatic Speech Recognition (ASR)** component[2].
- To interpret the user's command: **Natural Language Understanding (NLU)** component.
- To manage the dialogue process: **Dialogue Manager (DM)** component.
- To generate the response(s) to be presented to the user: **Natural Language Generation (NLG)** component.
- To output the generated system response in a voice form: **Text-To-Speech (TTS)** component[2].

Figure 2.1 shows the typical architecture of a pipeline-based task-oriented dialogue system considering the components above. First, the user utters a request directed to the system. The audio generated serves as input to the ASR component, which transcribes it and outputs a string that corresponds to the user command. This user command in string form is processed by the NLU module, which

[2] If the task-oriented dialogue system has speech-understanding/generation capabilities.

determines the domain of the user command and the intent to then retrieve the slots that apply to them. Then, the relevant pieces of information are obtained from the command to fill the slots. This information is given to the DM, which supplies two main functions[3]: first, it checks whether all the slot information has been filled in the NLU module and, then, it determines the action to execute depending on the outcome of this slot verification. With this information, the NLG module is in charge of generating a suitable output response that deals with the action determined by the DM. Finally, the output generated by the NLG module is converted into audio in the TTS component, which is then returned to the user.

As it can be observed, in pipeline-based task-oriented dialogue systems the situation is indeed similar to human cognition: the system obtains a conceptualisation of the environment according to information provided in previous dialogues and the user command in the NLU component and, then, it determines in the DM the action to execute considering this environment conceptualisation. The result is an output to be generated by the NLG module and/or a task execution (in case that all the necessary information to perform it has been obtained).

To develop each of the modules described in this section, there are several approaches in the literature. These are described in Chapter 3.

2.2 Semantic Web Technologies

The concept of Semantic Web (SW) –also referred as Web of Data or Web 3.0– was coined by Sir Tim Berners-Lee in 2001 (Berners-Lee, Hendler, & Lassila, 2001), in a context in which the World Wide Web –*Web* from now on–, had made information openly available to people in a human-understandable form, being a new paradigm for information sharing at the time. According to Allemang and Hendler (2011), the main features of the Web can be summarised in the following:

[3]In fact, some authors divide this module into two submodules: the *dialogue state tracker* and the *dialogue policy*, respectively (Gao, Galley, Li, et al., 2019).

- *The **AAA (Anyone can say Anything about Any topic)** slogan.* The Web has provided people with freedom to express and publish on the Web content about any topic.
- *The **Open World Assumption**.* As a consequence of the AAA slogan, it cannot be assumed that all information in the world has been published, not even all the information in regard to a specific topic/entity.
- *The Web as a **data wilderness**.* Considering the AAA slogan, in which people can contribute to the Web with anything, this means that there are all sorts of valuable information in any number of forms, and, due to this, it may not be understandable.
- *The **network effect**.* As anybody can publish anything on the Web, the more people that can consume it, the more content is created and the more people that contribute to it, resulting in a participation spiral.
- *The **Nonunique Naming Assumption**.* As there is no coordination between Web contributors regarding the naming of the resources they create, it is safe to assume that the same entity may be referred in more than one way.

Considering these features, the Web includes all sorts of potentially valuable information, but in an unstructured way and only consumable by humans. With the goal of facilitating cooperation between humans and machines, the SW was conceived as an extension of the Web to make the information on it also accessible to machines:

> "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." (Berners-Lee et al., 2001)

This cooperation was based on the possibility of humans to delegate on machines "sophisticated tasks" (Berners-Lee et al., 2001) on the data on the Web. Furthermore, the SW not only allows to structure the data on the Web, but also allows to establish links between the different pieces of information in it.

As the SW intends to provide a structured and interoperable means of accessing and linking data on the Web, it is important to have a set of standards to represent, present and exploit information. The organism that is in charge of providing these resources is the World Wide Web Consortium (W3C).

The following lines will describe the main concepts in relation to the Semantic Web.

2.2.1 Linked Data

As described previously, one of the main objectives of the SW is to structure data so it can be consumed by both people and machines. In this sense, Linked Data (LD) stands for the set of best practices for the data in the Web to be interlinked, resulting in a global data space. For this data to be properly linked, stored and accessed, the commonly known as Linked Data Principles were defined (extracted from (Berners-Lee, 2006)):

- Use Uniform Resource Identifiers (URIs) as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information.
- Include links to other URIs, so that they can discover more things.

However, LD is not conceptualised as a replacement of other data sources (such as databases) but as complementary in the sense that the fulfilment of these principles is encouraged to be able to access, process and structure heterogeneous sources of data.

2.2.2 The Building Blocks of the SW: RDF, OWL and Ontologies

The **Resource Description Language (RDF)** is the data model convention created by the W3C to represent and relate the different

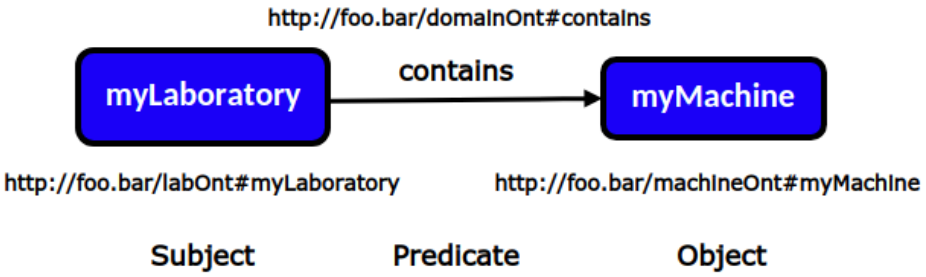


Figure 2.2: RDF triple representing "My laboratory contains my machine". Each element in the triple is represented in a HTTP URI form.

resources on the Web so they can also be understandable by machines as LD. This language describes resources in terms of triples that consist on the following components: the **subject**, the **predicate** and the **object**. To illustrate these components, Figure 2.2 depicts an RDF triple that represents the statement "My laboratory contains my machine". As it can also be observed in the Figure, each element of the triple is represented in a HTTP URI/Internationalized Resource Identifier (IRI)[4] form (Cyganiak, Wood, & Lanthaler, 2014). The use of URIs/IRIs to designate different pieces of data allows to identify them unambiguously. A set of RDF triples constitutes an RDF graph.

As specified above, RDF is a data model to describe resources in the Web and, therefore, it is a theoretical conception. So as to be interpreted by machines, it needs to be represented in a machine-readable format. For this, different serialisation formats have been created, some of which are standardised by the W3C and others not, but are widely used due to their easiness to be generated and understood by humans. The most common serialisation formats are the following:

- **RDF/XML**[5]. It is a standardised format to serialise RDF triples by using XML syntax. Although it is extensively used in the SW, it is not especially human-friendly to write or read.

[4]**URIs** are limited to US-ASCII characters, whereas **IRIs** support Unicode characters. Thus, **URIs are a subset of IRIs**. The main advantage of IRIs is that they reduce incompatibilities in the long run, as a wider range of characters are accepted. RDF accepts the use of both URIs and IRIs.

[5]`https://www.w3.org/TR/rdf-syntax-grammar/`

The following example serialises a situation in which a laboratory `myLaboratory` contains a machine `myMachine`:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
   syntax-ns#"
  xmlns:dom="http://foo.bar/domainOntology#">
  <rdf:Description rdf:about="http://foo.bar/
   laboratoryOntology#myLaboratory">
    <dom:contains rdf:resource="http://foo.bar/
     machineOntology#myMachine"/>
  </rdf:Description>
</rdf:RDF>
```

- **N-Triples**[6]. This non-standard format serializes RDF triples in plain-text form, and makes use of the absolute (that is, complete) URIs/IRIs of each element in the triple. These elements are juxtaposed, separated by spaces, and the triple represented ends with a dot. This representation is easily readable for humans. The following example shows the situation described above in the N-Triples format.

```
<http://foo.bar/laboratoryOntology#myLaboratory>
<http://foo.bar/domainOntology#contains>
<http://foo.bar/machineOntology#myMachine> .
```

- **Turtle** [7]. Turtle is a non-standard format inspired in N-Triples, and allows URI/IRI abbreviation by making use of prefixes, resulting in a compact format that is easily readable for humans. The following examples shows the situation described above in the Turtle format.

```
@PREFIX dom:<http://foo.bar/domainOntology#>
@PREFIX lab:<http://foo.bar/laboratoryOntology#>
@PREFIX mach:<http://foo.bar/machineOntology#>

lab:myLaboratory dom:contains mach:myMachine .
```

In order to to provide RDF with semantic significance, semantic extensions are used (Hayes & Patel-Schneider, 2014). In this sense,

[6] `https://www.w3.org/TR/rdf-testcases/#ntriples`
[7] `https://www.w3.org/TeamSubmission/turtle/`

RDF Schema (RDFS) is a basic semantic extension for RDF, that allows to group resources in classes (`rdfs:Class`) and determine specific properties (`rdfs:Property`) on them. These properties account for the belonging to a specific class (`rdf:type`), the hierarchical characteristics of classes and properties (`rdfs:subClassOf` and `rdfs:subPropertyOf`, respectively), the human-readable information about resources, such as a label (`rdfs:label`), comments (`rdfs:comment`) or where to find additional information (`rdfs:seeAlso`), as well as their provenance (`rdfs:isDefinedBy`). RDFS also allows to characterize[8] the resources that may be the subject (`rdfs:domain`) and the object (`rdfs:range`) of a triple in which a specific property is involved (e.g., for the property `contains`, as in the examples in this section, if its domain are resources from the class `Laboratories` and its range are resources from the class `Machines`, then it can be assumed that the subject of any triple with the predicate `contains` is of the class `Laboratories` and the object of the class `Machines`).

With RDF it is possible to represent data and their relations so that machines can interpret them explicitly but not implicitly, which means that additional information cannot be ***inferred*** from explicit representations. For example, considering the situation described in the previous examples, in RDF it cannot be assumed that `myMachine` is contained by `myLaboratory`, even when the relation `myLaboratory contains myMachine` is explicitly described. RDFS does allow a basic inference (for example, given the class `Machine` and the subclass `BigMachine`, it can be inferred that a member of the class `BigMachine` is also a member of the class `Machine`), but in a limited way.

To represent rich and complex knowledge, the **Web Ontology Language (OWL)**, which at the time is at its version 2.0 –it will be referred as simply OWL–, was created. OWL goes a step beyond representing content on the Web, and allows to model complex relations that can be exploited and interpreted by machines, providing a more expressive representation of Web content through the inference of new information by making use of reasoners. In a nutshell, OWL

[8]In no case domain and range establish *restrictions* on the class the subjects and objects may belong; these are descriptive features. In this sense, if a resource that is out of the domain of the property is the subject of the triple, it will be inferred that said resource belongs to the class in the domain, although it may not be consistent with reality.

supports the following (Motik & Patel-Schneider, 2012):

- Relationships between classes and properties. OWL enables establishing relationships between classes and properties other than hierarchical (i.e., `rdfs:SubClassOf`, `rdfs:SubPropertyOf`). In this sense, class/property disjointness (`owl:disjointWith`) is very important, as it allows to state that specific classes/properties cannot overlap. As for class disjointness, the members of a class cannot belong to its disjoint. For instance, given the disjoint classes `Machine` and `Laboratory`, the instances of `Machine` cannot belong to `Laboratory`, and vice-versa. For disjoint properties, the same individual cannot be related to the same (other) individual through properties that are disjoint. An example are the disjoint classes `hasParent` and `hasChild`, since a given person A cannot be the parent *and* the child of a given person B.

- Enriched properties. OWL provides richer characterisations of properties:

  - Inverse properties. When two properties are inverse, if the subject A is related to the object B through one of these, the relation `B inverseProperty A` is inferred. For example, considering the inverse properties `contains` and `isContainedBy`:

    * `myLaboratory contains myMachine .`
      implies `myMachine isContainedBy myLaboratory .`

  - Symmetrical properties. If a property is symmetrical, it means that, given the relation `A property B`, the property `B property A` is inferred. For instance:

    * `John isMarriedTo Mary .`
      implies `Mary isMarriedTo John .`

  - Transitive properties. When a property is transitive, if the same property is given between a subject A and an object B but also between subject B and object C, it is inferred that `A property C`, as it can be seen below:

    * `CPU isPartOf motherboard .`
      `motherboard isPartOf PC .`
      implies `CPU isPartOf PC .`

- Functional/InverseFunctional properties. Functional and InverseFunctional allow to infer equality between concepts. A typical example of functional property is `hasMother`. Since a person can only, biologically speaking, have one mother, if a subject A is related to objects B and C through the property `hasMother`, it can be inferred that B and C are equal (and can be related through the property `owl:sameAs`). For instance:
  - `Mark hasMother Mary .`
    `Mark hasMother MarySmith .`
    implies `Mary owl:sameAs MarySmith .`

  As for InverseFunctional, these properties establish a one-to-one relationship between a subject and an object. `hasID` is an example, since a person can only have an ID and an ID can only apply to one person:
  - `Mary hasID 45623964B .`
- Reflexive/Irreflexive properties. Reflexive properties allow an individual to be related to themselves. For example, `combs` would be reflexive, as a person can comb themselves (i.e., `A combs A`). Irreflexive properties do not allow that.
- Property composition (Krötzsch, Simancik, & Horrocks, 2012). In OWL it is also possible to define properties as the result of chaining two or more properties (also known as *complex role inclusion axioms).* For example, given the statement `A uncleOf B`, an equivalent affirmation would be that "`A` is the brother of the parent of `B`" and, thus, `uncleOf` can be also expressed by chaining the properties `brotherOf` and `parentOf`:
  - `SubObjectPropertyOf`
    `(ObjectPropertyChain(brotherOf ParentOf)`
    `uncleOf)`
  - `A brotherOf C .`
    `C parentOf B .`
    implies `A uncleOf B .`

- Restrictions. OWL allows to establish restrictions on the range of properties, such as cardinality, which determines, for example, the number of minimum or maximum objects that subjects of a given class may be related to through a specific

property. For instance, given the property `hasChild`, with a domain `Mother`, it can be established that any member of the class `Mother` must be related to a ***minimum* of one instance** (without specifying its class) through the property `hasChild`, as for a person to be considered a mother needs to have at least one child. If the cardinality also states that the individual must be of the class `Child`, it is considered a *qualified cardinality*. The difference is illustrated in the following formalisms:

- ```
  Class(Mother,
    subClassOf(Restriction
     (hasChild, minCardinality(1))))
  ```
  Regular cardinality: a mother is related to minimum one instance through the property `hasChild`.
- ```
  Class(Mother,
    subClassOf(QualifiedRestriction
     (hasChild valuesFrom(Child) minCardinality(1)
     )))
  ```
  Qualified cardinality: a mother is related to minimum one instance of the class `Child` through the property `hasChild`.

Other restrictions in OWL enable to define classes according to specific property values; that is, to establish *object value restrictions*. For example, given a class `NonVegetarianFood`, it may have a restriction that sets that any individual that contains meat (`hasIngredient meat`) belongs to it:

- ```
  Class(NonVegetarianFood,
    subClassOf(Restriction
     (hasIngredient, hasValue(meat))))
  ```
- ```
  Burger hasIngredient meat .
  ```
  implies `Burger a NonVegetarianFood .`

- Literals and datatypes. Ontologies in OWL can also use literal values for classes and individuals. For example, a given individual `MachineA` may have its serial number associated through the property `hasSerialNumber`. As it is not necessary to model an individual for that information, a literal may be used. OWL also allows to establish how the literal must be read as; that is, its *datatype*. Datatypes account for different literal types, such as booleans (`xsd:boolean`) or numbers (`xsd:integer`,

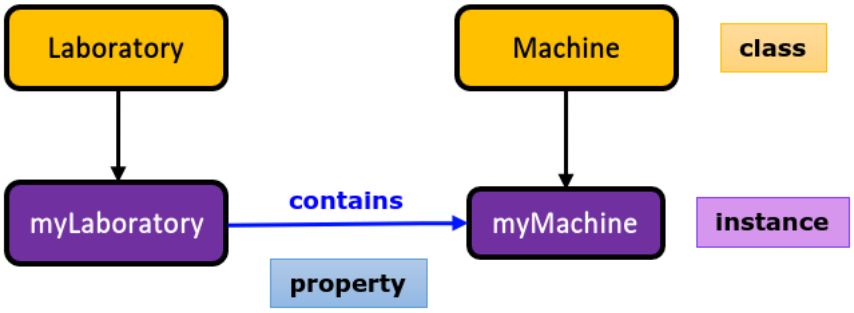


Figure 2.3: Basic components of an ontology.

`xsd:float`), among others. The use of literals and datatypes can be observed in the following examples:

- `MachineA hasSerialNumber "726S4872" .`
  No datatype.
- `MachineA hasSerialNumber "726S4872"^^xsd:string .`
  The literal "726S4872" must be read as a string.

Taking into account the wide range of relations that OWL allows (that is, its *expressivity*), this language also enables ontology development environments to verify, through reasoning, the consistency of data. If an inconsistent statement is found (e.g., a member of the class `Laboratory` is also a member of the class `Machine`, and those two classes are disjoint), it is possible for the reasoner to raise inconsistencies on data, that allow to identify modelling errors.

Considering the characteristics of RDFS and the modelling complexity that OWL allows, it is possible to represent complex knowledge in the form of **ontologies**. Ontologies are classically defined as "a formal, explicit specification of a shared conceptualisation" (Gruber, 1993). That is, ontologies are a formal representation to define and classify the concepts and the resources in the Web, as well as the relationships between them, regarding a specific subject. Ontologies consist of three basic components, all of them identified through IRIs, which can also be seen in Figure 2.3:

- **Classes** define groups or categories, which can be arranged hierarchically by using super- and sub-classes.

- **Properties** are equivalent to predicates, and allow to relate and describe classes and individuals.

- **Instances** (or **individuals**) represent the entities that belong to a class. The set of instances in an ontology is called **ontology instantiation**.

To formally represent the logic in OWL ontologies and describe their expressivity, formalisms and concepts from the Description Logic (DLog) field are used. Description logics[9] allow to describe and represent knowledge in a structured form and enable computers to perform reasoning on it, and constitute the basis of OWL ontologies (Baader, Horrocks, Lutz, & Sattler, 2017). Thus, it is possible to describe the expressivity of an ontology by using DLog terminology, considering the type of knowledge modelled. One of the most popular and basic description logics in DLog is $\mathcal{ALC}$ (Attributive Language with Complements) (Baader et al., 2017). Depending on the additional expressivity (in this case, of the ontology), $\mathcal{ALC}$ has a set of extensions, being the most important in the context of this thesis the following:

- $\mathcal{F}$ - Presence of functional properties.

- $\mathcal{H}$ - Presence of property hierarchy (i.e., presence of subproperties).

- $\mathcal{R}$ - Presence of limited complex role[10] inclusion axioms (Baader et al., 2017) and reflexive/irreflexive and disjoint properties.

- $\mathcal{O}$ - Presence of object value restrictions (e.g., `owl:hasValue`).

- $\mathcal{I}$ - Presence of inverse properties.

- $\mathcal{N}$ - Presence of cardinality restrictions.

- $\mathcal{Q}$ - Presence of qualified cardinality restrictions.

- $(\mathcal{D})$ - Presence of datatypes and/or literals.

[9] Note the difference between Description Logics as a field and description logic as a logic type used for description (which is in the field of Description Logics).

[10] In DL, *roles* are equivalent to properties in OWL.

To refer to ontologies with $\mathcal{ALC}$ expressivity that also present transitive properties, the label $\mathcal{S}$ is used.

According to Baader et al. (2017), OWL 2 has a $\mathcal{SROIQ}$ expressivity. Depending on the purpose or characteristics of the ontology, the presence of such rich expressivity may incur in reasoning efficiency problems. For this, three subsets (called *profiles*) of OWL have been released, which sacrifice different characteristics of OWL to ensure efficiency when performing reasoning on ontology data (Motik et al., 2012). The different profiles for OWL –which account for different ontology configurations depending on the purpose of the target application– are the following:

- **OWL 2 EL**. For applications that make use of ontologies with a large number of properties and/or classes.
- **OWL 2 QL**. For applications that make use of ontologies with a large number of instances.
- **OWL 2 RL**. For applications that perform reasoning through rule-based reasoners.

2.2.3 Querying the Semantic Web: SPARQL

SPARQL Protocol and RDF Query Language (SPARQL) (Harris & Seaborne, 2013; W3C, 2013), in its version 1.1, is the standard language for querying in the SW. It has been designed to query and handle content in RDF format, and its syntax is inspired in both SQL and Turtle languages.

To be able to extensively process data in RDF format, SPARQL supports the following operations on RDF data:

- **SELECT**. To retrieve values from a RDF graph according to a set of restrictions, supporting different output formats, such as tabular, XML, JSON or CSV/TSV.
- **CONSTRUCT**. To retrieve information in RDF format according to a set of restrictions.
- **ASK**. To retrieve a True/False result on a set of restrictions set in the query.

- **DESCRIBE**. Used to obtain relevant information from a specific instance in RDF format. This relevant information is not determined by the user/system performing the query, but by the query service (Harris & Seaborne, 2013).

- **INSERT**. Used to insert data in a given RDF graph.

Furthermore, SPARQL also supports more complex constructions, being common to order results (`ORDER BY`), set constraints (`FILTER`), avoid duplicates (`DISTINCT`) or set optional restrictions (`OPTIONAL`).

The following is a simple example of a SPARQL SELECT query used to retrieve the elements (`?machine`) that are contained by the individual `myLaboratory`:

```
PREFIX dom:<http://foo.bar/domainOntology#>
PREFIX lab:<http://foo.bar/laboratoryOntology#>
PREFIX mach:<http://foo.bar/machineOntology#>

SELECT ?machine

WHERE {
   lab:myLaboratory dom:contains ?machine .
 }
```

As it can be observed, this query consists of three main elements: the prefix declaration, the variables to return (preceded of `SELECT`), and the pattern to retrieve the desired information from (preceded of `WHERE`), in a similar manner as in SQL syntax.

2.3 Natural Language Processing

Natural Language Processing (NLP) is the branch of Computer Science that deals with providing computers with the ability of generating, processing and understanding natural language in the same way as humans. This discipline is intersected with the Linguistics field, which provides the foundations and mechanisms to enable the development of NLP resources.

The main challenge for the NLP field is to deal with the complicated nature of human language:

> "Human language is highly ambiguous [...], and also highly variable [...]. It is also ever changing and evolving. People are great at producing language and understanding language, and are capable of expressing, perceiving, and interpreting very elaborate and nuanced meanings. At the same time, while we humans are great users of language, we are also very poor at formally understanding and describing the rules that govern language." (Goldberg, 2017)

Following the quote above, NLP is centred in the research on the methods that allow computers to cope with the particularities of human language and extract the patterns and formulations to do so. Nevertheless, this discipline also provides mechanisms to use computers to work with language data at a shallower level (e.g., word frequencies or average number of words in a text/sentence).

Among the tasks that NLP applications perform, the most common are the following:

- **Speech recognition / speech generation**. These tasks deal with spoken language. On the one hand, speech recognition (or speech-to-text) deals with spoken language and transcribes it to obtain its textual form. On the other hand, speech generation (or text-to-speech) consists on the inverse task: given a text, its corresponding audio output is generated.
- **Syntactic analysis**. The main goal of syntactic analysis applications is to parse textual natural language sequences to obtain their syntactic representation; that is, the way the elements in the sequence relate to each other and the functions that these have in the sequence as a whole (e.g., subject, direct object).
- **Named Entity Recognition (NER)/Named Entity Recognition and Classification (NERC)**. NER/NERC applications identify, from textual input, the elements (such as words or phrases) that are considered as relevant entities (also known as named entities). In this sense, in NER tasks, the identification of such elements is performed and NERC, besides identifi-

cation, also determines the type of named entities. For example, in the sentence "John is from London", NER will determine that "John" and "London" are named entities, whereas NERC will additionally classify "John" as a person and "London" as a location.

- **Sentiment analysis**. The goal of this task is to extract, from a text, the subjective qualities that allow to characterise it, such as whether a text has a positive or negative connotation (e.g., a movie review).

- **Natural Language Generation (NLG)**. This task deals with the automatic generation of natural language sequences according to specific situations. Examples of the application of NLG are text summarisation or weather reports.

In order to perform these tasks, several approaches can be distinguished. Traditionally, NLP tasks have been performed by manually defining rules. However, modern approaches make use of Machine Learning (ML)-based techniques (including Deep Learning (DL)), which allow to automatically generate and process patterns from big amounts of data, usually obtaining better results than rule-based methods. These techniques simplify the amount of work to perform NLP tasks, although they rely on the data obtained and its quality. In this sense, rule-based approaches are still a valid option when quality data is not available.

To develop the modules for the semantic-based task-oriented dialogue system in this thesis, several NLP technologies will be used. Particularly, this section describes both the tools used and the resources exploited.

2.3.1 Tools

This section describes the two most relevant NLP tools in this thesis, which are the basis to obtain a correct interpretation of user commands: Foma (Hulden, 2009a) and Freeling (Carreras, Chao, Padró, & Padró, 2004). Among the many utilities of these tools –which will be described in the following sections–, in the context of this thesis they will be used to perform rule-based text processing

and syntactic analysis, respectively. More specifically, these tools will be used to perform tasks that are analogous to NERC.

2.3.1.1 Foma

Foma (Hulden, 2009a) is a compiler for finite-state automata (FSAs) and transducers (FSTs). It is intended to perform NLP tasks, although it can be used in other cases where FSAs and/or FSTs are needed.

For the NLP field, some usage examples of Foma are the generation of morphological analysers (such as Part-of-Speech (POS) taggers) and generators (e.g., word generators based on morphological features) and spell-checking applications.

To determine the processing to be performed, Foma makes use of grammars and/or lexicon files. For grammars, the same regular expressions in Xerox Finite State Tool (XFST)[11] (Beesley & Karttunen, 2003) may be used as rules, which have a human-relatively-readable format. Lexicons, on the other hand, are supplementary files that allow to model morphological information (e.g., the past tense of a regular verb is obtained by adding the suffix -ed to the infinitive form of the verb).

2.3.1.2 Freeling

Freeling (Carreras et al., 2004) is a library that allows to analyse language in textual form. According to their developers, Freeling allows to perform linguistic analysis at a wide range of levels, developed as mostly independent modules[12,13]:

- Tokenisation
- Phonetic encoding
- Sentence splitting
- Syntactic parsing
  - Shallow
  - Full/dependency

[11] In fact, Foma is considered the open-source analog to XFST.

[12] Extracted from Freeling's documentation in `https://freeling-user-manual.readthedocs.io/en/v4.0/`

[13] Some modules require the input to be previously processed by other modules. For example, tokenisation is usually required before any analysis.

- Morphological analysis
  - Punctuation detection
  - Number detection
  - Date detection
  - Dictionary search
  - Multi-word recognition
  - NER/NERC
  - Quantity detection
  - Part-of-Speech (POS) tagging
- WordNet sense annotation
- UKB[14] sense disambiguation
- Semantic Role Labelling
- Co-reference resolution

Furthermore, Freeling is available for multiple languages, such as Spanish, English, German, French, Catalan and Italian. However, not all functionalities are available for all languages[15].

As for the output generated, Freeling is able to produce the desired linguistic analysis in six different format types[16]:

- **Freeling**. Text-based, pseudo-column format that is human-readable.
- **CoNLL**. This structure is an adapted version of the one used in the *CoNLL* competitions which, as in the original, includes morphological (e.g., POS tags), syntactic (e.g., dependencies) and miscellaneous information (e.g., comments) for each token in a given sentence.
- **XML**. XML generic format for Freeling annotations.
- **JSON**. JSON generic format for Freeling annotations.
- **NAF**. Output in NLP Annotation Format (NAF)[17], which is a specialised XML-based schema for linguistic annotations.
- **Train**. Output generated in a specific format that allows to train POS taggers.

[14] `https://ixa2.si.ehu.eus/ukb/`

[15] `https://freeling-user-manual.readthedocs.io/en/v4.0/basics/`

[16] `https://freeling-user-manual.readthedocs.io/en/v4.0/analyzer/#output-format`

[17] `https://github.com/newsreader/NAF`

2.3.2 Resources

Considering that the task-oriented dialogue system developed in the context of this thesis is conceived as a generic and easy-adaptable solution, it is very important to take into consideration the implications that this entails. In this sense, among other uses, ontologies are used to interpret user commands and, to do so, linguistic information needs to be instantiated. However, manual population of the ontology with this information is a time and resource consuming task, since it needs to be instantiated as comprehensively as possible to cover the variability of natural language commands. Due to this, the tendency is to apply semi-automatic or automatic techniques to reduce the time and effort of doing so (Muscetti, Rinaldi, Russo, & Tommasino, 2022; Sanagavarapu, Iyer, & Reddy, 2022), which is specially relevant when adapting the system to different use cases. In this context, the exploitation of existing linguistic resources is key to simplify the ontology population task. These linguistic resources include linguistic information from one or more different perspectives, such as lexical or semantic information, which are useful to deal with different aspects of user command interpretation, such as identification of actions or user intents and slot filling.

This section reports the main resources exploited in this thesis: FrameNet –which deals with predicate information–, WordNet and the Multilingual Central Repository (MCR) –lexical information–, SUMO –semantic information– and Predicate Matrix (PM), which integrates different semantic, lexical, predicate and verbal resources –including the ones mentioned above– into a single one.

2.3.2.1 FrameNet

FrameNet (Baker, Fillmore, & Lowe, 1998) is a lexical resource for English developed at the University of Berkeley. The concept of FrameNet originates from Frame Semantics, which is the linguistic theory that asserts that words evoke what are called *frames*. In a nutshell, FrameNet constitutes a multilingual predicate resource that aims to model situations (*frames*) and the words that elicit them (*lexical units*) in a comprehensive way. Moreover, it also provides with the actors that take part on these situations (*frame elements*) in a compulsory manner (that role must be present) or optionally (that

role may be present or not), as a sort of a slot-modelling approach.

For example, for the frame *Motion*, as in FrameNet's online version[18]:

> "*Some entity (**Theme**) starts out in one place (**Source**) and ends up in some other place (**Goal**), having covered some space between the two (**Path**). Alternatively, the **Area** or **Direction** in which the **Theme** moves or the **Distance** of the movement may be mentioned*".

In this frame definition, the situation is described, along with the mention of the different actors that take place in said situation and both distinguishing between compulsory frame elements (such as ***Theme*** or ***Source***) and the optional ones (e.g., ***Direction***). These frame elements are also described one by one with more detail in each frame entry.

The situations described in frames are evoked by lexical units (LUs), as noted above. LUs correspond to words (predominantly verbs, although there are also nouns and adjectives, in a minor scale) that are paired with a meaning and a semantic frame. Thus, a polysemous word (a word that has different meanings) would have as many LUs as meanings it has, and each meaning may be linked to one or more frames[19].

Each frame can have different LUs mapped, which allows grouping semantically similar words through frames, which adds special value to its data.

Furthermore, FrameNet has been developed for other languages other than English, such as Spanish (Spanish FrameNet - SFN (Subirats & Sato, 2003)) German (Saarbrücken Lexical Semantics Acquisition Project - SALSA (Burchardt et al., 2006)) and Japanese (Japanese FrameNet - JFN (Ohara et al., 2004)). However, the degree of comprehensiveness is not as high as in English. The four previously mentioned FrameNets are available online for free use[20]. Other languages include French (ASFALDA French FrameNet (Candito et al., 2014)), Chinese (Chinese FrameNet - CFN (You & Liu,

[18] `http://sato.fm.senshu-u.ac.jp/frameSQL/fn2_15/notes/index.html`

[19] As FrameNet is a work in progress, some meanings may not be mapped to any frame yet.

[20] For English: `http://sato.fm.senshu-u.ac.jp/frameSQL/fn2_15/notes/index.html`; through this link the other resources can be accessed.

2005)), Brazilian Portuguese (Salomão, Torrent, & Sampaio, 2013), Swedish (Heppin & Gronostaj, 2012) and Korean (Nam et al., 2014).

2.3.2.2 WordNet

WordNet (Miller, 1995) is a lexical knowledge base that stores nouns, verbs, adjectives and adverbs –and their corresponding senses– and groups them semantically into what are called *synsets*, which are described in Miller (1995) as 'sets of cognitive synonyms'. In this sense, words are interlinked in semantic terms, taking into account the senses of each word.

There is a wide range of relations among words in WordNet, being the most remarkable *synonymy* and *super-subordinate relations*, that mainly state upper and lower categories between words. An example is *hyperonymy-hyponymy*, in which a set of terms –*hyponyms*– are specializations of a more general term –*hyperonym*–, as in Flower (*hyperonym*) – Rose, Sunflower (*hyponym*).

The considerations above show that WordNet is a very powerful tool in terms of lexical information and semantic relations between words. Another of its strengths is that, due to the robustness of the concept, it has been adapted to several other languages, such as Chinese (Huang et al., 2010), Italian (Artale, Magnini, & Strapparava, 1997), Spanish (Fernández-Montraveta, Vázquez, & Fellbaum, 2008) or French (Sagot & Fivser, 2008), among many others[21].

It is an open resource, and it may be accessed online[22] or through freely downloading its data.

2.3.2.3 Suggested Upper Merged Ontology (SUMO)

The Suggested Upper Merged Ontology (SUMO) is intended as an upper level ontology to serve as a "foundation for more specific domain ontologies" (Niles & Pease, 2001) by including a wide range of general-purpose terms that comprehensively cover different fields, such as Linguistics, Computer Science or Artificial Intelligence. Also,

[21] `http://compling.hss.ntu.edu.sg/omw/`

[22] `http://wordnetweb.princeton.edu/perl/webwn`, for English.

these terms are formally defined through axioms, which makes SUMO a specially enriched resource. Thus, the aim of SUMO is to provide a comprehensive, precisely defined term ontology.

In order to obtain such a wide-coverage ontology, other ontological resources have been integrated, such as the MId-Level Ontology (MILO), or ontologies related to Communications, User interfaces, Law or Music[23]. Currently, SUMO includes around 25,000 terms and is available in several languages: Hindi, Chinese, Italian, German, Czech and English. Moreover, it is also mapped to the WordNet lexicon allowing the association of words to specific SUMO terms.

2.3.2.4 Multilingual Central Repository (MCR)

The Multilingual Central Repository (MCR), which at the time of writing this thesis is on version 3.0, aims to provide a powerful and rich multilingual lexical knowledge base (Atserias et al., 2004). This lexical knowledge base integrates data from multiple resources, including WordNet –in its 3.0 version– in six different language versions (English, Spanish, Catalan, Basque, Galician and Portuguese). Moreover, following the EuroWordNet architecture (Atserias et al., 2004), the MCR interconnects these wordnets using interlingual indices (ILI), also based in WordNet 3.0, for equivalent synsets in different languages. Furthermore, the repository has been enriched with ontological knowledge coming from semantic resources such as Base Level Concepts (Rosch, 1977), WordNet domains (Bentivogli, Forner, Magnini, & Pianta, 2004) and SUMO[24] (Guinovart, Gonzalez-Dios, Oliver, & Rigau, 2021; Pease, Niles, & Li, 2002).

All in all, MCR constitutes a very powerful multilingual lexical repository, which is available both online [25] and as a MySQL database [26].

[23] `https://github.com/ontologyportal/sumo`

[24] Equivalence ("="), subsumption ("+") and instantiation ("@") mapping relations (Alvez, Gonzalez-Dios, & Rigau, 2019).

[25] `https://adimen.si.ehu.es/cgi-bin/wei/public/wei.consult.perl`

[26] `http://adimen.si.ehu.es/web/files/mcr30/mcr30-2016.zip`

2.3.2.5 Predicate Matrix (PM)

Predicate Matrix (PM) (De Lacalle, Laparra, & Rigau, 2014) is a lexical resource that arised from the necessity of the integration of verbal and predicate-related resources. Examples of these resources are VerbNet (Schuler, 2006), FrameNet and WordNet. Each resource, according to De Lacalle, Laparra, Aldabe, and Rigau (2016), presents characteristics that the rest of alternatives do not offer which, in the end, generated the aforementioned need to integrate them into a single repository.

To perform the mappings between resources, PM uses Semlink (Palmer, 2009), which makes use of manual mappings, and its coverage is extended through automatic methods. In this case, automation allows to update the set of mappings when any of the knowledge resources is updated with low maintenance costs. Another reason to extend Semlink's coverage is due to the fact that Semlink is unable to perform specific mappings between resources that have different granularity (de Lacalle et al., 2014), and some mappings have blank (*NULL*) correspondences, as it can be seen in Table 2.1. This Table shows that verbs (in the *member* column) may have different senses (in the *class* column), and that these senses may not have a corresponding FrameNet lexical unit (in the column *lexical-unit*), although this does not imply that the VerbNet form does not correspond to a specific FrameNet frame (*frame* column). In this sense, the automatic mappings to construct PM reduce these *NULL* elements, although some of them still remain.

In regard to the resources integrated, PM includes data from a wide range of repositories, including the previously mentioned VerbNet, FrameNet and WordNet, along with the MCR (Gonzalez-Agirre, Laparra, & Rigau, 2012), the Basque Verb Index (BVI) (Estarrona, Aldezabal, & de Ilarraza, 2020), AnCora (Taulé, Martí, & Recasens, 2008), PropBank (P. Kingsbury & Palmer, 2003), NomBank (Meyers et al., 2004), the Event and Situation Ontology (ESO) (Segers et al., 2015) and SUMO. Thus, it constitutes a very comprehensive knowledge base, that not only gathers knowledge from verb/predicate resources, but lexical (such as WordNet or NomBank) and semantic (e.g., ESO and SUMO). Furthermore, the integration of many resources in different languages (either because each repository was available in several languages –such as WordNet and their ILIs– or

Table 2.1: Example of Semlink's mapping limitations between VerbNet and FrameNet (extracted from de Lacalle et al. (2014) and adapted).

VerbNet		**FrameNet**	
class	**member**	**frame**	**lexical-unit**
13.1-1	sell	Commerce_sell	sell.v
13.5.1	buy	Commerce_buy	buy.v
53.1-1	delay	Hindering	delay.v
53.1-1	delay	Change_event_time	*NULL*
13.5.3	employ	Employing	*NULL*
105	employ	Using	*NULL*

due to the integration of language-specific resources, such as the BVI) has also allowed to obtain mappings in different languages, which, on the whole, makes PM a "multilingual interoperable predicate lexicon" (De Lacalle et al., 2016).

It is worth noting that PM is centred in verbal information. Thus, for resources that account for multiple parts of speech (such as nouns or adjectives in WordNet or frames that have nominal lexical units in FrameNet), only verbal data is integrated, so as to be able to properly map the data from all the resources involved.

In this thesis, PM will serve as the core source to access to the rest of the resources described in this section.

Chapter 3

Research Framework

Given the fundamental technologies in this thesis, described in the previous Chapter, a review of the available literature regarding them is presented in this one. First of all, the different approaches for the development of each of the components in pipeline-based task-oriented dialogue systems are described. Moreover, the state of the art regarding different aspects of task-oriented dialogue systems is also overviewed: their design as Lifelong Learning (LL) systems, their application in industrial contexts and the use of Semantic Technologies to develop them.

3.1 Pipeline-Based Task-Oriented Dialogue Systems

Two main architectures exist for the design of task-oriented dialogue systems: the pipeline-based and the end-to-end ones. The former consist of a series of modules (described in Section 2.1.1.2) that perform different dialogue-related tasks, which are constructed in a pipelined structure in which, thus, the output of a module is the input of the next one. As for the latter, this process is performed in a single module. These systems are usually based on neural methods (H. Chen, Liu, Yin, & Tang, 2017) and, thus, require big amounts of data to be obtained. Although some datasets have been released for this matter, such as the presented in Budzianowski et al. (2018), these correspond to a set of limited domains. This thesis will focus on

pipeline-based task-oriented dialogue systems, as well as this Section.

Traditional approaches for pipeline-based task-oriented dialogue systems rely on rules and templates for natural language understanding and dialogue management (Goddeau, Meng, Polifroni, Seneff, & Busayapongchai, 1996; Ward & Issar, 1994; Wei et al., 2018). Nevertheless, recent advances in these fields have allowed the use of machine-learning-based techniques, both traditional Machine Learning (ML) (Lee, 2013; Lee & Eskenazi, 2013; Williams, 2013) and, more recently, Deep Learning (DL) (H. Chen et al., 2017; Henderson, Thomson, & Young, 2013; Mrkšić et al., 2015), which need large amounts of training data to be developed as a counterpart to not having to manually define rules or templates.

The following lines will describe in more detail the different technologies used in each module of the typical architecture of a pipeline-based task-oriented dialogue system. However, since the ASR and TTS modules are out of the scope of this thesis, related work on these technologies will not be provided in this section.

3.1.1 Natural Language Understanding Component

The objective of the NLU component is to extract the meaning of a user command and obtain a "semantic representation" (Skantze, 2007). The simplest approach to obtain this semantic representation is to first extract the key elements from the user command, along with their semantic meaning (Skantze, 2007), as it can be seen in Example (4).

(4) a. "I want to go to the Robotics laboratory".

b. `{action:` `go`
`destination:` `Robotics laboratory}`

To perform this task, linguistic methods have been widely used in the literature. Some approaches rely on syntactic information, such as dependency structure (Scheutz, Cantrell, & Schermerhorn, 2011; Thomas & Jenkins, 2012), syntactic parsing (Fasola & Mataric, 2013) or Contextual Knowledge Structures (triples consisting of *Subject-Verb-Object* relations) (Javed & Muralidhara, 2018). Also, Sidorov,

Velasquez, Stamatatos, Gelbukh, and Chanona-Hernández (2012) present syntactic features that are suitable for Machine Learning (ML), obtained from syntactic trees and resembling *n-grams*: the *syntactic n-grams* (*sn-grams*).

In this context, key element extraction resembles a NERC problem, in the sense that the key elements have to be identified and classified. The literature distinguishes two basic approaches to NERC: rule-based and ML methods (Maynard, Bontcheva, & Augenstein, 2016). For rule-based methods, Context-Free Grammars (CFG) are usually generated (Skantze, 2007), which can also be enhanced to achieve robustness (Kasper, Kiefer, Krieger, Rupp, & Worm, 1999; Van Noord, Bouma, Koeling, & Nederhof, 1999), although patterns, regular expressions or finite-state machines are also used (Nadeau & Sekine, 2007). As for ML-based methods, feature-based, supervised systems based on SVM, CRF, Maximum Entropy and Averaged Perceptrons have traditionally obtained the best results in NERC tasks (Agerri & Rigau, 2016). In these tasks, the use of linguistic-based features such as lemma or POS has deemed to be relevant (Agerri & Rigau, 2016). Nevertheless, more recent approaches make use of DL techniques, being Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) the most common (Yadav & Bethard, 2019), along with Transformers (Vaswani et al., 2017).

The use of one approach or the other depends on the circumstances of the experimentation: machine-learning-based methods are easier to maintain than rule-based ones, but require enough training data to train the models. However, despite the fact that manually generating rules is a time consuming task, rule-based systems are a more robust starting point in specific domains where there is a lack of raw or annotated data.

These key elements are to be processed by the NLU to perform three main functions (J. Liu, Li, & Lin, 2019): to detect the domain for which a user command has been uttered, to identify the intent of said command to retrieve the slots to be filled, and to fill the slots with the information in the user command[1].

Usually, task-oriented dialogue systems are limited to single domains (Jurafsky & Martin, 2020). However, some approaches are compatible with multi-domain settings. In this sense, usually the do-

[1] *vid.* Chapter 2.1.1.1.

main detection task is performed jointly with intent detection. The assumption is that, once the intent is obtained, the domain is also retrieved (Z. Wang et al., 2014).

For intent detection, there is a wide range of works in the literature that deal with this task (Gupta, Hewitt, & Kirchhoff, 2019). In this sense, three main approaches to perform intent detection can be distinguished: rule-based methods, ontology-based methods, and ML-based methods. For machine-learning and ontology-based methodologies, a fair amount of data is needed and, thus, cannot cope with scenarios with limited or nonexistent labelled data, which implies the necessity of performing manual work, which is resource-consuming (H. Chen et al., 2017). This last scenario is also found in rule-based methods, which have the advantage of not requiring any training data to be trained, but have as a counterpart the fact of requiring manual work to be built.

In rule-based approaches, the most common methodology is to perform a manual analysis of potential commands that can be directed to the dialogue system in regard to a specific intent, in order to extract patterns. These patterns are usually word-based –as in GUS (Bobrow et al., 1977)–, as Example (5) shows. Rule-based approaches are very useful in contexts in which training data is scarce. However, rules are costly to maintain when changes need to be made.

(5) Word-based patterns to detect the intent *set-alarm*, in the *alarm-clock* domain (Jurafsky & Martin, 2020).

a. `wake me (up) | set (the | an) alarm | get me up`

In ontology-based methods, intent detection is performed through the domain ontology, which needs to be populated with relevant information to perform this task, which represents the main challenge in this setting since in most cases it requires manual work to do so. In this context, ontology-based methods are very similar to rule-based approaches, as the ontology population process is usually performed according to word-based patterns retrieved from domain data (Cassier, Sellami, & Lorré, 2019; Quamar et al., 2020). However, in this case, ontology-based methods can be considered as a step forward from rule-based approaches, since patterns are usually obtained through automatic or semi-automatic methods –that combine expert knowledge with automatic methods– and do not require as much

manual work than the rule-based ones. Furthermore, this approach allows to obtain a structured representation of intent-relevant data, which has as advantage that the maintenance of the solution is simpler: if the current instantiation needs to be updated, new examples would just need to be instantiated, without having to potentially modify existing rules –and the risk that that may imply.

As a side note, and regarding ontology population at a higher level, a well-known challenge to the Semantic Web community is how to reduce the amount of manual work. As noted by Kontopoulos, Mitzias, Riga, and Kompatsiaris (2017), most approaches for ontology population make use of textual input and rely on Natural Language Processing techniques to obtain the necessary knowledge to populate the ontology (Corcoglioniti, Rospocher, & Aprosio, 2016; Makki, 2017), whereas approaches that rely on data that is structured at some degree are less common (Kontopoulos et al., 2017; Leshcheva, Blagov, & Pleshkova, 2017). Considering this, the authors in Kontopoulos et al. (2017) use structured knowledge in Linked Data for ontology population.

For machine-learning-based methods for intent detection, several approaches can be observed in the literature that make use of traditional ML algorithms such as Support Vector Machines (Cortes & Vapnik, 1995) or logistic regression (Bishop, 2006). However, modern approaches employ DL methods, such as word embeddings or CNNs (Q. Chen, Zhuo, & Wang, 2019; J. Liu et al., 2019; Louvan & Magnini, 2020). Among these two, DL methods obtain better results than regular, traditional ML algorithms, but both approaches require a fair amount of data to perform training tasks, especially DL models.

Considering the previous remarks, intent detection seems to require either a large volume of data or manual work, which nowadays still represents a challenge in the NLP community.

Finally, for slot filling, this task is often characterized in a similar way as the key element extraction task described above. In this sense, slot filling is considered a sequence labelling task in which words are assigned a tag according to the slot they belong to (H. Chen et al., 2017; Firdaus, Kumar, Ekbal, & Bhattacharyya, 2019). However, in this thesis, these two problems will be considered as different: key element extraction will be considered as a preliminary step to support

the slot filling task.

3.1.2 Dialogue Management Component

For dialogue management, rule-based and statistically-based approaches are the most predominant methods in the literature to determine the next step to be performed by the dialogue system.

Rule-based dialogue management is widely extended both in traditional and modern approaches (Bohus & Rudnicky, 2003; Fast et al., 2018; Koller, Baumann, & Köhn, 2018). In these, the dialogue flow may be encoded in terms of patterns and their associated responses or more sophisticated approaches (Brabra et al., 2021), such as the use of preconditions and associated actions that rely on the output generated by the NLU component (Bohus & Rudnicky, 2003; Thorne, 2017). Other dialogue management architectures consider dialogues as a sequence of steps to be modelled as a finite-state model (Fast et al., 2018; Koller et al., 2018), in which the dialogue process transitions from a state to another according to a set of conditions given the current dialogue state and the information provided by the user. An example diagram for finite-state dialogue modelling can be observed in Figure 3.1.

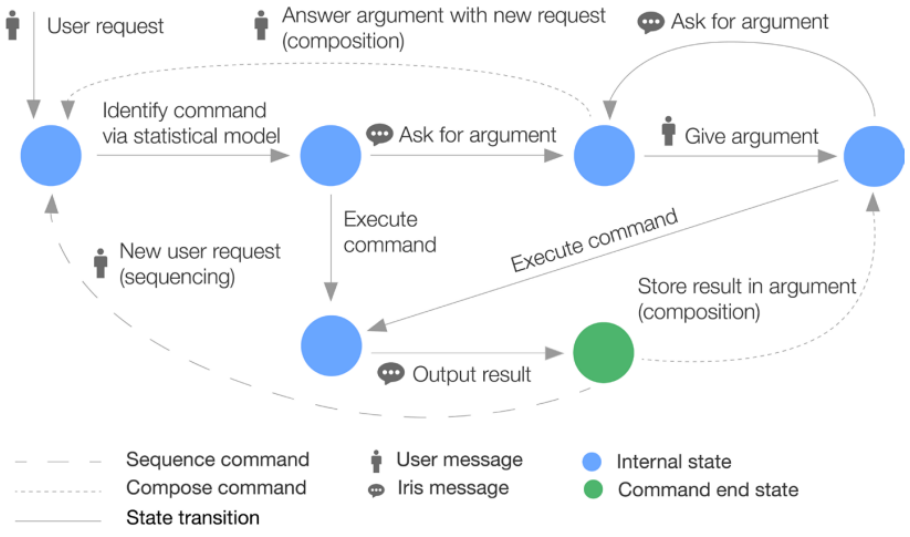


Figure 3.1: Finite-state dialogue modelling diagram (Fast et al., 2018).

The main advantage of rule-based approaches is that no training data is needed and rules are relatively easy to implement. However, the main disadvantages are that rules require effort to be constructed and maintenance costs are high, as adaptations to cover new functionalities imply expert manual work. Furthermore, as Brabra et al. (2021) point out, user profiling is also supported, although not extensively used (Smith et al., 2011). Other rule-based dialogue managers also enable the exploitation of past interactions to cover missing information (Bohus & Rudnicky, 2003; Smith et al., 2011).

Statistically-based dialogue managers conceive the dialogue process as a probabilistic problem. In this context, to model the dialogue as a Markov decision process (MDP) is a popular approach in many implementations (Levin, Pieraccini, & Eckert, 1998, 2000; S. J. Young, 2000). This process is based on a representation of the dialogue as a set of states and the use of a reward system, in which the transitions to some states give positive or negative rewards, contributing with a Reinforcement Learning setting. The system's goal is to obtain the maximum reward when fulfilling a user request. The use of MDPs has proven to be useful in restricted contexts where the goal is clear, such as train/plane booking or timetable information retrieval (Roy, Pineau, & Thrun, 2000). However, in more variable contexts, such as the ones in which there are several actions to be performed or noisy environments, it is assumed that the user intention is not always clear; that is, it is partially observable. To tackle with this uncertainty, partially observable Markov decision processes (POMDP) have been widely used in the literature (Lei, Wang, & Yuan, 2019; Roy et al., 2000; S. Young, 2006; S. Young et al., 2010). The main difference between POMDPs and MDPs is that MDPs have a certainty of the current state of the dialogue and decisions are made according to that state, whereas POMDPs add an uncertainty component that determines the current state as a *belief* obtained from a series of *observations*. In this case, the *observations* would be the information given by a user interaction, and the dialogue system *believes* that the state of the dialogue is the current one based on these observations.

Although statistical methods allow to model the dialogue process in a more efficient way than rules and the on-line optimisation of the model through the use of rewards, dialogue domain data is necessary for training an initial implementation (S. Young, Gašić, Thomson, & Williams, 2013).

Finally, the combination of rule- and statistically-based techniques has also been observed in the literature in which statistically-based dialogue managers benefit from the knowledge in rules (Williams, 2008; Yoshino, Watanabe, Le Roux, & Hershey, 2013).

3.1.3 Natural Language Generation Component

Peng et al. (2020) distinguish between two main approaches to generate the responses to be output by the dialogue system: template-based methods and statistical language models.

Template-based methods consist on a set of templates that are generated according to the different outcomes that the interpretation of a user command may imply (e.g., an argument is missing or the system has not understood the user command). These may vary in complexity, which may include from a set of slots to be filled with the information related to the use case at hand to structures that account for different variants and realisations (Gatt & Krahmer, 2018). To illustrate the former, Example (6) shows a simple template structure to be used in an alarm assistant. In it, the predefined response includes two slots to be filled: `time`, to account for the time to be woken up and `recurrent`, which refers to the repetition of the alarm set. The advantage of this approach is that the probability of generating incorrect or ungrammatical sequences is very low. However, the naturality of the command may be compromised. According to Jurafsky and Martin (2020), this approach is widely used in frame-based task-oriented dialogue systems.

(6) "I will wake you up at `$time $recurrent` ."
"I will wake you up at *half past eight every Friday* ."

As for more complex template-based models, Example (7), extracted from Langkilde and Knight (1998), shows a template based in abstract meaning representations. In this case, lexical and morphological information, extracted from linguistic resources, is also included to generate a wider range of possible outputs. In this Example, the structure is defined in terms of `label / concept` (e.g. `m4 / |dog<canid|`, that is equivalent to determine that `m4` is an instance of `dog`). Furthermore, it can be restricted to a specific number

(`:quant plural` for the plural form). In this case, this template allows only plural variants for `dog` (e.g. "the dogs", "dogs") and all number variants for `bone` ("bone", "bones", "the bone", etc.), which combined with the verbal form `eat`, may result in the different realizations in Example (7b). This approach allows to obtain a wider range of expressions.

(7) a.
```
(m3 / |eat,take in|
      :agent (m4 / |dog<canid|
               :quant plural)
      :patient (m5 / |os,bone|))
```

b. Possible realizations: "The dogs ate the bone", "Dogs will eat a bone", "Dogs eat bones", among others (Langkilde & Knight, 1998).

As for statistical language models, modern approaches include DL techniques. Gao et al. (2019) highlights the strategy in Wen et al. (2015) and its semantically-controlled LSTM (SC-LSTM), in which semantic information is used as a feature for NLG. This technique, as DL-based techniques in general, requires big amounts of training data. Taking this into account, Peng et al. (2020) propose SC-GPT, a neural language model based on Transformers for which, according to the authors, only a small amount of domain labelled data for fine-tuning is necessary for its adaptation to new domains. This model shows very positive results compared to other models that require bigger sets of training data, and it is an interesting asset to develop DL-based NLG components, even when labelled domain data is scarce.

Considering the two main approaches for NLG presented, template-based techniques are more robust than statistical language models, and are highly suitable for low-resourced scenarios, but require manual work for template design. On the contrary, DL techniques are more prone than errors than the template-based ones, but they do not require manual work. Furthermore, although these approaches traditionally require big amounts of data, there is research oriented towards their use in scenarios that lack training data.

3.2 Lifelong Learning in Task-Oriented Dialogue Systems

Lifelong Learning (LL) is a discipline that has been treated in the literature for a long time (Thrun, 1998). The objective of achieving LL systems is that these are able "to continually learn over time by accommodating new knowledge while retaining previously learned experiences" (Parisi, Kemker, Part, Kanan, & Wermter, 2019). Considering this, the implementation of LL capabilities has been studied in the context of several fields, such as Machine Learning (ML) or Artificial Intelligence (AI) (Z. Chen & Liu, 2018; B. Liu & Mazumder, 2021; Parisi et al., 2019). For each of them, different methods have been used, as these depend on the knowledge to gather and the learning strategy (Veron, Ghannay, Ligozat, & Rosset, 2019). In many cases, the systems developed around these fields are implemented in real-world settings, in which information is constantly obtained. In these contexts, LL systems have been deemed as crucial (Parisi et al., 2019).

In general, LL is achieved through Machine Learning (ML) and Deep Learning (DL) techniques (B. Liu & Mazumder, 2021; Parisi et al., 2019). Related to this, one of the main concerns in LL is *catastrophic forgetting*, in which new information interferes with previously acquired knowledge, which usually occurs when the new knowledge differs significantly from the one stored (Parisi et al., 2019). This challenge has been broadly treated in the literature, with a wide range of proposals (Choy, Srinivasan, & Cheu, 2006; Gepperth & Karaoguz, 2016; Goodfellow, Mirza, Xiao, Courville, & Bengio, 2013; Kirkpatrick et al., 2017). However, it is still a current issue to be considered and prevented when developing LL systems.

In LL dialogue systems, as in LL in general, the system's performance should not be limited by its initially-implemented knowledge or capabilities. Thus, the objective is that the system is capable of constantly improving by effortlessly acquiring and accumulating knowledge through new interactions with the user (Z. Chen & Liu, 2018). For this, Veron et al. (2019) identify two main actions: to identify when new knowledge appears and to determine when and what to ask the user to obtain it. The dialogue system in Li, Miller, Chopra, Ranzato, and Weston (2016) asks questions in three situa-

tions: when the system does not understand and needs rephrasing or a validation, when the system needs additional information to infer new knowledge and when the system does not know a specific answer. In turn, B. Liu and Mazumder (2021), in their Lifelong INteractive learning in Conversation (LINC) paradigm, the system is able to learn in three areas: factual knowledge, language expressions and skills. For the first, the objective is to acquire new facts from user utterances, such as asking for unknown words. However, the authors identify some challenges, such as co-reference resolution and the association of different forms to the same entities (e.g. Obama - Barack Obama). For language learning, the authors distinguish between natural dialogues with the user and user demonstrations (in these, if the system does not understand the user command, the user may perform the intended action through a GUI, remote control or similar tools (Forbes, Rao, Zettlemoyer, & Cakmak, 2015; S. I. Wang, Ginn, Liang, & Manning, 2017)). Finally, the skills to learn are related with the emotional state of the user or the environment conditions (Q. Liu et al., 2020; Zhou, Huang, Zhang, Zhu, & Liu, 2018; Zhou, Young, et al., 2018).

B. Liu and Mazumder (2021) also highlight one of the major challenges in LL in dialogue systems: the fact that user information may be incorrect. To deal with this, the authors propose potential strategies, such as the storage of unverified knowledge in another database, to be verified through interaction with other users or the detection of contradictory knowledge.

Taking into account all the considerations above, and the conclusions in Veron et al. (2019), a LL task-oriented dialogue system should be able to perform the following actions: to identify new knowledge, determine the right moment to ask questions according to specific criteria, to have a strategy to store newly acquired knowledge and, finally, to decide when knowledge is suitable for learning.

3.3 Task-Oriented Dialogue Systems in Industrial Environments

The lines above have shown that DL-based methods are widely used for task-oriented dialogue systems, with promising results, as-

suming that there is enough data to train the models. In industrial scenarios, however, training data is scarce, and the use of ML approaches is still very limited. Although there are attempts to combine rules and machine-learning techniques (Suendermann et al., 2009), rule-based approaches are generally used in these scenarios due to their specific characteristics (Goddeau et al., 1996; Jurafsky & Martin, 2020). As a consequence, most task-oriented dialogue systems for industrial scenarios are heavily adapted to the task they have been designed for and cannot be reused in other contexts, and developing new ones for new use cases is bound to expert work and high time and costs (Jurafsky & Martin, 2020).

Furthermore, interaction in industrial contexts is usually oriented to one-way communication, from human to robot, and the system does not interact with the user in case there are inconsistencies or missing information (Gustavsson, Syberfeldt, Brewster, & Wang, 2017; Stenmark & Nugues, 2013). In most cases, this interaction is limited to specific commands (Gustavsson et al., 2017) and, in general, the possibility of using natural language is restricted.

Nevertheless, interaction with industrial systems is being oriented towards a natural communication (Maurtua et al., 2017; Stenmark & Nugues, 2013). The authors in Stenmark and Nugues (2013) present a system that makes use of existing predicate-argument resources (Propbank (P. R. Kingsbury & Palmer, 2002)) to map natural commands to logical representations (e.g., *put(piece, box)* for "Put the piece in the box"). The reuse of existing, comprehensive resources of these characteristics allow a high flexibility in the type of requests that can be directed towards the system. Furthermore, the domain is represented by making use of ontologies. However, this proposal has some limitations in terms of dialogue (it is only unidirectional at the moment) and the reported implementation does not include variants (i.e., synonyms) for the different terms involved in the interaction (e.g., objects).

3.4 Use of Semantic Technologies in Task-Oriented Dialogue Systems

In current approaches, Semantic Technologies have been considered both for the natural language understanding and the dialogue management components of task-oriented dialogue systems, as they are a powerful tool that allows to define in detail the domain and reduce ambiguity between agents (Antonelli & Bruno, 2017).

Most of the task-oriented dialogue systems in the literature that make use of ontologies are limited to highly specific use cases and rely on these technologies for domain modelling (Yakoub, Selouani, & Nkambou, 2015), although the tendency to use them for dialogue management purposes is increasing. The dialogue system presented in Altinok (2018) uses ontologies to model the domain, as well as certain state-related dialogue information (e.g., which is the current product that is discussed in the conversation) to implement a very simple dialogue state tracker. In industrial scenarios, the work in Maurtua et al. (2017) makes use of an ontology to model the domain in terms of possible actions to be performed by the robot and a description of the scenario.

The approach in Wessel, Acharya, Carpenter, and Yin (2019), OntoVPA, aims to obtain a dialogue system that is fully managed by ontologies, in which there is a distinction between a domain ontology and a dialogue ontology, which is used to manage the dialogue, keep track of the state of the dialogue and to store and control the responses and requests to be presented to the user. In this approach, requests and responses are highly dependent on the use case and, most importantly, the ontologies developed are not publically available. In the case of Teixeira, Maran, and Dragoni (2021), with a similar approach to Wessel et al. (2019), the ontology used, Convology, is intended for dialogue policy planning to optimize the best path to complete a dialogue by using AI. However, this approach is limited to health-related applications and, as in OntoVPA, the ontologies are not published online.

Taking into account these considerations, Semantic Technologies are being currently used to develop task-oriented dialogue systems, although there are not many attempts to totally manage these sys-

tems through them. Thus, it is a field that it is worth exploring given the many benefits of using ontologies, such as interoperability or inference capabilities.

Chapter 4

The Task-Oriented Dialogue management Ontology (TODO)

The Task-Oriented Dialogue management Ontology (TODO) aims to provide a core and complete base for semantic-based task-oriented dialogue systems in the context of industrial scenarios in terms of, on the one hand, domain and dialogue modelling and, on the other hand, dialogue management and tracing support. Furthermore, its modular structure, besides grouping specific knowledge in independent components, allows to easily extend each of the modules, attending the necessities of the different use cases. These characteristics and its generic design allow an easy adaptation of the ontology to different use cases and languages, with a considerable reduction of time and costs when developing task-oriented dialogue systems that allow a natural communication with limited amounts of data –as it will be seen in Chapter 6–, contributing to overcome the main challenges in regard to task-oriented dialogue systems for industrial scenarios[1].

Towards easily adapting TODO to different use cases and languages –and considering that ontology instantiation is a task that requires of a great amount of time and effort–, this Chapter also describes a strategy that leverages existing lexical, multilingual resources to semi-automatically populate domain ontologies –such as TODO's domain module– with a significant part of the knowledge

[1] *vid.* Chapter 1.

required for a specific scenario and language.

4.1 Ontology Development Methodology

In ontology development, two main considerations arise: first, ontologies have to be "carefully designed and implemented" (Esnaola-González, Bermúdez, Fernández, & Arnaiz, 2021), so as to properly model all the necessary information for their final use. Also, ontology development is becoming more and more centered in reuse (Suárez-Figueroa, Gómez-Pérez, & Fernández-López, 2012). Given the above, it is important to follow a well-defined design methodology to develop ontologies that are optimal both for their intended function and to be reused by others. In the development process of TODO, the methodology followed is LOT (Linked Open Terms), in its industrial version (Poveda-Villalón, Fernández-Izquierdo, & García-Castro, 2019), as it focuses on design of ontologies oriented to industrial scenarios. This methodology sets four main steps of development:

- **Requirements specification**. It defines the motivation and the requirements to be fulfilled by the ontology, through the Ontology Requirement Specification Document (ORSD) (Suárez-Figueroa, Gómez-Pérez, & Villazón-Terrazas, 2009). The ORSD defines the purpose, scope, intended uses and requirements – defined as Competency Questions (CQ)– of the ontology.
- **Implementation**. By considering the requirements set in the previous step, the ontology is constructed and evaluated.
- **Publication**. Once the ontology has been created and properly annotated, its documentation is generated, and both ontology and documentation are published and made accessible online.
- **Maintenance**. This step includes periodical revisions with the aim to solve issues, add improvements, etc.

The following sections will document the TODO design process in terms of the first three steps defined in LOT, as the last one is understood as further periodic maintenance work after an initial version of TODO has been released.

4.2 Requirements Specification

Towards developing TODO, the ORSD leads to determine the specifications for its functional requirements; that is, the knowledge that the ontology must cover. In this sense, it is important to bear in mind that the main objective of the dialogue is to obtain a command that is understandable for a target system from a user request expressed in natural language.

To determine the required knowledge, three experts in collaborative industrial work and dialogue systems were interviewed to gather information about their necessities, the characteristics they considered that had to be covered by a task-oriented dialogue system and which type of interactions were expected. By using the information obtained, a series of requirements that had to be covered by the ontology were identified and codified as Competency Questions (CQs). In this sense, the necessities identified were related to the modelling of the elements contained in a specific domain and the mechanisms to, on the one hand, associate a natural language instruction with a specific command for the target system to execute and, on the other hand, to orchestrate and recreate the dialogue process.

Considering this, for TODO, a total of 99 CQs were obtained –reported in Appendix A–, which can be grouped into the following 11 Basic Competency Questions (BCQs):

- **BCQ01**. What are the elements that are present in the scenario? (CQ01-02)
- **BCQ02**. Which is the action to be performed by the target system given a series of key elements obtained from a user request? (CQ03-11)
- **BCQ03**. Which are the arguments of a specific action? (CQ12-13)
- **BCQ04**. Given a set of arguments from a specific action, to what argument can a key element from the user request be associated to? (CQ14-18)
- **BCQ05**. Which is the format of the information that has to be provided to the target system? (CQ19-26)

- **BCQ06**. Which is the first/next step of the dialogue? (CQ27-39)
- **BCQ07**. What should be told to the user given a specific situation? (CQ40-80)
- **BCQ08**. Given some output to the user, it is some input from the user required or not? (CQ81)
- **BCQ09**. Which is the type of information required by the request provided to the user? (CQ82-84)
- **BCQ10**. Which step is currently being performed in the dialogue? (CQ85)
- **BCQ11**. Which is the trace –and the information that includes– for an element? (CQ86-99)

These CQs have been used to define the scope of the ontology and to delimit its different areas of knowledge, along with the classes and relations to be modelled according to the remarks above. Furthermore, they have also helped to set the criteria to search for relevant ontological resources for reuse. These considerations will be addressed in the following sections, related to LOT's implementation step (Sections 4.3 and 4.4).

4.3 Ontology Modules

Considering the CQs, it has been observed that the areas of knowledge that TODO must address can be clearly separated and, due to this, TODO has been implemented as a modular ontology. This modular approach provides many beneficial aspects to ontologies in terms of maintenance, reasoning-processing[2], validation, comprehension, collaborative effort and reuse (Keet, 2018).

The requirements established in the CQs allow to determine that TODO's coverage areas correspond to two main dimensions: domain modelling (i.e., knowledge that is strictly related to the scenario in which the dialogue will take place) and dialogue modelling

[2]Reasoners and processing tools may take more time to work over big ontologies.

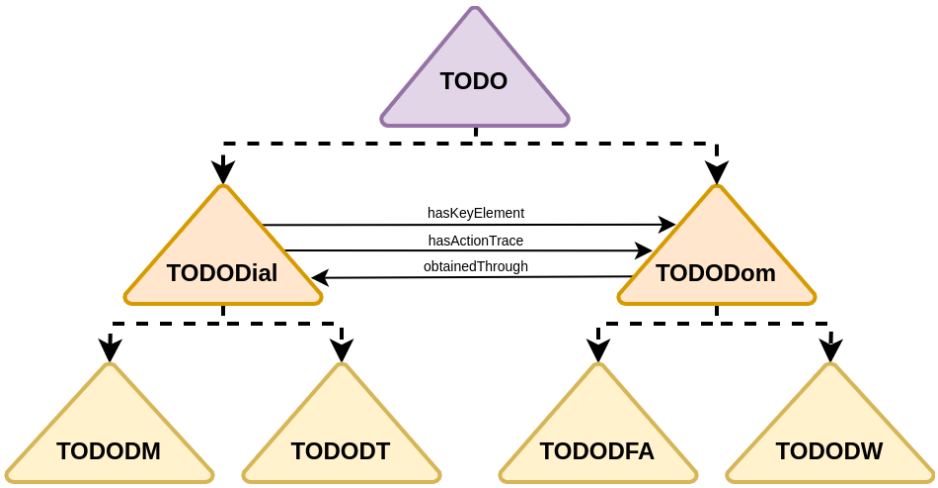


Figure 4.1: Overview of TODO: modules. Each color stands for the different module levels in the hierarchy, from less (top) to more specific (bottom) knowledge.

(i.e., knowledge related to the dialogue process). Furthermore, each of these dimensions cover specific areas of knowledge that have certain independence, which will be implemented as submodules: for the domain dimension, the elements that take part of the scenario (e.g., machines, robots) and the actions that apply to the use case's target system; for the dialogue dimension, the modelling of the dialogue process and the storage of dialogue traces to recreate from completed interactions and ultimately learn from them.

The general overview of TODO and its modules and the main relations between them can be seen in Figure 4.1. The limited number of relations between the modules accounts for TODO's modularity, as it will be seen in the evaluations in Section 4.4.

The following sections will describe the dialogue- and domain-related modules, along with each of their submodules.

4.3.1 TODODial - Dialogue Module

The TODODial module covers the knowledge that is necessary for the task-oriented dialogue system to manage and keep track of the dialogue process. More specifically, the TODODial module is responsible for the orchestration of the full dialogue process by responding

to the following BCQs:

- **BCQ06**. Which is the first/next step of the dialogue? (CQ27-39)
- **BCQ07**. What should be told to the user given a specific situation? (CQ40-80)
- **BCQ08**. Given some output to the user, it is some input from the user required or not? (CQ81)
- **BCQ09**. Which is the type of information required by the request provided to the user? (CQ82-84)
- **BCQ10**. Which step is currently being performed in the dialogue? (CQ85)
- **BCQ11**. Which is the trace –and the information that includes– for an element? (CQ86-99)

On the one hand, this module is in charge of managing the sequence of steps to be taken in a given dialogue by the task-oriented dialogue system. For each step, it also determines whether an output should be provided to the user and its content depending on the situation. Furthermore, in case a given dialogue system output requires an interaction from the user, TODODial also establishes of which type must be the user response to that output.

On the other hand, TODODial is also responsible of keeping track of ongoing and finished dialogues through traces that need to be stored. Considering this, TODODial is divided in two submodules, TODO Dialogue Management (TODODM) and TODO Dialogue Tracing (TODODT), which cover the two areas of knowledge described above respectively, as both are descriptive and independent enough to be treated separately.

Figure 4.2 shows a simple overview of the main classes and relations in TODODial, showing the ones corresponding to the Dialogue Tracing module classes in purple and the Dialogue Management module in yellow. The following sections will provide more details on both modules.

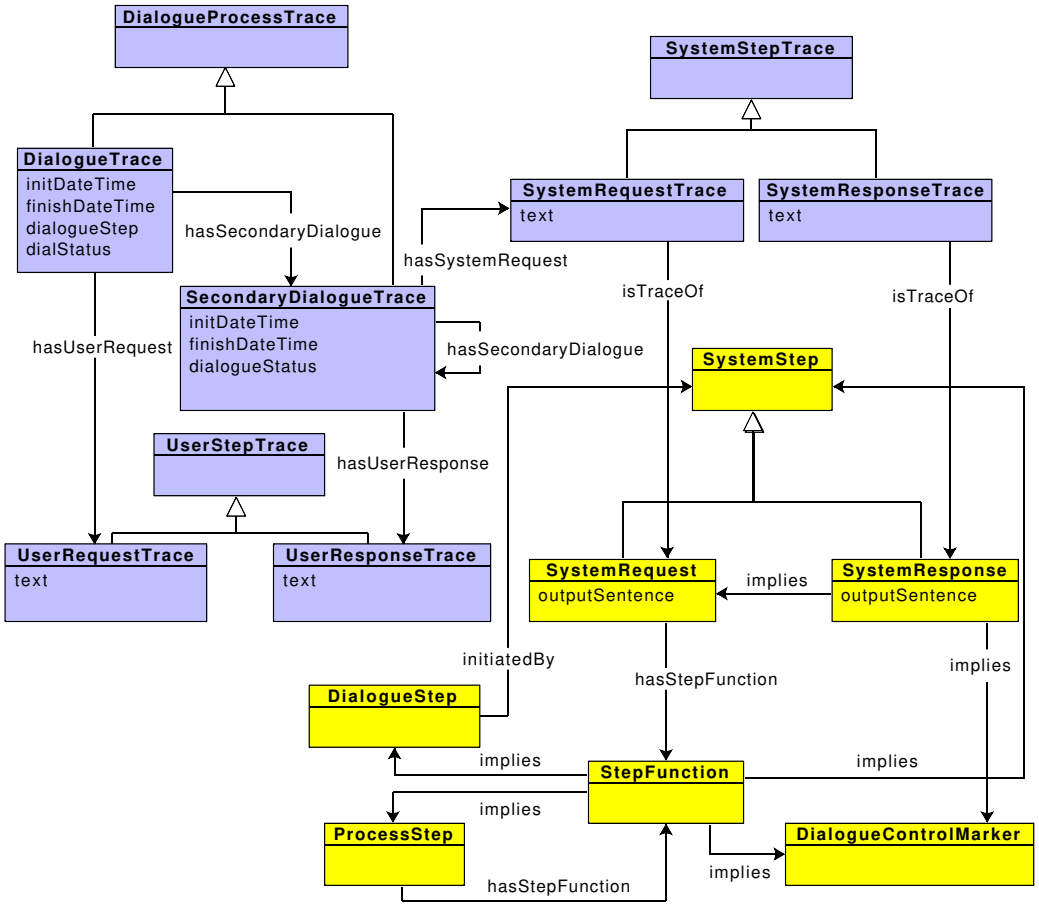


Figure 4.2: Simple overview of TODODial. Purple classes correspond to TODODT and yellow classes to TODODM.

4.3.1.1 TODODM - Dialogue Management Module

The TODODM module covers the knowledge related to the sequence of steps to be performed by the task-oriented dialogue system's dialogue manager component in order to maintain a dialogue with the user. Furthermore, it determines the output to be presented to the user in case the system needs to interact with them. To do so, this module tackles the following BCQs:

- **BCQ06**. Which is the first/next step of the dialogue? (CQ27-39)
- **BCQ07**. What should be told to the user given a specific situation? (CQ40-80)

- **BCQ08**. Given some output to the user, it is some input from the user required or not? (CQ81)
- **BCQ09**. Which is the type of information required by the request provided to the user? (CQ82-84)

This module distinguishes two types of steps to be performed by the task-oriented dialogue system's dialogue manager component: dialogue steps (`DialogueStep`) –which require some input by the user– and process steps (`ProcessStep`) –which do not require any user input and are intermediary steps that have the function of processing the information obtained from the user.

For dialogue steps, TODODM also establishes the output to be presented to the user (`SystemStep`), which can be of two types: responses (`SystemResponse`), which convey a piece of information, and requests (`SystemRequest`), which prompt the user to obtain information. For each individual of these response/request classes, the property `OutputSentence` contains a string that determines the output to be displayed, which is designed parametrically so as to be reused in different situations and use cases, with the possibility of including their equivalents in different languages. Examples (8) and (9) provide instances of responses and requests, respectively.

(8) "I have too many options with the information you provided me."

(9) a. "Do you want me to {Action}?" (parametric value)
b. "Do you want me to {show you the list of tools}?"

Furthermore, to correctly interpret a user input given a request, it is important to determine the type of response required. In this sense, TODODM distinguishes between two types of requests considering the required user response type: content requests (`ContentRequest`), which require an informative user response, and yes/no requests (`YesNoRequest`), which require a confirmation on the information provided. Examples 10 and 11 show and example of a content request and a yes/no request, respectively.

(10) a. "Can you please tell me the cartridge colour?"

b. **Expected response:** "yellow", "magenta"

(11) a. "Is 'magenta' the cartridge colour?"

b. **Expected response:** "yes", "no"

Finally, given a user input, it is necessary to process its content and to determine the next step to be performed by the task-oriented dialogue system. These necessities are covered by step functions (`StepFunction`), which are directly linked to specific functions in the dialogue manager that allow to process the user input (e.g., if the task-oriented dialogue system has requested the user to provide a cartridge colour, the corresponding step function will determine if the user input is suitable for the request performed by the dialogue system). These step functions are intended to be generic and, thus, to be reused among use cases.

To model the sequence of steps that take part in the dialogue process, each process step and system request is bound to a step function through the object property `hasStepFunction`. Considering their output when executed in the dialogue manager, each step function has a set of implications –modelled through the object property `implies` and its subproperties– that determine the next step to perform. The possible implications for step functions can be system steps, process steps, dialogue steps –described above–, or dialogue control markers (`DialogueControlMarker`), which are predefined dialogue actions that establish whether the dialogue process needs to *continue*, *finish* or *restart*.

4.3.1.2 TODODT - Dialogue Tracing Module

The TODODT module aims to model the necessary concepts to keep track of ongoing or finished dialogues. More specifically, this module has two main functions: to store the dialogue state and to gather the necessary information to recreate the dialogue, which can be exploited to detect the source of unsuccessful dialogues or to learn from completed interactions. For this, this module covers the following BCQs:

- **BCQ10**. Which step is currently being performed in the dialogue? (CQ85)

- **BCQ11**. Which is the trace –and the information that includes– for an element? (CQ86-99)

Dialogue processes are traced through the class `DialogueProcessTrace`. For each dialogue, which is modelled as a trace (`DialogueTrace`), a set of traces are stored through data properties that allow to characterize it: the starting time (`initDateTime`), the current step of the dialogue (`currentStep`) and whether the dialogue is currently ongoing or has finished (`status`). If the dialogue has come to an end, the time the dialogue finished is also stored (`finishDateTime`).

In task-oriented dialogue systems, the dialogue process revolves around an initial user request (`userRequestTrace`), which is associated to a dialogue trace through the object property `hasUserRequest`. For each user request, a trace of the transcription obtained for said request is stored (`text`).

For example, given a user interaction "Put the magenta cartridges into container 2":

1. A dialogue trace is created: `dialogueTrace001`.
2. For that dialogue trace, the starting time is assigned, along with the dialogue status and the current step of the dialogue:
   ```
   :dialogueTrace001 tododt:initDateTime
     '2021-10-26T21:32:52'.
   :dialogueTrace001 tododt:dialogueStatus 'open'.
   :dialogueTrace001 tododt:dialogueStep
     'ObtainKeyElements' .
   ```
3. A user request trace is created: `userRequestTrace001`, associated with the dialogue trace:
   ```
   :dialogueTrace001 tododt:hasUserRequest
     :userRequestTrace001.
   ```
4. For that user request trace, the transcription obtained is stored:
   ```
   :userRequestTrace001 tododt:text 'Put the magenta
     cartridges into container 2'.
   ```

If the task-oriented dialogue system has been able to obtain a full interpretation of the command, the dialogue will finish and, thus, the dialogue trace's status will be updated to reflect so:

```
:dialogueTrace001 tododt:finishDateTime
  '2021-10-26T21:35:42'.
:dialogueTrace001 tododt:dialogueStatus 'close'.
```

However, if the initial user request does not contain enough information or the task-oriented dialogue system is not able to fully interpret the command, an interaction with the user is necessary. These interactions initiated by the system and directed to the user are secondary dialogues (`SecondaryDialogueTrace`), which can be related to dialogue traces or to other secondary dialogues (`hasSecondary Dialogue`), depending on whether the extra user input is required during a dialogue or a secondary dialogue. In the same way as dialogues, for secondary dialogues starting and finishing time and status traces are stored.

As noted in the description of TODODM, interactions from the system to the user can be either responses or requests. For each secondary dialogue, a trace for each of these elements is generated (`SystemResponseTrace` and `SystemRequestTrace`, respectively).

Since the outputs for system responses and requests are designed parametrically, each trace contains the exact output provided to the user in the interaction (`text`), which can be complemented with a tag that stands for the language (e.g., `@en`, for English).

Finally, once the user has answered the request performed by the system, a trace is generated (`userResponseTrace`), which, in the same way as user requests, contains the transcription obtained from the user response through the data property `text`.

If the user response has provided the information requested, the secondary dialogue will update its status to 'close', in the same way as dialogues. However, if the information is still not enough, a new secondary dialogue will be generated.

For instance, given the user request "Put the magenta cartridges", which is missing information (the destination), and its dialogue trace `dialogueTrace002`:

1. A secondary dialogue trace, `secondaryDialogueTrace001`, is created and associated to the dialogue trace:
   ```
   :dialogueTrace002 tododt:hasSecondaryDialogue
      :secondaryDialogueTrace001 .
   ```

2. For that secondary dialogue trace, the starting time is assigned, along with its status:
```
:secondaryDialogueTrace001 tododt:initDateTime
   '2021-10-26T21:37:52' .
:secondaryDialogueTrace001 tododt:dialogueStatus
   'open' .
```

3. A trace for the system response and the system request are created: `systemResponseTrace001` and `systemRequestTrace001`.

4. Each system response and request traces include a trace of the output provided to the user with its corresponding language tag:
```
:systemResponseTrace001 tododt:text 'I am missing
  some information'@en .
:systemRequestTrace001 tododt:text 'Can you please
  tell me the destination?'@en .
```

5. When the user answers the system request, a user response trace is created: `userResponseTrace001`.

6. For that user response trace, the transcription obtained is stored, with its corresponding language tag:
```
:userRequestTrace001 tododt:text 'Into container
  2'@en .
```

7. If the information provided by the user covers the information requested, the secondary dialogue will close:
```
:secondaryDialogueTrace001 tododt:finishDateTime
  '2021-10-26T21:40:42'.
:secondaryDialogueTrace001 tododt:dialogueStatus
  'close' .
```

4.3.1.3 TODODial on Top of TODODM and TODODT

Although the submodules in TODODial can be used independently, in the context of their use as part of the core of task-oriented dialogue systems they are related to each other. In this sense, the TODODial module is in charge of joining TODODM and TODODT[3].

[3]For an overview of the classes and relations involved, Figure 4.2 may be consulted in page 57.

In this case, as described above, TODODT considers traces for system responses and requests, along with the output presented to the user. So as to keep track of the system response and/or request directed to the user, TODODial establishes a relation between system response and request traces in TODODT to their corresponding system responses and requests in TODODM through the object property `tododial:isTraceOf`.

For example, for a given system response trace `systemResponseTrace001` and a system resquest trace `systemRequestTrace001`:

```
:systemResponseTrace001 tododial:isTraceOf
  tododm:systemResponseMissingInfo.
:systemRequestTrace001 tododial:isTraceOf
  tododm:systemRequestMissingInfo.
```

4.3.2 TODODom - Domain Module

The TODODom module deals with the concepts used for domain modelling, in terms of the actions that the target system can perform and the elements that are present in the scenario. Most importantly, this module is in charge of supporting the process to obtain a command that can be executed by the target system from a natural interaction. To do so, it covers the requirements in the following BCQs:

- **BCQ01**. What are the elements that are present in the scenario? (CQ01-02)
- **BCQ02**. Which is the action to be performed by the target system given a series of key elements obtained from a user request? (CQ03-11)
- **BCQ03**. Which are the arguments of a specific action? (CQ12-13)
- **BCQ04**. Given a set of arguments from a specific action, to what argument can a key element from the user request be associated to? (CQ14-18)
- **BCQ05**. Which is the format of the information that has to be provided to the target system? (CQ19-26)

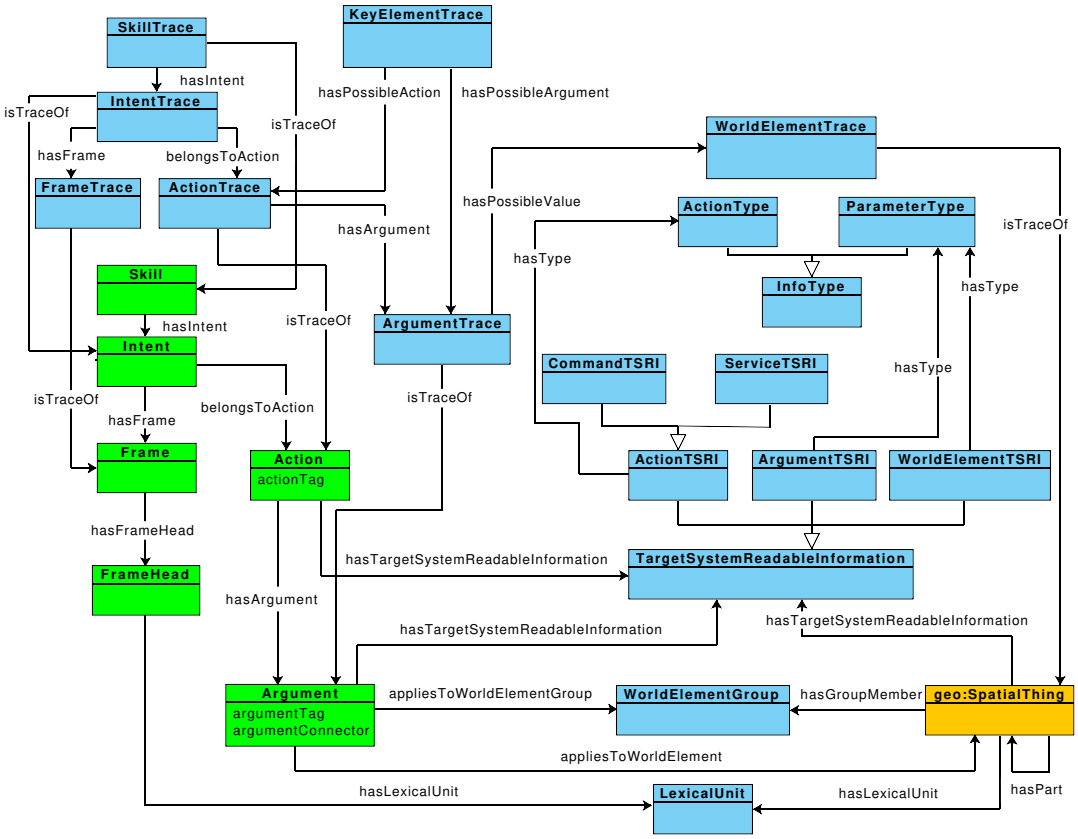


Figure 4.3: Simple overview of TODODom. Green classes correspond to TODODFA, the orange one to TODODW and the blue ones to TODODom.

Taking into account these considerations, this module has two submodules: TODO Domain Frame-Action (TODODFA), which deals with target system actions, and TODO Domain World (TODODW), which deals with scenario elements. The main classes in each module can be seen in Figure 4.3.

4.3.2.1 TODODFA - Domain Frame-Action Module

The TODODFA module is in charge of, on the one hand, modelling the actions that the target system can execute and the knowledge to obtain them from a user command and, on the other hand, to represent the necessary information for generating the output for the target system accordingly, dealing with the following BCQs:

- **BCQ02**. Which is the action to be performed by the target system given a series of key elements obtained from a user request? (CQ03-11)
- **BCQ03**. Which are the arguments of a specific action? (CQ12-13)

TODODFA considers the possible functionalities that the target system can perform as skills (`Skill`). For instance, an information system that can *manage maintenance tasks* and that also can *give assistance on a maintenance procedure* has two skills.

Each skill is related to one or more intents (`Intent`), which represent the user objective when directing a request to the task-oriented dialogue system, through the object property `hasIntent`. For example, for the information system mentioned above and its *maintenance task management* skill (`InfoSystem_ManageMaintenanceTask`), the user may have two intents: to show the blueprints (`showBlueprint Intent`) or a exploded view of the components (`showExplodedView Intent`) involved in the task:

```
:InfoSystem_ManageMaintenanceTask tododfa:hasIntent
  :showBlueprintIntent.
:InfoSystem_ManageMaintenanceTask tododfa:hasIntent
  :showExplodedViewIntent.
```

For each intent, an action (`Action`) is associated (`belongsToActi on`). Actions depict the functions that the target system can execute, which may be executed according to the value of specific parameters. These parameters are arguments (`Argument`), which are inspired in the concept of *slot* in GUS (Bobrow et al., 1977). Actions are related to arguments through the object property `hasArgument`, which can determine if an argument is strictly necessary for the action to execute or it is optional through its subproperties `hasCoreArgument` and `hasOptionalArgument`, respectively.

So as to provide support to determine the intent (and, thus, the action) from a natural command, intents are also related to one or more frames (`Frame`). These frames, which are inspired in Frame Semantics (Fillmore et al., 1976), model situations that can be elicited by specific words, the so-called frame heads (`FrameHead`). For example, the frame *Needing*, which represents a situation in which somebody needs something, may be elicited by a frame head *need*. The

reason for this modelling is that the situations depicted by frames represent the circumstances of an intent. For instance, for the intent `showBlueprintIntent`, the frame `Needing` may apply since this intent can be conceptualised as a situation in which a user *needs* a blueprint.

Finally, actions and arguments have a set of data properties in order to fill the corresponding parametric values in TODODM's system steps' output sentences. Both actions and arguments have a tag associated that identifies them in an output sentence (`actionTag` and `argumentTag`, respectively). Arguments may have a second data property, `argumentConnector`, which is a complementary word –generally a preposition– to introduce an argument tag so the output sentence constructed is grammatically correct and conveys the intended message. However, it is not always necessary, and it may have an empty value. Examples (12) and (13) show a usage instance of these data properties in an output sentence:

(12) ¿Quieres {`ActionTag`} {`ArgumentConnector`} {`ArgumentTag`} {`ArgumentValue`} {`ArgumentConnector`} {`ArgumentTag`} {`ArgumentValue`}?
Do you want me {`ActionTag`} {`ArgumentConnector`} {`ArgumentTag`} {`ArgumentValue`} {`ArgumentConnector`} {`ArgumentTag`} {`ArgumentValue`}?

(13) ¿Quieres {que coja y ponga} {} {los cartuchos del tipo} {amarillo} {en} {el contenedor} {1}?
Do you want me {to pick and place} {} {the cartridges that are} {yellow} {in} {container} {1}?

4.3.2.2 TODODW - Domain World Module

The TODODW module models the set of elements that are available in the scenario, covering the requirements in BCQ01:

- **BCQ01**. What are the elements that are present in the scenario? (CQ01-02)

Examples of the scenario elements –or world elements– to model are spaces, objects or people, depending on the use case. For this,

the WGS84 Geo Positioning ontology[4] has been reused. Given the strong dependence of this module on the scenario, this module is practically empty out-of-the-box, with an only class, `geo:SpatialThing`, and the relations `dcterms:hasPart` and its inverse `dcterms:hasPartOf`[5], that aim to model the world elements contained by others; for instance, a laboratory that has machines in it:

```
:laboratory1 dcterms:hasPart :machineA .
:machineA dcterms:isPartOf :laboratory1 .
```

This is a module, thus, that must be customized according to the characteristics of each adaptation and/or application. To do so, the LOT methodology (Poveda-Villalón et al., 2019) should be applied.

4.3.2.3 TODODom on Top of TODODFA and TODODW

TODODom's modules, although they can work independently, have been designed to interact with each other to comprehensively characterize the domain of a particular use case. For this, the role of the TODODom module is to connect these two submodules through a series of intermediate classes[6] that deal with the following BCQs:

- **BCQ04**. Given a set of arguments from a specific action, to what argument can a key element from the user request be associated to? (CQ14-18)
- **BCQ05**. Which is the format of the information that has to be provided to the target system? (CQ19-26)
- **BCQ11**. Which is the trace –and the information that includes– for an element? (CQ86-99)

In the process of modelling arguments, it is important to determine which world elements can be values of a given argument. For this, similar world elements may be grouped (`WorldElementGroup`), through the object property `hasGroupMember`. So as to relate arguments to world element groups, the object property `appliesToWorld`

[4] `http://www.w3.org/2003/01/geo/wgs84_pos`

[5] `http://purl.org/dc/terms/`

[6] For an overview of the classes and relations involved, Figure 4.3 may be consulted in page 64.

`ElementGroup` is used. For instance, considering an action `logistics Robot_move`, in which a logistics robot has to move an object to some target destination and has the core arguments `object` and `destination`. In this case, the world elements that can be values of `object` have a technical restriction that states that these elements must be in a box. For this, the elements that are in a box may be modelled as part of a group, `group_inBox`, to be matched to the argument:

```
:logisticsRobot_move tododfa:hasCoreArgument :object.
:WorldElementInBox1 a tododw:Object.
:WorldElementInBox2 a tododw:Object.
:WorldElementNotInBox a tododw:Object.
:group_inBox tododom:hasGroupMember :WorldElementInBox1.
:group_inBox tododom:hasGroupMember :WorldElementInBox2.
:object tododom:appliesToWorldElementGroup :group_inBox.
```

In order for the target system to execute a command obtained through user interaction, it is necessary to establish the format in which the information obtained must be sent to the target system (`TargetSystemReadableInformation` –or TSRI). In this sense, this information must be modelled for the action to execute, the parameters (arguments) for that execution and the values of these parameters (world elements). For actions, so as to determine the information necessary for their execution, according to how the action is implemented in the target system –e.g., as a robot command or as a REST service–, the class `ActionTSRI` has been modelled. Since in industrial scenarios the most common implementations are robot commands and REST services, `ActionTSRI` has two subclasses: `CommandTSRI` and `ServiceTSRI`. In the case of arguments and world elements, it is possible that the same world element may be the value of different arguments of different actions. Even more, each action may require a different format (a `ParameterType`) for the same world element. For example, for actions A and B, that involve a machine as the value of one of its arguments: action A's `pieceOfInformation` argument may require the machine to be identified through an identifier (e.g., "Machine123") and action B's `destination` argument may need its coordinates (e.g., "XYZ"). So as to provide the target system with the correct format, the TSRIs for both arguments (`ArgumentTSRI`) and world elements (`WorldElementTSRI`) are related to a member of the class `ParameterType` through the object property `hasType`. By doing so, the argument can set the parameter type required by

the action when a given world element is assigned as its value. This example is illustrated below:

```
:machineA tododom:hasTargetSystemReadableInfo
  :coord_machineA .
:coord_machineA geo:lat "1" .
:coord_machineA geo:long "2" .
:coord_machineA geo:alt "1" .
:coord_machineA tododom:hasType :coordinate .

:machineA tododom:hasTargetSystemReadableInfo
  :ID_machineA .
:ID_machineA tododom:val "123" .
:ID_machineA tododom:hasType :ID .

:actionA tododfa:hasArgument :pieceOfInformation .
:pieceOfInformation tododom:hasType :ID .

:actionB tododfa:hasArgument :destination .
:destination tododom:hasType :coordinate .
```

Thus:
Action A: `pieceOfInformation` when the value is `machineA` → "123".
Action B: `destination` when the value is `machineA` → "1, 2, 1".

Another common characteristic shared between specific individuals in TODODFA –more specifically, the ones that belong to the class `FrameHead`– and TODODW –world elements– is that both elements are explicitly referred to in the user command. Considering the fact that TODO has been designed to cope with natural language, it is important to take into account that users can refer to the same actions or elements in different ways –for instance, to refer to a *computer*, the words *PC* and *computer* may be used. For this, the different word variants in which a user can refer to them in a natural command are modelled as individuals of the `LexicalUnit` class. For each frame head or world element, there is a set of lexical units associated through the object property `hasLexicalUnit`, which cover the different variants or synonyms to refer to them, considering different languages to support multilingualism. The following example determines the possible variants for `computer` (*PC*, *computer* and *laptop*):

```
:PC.N a tododom:LexicalUnit .
:computer.N a tododom:LexicalUnit .
:laptop.N a tododom:LexicalUnit .
:computer tododom:hasLexicalUnit :PC.N .
:computer tododom:hasLexicalUnit :computer.N .
:computer tododom:hasLexicalUnit :laptop.N .
```

As a side note, as this modelling implies that a considerable amount of data needs to be instantiated, Section 4.6 provides a strategy that leverages existing multilingual resources to semi-automatically instantiate this information.

Finally, TODODom also models the necessary classes and object properties to store traces for actions, intents, arguments, the key elements extracted from a user input, frames, skills and world elements. These traces are related to their corresponding class individuals in TODODFA and TODODW through the object property `isTraceOf`. Furthermore, these traces relate with each other in the same way as their corresponding classes. For instance:

```
:logisticsRobot_move a tododfa:Action.
:object a tododom:Argument.
:logisticsRobot_move tododfa:hasCoreArgument :object.

:logisticsRobot_move_trace a tododom:ActionTrace.
:logisticsRobot_move_trace tododom:isTraceOf
  :logisticsRobotmove.
:object_trace a tododom:ArgumentTrace.
:object_trace tododom:isTraceOf :object.
:logisticsRobot_move_trace tododfa:hasCoreArgument
  :object_trace.
```

These traces have the objective of storing the different interpretations that a user command may have. That is, it is possible that, given a key element (stored as key element trace), it may apply to more than one action, to more than an argument or more than an argument value. For example, if the target system is able to, among other actions, show a list of tools and additional information for a given manufacturing procedure and the user utters the command "Show me more", the possible actions to be performed are to show the list of tools and to show additional information. To con-

vey this information, key element traces are related to the different possible action or argument interpretation options that may be obtained from them through the object property `hasPossible` and their subproperties `hasPossibleAction`, `hasPossibleArgument`, and argument traces and their possible values through the subproperty `hasPossibleValue`.

4.3.3 TODO on Top of TODODial and TODODom

The previous sections have depicted the modelling of the dialogue process through the domain and dialogue dimensions through the TODODom and TODODial modules and their submodules. Although both dimensions are independent, to obtain a full modelling of the dialogue process there is knowledge from both modules that should be related to each other: traces.

As described previously, TODODial, among others, stores traces for dialogues and secondary dialogues and user requests and responses, whereas TODODom stores the domain traces obtained from a user command (both requests and responses). The main objective of TODO regarding traces is that the domain information generated in the context of a dialogue is stored so a given dialogue process can be easily recreated from its traces. In this sense, the knowledge required is, on the one hand, the key elements obtained from each user interaction and the domain information obtained from them (e.g., arguments, world elements) and, on the other hand, at which point of the dialogue was this information obtained (e.g., a secondary dialogue).

To obtain the intended dialogue recreation, given a dialogue initiated by a user request:

1. A user request trace is generated:
   `:userRequest001 a tododial:UserRequestTrace .`

2. The user request trace is associated to the key element trace generated with its key elements by using the object property `hasKeyElement`:

   `:keyElementTrace001 a tododom:KeyElementTrace .`

```
:userRequest001 tododom:hasKeyElement
  :keyElementTrace001 .
```

3. When an action is obtained, the user request trace is related to the action trace generated through the object property `todo:hasActionTrace`. This action trace will be related to its corresponding argument traces and their values –as world element traces– as soon as they have been obtained in the dialogue:
```
:userRequest001 todo:hasActionTrace :actionTrace001 .
:actionTrace001 tododfa:hasCoreArgument
  :argumentTrace001 .
:argumentTrace001 tododom:hasValue
  :worldElementTrace001 .
[...]
```

4. If there is information missing in the initial user request and a secondary dialogue is necessary, the obtained information will be related to the secondary dialogue used to obtain it though the object property `todo:obtainedThrough`:
```
:worldElementTrace001 todo:obtainedThrough
  :secondaryDialogue001.
```

The relations between TODO's modules can be seen in Figure 4.1, in page 55.

4.4 Evaluation

According to Poveda-Villalón (2016), there are several metrics in the literature to perform ontology evaluation and validation. However, most of these metrics evaluate ontologies from a structural point of view (e.g., whether there are inconsistencies or ill-formed data) or subject-related data (e.g., whether the information in the ontology is correct or the domain is fully covered). In the first case, although relevant, these metrics may not be descriptive enough to properly assess the quality of an ontology (Esnaola-González et al., 2021), whereas in the second case, the metrics must rely on other sources (e.g., gold standards) in order to perform evaluation (Raad & Cruz, 2015), which in some cases is not viable due to the nature of the ontology to evaluate.

Furthermore, and as noted by Poveda-Villalón, Espinoza-Arias, Garijo, and Corcho (2020), ontologies, as other data resources, should follow the **FAIR** –Findable, Accessible, Interoperable, Reusable– data principles (Wilkinson et al., 2016) to ensure optimal reusability conditions, which is one of the main objectives of their development in the context of the Semantic Web.

Following the considerations above and the approach and tools described in Esnaola-González et al. (2021), the following sections present an ontology evaluation framework based on four points of view (**structural metrics**, **design correctness**, **adherence to FAIR principles** and **modularity quality**), that aim to provide a non-biased ontology evaluation. Some discussion is also provided on the possibility of customising the ontology through module modification[7].

4.4.1 Structural Metrics

This evaluation approach aims to provide figures describing the data modelled in the ontology, more than assessing its overall quality (Esnaola-González et al., 2021).

The source to obtain ontology structural information has been Protégé, in its version 5.2.0. More specifically, its Ontology Metrics tab has been used, which includes descriptive data from the ontology in terms of general metrics –*Metrics*– (number of classes, objects, data properties, individuals, etc.) and axioms for classes (e.g., number of subclasses), object properties (e.g., number of inverse object properties), data properties (e.g., number of disjoint data properties), individuals (e.g., number of equivalent individuals) and annotations (e.g., information related to domains and ranges).

The most relevant information in the *Metrics* tab is provided in Table 4.1. Apart from offering quantitative measures (e.g., number of axioms, classes and properties –object properties (OP) and data properties (DP)), Table 4.1 also gives information about the expressivity of the ontology at hand. Since TODO's modules aim to model complex relationships between concepts, the ontologies show a con-

[7]The evaluations have been performed in the 2.1 release of TODO: `https://github.com/cristinacm/todo/releases/tag/2.1`

Table 4.1: Structural metrics obtained through Protégé's Ontology Metrics tab. Values in parentheses do not consider imported modules. OP: object properties; DP: data properties; Ann: annotations; DL Expr: Description Logics expressivity.

Ontology	**Axioms**	**Class**	**OP**	**DP**	**Ann**	**DL Expr**
TODODial	826(31)	73(0)	23(3)	13(0)	19	$\mathcal{SHIQ}(\mathcal{D})$
TODODM	588	59	15	4	19	$\mathcal{ALCHI}(\mathcal{D})$
TODODT	207	14	5	9	19	$\mathcal{SHQ}(\mathcal{D})$
TODODom	695(469)	37(28)	50(30)	20(17)	19	$\mathcal{ALCHIQ}(\mathcal{D})$
TODODFA	208	8	18	3	19	$\mathcal{ALCHIQ}(\mathcal{D})$
TODODW	18	1	2	0	16	$\mathcal{ALI}$

siderably rich expressivity[8]. However, TODODW is not as rich as the other modules, which makes sense taking into consideration its out-of-the-box simplicity.

Regarding size, modules in TODO are relatively small (between 1 and 59 classes). As for the highest number of classes, TODODM presents 59 classes since, among other elements, it models all types of interactions –requests and responses– that can be made by the system (that is, `tododm:SystemSteps`[9]). At the other end, TODODW only presents a single class, again, due to the fact that it is highly dependant on the target use case and that it must be modelled according to the characteristics of the scenario.

4.4.2 Design Correctness Metrics

So as to assess the design of the ontologies presented in this work, the tool OOPS! (Poveda-Villalón, Gómez-Pérez, & Suárez-Figueroa, 2014) has been used. This tool checks the ontology to evaluate against a set of 41 pitfalls, which are considered to be the most common ones in ontology design. These pitfalls are also classified according to three levels of importance (Poveda-Villalón et al., 2014):

- **Critical** (C): It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability,

[8]The meaning of the naming conventions used in description logics for expressivity can be found in Chapter 2.2.2.

[9]*vid.* Chapter 4.3.1.1.

Table 4.2: Results of the evaluation on design correctness performed by OOPS!

Ontology	**M**	**I**	**C**	**Notes**
TODODial	1	1	0	P11 (I), P13 (M)
TODODM	2	1	0	P11 (I), P13 (M), P22 (M)
TODODT	3	1	0	P04 (M), P11 (I), P13 (M), P22 (M)
TODODom	4	1	0	P04 (M), P08 (M), P11 (I), P13 (M), P22 (M)
TODODFA	0	1	0	P11 (I)
TODODW	2	1	0	P04 (M), P08 (M), P11 (I)

etc.

- **Important** (I): Though not critical for the ontology to function, it is important to correct this type of pitfall.
- **Minor** (M): It is not really a problem, but by correcting it the ontology will be nicer.

Table 4.2 shows the results obtained by OOPS! for each of the modules of TODO in terms of number of pitfalls according to their importance level. First of all, it is worth noting that no critical pitfalls have been reported in any of TODO's modules.

For the important pitfalls, the Table shows that the one that all modules share is **P11**, which states that ***properties lack the modelling of domain and range***. As the function of domain and range is to make inferences regarding the class of the elements related by a property, rather than setting restrictions (i.e., raise inconsistencies if domain or range are violated), in the scope of this work this modelling would be problematic (undesired inferences may occur). To illustrate this with an example, given the property `contains` and its domain `Laboratories` and range `Machines`, it will be inferred that the subject of the triples of the property `contains` belong to the class `Laboratories` and the objects to the class `Machines`. That is, given the triple `X contains Y`, it will be inferred that `X` is an instance of `Laboratories` and `Y` an instance of `Machines`. However, if the individual `A` belongs to the class `Machines` and individual `B` to `Laboratories` and the triple `A contains B` exists, the reasoner, instead of raising an inconsistency (according to the ontology

modelling, `Laboratories` contain `Machines` and not the other way around), would provoke that `A` would belong to two classes: the class `Machines` (its modelled class) and the class `Laboratories` (by inference), and the same would happen with `B` (i.e., `Laboratories` by modelling and `Machines` by inference). This is an undesired outcome and would not be useful for the purpose of the ontologies. Nevertheless, this can be solved by modelling these classes as disjoint, and in that case the reasoner would raise and inconsistency given this situation. Since this action is not critical –although desirable–, the correction of this pitfall has been envisioned for future versions of TODO.

For the minor pitfalls, the pitfalls reported are **P04**, **P08**, **P13** and **P22**. **P04** indicates that *there are elements that are isolated* in the ontology and, thus, are not connected to other ontology elements. As it can be observed, this pitfall occurs in modules that are connected to others (e.g., TODODom and TODODial being connected through action traces). In this sense, these isolated elements are in fact connected to other elements outside the module. For **P08**, this pitfall arises when any *class/property lacks some annotation* (e.g., description, label). In these ontologies, this pitfall refers to imported classes and properties, the descriptions and labels of which cannot be detected by the tool and are out of the scope of TODO's modelling. On the other hand, **P13** points out the *lack of inverse object properties*. Considering that in dialogue-oriented scenarios it is crucial to obtain responses in the lowest time possible and inverse relationships may increase reasoning time, the number of inverse relationships in TODO has been reduced to the strictly necessary ones. For that reason, the object properties detected in this pitfall do not have an inverse relationship. Finally, **P22** states that *name conventions are not correctly followed*. Since this pitfall does not indicate the specific elements to be reviewed, all the affected modules have been manually reviewed to ensure that all classes and properties follow the same naming conventions.

All in all, the considerations above prove that, according to OOPS!, the modules in TODO could be considered as correctly designed.

4.4.3 Adherence to FAIR Principles

This section analyses whether TODO and its modules comply with the FAIR principles (Wilkinson et al., 2016). In a nutshell, FAIR principles establish a set of criteria to, on the one hand, support the reuse of data and non-data (e.g., formal representations of workflows (Wilkinson et al., 2016)) resources and, on the other hand, to allow machines to find and consume these resources. These criteria are grouped into four dimensions –**F**indable, **A**ccessible, **I**nteroperable and **R**eusable–, and each of them have a series of characteristics to satisfy, which will be further described in the following lines.

To determine the adherence of TODO and its modules to the FAIR principles, the tool FOOPS! (Garijo, Corcho, & Poveda-Villalón, 2021), is used. To do so, this tool checks whether the ontology to analyse complies with the FAIR principles according to 24 items, distributed across its four dimensions, and their description by Wilkinson et al. (2016) (reproduced from Poveda-Villalón et al. (2020)):

- To be **Findable** (9 items in FOOPS!). Characteristics:
  - (Meta)Data are assigned a globally unique and persistent identifier.
  - Data is described with rich metadata.
  - Metadata clearly and explicitly include the identifier of the data it describes.
  - (Meta)Data are registered or indexed in a searchable resource.
- To be **Accessible** (3 items in FOOPS!). Characteristics:
  - (Meta)Data are retrievable by their identifier using a standardized communications protocol.
  - The protocol is open, free, and universally implementable.
  - The protocol allows for an authentication and authorization procedure, where necessary.
  - Metadata are accessible, even when the data are no longer available.[10]

[10]Not analysed in FOOPS!

Table 4.3: Results on adherence to FAIR principles, obtained by FOOPS!

	***F*indable**	***A*ccess.**	***I*nteroper.**	***R*eusable**	**Overall**
TODO	6.83/9	3/3	3/3	6.25/9	80%
TODODial	5.83/9	3/3	3/3	7.63/9	81%
TODODT	5.83/9	3/3	2/3	7.63/9	77%
TODODM	5.83/9	3/3	2/3	7.72/9	77%
TODODom	5.83/9	3/3	3/3	7.63/9	81%
TODODFA	5.83/9	3/3	2/3	7.77/9	78%
TODODW	5.83/9	3/3	3/3	5.63/9	73%

- To be **Interoperable** (3 items in FOOPS!). Characteristics:
  - (Meta)Data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
  - (Meta)Data use vocabularies that follow FAIR principles.
  - (Meta)Data include qualified references to other (meta) data.[10]
- To be **Reusable** (9 items in FOOPS!). Characteristics:
  - (Meta)Data are richly described with a plurality of accurate and relevant attributes.
  - (Meta)Data are released with a clear and accessible data usage license.
  - (Meta)Data are associated with detailed provenance.
  - (Meta)Data meet domain-relevant community standards.[10]

For each of the items to analyse, FOOPS! returns the degree of compliance of an item for the ontology at hand and, considering this, a score is obtained for each FAIR dimension. This information is shown in Table 4.3, which also adds the overall score across all dimensions for each ontology. These results show that TODO and all its modules are fully accessible and most of them fully interoperable –as it will be commented later on in this section–, according to the criteria established. Overall, and also considering the rest of dimensions, all modules achieve scores of more than 73%, what indicates that TODO modules are of good quality in terms of their adherence to FAIR principles.

Table 4.4: Distribution of pitfalls found by FOOPS!

	***F*indable**	***A*ccess.**	***I*nterop.**	***R*eusable**
TODO	VER1, VER2, OM1	N/A	N/A	OM2, OM3, VOC3, VOC4, OM5_2
TODODial	VER1, VER2, OM1, FIND2	N/A	N/A	OM2, OM3, OM5_2
TODODT	VER1, VER2, OM1, FIND2	N/A	VOC2	OM2, OM3, OM5_2
TODODM	VER1, VER2, OM1, FIND2	N/A	VOC2	OM2, OM3, OM5_2
TODODom	VER1, VER2, OM1, FIND2	N/A	N/A	OM2, OM3, OM5_2
TODODFA	VER1, VER2, OM1, FIND2	N/A	VOC2	OM2, OM3, OM5_2
TODODW	VER1, VER2, OM1, FIND2	N/A	N/A	OM2, OM3, VOC3, VOC4, OM5_2

FOOPS! also reports the FAIR non-compliant items –pitfalls from now on– detected for each FAIR dimension and analysed ontology, respectively, which can be seen in Table 4.4. For TODO, it can be observed that the pitfalls detected are common among modules:

- For the **Findable** dimension, the VER1, VER2 and OM1 pitfalls observed in the TODO modules all refer to the lack of a version IRI for the ontology. In this case, previous versions of the ontology can be found in the Github repository as releases[11]. To solve this pitfall, further versions of TODO will include a version IRI.
- For the **Interoperable** dimension, TODODT, TODODM and TODODFA present the VOC2 pitfall, which determines that no terms or vocabularies were reused. In these cases, the reason to not reuse this type of resources was that no terms or vocabularies were found to include them as part of their modelling.
- Finally, for the **Reusable** dimension, all the modules analysed share the OM2, OM3 and OM5_2 pitfalls. For the OM2 pitfall, it determines that the modules lack a citation, which is a pitfall that can be solved in further versions. For the OM3 and OM5_2

[11] `https://github.com/cristinacm/todo/releases`

ones, they establish that a set of optional metadata is missing. Since the presence of these metadata is not deemed as necessary, their addition will be considered in further versions, but not as a critical action. Besides the pitfalls mentioned above, the VOC3 and VOC4 ones (related to the fact of not having labels and descriptions for classes, respectively) have also been found in TODODW and TODO. For the former, this pitfall is triggered since it does not have classes of its own. For the latter, the classes and object properties involved come from other imported ontologies which do have labels and definitions for them, but are not detected by FOOPS!.

Considering the overall results obtained and the remarks above regarding the pitfalls detected by FOOPS!, it can be determined that TODO and its modules widely comply with FAIR principles.

4.4.4 Modularity Quality

Considering that TODO is a modular ontology, it is of special relevance to evaluate the quality of each of the ontology modules. For this, the approach in Khan and Keet (2016) is considered to provide a set of comprehensive measures that determine the quality of an ontology module. This work proposes a set of 14 different module types, to be used depending on the final usage of the evaluated ontology module. For each type, the authors provide a set of reference metrics and values to evaluate if the ontology module is of high quality. In regard to TODO, all modules belong to the T2 type –*Subject domain modules*.

For the T2 modules, reference values for the most relevant metrics are **small cohesion** (i.e., "the extent to which entities in a module are related to each other" (Khan & Keet, 2016)), **coupling** (i.e., the degree in which the concepts in a module are related to concepts in other modules) and **redundancy** (i.e., "the duplication of axioms within a set of ontology modules" (Khan & Keet, 2016)) and **large encapsulation** (i.e., whether a module can be easily replaced by another or modified without side effects). Other relevant metrics are **size** and **number of axioms**, among others, which can be seen in Table 4.5.

To evaluate the quality of the modules in TODO, the Tool for Ontology Modularity Metrics (TOMM) (Khan & Keet, 2016) has been used. For that matter, three evaluations have been performed considering the modular hierarchy in TODO, and each submodule has been evaluated in relation to its supermodule, going up one level. Thus, the evaluations performed are the following:

1. The quality of TODODFA and TODODW in relation to TODODom.
2. The quality of TODODM and TODODT in relation to TODODial.
3. The quality of TODODial (including TODODT and TODODM) and TODODom (including TODODFA and TODODW) in relation to TODO.

The results obtained from TOMM are included in Table 4.5, which includes the relevant metrics to the T2 module type, separated into the most relevant ones and the rest Khan and Keet (2016), along with the score obtained for each of them. For each metric, the reference value for T2 modules (Ref. value) is also reported.

At a first glance, some anomalies in the results in Table 4.5 can be seen. On the one hand, there are some values for appropriateness with an asterisk, the original value of which obtained by TOMM, in fact, was *-1.0*. This is due to the fact that the approximation to obtain appropriateness implemented in TOMM establishes that modules should contain a number of axioms within the range of 0-500 (Keet, 2018). If the number is higher, TOMM sets a default value *-1.0*. To obtain appropriateness values without considering the number of axioms restrictions (the values in the table), the following formula for appropriateness in Keet (2018) has been applied:

$$Appropriate(x) = \frac{1}{2} - \frac{1}{2}cos(x\frac{\pi}{250}) \tag{4.1}$$

The next anomaly can be observed for inheritance richness, in which the modules in TODODom achieve a *NaN* score. Since this metric considers the class structure of the ontology to evaluate and these modules do not have any subclasses modelled, it is logical to

Table 4.5: Results of the evaluation on modularity quality. Values with asterisks refer to values obtained manually. *Ref. value* corresponds to the reference values for T2-type modules (Khan & Keet, 2016).

Metrics	**TODODom**		**TODODial**		**TODO**		**Ref. value**
Most Relevant	**DFA**	**DW**	**DM**	**DT**	**Dial**	**Dom**	
Cohesion	0.01	0.00	0.07	0.03	0.04	0.00	*0-0.25*
Encapsulation	0.97	0.62	0.99	0.96	0.99	0.99	*0.75-1*
Coupling	0.00	0.00	0.00	0.00	0.00	0.00	*0-0.25*
Redundancy	0.06	0.06	0.02	0.02	0.10	0.01	*0-0.25*
Others							
Size	29	3	78	28	109	107	*10-1103*
Number of axioms	208	17	588	207	796	650	*46-3954*
Appropriateness	0.93	0.01	0.28*	0.93	0.92*	0.65*	*0.51-0.75*
Atomic size	3.97	1.67	4.09	4.14	4.04	5.26	*3.42-7.66*
Intramodule distance	8	0	4115	74	4189	80	*0-340833*
Attribute richness	3.75	0.00	1.14	2.14	1.33	4.41	*0-3.44*
Inheritance richness	NaN	NaN	6.00	2.20	4.64	2.00	*1-6.44*

assume that this is the reason that these modules do not have a numerical score.

For the rest of results in Table 4.5, for TODODom-related metrics, TODODFA achieves the reference values in general, so it can be stated that it is a module of high quality. It is worth mentioning the small cohesion values which, as stated at the beginning of this chapter, account for the quality of the modularity in TODO, as, according to this metric, its modules are practically independent from each other.

In the case of TODODW, size-related metrics (size, number of actions, appropriateness and atomic size) do not fit the defined reference values. This is due to the fact that this ontology is intended to be expanded according to the use case and, thus, it has a reduced size. In the case of TODODial- and TODO- related metrics, the results show that in both cases the implicated modules widely satisfy

the established criteria.

All in all, the results in Table 4.5 show that TODO's modules satisfy the most relevant aspects to assess the quality of T2 modules. Therefore, and also considering the rest of metrics, it can be stated that TODO's modules are of high quality.

4.4.5 Ontology Customization by Module Modification

Among the advantages of the generality and modularity of TODO, there is the fact that it is possible to easily modify the ontology by means of specific parts of each module, according to the requirements of the use case. In general, these modifications may be performed in TODODom and its submodules, since it is the module that is inherently linked to the use case, leaving the rest of modules practically the same. The clearest example of this customisation is TODODW, which, as noted in Section 4.3.2.2, must be extended taking into account the scenario the dialogue system will be implemented in, which will define the classes and further relations between them[12]. However, any module from TODO can be easily modified.

4.5 Documentation and Publication

One of the main premises on ontology engineering methodologies (including LOT) is the concept of ontologies as resources to be reused. For that purpose, the creation of high quality documentation –that will result in a better understanding by potential users of the contents of the ontology and, eventually, in its proper reuse–, is strongly encouraged (Peroni, Shotton, & Vitali, 2012). Due to this, there are several intents in the literature at developing tools that ease the process of providing documentation for ontologies (Garijo, 2017).

To generate TODO's documentation, WIDOCO (a WIzard for DOCumenting Ontologies) (Garijo, 2017) has been used. This tool creates enriched documentation for ontologies in HTML format and

[12] *vid.* use cases in Chapter 6.

provides human-readable descriptions of the ontology from its metadata. Furthermore, it allows further extension of its contents by users.

As W3C's Data on the Web Best Practices (Farias Lóscio, Burle, & Calegari, 2017) state, use of appropriate metadata is a fundamental requirement for data that is to be published on the Web to make it readable and understandable by both humans and machines, along with providing relevant descriptive data about the resource at hand.

The guidelines for complete vocabulary metadata from Garijo and Poveda-Villalón[13] have been used to define the metadata to annotate TODO and its modules, as they consider them the most complete for this matter after reviewing other guidelines. The set of properties used to annotate TODO, and obtained from the aforementioned guidelines, are imported from the following vocabularies:

owl	<http://www.w3.org/2002/07/owl#>
rdfs	<http://www.w3.org/2000/01/rdf-schema#>
skos	<http://www.w3.org/2004/02/skos/core#>
vs	<http://www.w3.org/2003/06/sw-vocab-status/ns#>
dcterms	<http://purl.org/dc/terms/>
vann	<http://purl.org/vocab/vann/>

Moreover, Garijo and Poveda-Villalón[13] also classify metadata in terms of whether their use in ontology annotation is strongly advised (**recommended**) or encouraged but elective (**optional**). Considering this, each of the modules in TODO include most recommended metadata.

As for overall ontology metadata, the metadata used to annotate TODO is the following:

[13] `https://w3id.org/widoco/bestPractices`

vann:preferredNamespaceUri	The ontology's main URI.
vann:preferredNamespacePrefix	The ontology's prefix.
dcterms:title	The ontology's title.
dcterms:description	A description of the ontology.
owl:versionInfo	Ontology's version information.
dcterms:created	Creation date of the ontology.
dcterms:modified	Modification date of the ontology. (optional)
dcterms:creator	People who created the ontology.
dcterms:contributor	People who have contributed to the development of the ontology.
cc:license	The ontology's license.

In addition, the following metadata for classes, properties and data properties is included in TODO:

rdfs:label	The term's human-readable label
rdfs:comment	The term's definition
rdfs:isDefinedBy	The source used to define the term
skos:example [**optional**]	Usage examples of the term
vs:term_status [**optional**]	Usage status of the term (*stable*, *unstable*, *testing* and *archaic*)

To provide TODO and its modules with stable and secure URIs, the re-direction service `w3id.org` has been used[14]. The URIs for each of the modules in TODO are the following:

- TODO: `https://w3id.org/todo`
- TODODial: `https://w3id.org/todo/tododial`
- TODODM: `https://w3id.org/todo/tododm`
- TODODT: `https://w3id.org/todo/tododt`
- TODODom: `https://w3id.org/todo/tododom`
- TODODFA: `https://w3id.org/todo/tododfa`

[14] `https://w3id.org`

- TODODW: `https://w3id.org/todo/tododw`

Furthermore, TODO and its documentation can be accessible through the Linked Open Vocabularies (LOV) (Vandenbussche, Atemezing, Poveda-Villalón, & Vatant, 2017) vocabulary catalogue.

Finally, TODO's source files are also available in TODO's GitHub repository: `https://github.com/cristinacm/todo`

4.6 Semi-Automatic Population of Intent-Relevant Information

To contribute to an easy adaptation of semantic-based task-oriented dialogue systems, this Section describes a generic strategy that allows to semi-automatically populate their domain ontology –in the context of this work it would be TODODom– with the relevant information to identify user intents from natural language requests.

4.6.1 Strategy for Semi-Automatic Population of Intent-Relevant Information

Communication based on natural language can use a wide variety of words and expressions to convey the same meaning. Therefore, a dialogue system that interacts with users must be able to recognise different words that evoke the same situation or refer to the same entity to perform successful intent detection[15]. In an ontology-based approach, such as the one proposed in this thesis, it is very important that the maximum number of these possibilities is considered to populate the domain ontology with this information. Doing so manually would be a high time- and resource-consuming task, and the tendency is to apply semi-automatic or automatic techniques (Benabbas, Hornig, & Nicklas, 2018; Makki, 2017). Motivated by this, the strategy presented in this Section aims to overcome the challenge of instantiating the domain ontology with intents and their trigger words by semi-automatically exploiting existing multilingual lexical

[15]For instance, the verbs *inform* and *tell* may elicit the *give information* intent.

resources, without reducing the quality of the results and decreasing time and costs in the adaptation to different scenarios, applications and languages.

The core linguistic resource used for collecting the necessary information for the ontology population process is Predicate Matrix (PM), since it integrates and maps, among others, the information contained in FrameNet, the Suggested Upper Merged Ontology (SUMO) and the Multilingual Central Repository (MCR)–the resources that include most of the relevant data for this work[16].

The strategy will guide the ontologist through the selection of the relevant information within PM, and will gather, semi-automatically, the relevant intents (through frames), the associated trigger words (through lexical units (LUs)) and their corresponding synonyms for a specific use case in which the dialogue will take place. Considering this, the main steps of the strategy are the following:

0. **Use Case Characterisation**. In this step, the main characteristics of the use case are identified, such as the the different situations (associated to intents) that surround it or the target language.

1. **Linguistic-Resource-Driven Data Selection**. For the given situations identified for the use case, a data selection from linguistic resources is performed, following three phases: **frame selection**, **lexical unit selection** and **semantic extension and filtering**.

2. **Automatic Data Gathering for Intents**. Considering the selected data, the relevant knowledge to the intents is automatically gathered and structured in terms of the intents, their corresponding frames and the words that evoke them that are relevant to the use case. Up to this point, the gathered information is agnostic to the structure of the target domain ontology.

3. **Query Generation and Ontology Population**. Once the relevant intent-related information is automatically obtained, the query for the final population of the target domain ontology will be generated.

[16]The description of each of these resources can be found in Chapter 2.3.2.

The following sections detail the aforementioned steps for a guide robot use case for interactions in Spanish, in which the situations involved are to be taken to a given place, to give directions, or just to present information.

4.6.1.1 Linguistic-Resource-Driven Data Selection

The goal of this first main step is to obtain the necessary information that will serve as input for the automatic data gathering step. This data will be selected according to the situations identified for the use case.

Frame Selection

The first stage in the process is to select the most suitable frames for the previously defined events in the use case. For this step, the ontology engineer will choose the relevant frames for the use case from the list of available FrameNet frames in PM.

For instance, in a guidance scenario, one frame to be chosen would be *Motion*, that represents the movement from one point to another.

Lexical Unit Selection

For each selected frame, a set of LUs in English are automatically extracted from PM and presented to the ontology engineer, who will select the relevant LUs according to the use case. This step is necessary because it is not evident that all LUs are suitable to all the cases the chosen frame may apply to.

To illustrate this with an example, the frame chosen in the previous section, *Motion*, applied to the guide robot use case, in which a robot may be asked to guide the user to some final destination, has the following LUs associated: *fn:blow.v*, *fn:coast.v*, *fn:drift.v*, *fn:float.v*, *fn:fly.v*, *fn:glide.v*, *fn:go.v*, *fn:meander.v*, *fn:move.v*, *fn:roll.v*, *fn:slide.v*, *fn:snake.v*, *fn:soar.v*, *fn:spin.v*, *fn:swerve.v*, *fn:swing.v*, *fn:undulate.v*, *fn:weave.v*, *fn:wind.v*, *fn:zigzag.v* and *fn:NULL*[17,18] associated in PM. Within the use case, some LUs such as *fn:fly.v* do not apply, and must therefore not be selected. Indeed, in this case, the relevant LUs to

[17] *vid.* Chapter 2.3.2.5.

[18] The effect of selecting *NULL* LUs in the results obtained will be analysed in Section 4.6.3.

be selected by the ontology engineer are only two: *go.v* and *move.v*.

Semantic Extension and Filtering

The objective of this step is to extend semantically the previously selected LUs by exploiting MCR synsets and SUMO tags.

For that, firstly, all the synsets related to a given LU are obtained. To do that, an automatic semantic extension is performed, through the links from LUs to MCR synsets in PM. Again, not all the synsets obtained through this method are relevant to the use case due to polysemy[19]. Therefore, a manual selection is required.

In PM, synsets are mapped to SUMO tags, which aim to group different synsets into single concepts. Since SUMO tags are more human-readable than synsets and the amount of tags is considerably lower, the selection of synsets is proposed to be performed through SUMO tags. For that, the associated SUMO tags for the whole synset list are automatically extracted from PM and presented to the ontology engineer for the final selection.

To continue with the previous example, for *fn:go.v*, the *mcr:SubjectiveAssessmentAttribute*, *mcr:OccupationalRole;duration*, *mcr:Motion* and *mcr:Death* SUMO tags are presented to the ontology engineer, from which they will select the most relevant ones for the use case (all except *mcr:Death*).

4.6.1.2 Automatic Data Gathering for Intents

In this step, the final words in the target language that evoke the desired intents are automatically obtained considering the manually selected data in previous steps.

Taking as input the set of selected frames, LUs and SUMO tags, the corresponding synsets will be automatically retrieved from PM. After that, the words in the target language that belong to these synsets will be obtained from the MCR, also automatically. As a result of this step, a set of **synonyms in the target language(s)** for each LU will be obtained.

[19]For example, *go* may imply movement from one place to another or to pass away.

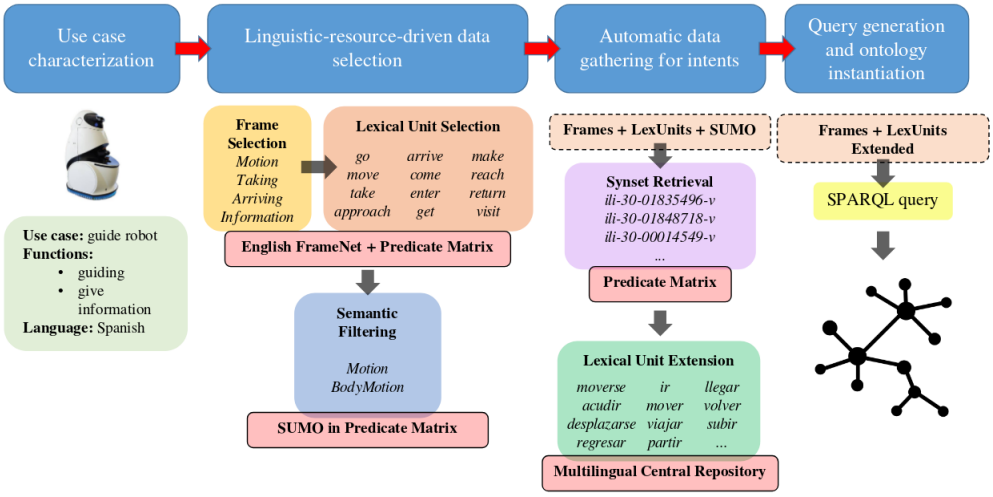


Figure 4.4: Diagram representing the ontology population process presented in this thesis in the context of a guide robot scenario.

In the example used previously, for the frame *Motion* and the relevant synsets for the LU *go.v*, the resulting synonyms in Spanish are the following: *acudir*, *desplazarse*, *ir*, *mover*, *moverse*, *viajar* and *partir*[20].

At this point, all the necessary data to support intent detection has been obtained. A summary of the steps of the strategy considering the use case of the example above (with the rest of relevant frames involved) can be seen in Figure 4.4.

4.6.1.3 Query Generation and Ontology Population

In this last step, the SPARQL query necessary to populate the ontology with the gathered data is automatically created according to the modelling of the target ontology. This SPARQL, an excerpt of which can be seen in Listing 4.1, performs three main operations:

- Creates an individual for each frame, frame head and LU.
- Relates each LU to their corresponding frame head(s).
- Relates each frame head to their corresponding frame(s).

[20] *Attend*, *move*, *go*, *move*, *move*, *travel* and *leave*, respectively.

Listing 4.1: Extract of SPARQL result query for the *Motion* frame.

```
DEFINE input:inference 'urn:RobotGuideInst'
prefix domainOnt: <http://www.foo.bar/tododom>;
prefix domainOntFrame: <http://www.foo.bar/tododfa>;
prefix domainOntWorld: <http://www.foo.bar/tododw>;
prefix domainOntInst: <http://www.foo.bar/tododom-inst>;
insert in <urn:RobotGuideInst>{
domainOntInst:Motion a domainOntFrame:Frame .
domainOntInst:Motion domainOnt:IDval "Motion" .
domainOntInst:go_Motion a domainOntFrame:FrameHead .
domainOntInst:go_Motion domainOnt:IDval "go_Motion" .
domainOntInst:Motion domainOntFrame:hasFrameHead
domainOntInst:go_Motion .
domainOntInst:acudir a domainOnt:LexicalUnit .
domainOntInst:acudir domainOnt:IDval "acudir" .
domainOntInst:go_Motion domainOnt:hasLexicalUnit
domainOntInst:acudir .
domainOntInst:desplazarse a domainOnt:LexicalUnit .
domainOntInst:desplazarse domainOnt:IDval ``desplazarse".
}
```

Once the query has been created, the final step is to execute it to finish the population process.

4.6.2 Strategy in Use

To validate the strategy, it has been used in two interaction use cases that are of relevance in industrial scenarios: a guide/logistics robot and a CMMS (Computerized Maintenance Management System). In both of these scenarios, the language for interaction is Spanish.

The following lines document the process followed in each use case to apply the strategy.

4.6.2.1 Guide/Logistics Use Case

For the **guide/logistics use case**, the robot should be able to guide the user to their destination of choice and to offer information from specific elements (a room, a machine, etc.). The situations that may be identified from this use case, thus, are to move from one point to another, and to ask for information or details about

something. After checking the FrameNet frames in PM, the ones deemed more relevant in this use case are the following: *Motion*, *Taking* and *Arriving* [21].

The next step is to select the appropriate LUs for each frame in PM. In this case, the total number of LUs for all the frames selected in the previous step are 34. Although many of the LUs do cover the use case, such as *move* and *arrive*, there are others, such as *fly* or *blow*, that are too specific and do not apply to the use case and must not be selected.

After this selection, the number of LUs in English is reduced to 12: *go*, *move* (*Motion*); *take* (*Taking*); *approach*, *arrive*, *come*, *enter*, *get*, *make*, *reach*, *return*, *visit* (*Arriving*). Also, *NULL* elements have been considered for evaluation purposes.

For semantic filtering, in this case, the number of available SUMO tags is 34. In this use case, the application of semantic filtering is specially relevant, since from the initial 34 available tags, only two are applicable to the use case and have been selected: *BodyMotion* and *Motion*. The rest of tags were related to situations that were not considered in the use case, such as *Death* or *Cooking* and, thus, were not part of the final selection. This considerable reduction of tags reinforces the importance of this filtering step.

Finally, after having selected the frames, LUs in English and SUMO tags, this information is used to automatically extract the applicable synsets and their translation equivalents in Spanish which, in this case, make a total of 32. The vast majority of the synonyms are relevant to the use case (e.g., *moverse*, *acudir*, *desplazarse*, *regresar*[22]), although a very small portion of them are not entirely appropriate (e.g., *estar_activo – to be active*).

The summary of the information gathered during the population strategy for this use case can be seen in Table 4.6.

[21] Although the original FrameNet includes the frame *Information*, which would suit this use case, it is not included in PM due to the fact that it only has nominal LUs. In the context of this work, this frame will remain out of the evaluation experiments carried out in Section 4.6.3.

[22] *move*, *go*, *move* and *return*, respectively.

Table 4.6: Summary of the information obtained through the use of the population strategy for the guide/logistics use case.

Manual selection			**Automatic selection**
Frames	**Lexical Units**	**SUMO tags**	**Synonyms**
Motion Taking Arriving	**Motion**: go, move **Taking**: take **Arriving**: approach, arrive, come, enter, get, make, reach, return, visit	Motion, BodyMotion (all)	abordar, acercar, acercarse, acostar, acudir, alcanzar, aproximar, aproximarse, arribar, derivar, desplazarse, devolver, encarar, entrar, estar_activo, ir, llegar, mover, moverse, partir, provenir, regresar, reincorporar, resultar, retornar, salir, salir_a_escena, subir, trasladar, venir, viajar, volver

4.6.2.2 CMMS Use Case

In the **CMMS** use case, users should be able to ask about problem solving protocols, request blueprints and similar tasks. Considering the previous remarks, the identifiable situations in this use case are showing information and reporting and solving problems. The applicable frames are, thus, *Resolve_problem*, *Evidence*, *Reporting* and *Communication*.

Regarding the LUs, a total of 30 results have been picked from the previously selected frames. After discarding some LUs that were not applicable to the current use case, such as *contradict* or *deal*, the selected LUs are the following: *solve*, *resolve* (*Resolve_problem*); *indicate*, *reveal*, *show* (*Evidence and Communication*); *inform*, *report*, *tell* (*Reporting*).

In this next step, the number of SUMO tags to choose from is 9 which, after user selection, are reduced to *Communication*, *VisualAttribute* and *IntentionalPsychologicalProcess*. Given this selection, the total number of automatically obtained word synonyms in Spanish for this use case is 39, being most of them appropriate to the use case (e.g., *solucionar*, *informar*, *presentar*[23]). However, other results are not applicable to the use case, such as *denunciar* (*report*, in the legal

[23] *solve*, *inform* and *show*, respectively.

Table 4.7: Summary of the information obtained through the use of the population strategy for the CMMS use case.

Manual selection			**Automatic selection**
Frames	**Lexical Units**	**SUMO tags**	**Synonyms**
Resolve problem Evidence Reporting Communication	**Resolve problem:** solve, resolve **Evidence, Communication:** indicate, reveal, show **Reporting:** inform, report, tell	Communication, VisualAttribute, IntentionalPsychologicalProcess (all)	adjudicar, advertir, afirmar, anunciar, apuntar, asegurar, avisar, comunicar, contar, dar_información, dar_parte, decidir, decir, declarar, deducir, demostrar, denunciar, describir, descubrir, desvelar, dirimir, divulgar, evidenciar, explicar, exponer, indicar, informar, manifestar, narrar, recontar, relatar, reportar, resolver, revelar, señalar, sentenciar, solucionar, solventar, solver

sense of the word).

The summary of the information gathered during the population strategy for this use case can be seen in Table 4.7.

As this validation task has shown, this strategy can be used in different industrial use cases. For this, it has been applied in all the use cases for which KIDE4I has been implemented and adapted in the context of this thesis, which are described in Chapters 5 and 6.

4.6.3 Evaluation

To determine the suitability of the strategy, a gold standard has been created for the two use cases described above, the contents of which have been compared with the results obtained by applying the strategy in each use case. For the gold standard construction, all the possible synonyms in the target language (i.e., Spanish) have been automatically retrieved by using the methods described in the strategy (in this case, only with a manual frame selection, without LU and SUMO filtering), followed by an expert manual selection of the relevant ones.

For this evaluation, different configurations of the manual selection steps described previously have been applied, and their corresponding synonyms have been obtained through the resulting synsets. Among these, combinations that consider *NULL* elements and combinations that do not are distinguished. The reason for this is that, as described in Section 2.3.2.5, although *NULL* elements account for partial mappings between resources, these act as regular LUs in PM and, thus, synonyms can be obtained from them. Considering this, the configurations evaluated are the following:

- **Frame selection (F)**. All LUs for the selected frames and their associated synonyms.
- **Frame + LU selection (F+LU)**. Selection of LUs for the selected frames.
- **Frame + LU selection + *NULL* elements (F+LU + NULL)**. Selection of LUs for the selected frames, considering *NULL* elements.
- **Frame selection + SUMO filtering (F+SUMO)**. All LUs for the selected frames, plus SUMO filtering.
- **Frame + LU selection + SUMO filtering (F+LU+SUMO)**. Selection of LUs for the selected frames, plus SUMO filtering. **That is, the strategy's approach**.
- **Frame + LU selection + SUMO filtering + *NULL* elements (F+LU+SUMO+NULL)**. Selection of LUs for the selected frames, considering *NULL* elements, plus SUMO filtering.

Table 4.8 shows the number of synonyms obtained in each combination. For each of the configurations without *NULL* elements, precision-recall-F1 evaluations have been performed, the results of which can be seen in Table 4.9. These evaluations show that the less filtering, the more recall, as the more synonyms obtained, the more possibility they will include the gold ones. In terms of F1, on average, unique filtering configurations (F+LU and F+SUMO) obtain practically the same results. It is especially relevant that SUMO filtering is able to obtain on average better precision results than LU

Table 4.8: Number of translation equivalents for each PM data configuration selected and use case.

	Guide/Logistics	**CMMS**
Gold	25	17
F	667	137
F+LU	170	69
F+LU+NULL	620	69
F+SUMO	139	70
F+LU+SUMO	32	39
F+LU+SUMO+NULL	113	39

Table 4.9: Precision (P), recall (R) and F1 metrics for all configurations for each use case, and average results.

	Guide/Logistics			**CMMS**			**Average**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
F	0.04	**1**	0.07	0.12	**1**	0.22	0.08	**1**	0.15
F+LU	0.13	0.88	0.23	0.23	0.94	0.37	0.18	0.91	0.30
F+SUMO	0.14	0.76	0.23	0.21	0.88	0.34	0.25	0.82	0.29
F+LU+SUMO	**0.50**	0.64	**0.56**	**0.28**	0.64	**0.39**	**0.39**	0.64	**0.48**

selection –+7 points; 0.25 for SUMO *vs* 0.18 for LU–, which emphasises the importance of a semantic filtering step in this type of task. Nevertheless, it is important to point out that both configurations help improve the results, and that their combination allows to significantly better the baseline figures obtained with F (and are, thus, complementary).

All in all, these results show that manual filtering is necessary in this type of task, as it has been proved that improves up to 30 points the F1 measure on average (0.15 for F *vs* 0.48 for F+LU+SUMO).

Finally, the effect of considering *NULL* elements in the results obtained by the strategy has also been explored in the guide/logistics use case[24]. As it can be seen in Table 4.8, the addition of these elements significantly increases the number of translation equivalents obtained –from 170 to 620 in the F+LU configuration and from 32 to 113 in the F+LU+SUMO one. This may lead to two possible

[24]The CMMS use case does not have any *NULL* elements

Table 4.10: Precision (P), recall (R) and F1 metrics for the F+LU and F+LU+SUMO configurations, comparing the addition and non-addition of *NULL* elements. Results for the guide/logistics use case.

	P	**R**	**F1**
F+LU	**0.13**	0.88	**0.23**
F+LU+NULL	0.04	**0.96**	0.07
F+LU+SUMO	**0.50**	0.64	**0.56**
F+LU+SUMO+NULL	0.15	**0.72**	0.26

implications: on the one hand, that these elements may add relevant equivalents to the list that were not considered, having a positive effect in the results, and, on the other hand, that *NULL* elements add noisy data, which will materialise in a negative impact in the results.

To determine whether the addition of *NULL* elements have positive or negative consequences in the results obtained, the same evaluations performed above have been carried out on the configurations with *NULL*s, the results of which are gathered in Table 4.10. The same impact in recall regarding the number of synonyms described above can be observed, as the *NULL* configurations have higher recall values. For the rest of metrics, the no-*NULL* configurations show better results, especially in the F+LU+SUMO one, where the difference is of 35 points in precision and of 30 points in F1. Thus, it has been proved that the consideration of *NULL* elements do provide noisy data and, due to this, must be discarded from the LU selection step.

Considering the above, the results obtained allow to determine that ontology population for intent-relevant information is a complex task that deserves further investigation.

4.6.4 Other Remarks

The validation of the strategy has shown that the information gathered in the use cases reported above allows to semi-automatically instantiate the domain ontology with relevant data for intent detection, reducing the effort to do so and potential intent identification

errors caused by the fact that manual instantiations may include limited amounts of linguistic information. Furthermore, the incremental filtering through the defined steps allows to drastically reduce options, and semantic filtering allows to fine-tune the LUs to be associated to an intent.

As a side note, manual selection is not always a straight-forward process: frame names may be ambiguous and it can be difficult to assess if they are suitable for the use case at hand. For this, the online version of FrameNet[25] –with a description of each frame in human-friendly terms and a list of its LUs– may be used to distinguish them.

4.7 Conclusions

This Chapter has presented the Task-Oriented Dialogue management Ontology (TODO). This ontology has been designed to be the core element for implementing generic task-oriented dialogue systems that enable a natural communication between human workers and industrial systems with reduced amounts of data, while allowing their easy adaptation to different use cases.

TODO has been developed following a well-defined methodology, Linked Open Terms (LOT) (Poveda-Villalón et al., 2019), to ensure an accurate design and proper reuse. As part of this methodology, the requirements of the ontology have been obtained through interviews with experts to determine the necessities to be covered. From this expert knowledge, 99 CQs have been defined, which can be grouped into 11 BCQs.

Taking into account the analysis of the CQs, in which different areas of knowledge have been identified, TODO has been implemented as a modular ontology. This approach also has many benefits in different aspects, such as maintenance, reuse and adaptation. Considering this, TODO consists of the following modules:

- **Dialogue (TODODial)**. Information related to the dialogue process.

[25] `http://sato.fm.senshu-u.ac.jp/frameSQL/fn2_15/notes/`

  - **Dialogue Management (TODODM)**. Modelling of the dialogue process.
  - **Dialogue Tracing (TODODT)**. Necessary knowledge to store dialogue traces to, for instance, learn from previous interactions.

- **Domain (TODODom)**. Information related to the domain.
  - **Domain Frame-Action (TODODFA)**. Modelling of the actions that are executable by the target system.
  - **Domain World (TODODW)**. Modelling of the elements that are present in the domain scenario.

This Chapter has also proved that TODO is a quality ontology in terms of design, adherence to FAIR principles and the quality of its modularity, according to the evaluations performed to obtain an objective overview of its quality. Furthermore, these evaluations have also helped to identify minor issues to be solved in future versions of TODO.

The ontology has been published online with an extensive documentation that is both machine- and human-readable that allows its proper reuse, which has been generated by following a series of guidelines for this matter. Moreover, to be easily found and accessed, it has also been published in a vocabulary catalogue.

Considering that ontology population in this context is an costly process, this Chapter also presents a strategy to semi-automatically populate domain ontologies (in this case, TODODom) with the necessary information to be able to determine which is the user intent and the action to execute by the target system given a user command. This strategy combines manual and automatic work to exploit existing multilingual lexical, syntactic and semantic resources to generate a SPARQL query to be executed against the domain ontology. As a result, the time and effort to instantiate the ontology is reduced.

The validation and evaluation task of this strategy has determined that a manual filtering stage is necessary to obtain quality data for instantiation, and that the use of lexical and semantic information is of special relevance. In fact, the best results are obtained when these two sources of information are combined, which validates the approach for this strategy. Furthermore, it has been determined that

PM's *NULL* elements provide noise to the results and, thus, should be discarded from the manual selection stage.

The results obtained regarding the strategy are promising, although it is necessary to keep investigating on its improvement, which is a task that is devised as future work. In this sense, the results reported in this thesis show that the research in this matter is going in the right direction.

Moreover, to improve the usability of this strategy, the development of a GUI for its use has been identified as a minor action to consider in the future.

Chapter 5

KIDE4I: Generic Semantic Task-Oriented Dialogue System

KIDE4I (Knowledge-drIven Dialogue framEwork for Industry) is a generic task-oriented dialogue system that, supported by TODO, is easy to adapt to different industrial scenarios, allows a natural communication between workers and industrial systems and is capable of obtaining commands that are executable by the target system given a user request. When the instructions are not clear, or when key information required to fulfil the goal is missing, KIDE4I is designed to engage in a conversation with the user to ask for missing information.

Furthermore, considering that KIDE4I takes as input natural language expressions, it is logical to assume that new structures that have not been considered in the implementation phase may appear. For this, KIDE4I uses Lifelong Learning (LL) (B. Liu & Mazumder, 2021) methods and techniques to be able to learn over time from new interactions. More specifically, and in line with the state-of-the-art regarding this approach applied to dialogue systems (Veron et al., 2019), LL approaches will be used to enhance domain-related components.

The architecture of KIDE4I can be seen in Figure 5.1. In it, the 4 main components of the dialogue system can be distinguished: the **key element extraction** component, the **polarity interpreter**, the **semantic repository** and, finally, the **dialogue manager**, the

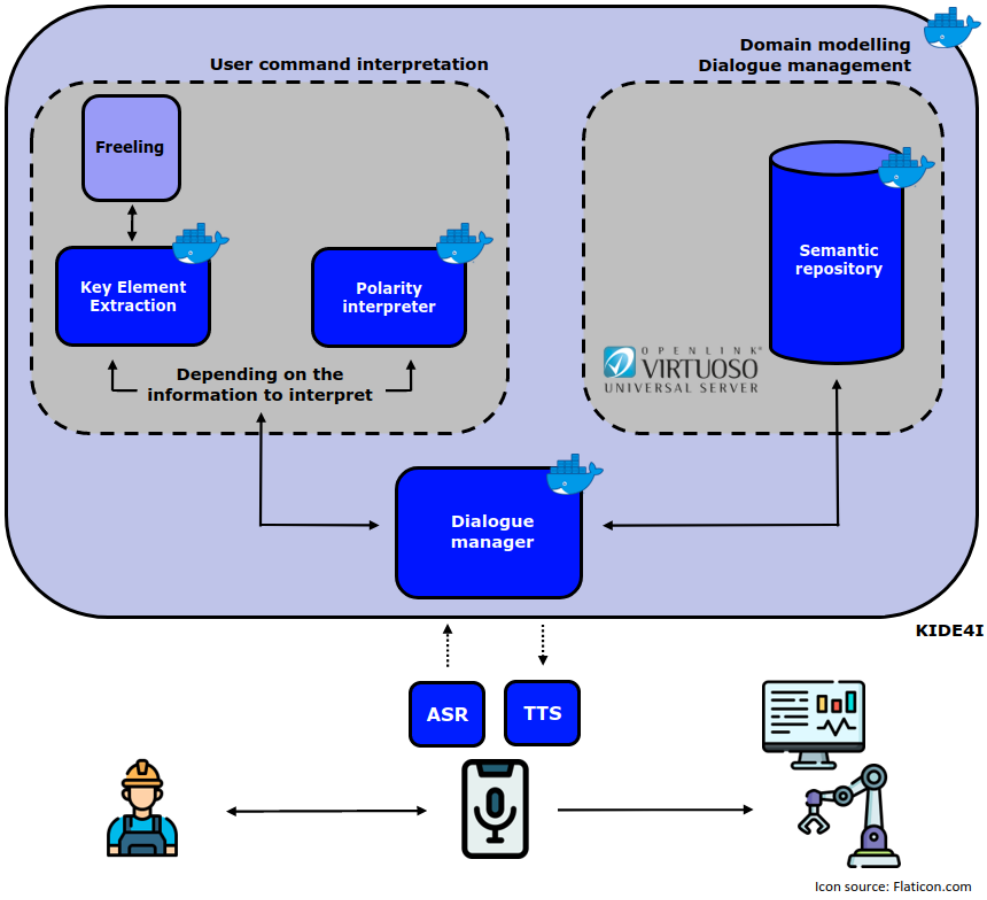


Figure 5.1: KIDE4I architecture.

design and implementation process of which will be described in this Chapter.

In a regular interaction sequence between a user and KIDE4I, the user will perform a voice command, which will be transcribed and sent to KIDE4I. At this point, KIDE4I's **dialogue manager** will interpret the command by first extracting its relevant key elements according to a set of rules or a ML-based model implemented in the **key element extraction** component. Once these key elements have been obtained, the information on the **semantic repository** –which is modelled according to TODO– is exploited so as to obtain, from these, the action to be sent to the target system and the necessary elements needed for that action to execute successfully (arguments) in a target-system-readable format. After processing all the information in the command, the **dialogue manager** checks again with the **se-**

Figure 5.2: Teknibot is a guide robot that is able to guide its users to a given destination.

mantic repository whether all the necessary information has been obtained. In some cases, there will be information that is missing or the system will need the user to confirm certain pieces of information, and the system will require a response from the user. In those cases, the semantic repository will provide the **dialogue manager** with the request to present to the user and the type of information expected from that response (i.e., a piece of information or a confirmation –in the form of *yes*/*no* and equivalents). After receiving the user response, and depending on the type of information to process, the **dialogue manager** makes use of the **key element extraction** component, mentioned above, or the **polarity interpreter** to interpret pieces of information or confirmations, respectively. Once the **dialogue manager** checks with the **semantic repository** that all the necessary information has been obtained, the command to be sent to the target system is generated and sent for its execution.

All components have been designed as generically as possible to be as generic and reusable as possible. These components have also been developed to be language independent and, thus, no additional effort to adapt KIDE4I to different languages is necessary.

An initial version of KIDE4I has been implemented to be used as starting point for further adaptations. This implementation – **KIDE4Guide**– has been developed for a guidance use case, which consists on interaction in Spanish with Teknibot, a guide robot that can be seen in Figure 5.2. This robot shows similar characteristics

to logistics robots –which are relevant in industrial scenarios–, and is able to move from one point to another: given a destination –obtained from user interactions–, it is able to guide its target users to their destination of choice in a given environment and, additionally, it is capable of giving information about specific objects or spaces. In this case, KIDE4Guide has been implemented for its use in the research centre Tekniker and its different laboratories, workshops, people, etc. The implementation process for this use case will be described through this Chapter.

5.1 Key Element Extraction

The main function of KIDE4I's Key Element Extraction (KEE) component is to obtain the relevant key elements from a transcribed user voice command that conveys a piece of information. These key elements contain the core information of the command, and to extract them correctly is key to a successful interpretation.

In the context of this thesis, KEE resembles a NERC problem, in the sense that key elements have to be identified and classified. For this, the literature distinguishes between two basic approaches: rule-based and ML methods (Maynard et al., 2016). The use of one approach or the other depends on the development process circumstances. On the one hand, rule-based systems are a robust starting point in specific domains where there is a lack of raw or annotated data, such as industrial scenarios. However, manually-generated rules are hard to come by and maintain, as they require human intervention. On the other hand, ML-based methods are easier to maintain and adapt than rule-based ones, since they have the ability to automatically learn from new sequences while preserving previous knowledge, but require training data to build the models. Due to this, LL settings generally rely on ML-based methods.

To implement a ML-based KEE component, the access to annotated data to train the models is essential. Since one of the main challenges regarding industrial scenarios is precisely the lack of annotated data, which hinders the use of ML methods to perform key element extraction. One way to alleviate this problem is to leverage out-of-domain data –obtained from different but similar domains– to

augment existing domain training data, an approach that has been proved to obtain positive results (Fromreide & Søgaard, 2014; Persson, 2017).

Considering the remarks above, the methodology in this thesis in regard to KIDE4I's KEE component development is to first model hand-made rules that enable a rule-based key element extraction to be able to automatically generate annotated data, which can be followed by a manual expert correction. This annotated data will be used to train the models to perform ML-based KEE, with the objective of developing a KEE component that is able to learn over time with no human intervention involved. To do so, a LL setting will be resembled, in which new examples are reused to improve the models and, thus, domain data is augmented with out-of-domain data.

The following sections describe the generic design of both rule-based and ML-based approaches to KEE for KIDE4I. Then, to provide an experimental setting for the validation and evaluation of the methodology described above, both KEE approaches have been implemented in KIDE4Guide. For each of them, the definitions and rules modelled and the annotated datasets generated are detailed. After that, an experimentation task is reported, in which the performance of both approaches according to different domain and out-of-domain data configurations is evaluated to determine the most suitable strategy for implementation and future adaptations.

5.1.1 Key Element Types in Industrial Scenarios

In industrial scenarios in which task-oriented dialogue systems may be involved (e.g., collaborative robotics and exploitation of information systems), the most usual interaction structures consist of a **verbal** sequence and its **target**. In this sense, the verbal sequence refers to the action to be executed by the target system and the target is its complementary information. Example (14) shows instances of these key elements in a set of samples of potential commands to be directed to KIDE4I.

(14) a. I want you to **pick**$_{verb}$ the **yellow piece**$_{target}$. (collaborative robotics use case)

b. Please **show**$_{verb}$ me the **next step**$_{target}$ of the current procedure. (exploitation of information systems use case)

c. I would like to **see**$_{verb}$ the **next work order**$_{target}$ for the **pallet truck**$_{target}$. (exploitation of information systems use case)

However, the fact that *verbs* and *targets* appear in these commands does not imply that among use cases these key elements share the same characteristics; each of them may have their own particularities. This can be observed in Example (14c), which has a key element of the *target* type for the item the user wants to *see* –the *next work order*– and another one for the restriction for the work order to show –it has to be for the *pallet truck*.

Furthermore, key elements tend to follow specific morphosyntactic structures (for example, *action*s usually correspond to verbal forms). In the same way as above, use-case particularities may also be set for the different morphosyntactic structures followed by a given key element. For instance, in Examples (14a) and (14b), the *target*s consist of an adjective (*yellow* and *next*, respectively), followed by a noun (*piece* and *step*, respectively), but in Example (14c), the targets consist, on the one hand, in an adjective (*next*) followed by a nominal construction (*work order*) and, on the other hand, in a nominal structure (*pallet truck*). These observations are of special relevance, as it can be determined that morphosyntactic information plays a very important role in the KEE process, as it will be seen in the following sections.

5.1.2 Design

The process to obtain and classify the key elements contained in a command is described in the following lines, and the design of the KEE component to do so can be seen in Figure 5.3.

First of all, when the command arrives to the key element extraction component, it is **pre-processed** so as to obtain its morphosyntactic information. For that, the linguistic tool Freeling (Carreras et al., 2004)[1] and its *full parsing* analysis functionality in the *CoNLL*

[1] In its version 4.0.

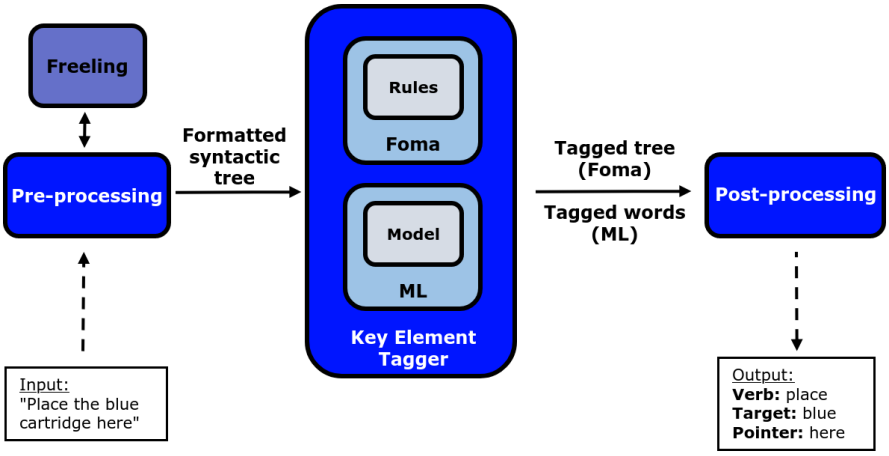


Figure 5.3: Architecture of KIDE4I's key element extraction component.

output format is used. From this analysis, the syntactic tree representation of the user command is extracted.

This syntactic tree corresponding to the user command serves as input to the next subcomponent, the **key element tagger**. This subcomponent delimits and classifies the key elements from the command according to their syntactic structure by making use of rules or supervised models, as it will be described in the following sections. The output of the key element tagger is the command's keyword-tagged syntactic tree –in the case of a rule-based approximation– or a set of tags corresponding to each word in the command –in the case of a ML-based approximation.

Finally, to retrieve from the key element tagger output the list of detected and classified keywords, a **post-processing** step is performed.

The result of this KEE component is the set of key elements extracted and classified from the user command, according to the characteristics of the use case (i.e., the different key element classes).

5.1.2.1 Rule-Based Key Element Extraction

Towards implementing the KEE component, the rule-based approach presented in this Section allows to construct the dialogue sys-

tem when no data is available to train ML models for key element tagging. Furthermore, it is also used to obtain annotated domain data to train these models. This approach solely relies on domain knowledge for key element tagging and, thus, it does not depend on external data and its availability, which represents one of the main challenges when developing such systems in low-resourced scenarios, such as industrial ones.

The rules are to be implemented in Foma (Hulden, 2009b), which allows to define the key element structures and the rules for their identification and classification. These rules are to be manually written by an expert by introspection of data for each use case.

So as to create the rules, the first step is to define the structures that correspond to the elements to detect. Example (15) shows the definition of an element and the observations that justify it for a potential industrial scenario, in which the user requests the system to take or bring something somewhere, which would be of the *target* type. The definition of this target determines its structure particularities in this potential use case.

(15) Definition: *target*s are `Noun`s + their modifiers (*prepositional phrases*, *numbers*, *adjectives*, etc.).

a. Quiero **lubricante**. (Single `Noun`)
I want **lubricant**.

b. Llévalo al **taller *de montaje***. (`Noun` + *Prepositional phrase* ('de montaje'))
Take it to the ***assembly*** **laboratory**.

c. Llévalo al **laboratorio *4***. (`Noun` + *Number* ('4'))
Take it to **laboratory *4***.

After defining the elements, the rule to tag them is defined. Examples (16) and (17) show a definition and a rule as they would appear in the grammar[2]

(16) Definition: 'A *target* can be a single noun (`SNDest`), or can be followed by a Number (`SNNum`), an Adjective Phrase (`SAdj`), or a Prepositional Phrase (`SPDest`)'.

[2]However, the definition has been simplified for a better understanding.

a. `define Target [SNDest [SNNum | SAdj | SPDest]* ];`

(17) Rule: 'Place a tag at the beginning and at the end of a *target*. Also, the longest match has to be selected (`@->`)'.

a. `define TagTarget [Target @-> "<TARGET>" ... "</TARGET>"];`

Therefore, given an input tree obtained from the pre-processing step and a set of definitions and rules, the rule-based key element tagger returns the same tree with each key element delimited by tags, at chunk level. These tags are parsed in the post-processing step to obtain the list of key elements for the user command. For the sentence 'Quiero lubricante', the inputs and outputs from the different processing steps in the rule-based system are shown in Example (18).

(18) a. Input: "Quiero lubricante".

b. Pre-processing output:
`(grup-verb:1(verb:1)(sn:2(grup-nom-ms:2(n-ms:2)))(F-term:3))`

c. Key element tagger output:
`(grup-verb:1`**`<VERB>`**`(verb:1)`**`</VERB><TARGET>`**
`(sn:2(grup-nom-ms:2(n-ms:2)))`**`</TARGET>`**`)`

d. Post-processing:
- Verb: **`(verb:1)`** → *quiero*
- Dest: **`(sn:2(grup-nom-ms:2(n-ms:2)))`** → *lubricante*

e. Post-processing output:
- Verb: *quiero*
- Target: *lubricante*

5.1.2.2 Machine-Learning-Based Key Element Extraction

For machine-learning-based key element extraction, since it is regarded as a sequence labelling process, and following usual practice in NERC tasks, the key element tagger is designed to annotate its input at word level following the BIO schema –which marks each word as beginning (B), being inside (I) or being outside of a key element (O), respectively– along with the key element type, as Example (19) shows.

(19) [**Take**$_{B-ACTION}$] it$_O$ to$_O$ the$_O$ [**assembly**$_{B-TARGET}$ **laboratory**$_{I-TARGET}$].

The features to be considered for each token[3] to perform supervised analysis are its syntactic features –extracted from the syntactic tree obtained from Freeling– and its cluster, according to a set of precomputed Clark Clusters for Spanish (Agerri & Rigau, 2018). Clark clusters deal with word variations or synonyms that express similar meaning, which are usually assigned to the same cluster. Additionally, previous and next token's features are also used to represent the current token as the meaning for a word, besides themselves, is also determined by its context (Firth, 1957). In this approach, thus, the following 18 features are used to represent each token in a given user command:

- Syntactic features (15 features): *terminal*, *parent1*, *parent2*, and *parent3*. *terminal* stands for the syntactic element in the lowest position in the tree. *parentN* is the N-th parent of a non-terminal category in the syntactic tree, counting from lower to higher position.
  - For the previous token, the tag *_previous* is added to the original feature names. Thus, the features to refer to the previous token are *terminal_previous*, *parent1_previous*, *parent2_previous*, and *parent3_previous*. If the current token is the first one –and, therefore, no previous token can be referred to–, the value for each of these features is "top".
  - For the next token, the tag *_next* is added to the original feature names. Thus, the features to refer to the next token are *terminal_next*, *parent1_next*, *parent2_next*, and *parent3_next*. If the current token is the last one –and, therefore, no following token can be referred to–, the value for each of these features is "last".
- Other relevant features (3 features): *cluster*. The cluster the token belongs to.

[3]Although the ML-based KEE component performs at word level, the annotation process is initially made at token level (e.g., given the word 'can't', its tokens would be 'can' and 'not'). The tag associated to the word in the post-processing step is the tag obtained for its first token (e.g., the tag for 'can't' would be the tag obtained for 'can').

Table 5.1: Syntactic features for the sentence 'Quiero lubricante'. For extension reasons, only the features for the current word are depicted.

word	**par3**	**par2**	**par1**	**terminal**	**cluster**
Quiero	0	0	grup-verb	verb	59
lubricante	0	sn	grup-nom	n	99
.	0	0	0	F-term	100

- To refer to the previous token, the feature *cluster_previous* is used. If the current token is the first one, the value for this feature is "top".
- To refer to the next token, the feature *cluster_next* is used. If the current token is the last one, the value for this feature is "last".

Tree-structure-related features are ordered from the highest to the lowest position in the tree, and Number (singular/plural) and Gender (masculine/feminine) marks have been removed to provide a generic analysis, as illustrated in the example in Table 5.1.

As for the algorithms to use, the Support Vector Machine (SVM) (in its linear implementation) and Conditional Random Fields (CRF) (in which key element extraction is considered a sequence labelling problem) algorithms have been considered to develop the component, as they are among the most used algorithms for supervised NERC tasks in the literature for their good results (Goyal, Gupta, & Kumar, 2018). In both cases, each command is annotated at token level.

Linear SVM is provided by Weka (Hall et al., 2009) and, for the CRF method, the *python-crfsuite* library implementation[4] is considered. Default parameters are to be used in both methods, as no fine-tuning is expected in this setting.

Thus, given a set of syntactic and cluster features for a user command obtained in the pre-processing step, the ML-based key element extraction tagger annotates each token in the sentence with the corresponding tag. In the post-processing step, these tags are aligned with their corresponding word from the user command and grouped

[4] `https://python-crfsuite.readthedocs.io/en/latest/`

into chunks. The final output, thus, is presented at chunk level. This process can be observed in Example (20).

(20) a. Input: "Quiero lubricante".

b. Pre-processing:
`(grup-verb:1(verb:1)(sn:2(grup-nom-ms:2(n-ms:2)))(F-term:3))`

c. Pre-processing output:
`[[0, 0, grup-verb, verb, 59], [0, sn, grup-nom, n, 99], [0, 0, 0, F-term, 100]]`

d. Key element tagger output:
`[["B-ACTION"], ["B-TARGET"], ["O"]]`

e. Post-processing:
- *Quiero* → "B-ACTION"
- *lubricante* → "B-TARGET"
- . → "O"

f. Post-processing output:
- Verb: *quiero*
- Target: *lubricante*

5.1.3 Implementation

The following sections will describe the process to implement the KEE component for KIDE4Guide. First of all, the process to model the definitions and rules for the rule-based KEE component will be reported, along with some examples for each of them.

As for the ML-based KEE component, both domain and out-of-domain[5] corpora have been selected and annotated with the rule-based KEE component. Several experiments have been carried out to assess whether domain data augmentation with data from similar domains is useful in this setting and to determine the best approach for key element extraction. In them, different configurations of the annotated datasets have been considered to train different ML-based models. The results and conclusions of such experiments are also reported.

[5]From different but similar domains to the one at hand.

5.1.3.1 Rule-Based Key Element Extraction

In order to correctly model the rules for KIDE4Guide's KEE component, it is necessary to perform a thorough analysis of the characteristics of the use case, as they must comprehensively consider the possible command variations that may be found in an interaction scenario.

In this case, given a user request, the relevant information to extract from commands by KIDE4Guide are **actions** and **destinations** (which consist of a *target*, *preposition* and the target that depends on the preposition –the *complement*). The action is the element that describes the act to perform, associated with the destination ('***drink*** water'), whereas the destination is the spatial point where the user wants the robot to lead them. It can designate a specific point ('meeting room') or an element related to the destination point ('coffee' for 'coffee machine'). Additionally, and to provide more level of detail to the destination, the prepositional phrases inside the expressions that refer to a destination are also considered ("Quiero una sala *con* ***PC***" - "I want a room *with a* ***PC***"). Example (21) shows an instance of these key elements.

(21) a. Quiero $\mathbf{ir}_{action}$ a una [$\mathbf{sala}_{target}$ $\mathbf{con}_{preposition}$ $\mathbf{PC}_{complement}]_{destination}$
b. I want to $\mathbf{go}_{action}$ to a [$\mathbf{meeting\ room}_{target}$ $\mathbf{with}_{preposition}$ a $\mathbf{PC}_{complement}]_{destination}$

However, there are a few restrictions regarding prepositional phrases, since the required information depends on the words related to the preposition:

1. The annotated prepositions are the ones that denote places that contain something (*con*, for Spanish; *with* for English). For example:

   (22) a. Quiero ir a una [**sala** *con* ***PC***]
   b. I want to go to a [**room** *with a* ***PC***]

2. The relevant elements to tag are prepositional phrases that are restricted to common nouns. Thus, prepositional phrases with

proper nouns as complements are not to be tagged. For instance, in Example 2, *John Smith* does not belong to the term *meeting*:

(23) a. Tengo una [**reunión**] *con* [***John Smith***]
b. I have a [**meeting**] *with* [***John Smith***]

Considering this, for KIDE4Guide, a total of 47 definitions and 10 rules have been modelled for Foma. Examples (24) and (25) include a sample of definitions and rules, respectively[6].

(24) a. Definitions for numbers (`Number`), which are present in the syntactic tree as word indices:

```
define Num ["0"|1|2|3|4|5|6|7|8|9];
define Dec [1|2|3|4|5|6|7|8|9];
define Number [Dec* Num];
```

b. Definitions for prepositions: regular prepositions (`Prep`) and prepositions that denote "containing" (e.g., *in*) (`Prep Content`):

```
define Prep [{(prep:} Number {)}];
define PrepContent [{(prep-cont:} Number {)}];
```

(25) a. Rule to tag prepositions (`Prep`/`PrepContent`) with the tags `<PREP></PREP>`:

```
define TagPrep [[Prep | PrepContent] @->
"<PREP>" ...  "</PREP>"];
```

As for the tag set, KIDE4Guide's implementation of the rule-based key element tagger tags the sentence at different syntactic levels to extract the chunks corresponding to the key elements. This is reflected in the different tags considered:

- ***GV***. A verb phrase. E.g., *I want* ***to go to a meeting room with a PC***. This tag is used to separate complex sentences (i.e., sentences with more than one verb).

[6]KIDE4Guide's complete rule set can be seen in Appendix C.

- ***GV-DEST***. A verbal form. E.g., *I want to* ***go*** *to a meeting room with a PC*. This tag applies to *action*s.
- ***DEST***. Core target element or element that depends on the preposition. *I want to go to a* ***meeting room*** *with a* ***PC***. This tag applies to *target*s and *complement*s.
- ***PREP***. *Containing* preposition in a prepositional phrase. *I want to go to a meeting room* ***with*** *a PC*. This tag applies to *preposition*s.
- ***SP***. Whole prepositional phrase (`PREP` + `DEST`). *I want to go to a meeting room* ***with a PC***. This tag is used to determine the *complement* that depends on the *preposition*.
- ***SP-DEST***. Whole target element that includes a prepositional phrase (`DEST` + `SP`) *I want to go to a* ***meeting room with a PC***.

As for the post-processed output, this component returns the resulting set of key elements according to the following schema:

- ***ACTION***. *I want to* ***go*** *to a meeting room with a PC*.
- ***DEST***. Element that conveys all the destination information. *I want to go to a* ***meeting room with a PC***.

  This element has a series of specialisations:

  - ***TARGET***. Core target element. *I want to go to a* ***meeting room*** *with a PC*.
  - ***PREP***. *Containing* preposition in a prepositional phrase. *I want to go to a meeting room* ***with*** *a PC*.
  - ***COMPL***. Element that depends on the previous preposition. It conveys the element that is contained in a target element. *I want to go to a meeting room with a* ***PC***.

The following sections present the annotated datasets generated and used to implement the ML-based key element extraction component for KIDE4Guide, which represent either the domain or different –but similar– domains (which will be considered as out-of-domain datasets for practical purposes).

5.1.3.2 Machine-Learning-Based Key Element Extraction: Domain Data

This section describes IMH, the dataset used to train the ML-based KEE component with domain data. IMH is a dataset in Spanish for the guidance scenario. It consists of a set of commands directed to a guide robot, obtained through a experimentation task in which participants[7] were asked to perform requests to a guide robot, KTBot (Susperregi et al., 2012), by interacting with a mobile device with an app that serves as a link between the person (through a voice command) and the robot.

In total, IMH includes 179 sentences, with a total of 1151 words and 1232 tokens and an average of 6.43 words per sentence. The dataset includes both simple and compound/complex sentences, meaning that there is only one verb or more than one, respectively. The complexity of the vast majority of the compound/complex sentences is reduced to main verbs that have another verb (or verb phrase) as a direct object, as can be seen in Example (26).

(26) a. *Quiero* **salir** .
'I want to exit.'
b. *Quiero* **beber** agua .
I want to drink water.'

The dataset has been semi-automatically annotated using the rule-based approximation for KEE in KIDE4Guide and manually revised and corrected when necessary by an expert linguist. In total, IMH includes 368 key elements, annotated with the following tags, inspired by the rule-based KEE output:

- ***ACTION***. *I want to* ***go*** *to a meeting room with a PC.*
- ***DEST-TARGET***. Core target element. *I want to go to a* ***meeting room*** *with a PC.*
- ***DEST-PREP***. *Containing* preposition in a prepositional phrase. *I want to go to a meeting room* ***with*** *a PC.*

[7]These participants came from the Instituto de Máquina-Herramienta, in Elgoibar, Spain. The initials from this institution (IMH) were used to name the corpus.

Table 5.2: Example of annotated sentence from IMH corpus

word	**tag**
quiero	B-ACTION
una	O
sala	B-DEST-TARGET
de	I-DEST-TARGET
reuniones	I-DEST-TARGET
con	I-DEST-PREP
micrófono	I-DEST-COMPL
∗fullstop∗	O

- ***DEST-COMPL***. Element that depends on the previous preposition. It conveys the element that is contained in a target element. *I want to go to a meeting room with a* ***PC***.

The corpus is annotated at word level using the BIO schema, considering the chunks obtained by the rule-based key element extraction component. Table 5.2 shows an example of a sentence with its corresponding tags. As it can be observed, DEST-TARGET, DEST-PREP and DEST-COMPL are classifications for the same chunk. In this sense, DEST is the key element class, and TARGET, PREP and COMPL are specifications of DEST.

In order to train ML models to perform key element extraction with this data and evaluate their performance, different IMH sets and subsets have been considered:

- **IMH hand-corrected, complete (IMH-HC)**. This is the whole IMH corpus, manually corrected.
- **IMH hand-corrected, for training (IMH-TR-HC)**. It corresponds to $\frac{2}{3}$ of IMH-HC.
- **IMH hand-corrected, for testing (IMH-TS-HC)**. It corresponds to the remaining $\frac{1}{3}$ of IMH-HC.

The specific details of these sets –such as number of sentences, words or key elements by type– can be found in Tables 5.3 and 5.4.

Table 5.3: Size of domain datasets and number of occurrences for each key element, at chunk level. The average number of words per chunk (AWC) is also provided.

Data	Sentences	Words	Tokens	Chunks	
				ACTION	DEST
IMH-HC	179	1151	1232	190 (AWC 1)	178 (AWC 1.88)
IMH-TR-HC	120	759	819	130 (AWC 1)	115 (AWC 1.82)
IMH-TS-HC	59	392	413	60 (AWC 1)	63 (AWC 1.98)

Table 5.4: Number of word occurrences for each DEST specialisation in domain datasets.

Data	DEST-TARGET	DEST-PREP	DEST-COMPL
IMH-HC	316	9	9
IMH-TR-HC	203	3	3
IMH-TS-HC	113	6	6

5.1.3.3 Machine-Learning-Based Key Element Extraction: Out-Of-Domain Data

So as to increase the size of IMH using out-of-domain data, 2 corpora have been used: the Human Robot Interaction Corpus (HuRIC) (Bastianelli et al., 2014) and the Collaborative Manipulation Corpus (CMC) (Scalise, Li, Admoni, Rosenthal, & Srinivasa, 2018). These corpora contain English sentences from a domain close to IMH's though different, as it will be described further.

For these corpora to be compatible with IMH, both have been formatted accordingly: each sentence has been automatically translated to Spanish, using Google Translate services[8], without any further revision. Following their translation, all sentences have been annotated using KIDE4Guide's rule-based KEE component. The following lines will describe the specific characteristics of each corpus and the preprocessing tasks performed on them.

[8]Through the `GOOGLETRANSLATE` function in Google Sheets, as of July 2019. More information can be found in `https://support.google.com/docs/answer/3093331?hl=en`.

Human Robot Interaction Corpus (HuRIC)

HuRIC is a corpus that has been generated from interactions with house service robots, which are annotated taking into account linguistic information, such as morphosyntactic and semantic information. This information has been discarded at this work, as the same tagset as IMH has been used to annotate each command.

The translated version generated and used in this thesis comprises 318 sentences, with 2228 words and 2697 tokens and an average of 7.03 words per sentence. Example (27) includes a sample of the interactions included in the original version of HuRIC, along with their automatically-generated translation.

(27) a. "Please get the cushion from the bed".
"Por favor el cojín de la cama".

b. "Go to the living room".
"Ir a la sala de estar".

c. "Please walk slowly to the kitchen".
"Por favor, caminar lentamente a la cocina".

As in IMH, the automatically-generated annotations by the rule-based KEE have been manually corrected by a linguist. However, to assess the effect of introducing noisy data in the training, the automatically-generated version has also been preserved. The corrected version includes 810 key elements, whereas the automatically-generated one includes 811, 632 of which are correct (respect the manually-corrected version). Thus, the automatically-generated version has a 78% of correct key elements.

Considering this, and to test different configurations of data to train key element extraction models, the following sets and subsets have been defined:

- **HuRIC not-hand-corrected, complete (HURIC-NHC)**. It corresponds to the whole HuRIC, automatically tagged using the rule-based key element extraction system.

- **HuRIC not-hand-corrected, for training (HURIC-TR-NHC)**. It corresponds to $\frac{2}{3}$ of HURIC-NHC.

- **HuRIC hand-corrected, complete (HURIC-HC)**. It corresponds to the whole HuRIC, automatically tagged using the rule-based key element extraction system and hand-corrected after.
- **HuRIC hand-corrected, for training (HURIC-TR-HC)**. It corresponds to $\frac{2}{3}$ of HURIC-HC.
- **HuRIC hand-corrected, for testing (HURIC-TS-HC)**. It corresponds to the remaining $\frac{1}{3}$ of HURIC-HC.

Collaborative Manipulation Corpus (CMC)

CMC has been obtained from interactions regarding object specification in a manipulation scenario. Out-of-the-box, CMC includes sentences with multiple lengths, from 1 to 54 words. To avoid one-word and very long sentences (that eventually may pose automatic translation problems), only the sentences between 2 and 10 words have been kept for experimentation in this thesis. The maximum number of words has been defined considering the average sentence length in the other two datasets, rounded up to consider slightly longer sets of sentences.

The filtered and translated version of CMC includes 435 sentences, with 3288 words and 3583 tokens and an average of 7.56 words per sentence.

Example (28) is a sample of the sentences included in the original CMC, along with their automatic translation.

(28) a. "Get the yellow block".
"Obtener el bloque amarillo".
b. "Pick up the closest green block".
"Recoger el bloque verde más cercano."
c. "Pick up the blue block on the right".
"Recoger el bloque azul a la derecha."

For the experimentation tasks, the whole CMC, automatically tagged using the KIDE4Guide's rule-based key element extraction component –from now on, **CMC-NHC**–, has been used, which includes a total of 760 key elements. In this case, no manual revision was performed.

Table 5.5: Size of out-of-domain datasets and number of occurrences for each key element, at chunk level. The average number of words per chunk (AWC) is also provided.

Data	**Sentences**	**Words**	**Tokens**	**Chunks**	
				ACTION	**DEST**
HURIC-NHC	318	2228	2697	283 (AWC 1)	529 (AWC 1.49)
HURIC-TR-NHC	212	1548	1887	177 (AWC 1)	357 (AWC 1.56)
HURIC-HC	318	2228	2697	322 (AWC 1)	488 (AWC 1.87)
HURIC-TR-HC	212	1548	1887	211 (AWC 1)	316 (AWC 2.11)
HURIC-TS-HC	106	680	810	111 (AWC 1)	172 (AWC 1.44)
CMC-NHC	435	3288	3583	140 (AWC 1)	620 (AWC 1.73)

Table 5.6: Number of word occurrences for each DEST specialization in domain datasets.

Data	**DEST-TARGET**	**DEST-PREP**	**DEST-COMPL**
HURIC-NHC	741	20	27
HURIC-TR-NHC	528	11	17
HURIC-HC	859	11	44
HURIC-TR-HC	617	9	41
HURIC-TS-HC	242	2	3
CMC-NHC	1074	0	0

The specific details of out-of-domain datasets –such as number of sentences, words or key elements by type– can be found in Tables 5.5 and 5.6.

5.1.3.4 Experimentation

So as to evaluate the performance of both rule and supervised approaches to develop the KEE component and to assess the best approach for future adaptations, an experimentation task consisting on three subtasks has been defined. In each experimentation subtask, different configurations of data and models for training and testing have been explored. The aim of doing so is to give a wide overview of what can be expected both from rules and ML-based methods when domain and/or out-of-domain data is involved. These subtasks are

motivated by the following premises:

- **A ML-based KEE component trained with domain data can achieve similar results as rules in domain data**. As rules have been created through the analysis of potential domain sequences, the rule-based system will obtain good results to extract key elements from structures considered as from the domain. For similar reasons, the use of domain data for training the ML-based KEE will also obtain good results. This premise is tackled in the first subtask, and partially in the second.
- **Out-of-domain data for training the ML-based KEE component can be used to annotate domain examples**. Using automatically-annotated out-of-domain data from different but similar domains to train the ML-based KEE will obtain acceptable results to annotate domain data due to domain proximity. This premise is tackled in the second subtask.
- **When new examples appear, the combination of domain and out-of-domain data for training the ML-based key element extraction component will outperform domain-based approaches –especially if out-of-domain data is corrected**. When new examples appear, domain-data-based approximations will reduce their performance compared to when domain examples are involved, as neither rules nor domain-based ML models have been specifically designed for this type of data. However, the combination of domain and automatically generated out-of-domain data –assuming that there may be errors in the training set– will achieve better results, as more generalisations will be made. In this sense, the best results will be obtained with revised data. This premise is tackled in the third subtask.

For evaluation, and following usual practice, the results for each of the subtasks in this Section will be reported in terms of standard precision, recall and F1 measures, used on sequence labelling tasks (He et al., 2020).

Furthermore, these subtasks have been complemented by a qualitative evaluation that compares the results obtained by rules and the ML-based method that performs best among subtasks, in terms of the structures that both approaches are able –or not– to detect.

Table 5.7: Results of key element identification and classification algorithms on **IMH-TS-HC** at chunk level.

Algorithm	Training set	Precision	Recall	F1	Acc
Rules	-	83.92	88.89	86.33	95.30
SVM	IMH-TR-HC	82.43	**90.37**	86.22	94.78
CRF	IMH-TR-HC	**92.42**	**90.37**	**91.39**	**96.34**

First subtask

For the first subtask, only domain data has been considered. To perform the experiments, the rule-based KEE component, as well as the supervised models trained with **IMH-TR-HC**, have been used to annotate the test set, **IMH-TS-HC**. The results are reported in Table 5.7. The results have been calculated over whole chunks, using the evaluation scripts for the CoNLL18's NER task[9]. Additionally, accuracy values at word level are also reported.

As expected, the rule-based method is able to obtain good results –as rules were designed considering the domain–, but supervised systems yield better results in general. Overall, the CRF method performs best with an F1 of 91.4% and an accuracy of 96.34%.

Second subtask

Considering the results of the first subtask, it is important to keep in mind that the size of the test dataset is very small and may not be fully representative for the evaluation of the supervised methods. To solve this test set size gap, **IMH-HC** has been used as test set in this second subtask.

To assess the performance of domain-based methods over the whole domain data, two evaluations have been performed, depending on the approach to implement the KEE component. For the rule-based approach, **IMH-HC** has been tagged by using rules, achieving a 96.79% of accuracy, which is, again, an expected behaviour. As for the ML-based approach, 10-fold cross validation techniques have been applied using **IMH-HC**. The results obtained show a 84.01% of accuracy for SVM and a 96.66% for CRF. Once more, among ML

[9]More details on the evaluation task in CoNLL18 and a link to the script can be found at `http://universaldependencies.org/conll18/evaluation.html`

Table 5.8: Results of key element identification and classification algorithms on **IMH-HC** at chunk level.

Algorithm	**Training set**	**Precision**	**Recall**	**F1**	**Acc**
Rules	-	85.68	89.90	87.74	95.74
SVM	HURIC-NHC	61.26	65.54	63.33	83.23
CRF	HURIC-NHC	**76.61**	**67.88**	**71.98**	**87.32**
SVM	CMC-NHC	**55.49**	**51.04**	**53.17**	**78.54**
CRF	CMC-NHC	50.75	26.42	34.75	68.20
SVM	HURIC-NHC + CMC-NHC	67.67	69.95	68.79	87.14
CRF	HURIC-NHC + CMC-NHC	**76.32**	**67.62**	**71.70**	**86.79**

algorithms, CRF obtains the best results.

As the data for the domain is small, the possibility of augmenting the domain with examples from different but similar domains is explored.As a first step, the suitability of out-of-domain data as training data to tag domain examples is assessed in this second subtask. For this, the automatically-annotated versions of the out-of-domain corpora (that is, **HURIC-NHC** and **CMC-NHC**) have been used as training sets and evaluated on **IMH-HC**, assessing the performance of the models trained with these corpora both individually and in combination.

In these experiments, as Table 5.8 shows, rules still achieve the best results, with a 87.74% of F1. As for the training sets without combination of datasets, the models trained with HURIC-NHC obtain much better results than the ones trained with CMC-NHC in both algorithms (66.33% vs 53.17% in F1 for SVM and 71.98% vs 34.75% in F1 for CRF). When combining both datasets, the results obtained by **HURIC-NHC**-trained models in isolation present a slight drop in performance in the CRF algorithm and a slight improvement in SVM (63.33% vs 68.79% F1 in SVM and 71.90 vs 71.70 in CRF).

These results hint that CMC may not be an appropriate candidate for the guidance domain. The possible reasons for this reside in the type of sentences in the corpus, that may not be similar

enough to the domain, and the quality of the automatic translation on longer sentences. While both ML-based methods yield worse results when trained with **CMC-NHC** instead of **HuRIC-NHC**, the performance drop is larger for CRF, with more than 37% in F1. As for SVM, it seems to be more robust, although the performance drop in F1 is still around a 10%. For these reasons, further experimentations have only considered HuRIC for out-of-domain data and CMC has been discarded.

All in all, the first and second subtasks have shown that, given domain examples, machine-learning models obtain very good results when trained with domain data –in the case of CRF, with better results than hand-made rules– and, when trained with out-of-domain data, are capable of obtaining very promising results although, in this case, rules perform better.

Third subtask

The third experimentation subtask is motivated by the fact that, in a context in which natural expressions are to be interpreted, such as in KIDE4I, interactions that have not been considered may appear. This may cause that rules or ML-based methods that work well on domain data do not perform equally well on out-of-domain data.

In this sense, this subtask has evaluated the performance of rules and ML-based methods on out-of-domain data. For this, **HURIC-TS-HC** has been considered as out-of-domain data –and, thus, as test set– since it belongs to a similar domain.

On the one hand, evaluation on the methods that exclusively rely on domain data (that is, rules and ML models trained with domain data) has been performed to check whether the data generated for the domain is capable of generalising over out-of-domain data. The results, which can be observed in Table 5.9, show that rules obtain acceptable results on out-of-domain data (82.03% for F1 and 90.08% of accuracy). However, in this case, ML-based methods perform better. Among ML algorithms, CRF performs best, with a 91.16% in F1 and a 95.15% in accuracy). Thus, it can be stated that, for out-of-domain data, ML models generalise better than hand-curated rules when trained with domain data, favouring an incremental supervised approach.

Nevertheless, since the objective when developing KIDE4I is to

Table 5.9: Out-of-domain data evaluation results using methods that rely on domain data, performed on **HURIC-TS-HC**. For supervised methods, all tree-structure features were used for training.

Algorithm	**Training set**	**Prec**	**Rec**	**F1**	**Acc**
Rules (Baseline)	-	79.87	84.32	82.03	90.08
SVM	IMH-HC	83.55	90.24	86.77	85.57
CRF	IMH-HC	**89.04**	**93.38**	**91.16**	**95.15**

Table 5.10: New data evaluation results using methods that rely on domain and out-of-domain data, performed on **HURIC-TS-HC**. For supervised methods, all tree-structure features were used for training.

Algorithm	**Training set**	**Prec**	**Rec**	**F1**	**Acc**
SVM	IMH-HC + HURIC-TR-NHC	80.80	**90.94**	85.57	86.21
CRF	IMH-HC + HURIC-TR-NHC	**82.08**	**90.94**	**86.28**	**92.34**
SVM	IMH-HC + HURIC-TR-HC	90.03	94.43	92.18	97.06
CRF	IMH-HC + HURIC-TR-HC	**97.92**	**98.26**	**98.09**	**98.98**

follow a LL approach, it is logical to assume that new variants obtained in interactions will be reused to update the existing ML models. In this regard, out-of-domain data can be directly reused –assuming that there may be errors–, or can be manually corrected before adding it to the model. Considering this, given out-of-domain data, the performance of ML models trained with domain data (**IMH-HC**) combined with not-corrected (**HURIC-TR-NHC**) or corrected (**HURIC-TR-HC**) out-of-domain data –and the impact of using one or the other– for training has also been assessed.

First of all, as it can be seen in Table 5.10, CRF achieves the best results in both settings, although when out-of-domain data is not corrected (**IMH-HC** + **HURIC-TR-NHC** for training), the difference between the two algorithms is very small. Also, noisy data affects negatively the performance of the supervised systems (86.28% F1 for CRF with the **HURIC-TR-NHC** subset as out-of-domain data for training *vs* 98.09% F1 for the same algorithm, but with the **HURIC-TR-HC** subset as out-of-domain data for training). In

fact, in this setting it seems that it is better to use a smaller –but correct– training set than to augment it with noisy data (91.16%[10] vs 86.28% for CRF). The best results are obtained when adding the hand-corrected version of HuRIC to domain information (98.09% F1 for CRF).

Comparative Analysis

To qualitatively evaluate the performance of the two approaches for KEE presented in this thesis, a comparison between the chunks obtained by the rule-based approach and by the ML-based one (using a CRF model trained with corrected out-of-domain data, as it is the best supervised approach in the experimentation tasks) is provided. The goal is to analyse which structures cannot be resolved by any of the systems, and the type of structures that the supervised approach is able to solve that the rule-based one is not, and vice-versa.

In this evaluation task, 34 chunks have not been correctly identified and classified by one or another approach or neither of them. By analysing those 34 chunks, it can be observed that, in this case, the rule-based system only performs better in detecting certain destinations formed by single elements of the category `noun` (e.g., *sofa*, *table*, *bed*). On the other hand, CRF is able to solve a certain complex structure that the rules are unable to detect. This structure can be seen in Example (29).

(29) *X ({Adjective}) de {Determiner} Y*, being X and Y words that belong to the category `noun`.
   a. 'Sofá de la habitación' - 'Sofa of the room'[11]
   b. 'Extremo de esta tabla' - 'End of this table'
   c. 'Lado derecho de la cama' - 'Right side of the bed'

Although it seems unlikely to be able to justify the effect of one structure when the difference between the baseline and CRF (hand-rev) is of an 16% in F1 (82.03% *vs* 98.09%), it is important to point out that this structure is fairly common in the dataset, as it represents a 17.64% (more specifically, 6 cases) of the 34 total chunks revised. Furthermore, CRF is also able to tag certain verbs that the baseline

[10] Obtained by using **IMH-HC** to train a CRF model, in Table 5.9.

[11] 'The room's sofa' is preferable, but this translation is closer to the equivalent structure in Spanish.

is not. Again, these verbs represent an important part of the total chunks: a 50% (17 cases).

The chunks that none of the approaches succeed in identifying and classifying are 13. These chunks include, for example, sentences with a complex structure and elements that have not been correctly tagged by the syntactic parser, as it can be seen in Examples (30) and (31), respectively. On the one hand, Example (30a) provides a case of recursive prepositional phrases (a prepositional phrase inside another) and (30b) a command that contains a relative clause. On the other hand, Example (31a) shows a case in which *estar* has been analyzed as a verb (since isolated it is indeed a verb) and not as a part of a nominal whole, *sala de estar* (*living room*).

(30) Complex structures
   a. [*caja de la izquierda de la grabadora*]
   [*box at the right of the tape recorder*]
   b. [*cojín negro que se encuentra en la cama*]
   [*black pillow that is on the bed*]

(31) Not correctly parsed
   a. [*sala de estar*]
   [*living room*]

All in all, and taking into account the results obtained, supervised methods have been proved to be suitable to implement the KEE component from a LL perspective. In this sense, ML-based methods are worth using in this context, although the starting point for all adaptations when no annotated data is available will be the rule-based system. The implementation of a rule-based KEE component will allow to obtain a fully functional initial implementation of the dialogue system and, when sufficient annotated data is available, to obtain a LL-inspired KEE component through the use of ML-based methods.

5.2 Polarity Interpreter

In certain situations, the system needs the user to confirm specific pieces of information and it requests the user to provide a re-

sponse that corresponds to an affirmation or a negation. Since one of KIDE4I's main characteristics is to process natural language commands, affirmations or negations can be provided in various forms other than typical *yes* or *no*. Thus, a component that determines the polarity of the response (that is, if the user responds positively or negatively to a system request), is necessary. This function is supplied by the **polarity interpreter**.

As far as the authors are concerned, the resources that determine if a string is equivalent to *yes* or *no* are scarce –even more in languages other than English– and in a very initial development stage. Considering this, the polarity interpreter makes use of sentiment analysis technologies –which are in a more advanced state– for the target language. For example, in the case of Spanish, which is the target language for the use cases reported in this work, this component implements the library *senti-py*[12], which, given a text, provides a polarity score. Then, the command is classified as positive or negative according to a defined threshold. The output of this module consists on a boolean value that determines if the input has a **positive** (i.e., **1**) or a **negative** polarity (i.e., **0**).

Since this component is only dependent on the target language, it can be reused through use cases as long as the target language is the same.

The next lines will describe the methodology used to define the thresholds for classification.

5.2.1 Methodology for the Definition of Polarity Thresholds

So as to define the threshold that will determine whether a command is of a positive or a negative polarity, a series of tests need to be performed to reduce possible errors to a minimum. These tests consist on classifying several examples using the library chosen to obtain the polarity scores. The examples are divided in three categories:

- **Positive**. Sequences that have a clearly positive polarity. E.g., "yes", "absolutely", "of course". For this type, 7 examples were

[12] `https://github.com/aylliote/senti-py`

Table 5.11: Spanish words and scores for each polarity category obtained with the *senti-py* library and average scores for each category. **(a)** Positive; **(b)** Neutral-positive; **(c)** Negative.

Sequence	Score
sí	0.54
afirmativo	0.54
vale	0.91
correcto	0.73
bien	0.65
claro	0.33
efectivamente	0.23
Average	0.56

Sequence	Score
si tú lo dices	0.54
supongo	0.39
eso creo	0.52
mismamente	0.09
tal vez	0.61
Average	0.43

Sequence	Score
no	0.11
negativo	0.65
en absoluto	0.41
de ninguna manera	0.22
pues no	0.16
nunca	0.25
jamás	0.17
Average	0.28

classified to determine the classification thresholds.

- **Neutral-positive**. Sequences that are positive, but denote a certain neutrality. E.g., "I guess", "if you say so", "I think so". For this type, 5 examples were classified to determine the classification thresholds.

- **Negative**. Sequences that have a clearly negative polarity. E.g., "no", "never", "absolutely not". For this type, 7 examples were classified to determine the classification thresholds.

After defining and classifying each sequence, an average score for each category is obtained. These average scores will help to determine, first of all, the overall performance of the system (if, for example, the average score for negative sequences is higher than the score for the members of the positive category, that would mean that the system is not suitable for this task). Once it has been confirmed that the system does not provide unexpected results in general, the average score for the neutral-positive category will be considered to set the polarity threshold, and the scores for the positive and negative categories will be used to adjust it.

For example, in the context of this thesis, in which the library *senti-py* is used as classifier, three experts were asked to provide sequences for each category, and the most common ones among them were classified using *senti-py*. Table 5.11 shows the classified sequences and their corresponding scores obtained by *senti-py*, along with the average scores for each category. First of all, *senti-py* shows a congruent performance: the average scores for positive sequences

are the highest, followed by the neutral-positive ones, having the negative ones the lowest average score.

By observing the average scores for each category, it can be determined that the threshold to be used to assign a positive or negative polarity to a specific sequence must be around 0.43. So as to adjust this threshold, the results for positive and negative sequences in Table 5.11 should be considered, putting special attention in the higher values for the negative ones. In this case, it can be observed that all negative constructions get very low scores in general, except *negativo* (*negative)* –0.65– and *en absoluto* (*absolutely not*) –0.41–. Since *negativo* is an outlier, it has not been considered for adjustment, but *en absoluto* has. Considering these remarks, the threshold in this case has been adjusted to 0.42.

Nevertheless, it is important to note that this module is on an early stage of implementation and may be improved. In fact, part of the future work of this thesis is set to obtain a more robust polarity interpreter that is capable of learning over time.

5.3 Semantic Repository

The **semantic repository** stores all the information that allows the dialogue manager to function. It contains both knowledge at class level –commonly known as **terminological box** or ***TBOX***– and the individuals that belong to those classes –**assertional box** or ***ABOX*** (Keet, 2020).

The core of this semantic repository –and, hence, the dialogue system– that corresponds to the *TBOX* in this framework is the Task-Oriented Dialogue management Ontology (TODO), presented in Chapter 4.

The *ABOX* stores, for the dialogue management dimension, the instances for (i) the requests and responses that the dialogue system can output to the user, (ii) the processing functions that the dialogue manager must perform given the key elements obtained from a user command and (iii) the implications of the different outputs of these functions. The advantage of this approach is that most instances and relations of this dimension can be reused through use cases and even

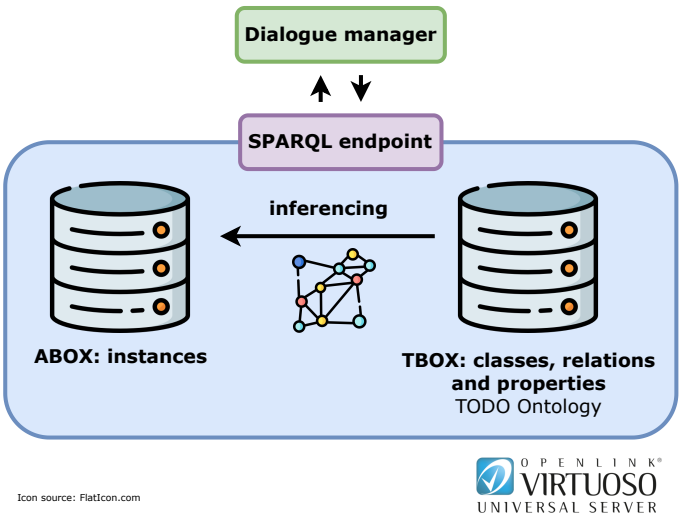


Figure 5.4: Simple overview of KIDE4I's semantic repository.

languages, only requiring translations for requests and responses.

Regarding domain-related knowledge, the *ABOX* contains the instances for (i) the modelling of all the world elements of the domain and their relations, (ii) the different actions that can be performed by the target system, along with their arguments, (iii) the world elements that could belong to those arguments, (iv) the different variants to refer to both domain elements and actions in the target language and (v) the target-system-readable equivalents for world elements and actions. Finally, the *ABOX* is also the destination for the traces generated in each interaction with the dialogue system.

The *TBOX* serves as the source for the inferences that can be performed on the *ABOX* according to the knowledge in the *TBOX*. For example, consider a member of the class `Action`, which is related to its corresponding `Arguments` through the properties `hasCoreArgument` and `hasOptionalArgument`. In this scenario, it is possible to obtain the whole set of arguments of that `Action` –without making distinctions on whether that `Argument` is required by the `Action` or not– through the superproperty `hasArgument`, as it is inferred that the `Actions` and `Arguments` related through `hasCoreArgument` and `hasOptionalArgument` are also related through `hasArgument`.

To sum up, the semantic repository determines the outputs the dialogue system will show to the user, the flow of the dialogue and

allows the dialogue manager to interpret user commands to obtain a readable input for the target system. This means that the total control of the dialogue process depends on semantic-technology-based resources. Figure 5.4 shows a simple overview of the semantic repository component.

5.3.1 Implementation

Considering the description above, and regarding the implementation of the semantic repository for KIDE4Guide and further adaptations, two main tasks may be devised: domain information must be modelled and instantiated, as well as dialogue-related information. In this sense, domain information is strictly related to the use case at hand and dialogue information is intended to be generic and reused among use case adaptations.

As for the domain, for the *TBOX* the corresponding classes for the spaces, objects and people from Tekniker have been modelled in TODODW according to a set of CQs defined to comply with the necessities of the use case. For this, terms from the GEO (Brickley, n.d.) and FOAF (Brickley & Miller, n.d.) ontologies have been reused. The resulting modelling of TODODW, with nine classes, can be seen in Figure 5.5. In total, the domain *TBOX* for KIDE4Guide includes 43 classes.

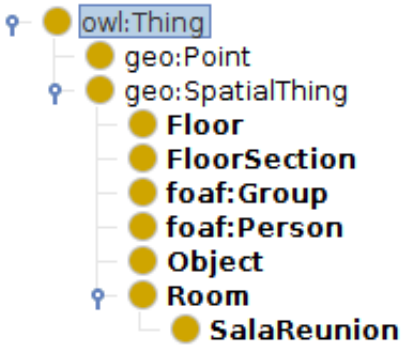


Figure 5.5: Classes in TODODW for KIDE4Guide.

Regarding the *ABOX*, the instances of the classes included in TODODW have been extracted from existing databases and instantiated automatically through ODBA mapping rules, and the relations

between them, modelled. To perform the mappings, three different types of information are necessary: a prefix declaration (in the same way as in SPARQL queries), the set of parameters of the database and, finally, the rules. Each rule consists of two elements: the first one is a parametric triple that accounts for the instance to be created from the information in the database. As for the second element, it contains the SQL command to retrieve the necessary information from the database, which will be used to fill the parametric values in the previously defined triple. Examples of these rules can be seen in Example (32). The rule in Example (32a) has been defined to map the people in the database as members of the class `foaf:Person` and the one in Example (32b) has been defined to generate each person's email address and set it as a property of the person's individual.

(32) a.
```
target :{id} a foaf:Person .
source select id from personas
```
b.
```
target :{id} foaf:mbox {email} .
source select id, concat(email,"@") as email
from personas
```

Furthermore, the actions that can be performed by the system have also been modelled and, for frame-related information (frames, frame heads and related lexical units), the population strategy in Chapter 4.6.1 (which will be referred as *the population strategy* from now on), which makes use of multilingual existing language resources, has been used. With this strategy, in total, 29 frames, 62 frame heads and 258 lexical units have been semi-automatically instantiated. As for the machine-readable information for each element in the domain, it has been modelled considering the requirements of the robot and its components. Finally, the different lexical units corresponding to the different domain elements that could not be obtained through the automatic methods mentioned above have been obtained by combining thesauri and expert knowledge[13]. In total, the domain *ABOX* includes 604 instances, 449 of which have been obtained through automatic or semi-automatic methods.

For the dialogue, the *TBOX* has been modelled to include the expected outcomes of possible interactions (i.e., `SystemRequest`s and `SystemResponse`s), generically enough to be reused in other use case adaptations. In total, the dialogue *TBOX* includes 80 classes.

[13]E.g., *tv* or *tele* for *televisión* (*TV*).

Table 5.12: Classes and individuals for the KIDE4Guide implementation.

	Classes	**Individuals**
Dialogue	80	110
Domain	43	604 (449)
Total	124	714 (449)

The *ABOX* includes the instances for each of the potential interactions and the instances corresponding to the tasks to perform by the dialogue manager (i.e., `StepFunctions`). In total, the dialogue *ABOX* includes 110 individuals.

The information on the modelling and instantiation of the ontology for this use case implementation is shown in Table 5.12, in which the values in parentheses correspond to individuals obtained through automatic (or semi-automatic) methods. As it can be observed, more than 70% of the instances for the domain have been obtained automatically or semi-automatically.

5.4 Dialogue Manager

The objective of KIDE4I's dialogue manager is to manage the dialogue process and to generate and instantiate the corresponding dialogue-related traces. To do so, this component consists of two REST services: ***init*** and ***userInput***, the logic of which can be seen in Figure 5.6. The first one generates a dialogue identifier and retrieves from the semantic repository all the necessary information to initiate a dialogue with the user, such as the first step of the dialogue process. Usually, this *init* service fetches the initial system request for the user (e.g., when the dialogue is modelled to initiate the dialogue with a greeting or a request for user input).

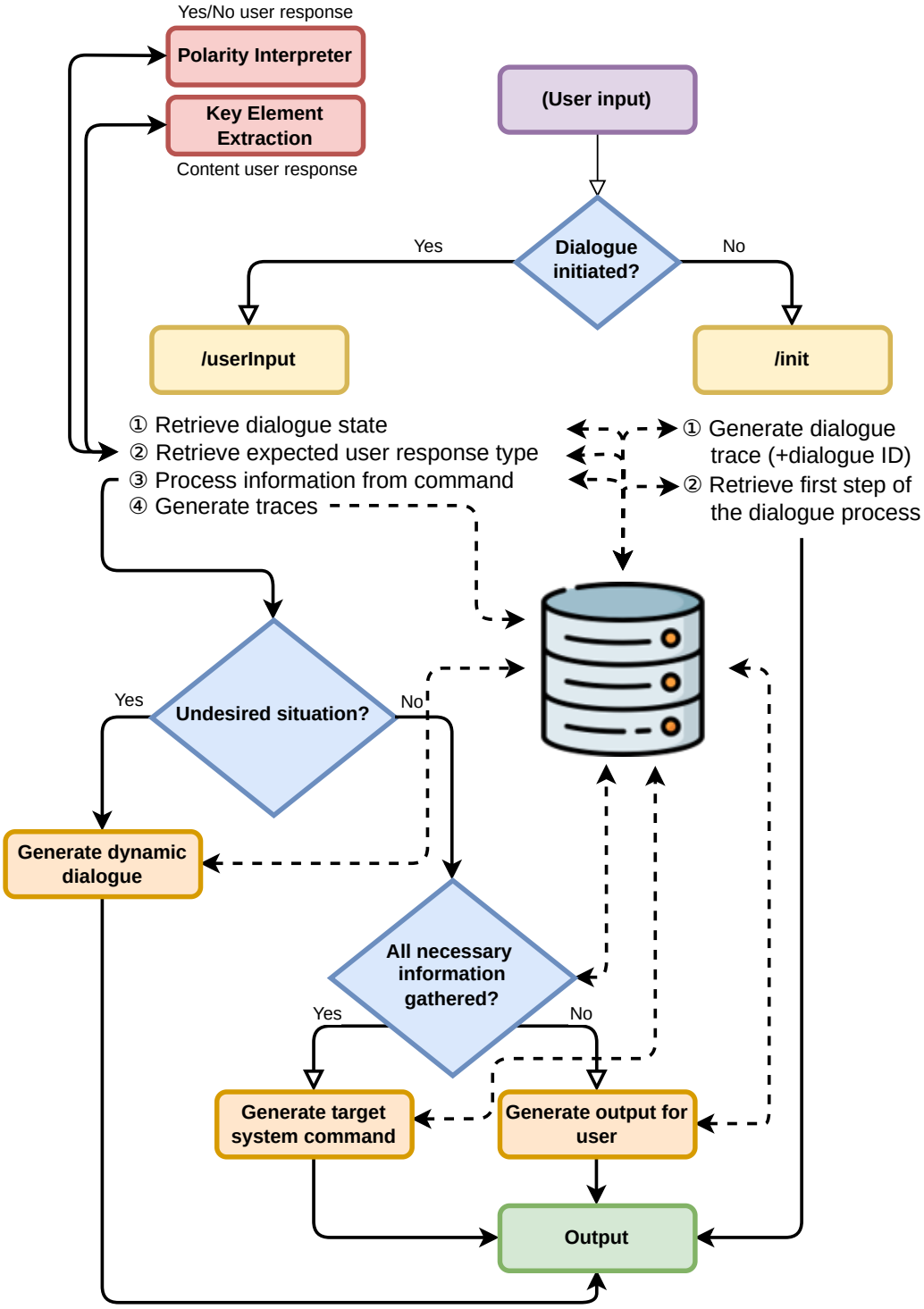


Figure 5.6: Overview of the Dialogue Manager component logic.

The ***userInput*** service is the target service for each user interaction. Given a user input, the dialogue manager relies on the knowledge in the semantic repository to determine of which type must be considering previous system output –either a *yes/no response* (i.e., *yes* or *no*) or a *content response* (i.e., a response that conveys a certain information). Depending on that knowledge, this component calls the polarity interpreter or the KEE modules to obtain an interpretation, respectively. Once an interpretation is obtained, the dialogue manager relies again on the semantic repository to determine the next function to execute or the action or world elements the user is referring to, and assert whether the information obtained is consistent or sufficient to obtain a readable command for the target system.

When the dialogue manager has checked with the semantic repository that all the information necessary to obtain a target-system-readable output is gathered, it generates the command for the target system in the corresponding format by exploiting, again, the information in the semantic repository.

Considering the strong dependency on the semantic repository, a set of functions have been defined in the dialogue manager to retrieve the necessary information from the ontology. These functions include the SPARQL query to execute according to their objective. Once more, these queries have been defined so as to be as generic as possible, leaving changes to be performed only if the particularities of the use case require so, enhancing the reusability of the component in different use cases. The defined functions supply the following actions:

- **Functions related to command interpretation:**
  - **Check the compatibility of a given target with the verbal key element** (if present). To do so, a SPARQL is executed to obtain two pieces of information: (i) whether a world element can be obtained from the target's everygrams[14] (through a full-text search with `bif:contains`) and to which `WorldElementGroup` belongs and (ii) whether a `FrameHead` can be obtained from the verbal key element's

[14] All possible ngrams for a sequence. More information can be found in `https://buildmedia.readthedocs.org/media/pdf/nltk/latest/nltk.pdf`.

everygrams (through full-text search) and the `WorldElementGroups` that apply to said `FrameHead` (through the property `appliesToFrameHead`). With this information, it is checked that both world element and `FrameHead` have `WorldElementGroups` in common. If so, both the verbal and target key elements are compatible. If the query returns any of the elements but are not compatible, the system can continue with one of them and, if the query returns no results, it means that the system cannot continue because it was not able to resolve any of the two elements.

For example, consider a guiding scenario. In it, there exists a platform in which screwing and deburring can be performed and the modelling of the world is the following:

* World element groups and world elements:
  - `Fasteners`: `screw, nail, bolt`.
  - `Materials`: `steel, aluminium`.
* `FrameHeads`:
  - Screw
    → `Fasteners appliesToFrameHead Screw .`
  - Debur
    → `Materials appliesToFrameHead Debur .`

If, given this modelling, the user wants to go to this platform by asking for a place to screw/debur, four situations may arise:

* Both elements are in the scenario modelling and are compatible: "I want to *screw* a *nail*" / "I need to *debur steel*". The system will continue.
* Both elements are in the scenario modelling and are not compatible: "I need to *screw aluminium*". The system will ask the user which one should it continue with.
* Only one of the elements is in the scenario modelling: "I need to *screw* the *door*". As only `Screw` is present in the scenario modelling, the system will continue with `Screw`.
* None of the elements is present in the scenario modelling: "I have to *paint* the *door*". As *screw* nor *door* are present in the scenario modelling, the system can-

not continue and will request the user to reformulate their command.

- **Get the `Intent` and `Action` from the key elements** extracted from the user command. Two approaches can be considered:
  - **Get `Intent` and `Action` from verb**. In this function, the SPARQL query performs a full-text search to obtain a `FrameHead` through the `IDval` of its `Lexical Unit`. From the `FrameHead`, the `Frame` is obtained, which is used to get the `Intent` and, from it, the `Action`.
  - **Get `Intent` and `Action` from target** (e.g., the `Intent` and `Action` could not be obtained from the verb or no verbal key elements have been uttered and/or detected by the KEE component). In this function, the everygram for the target (in case it consists of more than one word) is generated. This everygram is used in the SPARQL query to get a `LexicalUnit` from its `IDval` through full-text search. From this `LexicalUnit`, the corresponding world element(s) and `WorldElementGroup`(s) are obtained. Then, the query checks the `Argument`s that the `WorldElementGroup`(s) applies for and, finally, the `Intent` and `Action` are obtained. Depending on the dialogue step, it is possible to filter the `Action`s to obtain (e.g., the dialogue system has many `Action` candidates and asks the user for the target to determine which of the candidates is the user referring to).
- **Obtain the `Arguments` for an `Action`**. In this function, the SPARQL query, given the `Action` URI, gets the `Argument`s for that `Action`.
- **Assign a given non-verbal key element (i.e., target) as the value of an `Argument`**. If this assignation is successful, the `TargetSystemReadableInfo` for both the `Argument` and the assigned value are obtained. In this function, a similar procedure than to get the `Intent` and `Action` from the target is followed: the everygram of the target is generated, and it is used to obtain, through the SPARQL query, its corresponding world element or `WorldElementGroup` and, finally, the `Argument`. In this

case, the corresponding `Action` is filtered to avoid assignation to `Argument`s belonging to other `Action`s. Depending on the dialogue step, it is possible to filter the `Argument`s that apply (e.g., the dialogue system is missing the value of a specific `Argument` and asks the user to provide its value).

- **Functions related to dialogue management:**
  - **Get the first action to perform by the dialogue system** (e.g., the first `DialogueStep` or the first `System Request`). In this function, the implications for the different outcomes of the first action are also retrieved.
  - **Get the next action to perform by the dialogue system** (e.g., the next `DialogueStep` or the next `System Request`). In this function, the implications for the different outcomes of the next action (also including `System Response`s or `SystemRequest`s) are also retrieved.
  - **Get the dialogue state from the ontology**.
- **Functions related to dialogue traces:**
  - **Insert a `DialogueTrace`, along with the initialization time**.
  - **Insert a `UserRequestTrace`, along with its corresponding `KeyElementTrace`(s)**. The generated `User RequestTrace` is associated to the current `DialogueTrace`.
  - **Insert an `ActionTrace` as a possible action that applies to a `KeyElementTrace`**. This `ActionTrace` is related to its corresponding `Action`.
  - **Insert the `ActionTrace` corresponding to a given `Key ElementTrace` and relate both of them**. This `ActionTrace` is also related to its corresponding `Action`.
  - **Insert an `ArgumentTrace` as possible argument for an `ActionTrace`**. Additionally, a `WorldElementTrace` is created as a possible value of the `ArgumentTrace`. This `WorldElementTrace` is related to its corresponding world element.

– **Insert a `SecondaryDialogueTrace`, related to a specific `DialogueTrace` and to its corresponding `System Responses` and/or `SystemRequest`**. The initialization time is also modelled.

As the functions related with dialogue management and traces rely on a stable set of classes and relations in TODODial, these functions can be reused among use cases. However, the functions related to command interpretation rely on the modelling of the domain, which is more variable due to the strong dependence of TODODW with the domain. For this, the function structure and the SPARQL queries in these functions are potentially subject to adaptations considering how the domain is modelled, especially when individuals instantiated in TODODW are involved.

Additionally, the dialogue manager has been designed to be capable of gathering user feedback through the generation of dynamic dialogues. These dynamic dialogues will be triggered when users find themselves in undesired situations when using the dialogue system (e.g., the dialogue system has obtained an incorrect interpretation of a user command). Through these, it is possible to increment the knowledge stored in the semantic repository (and, more specifically, in the domain-related section of the *ABOX*) to allow an incremental improvement of the whole dialogue system. When this feedback is gathered, the new information must be processed and the ABOX updated so that it is exploitable by the dialogue system. In this sense, the effort in this setting is directed towards obtaining a dialogue system that is capable of learning continuously ensuring robustness and reliability. This functionality will be implemented in KIDE4I's initial implementation and adaptations as part of future work.

5.4.1 Implementation

Considering that in most cases the functions for dialogue management and tracing are common in use cases, these have been implemented in KIDE4Guide as described in the lines above.

However, the command-interpretation-related functions do present some particularities that are related with the use case. These particularities are bound to the characteristics of the key elements. For

this, the information regarding containing and contained elements (e.g., "a room *with a PC*") is considered to obtain the `Intent` and `Action` from the target and to assign a key element as the value of an `Argument`. Most modifications in these functions are related to the structure of the world elements in TODODW and the set of key element types for the use case.

So as to keep a dialogue with the user, the logic implemented in KIDE4Guide for the dialogue process is described in the following lines, as well as in Figure 5.7. First of all, the dialogue is initiated by the system, which greets the user and requests for their command. Once the user has provided their command and it has been interpreted by the system, the dialogue manager determines whether all the necessary information for the target system to perform the action requested by the user has been obtained. If there is information missing, the system returns to the user and asks for the specific information to provide. If all information has been gathered, the dialogue manager checks if any information has been inferred from the user command (i.e., it has not been specifically asked to the user). This check is motivated by the fact that information that has not been specifically provided by the user (and thus, inferred by the system) is more prone to interpretation errors than information asked directly to the user (e.g., the system asks the user to provide the destination), as the latter has been designed to limit the results of its interpretation (in the case of the previous example, the argument to assign a value to). Due to this, if there is any inferred information, the system will ask the user for confirmation that the command to send to the target system is correct. If it is not correct, the system will ask the user to check the assigned value of each action argument and correct if necessary. If it is correct –or the values for the action arguments have been corrected–, the system will send the command to the target system for execution. Finally, the system will ask the user if they have any more requests. If the user answers negatively, the dialogue process will finish. If the user response is positive, the system will ask the user for their command and the dialogue process will restart.

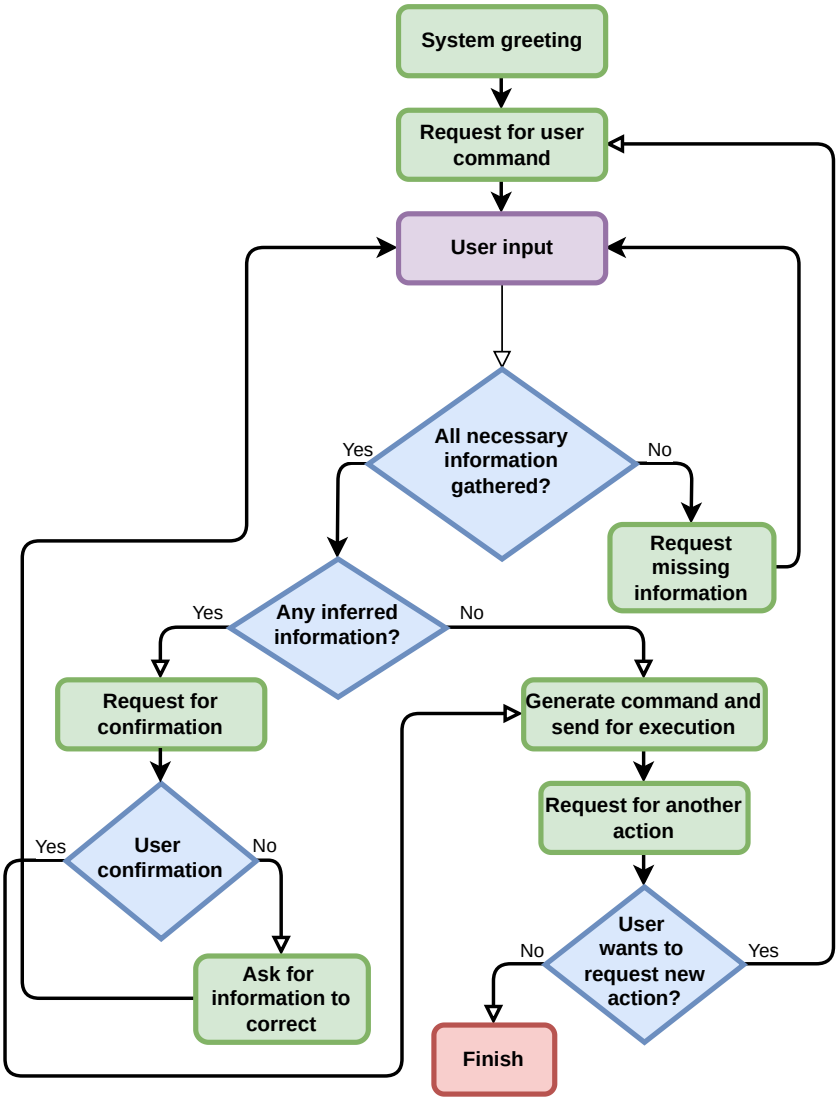


Figure 5.7: Dialogue Manager interaction logic in KIDE4Guide.

5.5 Conclusions

This Chapter describes the Knowledge-drIven Dialogue framEwork for Industry (KIDE4I), the generic semantics-based task-oriented dialogue system for industrial scenarios developed in the context of this thesis. The use of TODO as the core ontology for KIDE4I and the development of each of its components from a generic perspective contribute to tackle the main limitations of current approaches to task-oriented dialogue systems for industry[15], namely the lack of training data to use state-of-the art techniques such as DL, restrictions in natural language communication and high specificity, which limits their capacity to be reused. Furthermore, it does not require big amounts of data to be implemented, as reuse of existing data has been encouraged both in this Chapter and throughout this thesis.

KIDE4I consists of four modules: the key element extraction (KEE) component, the polarity interpreter, the semantic repository and the dialogue manager. The design process for each of them is reported in this Chapter. Furthermore, these have also been implemented for a guidance scenario to have an initial implementation of KIDE4I –KIDE4Guide– with the objective of reusing the resources generated in future adaptations and to evaluate, in the case of the KEE component, the approaches explored.

The design of the KEE component has been tackled as a NERC problem due to the resemblance of both tasks, which can be performed through the use of rules or ML-based methods, taking into account the limitations of the latter regarding available training data. In this context, this thesis has also contributed with a strategy to perform semi-supervised KEE by augmenting domain training data with data from different but similar domains, coping with the scarcity of such resources.

From the validation and evaluation of the rule-based and the ML-based KEE component in KIDE4Guide, as well as the strategy above, the following conclusions have been extracted:

- To annotate domain data:
  - ML-based KEE performed with models trained with do-

[15] *vid.* Chapter 1.

main data are able to obtain similar results as the rule-based implementation. In fact, the models trained with the CRF algorithm obtain better results than rules.

- ML-based KEE performed with models trained with out-of-domain data obtain promising results. However, they do not achieve the performance of models trained with domain data and it is very important to assess the quality of the out-of-domain training data, as it may considerably affect the results.

- To annotate out-of-domain data:
  - Domain-based methods (rules and ML models trained with domain data) achieve good results.
  - The evaluation of augmentation of domain data with out-of-domain data depends on whether out-of-domain data is corrected (considering that domain data is always correct):
    * Best results are obtained when out-of-domain training data is corrected, better than models trained with only domain data.
    * If out-of-domain data is not corrected, the performance in comparison to corrected data drops, and, in that case, domain-based methods obtain better results.

- Among the supervised algorithms evaluated, CRF performs better than SVM.

Considering this, the methodology to implement the KEE component is to first use Foma rules as starting point which, besides allowing a full functionality of the component with good results, contribute to the generation of domain training data towards the development of a ML-based KEE component.

The polarity interpreter has been implemented by making use of sentiment analysis technologies given the scarcity of specific resources for its development, specially for languages other than English. For this component, a set of thresholds that determine whether a command has a positive or a negative connotation have been defined to ensure a correct polarity interpretation of commands. To obtain these thresholds, a methodology has been presented and used, with

the advantage that it should only be performed when the target language is changed, as the task this component deals with is not related to the domain *per se*. Thus, implementations of this component can be reused as long as the target language stays the same. The task of revisiting the design of this component to ensure its robustness is encouraged, considering the technologies used and their development stage, in line with the fact that this component is out of the scope of this thesis.

The semantic repository includes both the terms and relations ($TBOX$) and individuals ($ABOX$) necessary for the dialogue manager to function. The core of the $TBOX$ is TODO, and the instances in the $ABOX$ are instantiated according to the use case necessities. To ease the instantiation process, the following actions are distinguished:

- The reuse of instances from the KIDE4Guide (especially for the dialogue component).
- The automatic instantiation of world elements from existing databases or other resources, where possible.
- The use of the strategy to semi-automatically populate the domain ontology with intent-related information from existing multilingual linguistic repositories presented in Chapter 4.

Finally, the dialogue manager orchestrates the other modules to obtain, from a user instruction, an executable command to be directed to the target system. This component is fully supported on the logic in the semantic repository, and the functions necessary to exploit the knowledge in it have been defined in KIDE4Guide to be reused in future adaptations. These functions and the SPARQL queries in them may only require small adjustments to cover the particularities of the use case.

To wrap up, the generic design of KIDE4I is reflected in each of its modules, which is reinforced by the fact that, given an initial, generic implementation, the effort to adapt KIDE4I to other use cases is considerably reduced, mainly motivated by the use and reuse of existing ontologies and resources. This makes KIDE4I an interesting asset for the implementation of task-oriented dialogue systems in industrial scenarios.

Chapter 6

KIDE4I in Use

To reduce the effort when adapting KIDE4I to different use cases, a methodology that wraps up all the techniques reported throughout this thesis for that matter has been designed. Having as starting point KIDE4Guide –the initial implementation of KIDE4I for the guidance scenario–, described in the previous Chapter, this adaptation methodology is put in practice through its application in three different use cases that are relevant to industrial settings for the Spanish language: a **bin-picking robot**, a computerised **maintenance management software** (CMMS) and an **assistant for maintenance procedure execution**. As noted previously, the guide robot implementation has set some of the bases for these adaptations, such as the basic rules for the key element extraction component, as it will be described later on.

The guide implementation and bin-picking and maintenance procedure assistant adaptations have been validated and evaluated within three user studies[1], the results of which have been reported in terms of qualitative and quantitative analysis. For qualitative evaluation, the SASSI questionnaire (Hone & Graham, 2000) has been used to evaluate the adaptations through different perspectives. Regarding quantitative evaluation, the dialogue completion rate, the number of turns to successfully complete a dialogue, the number of errors according to an identified set and response time in both user studies have been considered.

[1]The CMMS adaptation is at an integration stage with the CMMS software that is external to the development of this thesis.

6.1 Use Case Adaptation Methodology

The process of adaptation of KIDE4I to different use cases consists of four steps, basically based on the different components that are at play in the architecture of the dialogue system. It is important to remark that KIDE4I's components are designed to be language-independent and, thus, this same process applies to all languages:

1. **Characterisation of the use case**. This preliminary step allows the developer to identify the necessities of the use case in order to be applied to the different modules of the dialogue system. This necessities include the type of interactions to be solved, the elements included in the domain, the key elements that refer to them and their possible syntactic structures, the target system's functionalities to be identified and the different situations (defined through frames[2]) that apply to each functionality in the use case.

2. **Modelling of the key element extraction component**. After having identified the key elements to be extracted, two main steps can be distinguished to obtain a functional KEE component:

   - **Definition of Foma rules** to delimit the structures that correspond to the key elements –defined in the previous step– from the command's syntactic tree. The rules defined in KIDE4Guide may serve as a starting point, as they define basic structures that are common in industrial scenarios.
   - **Adaptation of the post-processing subcomponent** to obtain the set of relevant key elements to be used as input for the dialogue system.

3. **Ontology modelling and instantiation**. In this step, the necessary information to model the use case must be identified –through CQs– and instantiated into the TODO ontology. This step follows different phases, which are closely related to the different TODO modules:

[2] *vid.* Chapter 4.6.1.

(a) **Modelling and instantiation of the domain** (TODODom). This phase is associated to two main blocks of knowledge, both related to the domain of the use case: world elements and action- and frame-related elements.

- **World elements** (TODODW). Given that the classes in TODODW are highly dependent on the use case[3], in this phase the domain elements are modelled and instantiated: objects, people, machines, spaces, etc., along with the relations that are relevant for the use case (e.g., a given workshop contains a given machine).
- **Frame- and action-related elements** (TODODFA). The frames and related information required to successfully identify and process actions (e.g., arguments) are instantiated. To obtain a significant part of this information (frames, frame heads and lexical units), the strategy described in Section 4.6 may be used.

For each of these blocks, the machine-readable information for the target system to perform such actions (i.e., `TargetSystemReadableInformation`) is also instantiated, along with the different words to refer to them in natural language when directing a command to the system (lexical units). These words are mostly obtained through automatic methods that rely on existing resources from the natural language processing field or database information and, when necessary, manually.

(b) **Modelling and instantiation of information related to dialogue management** (TODODM). This instantiation phase consists on the definition of the logic implications of the different outcomes of each interaction between the system and the user. In this stage, the responses and requests of the dialogue system are also defined. This information can be reused from the KIDE4Guide implementation and, if necessary, new elements must be modelled.

4. **Adaptation of the source code of the dialogue manager**. Although the source code is intended to be generic, it still needs minimal modifications that deal with the particularities of each use case, which are basically two: key element processing (different use cases may have different configurations of key ele-

[3] *vid.* Chapter 4.

ments) and, if necessary, ontology queries to correctly interpret the commands directed to the dialogue system. These particularities, as noted in Chapter 5.4, are encapsulated in functions that have been designed to be easily adaptable to simplify this task. Further modifications are also needed when additional functionalities are needed, considering that the most typical ones are already defined.

The following sections will describe the adaptation process for each of the use cases previously mentioned.

6.2 Use Case Adaptations

In industrial scenarios, the most typical interactions for workers are, on the one hand, with information systems that retrieve information about maintenance tasks or access specific information such as blueprints or technical manuals. On the other hand, workers also take part in collaborative tasks with robots, in which both work together towards completing an assignment (Romero et al., 2016).

In this thesis, three different KIDE4I adaptations have been carried out for three use cases that are relevant in current industrial setups: a bin-picking robot, a Computerised Maintenance Management Software (CMMS) and an assistant for maintenance procedure execution, which are described in the following sections. These adaptations have been performed by using KIDE4Guide, described in Chapter 5, as starting point.

6.2.1 KIDE4BinPicking: Bin-Picking Robot

KIDE4BinPicking is the result of adapting KIDE4I for interaction with a bin-picking robot (in Figure 6.1) that is able to pick different printer cartridges from a table, identify their brand and colour and classify them between two different containers according to the criteria established by the operator (that is, whether the cartridges of a specific colour or brand should be placed in a container or another). It is also possible to interact with the system through gestures to convey the destination container or cartridge or to make the robot stop

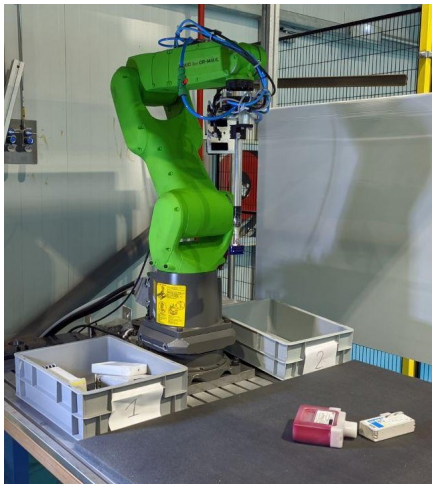


Figure 6.1: Bin-picking robot the dialogue system has been adapted for.

or continue. In this sense, gestures can be complementary to voice commands. As in KIDE4Guide, the target language for interaction is Spanish.

The key elements to extract are **actions** and **targets** –which may correspond to **brands**, **colours** and **containers**–, as can be observed in Example (33). Furthermore, the key element tagger is also expected to detect **pointers** that may imply the presence of a gesture referred to a container or cartridge, such as "here" or "this", as Examples (34) and (35) show.

(33) **Pon**$_{action}$ el **azul**$_{target-colour}$ en el contenedor **1**$_{target-container}$
Put$_{action}$ the **blue**$_{target-colour}$ one in container **1**$_{target-container}$

(34) **Pon**$_{action}$ **este**$_{pointer-cartridge}$ en el contenedor **1**$_{target-container}$
Put$_{action}$ **this**$_{pointer-cartridge}$ one in container **1**$_{target-container}$

(35) **Pon**$_{action}$ el **azul**$_{target-colour}$ **aquí**$_{pointer-container}$
Put$_{action}$ the **blue**$_{target-colour}$ one **here**$_{pointer-container}$

An initial analysis of the data generated for KIDE4Guide showed that most definitions for its KEE component could be reused, and only small modifications on some definitions and rules for KEE have been necessary, especially to include pointers. For the post-processing

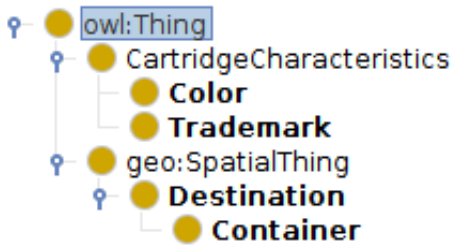


Figure 6.2: Classes in TODODW for KIDE4BinPicking.

step in the KEE component, only minimal adaptations have been required. This reuse of definitions has drastically reduced the time to obtain Foma rules which, added to the generality of the interpreter, has allowed to obtain a functional KEE component in a reasonable amount of time.

The modelling and instantiation of the world elements from the domain has been performed in terms of the colours and brands that can be recognised by the robot (*cyan*, *magenta*, *black* and *yellow* and *Epson*, *Canon*, *HP* and *Brother*, respectively) and the containers in the scenario (*1* and *2*). For cartridge colours, the Printer Vocabulary ontology (Rodriguez-Castro, Torok, & Hepp, n.d.) has been reused, as well as the GEO ontology. The classes in TODODW for this adaptation can be seen in Figure 6.2. As in the guide robot use case, actions have also been instantiated and frame-specific information has been obtained by following the population strategy, with which a total of 19 frames, 62 frame heads and 141 lexical units have been semi-automatically instantiated. Machine-readable information has been modelled according to the robot's requirements and lexical units have been obtained from linguistic resources (through thesauri and the strategy) and expert knowledge.

Tables 6.1 and 6.2 show the number of classes and instances for each KIDE4I adaptation, respectively, along with the number of reused or automatically-obtained elements. According to these Tables, in which *BP* stands for KIDE4BinPicking, a 96.6% of the classes for this adaptation have been reused, and a 78.4% of the instances have been obtained automatically, which means that the effort to model and instantiate the ontology has been highly reduced. Interestingly enough, the original dialogue management classes and instances have also been reused, although minor modifications had to be performed in the dialogue logic.

Table 6.1: Classes for each KIDE4I adaptation: total and reused from other resources (TODO included).

	Dialogue		**Domain**		**All**	
	Total	**Reused**	**Total**	**Reused**	**Total**	**Reused**
BP	80	80 (100%)	40	36 (90%)	120	116 (96.6%)
CMMS	80	80 (100%)	37	37 (100%)	115	115 (100%)
Assistant	82	80 (97.6%)	134	132 (98.5%)	216	212 (98.1%)

Table 6.2: Instances for each KIDE4I adaptation: total and obtained automatically.

	Dialogue		**Domain**		**All**	
	Total	**Auto**	**Total**	**Auto**	**Total**	**Auto**
BP	80	80 (100%)	278	201 (72.3%)	358	281 (78.4%)
CMMS	80	80 (100%)	11711	11644 (99.6%)	11791	11744 (99.6%)
Assistant	80	80 (100%)	584	516 (88.4%)	664	596 (89.8%)

6.2.2 KIDE4CMMS: Information Systems for Maintenance Management

The second use case consists on interaction with a Computerized Maintenance Management System (CMMS). By using this software, which is typically used to manage maintenance actions, users are able to access maintenance-related information, such as work orders or blueprints.

KIDE4I has been adapted so as to track work orders, request for blueprints, problem solving protocols or exploded views and check stock for a specific machine or machine component through natural language commands in Spanish. Furthermore, users can also fill forms on the system's request. The result is KIDE4CMMS.

So as to interpret each of the commands directed to the system, the key elements to extract are **actions**, **targets** and **items**. As Example (36) shows, *targets*, along with *actions*, determine the action to perform –in this case, to show a work order–, whereas *items* are the action arguments –the work order identifier and the machine that work order is for.

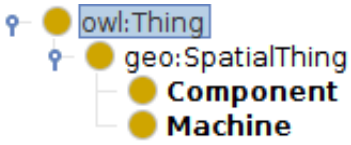


Figure 6.3: Classes in TODODW for KIDE4CMMS.

(36) a. **Muéstrame**$_{action}$ la **orden de trabajo**$_{target}$ **85**$_{item}$ de la **fresadora**$_{item}$

b. **Show**$_{action}$ me the **work order**$_{target}$ **85**$_{item}$ for the **milling machine**$_{item}$

As in the previous cases, KIDE4Guide's definitions have been reused, and only six specific rules have been defined for this use case. To cover the different key element tags, the interpreter has been modified to include them.

For the domain modelling and instantiation, the available error codes –for the problem solving protocols–, machines and components have been modelled and instantiated, along with the IDs for work orders, which have been defined as numerical patterns to be checked by the target system. In this case, the classes for machines and components have been aligned with the VAR ontology (Fernández, Casla, Esnaola, Parigot, & Marguglio, n.d.). The classes in TODODW for this adaptation can be seen in Figure 6.3.

Machines, components and their corresponding lexical units, which are in total more than 10,000 instances, have been automatically instantiated from the target system's database. Examples (37) and (38) show a sample of the machines and components instantiated in the ontology, respectively.

(37) a. "Cortadora aluminio 602"

b. "Elevador de personas Manitu"

(38) a. "Ventilador 165232"

b. "Bobina BO-O320"

Each action the system is able to perform has been modelled, and the lexical units to identify targets and the rest of the elements of the scenario have been instantiated as in previous cases, both reusing

existing lexical resources and linguistic knowledge. Following the CMMS' requirements, target-system-readable information has also been modelled. Finally, action- and frame-related data have been instantiated by following, again, the population strategy, which has enabled the semi-automatic instantiation of 4 frames, 4 frame heads and 12 lexical units.

Finally, as Tables 6.1 and 6.2, in page 153, show, all classes for this use case have been reused, and a 99.6% of the instances have been obtained through automatic methods. For the dialogue part, the original modelling and instances have been totally reused.

6.2.3 KIDE4Assistant: Information Systems for Maintenance Assistance

The third and last KIDE4I adaptation in this thesis, KIDE4-Assistant, is an assistant for maintenance procedure execution. Given a set of maintenance procedures, previously extracted from technical manuals, the system is able to guide the user through the processes described in them. KIDE4Assistant has been designed for Spanish, and manuals are also in this language.

In this use case, procedures are structured in *methods*, *tasks* and *steps*. *Methods* determine different ways to perform the same procedure (e.g., in normal conditions or in a clean room); each *method* has a set of *tasks* (e.g., extract a battery, install a battery), and each *task* consists of a set of *steps* (e.g., disconnect the machine, open the lid). Given a procedure, KIDE4Assistant requests the user to select the method to follow (in the case the procedure has more than one). Then, the system gives the description of the current method, task or step. Users are able to navigate through the different elements of the procedure by (i) requesting for the next or previous step or task (given the current step), (ii) to repeat the information that was just given by the system, (iii) to restart a method or task (i.e., start over again from the first step of the first task of the current method or to start over again from the first step of the current task, respectively), (iv) to obtain other related information such as the list of necessary tools to perform the procedure or (v) a more extensive description or

(vi) additional information[4], in the form of text and images. Texts and images are shown in a screen so users can easily follow the information provided by the system.

Considering the description above, the key elements to be identified by the target system are **actions** and **targets**, which correspond to the keyword used to determine the action to perform (Example (39), for the *Show tool list* action) or the reference element of the action (Example (40)).

(39) **Muéstrame**$_{action}$ la lista de **herramientas**$_{target-determineAction}$
Show$_{action}$ me the **tool**$_{target-determineAction}$ list.

(40) **Reinicia**$_{action}$ la **tarea**$_{target-reference}$
Restart$_{action}$ the **task**$_{target-reference}$

In this case, as the key elements to identify are common with the guide use case ones, both definitions, rules and interpreter scripts are also common.

For the domain ontology modelling and instantiation phase, 6 procedures –from 2 robotic arms and a controller– have been automatically instantiated into TODODW. To do so, these procedures have been extracted from their corresponding manuals and formatted as JSON files, from which procedure information has been obtained. From this JSON, the relations between *procedures*, *methods*, *tasks* and *steps* are instantiated, including sequential relations (e.g., *Step* 2 comes after *Step* 1 and before *Step* 3). Furthermore, for each method, their tool list – with its corresponding tool individuals– has been instantiated. For each structural element of the procedure (i.e., *methods*, *tasks* and *steps*), additional information and/or extended information has also been automatically included. Finally, the elements that make reference to procedure parts –*procedures*, *methods*, *tasks*, *steps* and *tools*, which correspond to the key elements labelled as *target*s– have also been included. So as to be able to instantiate this information, the VAR ontology (Fernández et al., n.d.) –which is intended as a "workplace digital twin" (Fernández et al., n.d.) by representing workplaces, processes and workers– has been reused. The set of classes for KIDE4Assistant can be seen in Figure 6.4.

[4]To be requested through general questions such as "Can you give me more information?".

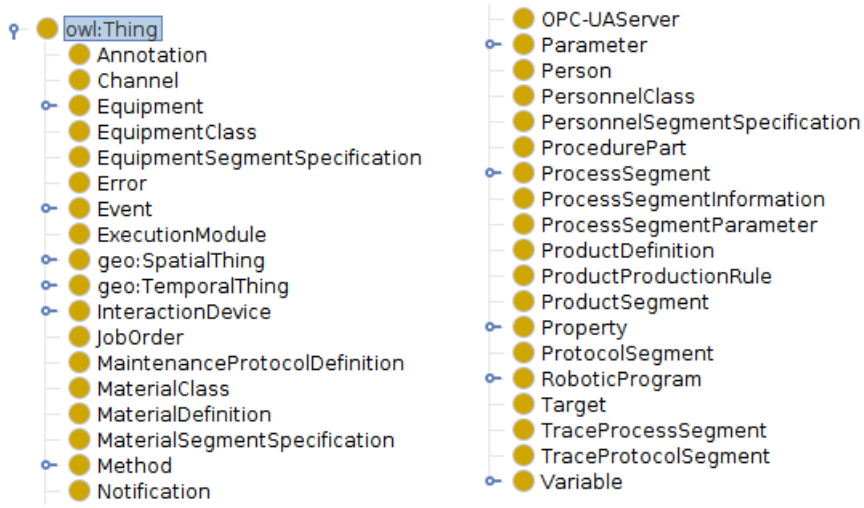


Figure 6.4: Classes in TODODW for KIDE4Assistant.

The rest of the domain information has been modelled and instantiated as in the previously described use cases. For the population strategy, 22 frames, 50 frame heads and 187 lexical units have been semi-automatically instantiated.

As for the dialogue, two classes have been modelled for this use case to cover an additional interaction regarding method selection:

- `MethodOptionsResponse` . The system informs the user that the selected procedure has more than one method.

- `MethodDisambiguationRequest` . The system requests the user to confirm that they want to follow the method in the request.

Tables 6.1 and 6.2, in page 153, show that nearly all the classes for this adaptation have been reused (a 98.1%), and an 89.8% of the instances have been obtained through automatic methods or reused from other use cases. Although all of the dialogue information has been reused, minimal modifications on the dialogue manager's logic have been needed to cover the necessities of the use case.

6.3 Evaluation

To evaluate and validate KIDE4I and its adaptations KIDE4-Guide, KIDE4BinPicking and KIDE4Assistant, three user studies have been defined and carried out.

The following sections will describe the experimental setup of the studies and the results obtained will be reported in qualitative and quantitative terms. For the qualitative analysis, the participants of each study were provided with the SASSI questionnaire (Hone & Graham, 2000) and, for the quantitative analysis, results will be presented from a dialogue perspective (i.e., whether the user intent was fulfilled in the dialogue, the number of steps required to complete a dialogue and the errors found according to their frequency) and from an interaction perspective (i.e., the system's response times).

6.3.1 Experimental Setup

In each user study, 12 subjects –which were considered as potential users of these applications– have been recruited. Each user was expected to perform a number of dialogues[5] (a dialogue was considered a set of interactions in which the user conveys with the system a specific action to perform) according to a set of instructions given to them by the study personnel, where a short description of the scenario was provided, along with the type of interactions that could be addressed to the system. After finishing their dialogues, each user was requested to fill a questionnaire. The questionnaire chosen to evaluate the dialogue system was the SASSI questionnaire (Hone & Graham, 2000), as it provides a comprehensive evaluation on speech-based dialogue systems and it is considered an standard resource to evaluate such systems. However, for some of the use cases, some additional questions were added to evaluate specific areas that are not covered by SASSI, such as security or productivity [6]. For the guide and bin-picking use cases, users were reminded that the object of their evaluation was the dialogue system and not third-party components such as the app itself, the ASR technology or the robot,

[5]The conditions regarding the number of dialogues will be specified in each experimental description.

[6]*vid.* Appendix B.

if present. In the case of the assistant use case, users evaluated the performance of the system as a whole.

The following sections provide specific details for each of the user studies carried out in the context of this work.

6.3.1.1 Guide/Logistics Robot

In this study, users were expected to interact with the guide robot through voice commands in order to be guided to a destination of their choice.

To provide users with an interface to interact with the robot through voice commands, an Android application was developed, which is able to access the dialogue system, deployed in a server machine. The application consists of a button, `SPEAK`, used to interact with the robot, as it can be seen in Figure 6.5. For simplicity, users were provided with a mobile phone with the application installed, so it was not necessary for them to install it on their phones.

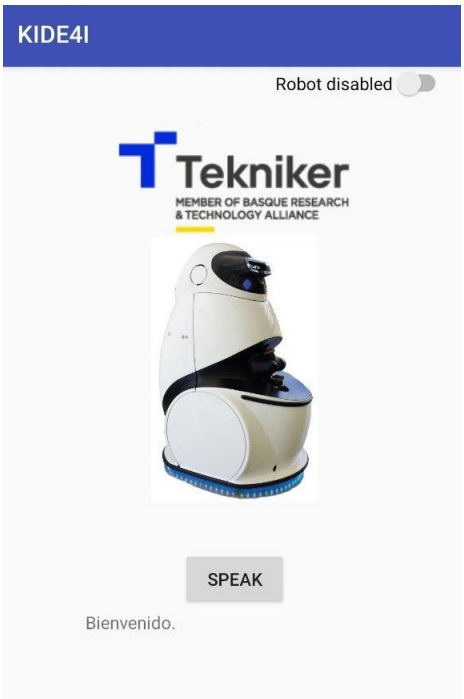


Figure 6.5: Screenshot from the mobile application used to interact with KIDE4I.

Table 6.3: Demographic data for participants in the guide user study. **(a)** Gender information. **(b)** Age information. **(c)** Frequency of voice interaction with everyday devices.

Gender	
M	33%
F	42%
N/D	25%

Age	
24-34	58%
35-44	17%
45-52	17%
N/D	8%

Interaction with everyday devices	
Never	33.3%
Sometimes	58.3%
Frequently	8.3%

In this case, the presence of the robot was emulated (that is, the robot was not physically present). However, each time a dialogue was successfully completed, users received a simulation of their command being sent to the robot to reproduce the use case scenario in the most precise way possible.

The available destinations were defined in a set of maps that contained a selection of destinations that included laboratories, workshops, machines, people, spaces and other objects that corresponded to the Tekniker facility. These maps were provided to each of the users so as to perform their commands, along with some instructions about the experimentation itself[7], such as wording restrictions and the app's basic controls.

The commands directed to KIDE4Guide had no wording restrictions, which meant that destinations could be referred explicitly (e.g., "Take me to the vending machine") or implicitly (e.g., "I want to eat something"). However, some sequences were not supported (such as coordination of destinations; e.g., "I want to go to the meeting room **and** then to the toilet"), and users were instructed about them. Users did not receive any further instructions regarding the commands to direct towards the dialogue system to ensure a natural interaction and not interfere with their interactions. In this use case, users were required to perform at least five dialogues.

Table 6.3 shows demographic data for the participants of the study. All subjects are familiar with technologies in general as they work in domains that require a knowledge of them.

[7] *vid.* Appendix D.

6.3.1.2 Bin-Picking Robot

In this user study, the objective was to interact with the robot though voice commands to indicate which cartridges would go to one of the containers of choice, whereas the rest were to be placed into the other container.

In this case, the robot, depicted in Figures 6.1 and 6.6, was physically present in a manufacturing laboratory. Due to this, users were provided with an industrial headset with microphone, designed to prevent ambient noise to interfere with the voice captured and to protect them from said noise. The interfaces provided to interact with the robot were the same phone and app than in the guide use case.

Figure 6.6: User interacting with the bin-picking robot according to the conditions defined for the experimentation.

Before the experimentation, users were instructed[8] about the objectives of the study and minimal interaction restrictions. As in the previous user study, there were not wording restrictions in general, except for coordination of targets (e.g., "Put the black ones into container 1 **and** the yellow ones into container 2"). Furthermore, they were shown which cartridges were available and how they looked like, so they could see if the robot was correctly performing the action it

[8] *vid.* Appendix D.

Table 6.4: Demographic data for participants in the bin-picking user study. **(a)** Gender information. **(b)** Age information. **(c)** Frequency of voice interaction with everyday devices.

Gender	
M	50%
F	50%

Age	
24-34	42%
35-44	33.3%
45-54	16.6%
55-59	8%

Interaction with everyday devices	
Never	8.3%
Sometimes	83.3%
Frequently	8.3%

was ordered to execute.

During the experimentation, and for each interaction, users were asked to choose three or four cartridges, which were placed on the platform in front of the robot by the personnel in charge of the study. Then, they were asked to perform their commands using the app and the headset they were provided with from the position they were instructed to remain in. As in the previous use case, users were required to perform at least 5 dialogues. Figure 6.6 shows the conditions of the experimentation.

Table 6.4 shows demographic information regarding the study participants. As in the previous user study, all participants were familiar with technologies in general and were also related to some degree to industrial processes.

6.3.1.3 Maintenance Procedure Execution Assistant

In this user study, participants had to interact with a maintenance procedure assistant (an image of which can be seen in Figure 6.7) to receive instructions about the procedures to perform to resolve certain maintenance situations. The design of this assistant integrates two technologies to retrieve maintenance procedures and presenting them to the user: a **Question Answering (QA)** module and a **dialogue** module, supplied by KIDE4Assistant. In this sense, two different types of procedures were distinguished, which were provided by each of the two modules in the assistant's pipeline:

- **General procedures**. These procedures were not stored in a structured way, and were presented to the user as a whole,

Figure 6.7: Screenshot from the maintenance procedure assistant.

including images. The retrieval of the user manual excerpts was performed by the QA module.

- **Guided procedures**. These procedures were stored in the ontology, and were presented to the user according to that modelling, in an assisted way. The navigation through the procedures was performed through KIDE4Assistant, and procedure information was obtained from its ontology instantiation.

To interact with the assistant, users were provided with a tablet with access to it and a headset with microphone. To initiate their interactions, users were required to press a button.

Before the experimentation, users were instructed[9] about the objective of the experimentation, along with some instructions regarding how to use the assistant and a video with a usage example. In this case, there were not wording restrictions at all to interact with the assistant.

[9] *vid.* Appendix D.

Table 6.5: Demographic data for participants in the maintenance procedure execution assistant ssistant user study. **(a)** Gender information. **(b)** Age information. **(c)** Frequency of voice interaction with everyday devices.

Gender	
M	8%
F	92%

Age	
19-23	58.3%
24-34	16.6%
35-44	8.3%
45-54	8.3%
55-59	8.3%

Interaction with everyday devices	
Never	25%
Sometimes	58.3%
Frequently	16.6%

For this study, users had to request maintenance procedure information for two arm robots –from now on, RobotA and RobotB. To run the experimentation smoothly, the participants came in turns. At each turn, two participants were selected: one of them was assigned to RobotA and the other to RobotB, and were taken to their respective locations. Once in there, the experimentation staff provided the user with the situations to resolve for the robot. For the general procedures, four situations were presented, and for the guided ones, two. Each user was required to ask about two general procedures of their choice and to be assisted through one guided procedure. When each of the participants finished their tasks, they were assigned to the other robot, with the same experimental conditions. Thus, in total, each user was required to request procedure information about six different maintenance situations.

Note that in the context of this thesis qualitative evaluation for this use case will be performed on the experimental task as a whole, whereas only the selected guided procedures, which are supplied by KIDE4Assistant, will be the object of quantitative evaluation.

Table 6.5 shows demographic information about the study participants, which were mostly alumni from a dual engineering degree, along with their professors.

6.3.2 Results

To provide a comprehensive evaluation of the system through the user studies described previously, results will be reported at qualita-

tive and quantitative level:

- **Qualitative evaluation**. At this level, responses from the SASSI questionnaire will be analyzed.
- **Quantitative evaluation**. Evaluation at this level will provide quantitative information on the systems evaluated by considering different units of analysis:
  - **Dialogue**. From this perspective, the dialogue as a whole (i.e., a series of interactions between the system and the user to achieve an executable action) is assessed. To do so, three aspects are evaluated:
    - **Dialogue completion rate**. Whether a dialogue has been successful or not (Wu et al., 2019).
    - **Dialogue completion steps**. How many steps were necessary to complete a dialogue.
    - **Errors caused by module**. Number of cases the user goal was not fulfilled by the system due to a specific reason in interpretation.
  - **Interaction**. Here, the interactions performed in each dialogue turn by the system and the user are evaluated considering the following information:
    - **Response time**. Time needed by the system to provide a response given a user request.

6.3.2.1 Qualitative Evaluation: SASSI Questionnaire

The SASSI questionnaire allows to comprehensively evaluate different aspects of KIDE4I adaptations. So as to cover other areas that are relevant for this study regarding the system's industrial application, additional questions have been added to SASSI[10]. Thus, the aspects covered by the questionnaires are the following (Hone & Graham, 2000):

- **Response accuracy** (SASSI). It refers to whether the system is able to interpret an user command and generates an appropriate response.

[10] *vid.* Appendix B.

- **Likeability** (SASSI). It refers to the user perception of the system in terms of usefulness, pleasantness and friendliness.
- **Cognitive demand** (SASSI). It stands for the mental effort required by the user to interact with the system. In industrial scenarios, this aspect is especially relevant, since one of the main objectives of KIDE4I is to simplify the performance of specific tasks.
- **Annoyance** (SASSI). This aspect evaluates how repetitive or annoying it is to interact with the system.
- **Habitability** (SASSI). It refers to whether the user knows what to say to the system.
- **Speed** (SASSI). It evaluates if the system response given a user interaction is fast.
- **Verbosity**. It determines whether the system interactions are too long. In this sense, the system should give the correct amount of information so as users can perform their tasks in the minimum amount of time.
- **Productivity**. This question aims to determine if the fact of using this system would increase the user's productivity, as it is also one of the main objectives of KIDE4I.
- **Security**. As user security is of utmost importance in industrial scenarios, this item evaluates if the design of the system allows users to perform the intended tasks by preserving a secure distance.

Due to the different contexts in which the user studies were performed, different SASSI result reports will be provided. On the one hand, for the guide and bin-picking user studies, the information on the questionnaires account exclusively for the KIDE4I adaptations. On the other hand, for the maintenance procedure assistant user study, the questionnaires apply to the assistant as a whole and, thus, modules other than KIDE4Assistant were also evaluated.

Guide and Bin-Picking User Studies

In these user studies, each question in SASSI consists of a 6-point Likert scale, where 1 stands for *Strongly disagree* and 6 to *Strongly agree*. As it can be observed, the scales for the questionnaires were modified from the original 7-point Likert scale to a 6-point Likert one. This has been done with the objective of removing the neutral point and obtain more representative scores.

Figures 6.8 and 6.9 provide the results obtained from the questionnaire for the SASSI questions and for the additional questions. In general, it can be observed that the results are very positive in both use cases, with median scores between 5 and 6 for common questions between use cases and 4 and 6 for additional questions for the bin-picking use case.

The results in Figure 6.8 show that some aspects had more consensus between participants than others. The aspects that denoted more variability were annoyance, habitability and speed. For annoyance, it can be observed that most answers are among the highest ratings, but also a significant[11] amount of answers have obtained the lowest ratings. This is caused by the results obtained for question number 24 ("*The interaction with the system is repetitive*"), which was rated with an average score of 2.38. Since the tasks to perform in the user studies were strongly related to industrial scenarios, which inherently consist of very specific actions on specific elements, this was an expected outcome. Regarding habitability, results obtained more average scores than the rest of the evaluation items. In this case, although they were given instructions about the task and interactions, participants were not sure about what to say to the system not because it was not clear, but because they were afraid the system would fail if they were too natural in their interactions. Finally, scores for speed were different between use cases. In the guide use case, results for this aspect were more variate, and for the bin-picking use case were more constant. Since the complexity of the guide use case was higher than in the bin-picking use case, in some cases KIDE4I needed more time to process user commands[12]. However, users considered the system to be fast in general terms.

[11] *Significant* in the sense that these observations are not outliers.

[12] A more thorough analysis regarding this aspect and the differences in response times between use cases can be found in Section 6.3.2.3.

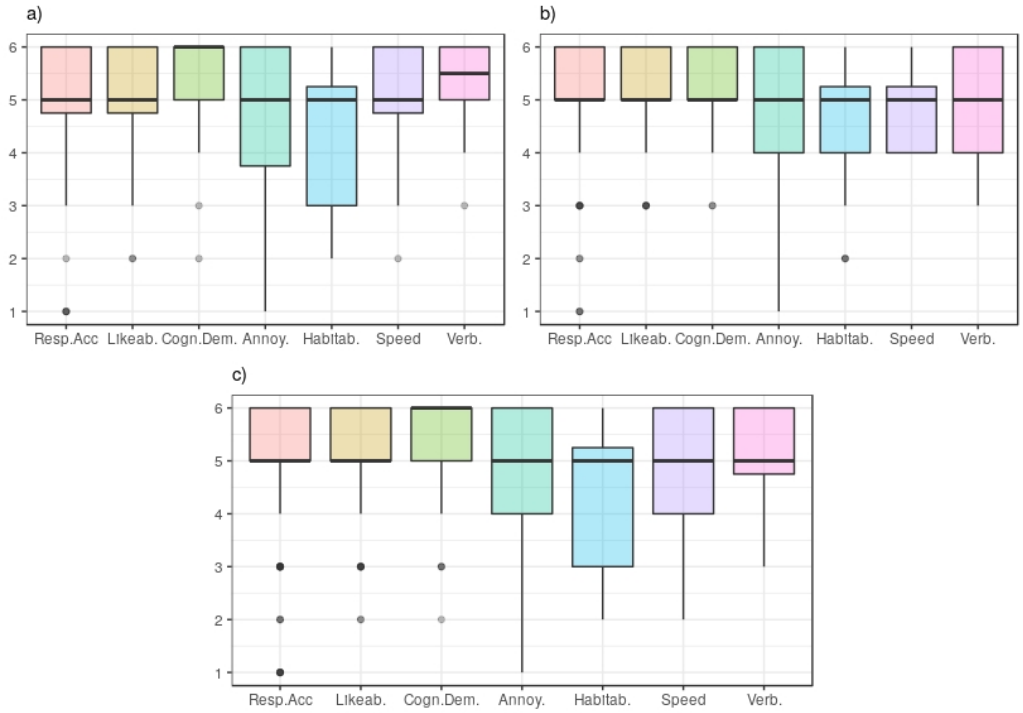


Figure 6.8: Results obtained from user questionnaires. **(a)** Results obtained for the guide use case. **(b)** Results obtained for the bin-picking use case. **(c)** Results obtained considering both experimentations.

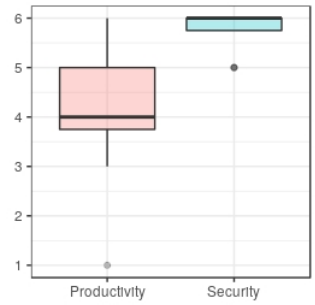


Figure 6.9: Results obtained from user questionnaires: additional questions for the bin-picking use case.

Table 6.6: Scores and standard deviation (SD) values over three different age groups for the two user studies reported in this section.

Aspect	**24-34**	**35-44**	**45-59**	**Total average**	**SD**
Response Accuracy	5.07	4.52	4.68	4.75	0.29
Likeability	5.21	4.94	4.95	5.03	0.15
Cognitive Demand	5.45	5.30	5.43	5.40	0.08
Annoyance	4.59	4.31	4.80	4.57	0.25
Habitability	4.62	4.38	4.35	4.45	0.15
Speed	5.17	4.55	4.63	4.78	0.34
Verbosity	4.75	5.15	5.42	5.11	0.34

On the other hand, the most appreciated aspects among users were response accuracy, likeability, cognitive demand and verbosity, with more consensus between participants. The most relevant results are the ones obtained for cognitive demand, as this aspect refers to one of the main objectives of KIDE4I and validates the easiness of use of the system, which makes it highly suitable in industrial scenarios.

All in all, the system has been considered to be accurate, useful, efficient, not demanding, flexible, fast and that, in general, the amount of information provided is correct.

For additional questions for the bin-picking use case –in Figure 6.9–, they have also been evaluated very positively, and it is specially relevant that the feeling of security by using this system is very high, reaching nearly a perfect score among users. Furthermore, these results show that most users consider that the system would be a plus in their productivity.

Table 6.6 shows the scores for the different evaluated aspects in the two user studies, averaged over the three age groups stated in Tables 6.3 and 6.4 (24-34, 35-44, 45-59). The Table shows that the scores given by participants of different ages is roughly the same, with relatively small standard deviation values. However, and despite the small standard deviation values, these figures show some tendencies according to age. In general, the 24-34 age group assessed more positively the evaluated aspects, except for verbosity and annoyance, in which the results show the opposite, being the 45-59 group the most satisfied. This can be associated to the fact that older people tend to appreciate to be provided a good amount of information, whereas younger people prefer a fast execution rather than information.

Maintenance Procedure Assistant User Study

For the maintenance procedure assistant study, each question in SASSI consists of a 7-point Likert scale, where 1 stands for *Strongly disagree*, 4 for *Neutral* and 7 for *Strongly agree*. In this case, the original SASSI scores have been kept, as it was a decision that depended on sources that are external to this thesis. Here, the SASSI questionnaire evaluates the performance of the assistant as whole, which includes KIDE4Assistant as one of its modules.

Figure 6.10 shows the results obtained from the questionnaire for this user study. In general, the results obtained have been positive, with median scores between 4.5 and 6 in most evaluation items, although a special case, habitability, had a median of 3 points.

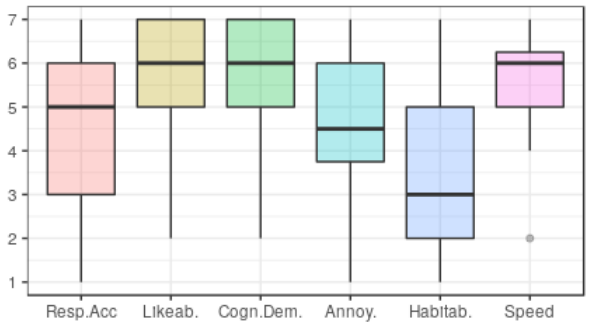


Figure 6.10: Questionnaire results obtained for the procedure assistant use case.

Regarding the aspects with more variability between users, response accuracy, annoyance and habitability (being the two latter common with the other two use cases) showed a wider range of responses than the rest of evaluation items, with standard deviation values of 1.61, 1.61 and 1.54. In fact, these three aspects have in common that a significant range of responses have occurred in all points of the scale.

For response accuracy, users were concerned about whether the system was doing what they expected or wanted, and, thus, the responses to questions 4 ("*The system didn't always do what I wanted*") and 5 ("*The system didn't always do what I expected*") obtained the lowest scores, with 3.92 and 3.83 points, respectively, which were basically in the neutral range. In this sense, in some cases users expected the system to accept a wider range of requests in guided

procedures besides navigation commands, such as questions regarding a step presented by the system (e.g., if the step involves opening a lid, some users expected the system to answer questions of the type of "Where is the lid?"). This originated in a significant volume of requests that were out of the scope of the system's functionality, as it will be seen in Section 6.3.2.2. In some other cases, users expected images to be shown, which were not always returned by the assistant as the original manual did not include them. For annoyance and habitability, the same situations as in the other two user studies could be observed, in which most users found the task repetitive and had doubts whether they were using the right words to interact with the system.

On the other hand, the most appreciated aspects were likeability, cognitive demand and speed.

All things considered, the maintenance procedure assistant, including KIDE4Assistant, has been considered as pleasant to use, not demanding cognitively speaking, fast and competent in the accuracy of its responses.

Finally, in this experimentation, age groups are clearly unbalanced (a 75%[13] of the users fall into the youngest category), which is directly related to the distinction alumni-professors. Due to this, an analysis of the results according to age groups would not be accurate and, thus, it has not been performed for this user study.

6.3.2.2 Quantitative Evaluation at Dialogue Level

This level of analysis aims to evaluate the dialogue system adaptations involved in the user studies by whether the **interaction goal has been fulfilled** (**dialogue completion**), **how many turns** did it take for that interaction goal to be reached (**dialogue steps**) and the number and classification of the errors that caused a dialogue not to be successful or to require some reformulation from the user (**errors caused by module**).

To assess **dialogue completion**, all dialogues have been analysed by a group of experts, who were expected to determine whether the user goal was successfully **completed** or **not completed**. In

[13] *vid.* Table 6.6.

Table 6.7: Dialogue completion results for the user studies reported in this work. Values in parentheses stand for dialogues classified as *partially completed*. % stands for percentages and # for absolute numbers.

	Guide		**BinPicking**		**Assistant**	
	%	#	%	#	%	#
Completed	84.34(8.43)	70(7)	82.67	62	78(7)	230(20)
Not completed	15.66	13	17.33	13	22	64
Total		83		75		294

the case of completed tasks, dialogues were classified between *fully completed* or *partially completed*, depending on whether the user had to reformulate the query at some point of the dialogue (see Example (41)).

(41) a. **Initial - not resolved:** "Tengo sed"
"I am thirsty"

b. **System asks for reformulation:** "No consigo entender tu petición. ¿Puedes reformular tu comando para que te entienda mejor? ¿En qué puedo ayudarte?"
"I cannot understand your command. Can you reformulate so I can understand you better? What can I help you with?"

c. **Reformulation - resolved:** "Quiero beber"
"I want to drink"

Table 6.7 shows the percentage of the dialogue completion rates for each use case. As it can be observed, the results are very positive, where the successful completion rates are within the range between 78% and 84.34%. For KIDE4Guide, an 8.43% of these dialogues were classified as partially completed, and for KIDE4Assistant, a 7%.

Regarding **dialogue steps**, Table 6.8 shows the average number of steps required to complete the dialogue. It is important to keep in mind that each dialogue includes one or two steps that are included by default in each adaptation as part of their design: in the guide use case, the system initiates the dialogue by presenting itself and, in both the guide and bin-picking use cases, when an action is obtained, the system asks the user for confirmation. In the maintenance procedure

Table 6.8: Number of average number of steps to successfully complete a dialogue. Values in parentheses stand for values that exclude steps implemented by default.

Guide	**BinPicking**	**Assistant**
5.4 (3.4)	4.3 (3.3)	1.5

assistant use case, the user initiates the dialogues in most cases and the system does not ask for any confirmation. For this use case, it is also relevant to mention that the complexity of the interactions is lower regarding the rest of use cases, and that explains the difference in average dialogue steps with the other adaptations. Come as it may, the number of dialogues required to achieve the user's goal is positive enough to determine that KIDE4I allows an agile interaction between users and the system.

Finally, as a necessary step to improve the system in future versions, the dialogues that did not fulfil the goal of the user or required reformulations have been analysed to identify the **source of the errors** that led to unsuccessful interpretations. The errors identified are the following:

- **Automatic Speech Recognition** (ASR). Not accurate transcriptions (Example (42)).

  (42) a. **Obtained:** "Quiero *unas alas* con algún ordenador"
  "I want *a pair of wings* with some computer"
  b. **Correct:** "Quiero *una sala* con algún ordenador"
  "I want a room with some computer"

- **Syntactic analysis**. Structures that are not correctly analysed –Example (43)– or words with wrong lemmas (usually for words that are not in the tool's dictionary) –Example (44).

  (43) a. **Obtained:** "El contenedor 2 es $para_{verb}$ la marca Canon"
  "Container 2 is stop the Canon brand"
  b. **Correct:** "El contenedor 2 es $para_{preposition}$ la marca Canon"
  "Container 2 is for the Canon brand"

(44) a. **Obtained:** "Pon los cartuchos $\text{HP}_{lemma:\text{“}h\_p\text{”}}$ en el contenedor 2"
"Put the HP cartridges in container 2"

b. **Correct:** "Pon los cartuchos $\text{HP}_{lemma:\text{“}hp\text{”}}$ en el contenedor 2"
"Put the HP cartridges in container 2"

- **Rules**. Structures that have not been considered in the definitions and/or rules.

- **Polarity interpreter**. Classification errors in the polarity interpreter component (Example (45)).

(45) a. **Obtained:**
SYSTEM: "¿Quieres que te guíe hacia la cafetera?"
"Do you want me to guide you to the coffee machine?"
USER: "$\text{Efectivamente}_{polarity:NO}$"
"Indeed"

b. **Correct:**
SYSTEM: "¿Quieres que te guíe hacia la cafetera?"
"Do you want me to guide you to the coffee machine?"
USER: "$\text{Efectivamente}_{polarity:YES}$"
"Indeed"

- **Ontology-related errors**. Errors in both the ontology modelling or the way information is retrieved from the ontology.

- **Out-of-scope errors**. Errors caused by user interactions that were out of the scope of the functionalities of the adaptation (Example (46), from the maintenance procedure assistant).

(46) a. SYSTEM: "El paso siguiente, el 3º de 6, es 'Conecte el cable de batería a la tarjeta de interfaz de codificador'."
"The next step, the 3th of 6, is 'Connect the battery cable to the encoder interface card'."

b. USER: "¿Dónde está el cable?"
"Where is the cable?"

– **Actions supported by the system:** a procedure-navigation-related interaction, a request for the tool list or a general petition for additional information.

Table 6.9: Sources of the errors observed in the *partially completed* and *not completed* dialogues performed in the user studies.

	Guide		**BinPicking**		**Assistant**	
	%	#	%	#	%	#
ASR	20	4	-	-	18	15
Syntactic analysis	10	2	92	12	-	-
Rules	-	-	8	1	-	-
Polarity interpreter	5	1	-	-	4	3
Ontology-related	65	13	-	-	14	12
Out-of-scope	-	-	-	-	64	54
Total		20		13		84

Table 6.9 shows the number of cases for the identified error sources that lead to not completed dialogues in the user studies. As it can be seen, for KIDE4Guide and KIDE4Assistant the typology of errors is more varied than in KIDE4BinPicking, being the most common errors for KIDE4Guide the ones related with the modelling of the ontology. This is due to the higher complexity of the guide scenario, since this adaptation includes a wide variety of spaces, the elements contained in them, and the fact that it is possible to refer implicitly to spaces (e.g., "I want to *eat*" $\rightarrow$ *vending machine*). For KIDE4BinPicking, errors predominantly stemmed from incorrect syntactic analyses of user commands. More specifically, these had to do with one of the brands involved, HP, the lemma of which was obtained incorrectly due to the fact that it was not included in the tool's dictionary. For KIDE4Assistant, the most common source of errors were out-of-scope interactions, which also occur in a remarkable number of occasions, being a 64% of the total interactions in the user study. This may explain the lower completion rate compared to the other adaptations.

After observing and analysing these errors, the rules and ontology-related ones have been solved. For syntactic analysis, required adaptations have been performed in Freeling's dictionaries. Regarding ASR and the polarity interpreter, although the impact of these errors is not critical, new tools are expected to be explored as part of

future work. As for the out-of-scope ones, these errors are caused by usage errors more than by system-related issues and, thus, are not subject to revision.

6.3.2.3 Quantitative Evaluation at Interaction Level: Response Time

To perform a qualitative evaluation of the different KIDE4I adaptations, the time taken by the system to respond given a user interaction has been analysed. For this, the average system response time has been extracted for each of the adaptations to evaluate and reported in Table 6.10 (Guide for KIDE4Guide, BinPicking for KIDE4BinPicking and Assistant for KIDE4Assistant).

Table 6.10: Average time of response for each use case.

Guide	**BinPicking**	**Assistant**
1.25s	0.75s	1.88s

According to Table 6.10, all adaptations have very fast response times, which is endorsed by the results obtained from the user questionnaires for the speed aspect, reported in Section 6.3.2.1. However, it is worth discussing the difference in response times between use cases. On the one hand, the guide use case is complex in its modelling, which is related to the fact that a single destination can be referred in different ways, explicitly or implicitly (for example, through an object contained in said destination or an action that can be performed in it). Due to this variety of interactions, response times may increase, as more –and more complex– queries may be executed. This can be seen in Examples (47) and (48), where a television is required, but it is referred differently: Example (47) sets a restriction on all the meeting rooms available, whereas (48) just requests for a television. Between the 2 examples there is a difference of 0.3 seconds, which is explained by the difference in complexity between requests.

These examples also show that confirmations/negations also require a very low amount of time to process.

(47) a. **User:** "Quiero una sala con televisión"
"I want a room with a TV"

b. **System:** "Tengo demasiadas opciones que se corresponden con *sala con televisión*. ¿Te parece bien que te guíe hacia la puerta de la sala de reunión Sua 2?"
"I have too many options that correspond with *room with a tv*. Is it okay if I take you to the entrance of meeting room Sua 2?" [**Time elapsed:** 2.17s]

c. **User:** "sí"
"yes"

d. **System:** "¡Ya está! ¿Necesitas algo más?"
"Done! Do you need anything else?" [**Time elapsed:** 0.19s]

(48) a. **User:** "Quiero ver la televisión"
"I want to watch TV"

b. **System:** "Tengo demasiadas opciones que se corresponden con *televisión*. ¿Te parece bien que te guíe hacia la puerta de la sala de reunión Aura 1?"
"I have too many options that correspond with *TV*. Is it okay if I take you to the entrance of meeting room Aura 1?" [**Time elapsed:** 1.87s]

c. **User:** "sí"
"yes"

d. **System:** "¡Ya está! ¿Necesitas algo más?"
"Done! Do you need anything else?" [**Time elapsed:** 0.16s]

As for the bin-picking use case, the complexity of the scenario is simpler and, thus, the time required to process user commands is lower (in fact, the lowest of all three use cases). This can be observed in Examples (49) and (50). In Example (49), the system only takes 0.87 seconds to process the initial command and to detect that there is one argument (the destination container) missing and, in Example (50), if the user command contains all the necessary information, it takes 1.11 seconds to process the whole command. As in the guide use case, affirmations/negations take around 0.2 seconds to process.

(49) a. **User:** "Coge el cartucho Brother"
"Pick the Brother cartridge"

b. **System:** "Hay información que me falta. Necesito que me digas el contenedor donde quieres depositar el cartucho, por favor."
"I am missing some information. Tell me the container you want to place the cartridge, please." [**Time elapsed:** 0.87s]

c. **User:** "En el contenedor 2"
"In container 2"

d. **System:** "Entonces, quieres que coja y ponga los cartuchos del tipo Brother en 2, ¿verdad?"
"Then, you want me to pick and place the Brother cartridges in 2, isn't it?" [**Time elapsed:** 0.21s]

(50) a. **User:** "Lleva el magenta al 2"
"Take the magenta one to 2"

b. **System:** "Entonces, quieres que coja y ponga los cartuchos del tipo magenta en 2, ¿verdad?"
"Then, you want me to pick and place the magenta cartridges in 2, isn't it?" [**Time elapsed:** 1.11s]

c. **User:** "sí"
"yes"

d. **System:** [Command sent to robot] [**Time elapsed:** 0.21s]

These examples also show that the initial command is the command that takes longer to interpret, as it requires more processing tasks to perform because it usually contains more information to process (i.e., it is common that the action and most argument values are included in the initial command).

Regarding KIDE4Assistant, this adaptation has the highest response times, although it is still low and suitable for a smooth interaction between users and the dialogue system. It has been observed that, in a significant amount of cases, longest response times were obtained due to the system not understanding the user request, as more processing steps were taken to try to interpret the command. Considering the amount of out-of-scope requests –noted in Section 6.3.2.2– which are interactions that the system could not interpret, it is safe to assume that the average response time has been affected by these interactions. Example (51) shows the time taken for an

understood command to get a response, and Example (52) for an out-of-scope command:

(51) a. **User:** "Terminado"
"Finished"

b. **System:** "Tengo varias opciones para tu comando. ¿Quieres que te muestre el siguiente?"
"I have many options for your command. Do you want me to show you the next one?" [**Time elapsed:** 0.93s]

c. **User:** "sí"
"yes"

d. **System:** [Retrieve procedure information] [**Time elapsed:** 1s]

(52) a. **System:** "El paso siguiente, el 3º de 6, es 'Conecte el cable de batería a la tarjeta de interfaz de codificador."
"The next step, the 3th of 6, is 'Connect the battery cable to the encoder interface card'."

b. **User:** "¿Dónde está el cable?"
"Where is the cable?"

c. **System:** "No consigo entender tu petición. ¿Puedes reformular tu comando para que te entienda mejor? ¿En qué puedo ayudarte?"
"I cannot understand your command. Can you reformulate so I can understand you better? What can I help you with?" [**Time elapsed:** 2.15s]

- **Action expected by the system:** an interaction related to procedure navigation, a request for the tool list or a general petition for additional information.

In some other occasions, longer response times were also seen when the system had to retrieve the procedure information requested by the user, as it can be seen in Example (53). However, this did not occur in all cases, as Example (54) shows:

(53) a. **User:** "Qué herramientas necesito"
"Which tools do I need?"

b. **System:** [Retrieve tool list] [**Time elapsed:** 2.12s]

(54) a. **User:** "Vale"
"Okay."
b. **System:** [Retrieve next step information] [**Time elapsed:** 1.01s]

In any case, all KIDE4I adaptations have shown good response times, which reinforces its usability in different use cases.

6.4 Conclusions

This Chapter wraps up all the work performed in this thesis, which is put into practice by reporting the adaptation process of KIDE4I to three use cases that are relevant in industrial scenarios: bin-picking, CMMSs and assistance for maintenance procedure execution.

To perform each adaptation, a methodology has been defined and used, which is based on the different resources, techniques and modules reported throughout this thesis and centred on making the adaptation process as simple as possible, basically based on resource reuse.

The characteristics of each of the adaptations –KIDE4BinPicking, KIDE4CMMS and KIDE4Assistant– have also been reported, along with their adaptation process. Considering the following, the process to adapt KIDE4I to these use cases is fairly simple:

- For KEE: reuse of most rules from the KIDE4Guide implementation.
- Reuse of the polarity interpreter from the KIDE4Guide implementation.
- For ontology modelling:
  - Use of TODO.
  - Reuse of vocabularies for TODODW modelling.
- For ontology instantiation:
  - Reuse of instances for TODODM from KIDE4Guide.

  - Use of the population strategy in Chapter 4.6.1 for TODO-DFA and TODODom instantiation.
  - Automatic instantiation of domain knowledge from databases.

- Reuse of the dialogue manager functions from the KIDE4Guide use case.

In this sense, it can be seen that, once an initial implementation is obtained, the effort to adapt KIDE4I to other use cases is considerably reduced. This validates the work performed in this thesis for that matter, while consolidating the objective of having a generic approximation to natural, task-oriented dialogue for industrial scenarios.

Moreover, this Chapter also provides a framework to evaluate task-oriented dialogue systems in industrial scenarios. This framework considers evaluation from a qualitative and a quantitative point of view. The former is performed by using the SASSI (Hone & Graham, 2000) questionnaire, whereas the latter evaluates the rate of completed dialogues, the number of necessary steps to complete a dialogue, the number of cases in which an error of a specific type has caused the dialogue not being successful (or partially successful) and the system's response time.

This Chapter also reports the three user studies that have been carried out to evaluate KIDE4Guide and the KIDE4BinPicking and KIDE4Assistant adaptations through the evaluation framework reported in this thesis. Considering the results obtained, it can be concluded that KIDE4I is user friendly and allows natural, accurate, fast and secure interactions between workers and industrial systems, triggering human acceptation and enhancing their working conditions.

Chapter 7

Conclusions and Further Work

Task-oriented dialogue systems are powerful technologies that allow industrial workers to perform multiple tasks at once by delegating simpler tasks through natural voice commands, with a minimal impact in their cognitive demand and guaranteeing optimal work conditions. However, the design process of such tools in industrial scenarios is hindered by a lack of labelled data to develop the system components using state-of-the-art techniques, such as Deep Learning (DL). As a consequence, industrial task-oriented dialogue systems usually restrict their natural language capabilities and are highly specific for the target application, limiting the possibility of reusing resources. Furthermore, these systems are often static, and manual, complex work is required to add new functionalities.

To overcome these challenges, the research in this thesis has been oriented towards developing a generic task-oriented dialogue system that allows a natural communication between human workers and industrial systems. The use of semantic technologies as the system's core and its modular, generic design have enabled to develop a quality system that is easily adaptable to different use cases with minimum adaptations and without requiring large amounts of data. In this line, it has been proved that the resources generated for the implementation of the different modules are partially or totally reusable among use cases, which significantly reduces the time, cost and resources necessary to adapt the system to new use cases. Moreover,

this thesis has contributed with different resources, techniques and methodologies based on the reuse of existing data, which have been validated and evaluated in experimental industrial settings.

Furthermore, the use of Semantic Web Technologies in combination with NLP techniques to develop task-oriented dialogue systems has shown the potential of ontologies to be the core of these applications, other than just modelling the domain, opening new and promising insights in this research area.

7.1 Contributions

The following sections summarise the main contributions of this thesis. Considering the structure of the research in this document, these contributions will be presented around three basic pillars: the Task-Oriented Dialogue management Ontology (TODO), the Knowledge-drIven Dialogue framEwork for Industry (KIDE4I) and the implementation and adaptation of KIDE4I in industrial use cases.

7.1.1 The Task-Oriented Dialogue management Ontology (TODO)

TODO has been developed to be the core ontology of semantic-based task-oriented dialogue systems for industrial scenarios, covering both the domain- and dialogue-related knowledge to enable a natural communication between human workers and industrial automatisms, as described in Chapter 4. The use of TODO in this setting allows to reduce the effort to obtain, implement and adapt a task-oriented dialogue system with limited amounts of data.

This ontology has been designed by following a well-known methodology, LOT (Poveda-Villalón et al., 2019), in its industrial version, which makes it highly suitable for the context in this thesis. By following this methodology, TODO has been implemented through a set of carefully-specified ontology requirements, obtained from expert input and codified as Competency Questions (CQs).

To enable the grouping of specific knowledge in independent com-

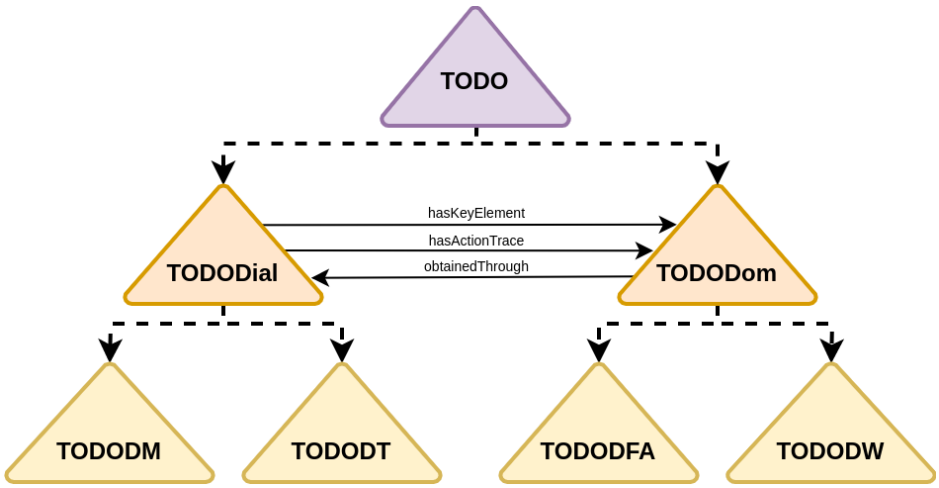


Figure 7.1: General overview of TODO.

ponents to ease the modelling process, this ontology has been designed to be modular, as Figure 7.1 shows. Considering this and the requirements set, TODO consists of the following modules and submodules:

- **Dialogue (TODODial)**. Information related to the dialogue process.
  - **Dialogue Management (TODODM)**. Modelling of the dialogue process.
  - **Dialogue Tracing (TODODT)**. Necessary knowledge to store dialogue traces to, for example, learn from previous interactions.
- **Domain (TODODom)**. Information related to the domain.
  - **Domain Frame-Action (TODODFA)**. Modelling of the actions that are executable by the target system.
  - **Domain World (TODODW)**. Modelling of the elements that are present in the domain scenario.

Each of these modules and submodules reuse terms from other ontologies when possible.

TODO has been published online for its reuse. For that matter, comprehensive documentation has been generated for all modules

and submodules in TODO. Furthermore, these have been objectively evaluated considering different aspects and it has been proved that TODO is a high-quality ontology.

7.1.2 The Knowledge-drIven Dialogue framEwork for Industry (KIDE4I)

KIDE4I, presented in Chapter 5, is the generic semantics-based task-oriented dialogue system for industrial scenarios to enable a natural human-machine interaction. This dialogue system uses the TODO ontology as its core, which has been key to enable the system to be implemented without requiring big amounts of data to do so and to be easily adapted to different use cases. This is also achieved by encouraging the reuse of existing information in the adaptation process of its different modules.

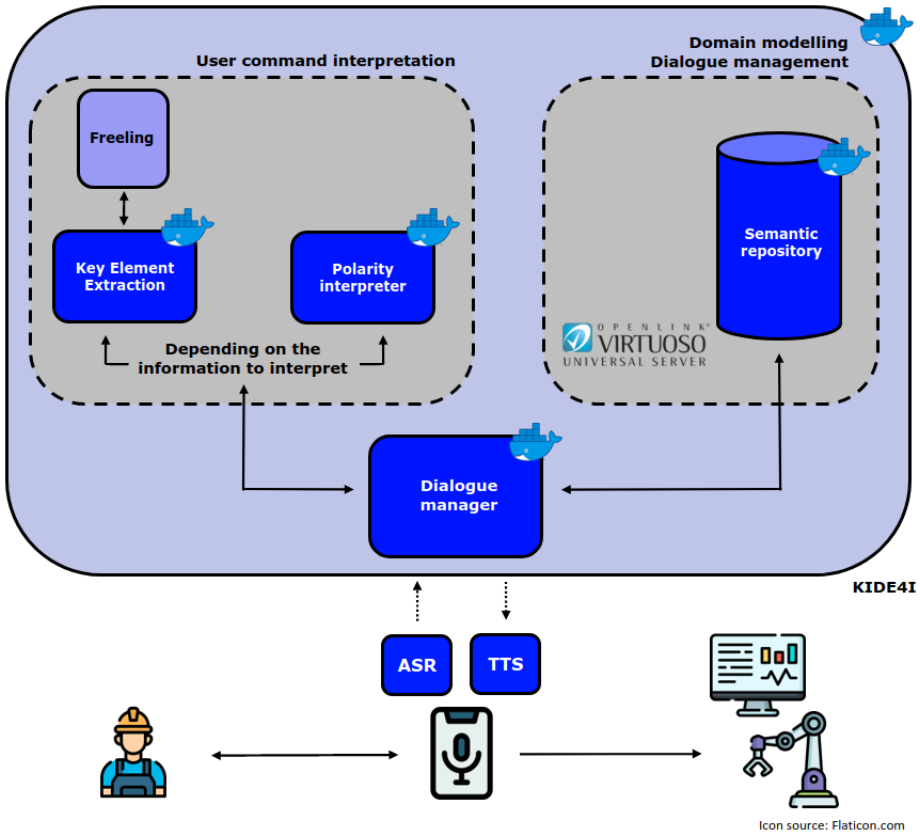


Figure 7.2: KIDE4I architecture.

KIDE4I's architecture consists of four modules, and it is inspired in the typical architecture of task-oriented dialogue systems[1]: the key element extraction component, the polarity interpreter, the semantic repository and the dialogue manager, as illustrated in Figure 7.2. Each of these components has been developed from a generic perspective to be reused in different adaptations with minimum adjustments. However, the components that are most related with the domain (e.g., the *ABOX* from the semantic repository or the key element extraction component) usually require domain-specific adjustments. To ease this process and reduce the time and costs to do so, different actions are depicted (and encouraged) in this thesis. These basically consist on the reuse of resources from an initial implementation or other adaptations of the system (such as ontology instances or key element extraction rules) and the automatic instantiation of world elements from existing databases or other resources to populate the ontology. In this context, this thesis also contributes with two strategies:

- A strategy to leverage multilingual linguistic resources to semi-automatically instantiate intent-relevant information in the domain ontology. This contribution, presented in Section 4.6, is relevant to reduce the time and effort needed to populate the ontology when new KIDE4I adaptations are needed.

- A strategy to reuse data from different –but similar– domains to augment domain data for the training of ML-based models to perform key element extraction. This contribution, described in Section 5.1, is relevant to overcome the lack of labelled data in industrial scenarios.

7.1.3 Implementation and Adaptation of KIDE4I to Industrial Use Cases

To assess the genericity and easiness of adaptation of KIDE4I to different industrial applications, four use cases have been considered: a guidance scenario, a bin-picking robot scenario, a maintenance management software scenario and a maintenance procedure

[1] *vid.* Chapter 2.1.1.

execution assistant scenario. Their adaptation process is described in Chapter 6.

So as to set a starting point for the rest of adaptations, Chapter 5 describes KIDE4I's implementation process for the guidance scenario, obtaining KIDE4Guide. For this implementation, the set of rules for key element extraction have been created from scratch, a generic instantiation of TODODial has been obtained and a first implementation of the dialogue manager has been developed. All of these resources have been generated to be reused in other use case adaptations to reduce the adaptation effort.

To easily adapt KIDE4I to other use cases, a methodology to do so has been designed and reported in Chapter 6.1. By following it, three KIDE4I adaptations have been obtained for the remaining three use cases above: KIDE4BinPicking, KIDE4CMMS and KIDE4Assistant, respectively. The adaptation process of KIDE4I for each use case has been reported and deemed as a relatively simple process, mainly supported by the reuse of the resources generated for KIDE4Guide and the use of the strategy to semi-automatically instantiate the intent-relevant information in the domain ontology, as well as the use of information stored in databases or similar repositories.

In this context, this thesis also provides an evaluation framework for task-oriented dialogue systems in industrial scenarios, which has been performed over KIDE4Guide, KIDE4BinPicking and KIDE4Assistant through their respective user studies. Within this framework, each adaptation has been evaluated from a qualitative and quantitative point of view.

7.2 Main Conclusions

Considering the contributions of this thesis and the work carried out for each of them, this Section aims to gather the main conclusions reached from the results obtained.

The process to design and implement an ontology is a complex task, but the use of a methodology to do so provides the actions to be performed to ensure that, in the end, a quality ontology is obtained. The main advantage of developing an ontology for its use in

task-oriented dialogue systems is that this effort has to be performed once, and then the result can be reused. This outcome is opposed to typical industrial task-oriented dialogue systems, in which there is a lot of work involved in terms of conceptualisation and implementation and cannot be reutilised as they are highly specific to the target application, so new applications will require the same process. In this sense, the use of TODO as core of task-oriented dialogue systems simplifies this conceptualisation step, as it has been designed to be common among use cases thanks to the input of experts in the field in its design phase.

KIDE4I, supported on TODO, allows to develop a generic task-oriented dialogue system to cope with the limitations of industrial scenarios. Nevertheless, the use of ontologies in this context requires the knowledge that is not common between use cases –that is, domain knowledge– to be modelled, and ontologies to be instantiated. Ontology instantiation may be a manual, tedious process, but this thesis provides several mechanisms to be able to reduce the effort in this matter by providing a strategy that shows that the reuse of existing linguistic resources is very useful in this task. The evaluation of this strategy has shown that the use of such data is a very promising approach, although information needs to be carefully selected in the manual selection phase to ensure good results. In this sense, semantic information is very valuable in this selection phase. As for dialogue knowledge, it has been proven that it can be reused among use cases, only requiring adaptations to model the conditions of the use case at hand if necessary.

Regarding the different modules in KIDE4I, several conclusions can be extracted. As for the KEE component, the use of rules as starting point allows to implement the component when no data to do so is available, and the adaptation of these rules to different use cases benefits from the set of rules from the initial KIDE4I implementation. For the ML-based component, the use of out-of-domain data from different but similar domains to cope with scarce or non-existing domain training data is a very promising approach. However, models trained with only out-of-domain data are not able to improve the results obtained through rules. In this sense, the optimal methodology to benefit from this out-of-domain data is (i) to combine it with domain data and that both sets are hand-corrected and (ii) that the model is trained with a CRF algorithm. In addition, it is also very important to keep in mind that not all out-of-domain datasets are

suitable for this task and, thus, they should be carefully selected. For the polarity interpreter, the lack of resources for this specific task has been tackled by the use of a solution for sentiment analysis, a conceptually similar application, an approach that has proven to be useful in this context. As for the semantic repository, it is a very important component in KIDE4I, as it stores the knowledge in the ontology and its instantiation. In this sense, it is crucial that it has reasoning capabilities to cover the knowledge in TODO. Finally, the dialogue manager benefits of the semantic conception of KIDE4I, as many of its functions rely on SPARQL queries that are generic and are easy to adapt if necessary, rather than re-programming the whole function.

The generic conceptualisation of KIDE4I's modules, in the same way as the use of ontologies, has the advantage that their design and implementation is reusable. Considering this, the methodologies provided in this thesis for adaptation are also generic. The reuse of existing resources in KIDE4I's modules is also beneficial and, in combination with the rest of considerations, allows to reduce the effort to obtain new adaptations, as it has been observed in the ones reported in this thesis. In these, the adaptation process to different use cases has proven to be a simple task, mainly motivated by the reuse of resources to adapt each module, that mostly come from KIDE4I's initial implementation.

Finally, the results obtained in each of the user studies to evaluate KIDE4I's adaptations have concluded that the use of KIDE4I in different industrial scenarios contributes to a reliable communication between human workers and industrial systems, enabling accurate, fast and safe interactions, contributing to human well-being in industrial settings.

7.3 Future Work

This thesis has contributed with new insights in the development of task-oriented dialogue systems for industrial settings, especially in the use of semantic technologies as their core and a generic perspective. Nevertheless, there are actions that have not been dealt with in this thesis, which are devised as future work.

First of all, and to comply with the methodology used for TODO's design, ontology maintenance tasks are expected. These also include solving the pitfalls found in TODO's evaluation task which, although not critical, should be checked.

The implementation of a ML-based key element extraction component for each of the KIDE4I adaptations in this thesis is also envisioned. This is motivated by the fact that the user studies carried out have allowed to obtain domain data to be used for training the ML models. In this sense, effort should also be directed towards finding datasets from similar domains to augment the existing domain data.

Another area of future work deals with improving KIDE4I's polarity interpreter. The current implementation, although fully functional in the context of KIDE4I, may be enhanced by exploring other tools and resources to classify the polarity of commands and obtain a more robust component.

Minor actions identified include an experimentation task to evaluate KIDE4CMMS in the same way as the rest of KIDE4I's adaptations. Moreover, the development of a GUI that implements the intent-related information instantiation strategy is also devised.

In the long run, this thesis has also opened different paths for further investigation. As described throughout this work, KIDE4I's dialogue manager has been designed to dynamically generate dialogues to gather user feedback with the objective of improving the dialogue system from new interactions. However, this functionality has not been yet implemented, and this line of future work is oriented towards its implementation in the existing –and future– KIDE4I adaptations. To do so, research on the methods to generate these dialogues and the techniques to gather and exploit the feedback obtained from users is necessary, while putting special focus on KIDE4I's robustness and reliability.

Research on lines other than the ones identified in this section is also encouraged to contribute to the development of robust task-oriented dialogue systems that support a natural communication in industrial settings, which can even be extended to other scenarios of interest.

References

Agerri, R., & Rigau, G. (2016). Robust Multilingual Named Entity Recognition with Shallow Semi-Supervised Features. *Artificial Intelligence*, *238*, 63–82.

Agerri, R., & Rigau, G. (2018). Language Independent Sequence Labelling for Opinion Target Extraction. *Artificial Intelligence*(268), 85–95.

Allemang, D., & Hendler, J. (2011). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Elsevier.

Altinok, D. (2018). An Ontology-Based Dialogue Management System for Banking and Finance Dialogue Systems. *arXiv preprint arXiv:1804.04838*.

Alvez, J., Gonzalez-Dios, I., & Rigau, G. (2019). Commonsense Reasoning Using WordNet and SUMO: a Detailed Analysis. *arXiv preprint arXiv:1909.02314*.

Antonelli, D., & Bruno, G. (2017). Human-Robot Collaboration Using Industrial Robots. In *2017 2nd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2017)* (pp. 99–102).

Artale, A., Magnini, B., & Strapparava, C. (1997). WordNet for Italian and its Use for Lexical Discrimination. In *Congress of the Italian Association for Artificial Intelligence* (pp. 346–356).

Atserias, J., Villarejo, L., Rigau, G., Agirre, E., Carroll, J., Magnini, B., & Vossen, P. (2004). The MEANING Multilingual Central Repository. In *Proceedings of the 2nd International Global WordNet Conference, Jan 20-23, 2004* (pp. 23–30).

Baader, F., Horrocks, I., Lutz, C., & Sattler, U. (2017). *An Introduction to Description Logic*. Cambridge University Press.

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet Project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1* (pp. 86–90).

Bastianelli, E., Castellucci, G., Croce, D., Iocchi, L., Basili, R., & Nardi, D. (2014, May). HuRIC: a Human Robot Interaction Corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)* (pp. 4519–4526). Reykjavik, Iceland: European Languages Resources Association (ELRA). Retrieved from `http://www.lrec-conf.org/proceedings/lrec2014/pdf/531_Paper.pdf`

Beesley, K., & Karttunen, L. (2003, 01). Finite-State Morphology. *Bibliovault OAI Repository, the University of Chicago Press*.

Benabbas, A., Hornig, H., & Nicklas, D. (2018). Semi-Automatic Ontology Population for Online Quality Assessment of Particulate Matter Sensors. In *Intelligent Environments 2018* (pp. 119–128). IOS Press.

Bentivogli, L., Forner, P., Magnini, B., & Pianta, E. (2004). Revising the WordNet Domains Hierarchy: Semantics, Coverage and Balancing. In *Proceedings of the Workshop on Multilingual Linguistic Resources* (pp. 94–101).

Berners-Lee, T. (2006). *Linked Data.* Retrieved from `https://www.w3.org/DesignIssues/LinkedData.html` (Accessed: 09 March 2022)

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, *284*(5), 34–43.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., & Winograd, T. (1977). GUS, a Frame-Driven Dialog System. *Artificial Intelligence*, *8*(2), 155–173.

Bohus, D., & Rudnicky, A. (2003). Ravenclaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda.

Brabra, H., Báez, M., Benatallah, B., Gaaloul, W., Bouguelia, S., & Zamanirad, S. (2021). Dialogue Management in Conversational Systems: A Review of Approaches, Challenges, and Opportunities. *IEEE Transactions on Cognitive and Developmental Systems*.

Brickley, D. (n.d.). *Basic Geo (WGS84 lat/long) Vocabulary.* Retrieved from `http://www.w3.org/2003/01/geo/wgs84_pos#`

Brickley, D., & Miller, L. (n.d.). *FOAF Vocabulary.* Retrieved from `http://xmlns.com/foaf/spec/`

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., & Gać, M. (2018). MultiWOZ–A Large-

Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. *arXiv preprint arXiv:1810.00278*.

Bugmann, G., & Pires, J. N. (2005). Robot-by-Voice: Experiments on Commanding an Industrial Robot Using the Human Voice. *Industrial Robot: An International Journal*.

Burchardt, A., Erk, K., Frank, A., Kowalski, A., Padó, S., & Pinkal, M. (2006). The SALSA Corpus: a German Corpus Resource for Lexical Semantics. In *LREC* (pp. 969–974).

Candito, M., Amsili, P., Barque, L., Benamara, F., De Chalendar, G., Djemaa, M., ... others (2014). Developing a French FrameNet: Methodology and First Results. In *LREC-The 9th edition of the Language Resources and Evaluation Conference.*

Carreras, X., Chao, I., Padró, L., & Padró, M. (2004). FreeLing: An Open-Source Suite of Language Analyzers. In *LREC* (pp. 239–242).

Cassier, M., Sellami, Z., & Lorré, J.-P. (2019). Meeting Intents Detection Based on Ontology for Automatic Email Answering. In *30es Journées Francophones d'Ingénierie des Connaissances, IC 2019* (pp. 99–111).

Chen, H., Liu, X., Yin, D., & Tang, J. (2017, November). A Survey on Dialogue Systems: Recent Advances and New Frontiers. *SIGKDD Explor. Newsl.*, *19*(2), 25–35. Retrieved from `https://doi.org/10.1145/3166054.3166058`

Chen, Q., Zhuo, Z., & Wang, W. (2019). BERT for Joint Intent Classification and Slot Filling. *CoRR*. Retrieved from `http://arxiv.org/abs/1902.10909`

Chen, Z., & Liu, B. (2018). Lifelong Machine Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *12*(3), 1–207.

Choy, M. C., Srinivasan, D., & Cheu, R. L. (2006). Neural Networks for Continuous Online Learning and Control. *IEEE Transactions on Neural Networks*, *17*(6), 1511–1531.

Commission, E., for Research, D.-G., Innovation, Breque, M., De Nul, L., & Petridis, A. (2021). *Industry 5.0: Towards a Sustainable, Human-Centric and Resilient European Industry.* Publications Office.

Corcoglioniti, F., Rospocher, M., & Aprosio, A. P. (2016). Frame-Based Ontology Population with PIKES. *IEEE Transactions on Knowledge and Data Engineering*, *28*, 3261–3275.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine learning*, *20*(3), 273–297.

Cyganiak, R., Wood, D., & Lanthaler, M. (2014). *RDF 1.1 Concepts and Abstract Syntax.* Retrieved from `https://www.w3.org/TR/rdf11-concepts/` (Accessed: 12 March 2022)

De Lacalle, M. L., Laparra, E., Aldabe, I., & Rigau, G. (2016). Predicate Matrix: Automatically Extending the Semantic Interoperability between Predicate Resources. *Language Resources and Evaluation*, *50*(2), 263–289.

de Lacalle, M. L., Laparra, E., & Rigau, G. (2014). First Steps Towards a Predicate Matrix. In *Proceedings of the Seventh Global Wordnet Conference* (pp. 363–371).

De Lacalle, M. L., Laparra, E., & Rigau, G. (2014). Predicate Matrix: Extending SemLink through WordNet Mappings. In *LREC* (pp. 903–909).

Esnaola-González, I., Bermúdez, J., Fernández, I., & Arnaiz, A. (2021). EEPSA as a Core Ontology for Energy Efficiency and Thermal Comfort in Buildings. *Semantic Web*, *Pre-press*.

Estarrona, A., Aldezabal, I., & de Ilarraza, A. D. (2020). How the Corpus-Based Basque Verb Index Lexicon was Built. *Language Resources and Evaluation*, *54*(1), 73–95.

Farias Lóscio, B., Burle, C., & Calegari, N. (Eds.). (2017, January). *Data on the Web Best Practices.* Retrieved from `https://www.w3.org/TR/dwbp/` (Accessed: 9 Jan 2022)

Fasola, J., & Mataric, M. J. (2013). Using Semantic Fields to Model Dynamic Spatial Relations in a Robot Architecture for Natural Language Instruction of Service Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013* (pp. 143–150).

Fast, E., Chen, B., Mendelsohn, J., Bassen, J., & Bernstein, M. S. (2018). IRIS: A Conversational Agent for Complex Tasks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (pp. 1–12).

Fernández-Montraveta, A., Vázquez, G., & Fellbaum, C. (2008). The Spanish Version of WordNet 3.0. *Text Resources and Lexical Knowledge*, *8*, 175–182.

Fernández, I., Casla, P., Esnaola, I., Parigot, L., & Marguglio, A. (n.d.). Towards Adaptive, Interactive, Assistive and Collaborative Assembly Workplaces through Semantic Technologies.

Fillmore, C. J., et al. (1976). Frame Semantics and the Nature of Language. In *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech* (Vol. 280, pp. 20–32).

Firdaus, M., Kumar, A., Ekbal, A., & Bhattacharyya, P. (2019). A Multi-Task Hierarchical Approach for Intent Detection and Slot Filling. *Knowledge-Based Systems*, *183*, 104846.

Firth, J. R. (1957). A Synopsis of Linguistic Theory, 1930-1955. *Studies in Linguistic Analysis*.

Forbes, M., Rao, R. P., Zettlemoyer, L., & Cakmak, M. (2015). Robot Programming by Demonstration with Situated Spatial Language Understanding. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2014–2020).

Fromreide, H., & Søgaard, A. (2014). NER in Tweets Using Bagging and a Small Crowdsourced Dataset. In A. Przepiórkowski & M. Ogrodniczuk (Eds.), *Advances in Natural Language Processing* (pp. 45–51). Springer International Publishing.

Gao, J., Galley, M., Li, L., et al. (2019). Neural Approaches to Conversational AI. *Foundations and Trends®in Information Retrieval*, *13*(2-3), 127–298.

Gao, J., Xiong, C., Bennett, P., & Craswell, N. (2022, 01). *Neural Approaches to Conversational Information Retrieval.*

Garijo, D. (2017). WIDOCO: a Wizard for Documenting Ontologies. In *International Semantic Web Conference* (pp. 94–102).

Garijo, D., Corcho, O., & Poveda-Villalón, M. (2021). FOOPS!: An Ontology Pitfall Scanner for the FAIR Principles. , *2980*. Retrieved from `http://ceur-ws.org/Vol-2980/paper321.pdf`

Gatt, A., & Krahmer, E. (2018). Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation. *Journal of Artificial Intelligence Research*, *61*, 65–170.

Gepperth, A., & Karaoguz, C. (2016). A Bio-Inspired Incremental Learning Architecture for Applied Perceptual Problems. *Cognitive Computation*, *8*(5), 924–934.

Goddeau, D., Meng, H., Polifroni, J., Seneff, S., & Busayapongchai, S. (1996). A Form-Based Dialogue Manager for Spoken Language Applications. *Proceedings of Fourth International Conference on Spoken Language Processing. ICSLP '96*, *2*, 701–704 vol.2.

Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. *Synthesis Lectures on Human Language Technologies*, *10*(1), 1–309.

Gonzalez-Agirre, A., Laparra, E., & Rigau, G. (2012). Multilingual Central Repository version 3.0. In *LREC* (Vol. 2525, p. 2529).

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., & Bengio,

Y. (2013). An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv preprint arXiv:1312.6211*.

Goyal, A., Gupta, V., & Kumar, M. (2018). Recent Named Entity Recognition and Classification Techniques: A Systematic Review. *Computer Science Review*, *29*, 21–43. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1574013717302782`

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Guinovart, X. G., Gonzalez-Dios, I., Oliver, A., & Rigau, G. (2021). Multilingual Central Repository: a Cross-lingual Framework for Developing Wordnets. *arXiv preprint arXiv:2107.00333*.

Gupta, A., Hewitt, J., & Kirchhoff, K. (2019). *Simple, Fast, Accurate Intent Classification and Slot Labeling for Goal-Oriented Dialogue Systems.*

Gustavsson, P., Syberfeldt, A., Brewster, R., & Wang, L. (2017). Human-Robot Collaboration Demonstrator Combining Speech Recognition and Haptic Control. *Procedia CIRP*, *63*, 396–401.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: an Update. *SIGKDD Explorations*, *11*(1), 10–18.

Harris, S., & Seaborne, A. (2013). *SPARQL 1.1 Query Language.* Retrieved from `https://www.w3.org/TR/2013/REC-sparql11-query-20130321/` (Accessed: 12 March 2022)

Hayes, P. J., & Patel-Schneider, P. F. (2014). *RDF 1.1 Semantics.* Retrieved from `https://www.w3.org/TR/rdf11-mt/` (Accessed: 12 March 2022)

He, Z., Wang, Z., Wei, W., Feng, S., Mao, X., & Jiang, S. (2020). A Survey on Recent Advances in Sequence Labeling from Deep Learning Models. *arXiv preprint arXiv:2011.06727*.

Henderson, M., Thomson, B., & Young, S. (2013). Deep Neural Network Approach for the Dialog State Tracking Challenge. In *Proceedings of the SIGDIAL 2013 Conference* (pp. 467–471).

Heppin, K. F., & Gronostaj, M. T. (2012). The Rocky Road Towards a Swedish FrameNet-Creating SweFN. In *LREC* (pp. 256–261).

Hone, K. S., & Graham, R. (2000). Towards a Tool for the Subjective Assessment of Speech System Interfaces (SASSI). *Natural Language Engineering*, *6*(3-4), 287–303.

Huang, C.-R., Hsieh, S.-K., Hong, J.-F., Chen, Y.-Z., Su, I.-L., Chen,

Y.-X., & Huang, S.-W. (2010). Chinese Wordnet: Design, Implementation, and Application of an Infrastructure for Cross-Lingual Knowledge Processing. *Journal of Chinese Information Processing*, *24*(2), 14–23.

Hulden, M. (2009a). Foma: a Finite-State Compiler and Library. In *Proceedings of the Demonstrations Session at EACL 2009* (pp. 29–32).

Hulden, M. (2009b). Foma: a Finite-State Compiler Library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session* (pp. 29–32).

Javed, N., & Muralidhara, B. (2018). Semantic Interpretation of Tweets: A Contextual Knowledge-Based Approach for Tweet Analysis. In *Knowledge Computing and Its Applications* (pp. 75–98). Springer.

Jurafsky, D., & Martin, J. H. (2020). *Speech and Language Processing (Draft)*. Hentet.

Kasper, W., Kiefer, B., Krieger, H.-U., Rupp, C. J., & Worm, K. L. (1999). Charting the Depths of Robust Speech Parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (pp. 405–412).

Keet, M. (2018). *An Introduction to Ontology Engineering* (Vol. 1).

Keet, M. (2020). *An Introduction to Ontology Engineering* (Vol. 1).

Khan, Z. C., & Keet, C. M. (2016). Dependencies Between Modularity Metrics Towards Improved Modules. In E. Blomqvist, P. Ciancarini, F. Poggi, & F. Vitali (Eds.), *Knowledge Engineering and Knowledge Management* (pp. 400–415). Cham: Springer International Publishing.

Kildal, J., Fernández, I., Lluvia, I., Lázaro, I., Aceta, C., Vidal, N., & Susperregi, L. (2019). Evaluating the UX Obtained from a Service Robot that Provides Ancillary Way-Finding Support in an Industrial Environment. In *Advances in Manufacturing Technology XXXIII: Proceedings of the 17th International Conference on Manufacturing Research, 10-12 Sep 2019, Belfast* (Vol. 9, p. 61).

Kingsbury, P., & Palmer, M. (2003). Propbank: the Next Level of Treebank. In *Proceedings of Treebanks and Lexical Theories* (Vol. 3).

Kingsbury, P. R., & Palmer, M. (2002). From TreeBank to PropBank. In *LREC* (pp. 1989–1993).

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins,

G., Rusu, A. A., ... others (2017). Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526.

Koller, A., Baumann, T., & Köhn, A. (2018). DialogOS: Simple and Extensible Dialog Modeling.

Kontopoulos, E., Mitzias, P., Riga, M., & Kompatsiaris, I. (2017). A Domain-Agnostic Tool for Scalable Ontology Population and Enrichment from Diverse Linked Data Sources. In *DAMDID/RCDL* (pp. 184–190).

Krötzsch, M., Simancik, F., & Horrocks, I. (2012). A Description Logic Primer. *arXiv preprint arXiv:1201.4089*.

Langkilde, I., & Knight, K. (1998). Generation that Exploits Corpus-Based Statistical Knowledge. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics.*

Lee, S. (2013). Structured Discriminative Model for Dialog State Tracking. In *Proceedings of the SIGDIAL 2013 Conference* (pp. 442–451).

Lee, S., & Eskenazi, M. (2013, August). Recipe For Building Robust Spoken Dialog State Trackers: Dialog State Tracking Challenge System Description. In *Proceedings of the SIGDIAL 2013 Conference* (pp. 414–422). Metz, France: Association for Computational Linguistics.

Lei, S., Wang, X., & Yuan, C. (2019). Word-Based POMDP Dialog Management via Hybrid Learning. *IEEE Access*, *7*, 39236–39243.

Leshcheva, I., Blagov, E., & Pleshkova, A. (2017). Towards a Method of Ontology Population from Heterogeneous Sources of Structured Data. In *2017 IEEE 30th Neumann Colloquium (NC)* (pp. 29–34).

Levin, E., Pieraccini, R., & Eckert, W. (1998). Using Markov Decision Process for Learning Dialogue Strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)* (Vol. 1, pp. 201–204).

Levin, E., Pieraccini, R., & Eckert, W. (2000). A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Transactions on Speech and Audio Processing*, *8*(1), 11–23.

Li, J., Miller, A. H., Chopra, S., Ranzato, M., & Weston, J. (2016). Learning Through Dialogue interactions by asking questions. *arXiv preprint arXiv:1612.04936*.

Liu, B., & Mazumder, S. (2021). Lifelong and Continual Learning Dialogue Systems: Learning During Conversation. *Proceedings of AAAI-2021*.

Liu, J., Li, Y., & Lin, M. (2019). Review of Intent Detection Methods in the Human-Machine Dialogue System. In *Journal of Physics: Conference Series* (Vol. 1267, p. 012059).

Liu, Q., Chen, Y., Chen, B., Lou, J.-G., Chen, Z., Zhou, B., & Zhang, D. (2020). You Impress Me: Dialogue Generation via Mutual Persona Perception. *arXiv preprint arXiv:2004.05388*.

Louvan, S., & Magnini, B. (2020, December). Recent Neural Methods on Slot Filling and Intent Classification for Task-Oriented Dialogue Systems: A Survey. In *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 480–496). Barcelona, Spain (Online): International Committee on Computational Linguistics.

Luckow, A., Cook, M., Ashcraft, N., Weill, E., Djerekarov, E., & Vorster, B. (2016). Deep Learning in the Automotive Industry: Applications and Tools. In *2016 IEEE International Conference on Big Data (Big Data)* (pp. 3759–3768).

Madonna, M., Monica, L., Anastasi, S., & Di Nardo, M. (2019). Evolution of Cognitive Demand in the Human-Machine Interaction Integrated with Industry 4.0 Technologies. *WIT Transactions on The Built Environment*, *189*, 13–19.

Makki, J. (2017). OntoPRiMa: A Prototype for Automating Ontology Population. *International Journal of Web/Semantic Technology (IJWesT)*, *8*.

Maurtua, I., Fernández, I., Tellaeche, A., Kildal, J., Susperregi, L., Ibarguren, A., & Sierra, B. (2017). Natural Multimodal Communication for Human–Robot Collaboration. *International Journal of Advanced Robotic Systems*, *14*(4), 1–12.

Maynard, D., Bontcheva, K., & Augenstein, I. (2016). Natural Language Processing for the Semantic Web. *Synthesis Lectures on the Semantic Web: Theory and Technology*, *6*(2), 1–194.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., & Grishman, R. (2004). The NomBank Project: An Interim Report. In *Proceedings of the Workshop Frontiers in Corpus Annotation at NLT-NAACL 2004* (pp. 24–31).

Miller, G. A. (1995). WordNet: a Lexical Database for English. *Communications of the ACM*, *38*(11), 39–41.

Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., & Lutz, C. (2012). *OWL 2 Web Ontology Language Profiles*

(Second Edition). Retrieved from `https://www.w3.org/TR/owl2-profiles/` (Accessed: 12 March 2022)

Motik, B., & Patel-Schneider, P. F. (2012). *OWL 2 Web Ontology Language. Structural Specification and Functional-Style Syntax (Second Edition).* Retrieved from `https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/` (Accessed: 12 March 2022)

Mrkšić, N., Séaghdha, D. O., Thomson, B., Gašić, M., Su, P.-H., Vandyke, D., ... Young, S. (2015). Multi-Domain Dialog State Tracking Using Recurrent Neural Networks. *arXiv preprint arXiv:1506.07190*.

Muscetti, M., Rinaldi, A. M., Russo, C., & Tommasino, C. (2022). Multimedia Ontology Population through Semantic Analysis and Hierarchical Deep Features Extraction Techniques. *Knowledge and Information Systems*, *64*(5), 1283–1303.

Nadeau, D., & Sekine, S. (2007). A survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, *30*(1), 3–26.

Nam, S., Park, J., Kim, Y., Hahm, Y., Hwang, D., & Choi, K.-S. (2014). Korean FrameNet for Semantic Analysis. In *Proceedings of the 13th International Semantic Web Conference.*

Niles, I., & Pease, A. (2001). Towards a Standard Upper Ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems* (pp. 2–9).

Oborski, P. (2004). Man-Machine Interactions in Advanced Manufacturing Systems. *The International Journal of Advanced Manufacturing Technology*, *23*(3-4), 227–232.

Ohara, K., Fujii, S., Ishizaki, S., Ohori, T., Saito, H., & Suzuki, R. (2004). The Japanese FrameNet Project; An Introduction. In C. J. Fillmore, M. Pinkal, C. F. Baker, & K. Erk (Eds.), *Proceedings of the Workshop on Building Lexical Resources from Semantically Annotated Corpora* (pp. 9–12). Lisbon: LREC 2004.

Olaso Fernández, J. M. (2017). *Spoken Dialogue Systems: Architectures and Applications* (Unpublished doctoral dissertation). Euskal Herriko Unibertsitatea.

Palmer, M. (2009). Semlink: Linking PropBank, VerbNet and FrameNet. In *Proceedings of the Generative Lexicon Conference* (pp. 9–15).

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual Lifelong Learning with Neural Networks: A

Review. *Neural Networks*, *113*, 54–71.

Pease, A., Niles, I., & Li, J. (2002). The Suggested Upper Merged Ontology: A large Ontology for the Semantic Web and its Applications. In *Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web* (Vol. 28, pp. 7–10).

Peng, B., Zhu, C., Li, C., Li, X., Li, J., Zeng, M., & Gao, J. (2020). Few-Shot Natural Language Generation for Task-Oriented Dialog. *arXiv preprint arXiv:2002.12328*.

Peroni, S., Shotton, D., & Vitali, F. (2012). The Live OWL Documentation Environment: A Tool for the Automatic Generation of Ontology Documentation. In A. ten Teije et al. (Eds.), *Knowledge Engineering and Knowledge Management* (pp. 398–412). Berlin, Heidelberg: Springer Berlin Heidelberg.

Persson, A. (2017, May). The Effect of Excluding Out of Domain Training Data from Supervised Named-Entity Recognition. In *Proceedings of the 21st Nordic Conference on Computational Linguistics* (pp. 289–292). Gothenburg, Sweden: Association for Computational Linguistics.

Polkosky, M. D. (2005). Toward a Social-Cognitive Psychology of Speech Technology: Affective Responses to Speech-Based e-Service.

Poveda-Villalón, M. (2016). *Ontology Evaluation: a Pitfall-Based Approach to Ontology Diagnosis* (Unpublished doctoral dissertation).

Poveda-Villalón, M., Espinoza-Arias, P., Garijo, D., & Corcho, O. (2020). Coming to Terms with FAIR Ontologies. In C. M. Keet & M. Dumontier (Eds.), *Knowledge Engineering and Knowledge Management* (pp. 255–270). Cham: Springer International Publishing.

Poveda-Villalón, M., Gómez-Pérez, A., & Suárez-Figueroa, M. C. (2014). OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, *10*(2), 7–34.

Poveda-Villalón, M., Fernández-Izquierdo, A., & García-Castro, R. (2019, January). *Linked Open Terms (LOT) Methodology.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.2539305`

Quamar, A., Lei, C., Miller, D., Ozcan, F., Kreulen, J., Moore, R. J., & Efthymiou, V. (2020). An Ontology-Based Conversation System for Knowledge Bases. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (pp.

361–376).

Raad, J., & Cruz, C. (2015, 11). A Survey on Ontology Evaluation Methods..

Rodriguez-Castro, B., Torok, L., & Hepp, M. (n.d.). *Printer Vocabulary Ontology.* Retrieved from `http://purl.org/opdm/printer#`

Romero, D., Stahre, J., Wuest, T., Noran, O., Bernus, P., Fast-Berglund, A., & Gorecky, D. (2016). Towards an Operator 4.0 Typology: a Human-Centric Perspective on the Fourth Industrial Revolution Technologies. In *Proceedings of the International Conference on Computers and Industrial Engineering (CIE46), Tianjin, China* (pp. 29–31).

Rosch, E. (1977). Human Categorisation. *Studies in Cross-Cultural Psychology*, *I*(1), 1–49.

Roy, N., Pineau, J., & Thrun, S. (2000). Spoken Dialogue Management Using Probabilistic Reasoning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (pp. 93–100).

Sagot, B., & Fivser, D. (2008). Building a Free French WordNet from Multilingual Resources. In *OntoLex.*

Salomão, M. M. M., Torrent, T. T., & Sampaio, T. F. (2013). A Linguística Cognitiva Encontra a Linguística Computacional: Notícias do Projeto FrameNet Brasil. *Cadernos de Estudos Linguísticos*, *55*(1), 7–34.

Sanagavarapu, L. M., Iyer, V., & Reddy, R. (2022). A Deep Learning Approach for Ontology Enrichment from Unstructured Text. *Cybersecurity & High-Performance Computing Environments: Integrated Innovations, Practices, and Applications,.*

Scalise, R., Li, S., Admoni, H., Rosenthal, S., & Srinivasa, S. S. (2018). Natural Language Instructions for Human–Robot Collaborative Manipulation. *The International Journal of Robotics Research*, *37*(6), 558–565.

Scheutz, M., Cantrell, R., & Schermerhorn, P. W. (2011). Toward Human-Like Task-based Dialogue Processing for HRI..

Schuler, K. K. (2006). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon* (Unpublished doctoral dissertation). University of Pennsylvania.

Segers, R., Vossen, P., Rospocher, M., Serafini, L., Laparra, E., & Rigau, G. (2015). ESO: A Frame Based Ontology for Events and Implied Situations. *Proceedings of MAPLEX*, *2015*.

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., & Chanona-

Hernández, L. (2012). Syntactic Dependency-Based n-grams as Classification Features. In *Mexican International Conference on Artificial Intelligence* (pp. 1–11).

Skantze, G. (2007). *Error Handling in Spoken Dialogue Systems-Managing Uncertainty, Grounding and Miscommunication* (Unpublished doctoral dissertation). KTH, School of Computer Science and Communication (CSC), Speech, Music and Hearing, TMH.

Smith, C., Crook, N., Charlton, D., Boye, J., De La Camara, R. S., Turunen, M., ... others (2011). Interaction Strategies for an Affective Conversational Agent. *Presence*, *20*(5), 395–411.

Stenmark, M., & Nugues, P. (2013). Natural Language Programming of Industrial Robots. In *IEEE ISR 2013* (pp. 1–5).

Suárez-Figueroa, M. C., Gómez-Pérez, A., & Fernández-López, M. (2012). The NeOn Methodology for Ontology Engineering. In *Ontology Engineering in a Networked World* (pp. 9–34). Springer.

Suárez-Figueroa, M. C., Gómez-Pérez, A., & Villazón-Terrazas, B. (2009). How to Write and Use the Ontology Requirements Specification Document. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"* (pp. 966–982).

Subirats, C., & Sato, H. (2003). Surprise! Spanish FrameNet. In *In Proceedings of the Workshop on Frame Semantics at the XVII. International Congress of Linguists.*

Suendermann, D., Evanini, K., Liscombe, J., Hunter, P., Dayanidhi, K., & Pieraccini, R. (2009). From Rule-Based to Statistical Grammars: Continuous Improvement of Large-Scale Spoken Dialog Systems. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 4713–4716).

Susperregi, L., Fernández, I., Fernández, A., Fernández, S., Maurtua, I. n., & López de Vallejo, I. (2012). Interacting with a Robot: A Guide Robot Understanding Natural Language Instructions. In J. Bravo, D. López-de Ipiña, & F. Moya (Eds.), *Ubiquitous Computing and Ambient Intelligence* (pp. 185–192). Berlin, Heidelberg: Springer Berlin Heidelberg.

Taulé, M., Martí, M. A., & Recasens, M. (2008). AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Lrec.*

Teixeira, M. S., Maran, V., & Dragoni, M. (2021). The Interplay of a Conversational Ontology and AI Planning for Health Dialogue Management. In *Proceedings of the 36th Annual ACM*

Symposium on Applied Computing (pp. 611–619).

Thomas, B. J., & Jenkins, O. C. (2012). RoboFrameNet: Verb-Centric Semantics for Actions in Robot Middleware. In *IEEE International Conference on Robotics and Automation (ICRA), 2012* (pp. 4750–4755).

Thorne, C. (2017). Chatbots for Troubleshooting: A Survey. *Language and Linguistics Compass*, *11*(10), e12253.

Thrun, S. (1998). Lifelong Learning Algorithms. In *Learning to learn* (pp. 181–209). Springer.

Vandenbussche, P.-Y., Atemezing, G. A., Poveda-Villalón, M., & Vatant, B. (2017). Linked Open Vocabularies (LOV): a Gateway to Reusable Semantic Vocabularies on the Web. *Semantic Web*, *8*(3), 437–452.

Van Noord, G., Bouma, G., Koeling, R., & Nederhof, M.-J. (1999). Robust Grammatical Analysis for Spoken Dialogue Systems. *Natural Language Engineering*, *5*(1), 45–93.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is All You Need. *Advances in neural information processing systems*, *30*.

Veiga, G., Pires, J., & Nilsson, K. (2009). Experiments with Service-Oriented Architectures for Industrial Robotic Cells Programming. *Robotics and Computer-Integrated Manufacturing*, *25*(4-5), 746–755.

Veron, M., Ghannay, S., Ligozat, A.-L., & Rosset, S. (2019, April). Lifelong Learning and Task-Oriented Dialogue System: What Does It Mean? In *International Workshop on Spoken Dialogue Systems Technology.* Siracusa, Italy.

Villani, V., Pini, F., Leali, F., & Secchi, C. (2018). Survey on Human–Robot Collaboration in Industrial Settings: Safety, Intuitive Interfaces and Applications. *Mechatronics*, *55*, 248–266. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0957415818300321`

W3C, S. W. G. (2013). *SPARQL 1.1 Overview.* Retrieved from `https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/` (Accessed: 12 March 2022)

Wang, S. I., Ginn, S., Liang, P., & Manning, C. D. (2017). Naturalizing a Programming Language via Interactive Learning. *arXiv preprint arXiv:1704.06956*.

Wang, Z., Chen, H., Wang, G., Tian, H., Wu, H., & Wang, H. (2014). Policy Learning for Domain Selection in an Extensible Multi-Domain Spoken Dialogue System. In *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 57–67).

Ward, W., & Issar, S. (1994). *Recent Improvements in the CMU Spoken Language Understanding System* (Tech. Rep.). Carnegie-Mellon University Pittsburgh, PA School of Computer Science.

Wei, Z., Liu, Q., Peng, B., Tou, H., Chen, T., Huang, X.-J., ... Dai, X. (2018). Task-Oriented Dialogue System for Automatic Diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 201–207).

Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., & Young, S. (2015). Semantically Conditioned LSTM-Based Natural Language Generation for Spoken Dialogue systems. *arXiv preprint arXiv:1508.01745*.

Wessel, M., Acharya, G., Carpenter, J., & Yin, M. (2019). OntoVPA—an Ontology-Based Dialogue Management System for Virtual Personal Assistants. In *Advanced Social Interaction with Agents* (pp. 219–233). Springer.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., ... others (2016). The FAIR Guiding Principles for Scientific Data Management and Stewardship. *Scientific Data*, *3*(1), 1–9.

Williams, J. D. (2008). The Best of Both Worlds: Unifying Conventional Dialog Systems and POMDPs. In *Ninth Annual Conference of the International Speech Communication Association.*

Williams, J. D. (2013). Multi-Domain Learning and Generalization in Dialog State Tracking. In *Proceedings of the SIGDIAL 2013 Conference* (pp. 433–441).

Wu, W., Guo, Z., Zhou, X., Wu, H., Zhang, X., Lian, R., & Wang, H. (2019). Proactive Human-Machine Conversation with Explicit Conversation Goals. *arXiv preprint arXiv:1906.05572*.

Yadav, V., & Bethard, S. (2019). A Survey on Recent Advances in Named Entity Recognition from Deep Learning Models. *arXiv preprint arXiv:1910.11470*.

Yakoub, M. S., Selouani, S.-A., & Nkambou, R. (2015). Mobile Spoken Dialogue System Using Parser Dependencies and Ontology. *International Journal of Speech Technology*, *18*(3), 449–457.

Yang, Y., Li, Y., & Quan, X. (2020). Ubar: Towards Fully End-to-End Task-Oriented Dialog Systems with GPT-2. *arXiv preprint arXiv:2012.03539*.

Yoshino, K., Watanabe, S., Le Roux, J., & Hershey, J. R. (2013).

Statistical Dialogue Management Using Intention Dependency Graph. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing* (pp. 962–966).

You, L., & Liu, K. (2005). Building Chinese FrameNet Database. In *2005 International Conference on Natural Language Processing and Knowledge Engineering* (pp. 301–306).

Young, S. (2006). Using POMDPs for Dialog Management. In *2006 IEEE Spoken Language Technology Workshop* (pp. 8–13).

Young, S., Gać, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., & Yu, K. (2010). The Hidden Information State Model: A Practical Framework for POMDP-based Spoken Dialogue Management. *Computer Speech & Language*, *24*(2), 150–174.

Young, S., Gašić, M., Thomson, B., & Williams, J. D. (2013). POMDP-Based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*, *101*(5), 1160–1179.

Young, S. J. (2000). Probabilistic Methods in Spoken–Dialogue Systems. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, *358*(1769), 1389–1402.

Zhou, H., Huang, M., Zhang, T., Zhu, X., & Liu, B. (2018). Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32).

Zhou, H., Young, T., Huang, M., Zhao, H., Xu, J., & Zhu, X. (2018). Commonsense Knowledge Aware Conversation Generation with Graph Attention. In *IJCAI* (pp. 4623–4629).

Appendix A

TODO Competency Questions

Table A.1: Competency questions for the TODODW module.

BCQID	CQID	Competency Question	Answer
BCQ1	CQ01	What are the world elements that are present in the scenario?	
		What are the world elements that are present in the guidance scenario?	laboratoryA, laboratoryB, meetingRoomA
BCQ1	CQ02	What does the world element contain? (if it contains something)	
		What does the world element "laboratory" contain? (if it contains something)	machineA, machineB

Table A.2: Competency questions for the TODODFA module.

BCQID	CQID	Competency Question	Answer
BCQ2	CQ03	Which is the user intent given a series of key elements from the user command?	
		Which is the user intent if the key elements extracted from the user command are "show", "next"?	Next
BCQ2	CQ04	Which is the intended action given a series of key elements from the user command?	
		Which is the intended action if the key elements extracted from the user command are "show", "next"?	Action_next
BCQ2	CQ05	To which lexical unit corresponds a key element from the user command?	
		To which lexical unit corresponds the key element "abandons"?	abandon.V
BCQ2	CQ06	Which is the value assigned to the lexical unit that corresponds to the key element obtained from the user command?	
		Which is the value assigned to the lexical unit that corresponds to the key element "abandons"?	abandon
BCQ2	CQ07	To which lexical unit corresponds a given value?	
		To which lexical unit corresponds "abandon"?	abandon.V
BCQ2	CQ08	To which frame head corresponds the lexical unit?	
		To which frame head corresponds "abandon"?	stop_Activity_stop, switch_Replacing, stop_Process_stop
BCQ2	CQ09	To which frame corresponds the frame head?	
		To which frame corresponds the frame head "stop_Activity_stop"?	activity_stop
BCQ2	CQ10	To which user intent corresponds the frame?	
		To which user intent corresponds the frame "Activity_stop"?	next

BCQ2	CQ11	To which action(s) corresponds the user intent?	
		To which action(s) corresponds the user intent "Next"?	action_next
BCQ3	CQ12	Which is the action given an argument?	
		Which is the action given the argument "destination"?	action_take
BCQ3	CQ13	Which are the arguments of a specific action?	
		Which are the arguments of the "action_take" action?	origin, destination
BCQ4	CQ14	To which argument does a given key element correspond?	
		To which argument does the key element "laboratory" correspond?	destination, origin
BCQ4	CQ15	To which world element corresponds the lexical unit?	
		To which world element corresponds the lexical unit "laboratory.N"?	laboratoryA, laboratoryB
BCQ4	CQ16	To which world element group corresponds the world element?	
		To which world element group corresponds the world element "laboratory"?	spaces, laboratories
BCQ4	CQ17	To which argument corresponds a given world element or world element group?	
		To which argument corresponds the world element group "spaces"?	origin, destination
BCQ4	CQ18	Which it is the value of the argument according to its type and a world element?	
		Which it is the value for a coordinate-type argument if its value is "laboratoryA"?	x,y,z

Table A.3: Competency questions for the TODODom module (besides the CQ for TODODW and TODODFA).

BCQID	CQID	Competency Question	Answer
BCQ5	CQ19	Which is the action's target system readable information (TSRI)?	
		Which is the action's target system readable information (TSRI)?	execute_next
BCQ5	CQ20	Which is the type of a given target system readable information?	
		Which is the type of the "execute_next" target system readable information?	service
BCQ5	CQ21	*Which is the action-related relevant information according to its TSRI type?*	
		Which is the relevant information for an action of the type "service"?	implementation
BCQ5	CQ22	Which is the type of the TSRI of a given argument?	
		Which is the type of the TSRI of the argument "destination"?	coordinate
BCQ5	CQ23	Which is the TSRI of a given world element?	
		Which is the TSRI of the world element "laboratoryA"?	coordinates_labA, ID_labA
BCQ5	CQ24	Which is the type of the TSRI of a given world element?	
		Which is the type of the TSRI of the world element "laboratoryA"?	coordinate, ID
BCQ5	CQ25	Which is the TSRI of the world element taking into account the type of the argument it corresponds to?	
		Which is the TSRI of the world element "laboratoryA" taking into account that the type of the argument it corresponds to is "coordinate"?	coordinates_labA
BCQ5	CQ26	Which is the TSRI for the argument?	
		Which is the TSRI for the argument "destination"?	destination_TSRI

Table A.4: Competency questions for the TODODM module.

BCQID	CQID	Competency Question	Answer
BCQ1	CQ01	What are the world elements that are present in the scenario?	
		What are the world elements that are present in the guidance scenario?	laboratoryA, laboratoryB, meetingRoomA
BCQ1	CQ02	What does the world element contain? (if it contains something)	
		What does the world element "laboratory" contain? (if it contains something)	machineA, machineB

Table A.5: Competency questions for the TODODM module.

BCQID	CQID	Competency Question	Answer
BCQ6	CQ27	Which is the first step/next step of the dialogue?	
		Which is the first step/next step of the dialogue?	step_greeting
BCQ6	CQ28	The output provides some information to the user: what does it imply?	
		The output is a greeting: what does it imply?	request_userCommand
BCQ6	CQ29	Given a step: which is the step function associated?	
		Given a step "processStep_getAction": which is the step function associated?	stepFunction_getAction
BCQ6	CQ30	Given a step function: which are the implications?	
		Given a step function "stepFunction_obtainAction": which are the implications?	singleOptionImplies: processStep_getActionArgs noOptionsImplies: SResp_NoMoreOptionsRestart
BCQ6	CQ31	Which is the implication if the user response type is incorrect given a system request?	

		Which is the implication if the user response type is incorrect given a system request?	SResp_NotUnderstood
BCQ6	CQ32	Which is the implication if the step function returns more than one output?	
		Which is the implication if the step function "stepFunction_obtainAction" returns more than one output?	SResp_ActionOptionsAvailable
BCQ6	CQ33	Which is the implication if the step function returns more than one output and the length of that output does not exceed the maximum value of possible outputs?	
		Which is the implication if the step function "stepFunction_obtainAction" returns more than one output and the length of that output does not exceed the maximum value of possible outputs?	SResp_ActionOptionsAvailable
BCQ6	CQ34	Which is the implication if the step function returns more than one output and the length of that output exceeds the maximum value of possible outputs?	
		Which is the implication if the step function "stepFunction_obtainAction" returns more than one output and the length of that output exceeds the maximum value of possible outputs?	SResp_TooManyActionOptions
BCQ6	CQ35	Which is the implication if the step function does not return any output?	
		Which is the implication if the step function "stepFunction_obtainAction" does not return any output?	SResp_NoMoreOptionsRestart
BCQ6	CQ36	Which is the implication if the step function returns a single output?	
		Which is the implication if the step function "stepFunction_obtainAction" returns a single output?	processStep_getActionArgs
BCQ6	CQ37	Which is the implication if the user response for the request that corresponds to the step function is "No"?	

		Which is the implication if the user response for the request that corresponds to the step function "stepFunction_YesNoDetermine_NewDialogue" is "No"?	dialMarker_finishDial
BCQ6	CQ38	Which is the implication if the user response for the request that corresponds to the step function is "Yes"?	
		Which is the implication if the user response for the request that corresponds to the step function "stepFunction_YesNoDetermine_NewDialogue" is "Yes"?	dialMarker_restartDial
BCQ6	CQ39	Which is the implication if the user response has not been understood?	
		Which is the implication if the user response has not been understood?	SResp_NotUnderstood
BCQ7	CQ40	The current step involves an interaction with the user: which output should be provided?	
		The "step_greeting" step involves an interaction with the user: which output should be provided?	"Hello! I am at your service."
BCQ7	CQ41	What should be requested to the user?	
		What should be requested to the user?	SResp_NotUnderstood
BCQ7	CQ42	What should be requested to the user if the value of the argument is too broad and the system needs to know a specific characteristic to determine which is the intended value?	
		What should be requested to the user if the value of the argument is too broad and the system needs to know a specific characteristic to determine which is the intended value?	"Which colour is the piece?"
BCQ7	CQ43	What should be requested to the user if the system needs the user to be more specific with the value of the argument because it is too broad?	

		What should be requested to the user if the system needs the user to be more specific with the value of the "destination" argument because it is too broad?	"Could you please be more specific with the destination you want me to take you?"
BCQ7	CQ44	What should be requested to the user if the system needs to know the value of an argument?	
		What should be requested to the user if the system needs to know the value of the "destination" argument?	"Could you please tell me the destination you want me to take you?"
BCQ7	CQ45	What should be requested to the user if the system needs the user to repeat the value for an argument?	
		What should be requested to the user if the system needs the user to repeat the value for the "destination" argument?	"Could you please tell me the destination you want me to take you?"
BCQ7	CQ46	What should be requested to the user if the system wants the user to provide their initial command?	
		What should be requested to the user if the system wants them to provide their initial command?	"Give me your command."
BCQ7	CQ47	What should be requested to the user if the system needs the user to repeat the value for an argument because the one they provided is not compatible with the argument?	
		What should be requested to the user if the system needs the user to repeat the value for an argument because the one they provided is not compatible with the argument?	"Can you verify the information you just told me?"
BCQ7	CQ48	What should be requested to the user if the system needs the user to confirm that the system-provided option for an action is correct?	
		What should be requested to the user if the system needs the user to confirm that "action_next" is the desired action?	"Do you want me to show you the next step?"

BCQ7	CQ49	What should be requested to the user if the system needs to confirm if the user agrees with the suggestion made by the system for a value for an argument?	
		What should be requested to the user if the system needs to confirm if the user agrees with "laboratory A" as the value of an argument?	"I cannot find any meeting room with a machineA, but I can suggest laboratory A, which does have it."
BCQ7	CQ50	What should be requested to the user if the system needs the user to confirm the value inferred for an argument?	
		What should be requested to the user if the system needs the user to confirm that the inferred value "laboratory A" is the correct value for the argument?	"The destination is laboratory A, isn't it?"
BCQ7	CQ51	What should be requested to the user if the system needs the user to confirm the argument to which corresponds a given world element?	
		What should be requested to the user if the system needs the user to confirm that "laboratory A" is the destination?	"Is it the destination?"
BCQ7	CQ52	What should be requested to the user if the system needs the user to confirm that the provided value for the argument for disambiguation is correct?	
		What should be requested to the user if the system needs the user to confirm that the value for the "destination" argument is "laboratory A"?	"Do you want to go to laboratory A?"
BCQ7	CQ53	What should be requested to the user if the system needs the user to confirm that the value of a given argument is incorrect?	
		What should be requested to the user if the system needs the user to confirm that the value of the "destination" argument is incorrect?	"Did I get the destination wrong?"

BCQ7	CQ54	What should be requested to the user if the system needs them to confirm that the provided intent is the intended by the user with their request?	
		What should be requested to the user if the system needs them to confirm that "Next" is the intended by the user with their request "show me the next step"?	"So many interpretations! Do you need the next one?"
BCQ7	CQ55	What should be requested to the user if the system needs to know if the user wants to initiate a new dialogue process?	
		What should be requested to the user if the system needs to know if the user wants to initiate a new dialogue process?	"Do you need anything else?"
BCQ7	CQ56	What should be requested to the user if the system needs the user to confirm that the action and values for its arguments are correct before sending them to the target system?	
		What should be requested to the user if the system needs the user to confirm that the action "action_take" and the value for its "destination" argument "laboratory A" is correct before sending them to the target system?	"So, do you want me to take you to laboratory A, isn't it?"
BCQ7	CQ57	What should be told to the user if a UserRequest can be associated to more than one action?	
		What should be told to the user if their request "step" can be associated to more than one action (action_next, action_previous)?	"Your command can be associated to many actions!"
BCQ7	CQ58	What should be told to the user if the user needs details about an Argument?	
		What should be told to the user if the user needs details about the "Error" argument?	"The error number corresponds to the ID of the error that appears in the machine"

BCQ7	CQ59	What should be told to the user if the value provided for the argument is not compatible with said argument?	
		What should be told to the user if "laboratory A" is not compatible with the argument "Error"?	"'Laboratory' cannot be the error"
BCQ7	CQ60	What should be told to the user if the system has not been able to determine the argument value through the characteristics provided by the user?	
		What should be told to the user if the system has not been able to determine the piece through the characteristics provided by the user (colour)?	"I cannot guess the piece with the information you provided"
BCQ7	CQ61	What should be told to the user if the user input can be associated to more than one argument?	
		What should be told to the user if "laboratory" can be associated to more than one argument (origin, destination)?	"'Laboratory' can be assigned to many arguments!"
BCQ7	CQ62	What should be told to the user if the system has not been able to associate the user's input to one or more actions?	
		What should be told to the user if the system has not been able to associate the user's input to one or more actions?	"I could not associate your command to an action."
BCQ7	CQ63	What should be told to the user if the system has not been able to resolve a command?	
		What should be told to the user if the system has not been able to resolve a command?	"I could not resolve your command."
BCQ7	CQ64	What should be told to the user when the dialogue process has finished?	
		What should be told to the user when the dialogue process has finished?	"Done!"
BCQ7	CQ65	What should be told to the user to greet them?	
		What should be told to the user to greet them?	"Hello! I am at your service."

BCQ7	CQ66	What should be told to the user if the user input is not compatible with any action argument?	
		What should be told to the user if the user input "cake" is not compatible with any argument of the action "take"?	"The item "cake" is not compatible with any argument of the action"
BCQ7	CQ67	What should be told to the user if the system is missing some information?	
		What should be told to the user if the system is missing some information?	"I still need some information."
BCQ7	CQ68	What should be told to the user if the system does not have any more options to present to the user to choose from, but the dialogue will continue?	
		What should be told to the user if the system does not have any more options to present to the user to choose from, but the dialogue will continue?	"I do not have any more options to show you!"
BCQ7	CQ69	What should be told to the user if the system does not have any more options to present to the user to choose from, and the dialogue must be restarted?	
		What should be told to the user if the system does not have any more options to present to the user to choose from, and the dialogue must be restarted?	"I do not have any more options to show you and I will need you to start again."
BCQ7	CQ70	What should be told to the user if, after asking for clarification for the value of an argument, the system still cannot find a value?	
		What should be told to the user if, after asking for clarification for the value of an argument, the system still cannot find a value?	"I cannot find anythin that corresponds to your demand."
BCQ7	CQ71	What should be told to the user if the system has not understood a user input?	
		What should be told to the user if the system has not understood a user input?	"I am sorry, but I have not understood your command."

BCQ7	CQ72	What should be told to the user if a user response can apply to several possible options?	
		What should be told to the user if a user response can apply to several possible options?	"I have many options for your request."
BCQ7	CQ73	What should be told to the user if the system expects a yes/no response and the user provides a content response?	
		What should be told to the user if the system expects a yes/no response and the user says "show me the next step"?	"I only need you to answer yes or no, please."
BCQ7	CQ74	What should be told to the user if the system expects a content response and the user provides a yes/no response?	
		What should be told to the user if the system expects a content response and the user says "no"?	"I have not understood you. I need you to provide a specific response."
BCQ7	CQ75	What should be told to the user if the system has too many alternatives for an element to be resolved?	
		What should be told to the user if the system has too many alternatives for an element to be resolved?	"I have too many options taking into account the information you gave me."
BCQ7	CQ76	What should be told to the user if the system has too many alternatives for an action to be resolved?	
		What should be told to the user if the system has too many alternatives for an action to be resolved?	"Your command can be associated to many actions and I need you to be more specific."
BCQ7	CQ77	What should be told to the user if the system has too many alternatives for an argument to be resolved?	

		What should be told to the user if the system has too many alternatives for an argument to be resolved?	"Your command can be associated to many arguments and I need you to be more specific."
BCQ7	CQ78	What should be told to the user if the system has too many alternatives for a world element to be resolved?	
		What should be told to the user if the system has too many alternatives for a world element to be resolved?	"Your command can be associated to many elements and I need you to be more specific."
BCQ7	CQ79	What should be told to the user to report that the system has understood an user request?	
		What should be told to the user to report that the system has understood an user request?	"Understood!"
BCQ7	CQ80	What should be told to the user if, given that the system has requested the value of an argument, the user input is not compatible with said argument?	
		What should be told to the user if, given that the system has requested the value of the argument "error", the user input "laboratory" is not compatible?	"It cannot be the error."
BCQ8	CQ81	Given some output to the user, it is some input from the user required or not?	
		Given some output to the user, it is some input from the user required or not?	Yes
BCQ9	CQ82	Which is the type of information required by the request provided to the user?	
		Which is the type of information required by a user command request (request_userCommand)?	content
BCQ9	CQ83	Which is the type of response provided by the user?	

		Which is the type of response provided by the user if their input is "show me the next step"?	content
BCQ9	CQ84	The type of the user response obtained fits the information type required by the request?	
		The type of the user response obtained fits the information type required by the request?	Yes

Table A.6: Competency questions for the TODODT module.

BCQID	**CQID**	**Competency Question**	**Answer**
BCQ10	CQ85	Which is the current step of a given dialogue?	
		Which is the current step of a given dialogue?	dialStep_processUserRequest
BCQ11	CQ86	Which is the trace for an action / argument / frame / intent / skill / key element / world element?	
		Which is the trace for a given action?	actionTrace_2021-0302
BCQ11	CQ87	Which are the traces for the dialogues/steps/etc. that take part in the dialogue process?	
		Which is the trace for the dialogue in the dialogue process?	dialogueTrace_2021-0302213252
BCQ11	CQ88	At what time did the dialogue / secondary dialogue start?	
		At what time did the dialogue "dialogueTrace_20210302213252" start?	2021-03-02-T21:32:52.12679
BCQ11	CQ89	At what time did the dialogue / secondary dialogue finish?	
		At what time did the dialogue "dialogueTrace_20210302213252" finish?	2021-03-02-T21:40:52.12679
BCQ11	CQ90	Which is the user request that initiated a given dialogue?	
		Which is the user request that initiated the dialogue "dialogueTrace_20210302213252"?	"Show me the next"
BCQ11	CQ91	Which is the action associated to a given initial user request?	

		Which is the action associated to the initial user request "Show me the next"?	action_next
BCQ11	CQ92	Which is the secondary dialogue associated to a given dialogue / secondary dialogue?	
		Which is the secondary dialogue associated to the dialogue "dialogueTrace_20210302213252"?	"secondaryDialogueTrace_20210302213352"
BCQ11	CQ93	Which is the system request that initiates the secondary dialogue?	
		Which is the system request that initiates the secondary dialogue "secondaryDialogueTrace_20210302213352"?	"I need you to tell me the element you want to see next."
BCQ11	CQ94	Which is the response provided by the system given a secondary dialogue?	
		Which is the response provided by the system given the secondary dialogue "secondaryDialogueTrace_20210302213352"?	"I am still missing some information."
BCQ11	CQ95	Which is the transcription obtained from a user input for a given dialogue/secondary dialogue?	
		Which is the transcription obtained from a user input for the dialogue "dialogueTrace_20210302213252"?	"show me the next"
BCQ11	CQ96	Which key elements were extracted from a given user input for a given dialogue/secondary dialogue?	
		Which key elements were extracted from a given user input for the dialogue "dialogueTrace_20210302213252"?	verb: "show" target: "next"
BCQ11	CQ97	What has been told/requested to the user when a given system step was performed?	
		What has been told/requested to the user for the system step "SReq_Argument"?	"I need you to tell me the origin."
BCQ11	CQ98	What were the possible values to ask about when a system request was performed?	

		What were the possible values to ask about when the system request "SysReqTrace_20210302213352" was performed?	origin, destination
BCQ11	CQ99	Which was the chosen value to ask about in a given iteration of the system request?	
		Which was the chosen value to ask about in the system request "SysReqTrace_20210302213352"?	origin

Appendix B

The Subjective Assessment of Speech System Interfaces Questionnaire (SASSI)

This appendix –and, more precisely, Figure B.1– shows the questions for the SASSI questionnaire used in the three user studies in this paper. The SASSI questionnaire consists of a series of 34 questions, the answer range of which follow a 7-point Likert scale, meaning that each question has a neutral point for answers. For this thesis, this neutral point has been dismissed in some cases[1] in order to obtain more representative evaluations and, thus, each question can be answered in terms of a 6-point Likert scale. In any case, 1 stands for *Strongly disagree* and the maximum value (6 or 7) to *Strongly agree*.

As described in Chapter 6.3.1, extra questions have been added in the guidance and bin-picking user studies to evaluate some areas that were not covered by SASSI, which can be seen in Figure B.2. Some of these questions were extracted from other questionnaires (*Verbosity* and *Productivity*, from the SUISQ questionnaire (Polkosky, 2005)) and others were manually created (*Security*). So as to be integrated into the original SASSI questions, these extra questions were also evaluated by using the same scale as the rest of the SASSI questions.

It is important to note that the statements in SASSI can have positive ("*The system is accurate*") or negative ("*I felt tense using*

[1] *vid.* Chapter 6.3.1.

the system") connotations. So as to obtain consistent evaluations, it is necessary to rescale negative statements so as to be considered as positive. For example, if a negative statement has a score of 1, its rescaled score would be 7 –or 6, depending on the scale (Olaso Fernández, 2017).

Despite being originally in English, a Spanish version, translated by an expert, was provided to the study subjects.

Figure B.1: SASSI questions.

Response Accuracy

---

1. The system is accurate.
2. The system is unreliable.
3. The interaction with the system is unpredictable.
4. The system didn't always do what I wanted.
5. The system didn't always do what I expected.
6. The system is dependable.
7. The system makes few errors.
8. The interaction with the system is consistent.
9. The interaction with the system is efficient.

Likeability

---

10. The system is useful.
11. The system is pleasant.
12. The system is friendly.
13. I was able to recover easily from errors.
14. I enjoyed using the system.
15. It is clear how to speak to the system.

16. It is easy to learn to use the system.
17. I would use this system.
18. I felt in control of the interaction with the system.

Cognitive Demand

---

19. I felt confident using the system.
20. I felt tense using the system.
21. I felt calm using the system.
22. A high level of concentration is required when using the system.
23. The system is easy to use.

Annoyance

---

24. The interaction with the system is repetitive.
25. The interaction with the system is boring.
26. The interaction with the system is irritating.
27. The interaction with the system is frustrating.
28. The system is too inflexible.

Habitability

---

29. I sometimes wondered if I was using the right word.
30. I always knew what to say to the system.
31. I was not always sure what the system was doing.
32. It is easy to lose track of where you are in an interaction with the system.

Speed

33. The interaction with the system is fast.
34. The system responds too slowly.

Figure B.2: Extra questions added in user questionnaires.

Verbosity—guidance and bin-picking UCs

35. I felt like I had to wait too long for the system to stop talking so I could respond.

Productivity—bin-picking UC

36. The system would help me be productive.

Security—bin-picking UC

37. This system allows me to interact with the robot from a secure distance without problems.

Appendix C

Foma Generic Definitions and Rules

This Appendix provides the Foma definitions and rules defined for the initial implementation of KIDE4I, KIDE4Guide, which are intended to be reused in further KIDE4I adaptations.

C.1 Definitions

```
set recursive-define ON

define Num ["0"|1|2|3|4|5|6|7|8|9];

define Dec [1|2|3|4|5|6|7|8|9];

define Number [Dec* Num];

define Interj [{(interjeccio:} Number {)}];

define GenNum [{ms} | {mp} | {fs} | {fp}];

define Adv [{(sadv:} Number {)}];

define Espec [{(espec-} GenNum {:} Number [{(j-} | {(indef-}
| {(pos-} | {(dem-}] GenNum {:} Number {))}];

define CommonNom [{(n-} GenNum {:} Number {)}];

define ProperNom [{(w-} GenNum {:} Number {)}];
```

```
define Nom [[{(n-} | {(w-}] GenNum {:} Number {)}];

define PNPersonal [{(psubj-} GenNum {:} Number {)}];

define SPNPersonal [{(pron-} GenNum {:} Number PNPersonal

{)}];

define SNPersonal [{(sn:} Number SPNPersonal {)}];

define SAdj [{(s-adj:} Number {(s-a-} GenNum {:} Number
{(a-} GenNum {:} Number {}))} | {(s-a-} GenNum {:} Number
{(a-} GenNum {:} Number {))}];

define GrupNomCommon [{(grup-nom-} GenNum {:} Number
CommonNom* {)} | {(grup-nom-} GenNum {:} Number CommonNom*
SAdj {)}];

define GrupNom [{(grup-nom-} GenNum {:} Number Nom* {)} |
{(grup-nom-} GenNum {:} Number Nom* SAdj {)}];

define SNCommon [{(sn:} Number [Espec | GrupNomCommon |
SAdj | SPdeNom]+ {)}];

define SPde [{(sp-de:} Number SN {)}];

define SN [{(sn:} Number [Espec | GrupNom | SAdj | SPde]+
{)}];

define SPde [{(sp-de:} Number SN {)}];

define SN [{(sn:} Number [Espec | GrupNom | SAdj | SPde]+
{)}];

define SNDest [{(sn:} Number [Espec | GrupNom | SAdj |
SPdeNom | SNDest]+ {)}];

define SPdeNom [{(sp-de:} Number SNCommon {)}];

define SNCommon [{(sn:} Number [Espec | GrupNomCommon |
SAdj | SPdeNom]+ {)}];

define SPdeNom [{(sp-de:} Number SNCommon {)}];

define SNCommon [{(sn:} Number [Espec | GrupNomCommon |
SAdj | SPdeNom]+ {)}];

define SPdeNom [{(sp-de:} Number SNCommon {)}];
```

```
define SNCommon [{(sn:} Number [Espec | GrupNomCommon |
SAdj | SPdeNom]+ {)}];

define SNCoord [{(coor-n:} Number SNDest* {)}];

define Prep [{(prep:} Number {)}];

define PrepContent [{(prep-cont:} Number {)}];

define FullStop [{(F-term:} Number {)}];

define SNNum [{(sn:} Number {(numero-num:} Number {))}
{)}*];

define GrupSP [{(grup-sp:} Number [Prep | PrepContent |
SPde | SAdj]+ [SN | SNPersonal]+ [PrepContent | Prep |
SPdeNom | SAdj]* {)}];

define GrupSPCommonContent [{(grup-sp:} Number [PrepContent
| SPde | SAdj]+ [SNCommon]+ [PrepContent | SPdeNom | SAdj]*
{)}];

define SNDest [{(sn:} Number [Espec | GrupNom | SAdj |
SPdeNom | SNDest | GrupSPCommonContent]+ {)}];

define PN [{(patons:} Number {(paton-} [GenNum | {s}] {:}
Number {))}];

define Infinitive [{(infinitiu:} Number {(inf:} Number
{(forma-inf:} Number {)))}];

define Verb [{(verb:} Number {)} | {(verb:} Number Infinitive
{)}];

define SAdvInt [{(sadv:} Number {(adv-interrog:} Number {))}];


define MorfPron [{(morf-pron:} Number {)}];

define GVVerb [[{(grup-verb:} | {(grup-verb-inf:}] Number
Verb {)}];

define Dest [SNDest [{)}+ SAdj | {)}+ SNNum | {)}+ SNCoord]*
| SNDest | GrupNom [{)}+ SAdj | {)}+ SNNum]* | SNCoord];

define GrupVerbSingle [[{(grup-verb:} | {(grup-verb-inf:}
| {(relativa:} Number {(relatiu-sn:}] Number [{)} | Interj
| Adv | PN | Verb | MorfPron | SN | Infinitive | GrupSP | SAdj
```

```
| SNNum | SAdvInt | FullStop | GVVerb | SNCoord]+ {)}];
```

C.2 Rules

```
define VerbDest [[{(grup-verb:} | {(grup-verb-inf:}]
Number [Verb | Infinitive]+ {)} | Infinitive | Verb];

define SPDest [Dest )* GrupSPCommonContent];

define GrupVerbDest [GrupVerbSingle | Dest | SN |
SPDest];

define TagGrupVerbDest [GrupVerbDest @-> "<GV>" ...
"</GV>"];

define TagSN [SN @-> "<DEST>" ...  "</DEST>"];

define TagSP [GrupSPCommonContent @-> "<SP>" ...
"</SP>"];

define TagVerbDest [VerbDest @-> "<GV-DEST>" ...
"</GV-DEST>"];

define TagDest [Dest @-> "<DEST>" ...  "</DEST>"];

define RemoveFullStop FullStop -> 0;

define RemoveEspec Espec -> 0;

define Deletions RemoveFullStop .o.  RemoveEspec;

define TagPrep [[Prep | PrepContent] @-> "<PREP>" ...
"</PREP>"];

define TagSPDest [SPDest @-> "<SP-DEST>" ...
"</SP-DEST>"];

regex Deletions .o.  TagGrupVerbDest .o.  TagVerbDest .o.
TagSPDest .o.  TagSP .o.  TagDest .o.  TagPrep;
```

Appendix D

Documentation for User Studies

Estudio de usuario - Robot Guía

Instrucciones

26 de noviembre de 2021

El escenario

El escenario para esta experimentación es el siguiente:

> *El usuario accede a las instalaciones de Tekniker, que le son desconocidas en parte o en su totalidad. En la entrada, existe un robot guía con el que puede interactuar para preguntarle dónde está un espacio, persona, etc. concreto o dirigirle a un espacio a partir de una necesidad que el usuario exponga.*

Material proporcionado

Para llevar a cabo la experimentación, se proporcionará una serie de material y equipamiento por parte de los responsables de la experimentación.

• Teléfono móvil

El medio por el cual se interactuará con el robot será la aplicación móvil KIDE4I. Para ello, se proporcionará un teléfono móvil con la aplicación instalada, por lo que no es necesario usar dispositivos personales.

• Mapas del edificio

Se adjunta un documento en el que constan los mapas de las distintas plantas del edificio sobre el cual el robot guía emulará su función. En dicha documentación figura una selección de los destinos a los cuales el robot puede guiar.

En cada uno de los mapas, la localización de los lugares está señalada con un número, cuya descripción está en una lista en la parte inferior del documento. Esta descripción incluye los elementos que contiene cada destino.

Interactuar con el robot

Como se ha indicado anteriormente, la comunicación con el robot se realizará a través de KIDE4I, que tiene integrado el sistema de diálogo que permitirá dicha interacción.

• El sistema de diálogo

El sistema de diálogo es capaz de interpretar comandos expresados en lenguaje natural hablado y traducirlos a comandos entendibles por el sistema de destino (en este caso, Teknibot).

La interacción con el sistema de diálogo es libre, por lo que es posible, por ejemplo, referirse al destino de forma indirecta [1]. De todas formas, su diseño actual establece una serie de restricciones a la hora de lanzar un comando de voz:

1. El sistema de diálogo no soporta coordinación. Es decir, no podría resolver secuencias del tipo "*Llévame a la cafetera* ***y*** *a la sala*". El sistema tendrá en cuenta solo una de las acciones y en la mayoría de los casos será capaz de obtener una interpretación, aunque no está garantizada la correcta resolución del comando.

2. El sistema de diálogo no soporta oraciones compuestas (es decir, secuencias con más de un verbo). Por ello, en secuencias del tipo "***Tengo*** *una reunión,* ***necesito*** *una sala*", como en el caso anterior, el sistema tendrá en cuenta solo una de las secuencias y, aunque sea capaz de obtener una interpretación, no está garantizado que ésta sea correcta.

    - Como excepción, secuencias como "***Quiero ir*** *al baño*" sí están soportadas.

[1] Por ejemplo, '*Quiero beber*' cuando la intención es dirigirse a un lugar con agua o cualquier líquido para ingerir.

- **KIDE4I**

Para poder interactuar con el robot a través de KIDE4I, es necesario seguir las siguientes instrucciones.

La interfaz de la aplicación es la que se muestra en la siguiente imagen:

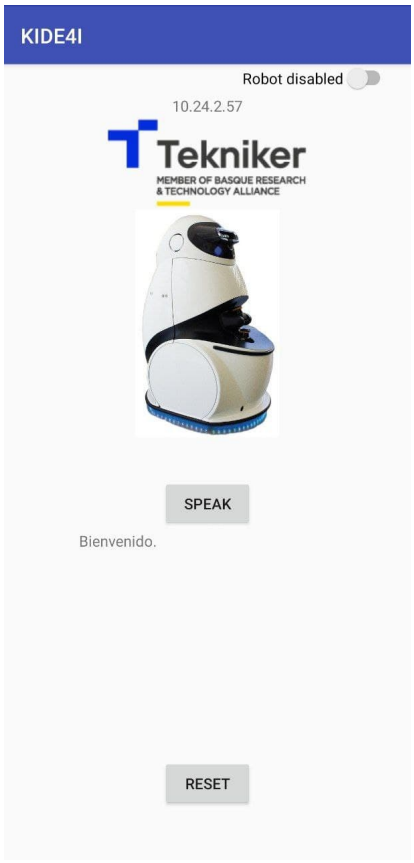


Figura 1: Interfaz de la aplicación

Se aprecian dos botones: `SPEAK` y `RESET`, para interactuar con el robot y reiniciar el diálogo, respectivamente.

Al iniciar la aplicación, es necesario pulsar el botón `SPEAK` y esperar a que el sistema de diálogo emita un saludo, que también se mostrará en la pantalla:

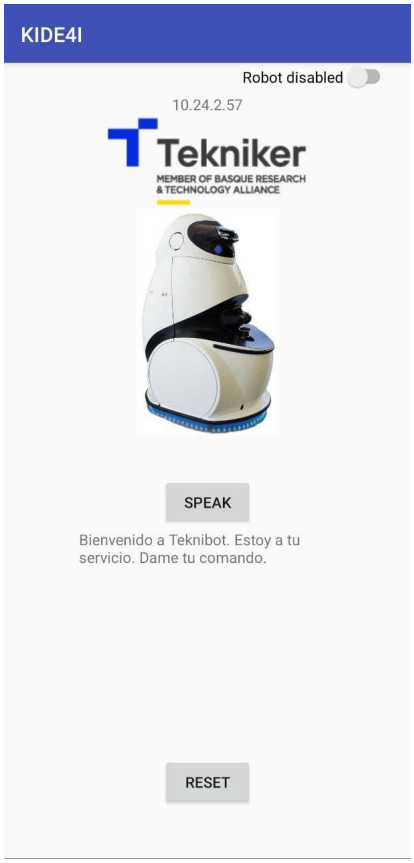


Figura 2: Saludo del sistema

En este momento, el sistema interactuará con el usuario para obtener la información que necesita. Para comunicarse con el robot es necesario pulsar una vez el botón `SPEAK` una vez que el sistema haya reproducido su respuesta. El mensaje "Start speaking" nos indica que la aplicación está esperando un comando o respuesta por parte del usuario:

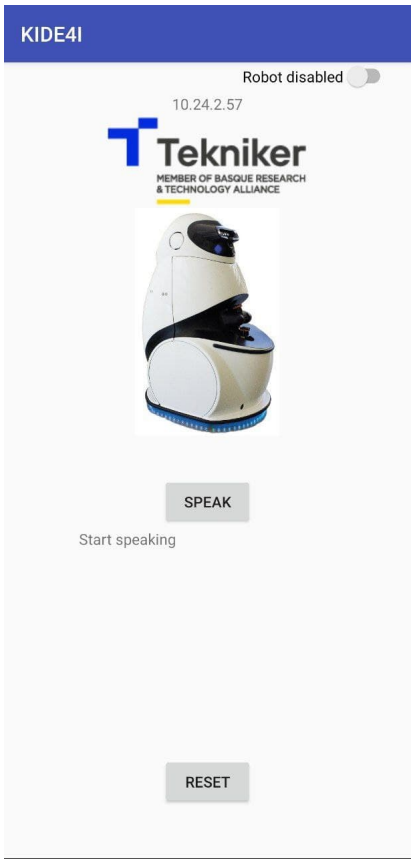


Figura 3: Mensaje “Start speaking"

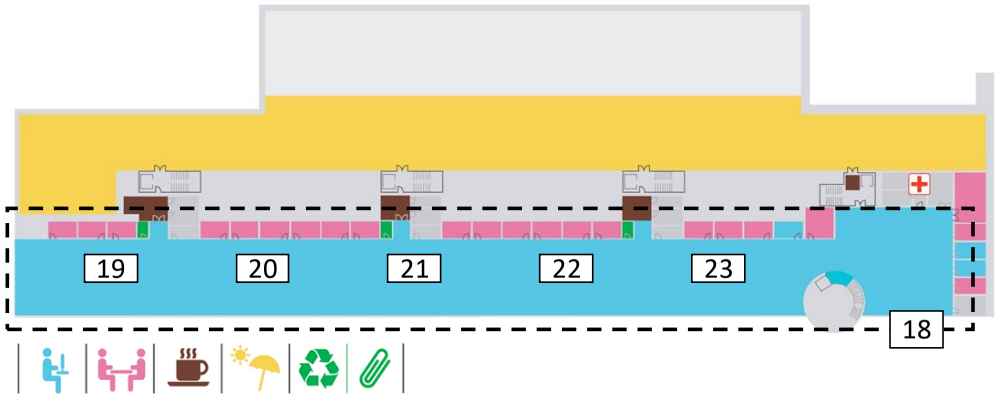
En cualquier momento se puede presionar el botón `RESET` para reinicializar el diálogo, por lo que después de pulsarlo es necesario pulsar de nuevo el botón `SPEAK` y esperar al saludo del sistema de diálogo.

PLANTA 0

1. **Taller de montaje**
2. **Laboratorio de procesos de fabricación**
   - Torno
3. **Auditorio**
   - Micrófono
   - Proyector
   - PC
   - Televisión
4. **Sala Aura 1**
   - PC
   - Televisión
   - Pizarra
5. **Sala Aura 2**
   - PC
   - Televisión
   - Proyector
   - Pizarra
6. **Sala Lurra 1**
   - PC
   - Televisión
   - Proyector
   - Pizarra
7. **Sala Lurra 2**
   - PC
   - Proyector
   - Pizarra
8. **Vestíbulo**
9. **Baño**
   - Retrete
   - Lavabo
   - Papel higiénico
10. **Cafetera**
    - Café
    - Chocolate
11. **Máquina expendedora**
    - Chocolatina
12. **Entrada**
    - Sofá
13. **Recepción**
    - Teléfono
    - Mostrador

PLANTA 1
14
15
16
17
14. Sala Haizea 1
PC
Televisión
Proyector
15. Sala Haizea 2
PC
Televisión
Proyector
Pizarra
16. Sala Sua 1
PC
Televisión
Proyector
Pizarra
17. Sala Sua 2
PC
Proyector
Pizarra

PLANTA 2



18. **Open Space**
19. **Metrología**
    - Ana Garrido
    - Javier Mochales
20. **Tribología**
    - Raquel Bayón
    - Beatriz Fernández
21. **Sistemas de Información Inteligentes**
    - Aitor Arnaiz
    - Nacho Lázaro
22. **Sistemas Automáticos Inteligentes**
    - Ander Ansuategui
    - Iñaki Maurtua
23. **Dirección**
    - Begoña Ansola
    - Luis Uriarte

Estudio de usuario - Bin-Picking

Información para el participante

Introducción

En TEKNIKER desarrollamos robots que puedan ayudar a las personas, tanto en entornos de trabajo como en otras actividades. Para ello, desarrollamos también nuevas formas de comunicación con sistemas de información que sean fáciles de usar para las personas, como por ejemplo el lenguaje hablado.

Actualmente, estos desarrollos están en el marco de una tesis doctoral que tiene como finalidad obtener un sistema de diálogo que pueda ser aplicable, entre otros, a entornos industriales y que, además, sea fácilmente adaptable a distintos casos de uso.

En la fase final de esta tesis doctoral, una parte esencial de nuestra investigación es la realización de estudios de usuario para evaluar nuestras tecnologías con potenciales usuarios finales. Esto nos permite mejorar nuestros desarrollos para que resulten útiles a todas las personas.

Objetivos de este estudio

El estudio en el que vas a participar como objetivo principal **evaluar un sistema de diálogo** y los distintos módulos que lo componen a través de un caso de uso potencialmente relevante en entornos industriales.

El objetivo único del estudio es evaluar nuestras tecnologías con un grupo de personas (posibles usuarios futuros de estas tecnologías). El objetivo no es en ningún caso evaluarte a ti.

En qué consiste el estudio

El escenario del estudio consiste en comunicarse de forma natural –tal y como lo hacemos habitualmente entre personas–, a través de **voz**, con un **brazo robot**, diseñado para tareas de *bin-picking* con cartuchos de tinta.

La finalidad es pedirle que coja una serie de cartuchos y los deposite en un contenedor específico.

Al disponer del robot en esta evaluación, también se evaluarán otras características de la interacción, como la sensación de seguridad del usuario.

Para ello, como participante en el estudio:

1. Te comunicarás con el robot a través de comandos de voz.
2. Para interactuar con el robot, harás uso de una aplicación en un teléfono móvil, el cual tiene integrado el sistema de diálogo a evaluar.
3. Dado un comando de voz que le envíes, el sistema dialogará contigo siempre que lo requiera para obtener la información necesaria para poder mandar el destino final al robot, el cual ejecutará la acción obtenida.
4. Responderás una serie de preguntas para describir tu experiencia utilizando estas tecnologías.

Datos que vamos a recoger

Todos los datos que recojamos durante el estudio van a ser anónimos. Esto significa que tu nombre u otros datos personales identificativos no van a figurar nunca junto a los datos que recojamos durante tu participación.

Los datos que vamos a recoger son los siguientes:

1. Tu perfil de usuario, generado a partir de tu edad, género y experiencia previa con tecnologías similares.
2. Tu opinión sobre las tecnologías que has probado (a través de cuestionarios).
3. Datos sobre el funcionamiento de la tecnología en el momento de la interacción en forma de *logs* y anotaciones recogidas por el personal responsable del estudio (por ejemplo, el resultado obtenido por cada uno de los módulos del sistema de diálogo).

Estos datos serán procesados por el equipo investigador de TEKNIKER involucrado en la tesis doctoral en la que se enmarca este estudio. En ningún caso figurará ningún dato personal identificativo.

Para participar, y teniendo en cuenta los puntos mencionados anteriormente, no es necesario firmar ningún documento de consentimiento informado, ya

que no vamos a guardar ningún dato personal que te pueda identificar en el futuro.

Tu participación en el estudio

Tu participación en este estudio es **voluntaria**. Durante el transcurso del estudio, si por cualquier motivo decides retirarte –aunque sea antes de que termine el estudio–, podrás hacerlo.

–Muchas gracias por tu participación–

Estudio de usuario - Asistente para Procedimientos de Mantenimiento

Instrucciones

Descripción general

El equipo de mantenimiento de la planta del que formas parte se encarga de realizar las tareas de mantenimiento programado, así como del mantenimiento de incidencias de los robots de ABB y de Stäubli. Para resolver dichas incidencias y consultar dudas disponen de 2 manuales:

- Manual de mantenimiento del brazo robótico IRB 120 de ABB
- Manual de mantenimiento del brazo robótico de Stäubli

En lugar de la tradicional búsqueda de información en los manuales, el equipo dispone de un asistente que te facilita la búsqueda de la información en los manuales técnicos asociados a cada robot. Además, el asistente ofrece la posibilidad de guiarte paso a paso en algunos de los procedimientos de mantenimiento descritos en los manuales.

Deberás elegir un mínimo de 3 escenarios de incidencia para cada uno de los robots: uno de los marcados como guiados y al menos otros dos de búsqueda de información general. Para cada uno de ellos, habla con el asistente de manera natural para preguntarle la información que quieres consultar.

Los escenarios guiados corresponden a procedimientos con subtareas y pasos. En estos escenarios podrás realizar distintos tipos de interacciones: utilizar comandos de navegación como "siguiente" o "anterior" para avanzar o retroceder por los pasos del procedimiento; solicitar la lista de herramientas para realizar el procedimiento; solicitar información adicional que extienda la información que te aporta el sistema, como por ejemplo más información sobre la tarea que se está realizando; también podrás preguntar acerca de algún dato concreto como el tamaño de los tornillos a usar, o la posición inicial del robot para llevar a cabo el procedimiento, etc.

Los escenarios de búsqueda general te mostrarán toda la información correspondiente al apartado del manual técnico encontrado en relación a tu pregunta, incluyendo textos e imágenes, de haberlas. Por ejemplo, si en la incidencia tuvieras que cambiar el arnés de cables en la base del robot podrías preguntarle "¿Cómo se cambia el arnés de cables de la base?" y el asistente te mostraría la información correspondiente a ese apartado del manual.

Robot IRB 120 (ABB)

Hay una serie de incidencias con el robot IRB 120. Elige una de las incidencias de tipo guiado y al menos otras dos de consulta general para este robot y hazle una pregunta al asistente para que te proporcione la información necesaria.

Procedimientos guiados

- La batería del robot IRB120 se ha estropeado. Hay que poner una nueva.
- Los ejes del robot IRB120 se han quedado descalibrados tras la última intervención y hay que calibrarlos.

Consultas generales

- Alguno de los cables del arnés se ha estropeado y es necesario cambiar el arnés de cables en la base.
- La tarjeta de interfaz de codificador (EIB) está fallando, es necesario quitarla.
- Estás realizando la tarea de mantenimiento de calibración del robot. Y has llegado al punto de tener que calibrar los ejes 5 y 6 pero no recuerdas como hacerlo de forma manual usando la herramienta de calibración. Pregúntale al sistema para que te muestre la información que necesitas.
- Estás realizando la tarea de mantenimiento de calibración del robot. Y has llegado al punto de tener que calibrar los ejes 5 y 6 pero no recuerdas como hacerlo de forma manual usando la herramienta de calibración. Pregúntale al sistema para que te muestre la información que necesitas.
- Como tarea de mantenimiento te han adjudicado inspeccionar las cubiertas de plástico. Consulta al sistema qué debes hacer para realizarla.

Robot Brazo Stäubli

Han llegado las siguientes incidencias que requieren de tu intervención. Elige una incidencia de tipo guiado y al menos otras dos de tipo general y realiza la pregunta oportuna para que el asistente te proporcione la información necesaria.

Procedimientos guiados

- El robot está sacando ruido chirriante, todo parece que los niveles de aceite no son adecuados, hay que comprobarlos.
- Está llegando un error de fallo del electrodistribuidor, que indica que se ha roto una de sus piezas y por tanto es necesario cambiar el electrodistribuidor completo.

Consultas generales

- Debes realizar el cambio de la junta plana del robot, y para ello lo primero que debes hacer es quitar la actual. Consulta por los pasos necesarios para la retirada de esta.
- Una vez el extraída, debes ponerle al robot una junta plana nueva. Si no sabes cómo montarla consulta al asistente.
- El robot tiene los capós deteriorados y debes cambiarlos.
- El robot ha sufrido un choque y necesita que revises que su funcionamiento es correcto tras la colisión.