applied sciences

MDPI

Article

A Modular Framework for Domain-Specific Conversational Systems Powered by Never-Ending Learning

Felipe Coelho de Abreu Pinna [1], Victor Takashi Hayashi [1,*], João Carlos Néto [1], Rosangela de Fátima Pereira Marquesone [1], Maísa Cristina Duarte [2], Rodrigo Suzuki Okada [2] and Wilson Vicente Ruggiero [1]

[1] Polytechnic School (EPUSP), Universidade de São Paulo, São Paulo 05508-010, Brazil; joaocarlos@larc.usp.br (J.C.N.)
[2] Bradesco Bank, Cidade de Deus, Osasco 06029-900, Brazil; maisa.duarte@bradesco.com.br (M.C.D.)
* Correspondence: victor.hayashi@usp.br

Abstract: Complex and long interactions (e.g., a change of topic during a conversation) justify the use of dialog systems to develop task-oriented chatbots and intelligent virtual assistants. The development of dialog systems requires considerable effort and takes more time to deliver when compared to regular BotBuilder tools because of time-consuming tasks such as training machine learning models and low module reusability. We propose a framework for building scalable dialog systems for specific domains using the semi-automatic methods of corpus, ontology, and code development. By separating the dialog application logic from domain knowledge in the form of an ontology, we were able to create a dialog system for the banking domain in the Portuguese language and quickly change the domain of the conversation by changing the ontology. Moreover, by using the principles of never-ending learning, unsupported operations or unanswered questions create triggers for system knowledge demand that can be gathered from external sources and added to the ontology, augmenting the system's ability to respond to more questions over time.

Keywords: natural language processing; dialog systems; chatbot; intelligent virtual assistants; task-oriented; ontology; knowledge base; never-ending learning; domain-specific; banking; Brazilian Portuguese; framework

Citation: Pinna, F.C.d.A.; Hayashi, V.T.; Néto, J.C.; Marquesone, R.d.F.P.; Duarte, M.C.; Okada, R.S.; Ruggiero, W.V. A Modular Framework for Domain-Specific Conversational Systems Powered by Never-Ending Learning. *Appl. Sci.* **2024**, *14*, 1585. https://doi.org/10.3390/app14041585

Academic Editor: Antonio Fernández-Caballero

Received: 22 December 2023
Revised: 7 February 2024
Accepted: 13 February 2024
Published: 16 February 2024

1. Introduction

Dialog systems require considerable effort when being implemented because they have multiple modules and apply machine learning algorithms to natural language processing and dialog management tasks most of the time. Additionally, they require textual data collection, annotation, and environment user simulation to train the inner modules. Most of the big tech companies, such as Google, Microsoft, and Amazon, have BotBuilder tools available for creating chatbots for any domain. They require little implementation and a lower amount of data to create the initial models used in conversational systems. However, these tools only provide a pretrained intent classifier (by the companies) and are fine-tuned to the specific domain of new chatbots. This approach is hard to maintain as a domain increases in complexity. Every time a new feature is added, new intents must be created. The dialog context is also a limitation of these tools, in which the user is restricted and cannot change topics during the dialog or is required to repeat information previously mentioned.

We present a framework for dialog systems for more complex domains, which require multiple interactions between user and machine to complete one operation. It allows the user to change context at any point of the conversation while tracking all previous information said in the dialog to avoid having to start over when the user switches to a different topic. The architecture and semi-automatic development methods improve the

implementation so as to have a shorter project implementation time so that the system can be delivered to the market more quickly.

When there is a need for an intelligent system to chat with a user who needs to accomplish a specific goal, a task-oriented dialog system is a conventional solution. These systems can hold knowledge of a specific domain to help the user in a specialized task, such as booking a hotel, searching for restaurants, or completing banking transactions. Sometimes, there is a need to have multiple conversational systems in different domains of a similar area, each having a specific knowledge base to meet a unique user need. This work presents a framework for building specific domain conversational systems, which helps develop multiple instances for distinct domains, lowering the amount of rework needed to create a new system for each new domain.

Facilitating the dialog system domain and its knowledge expansion is relevant since smaller iterative improvements have become the norm for software projects. Thus, if the system knowledge base is static, it may require the creation of a new ontology and it is likely that the models will need retraining each time a new dialog needs to be supported. In order to meet this need for faster updates and time to market, this framework proposes an ontology that can be updated during online use by human experts or by automatic knowledge extraction and uses the never-ending learning process.

For that, we propose a modular architecture to make it possible to quickly swap a module's implementation if necessary and keep some modules constant for similar domains (Section 3). BotBuilder tools and other dialog systems that inspired this work are presented in the Related Work section (Section 2). Further, we present the experimental results from a use case via an evaluation and the never-ending learning analysis (Section 4). The proposed framework allows for the creation of multiple implementations of conversational systems for similar domains with low effort in automatic information retrieval from external data sources (Section 5).

The proposed modular framework aims to provide scalable architectures for conversational agents. A dialog system for the banking domain in the Portuguese language was created using the proposed framework as a demonstration of its application, but the framework is intended to be domain-independent. Researchers and practitioners are expected to use the proposed architecture and specialize it for their specific domains by adjusting the training datasets and ontologies but reusing the organization of the proposed architecture. The present work aims to contribute to addressing the research gap in terms of the scalability of dialog systems based on ontologies [1,2].

2. Related Work

There are two main approaches for building a task-oriented chatbot: intent-driven or dialog-driven. In the first one, the main component is the intent classification model, where the response is given by its intent classification. Big tech companies have tools available to them for creating these systems with low effort (Section 2.1). Intent-driven systems perform very well for simple dialogues with a low number of turns. However, their decision tree form makes them increasingly complex and difficult to maintain as the domain increases in complexity.

Conversely, dialog systems (Section 2.2) are dialog-driven and have a solution for dialog management in the design; this includes dialog memory and context switching. This approach requires more custom implementations and is, therefore, harder to implement as quickly as possible with BotBuilder tools. However, it handles the complex domain much more elegantly and is easier to maintain. One significant aspect is that the domain knowledge is separated from the code.

2.1. BotBuilders

BotBuilders are tools for creating simple chatbots with a minimum requirement of data collection or preparation and code development. Some of the most popular tools in this category are Google Dialogflow. (https://dialogflow.cloud.google.com (ac-

cessed on 21 December 2023)), IBM Watson (https://www.ibm.com/watson (accessed on 21 December 2023)), and Facebook WitAI (https://wit.ai (accessed on 21 December 2023)). In order to create a new chatbot using these tools, you need to create some intentions for the system to recognize in the user's messages and provide a few examples for each one of them. This is possible because these companies have machine learning models trained for supported languages, and when users add examples to their specific needs, these models are fine-tuned to them.

This allows for rapidly creating new chatbots with very little NLP knowledge needed compared to creating a complete dialog system from scratch. However, these tools have some limitations in more complex domains and in keeping with the dialog context. Moreover, the language needs to be supported by their solution and may not be ready for certain vocabulary in a very specific domain.

As an intelligent agent increases in complexity, the number of needed intents rapidly increases. This makes the system harder to maintain, and for each new intent added, more examples are required for the classifier to correct each case separately, decreasing one of the main benefits of these tools, which is that of needing few examples. Keeping with the context of the dialog can also be difficult; in most of these tools, the context is given by a set of global variables, and topic switching must be addressed by the designer, usually by creating dialog paths in a tree structure. This requires that the creator thinks about all the possible paths the user may follow, and all exceptional cases must be mapped to deal with each one individually. This can fix a user into a specific path during the dialog; for example, in the money transfer operation, the user may be required to give each mandatory piece of information always in a specific order, and any changes in the topic during this process may be prevented.

2.2. Dialog Systems

In order to address these problems, dialog systems introduced solutions in their design that allow free context switching by the user, especially the Belief Tracker module of the dialog management step. One of the dialog systems that does this and inspired the framework of this work is Pydial [3]. Another aspect of dialog system architecture that makes it easier to maintain in more complex scenarios is the separation of the domain knowledge and the code. This separation, common to expert systems, makes it possible to change some functionality by only modifying the ontology without having to change the code. This is impossible for traditional solutions, in which the domain knowledge is intrinsic to the implementation.

Rasa [4] is a conversational AI platform that uses the modules of dialog systems to create conversational systems. It provides NLP models to fine-tune to the domain with a few examples, such as BotBuilder tools, while having the possibility to deal with more complex domains with the aspects of dialog systems. However, they require a larger amount of development to create a new chatbot or virtual assistant than those tools.

Most of the similar dialog systems do not have built-in security solutions. One example of privacy by design in this area is Snips [5], which is an embedding voice platform that is used to perform the NLU tasks of a dialog system without the need to be online with third-party servers that process the messages in a centralized manner. As shown there and in this work, it is possible to have a full dialog system in small devices, such as smartphones, by adopting the privat- by-design principle.

2.3. Comparison

A comparison of the proposed framework and the related work found in the literature is presented in this subsection and in Table 1. This discussion considers the following aspects:

- Learning: how the conversational system learns and expands its scope;
- Control: the level of control over the solution;
- Reusable: how reusable some of the modules of the conversational system are;
- Availability: the language or platform in which the solution is available;

- Context: it is related to the possibility of the platform to either be able to automatically manage the context of a conversation or to allow for developing a code for this;
- All-in: indicates whether the platform offers multiple services related to the task of understanding natural language.

These platforms are observed to seek to remove obstacles regarding the development of conversational agents, given that they present a high or medium level of usability due to the friendly interface and documentation. All platforms offer reconstructed entities and intentions and offer form filling as a form of composition, except Wit.ai and Watson conversation, which do not offer prebuilt intentions and the Watson conversation platform also offers the option of composing via block diagrams, in addition to the option of filling out the form. The DialogFlow and Amazon Lex platforms were the only ones that presented online integration, and DialofFlow presented integration with 14 systems. DialogFlow and Wit.ai were the only ones that, in addition to having SDK availability, also have integration via Webhook enabled. Note that, among the platforms compared, the only one that presented all the automatic resources analyzed was DialogFlow, contemplating the standard fallback intention, automatic context, and intentions linked on the same platform.

To the best of the authors' knowledge, this work is one of the first to propose a framework for dialog system construction for a specific domain, with a proof of concept in Brazilian Portuguese for the banking domain considering scalability aspects. In other languages, there are specialized ontologies, such as financial industry business ontology (FIBO) [6], but there was no similar knowledge base available for the banking domain in Brazilian Portuguese, which motivated the creation of the semi-automatic knowledge acquisition method.

There is a lack of benchmarks to evaluate natural language processing in Portuguese, which motivates the creation of some initiatives [7], but as far as we are concerned, there is no natural language benchmark for the banking domain in Brazilian Portuguese. Even in other languages, benchmarks for dialog system evaluation are new [8–10]. This gap should be subject to future research efforts, and such works might support formal comparisons of the different approaches using measurable criteria.

Table 1. Comparison of the proposed framework with related work.

Category	Solution	Learning	Control	Reusable	Cloud	Context	Security	All-in
Botbuilders	DialogFlow	manual	low	no	dependant	yes	no	yes
	Watson	manual	low	no	dependent	yes	no	no
	WitAI	manual	low	no	dependent	no	no	no
Dialog Systems	Pydial	iterative	medium	no	independent	yes	no	yes
	Rasa	iterative	high	no	independent	yes	no	yes
	Snips	manual	medium	no	independent	yes	yes	no
	Proposed	semi automatic	high	yes	independent	yes	yes	yes

3. Architecture

In this section, we present the architecture of the proposed framework for dialog systems. Each module has its inputs and outputs defined by a data structure. Consequently, all implementations should conform to the same protocol and could be exchanged without modifications to the rest of the system. Furthermore, there is the possibility of using this framework for multiple communication channels, knowledge bases, and applications, which can use the dialog system as the natural language interface for its users.

3.1. Architecture Overview

The overall architecture of the framework follows the design of previous dialog systems, such as [3], with four central modules: Natural Language Understanding (NLU), Belief Tracker, Policy, and Natural Language Generation (NLG). Some additional modules are present in the architecture to split responsibilities in a deliverable dialog system. The framework architecture is presented in Figure 1.

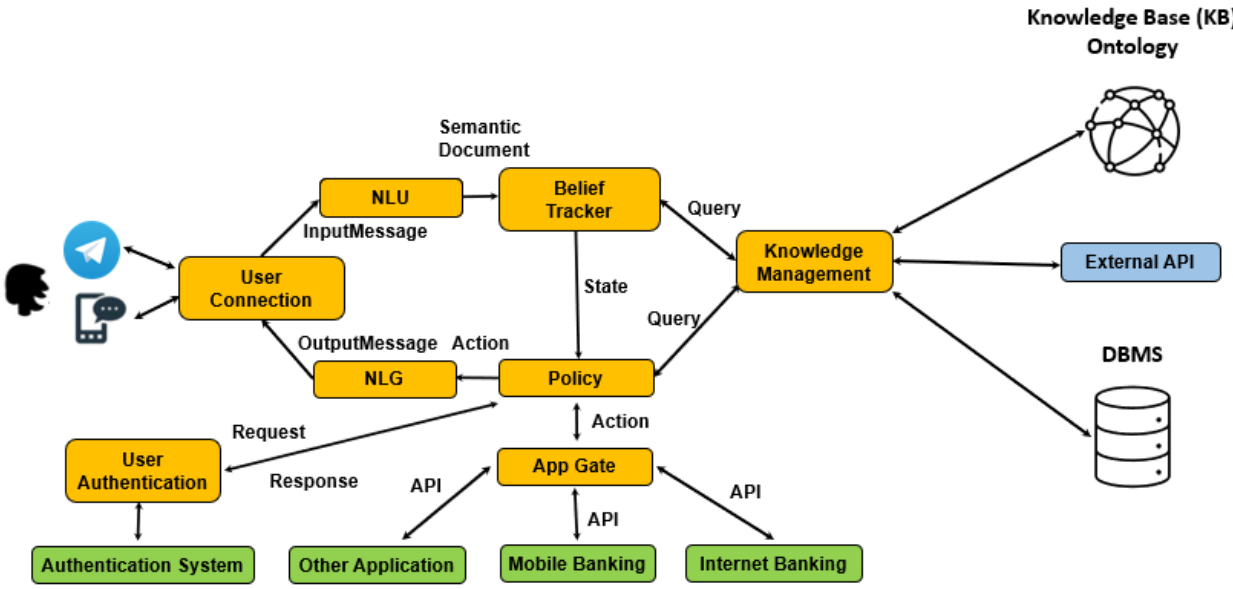

Figure 1. Architecture of the dialog system framework.

On the left side, we have the communication channels that could be used by the user, which could be in textual or audio form. On the right side are some of the knowledge sources the dialog system could use. In this work, we focus on ontologies, which is the preferred method, but others are also possible. At the bottom, some possible applications can use the dialog system as a new interface to interact with the user by employing natural language. A description of each module of the proposed framework is as follows:

- User Connection: this deals with all communication channel interfaces and converges all incoming messages to a specific format of InputMessage so that the system can process all kinds of messages in the same way;
- NLU: the Natural Language Understanding module has to interpret the user messages and extract helpful semantic information to assist the system in responding to the user, producing the Semantic Document with all the relevant semantic information;
- Belief Tracker: this updates the current dialog state with the new user message. This module keeps the short-term memory that contains information mentioned previously in this conversation;
- Policy: the dialog policy observes the current state and selects which action should be taken by the agent;
- NLG: the Natural Language Generation module transforms the action chosen by the policy into a natural language sentence to be sent to the user, delivered by the same channel through which the user sent the message;
- Knowledge Management: this has the role of managing the domain knowledge; it fetches external APIs, ontologies, or relational databases to provide domain information to the Belief Tracker and Policy modules;
- App Gate: this acts as a connector between the dialog system and the application. After a sufficient number of exchange messages between the user and the dialog system for a specific operation, the Policy module triggers the action for this module

to request an operation. This operation is performed by the external application, and its result returns to the dialog system to be included in the response;

- User Authentication: when the user requests an operation, this module authenticates the user. Authentication is not required when the user only asks informational questions. These questions are the user's doubts about the operations available or specific domain knowledge, whereas transactional messages aim to perform a task.

The data passed between modules are of standard types to allow each module to be replaced independently from others. The Semantic Document should hold semantic information extracted from the user message, such as classified intents, named entities, sentiments, and desired operations. That information is extracted by the NLU, usually by task-specialized machine learning models. The State contains the current dialog state, holding the data from the most recent Semantic Document and other information mentioned previously in the dialog; this is a representation of the short-term memory of the dialog system. The Action is the representation of the system's reaction to the current state; it holds the system intent, operation, and other slots needed to generate a response. For example, in a response about an account balance, the action might have an informed intent about the check balance operation with the slot balance with the balance available to the user. These data types can be extended to support other use cases; the data presented are a minimal example.

External services can be accessed via the dialog system by the APIs called by the App Gate module; these services can, thus, provide a conversational interface with their customers. The APIs provided by these services should be REST APIs that expose the core functionality of the service, allowing the conversational system to complete similar functions to what dedicated apps and webpages can do. Their details will depend on the specific application connected. For an Internet Banking application, a request could send the user account number and the operation check balance, while its response would be the balance for that user. Another possible request, after the user asks for a financial recommendation, might be sending user identification and one area of financial products; then, the response might be a list of recommended products for that customer, containing information such as product name, minimum value, and rate. The responses from the APIs are passed as data in the *Action* so that the NLG can use it to generate responses to the user using information from these external services.

Other dialog systems in works such as [3–5] have similar architectures, where the NLU, Belief Tracker, Policy, and NLG modules are common to all. This work proposes the inclusion of the Knowledge Management, App Gate, User Connection, and User Authentication modules, which perform important tasks for the agent, as described earlier.

Semi-automatic corpus generation based on BNF grammar and text data augmentation was used to train the NLP models used in the NLU and NLG modules, with a similar approach to what was carried out in [11–13] regarding the production of training text data. This allows for quickly changing the domain in which the models are trained because the grammar is capable of producing examples of annotated sentences for the specific domain with its vocabulary. The same approach was used to produce the test corpus for these models, which produces a similar set to the training and is the result of a real data evaluation set, which is not expected to be the same.

The ontologies were created manually by the authors based on the interaction with mobile applications and internet banking for browsers of Brazilian banks, selecting the most common operations and required data to complete them via the graphic interface. Protégé version 4.3 (https://protege.stanford.edu (accessed on 21 December 2023)) is an open-source tool for building and visualizing ontologies. It was used for creating the ontologies of the dialog system. The database used for accessing the ontology was the triplestore GraphDB (https://graphdb.ontotext.com (accessed on 21 December 2023)).

The data were obtained automatically from external sources using robotic process automation (RPA) TagUI (https://github.com/aisingapore/TagUI (accessed on 21 December 2023)), a free automation tool from AI Singapore [14]. PDF, DOC, and HTML files available

online were processed using the semi-automatic ontology development method presented in the literature [15]. This method is based on natural language processing (i.e., SpaCy and NLTK [16]), which was used to process Brazilian Portuguese texts that describe banking operations and services. This method also uses the Python OwlReady library [17].

The ontology development method presented in [15] and depicted in Figure 2 was enhanced with automatic triple selection between words based on their similarity and triple classification (hierarchical or nonhierarchical) based on the type of verb using WordNet.Br version 1.0. [18].

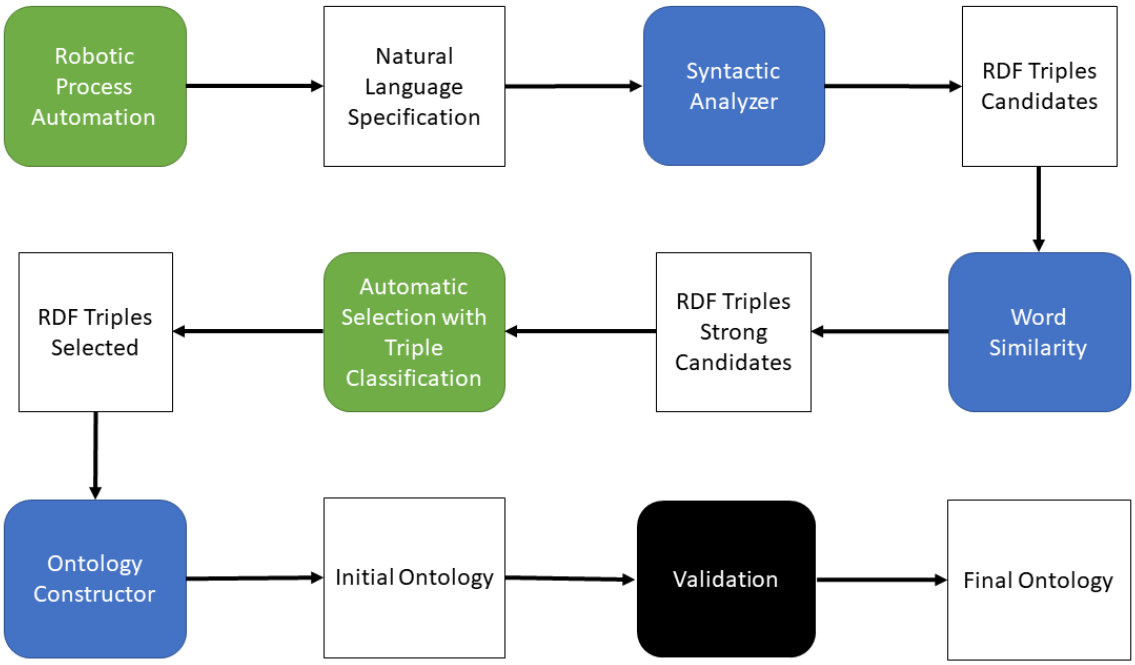

Figure 2. Semi-automatic ontology development method. The rounded boxes are the processing modules of the semi-automatic method, whereas the squared white boxes are the output files at each processing step. The blue boxes are modules from the original semi-automatic ontology development method. The green boxes are additional or enhanced modules in this work. The black box is the semi-automatic ontology validation step.

All modules were designed using tools for semi-automatic development, which allows the engineer to create diagrams of the solution and then generate the equivalent code. Specifically, the IBM Rhapsody version 7.6.1. (https://www.ibm.com/products/systems-design-rhapsody (accessed on 22 December 2023)) tool was used, in which the data structures and modules were given an interface definition, leaving the implementation to be completed. This facilitated the migration of the dialog system to multiple environments.

A mobile application for this dialog system was created for the iOS system. This shows that the architecture proposed can be fully implemented in mobile devices without the need for communication with third-party systems. Additionally, this is the device most of the clients will use to interact with the intelligent agent.

The never-ending learning process was conducted using an in-memory ontology to improve the execution time during the knowledge producer modules' learning step. At the end of each iteration, the working ontology was exported to a file that could then be used by the dialog system, and it was imported into GraphDB or Protégé. The import and export of ontologies were performed using Owlready 2 version 0.45 [17], which allowed for flexible ontology manipulation.

Except for one of the relation populator modules, the work of which was carried out by OpenNRE, the other modules from the architecture shown in Figure 3 were based on simple machine learning models trained for the domain. However, other solutions might

be as easily integrated as knowledge producers by producing candidate knowledge and training from other examples in the current ontology.

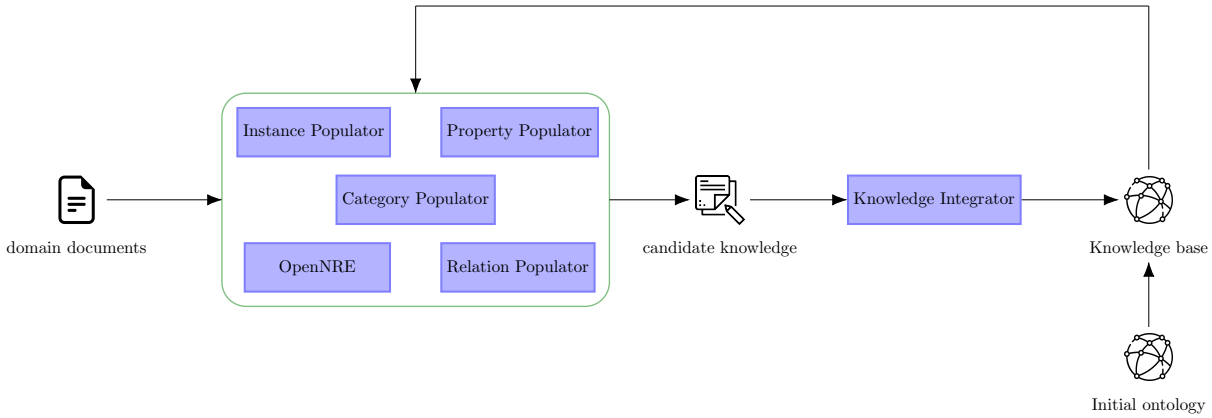

Figure 3. Never-ending learning process.

The architecture is not limited to any specific language or domain and can be implemented in various contexts. Later on, we will demonstrate an example of its application in the banking domain using the Portuguese language. One of the advantages of this architecture is its flexibility, as it can be adapted to other domains by changing the ontology. Furthermore, it can be implemented in different languages and even non-Latin alphabets as long as the NLU and NLG algorithms are adjusted accordingly.

3.2. Scalable Architecture

In addition to being functional, a conversational system architecture must be scalable and support a set of nonfunctional requirements.

When considering the proposed architecture, the system execution environment must be able to scale up and scale down (ideally in a preventive manner) to avoid the need for contingency actions for availability and performance requirements. The system must serve multiple users with low latency.

The solution proposed is an architecture based on microservices implemented in containers to promote the scalability of the execution environment. For communication between containers, the system uses message queues based on Kafka technology. Microservices act as specific servers, being defined by a specific functionality or responsibility for action. The microservices mentioned are organized in a communication infrastructure (execution environment) that allows them to interoperate with the required scalability.

In order to map the components in microservices, the processing times obtained in experiments carried out were taken into account, as well as the degree of coupling between the components. Therefore, the implementation of the following microservices and their respective containers was proposed:

- NLU: a microservice that mainly contains the existing NLU component;
- BeliefTracker, Policy, and KnowledgeManagement (KM): microservices that contain the aforementioned components. Note that KM is included not as a component but, rather, as a library;
- NLG: a microservice that mainly contains the NLG component and KnowledgeManagement (KM). Again, KM is included as a library;
- Conversational Agent: a microservice that contains the main code (orchestrator) of the conversational agent. In an orchestrated microservices architecture, this is the central microservice.

In addition to the mentioned microservices, there are also other components designed exclusively for the proposed solution:

- Session: a microservice to simulate the opening of a user session with the system;
- RLog: a component for capturing performance metrics;
- Queues: Kafka technology for asynchronous communication;
- Containers: Docker technology to facilitate the deployment of each microservice with the management of its dependencies;
- Container manager: Kubernetes technology, which consists of an open source container management platform, is also an orchestrator of these containers, their resources, and their scale. After generating the Docker image for each microservice, by using Kubernetes, it is possible to manage the resources that each one will use and scale from, depending on the availability of resources and demand.

In an orchestrated architecture, there is a central element that makes the decision on the flow to be followed to carry out the processing. In the case of the conversational system, this central element is the Conversational Agent microservice. Figure 4 presents the centralization of logical communication with the Conversational Agent microservice, which decides which microservice to call at each processing step.

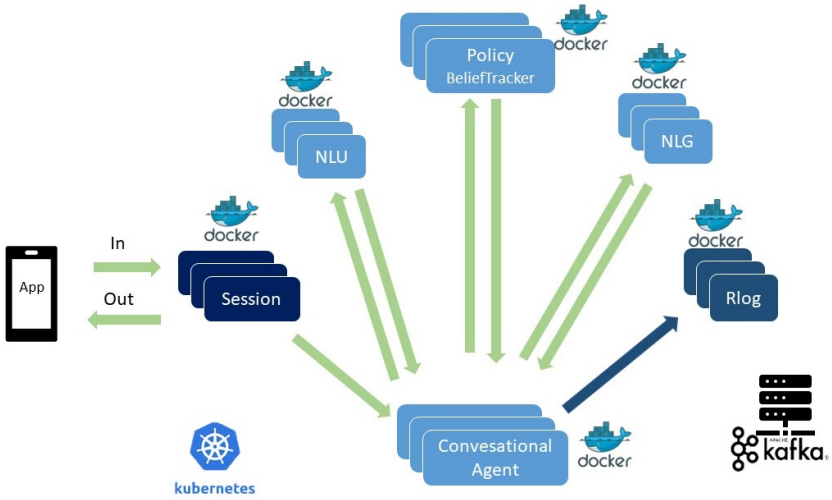

Figure 4. Proposed scalable architecture of the dialog system framework. Green arrows indicate synchronous communication and blue arrows indicate asynchronous communication.

3.3. Natural Language Processing

The NLU and NLG are the application natural language modules. Both need some adaptation for the specific domain. The NLU module classifies the sentences and identifies the named entities, some of which are domain-independent and domain-dependant. The approach proposes a reduction in rework when using the framework by having the intent classification be domain-independent, which could be one of the following classes: greetings, goodbye, perform, inform, confirm, and not confirm. These intents are enough for the completion of tasks and question-answering in most domains. Should a new intent be added, it would be necessary to change the NLU to produce the new classification and update Policy to handle this intent.

Other required classifications can be domain-dependent, such as operation, sentiment, and question type. One important aspect is that every possible label for each classification has a one-to-one mapping with an ontology element so that the ontology has the knowledge to address any inquiry detected by the NLU. With this mapping, it is easy for Knowledge Management to find the correct ontology element that the sentence is referring to; the ontology searches for the node with the classification label, and its properties or related nodes are sent to the other modules. This relevant approach allows for the Belief Tracker and Policy modules to process the dialog based on only the intent and Knowledge Management, and thus, it does not need to be changed for new domains unless a feature with a particular dialog flow is needed.

The context-independent slots filled by the NLU can be carried out by a general named entity recognition system that recognizes, for example, names, places, and dates. The domain-specific slots, which are needed for the tasks supported by the dialog system and their associated external application, can be implemented by some sentence-tagging algorithm, for example, using a recurrent neural network [19]. It is essential that the slot label also has a one-to-one mapping with ontology elements in both cases. Since Knowledge Management can deliver the mandatory slots for a supported operation, the Policy module can then verify if these slots are filled before triggering an execution for an external application. This approach can be easily used if the slots produced by the NLU have the same label as the required operation properties in the ontology.

The NLG module can be implemented using templates for each possible action taken by the dialog system, with placeholders for the response slots [3]. Another way is to have a generative model, eliminating the need for the designer to plan all possible responses and create a unique template for each. A machine translation model is an especially well-suited model; it can transform a set of triples from the ontology into a natural language sentence [20]. With this approach, the knowledge of what should be given in response is also contained in the ontology of the system.

The data needed to train the models of these modules can be created by domain specialists of a curatorship team, who know the users' behavior and language and can produce examples similar to what the user will say with the dialog system. An approach to reducing the amount of manual work and accelerating the application of the framework is to have a semi-automatic process of corpus creation.

3.4. Ontology

The ontology should have all the knowledge needed about the domain for the system to perform its tasks. Moreover, this information should be structured in a particular manner so that the Knowledge Management module can fetch the information. This work proposes the division into three ontologies:

- Task ontology: this ontology contains the primary knowledge to perform the dialog system tasks, including supported operations, mandatory slots, and disambiguation options. It is designed in such a way that the ontology classes are the tasks that can be performed, and each ontology individual is the task that one does not need to disassemble. These ontology individuals are related to the data that need to be demanded to carry out the operation.
- Support ontology: this ontology supports the system with additional information that is required by the task ontology, and it can be fetched in this system. In other words, it is the ontology that contains the long-term data. Such an ontology has its structure assembled from the data that are defined as relevant in the task ontology.
- Domain ontology: the knowledge needed for solving questions about the domain is present in this ontology. This knowledge can be definitions, relations, or properties, describing the knowledge of subjects related to the tasks; it does not describe the steps for its accomplishment but describes the clear doubts inherent to the activities.

There must be consistency among all ontologies, with the same terms for the operations in all of them. For example, the task ontology may require a set of slots related to a person to complete a money transfer. If the user wishes to transfer money to someone who is already known, this person's information can already be in the Support ontology so that all of their information can be fetched from this ontology without the need to be asked from the user during the dialog. In the same way, if the user wants to clarify doubts about any task that appears in the task ontology, such details must be contained in the domain ontology.

3.5. Never-Ending Learning

Never-ending learning is a learning approach that combines multiple tasks and constraints coupled together to acquire more knowledge and improve future learning [21]. A knowledge base in a never-ending learning system is updated constantly as more data are

acquired and self-reflection is performed, which prevents the system from being stuck in a learning plateau.

A study case for this proposal is never-ending language learning (NELL), a system designed to learn the web by starting from an initial ontology with hundreds of categories and relations between them. NELL has been running continually forever in a semi-supervised manner, and its knowledge base has millions of beliefs, showing that an agent could improve its reading over time. This behavior of continuous learning over time is what is desired in a conversational system, which could learn to answer new questions without explicit human intervention.

In this work, our interest is a specific domain, and our knowledge base should only include information relevant to that domain. Therefore, our approach was to apply the concepts of a never-ending learning system to a limited and specific domain. We aim to reduce the learning problem so that only domain knowledge is gained, but continued learning is still in effect, which is accomplished by using a limited quantity of in-domain input texts given to the agent.

Architecturally, the solution is very similar to NELL; the main difference lies in which learning method modules were selected so that the learning tasks and constraints can adapt to the domain and are feasible on a smaller scale. The never-ending learning architecture used is shown in Figure 3. The initial ontology was created manually by using the information available in one of the input documents, which was also used for the knowledge producer models' initial training. Another difference is the amount of data: while NELL was designed for big data, it may have some difficulties in learning specific concepts; in addition to being a defined domain, the dataset of the present work is also restricted, with low redundancy (small data [22]).

The modules in our architecture are the following:

- Instance Populator: A neural network for named entity recognition that was trained to identify classes from the current ontology in the text. These entities will then be candidate instances of those classes to be added;
- Property Populator: A set of simple heuristic rules to identify common ontology properties such as label and description, as well as other custom properties based on the data properties available in the ontology; for example, matching dates in the text with ontology instances that can have a date property;
- Category Populator: Another neural network for named entity recognition, but this time, it is used to search for other entity types that are not currently in the ontology [16];
- OpenNRE: An open-source neural network for relation extraction [23], which is used to find new relations between instances of the current ontology;
- Relation Populator: A random forest classifier [24] trained to find new relations between instances of the current ontology;
- Knowledge Integrator: This is the same module as in NELL; it accesses confidence in new beliefs so as to only integrate high-confidence beliefs and perform consistency checks, for example, if the subject and object of relations are of the correct type.

With these learning tasks, two coupling types could be observed, which is an important factor in never-ending learning systems when avoiding concept-drift [25]. The first is between those modules with different goals, such as the instance populator and the relation populator. In one step, these modules can produce some candidate beliefs that can be inserted into the ontology. Then, in the next step, when passing by the same input data, the relation populator was able to find new relations between instances that were just added to the previous step, showing the effect from one learning task to another. The other type is between thsoe modules with the same tasks. The relation populator and OpenNRE produce relations; at any given step, they produce different candidate beliefs and can then improve their respective predictions in the following step by using the new relations produced from the other module.

The presented architecture can be re-applied in other domains by providing some adjustments. It would be necessary to create knowledge-producing modules that are able to extract entities relevant to the specific domain. Some general modules from NELL or the

shown architecture, like OpenNRE, can be reused for other domains if desired. Additionally, the initial ontology needs to be created to contain the core domain information and some sample knowledge.

Some aspects of the original NELL proposal were not fully explored and should be the object of future research. They include a self-reflection process that is integrated into learning iterations and the further coupling the learning tasks.

4. Experimental Results

Given the main contribution of this work is a framework for a specific domain dialog system, the authors built an example of a dialog system for the banking domain in the Portuguese language as an example of the application of the framework. The development of this dialog system was driven by the semi-automatic generation of corpora, ontologies, and code to construct the system rapidly.

4.1. Specific Banking Domain for the Portuguese Language

The chosen domain for the dialog system is the banking domain, which has a vocabulary with specific terms, and security during its operation is mandatory. The system supports the four main transactions in an internet banking application, which are money transfers, balance checking, extract inquiries, and payments. These operations can be performed, or questions can be asked about them. Thus, the domain-specific classifications needed are operations, which type of question is asked, and the sentiment expressed. Some of the domain-specific slots are account number, bank number, and payment type.

In order to recognize these domain-specific classifications and entities, we trained machine learning models using an artificially generated dataset that was generated by the BNF grammar method mentioned in Section 3.1. We manually constructed a generation grammar and then used it to produce a training dataset with 15,000 labeled examples, as well as a test dataset with 600 examples. Both datasets had equal proportions of all classes. We evaluated this by using the validation dataset, measuring the F1 score of the classification, and naming the entity-recognition tasks. The intent model had a 96.08 F1 score, the NER model had an 85.71 F1 score, the operation classifier had a 100 F1 score, the question classifier had a 94.98 F1 score, and the emotion classifier had a 99.46 F1 score.

The three ontologies described in Section 3.4 created for the banking domain are shown in Figures 5–7. The task ontology contains the four operations: money transfer (“transferencia”), balance (“saldo”), extract inquiry (“extrato”), and payment (“pagamento”), as a child of the root node “procedimento”. The red nodes are classes and need further disambiguation, choosing which type of payment or money transfer the user wants. This ontology also shows that for the “interbancaria” money transfer, the data from the sender “pessoa_remetende” and receiver “pessoa_favorecida” are mandatory.

The domain ontology has a similar graph structure to the task ontology because when the user asks a question about a term in the task ontology, the same spot can be found in the domain ontology to obtain the knowledge to respond to the user’s question. In this ontology, there are the properties for each node, such as the maximum value for the money transfer between different banks, which the user could ask for. There is an annotation in each node with its definition so that questions such as “What is TED?” can be answered with the correct definition given by the specialist consulted during the creation of the ontology.

Finally, the support ontology has information from past operations that can be reused in a future dialog. For example, the yellow node is a money transfer for the same bank (“intrabancaria”) between two persons: “usuario_1” and “maria_silva”. Thus, if the same user wants to transfer to Maria Silva again, her information can be obtained from the support ontology, and the user does not need to repeat her banking information, such as account number and agency. Therefore, the first time a user makes a money transfer to an unknown person, it is necessary to explicitly inform all the information about the receiver, and with the approval of the user, this new contact can be saved in the support ontology to be used again in the future.

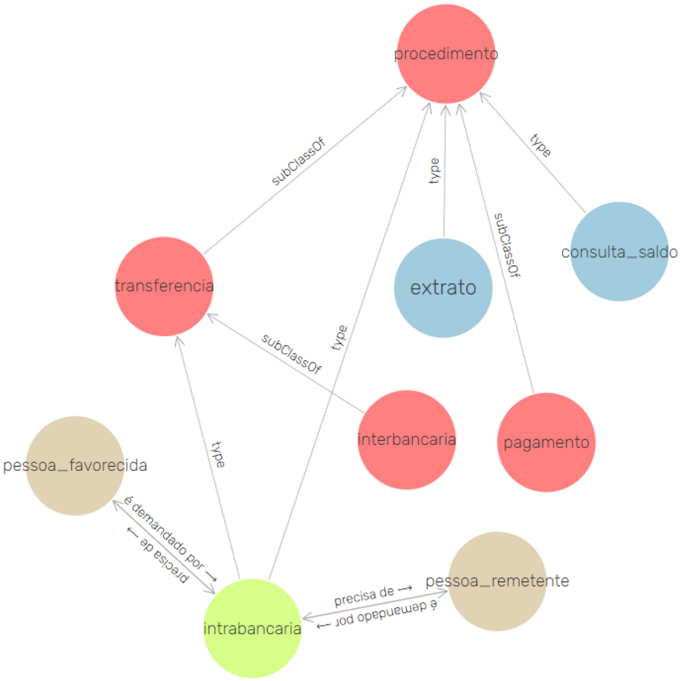

Figure 5. Task ontology. The red nodes are classes representing banking operations, procedure ("procedimento"), money transfer ("transferência"), interbank transfer ("interbancária"), and payment ("pagamento"). The green node is a child class representing the intrabank transfer ("interbancaria"), and the beige nodes represents the objects needed by this interbank transfer: favored person ("pessoa favorecida") and sender person ("pessoa remetende"). The blue nodes are instances of operations: extract inquiry ("extrato") and balance ("saldo"). The relations shown are needs ("precida de") and needed by ("é demandado por").

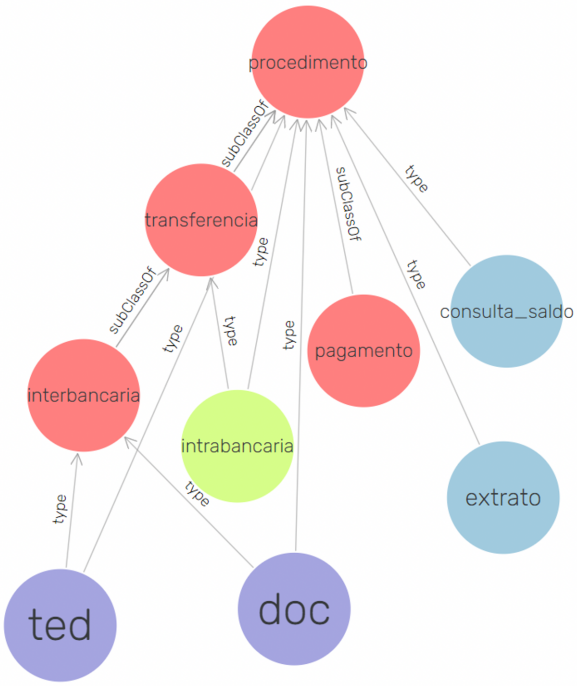

Figure 6. Domain ontology. The red nodes are classes representing banking operations, procedure ("procedimento"), money transfer ("transferência"), interbank transfer ("interbancária"), and payment ("pagamento"). The green node is a child class representing the intrabank transfer ("intrabancaria"). The blue nodes are instances of operations: extract inquiry ("extrato") and balance ("saldo"). The relations shown are needs ("precida de") and needed by ("é demandado por"). The purple noes are instaces of interbank transfers.

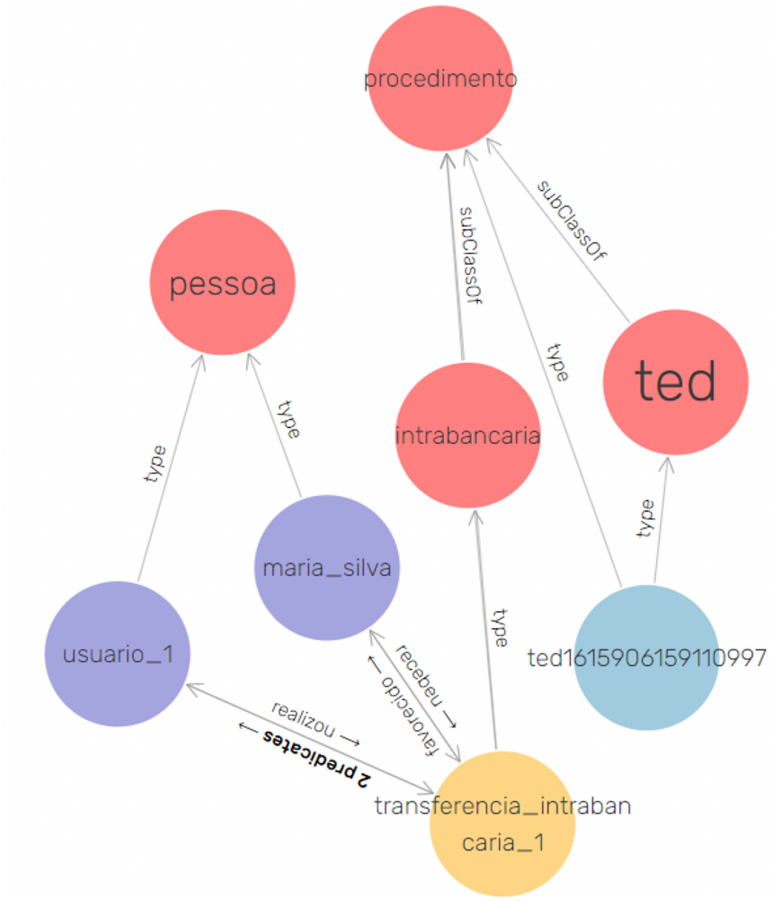

Figure 7. Support ontology. The red nodes are ontology classes: procedure ("procedimento"), intrabank transfer ("intrabancaria"), ted (a type of transfer), and person ("pessoa"). The blue node represent a instance of ted transfer. The purple node repsents instances of people: Maria Silva ("maria_silva") and user 1 ("usuario_1"). The yellow node represent a instance of intrabank transfer beetwen Maria Silva and user 1, the relations of this transfer are done by ("realizou") and recieved ("recebeu").

The estimated time savings in Table 2 were calculated to evaluate the results of using RPA with the semi-automatic ontology development method. Compared to the manual process, the automated process achieved a relative reduction of 76%, from 37 to 9 h, to build an ontology.

Table 2. Automation results using RPA and semi-automatic ontology development method.

Steps for Building an Ontology	Manual	Semi-Automatic with RPA
1. Domain Definition	2	2
2. Existing Ontology Analysis	2	2
3. Terms Definition	5	0.5
4. Hierarchy Definition	4	0
5. Property Definition	6	0.5
6. Definition of Required Data and Relationships	6	0.5
7. Rules and Restrictions Definition	2	0.5
8. Validation	5	2
9. Review	5	5
Total	37	9

4.2. Throughput Evaluation

A simulation process was carried out to test the scalability of the PoC of the conversational system using JMT (Java Modeling Tool version 1.2.5) software [26]. The objective was to determine the number of servers (in this case, containers) of each module (running in parallel) necessary to meet a given message arrival rate.

The closed simulation model was used (see Figure 8); it has no input or output for external requests or responses [27]. Requests (messages) are circulated internally. The total number of requests remains constant. A closed system can be considered one in which the "Out" is connected back to the "In", whereby requests that leave the system immediately enter the system after a delay ("think time") that can be configured. The flow of "Out-to-In" requests defines the throughput of the closed system. When analyzing a closed system, the number of requests (messages) is assumed to be provided. The purpose of the analysis is to determine the throughput of completing the requests.

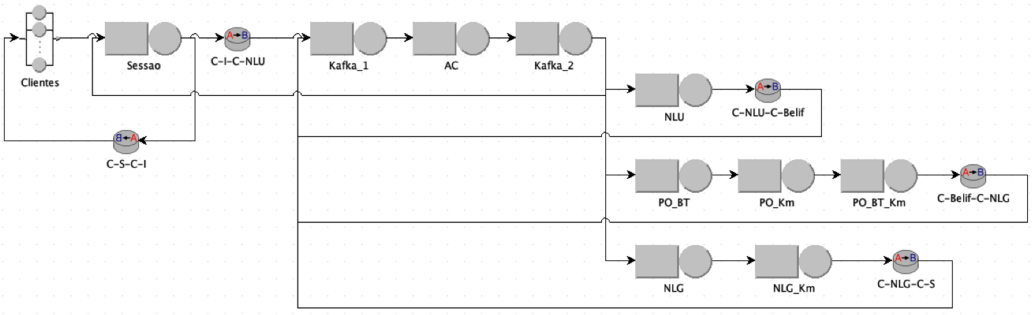

Figure 8. Java modeling tool queue model. The clients go from A letter to B letter in the icons. "Clientes" (from Portuguese) means "Clients" in English, and "Sessao" (from Portuguese) means "Session" in English.

In the solution adopted by the project, all messages enter and leave the Kafka component when passing from one container to another. In the simulation model considered, the Kafka component is represented by two queues: Kafka_1 and Kafka_2. Kafka_1 receives all the messages, and Kafka_2 sends the messages. However, for Kafka_2 to know how to route messages to the correct destination, a simulator feature was used that allows for identifying the messages leaving a queue destined for the next queue. This feature consists of identifying messages through classes. The following classes were defined:

- C-I (blue): The class of messages arriving from clients;
- C-NLU (red): The class of messages that go to the NLU queue;
- C-Belief (green): The class of messages that go to the Belief Tracker + Policy + KM queue;
- C-NLG (pink): The class of messages that go to the NLG + KM queue;
- C-S (orange): The class of messages that go to the Session queue.

At the exit of each queue, the message is converted to the class of the next queue before being forwarded to the Kafka_1 queue. Message classes are represented by different colors in the simulation results. The simulation was carried out using the closed model for the case of 1,000,000 msg/day to determine how many users are served simultaneously. The execution times of the modules used in the simulation were based on the average times obtained during the execution in eight virtual machines with 6 GB of memory and two virtual CPUs each. The obtained execution times used for throughput evaluation are presented in Table 3.

The following scenario was used in this simulation:

- The number of containers in the PO_Km module = 5;
- The number of NLG module containers = 11;
- The execution time of the modules based on Table 1;

- The delay in generating a new message by the client after receiving a response from the previous message = 7 s;
- The boundary condition, where the use of modules is less than or equal to 0.7.

In order to determine how many users were served simultaneously, corresponding to a flow of 11.57 messages per second, the number of users varied (50–150) when observing the flow corresponding to each number of users. The simulation results can be seen in the graph of the flow by number of users, illustrated in Figure 9.

Table 3. Execution times used for throughput evaluation.

Module	Average Execution Time (ms)
NLG	659.69
PO_KM	310.51
PO	83.55
NLU	44.80
PO_BT_KM	16.68
ROUTER	13.18
NLG_KM	8.81
SESSION	8.69
AC	1.38
PO_BT	0.04
BT	0.03
BT_KM	0.01

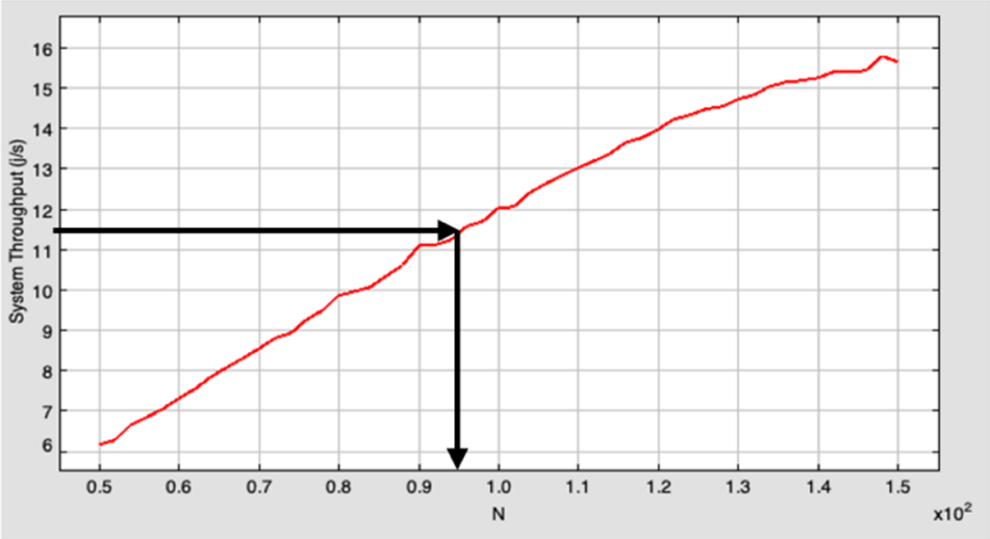

Figure 9. Throughput vs. the number of users. Considering the closed model, the flow of 11.5 msgs per second (horizontal black arrow) corresponds to approximately 95 customers (vertical black arrow). The red line represents the relationship between the entire system throughput and the total customers in the system.

An approximate flow rate of 11.5 messages per second is observed to correspond to 95 users with a "think time" of 7 s.

The graphs in Figures 10 and 11, respectively, illustrate the processor usage by the PO_Km and NLG modules with variations in the number of users.

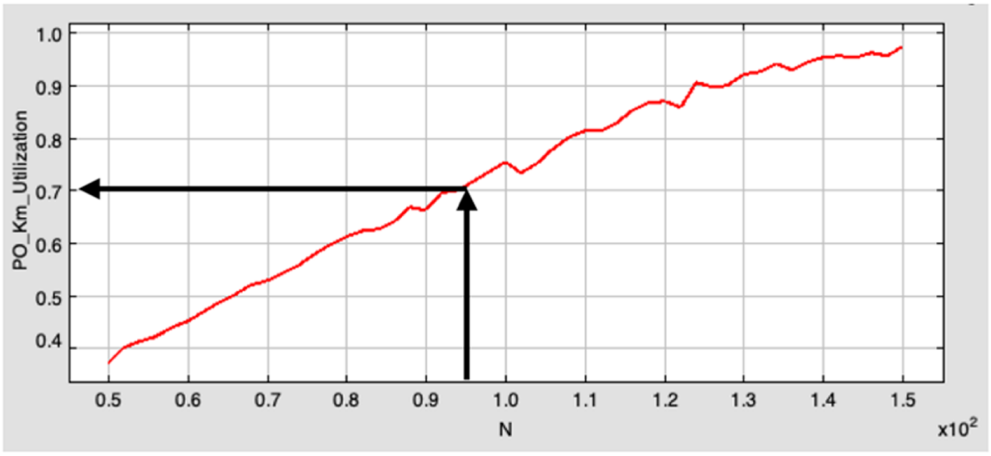

Figure 10. PO_Km module processor utilization. Approximately 95 customers (vertical black arrow) correspond to 70% utilization (horizontal black arrow). The red line represents the relationship between the PO_Km module processor utilization and the total customers in the system.

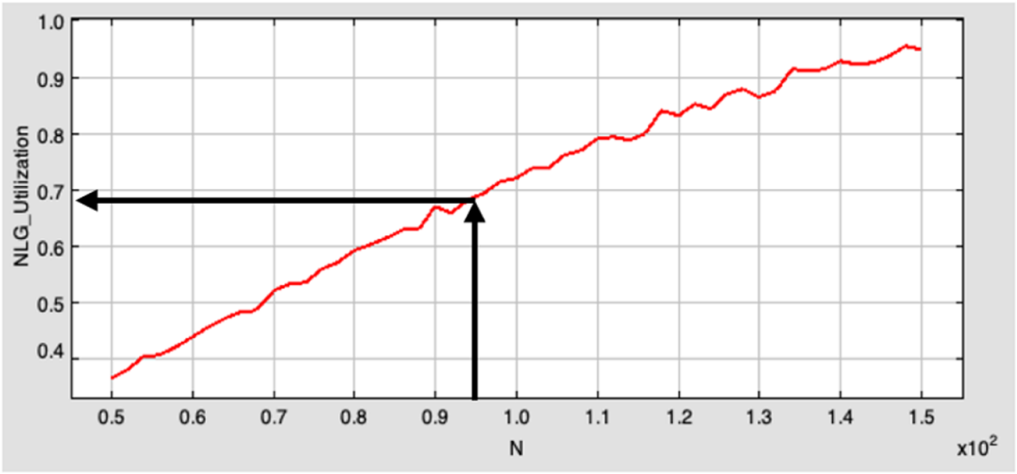

Figure 11. NLG module processor utilization. Approximately 95 customers (vertical black arrow) correspond to 68% utilization (horizontal black arrow). The red line represents the relationship between the PO_Km module processor utilization and the total customers in the system.

From the PoC simulation, only the utilization rates of the PO_Km and NLG queues are observed to need to be monitored, as their relative execution times are orders of magnitude higher than the execution times of the other queues. Therefore, these modules are system performance limiters. Specifically, NLG is the main bottleneck, followed by PO_Km.

Considering the identified performance issues, an NLG module optimization was performed without retraining the original model in the T5 format. The T5 model was converted into an ONNX model [28] and optimized by quantization. The optimization was conducted internally by using the FastT5 library, which saves the optimized model so that it can be loaded again without the need for a new optimization at each execution of the system.

The NLG using FastT5 version 0.1.4. proved to be around three times faster than the non-optimized one. By carrying out tests with the rest of the system, the improvement obtained was two times the ability to process texts, that is, half the time. As a result, the required number of NLG module containers to serve 11.5 msgs/s increased from 11 to 5.

4.3. Never-Ending Language Learning

The never-ending language learning (NELL) process is triggered by new knowledge demand, which could be carried out by a user who wishes to insert new knowledge into the system to cover more topics in the dialog or by a user inquiry that could not be solved with the available knowledge at that time. In the experiments, we, as the human supervisors, started the NELL process when we noticed a user's questions were not answered.

When this process was triggered, one automated bot searched for pages in a specific web portal with information relevant to our banking domain; then, it captured HTML pages or PDF files and extracted the text from them, creating a plain text file. This file was then consumed by the NELL process itself, and the knowledge-producing modules predicted new beliefs based on this document, which were analyzed before being inserted into the ontology. Afterward, this new knowledge could be used during the dialog to answer new questions.

For example, consider the initial ontology shown in Figure 12, which includes knowledge about one event (blue node), its presenters (green nodes), and the companies in which they work. Based on this ontology, the system is capable of conducting the dialog about this event, as shown in Figure 13. The first question the user asks is, "Who presents the event *Event1*?" and the system responds with the related presenters: "The presenters for the event *Event1*?" are: "*Talker1*, *Talker2*, and *Talker3*". Additionally, the system can answer other information about the event, as shown in the next question, "How does it happen?" which is responded by "The event *Event1* is online"; this information is a property in the ontology and is not shown in the previous figure.

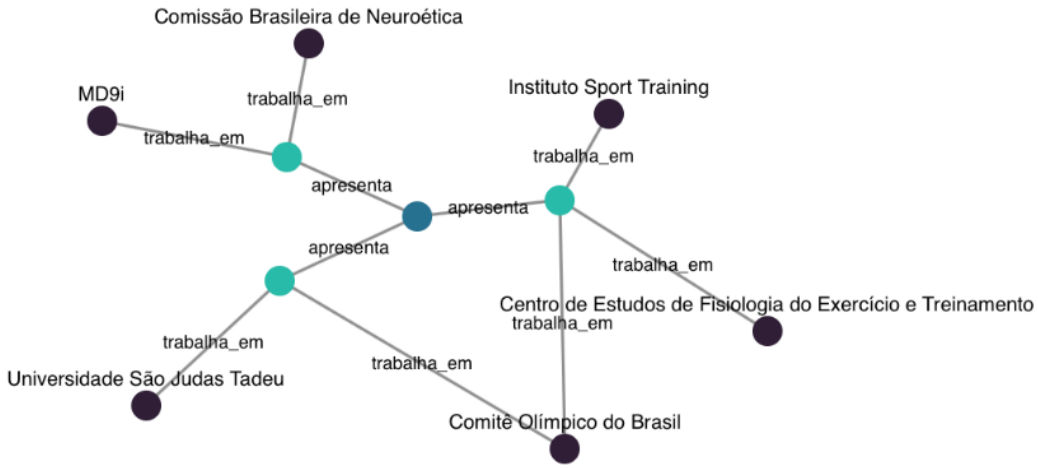

Figure 12. Initial ontology. The relations shown are "trabalha em" (from Portuguese) that means "works for" (in English), and "apresenta" (from Portuguese) that means "presents" (in English). The other terms shown are organization names.

However, it was not possible to answer the last question, 'What events are presented by *Talker5*?" due to unavailable data on the knowledge base; there is no instance of a presenter by this name in our current ontology. Then, when running the NELL process with a new document containing information about this new event, a snippet from this document can be seen in the following:

Event2

ingresso único, Grátis

17 de setembro de 2021, 14 h–15 h 30, Evento online via Netshow.me (. . .) DESCRIÇÃO O evento será composto por três mesas redondas voltadas ao tema da economia circular, uma nova forma de pensar as atividades econômicas associando desenvolvimento e melhoria no uso de recursos naturais, por meio de novos modelos de negócio, da reestruturação da relação de consumo e da otimização de processos de fabricação, com foco na utilização de insumos mais duráveis, recicláveis e renováveis. (. . .) MINI BIO DO(S) PALESTRANTE(S)

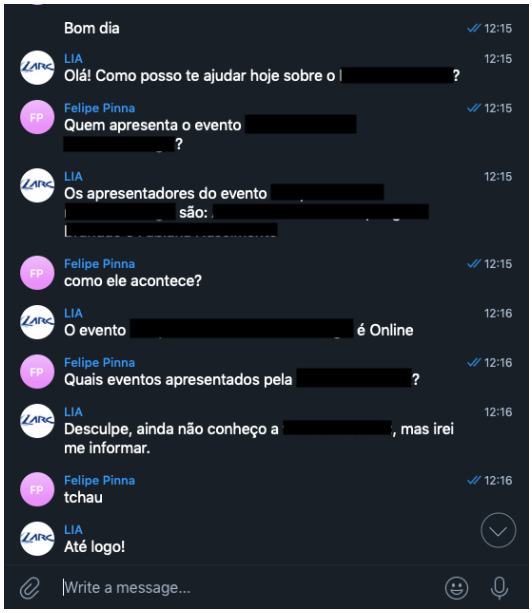

Figure 13. Dialog with the initial ontology. The user is represented by "Felipe Pinna", and the agent is "LIA"; the text below these names is each one's message. In English the dialog can be translated as the following. LIA: Hi! How can I have you today about ...? User: Who presents the event ...? LIA: The presenters of the event ... are ... User: How does it happen? LIA: The event ... is online. User: Which events are presented by ...? LIA: Sorry, I don't know yet about ..., but I will inform myself. User: bye. LIA: See you later.

Talker4 Especialista em Meio Ambiente da FIRJAN (Federação das Indústrias do Estado do Rio de Janeiro) (...)

Talker5: Ativista ambiental, empreendedora e idealizadora do MENOS 1 LIXO, movimento e negócio de impacto por meio de uma plataforma de educação ambiental. Fernanda também é colunista das revistas Glamour e Ela, defensora da ONU Meio Ambiente na campanha Mares Limpos e conselheira do Greenpeace Brasil, tendo sido premiada pela Geração Glamour de 2018 como a mulher mais influente no segmento da sustentabilidade de 2017. (...)

Which can be translated to English as the following:

Event2

single ticket, free

17 September 2021, 2 p.m.–3:30 p.m., Online event via Netshow.me (...) DESCRIPTION The event will consist of three round tables focused on the topic of circular economy, a new way of thinking about economic activities associating development and improvement in the use of natural resources, through new business models, restructuring the consumption relationship and optimization of manufacturing processes, focusing on the use of more durable, recyclable and renewable inputs. (...) MINI BIO OF THE SPEAKER(S)

Talker4 Environmental Specialist at FIRJAN (Federação das Indústrias do Estado do Rio de Janeiro) (...)

Talker5: Environmental activist, entrepreneur and creator of MENOS 1 LIXO, a movement and impact business through an environmental education platform. Fernanda is also a columnist for Glamour and Ela magazines, a defender of the UN Environment in the Mares Limpos campaign and an advisor to Greenpeace Brasil, having been awarded by Geração Glamour in 2018 as the most influential woman in the sustainability segment in 2017. (...)

The knowledge-producing modules processed this text and were able to predict a set of new beliefs, such as the following triples:

- Unary Relationship: isAnEvent("*Event2*");
- Unary Relationship: isATalker("*Talker4*");

- Unary Relationship: isATalker("*Talker5*");
- Binary Relationship: presents("*Talker4*", "*Event2*"), wherein the first attribute is the unary relationship isATalker("*Talker4*"), and the second attribute is the relationship isAnEvent("*Event2*");
- Binary Relationship: presents("*Talker5*", "*Event2*"), wherein the first attribute is the unary relationship isATalker("*Talker5*"), and the second attribute is the relationship isAnEvent("*Event2*").

After those beliefs were integrated into the ontology, it had the structure shown in Figure 14, which shows the nodes from the previous state, the new cloud of nodes from the new event, and the relations among all the instances. Then, when the dialog was resumed, the system was able to respond to the pending question, as shown in Figure 15, where the first message from the system is "Remember when you asked me about *Talker5*, the answer to your question is: "The event present by *Talker5* is *Event2*". Additionally, it was able to keep talking about the extended dialog domain after the user asked "Who is *Talker5*?" the system responded with the definition for her node in the ontology available in the input text for the NELL process.

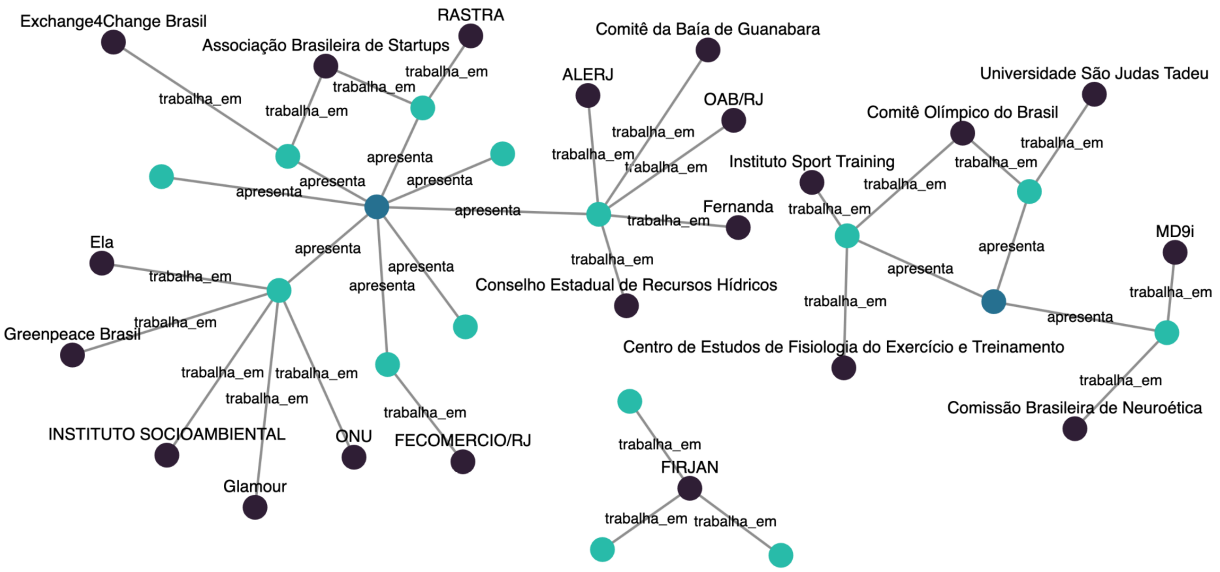

Figure 14. Ontology after one never-ending learning step. The relations shown are "trabalha em" (from Portuguese) that means "works for" (in English), and "apresenta" (from Portuguese) that means "presents" (in English). The other terms shown are organization names.

Our initial ontology was composed of 11 instances, 10 relations, 9 properties, and 3 categories. After processing a few documents, our final ontology had 98 instances, 106 relations, 114 properties, and 5 categories. This was a small-scale experiment to prove the availability of the learning method in a dialog system, and we should aim to process more documents in the future.

Different types of couplings can be explored [25]. The first type is concordance coupling, in which two classifiers have different inputs and produce the same classification. In this case, the two classifiers must agree, and by coupling, we can increase confidence in this knowledge.

In the never-ending learning system developed, there are two instance producers that observe different attributes of the text and produce the same types of instances. Thus, it was possible to add agreement coupling between them. The result of this coupling is shown in Table 4, where an increase in the number of instances of 128% and a growth of 120% in the number of properties is observed. Another relevant fact is that the number of iterations doubled from 4 to 8, showing that the coupling produced the expected effect of increasing the number of rounds of the system, in which new knowledge can be extracted from the same set of data.

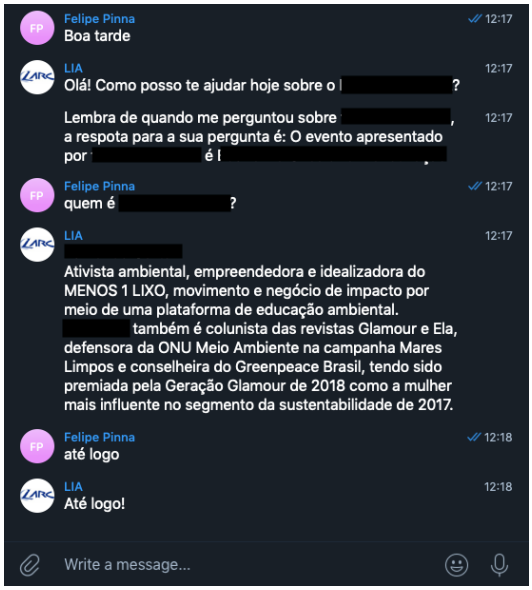

Figure 15. Dialog with the ontology after one never-ending learning step. In English the dialog can be translated as the following. User: Good afternoon. LIA: Hi! How can I help you today about ...? Remember when you asked me about ... the awnser for your question is: the event presented by ... is ... User: Who is ...? LIA: Environmental activist, entrepreneur and creator of MENOS 1 LIXO, a movement and impact business through an environmental education platform. ... is also a columnist for Glamour and Ela magazines, a defender of the UN Environment in the Mares Limpos campaign and an advisor to Greenpeace Brasil, having been awarded by Geração Glamour in 2018 as the most influential woman in the sustainability segment in 2017. User: see you soon. LIA: See you soon.

Table 4. Final ontology metrics after adding agreement and output couplings.

	Before	Agreement Coupling	Variation	Agreement + Output Couplings	Variation
Categories	14	14	0%	14	0%
Instances	114	260	128%	298	161%
Relationships	52	52	0%	52	0%
Properties	122	269	120%	320	162%
Iterations	4	8	100%	8	100%

Another type of coupling is output coupling, which can be divided into subtype coupling and mutual exclusion coupling. In these cases, two different classifiers receive the same input. In subtype coupling, the classes involved are a subtype of the other, for example, "person" and "actor", which the classifiers must agree on. Since "actor" is a subtype of "person", when the two classifiers agree, we can increase confidence in this knowledge. Mutual exclusion coupling is when the classes involved are mutually exclusive, for example, "person" and "company"; the classifiers must, thus, disagree to increase confidence in the knowledge.

Output coupling was added between the system instance producers, which produced a further increase in the number of instances and properties while maintaining the number of process iterations. The result was a total increase of 161% of instances, 162% of properties, and 100% of iterations when compared to the same endless-learning process performed without these couplings.

5. Conclusions

This work presents a framework for building a specific domain dialog system with a knowledge base. We propose a common architecture for these systems, aiming at re-using mod-

ule implementations when creating new instances for new similar domains, with this supported by ontologies and semi-automatic development methods for facilitating the maintenance and recreation of the project. When compared to the manual process, the automated process using RPA achieved a relative reduction of 76%, from 37 to 9 h, to build an ontology.

Specific knowledge was modeled in the task, domain, and support ontologies for the banking domain. Semi-automatic corpus generation and automatic data acquisition using robotic process automation (RPA) were used to lessen the required effort to build this specific domain dialog system. The never-ending learning proof of concept was described for the acquisition of new financial event information collection.

As the main contribution of this work, the proposed scalable architecture for specific domain dialog systems was proposed and used for the development of a conversational agent for the banking domain. Researchers and practitioners are expected to use the proposed architecture and specialize it for their specific domains by adjusting the training datasets and ontologies but re-using the organization of the proposed architecture.

The proposed never-ending learning approach allows the system to continuously learn over time by searching for knowledge when a question is not answered in a conversation. Then, after the knowledge base is updated, the system can remind the user about the pending question and respond, giving the perception of evolution over time. Further similar questions could be answered immediately since the required knowledge would be available in the dialog system ontology.

A dialog system for the banking domain in the Portuguese language was created using the proposed framework as a demonstration of its application. A fully functional implementation for mobile devices shows that it is possible to embed the dialog system in a simpler device without sharing user data with third parties.

For dialog-oriented systems or very simple task-oriented systems, the use of BotBuilder tools provides great speed in creating new chatbots with low data and development needs. This framework can be applied to most task-oriented systems that use a dialog system as a solution.

Future work may expand the never-ending learning principles by further exploring the coupled learning between tasks to increase the potential for discovering new knowledge in past documents and explore other aspects of the original NELL experiment, such as self-reflection. These opportunities for future research might lead to a more autonomous conversational system and a better use of available data from the documents utilized. Another consideration for future work is to use OpenAI GPT-3 (https://chat.openai.com/ (accessed on 21 December 2023)) in the named entity recognition and relationship identification tasks in the proposed dialog system.

Another possibility is to evaluate the user experience of the conversational assistants built with different tools by using specific study designs to compare various design choices [29,30]. Relevant work presents a framework for the qualitative analysis of chatbot dialogues in the customer service domain that was used to investigate key drivers of user experience, such as response relevance and understanding user interaction patterns [31].

The scalability evaluation executed in this work focuses only on throughput. Future research efforts must consider other relevant nonfunctional aspects, such as availability, security, maintainability, and relevant trade-offs, possibly using the ATAM software engineering method [32]. As security is relevant for dialog systems [33], it could be interesting to consider a federated learning approach [34,35].

Another relevant research avenue is to deploy the proposed method with other domains and languages. For this objective, one interesting opportunity is to use Wav2Vec version 2.0. Wav2Vec are speech recognition models for which the principle is self-supervised learning, which uses a large set of unlabeled data to learn good representations of the data; then, it learns the task of transcribing audio by consuming a small number of labeled data. This solution has multilingual models in which training is carried out with audio from different languages, seeking to explore elements of transcription tasks that are universal. These models have lower results than monolingual models for languages with many resources available but can be useful for languages with fewer resources available for model training [36].

Author Contributions: Conceptualization, J.C.N., M.C.D. and W.V.R.; methodology, J.C.N., M.C.D. and W.V.R.; software, F.C.d.A.P., V.T.H., J.C.N. and R.d.F.P.M.; validation, M.C.D., R.S.O. and W.V.R.; investigation, F.C.d.A.P., V.T.H., J.C.N. and R.d.F.P.M.; resources, F.C.d.A.P., V.T.H., J.C.N., R.d.F.P.M., M.C.D. and R.S.O.; data curation, F.C.d.A.P., V.T.H., J.C.N., R.d.F.P.M., M.C.D. and R.S.O.; writing—original draft preparation, F.C.d.A.P. and V.T.H.; writing—review and editing, F.C.d.A.P., V.T.H., J.C.N. and M.C.D.; supervision, J.C.N., M.C.D. and W.V.R.; project administration, J.C.N. and W.V.R.; funding acquisition, J.C.N. and W.V.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Graduate Program in Electrical Engineering (PPGEE) from the Polytechnic School of the Universidade de São Paulo.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors thank the support provided by the Universidade de São Paulo Support Foundation (FUSP) and the Laboratory of Computer Architecture and Networks (LARC) at the Computing and Digital Systems Engineering Department (PCS), Polytechnic School, Universidade de São Paulo, Brazil.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

References

1. Yu, C.; Zhang, C.; Hu, Z.; Zhan, Z. Gate-Enhanced Multi-domain Dialog State Tracking for Task-Oriented Dialogue Systems. In *Computational Intelligence for Engineering and Management Applications: Select Proceedings of CIEMA 2022*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 575–594.
2. Khan, M.A.; Huang, Y.; Feng, J.; Prasad, B.K.; Ali, Z.; Ullah, I.; Kefalas, P. A Multi-Attention Approach Using BERT and Stacked Bidirectional LSTM for Improved Dialogue State Tracking. *Appl. Sci.* **2023**, *13*, 1775. [CrossRef]
3. Ultes, S.; Rojas Barahona, L.M.; Su, P.H.; Vandyke, D.; Kim, D.; Casanueva, I.N.; Budzianowski, P.; Mrkšić, N.; Wen, T.H.; Gasic, M.; et al. PyDial: A Multi-domain Statistical Dialogue System Toolkit. In Proceedings of the ACL 2017, System Demonstrations, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 73–78.
4. Bocklisch, T.; Faulkner, J.; Pawlowski, N.; Nichol, A. Rasa: Open source language understanding and dialogue management. *arXiv* **2017**, arXiv:1712.05181.
5. Coucke, A.; Saade, A.; Ball, A.; Bluche, T.; Caulier, A.; Leroy, D.; Doumouro, C.; Gisselbrecht, T.; Caltagirone, F.; Lavril, T.; et al. Snips Voice Platform: An embedded Spoken Language Understanding system for private-by-design voice interfaces. *arXiv* **2018**, arXiv:1805.10190.
6. Bennett, M. The financial industry business ontology: Best practice for big data. *J. Bank. Regul.* **2013**, *14*, 255–268. [CrossRef]
7. de Melo, G.; Imaizumi, V.; Cozman, F. Winograd schemas in portuguese. In Proceedings of the Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional, Salvador, Brazil, 15–18 October 2019; SBC: Porto Alegre, Brazil, 2019; pp. 787–798.
8. Dziri, N.; Rashkin, H.; Linzen, T.; Reitter, D. Evaluating Attribution in Dialogue Systems: The BEGIN Benchmark. *Trans. Assoc. Comput. Linguist.* **2022**, *10*, 1066–1083. [CrossRef]
9. Dziri, N.; Kamalloo, E.; Milton, S.; Zaiane, O.; Yu, M.; Ponti, E.M.; Reddy, S. Faithdial: A faithful benchmark for information-seeking dialogue. *Trans. Assoc. Comput. Linguist.* **2022**, *10*, 1473–1490. [CrossRef]
10. Dai, Y.; He, W.; Li, B.; Wu, Y.; Cao, Z.; An, Z.; Sun, J.; Li, Y. CGoDial: A Large-Scale Benchmark for Chinese Goal-oriented Dialog Evaluation. *arXiv* **2022**, arXiv:2211.11617.
11. Chevalier-Boisvert, M.; Bahdanau, D.; Lahlou, S.; Willems, L.; Saharia, C.; Nguyen, T.H.; Bengio, Y. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv* **2018**, arXiv:1810.08272.
12. Harrison, B.; Ehsan, U.; Riedl, M.O. Guiding Reinforcement Learning Exploration Using Natural Language. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 1956–1958.
13. Bayer, M.; Kaufhold, M.A.; Reuter, C. A survey on data augmentation for text classification. *ACM Comput. Surv.* **2022**, *55*, 1–39. [CrossRef]
14. Rohaime, N.A.; Razak, N.I.A.; Thamrin, N.M.; Shyan, C.W. Integrated Invoicing Solution: A Robotic Process Automation with AI and OCR Approach. In Proceedings of the 2022 IEEE 20th Student Conference on Research and Development (SCOReD), Bangi, Malaysia, 8–9 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 30–33.
15. Carvalho, M.; Hayashi, V.; Pinna, F.; Marquesone, R.; Néto, J.; Ruggiero, W. Towards Modeling Semi-automatic Ontology based on Natural Language Processing. In Proceedings of the 25th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2021, Virtual, 18–21 July 2021; pp. 85–90.

16. Schmitt, X.; Kubler, S.; Robert, J.; Papadakis, M.; LeTraon, Y. A replicable comparison study of NER software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate. In Proceedings of the 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), Granada, Spain, 22–25 October 2019; IEEE: Piscataway, NJ, USA, 2019, pp. 338–343.
17. Lamy, J.B. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artif. Intell. Med.* **2017**, *80*, 11–28. [CrossRef] [PubMed]
18. Dias-da Silva, B.C. Wordnet. br: An exercise of human language technology research. In Proceedings of the Gwc 2006: Third International Wordnet Conference, Jeju Island, Republic of Korea, 22–26 January 2005; Masaryk University: Brno, Czech Republic, 2005; pp. 301–303.
19. Wang, Y.; Shen, Y.; Jin, H. A Bi-Model Based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), New Orleans, LO, USA, 1–6 June 2018; pp. 309–314.
20. Zhu, Y.; Wan, J.; Zhou, Z.; Chen, L.; Qiu, L.; Zhang, W.; Jiang, X.; Yu, Y. Triple-to-Text: Converting RDF Triples into High-Quality Natural Languages via Optimizing an Inverse KL Divergence. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, 21–25 July 2019; Piwowarski, B., Chevalier, M., Gaussier, É., Maarek, Y., Nie, J., Scholer, F., Eds.; ACM: New York, NY, USA, 2019; pp. 455–464. [CrossRef]
21. Mitchell, T.; Cohen, W.; Hruschka, E.; Talukdar, P.; Yang, B.; Betteridge, J.; Carlson, A.; Dalvi, B.; Gardner, M.; Kisiel, B.; et al. Never-ending learning. *Commun. ACM* **2018**, *61*, 103–115. [CrossRef]
22. Qi, G.J.; Luo, J. Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 2168–2187. [CrossRef] [PubMed]
23. Han, X.; Gao, T.; Yao, Y.; Ye, D.; Liu, Z.; Sun, M. OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction. In Proceedings of the EMNLP-IJCNLP: System Demonstrations, Hong Kong, China, 3–7 November 2019; pp. 169–174. [CrossRef]
24. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
25. Hruschka, E.R., Jr.; Duarte, M.C.; Nicoletti, M.C. Coupling as Strategy for Reducing Concept-Drift in Never-ending Learning Environments. *Fundam. Informaticae* **2013**, *124*, 47–61. [CrossRef]
26. Serazzri, G.; Casale, G.; Bertoli, M. Java modelling tools: An open source suite for queueing network modelling andworkload analysis. In Proceedings of the Third International Conference on the Quantitative Evaluation of Systems-(QEST'06), Los Angeles, CA, USA, 11–14 September 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 119–120.
27. Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*; Wiley: New York, NY, USA, 1991; Volume 1.
28. Kim, S.Y.; Lee, J.; Kim, C.H.; Lee, W.J.; Kim, S.W. Extending the ONNX Runtime Framework for the Processing-in-Memory Execution. In Proceedings of the 2022 International Conference on Electronics, Information, and Communication (ICEIC), Granada, Spain, 22–25 October 2019; IEEE: Piscataway, NJ, USA, 2022; pp. 1–4.
29. Følstad, A.; Skjuve, M. Chatbots for customer service: User experience and motivation. In Proceedings of the 1st International Conference on Conversational User Interfaces, Dublin, Ireland, 22–23 August 2019; pp. 1–9.
30. Haugeland, I.K.F.; Følstad, A.; Taylor, C.; Bjørkli, C.A. Understanding the user experience of customer service chatbots: An experimental study of chatbot interaction design. *Int. J. -Hum.-Comput. Stud.* **2022**, *161*, 102788. [CrossRef]
31. Følstad, A.; Taylor, C. Investigating the user experience of customer service chatbot interaction: A framework for qualitative analysis of chatbot dialogues. *Qual. User Exp.* **2021**, *6*, 6. [CrossRef]
32. Bass, L.; Clements, P.; Kazman, R. *Software Architecture in Practice*; Addison-Wesley Professional: Boston, MA, USA, 2003.
33. Tsinganos, N.; Fouliras, P.; Mavridis, I. Leveraging Dialogue State Tracking for Zero-Shot Chat-Based Social Engineering Attack Recognition. *Appl. Sci.* **2023**, *13*, 5110. [CrossRef]
34. Zafar, M.H.; Bukhari, S.M.S.; Abou Houran, M.; Moosavi, S.K.R.; Mansoor, M.; Al-Tawalbeh, N.; Sanfilippo, F. Step towards secure and reliable smart grids in Industry 5.0: A federated learning assisted hybrid deep learning model for electricity theft detection using smart meters. *Energy Rep.* **2023**, *10*, 3001–3019. [CrossRef]
35. Bukhari, S.M.S.; Zafar, M.H.; Abou Houran, M.; Moosavi, S.K.R.; Mansoor, M.; Muaaz, M.; Sanfilippo, F. Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with SCNN-Bi-LSTM for enhanced reliability. *Ad Hoc Netw.* **2024**, *155*, 103407. [CrossRef]
36. Baevski, A.; Zhou, Y.; Mohamed, A.; Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 12449–12460.