future internet

MDPI

Article

Technology Enhanced Learning Using Humanoid Robots

Diego Reforgiato Recupero

Department of Mathematics and Computer Science, University of Cagliari, 09124 Cagliari, Italy; diego.reforgiato@unica.it

Abstract: In this paper we present a mixture of technologies tailored for e-learning related to the Deep Learning, Sentiment Analysis, and Semantic Web domains, which we have employed to show four different use cases that we have validated in the field of Human-Robot Interaction. The approach has been designed using Zora, a humanoid robot that can be easily extended with new software behaviors. The goal is to make the robot able to engage users through natural language for different tasks. Using our software the robot can (i) talk to the user and understand their sentiments through a dedicated Semantic Sentiment Analysis engine; (ii) answer to open-dialog natural language utterances by means of a Generative Conversational Agent; (iii) perform action commands leveraging a defined Robot Action ontology and open-dialog natural language utterances; and (iv) detect which objects the user is handing by using convolutional neural networks trained on a huge collection of annotated objects. Each module can be extended with more data and information and the overall architectural design is general, flexible, and scalable and can be expanded with other components, thus enriching the interaction with the human. Different applications within the e-learning domains are foreseen: The robot can either be a trainer and autonomously perform physical actions (e.g., in rehabilitation centers) or it can interact with the users (performing simple tests or even identifying emotions) according to the program developed by the teachers.

Keywords: human-robot interaction; voice assistant technology; virtual reality; robotic operating system

check for updates

Citation: Reforgiato Recupero, D. Technology Enhanced Learning Using Humanoid Robots. *Future Internet* **2021**, *13*, 32. https://doi.org/10.3390/fi13020032

Academic Editor: Laurent Moccozet
Received: 31 December 2020
Accepted: 25 January 2021
Published: 27 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The latest findings within the domain of Artificial Intelligence (AI) and Robotics have paved the way for the design of an increasing number of advanced robot-oriented applications. This has further led to the hypothesis that robotics and Artificial Intelligence, in just over 100 years, will likely beat humans in performing any kind of job [1]. Looking back in history, the first autonomous robots independent from human operators were adopted for hazardous tasks, (e.g., exploration of deep oceans, volcanoes, and the surface of the Moon and Mars). It was well understood that the next generation of robots should have fit into everyday human life, a much harder task if we consider all the social development and intricacies of human beings. Thus, since the early 1990s, AI and robotics researchers have been developing robots that explicitly engaged users on a social level.

The challenge to designing an autonomous social robot is huge because the robot should correctly assess the actions of the people and respond accordingly. Furthermore, to increase the difficulties of this challenge, based on science fiction representations of advanced social robots, a person that interacts with it might hold a very high expectancy of its capabilities.

As a workaround, to simulate advanced behaviors and capabilities, some social robots were partially or fully remote controlled. Clearly, this was not a solution that could be applied on a large scale and an autonomous social robot should not require manual intervention to fully operate [2]. Instead, the design of an autonomous social robot that can independently interact with people in homes, schools, hospitals, and workplaces is a serious scientific challenge.

Besides, the creation of robots able to understand humans is only half the equation. Developers should also identify the features humans will need to trust and engage with robots. For example, during conversations people apply social-psychological constructs such as beliefs, emotions, and motivations to try to understand what another person does and why. The same would happen for human-robot interactions, therefore designing an autonomous social robot is not a matter of just understanding well the natural language text expressed by humans but also identifying hidden psychological elements, emotions, sarcasm and irony, jokes, and so on which might come in different ways (from words, sounds, facial and gesture expressions, and combinations of them).

Robot adoption has shifted to common places where daily life is carried out. Japan, for example, uses robots in restaurant kitchens to make sushi and chop vegetables. Some robots work as receptionists, cleaners, drink servers, and others are tailored for making coffee, starting with the beans, and serving drinks. Within the assisted living domain, robots have already proven to bring benefits. For example, some can help the elderly get out of bed and provide a sense of companionship against loneliness. Robots can also be programmed to interface with smart hospital elevators to reach any floor and distribute medications to patients. Or, they can be programmed to perform complex surgeries. Around the home, robots can help in keeping a house clean or cutting the grass and cleaning the pool. Within the education domain, robots have been employed as teacher's assistants, helping kids to perform easy tasks such as singing. Given their increasing and large adoption, a new discipline known as Educational Robotics has been designed to introduce students to robotics and programming interactively from a very early stage. Educational robotics provides a microcosm of technologies that support STEM (Science, Technology, Engineering, and Mathematics) education. Besides the fun, robots apply a variety of learning tools that engage the students.

Therefore, following these directions and exploiting some of the cutting-edge technologies that have appeared in the literature of various domains, in this paper we present four different modules we have developed on top of a humanoid robotic platform and that augment the robot with the following skills:

- The robot is able to perform sentiment analysis by understanding whether the statement expressed by a human is positive, neutral, or negative;
- The robot can generate answers to open-dialog natural language utterances of the user;
- The robot can perform an action command (walk, move, change posture, etc.) depending on open-dialogue natural language utterances that the user expresses;
- The robot can identify objects through its camera according to a two-level defined taxonomy.

Each skill can be used within the learning domain and represents a research problem that has been addressed, analyzed, and solved through a performance evaluation carried out on different datasets and domains. It is also straightforward to plugin further modules as the overall architectural design is general, flexible, and scalable. Therefore, the main scientific contributions of this paper have been summarized in the following items:

- We provide a general, flexible, and scalable architectural design for human-robot interaction where different modules tailored for the learning domain can be developed, embedded and whose priority of execution can be easily configured;
- We develop four modules that correspond to four use cases to Zora, a humanoid robotic platform which extends NAO. The reader may notice that everything we have developed for Zora, server-side frameworks, source codes, and use cases, is compatible with NAO, the humanoid and programmable robot from SoftBank, too;
- An evaluation has been carried out by six stakeholders on five different scenarios using a Software Architecture Analysis method and stakeholders provided useful feedback that we employed to further improve the architecture;
- Each module along with its related source code has been stored in a public repository that can be freely downloaded and each used case comes with a video example, showing the human-robot interaction.

The remainder of our paper is organized as follows. In Section 2 we report some related works within other software architectures employing NAO and the differences with the one we propose. Zora, the humanoid robot we have leveraged and extended for the human-robot interaction use cases, is depicted in Section 3. In Section 4 the architectural design of our system is proposed and described along with the four modules it contains: The Semantic Sentiment Analysis module (Section 4.3), the Generative Conversation Agent (Section 4.4), the Robot Action Ontology (Section 4.5), and the Object Detection module (Section 4.6). Moreover, this section includes the flowchart of the robot;s behavior with respect to the four mentioned modules and the priority of execution of any developed module with respect to the others.

In Section 5 we will show the evaluation analysis of the proposed architecture we have carried out and the results of the user experience. Finally, Section 6 ends the paper with conclusions and future works.

2. Related Work

In this section, we include other software architectures on top of NAO present in the literature and illustrate the differences with ours.

Authors in [3] have presented a software architecture used to drive several robotic platforms including a simulated agent and a real robot NAO. The main goal for which the architecture was designed was to develop the team behavior of a robot soccer team. The architecture consists of several parts: Platform interface, modular framework, communication, debug, tools, and tests. Some components are independent whereas others are platform-specific. Moreover, all the components have a simple user interface. Our proposed architecture allows using the same module for multiple NAOs at the same time moreover, it has been thought for a general domain and any module of any kind can be plugged in. Within the same domain (soccer cup), other authors have proposed in [4] an open-source accessory for NAO with the form of a backpack including an ODROID XU4 board to process algorithms externally with ROS (the robotic operating system https://www.ros.org/ (accessed on 25 January 2021)) compatibility. The developed software architecture is focused on the communication between the B-Human's framework [5] and ROS to have access to the robot's sensors close to real-time. The main difference with the one we propose in this paper is that it is completely based on ROS and the B-Human's libraries and focused on the soccer cup whereas our proposed architecture is not based on ROS and works in the general domain. Other authors have employed voice assistant technologies such as Google Assistant to provide NAO an architecture with further capabilities that let the robot exploit their strength and software development kit [6].

3. The Used Robotic Platform

Zora extends NAO, the first humanoid robot designed by Aldebaran Robotics, which was renovated by the SoftBank Group. The extension regards a software middleware layer that includes a control panel and high-level composer which allows non-expert users to program certain actions and behaviors for Zora to execute. They include physical and verbal actions (Zora speaks eight different languages) that the robot can perform.

Zora inherits from NAO the possibility to be programmed using Choregraphe (http://doc.aldebaran.com/1-14/software/choregraphe/index.html (accessed on 25 January 2021)), which lets developers:

- Design and integrate several robot actions and complex behaviors utilizing an integrated development environment making use of the Python programming language and external Internet libraries;
- Develop robot movements by leveraging a built-in and effective graphical user interface;
- Experiment the developed actions on the simulated robot;
- Experiment the developed actions on the real robot connected on the same local area network (LAN) as the developer's machine.

NAO is provided with a minimal Linux-based OS, known as NAOqi, which controls the robot's mics, speakers, and two high-definition cameras. NAO is provided with 4 mics in the head (2 in the front and 2 in the back) to store in its memory any sound (including audio from humans). Human natural language is analyzed and transformed into text using speech-to-text technologies of Nuance (https://www.nuance.com (accessed on 25 January 2021)). To increase the accuracy of the speech to text process, we are massively making use of cloud technologies. This enables faster pre-processing of the sound stored by the robot and removes background noise, which may compromise the analogical conversion into digital information. To capture images, to give wider operating space to the developer, and for further interactions with humans, Zora has two high-definition cameras providing a 640 × 480 resolution at 30 fps, contact and tactile sensors, sonars, an inertial unit, force-sensitive resistors, eye LEDs, infra-red, 2 loud speakers in the ears, and two network ports (wifi and Ethernet). More details can be found at http://doc.aldebaran.com/2-1/family/robots/index_robots.html (accessed on 25 January 2021).

4. The Proposed Architectural Design

Figure 1 shows the high-level architectural design we propose in this paper. The developer's pc is connected to the same LAN as the robot. They develop a module consisting of both the Choregraphe Component (CC) and the Server-Side-Support Component (SSSC). The former is uploaded into the robot whereas the latter into a cloud system. The CC is informed of the IP address of the SSSC. In the figure, there are four different modules loaded into the robot, each consisting of a CC and a SSSC. The four SSSCs reside in the cloud. To load a given module to a new robot, it would be enough to plug the robot in the same LAN, and just load the CC to the new robot as the SSSC can be shared with that of other robots. In the figure there is a second robot with the Object Detection CC loaded and enabled and whose related Object Detection SSSC is shared with that of the first robot. In such an example, two users may interact at the same time with the two robots. Sharing the same LAN for multiple robots might flag potential traffic congestion problems. However, the messages sent to/from the robot through the Choregraphe suite are optimized for network streaming. It is the responsibility of each module to make sure that the network protocol used for sending/receiving messages is efficient and optimized.

The robot uses its WiFi port to connect through a hot-spot to the Internet and to a local area network. If the developer needs to upload a new module, they connect to the same hot-spot in order to belong to the same local area network of the robot and can easily handle the uploads with the Choregraphe suite running in the pc. The Choregraphe Component (CC) and the Server-Side-Support Component (SSSC) are illustrated in the following. Next, each module shown in Figure 1 will be detailed.

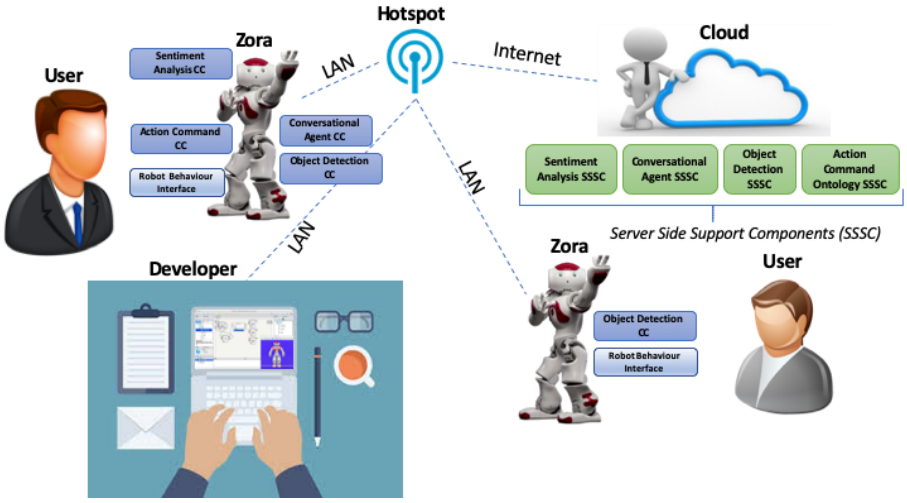

Figure 1. Proposed high-level architectural design.

4.1. Choregraphe Component (CC)

The first one, the Choregraphe Component (CC), is mandatory in Python programming language and is developed with the Choregraphe suite, which directly controls the robot behaviors and actions, and is responsible for any interaction between the robot and human. Any sensor of the robot that is employed (e.g., microphone, camera, tactile sensors) is controlled using API functions from this software component. Furthermore, this software component is not demanding as far as the computational resources are concerned (CPU, disk storage, and RAM) and is uploaded to the robot from the Choregraphe environment. As soon as the component is loaded into the robot it is enabled. For testing purposes, Choregraphe allows using a virtual robot, embedded in the same Choregraphe suite, that simulates the real robot along with all its capabilities. In the rest of the paper, we will use the term 'enable' to indicate a module that has been loaded into the robot and is ready to be 'activated' depending on the user's input text.

4.2. Server-Side-Support Component (SSSC)

The second part a new module might consist of is the server-side program, developed with any back-end technology, containing the support software (e.g., heavy computation, or storage capabilities), and exposing the REST APIs. The back-end is hosted either in the cloud or in a dedicated server. The Python program of the CC needs to know the IP address of the cloud system or the server where the back-end of the program resides to be able to call its REST APIs and exploit its capabilities. The reader notices that this component is not essential as a new module may consist of just the CC when there are not heavy computations to be executed. As an example, a simple Bingo game may be developed as Python code as a CC only and uploaded into the robot. In such a case, there would not be an external program in charge of any specific computation.

To note, each newly created module can run independently from each other and it is possible to include as many modules as desired. Moreover, when different CCs of multiple modules are loaded into the robot, a priority for each module must be assigned so that when the user speaks to the robot, one module at a time may be activated. The four modules presented in this paper and that are shown in Figure 1 have the following priority importance: Semantic Sentiment Analysis Module, Object Detection, Action Command Ontology Module, and Conversational Agent Module. This means that when the robot gets some text from the user, it checks in that order if one of those modules can be triggered. Section 4.7 will illustrate how the modules' priorities are set. Only one module can be triggered for the underlying text.

This scheme is scalable and flexible as:

- The Python programs directly loaded into the robot from Choregraphe are usually very light, and we can upload a very high number of them before the physical space of the robot ends;
- There are no limits to the number of SSSCs we can host/call into/from the cloud or a given server;
- The same SSSC can be uploaded in several robots and in such a case the robots may either lie within the same local area network or on a different network while still being able to share the same program running on a server or cloud system. As an example, in Figure 1 there is another robot connected to the same LAN, with the Object Detection CC loaded and whose Object Detection SSSC is shared with the existing one.

The robotic platform is only responsible for interaction with humans. Such an interaction evolves according to the underlying module that is being triggered and by what each SSSC returns depending on the user's action. Basically, the robot itself is a mere interface that interacts with humans through its sensors. The cognitive skills, learning skills, and any other smart capabilities are provided by the SSSC where NLP, Semantic Web, Computer Vision, AI, and other innovative research applications may be executed providing a more and more effective human-computer interaction.

Noticeably, the robot does not perform any action if there are no loaded modules. As soon as one module is loaded, the robot starts performing some action accordingly. When the robot is shut down and boots again, the loaded modules are still available in the robot. The programmer has to explicitly disable them (at least the related CC using the Choregraphe framework) to unload them from the robot.

4.3. Semantic Sentiment Analysis Module

In this section, we will describe the two components (CC and SSSC) that constitute this module.

Choregraphe Component (CC). The CC of the Semantic Sentiment Analysis module is responsible for interactions with humans through the sensors of the robot. It allows the robot to wait for natural language expressions spoken by the user which are converted into text by using speech-to-text technologies mentioned in Section 3. The Semantic Sentiment Analysis module is triggered when the user says *"Play Sentiment Analysis"*. In such a case the robot is ready to perform sentiment analysis and waits for a further text from the user. From now on, the input text spoken by the user is first processed through speech-to-text technologies and then forwarded to the SSSC for the Sentiment Analysis classification task. To note that at this point the Semantic Sentiment Analysis Module takes over any other loaded module and any further sentence expressed by the user is processed for Sentiment Analysis only. The SSSC returns as output to the CC one among the positive, negative, or neutral class. The robot performs a different action according to the returned output class and then goes back in listening mode waiting for further input from the user to be processed for the Sentiment Analysis. To exit from the current behavior and to allow future user expressions activating other enabled modules as well, the user needs to simply say exit.

Server-Side-Support Component (SSSC). The SSSC includes the Semantic Sentiment Analysis engine that has been developed following directions and insights of our previous works in [7–12]. The engine has been significantly extended, augmented, tuned, and improved to achieve better performance in a general domain. The task we have addressed was the simple polarity detection: Given an input sentence, return whether the sentence expresses a positive, negative, or neutral sentiment. We have employed a Deep Learning approach for such a classification task using Recurrent Neural Networks (RNNs).

We adopt a single model that uses LSTM units only and, at the same time, it keeps the ability of a CNN to learn the most effective features. In particular, we played with different word embeddings and our resulting approach is able to improve the pre-trained word embeddings at training time, by applying a new set of embeddings.

The architecture of the proposed semantic sentiment analysis engine is based on a 2-layer Bidirectional LSTM with attention, as shown in [8].

As training and validation sets, we have considered the entire dataset released by the Semantic Sentiment Analysis challenge held at ESWC 2018 [13]: 90% of it has been used as the training set and 10% as the validation set. The evaluation has been carried out using nine different word embeddings provided by the organizers of the challenge and those pre-trained on Google News [14]. We trained each model twice: Once with the fine tuning and once without it. The resulting 20 networks have been ensembled using three different strategies: Majority voting, weighted average, and prudential multiple consensus.

In Table 1 we show the results of the single models before and after performing the polarization of the embeddings.

The ensemble strategies described above allow further improvements in the performance of the system. Table 2 shows the F_1 score obtained by the different ensemble strategies on the final test set released by the challenge's organizers. The best results are obtained with the Majority Voting and Weighted Average strategy whose F_1 is 0.967, beating all the competing systems participating in the Semantic Sentiment Analysis Challenge previously mentioned.

Table 1. Results obtained before and after embedding polarization.

	Size	Epochs	Pre-Trained	Fine-Tuning
ESWC 2018 Embeddings	128	15	0.949	0.961
	128	30	0.949	0.963
	128	50	0.949	0.962
	256	15	0.951	0.963
	256	30	0.951	0.965
	256	50	0.951	0.965
	512	15	0.951	0.965
	512	30	0.955	0.966
	512	50	0.957	0.966
Google News Embeddings			0.936	0.963

Table 2. Results of the ensemble strategies.

Ensemble Strategy	F_1 Score
Majority Voting	0.967
Weighted Average	0.967
Balanced PMC	0.966

The system has been developed in JAVA and employs the Stanford CoreNLP [15] for standard preprocessing tasks, such as sentence detection, tokenization, POS tagging, and stop-word removal. We have also leveraged Deeplearning4j (https://deeplearning4j.org (accessed on 25 January 2021)) to deal with feature extraction and for the classification task. The Semantic Sentiment Analysis engine has been embedded in a server-side application, hosted in the cloud, which exposes REST APIs that take an input sentence and return to the CC one of the three output classes. Last but not least, the engine uses Deeplearning4j with cuDNN (https://developer.nvidia.com/cudnn (accessed on 25 January 2021)) in order to leverage NVidia GPUs technology to speed up the creation of the neural network, its training, and the prediction phases.

4.4. Generative Conversational Agent Module

In the following, we will describe the two components (CC and SSSC) this module consists of.

Choregraphe Component (CC). The Generative Conversational Agent Module is responsible for the open-domain dialog generation. With such a skill, the robot is therefore able to reply to the user with meaningful and coherent responses taking into account the user's dialog history and the current context. Once the CC is loaded into the robot from the Choregraphe suite, it is enabled and triggered as soon as the text spoken by the user does not activate any other module loaded into the robot.

Server-Side-Support Component (SSSC). Our system has been built on top of [16,17] and is based on the sequence to sequence modeling. The rationale for that is to have the approach able to understand a conversation extracting and learning semantics and syntax out of it. The other reason is to have the approach language-independent. We make use of GloVe word embeddings (https://nlp.stanford.edu/projects/glove/ (accessed on 25 January 2021)) for the embedding layer it uses. This layer takes words with the same distribution as input and output. Moreover, it is used by both the encoding and decoding processes. It disposes of two LSTM which analyze the dialog context and the unfinished answer being created. The output vectors of the two LSTM are integrated and delivered to the dense layers whose goal is to forecast the current token of the answer. We used the Cornell Movie Dialogs Corpus [18] as a training set. Like the Semantic Sentiment Analysis SSSC discussed within the previous section, the Generative Conversational Agent SSSC has been developed in Java using Deeplearning4j with cuDNN and, as such, it exploits GPUs

technology. It is hosted in the cloud with GPUs availability. A video that illustrates how the robot works with an old version of the Semantic Sentiment Analysis Module and the Generative Conversational Agent Module is publicly available (https://bit.ly/2GIxPma (accessed on 25 January 2021)).

4.5. Robot Action Commands Module

In this section, we will show the two components the module consists of. In particular, we will discuss the CC, employed for the interaction with the user, and the engine of the SSSC that we have implemented [19] to identify natural language sentences expressing robot action commands.

Choregraphe Component (CC). Once the module has been loaded into the robot, it is active and makes the robot wait for an input text from the user. As soon as the CC has an input text to process, it sends to the SSSC the text to be processed and waits for the output. The output from the SSSC might be an action that the robot performs or a question for the user asking to specify further the action that was meant. This occurs when the action command of the user misses some important entities (e.g., *Zora, life your foot*, where the user did not specify whether they meant the left or right foot). From the CC component, the user can also set the HOLD or SEQUENTIAL mode by simply saying *hold mode* or *sequential mode*. The details of these two modes will be given in Section 4.5.2.

Server-Side-Support Component (SSSC). In Section 4.5.1 we will describe the ontology we have defined for the identification of all the actions of Zora involving each body part whereas Section 4.5.2 includes the details related to the identification and mapping of natural language expressions to ontology actions instances.

4.5.1. Robot Action Commands Ontology

We have designed the robot action ontology (https://www.w3id.org/zoraActions (accessed on 25 January 2021), https://bit.ly/2M4B7EN (accessed on 25 January 2021)) in OWL/RDF using Protege (https://protege.stanford.edu (accessed on 25 January 2021)). It is divided in two sections:

- The first part deals with the robot and the actions and includes `RobotAction` and `RobotBody`, shown in Figures 2 and 3.
  The individuals of `RobotAction` and those of `RobotBody` have been linked through the `involves` and `isInvolved` relationships. Overall, `RobotAction` and `RobotBody` include a total of 48 individuals. As an example, there are 4 individuals for the `ArmAction` class (subclass of `BaseAction`): `ArmDown`, `ArmForward`, `ArmSide`, `ArmUp`, and that correspond to the four movements that either the left or right arm of the robot (two individuals of the class `RobotArm`, subclass of `RobotBody`) may perform;
- The second part of the ontology is employed for machine reading, reasoning, and action command identification of natural language expressions given by the users. It consists:
  1. `ActionWord`, that defines the list of keywords (individuals) for all the actions that the robot may perform;
  2. `BodyPartWord`, which defines the list of keywords for all the body parts of the robot.

  The keywords are used for the mapping of natural language expressions to ontology action instances that will be covered in Section 4.5.2.

It is recommended that the reader look at the entire ontology to see all the classes and individuals that have been defined.

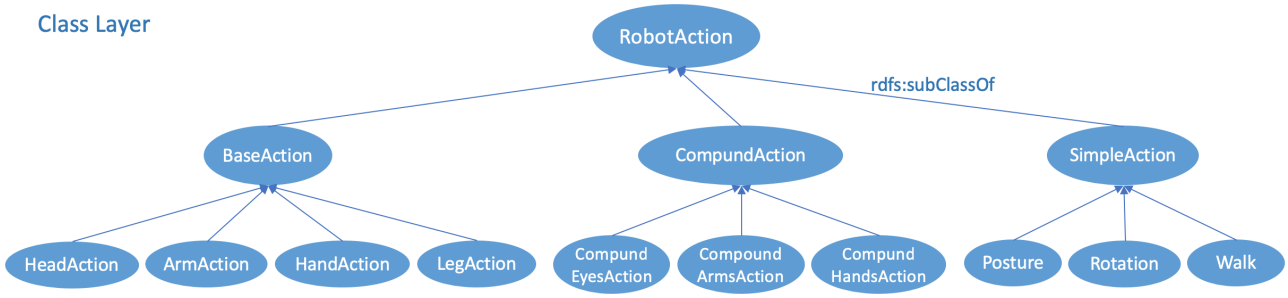

Figure 2. `rdf:subClassOf` relationship of actions in the Zora actions ontology.

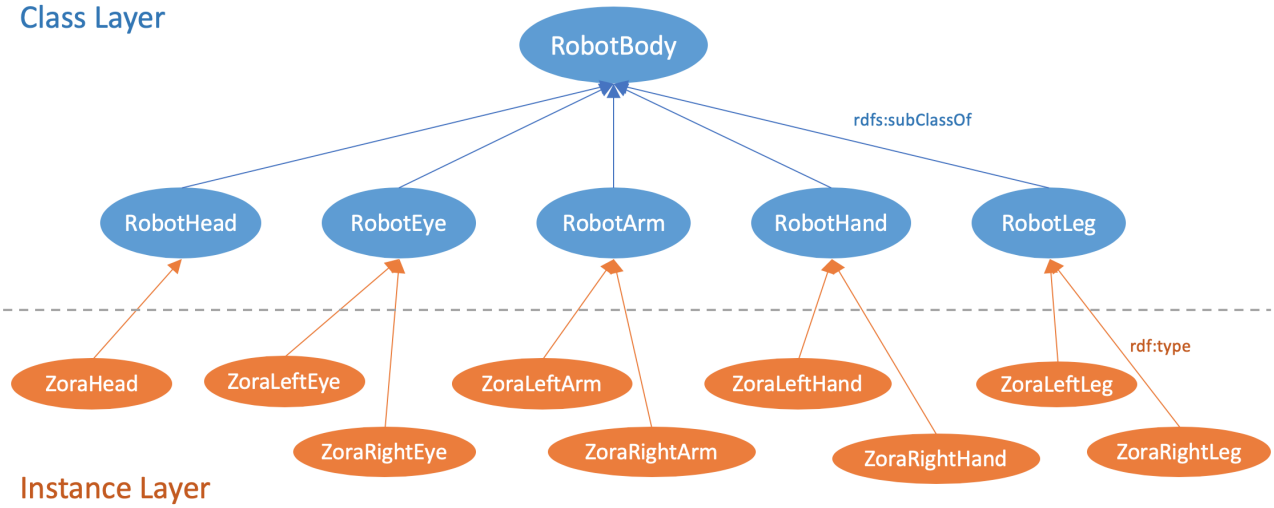

Figure 3. `rdf:subClassOf` relationship and `rdf:type` relationship section of the Zora actions ontology.

4.5.2. Mapping Natural Language Expressions to Ontology Actions Instances

To understand natural language expressions, the user's utterance is fed to the `NLP Engine` we have developed. It is built on top of Stanford CoreNLP and its output consists of a structured document that carries the syntactical information of words (e.g., Part-Of-Speech (POS), parse, and dependency trees).

To identify and extract the actions from the sentence, the `NLP Engine` starts identifying the verbs by searching the words tagged with the *verb* POS tag. Furthermore, by exploiting the information returned by Stanford CoreNLP (e.g., basic dependencies from the dependency parser), it also analyzes the words directly associated to the identified verbs to capture possible modifiers related to verb (e.g., *up*, *down*, etc.) or phrasal verbs. The detected expression is mapped against a list of verbs associated with the action instances of the ontology. Our method proceeds by checking the syntactic dependencies to understand whether the extracted verbal expressions are linked with a direct object dependency. After the verbs and body parts are identified, the ontology can now be analyzed to understand which actors are involved and which activity should be executed.

To note that single or multiple action commands might be given within the same sentence and will be executed sequentially. As the actions after the first one executed by NAO may be non compatible with the robot's position, we made use of the `incompatibleWithPrevious` relationship of the instances of `RobotAction`. When the user expresses a command that is not compatible with the current posture of the robot, Zora replies indicating such an incompatibility issue and without performing the underlying action.

To stress out the incompatibilities among actions, we have provided the robot with two modes: SEQUENTIAL and HOLD. With the former, the robot goes back to its default posture position after each action sentence is executed and the incompatibility might occur only in presence of multiple actions expressed within a single sentence. When set to HOLD mode, the robot remains in the position acquired after the last performed action of the

user. If the next action provided by the user is compatible with the robot's current posture, then the robot performs the new action, otherwise, the robot indicates the presence of incompatibility and skips that action remaining still. To let the robot change its position, the user will have to either change the robot mode to SEQUENTIAL, so that NAO will go back to its starting position, or give an action command compatible with the robot's current posture. In detail, the commands that the user must say to the CC to change the mode of this module are *Mode Sequential* and *Mode Hold*. A video showing this module in action with examples of incompatibilities and the different modes is publicly available (https://bit.ly/2EiFWEk (accessed on 25 January 2021)).

4.6. Object Detection Module

Choregraphe Component (CC).The CC of this other module is responsible for the interaction with the user. If the user says *recognize*, the CC is activated and informs the user that in five seconds the robot will take a picture (whose resolution is 640 x 800) from its front camera. Once the picture has been taken, it is forwarded to the SSSC to be processed and classified. The SSSC computes the bounding boxes and the related object categories and confidence values. The CC receives the output of the classification process performed by the SSSC and informs the user about the recognized objects asking if they are correct. The user, finally, does or does not confirm and their annotations are stored in a database within the SSSC and added to the existing training set. Periodically the classifier is re-trained with the updated (increased) training set.

Server-Side-Support Component (SSSC). The SSSC includes an object detection engine able to identify 91 different objects types. Moreover, to provide a more accurate object detection, for two object types (dogs and cats), the module performs one more fine-grained classification. In fact, when the recognized object is either a dog or a cat, the module employs two more classifiers to understand which breed the dog or the cat represents. Overall there are three classifiers we have trained: One for the general objects, one for the dog breeds, and the other for cat breeds.

We employed the TensorFlow Object Detection APIs (https://bit.ly/2lPqHJk (accessed on 25 January 2021)), which provides an open-source framework built on top of TensorFlow that makes it easy to construct, train, and deploy object detection models. TensorFlow Object Detection APIs can be used with different pre-trained models.

For the general object detection task, we have chosen a Single Shot MultiBox Detector (SSD) model with Mobilenet (ssd_mobilenet_v1_coco) which has been trained using the Microsoft COCO dataset (http://cocodataset.org (accessed on 25 January 2021)), and that consists of 2.5 M labeled instances in 328,000 images, containing 91 object types. The ssd_mobilenet_v1_coco is reported to have a mean average precision (mAP) of 21 on the COCO dataset. For further details on the SSD and the evaluation carried out on the COCO dataset please check the work of authors in [20].

For the object detection of the second level, we have employed the Oxford-IIIT-Pet dataset (http://www.robots.ox.ac.uk/~vgg/data/pets/ (accessed on 25 January 2021)), which consists of 4110 images distributed along 37 dog breeds and 12 cat breeds.

The SSSC has been embedded into a server-side application that exposes REST APIs that, given an input image, return the bounding box of each recognized object in the image along with a category and confidence value. We considered valid only the recognized objects having a confidence value equal to or higher than 60%. The SSSC is hosted in the cloud, where it can exploit GPU technology to largely speed up the classification process. Figure 4 shows an example image is already processed and with several bounding boxes related to different recognized items. The two with a label have a confidence value higher than 60%.

Figure 4. Example of objects detection. Several boxes are identified corresponding to different objects. Only those with confidence values higher than 60% are labeled and shown.

A video example of the interaction between the human and the object detection module of Zora is freely available (https://www.youtube.com/watch?v=WLjUPBTtdkE (accessed on 25 January 2021)).

4.7. Robot Behaviour Flowchart of the Proposed Modules

In this section, we will explain how the different modules we have described in the sections above are called and which reasoning the robot follows to decide to trigger each of them. Basically, there are particular statements that activate each of the module (e.g., 'play sentiment analysis', 'recognize', etc.). When the robot enables a specific module, there might be further options for the user: If the user specifies a command statement for that module, some option is activated, otherwise the module acts as programmed taking care of the user's expression. The scheme is illustrated in Algorithm 1. The input is the text from the user, already processed with speech-to-text technologies. First, the robot checks if a previous text activated the Semantic Sentiment Analysis module. If yes, the text is forwarded to the Semantic Sentiment Analysis SSSC to be processed. If not, the robot checks if the current text activates the Semantic Sentiment Analysis. The same thing happens with the Object Recognition Module. If none of the above occurs, the text is further checked if it toggles the SEQUENTIAL or HOLD mode of the Action Command Ontology Module. If it does not, the text is forwarded to the Action Command Ontology Module to identify potential robot actions. If at least one action command is recognized, then the Action Command Ontology Module is activated. Otherwise, the text is forwarded to the Conversational Agent Module which replies back to the user with an appropriate message related to the input text. Note that only one module at a time can be activated. To exit from the module currently activated, the user must say to the robot the word *exit*. The Action Command Ontology module has its peculiarities. In fact, when the user says an incomplete action command (e.g., *Zora, please lift your leg*), and there are no other modules activated, the Action Command Ontology module is activated. In such a case, the robot will ask the user to specify which leg it should lift. During the next interaction, the user will probably specify either *left* or *right*. Following the algorithm, the new text is forwarded again to the Action Command Ontology Module and the robot can therefore perform the required action. The module is therefore deactivated without the need for the word *exit*. The Conversational Agent Module does not need to be exited and it is activated when none of the other modules are. The described flowchart is hard-coded (it can be modified with different priority needs) into a Choregraphe script of a configuration module and if a new

module is added, this script should be updated depending on the programmer and their desires to include the new module and with which priority.

Algorithm 1 Robot Behavior Algorithm of the four modules described in the paper.

```
Data: Text from the user to be processed
Result: Robot Action
if the Semantic Sentiment Analysis module is activated then
    send the input to the Semantic Sentiment Analysis module;
    return
end
if input is "Play Sentiment Analysis" then
    activate the Sentiment Analysis module;
    return
end
if the Object Recognition Module is activated then
    send the input to the Object Recognition module;
    return
end
if the input is "Recognize" and there are not modules activated then
    activate the object recognition module;
    return
end
if the input is "Mode Sequential" or "Mode Hold" then
    set the mode of the Action Command Ontology module accordingly;
    return
end
Send the input to the Action Command Ontology module;
if no actions are identified then
    send the text to the Conversational Agent module;
    return
else
    activate the Action Command Ontology module;
    return
end
```

5. Experimental Results

In this section, we will show an evaluation analysis we have carried out for the proposed architecture in Section 5.1 and the results of the user experience interacting with the robot in Section 5.2.

5.1. Architecture Evaluation

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them [21].

Software architecture evaluation is built around a suite of three methods, all developed at the Software Engineering Institute: ATAM (Architecture Tradeoff Analysis Method) [22], SAAM (Software Architecture Analysis Method) [23], and ARID (Active Reviews for Intermediate Designs) [24]. Other methods created from their extension or mixing have appeared in the literature as well [25]. We have initially developed our architecture on a MacOS with 8GB RAM and 2,2 GHz Dual-Core Intel Core i7 using Choregraphe 2.1.4, connected to a hot-spot. Within the same hot-spot, the Zora robot was also connected. Two Amazon EC2 instances have been employed to run the modules. Python has been the adopted programming language (for CCs and SSSCs). Given the early development stage of our architecture, for our evaluation, we have chosen to use SAAM. In fact, it was first

proposed in 1994 with the basic goal to analyze system qualities early in the life cycle of the software architecture. This allows for a comparison of architectural options. Among the benefits that SAAM provides we have: (i) Potential problems can be detected in the early phase; (ii) the documentation can be improved and the understanding related to software architecture problems can be enhanced; (iii) it relates the different stakeholders (architect, maintainer, and developer); and (iv) it provides various techniques to improve the quality and classify different scenarios.

SAAM concentrates on modifiability in its various forms (such as portability, subsetability, and variability) and functionality. Modifiability is the ability to make changes to a system quickly and cost-effectively. Portability is the ability of the system to run under different computing environments. Subsetability is the ability to support the production of a subset of the system or incremental development. Variability represents how well the architecture can be expanded or modified to produce new architectures that differ in specific ways. Functionality is the ability of the system to do the work for which it was intended. The steps of a SAAM evaluation, which we have carried out within our analysis are the following:

- Identify and assemble stakeholders;
- Develop and prioritize scenarios;
- Classify scenarios as direct or indirect;
- Perform scenario evaluation;
- Reveal scenario interactions;
- Generate overall evaluation.

5.1.1. Identify Stakeholders

For the domain and context of our proposed architecture, we have chosen four different types of stakeholders: End-users, developers, testers, and the maintainers. End-users are those that use the robotic platform we have provided. Developers are people that are more interested in the clarity, completeness of the architecture, and clear interaction mechanisms. Testers are those checking error-handling consistency, cohesion, and conceptual integrity. Maintainers are those interested in maintaining the platform and in the ability to locate places of change. Two end-users, two developers, one tester, and one maintainer have been chosen for our analysis. The six persons were different and chosen among master students (end-users) and senior software engineers working within the university for a spin-off (developers, testers, and maintainers).

5.1.2. Identifying and Classifying Scenarios

Typical scenarios of the kind of evolution that the proposed system must support, such as functionality, development activities, and change activities, have been chosen. They can be either direct or indirect. Direct scenarios are those that can be executed by the system without any modification. Indirect scenarios are those that require edits. In the following, we list each chosen scenario and in parenthesis, we report if it is direct or indirect.

1. A maintainer with the task of installing the entire architecture using a different OS (Windows) than the one (MacOS) used for initial development (DIRECT);
2. A user interacting with the robot using each of the four presented modules (DIRECT);
3. A tester that performs all the needed debugging of the software architecture and the installed modules (DIRECT);
4. A developer needed to integrate into the current architecture a new module (INDIRECT);
5. A developer that needs to change the priority of activation of the installed modules (INDIRECT).

5.1.3. Scenarios Evaluation and Interactions

Scenario 1 has been evaluated by the maintainer. She used a PC with an OS (Windows) different than that used for development. She went through the documentation and was able to install all the required software. After making sure the installation was successful

and everything worked properly, she handed over everything to one end-user for deeper testing (indicated in Scenario 2).

Scenario 2 has been evaluated by two end-users. They have tested each of the four presented modules following the related user documentation and have thus interacted with the robot. They both have spent separately 2 h each playing with the robot. One used the architecture with the current settings (MacOS as OS). The other used the architecture configured by the maintainer as indicated in Scenario 1. They raised a few concerns. On the one hand, each module was correctly triggered by the user according to Algorithm 1. On the other hand, some of the modules returned an unexpected output showing some accuracy problem with the internal function of the module but not with the architecture itself. In particular, most of the problems were with the Conversational Agent module that often returned statements too far from the user input sentence. Besides the accuracy problems of the module (which can be improved with updates on just the module), the two users were satisfied with the overall interaction and usage.

Scenario 3 has been analyzed by a tester who set the debug mode through a configuration file and was able to generate the output of each module. He activated each of the four modules and looked at the output (in form of logs) of each (by looking at the cloud where each module was executed) and the used PC (which included the text to be sent and received to/from the robot for the interaction with the human). The tester did not find any critical issue or weakness in the entire testing process related to this scenario.

Scenarios 4 and 5 have been carried out by the two developers. One developed and embedded a Bingo game in the current architecture whereas the other developed an Odd or Even game. They went through the developer guide and the code identifying the needed components to interact with, their connections, and which interface to add. Hence they developed their games by means of CCs only. For the final integration of their games, they had to edit the main flow of the current modules depicted in Algorithm 1. As such, they identified the underlying interface (the Robot Behavior Interface depicted in Figure 1, which consists of a Python file running within the main Choregraphe software), and added the software hooks for their games. Basically, their games were activated by saying the words *Bingo Game* and *Odd or Even Game*. These two checks (one for each game) were included at the top of Algorithm 1, gaining priority with respect to the other modules. When one game was activated, the following interactions of the user affected the module related to the game only. As soon as the game was over, the control went back to the main Algorithm. The two developers were able to perform those changes and install smoothly the two modules. Their feedback was related to some improvement (more clear statements related to some technical aspects) in the documentation which would improve comprehension. The steps followed by the two developers to implement two new modules have been added to the developer guide as Hello World examples.

5.2. User Experience

In this section we will show the results of the user experience related to the interaction with the robot. We demoed the functionalities of the robot for about 30 minutes and then asked 10 participants to use the robot for performing four tasks: Sentiment analysis, conversation with the robot, ask the robot to perform some action, and ask the robot to identify some object. We took advantage of this session to gather further feedback about new potential use cases. Each of them had to fill a two-parts survey about their experience. The first part included six open questions (the first four also included five levels of reply, very bad, bad, normal, good, and very good), whereas the second part was a standard System Usability Scale (SUS) (https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (accessed on 25 January 2021)) questionnaire to assess the usability of the application. Here we summarize their answers to the open questions.

Q1. How do you find the interaction with the robot? Five users considered it good to use and three of them found it very good. The last two suggested to improve the speech-to-text capabilities because the robot could not understand well their statements.

We investigated why and it turned out that the problems derived from speech impediment (they were Italians that talked in English to the robot).

Q2. How effectively did the robot support you in classifying positive and negative sentences? Five users stated that the robot had an extremely positive accuracy (very good), whereas three found some errors on the ability of recognizing the correct sentiment (normal).

Q3. How effectively did the robot perform the action commands you gave? Nine users agreed on the same score (good) commenting that the robot always performed the action (from a list of possible actions) that was told to. One user (normal) commented that the robot had some problem identifying his commands. The problem was that the user employed some words not covered within the dictionary of the robot action ontology module and it was easy for us to add the missing terms.

Q4. How effectively did the robot support you in identifying objects? Eight users gave a score of good commenting that the robot identified the correct object they showed to her cameras. Two users (normal) commented that the robot did not correctly identify the objects they showed. The problem was that they were too far from the robot when performing this action. We fixed this by letting the robot say when the object detection module was triggered, to maintain a certain distance from her.

Q5. What are the main weaknesses of the resulting interaction? Users did not flag any particular weakness. One of them pointed out that weaknesses might appear when used extensively.

Q6. Can you think of any additional features to be included in the robot interaction? The suggested features were: (1) To have the robot say better instructions when she was turned on and each time a certain module was triggered and (2) the ability of saving all the data pertaining to the interaction with the user.

The SUS questionnaire confirmed a good opinion of the users, scoring 81.2/100, equivalent to an A grade and places our architecture in the 91% percentile rank (percentiles of SUS https://measuringu.com/interpret-sus-score/ (accessed on 25 January 2021)). All users felt very confident in playing with the proposed architecture (with an average score of 4.2 ± 0.5) and thought that it was easy to use (4.2 ± 0.6). In addition, they were happy to use it frequently (4.4 ± 0.3) and did not think that it was complex (1.5 ± 0.5) or that they would need the help of a technical person to use it in the future (1.7 ± 0.5).

6. Conclusions and Future Works

In this paper, we showed the architectural design for human-robot-interaction within the learning domain where we employed Deep Learning, Semantic Web, Sentiment Analysis, and Object Detection technologies for the development of four use cases. The architecture is scalable and flexible. The source code of the shown use cases is freely available on a public GitHub repository (https://github.com/hri-unica (accessed on 25 January 2021)). The robotic platform we leveraged is Zora, a NAO robot extended with a software middleware and accessible by non-expert users. The modules included in the architecture allowed the robot to perform sentiment analysis, generate answers to open-dialog natural language utterances, perform action commands depending on a robot action ontology that was developed, and identify objects which are seen by the robot camera. Each module has been described along with the two components it is composed of: The Choregraphe Component (CC), which is developed using the Python programming language within the Choregraphe suite, and the Server-Side-Support Component (SSSC), which contains the back-end and software needed for the underlying module. The SSSC resides in the cloud or in a server, where the most time-consuming computation lies and is performed. Moreover, it exposes REST APIs that may be called by the CC. One dedicated module handles the priority of the four modules through a configuration script that should be updated each time a new module is developed. Furthermore, we carried out an evaluation of the proposed architecture testing possible scenarios with several stakeholders who identified the following potential weaknesses. Only one Python file (the Robot Behavior

Interface) running within the main Choregraphe software and corresponding to the logic depicted in Algorithm 1 was concurrently affected by the last two scenarios, indicating low modularity of this component. From the developer's point of view, it would have been better to use a mechanism of module registration where information related to its activation and priority could be stored. This was invaluable feedback and, as such, we are already working to provide such a mechanism in the next version of our architecture. Furthermore, the interaction among different stakeholders was definitely a benefit for the resulting version of the documentation and architecture as it helped to see different sides of the same concept at a different level. Low coupling and high cohesion were two features observed by the developers involved in the last two scenarios.

One more evaluation has been carried out as far as the user experience is concerned. Answers from 10 participants to a two-part questionnaire showed good user opinions toward the proposed architecture.

Several other modules employing the sensors of Zora (microphones, cameras, etc.) can be added to the architecture to solve tasks in many other different domains. To this aim, there are several ongoing research efforts that we are carrying out to try to improve the human-robot interaction. One of them we are already exploring is to equip Zora with a virtual assistant such as Google Home so that the robot will be able to interact further with a human and to exploit the Google Home APIs to create new skills, controlling new home sensors, and, in general, have more knowledge.

As a further future direction, we would like to carry out a detailed analysis of the network traffic when incrementally using the proposed modules on several robots and over the same LAN.

Funding: This research received no external funding.

Data Availability Statement: Not Applicable, the study does not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Grace, K.; Salvatier, J.; Dafoe, A.; Zhang, B.; Evans, O. When Will AI Exceed Human Performance? Evidence from AI Experts. *J. Artif. Intell. Res.* **2018**, *62*. [CrossRef]
2. Breazeal, C.; Takanishi, A.; Kobayashi, T. Social Robots that Interact with People. In *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1349–1369. [CrossRef]
3. Mellmann, H.; Xu, Y.; Krause, T.; Holzhauer, F. NaoTH Software Architecture for an Autonomous Agent. In Proceedings of the SIMPAR 2010 Workshops, International Workshop on Standards and Common Platforms for Robotics, Darmstadt, Germany, 15–16 November 2010; pp. 316–327.
4. Mattamala, M.; Olave, G.; Gonzalez, C.; Hasbún, N.; Ruiz-del Solar, J. The NAO Backpack: An Open-hardware Add-on for Fast Software Development with the NAO Robot. In *RoboCup 2017: Robot World Cup XXI*; Springer: Cham, Switzerland, 2017. [CrossRef]
5. Röfer, T.; Laue, T. On B-Human's Code Releases in the Standard Platform League—Software Architecture and Impact. In *RoboCup 2013: Robot World Cup XVII*; Behnke, S., Veloso, M., Visser, A., Xiong, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 648–655.
6. Alonso, R.; Concas, E.; Recupero, D.R. A Flexible and Scalable Social Robot Architecture Employing Voice Assistant Technologies. In Proceedings of the Workshop on Adapted intEraction with SociAl Robots, cAESAR 2020, Cagliari, Italy, 17 March 2020; Carolis, B.N.D., Gena, C., Lieto, A., Rossi, S., Sciutti, A., Eds.; CEUR-WS.org: Leipzig, Germany; 2020; Volume 2724, pp. 36–40.
7. Atzeni, M.; Dridi, A.; Recupero, D.R. Using frame-based resources for sentiment analysis within the financial domain. *Prog. AI* **2018**, *7*, 273–294. [CrossRef]
8. Atzeni, M.; Recupero, D.R. Fine-Tuning of Word Embeddings for Semantic Sentiment Analysis. In *Communications in Computer and Information Science, Proceedings of the Semantic Web Challenges—5th SemWebEval Challenge at ESWC 2018, Heraklion, Greece, 3–7 June 2018*; Revised Selected Papers; Springer: Cham, Switzerland, 2018; pp. 140–150. [CrossRef]
9. Atzeni, M.; Dridi, A.; Recupero, D.R. Fine-Grained Sentiment Analysis on Financial Microblogs and News Headlines. In *Communications in Computer and Information Science, Semantic Web Challenges—4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, 28 May–1 June 2017*; Revised Selected Papers; Springer: Cham, Switzerland, 2017; Volume 769, pp. 124–128. [CrossRef]
10. Dridi, A.; Atzeni, M.; Recupero, D.R. Bearish-Bullish Sentiment Analysis on Financial Microblogs. In Proceedings of the 3rd International Workshop at ESWC on Emotions, Modality, Sentiment Analysis and the Semantic Web Co-Located with 14th ESWC 2017, Portroz, Slovenia, 28 May 2017.

11. Dridi, A.; Atzeni, M.; Reforgiato Recupero, D. FineNews: Fine-grained semantic sentiment analysis on financial microblogs and news. *Int. J. Mach. Learn. Cybern.* **2018**, *10*, 2199–2207. [CrossRef]
12. Recupero, D.R.; Dessì, D.; Concas, E. A Flexible and Scalable Architecture for Human-Robot Interaction. In *Lecture Notes in Computer Science, Proceedings of the Ambient Intelligence—15th European Conference, AmI 2019, Rome, Italy, 13–15 November 2019*; Proceedings; Chatzigiannakis, I., de Ruyter, B.E.R., Mavrommati, I., Eds.; Springer: Cham, Switzerland, 2019; Volume 11912, pp. 311–317. [CrossRef]
13. Dragoni, M.; Cambria, E. Semantic Sentiment Analysis Challenge at ESWC2018. In *Communications in Computer and Information Science, Proceedings of the Semantic Web Challenges—5th SemWebEval Challenge at ESWC 2018, Heraklion, Greece, 3–7 June 2018*; Revised Selected Papers; Springer: Cham, Switzerland, 2018; pp. 117–128. [CrossRef]
14. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Carson City, NV, USA, 5–10 December 2013; Curran Associates Inc.: New York City, NY, USA; 2013; Volume 2, pp. 3111–3119.
15. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, MD, USA, 23–24 June 2014; Association for Computational Linguistics: Philadelphia, PA, USA; 2014; pp. 55–60. [CrossRef]
16. Wang, Z.; Wang, Z.; Long, Y.; Wang, J.; Xu, Z.; Wang, B. Enhancing generative conversational service agents with dialog history and external knowledge. *Comput. Speech Lang.* **2019**, *54*, 71–85. [CrossRef]
17. Atzeni, M.; Reforgiato Recupero, D. Deep Learning and Sentiment Analysis for Human-Robot Interaction. In *Lecture Notes in Computer Science, Proceedings of the The Semantic Web: ESWC 2018 Satellite Events, Heraklion, Greece, 3–7 June 2018*; Gangemi, A., Gentile, A.L., Nuzzolese, A.G., Rudolph, S., Maleshkova, M., Paulheim, H., Pan, J.Z., Alam, M., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 14–18.
18. Danescu-Niculescu-Mizil, C.; Lee, L. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL, Portland, OR, USA, 23 June 2011.
19. Reforgiato Recupero, D.; Spiga, F. Knowledge acquisition from parsing natural language expressions for humanoid robot action commands. *Inf. Process. Manag.* **2020**, *57*, 102094. [CrossRef]
20. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Lecture Notes in Computer Science, Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 21–37.
21. Bass, L.; Clements, P.; Kazman, R. *Software Architecture In Practice*; 2003. Available online: https://ptgmedia.pearsoncmg.com/images/9780321815736/samplepages/0321815734.pdf (accessed on 26 January 2021).
22. Kazman, R.; Klein, M.; Barbacci, M.; Longstaff, T.; Lipson, H.; Carrière, S. The Architecture Tradeoff Analysis Method. In Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193), Monterey, CA, USA, 14 August 1998; pp. 68–78. [CrossRef]
23. Kazman, R.; Bass, L.; Abowd, G.; Webb, M. SAAM: A method for analyzing the properties of software architectures. In Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, 16–21 May 1994; pp. 81–90.
24. Clements, P. *Active Reviews for Intermediate Designs*; 2000; p. 26. Available online: https://resources.sei.cmu.edu/asset_files/TechnicalNote/2000_004_001_13685.pdf (accessed on 26 January 2021).
25. Ali Babar, M.; Gorton, I. Comparison of scenario-based software architecture evaluation methods. In Proceedings of the 11th Asia-Pacific Software Engineering Conference, Busan, Korea, 30 November–3 December 2004; pp. 600–607. [CrossRef]