Generating Voice User Interfaces from Web sites

GONZALO RIPA, MANUEL TORRE, MATIAS URBIETA, GUSTAVO ROSSI, ALEJANDRO FERNANDEZ, ALEX TACURI, and SERGIO FIRMENICH, LIFIA, Facultad de Informática/CICPBA, Universidad Nacional de La Plata, Argentina

Virtual assistants allow end users to interact with apps using a conversational mode. A particular kind of device supporting virtual assistants is the smart speaker, which allows end users to access contents and services, perform searches and command smart environments via voice interaction. These devices are gaining relevance among users, and the number of apps available for giving them more conversational capabilities grows constantly. However, there is a gap between what can be achieved using these devices and the information available on the web, i.e. most web apps don't have one smart speaker app counterpart. In this work, we present a two-step approach to defining conversational interfaces for virtual assistants based on existing web sites. During the first step, the user creates web content blocks, which describe target contents and the strategy for retrieving them. In the second step, the user uses web content blocks to specify the conversational interface. Although the approach could be considered generic enough for any kind of conversational interface, we focus our research on voice user interfaces, considering smart speaker apps as a major target. With our approach, apps based on voice user interfaces for smart speakers may be developed in a no-code manner. We describe and illustrate the approach by presenting usage examples and an evaluation using the Alexa service and an Amazon Echo device. The evaluation shows promising results for two case studies where there is no significant difference between the user experience when comparing solutions developed using the Alexa SDK versus apps using our approach.

Total words: 8260

Keywords:: virtual assistants, conversational interfaces, web apps

1 INTRODUCTION

Natural language processing and speech recognition technologies have experienced strong advances in the last decade. As a result, voice-based conversational interfaces have become ubiquitous. Nowadays, the use of conversational voice interfaces may occur in different contexts: car driving, smart environments, web access, etc. Many mobile apps react to voice commands. Voice-based conversational interfaces are also used for web accessibility, through voice commands [1] [2]. This evolution has enabled new kinds of devices, such as smart speakers, which reached broad consumption by end users [3].

Modern smart speakers are wirelessly connected loudspeakers that offer voice interaction to access information and services. They may host third-party apps; that is, users can install apps in their smart speakers (called skills in the case of Amazon Echo devices) in the same way that they install apps in their smartphones. Smart speakers' apps are frequently used in combination with smart devices and IoT platforms (for example, IFTTT [1] or Node-Red [4]). Smart speakers also come with built-in apps to respond to web searches, as is the case of Amazon Echo and Google Home, which are one step ahead with the ubiquity of access to web content and services.

Beyond general *queries* solved with a search, other apps focus on getting specific information and services already published by existing web apps, such as reading news from a news portal or asking for a product price from Amazon.com. Very well-known websites such as Expedia.com now offer an Amazon Echo app that lets end users search prices for accommodation and flights, among other functionalities. Suddenly, web app owners

[1] https://ifttt.com/amazon_alexa

Authors' address: Gonzalo Ripa, gripa@lifia.info.unlp.edu.ar; Manuel Torre; Matias Urbieta; Gustavo Rossi; Alejandro Fernandez; Alex Tacuri; Sergio Firmenich, LIFIA, Facultad de Informática/CICPBA, Universidad Nacional de La Plata, Argentina.

may (should) support another type of front-end interface to interact with their contents and services, based on voice-based user interaction. This could explain the number of available apps (called skills) in Amazon Echo's marketplace [3].

A similar process occurred when smartphones emerged, and web app owners started to deliver native mobile apps. But, unlike the case where users can visit any web site from the mobile web browser, for smart speakers when a native app counterpart does not exist, there is no possible way to access contents and services offered by a web site.

Web apps still play a very relevant role in the users' daily life; we use them for reading news, consuming different services, working, and even interacting with smart devices in the web of Things. However, despite the progress on Model-Driven Engineering [5] and Multi-Modal User Interfaces [6], most web sites are not developed with conversational technologies in mind and therefore delivering device-specific apps (e.g. for smart speakers) is usually expensive.

This paper aims to fill the gap between available smart speakers' apps and online web content and services users consume daily by browsing the web. We propose an approach for the creation of voice-based conversational interfaces. To do it, we developed a tool called SkillMaker, a web Browser extension that consists of two parts: a web content extraction tool and a model editor. In this way, users can create dynamic content blocks, that will be used later in the editor to define a voice-based way to access and read them. Content blocks can be thought of as parts of the web pages that organize data of a website sub-domain and are recognizable by users. One example of a content block for a news domain could include the title, image, and description from the first news appearing on the main page of a portal. Or in an e-commerce domain, the name, price, and description of some products listed on some sub-category from the own web page.

We present a model for the definition of Voice User Interfaces (VUIs) based on web contents, plus a JavaScript-based engine (called Skill Hub) to interpret these specifications. In this way, these interfaces may be executed on any device with an Internet connection and a JavaScript environment. The approach could be used by end users or by those web sites owners that want to deliver smart speaker skills. We have limited the research to voice interaction, although the approach has the potential to combine also visual interaction as the result of voice commands. As we will show, the approach generates VUIs that are as usable as native smart speaker apps. As a concrete case study, in this paper, we use the Amazon Echo smart speaker to evaluate the VUIs produced with SkillMaker, which were compared with native skills available in Amazon's marketplace. As the approach aims at being an alternative to native apps based on native SDKs like Alexa or Google Assistant, we conducted an experiment reporting promising results where there is no difference in usability satisfaction between native apps and solutions modeled with our approach. For this, we designed an experiment to answer the following research question: is the user experience equivalent for Native and SkillMaker apps?

In section 2, we introduce some background and related works. In Section 3, we introduce our approach to define Voice User Interfaces. We illustrate our approach in section 4 with some examples and supporting tools. In section 5, We conducted an evaluation showing no difference in user satisfaction between native apps and SkillMaker apps. Finally, we give some conclusions and future work in Section 6.

2 BACKGROUND AND RELATED WORKS

We have divided this section into three parts. The first one is about voice user interfaces and their main current use. The second one is oriented to defining basic concepts and discussing related works regarding voice interaction and web access. The third one is focused on discussing different works related to manipulating existing web pages.

2.1 Voice user interfaces and smart speakers

In the literature, have been used several terms to refer to the interaction between the user and the machine based on a conversational metaphor, e.g. virtual assistants, conversational interfaces, personal assistants, intelligent agents, etc. [7]. The machine feedback may be visual-based (lights, display), voice-based, movements (like in the case of robot-like assistants) or a combination of these [8]. In this paper, we focus on *Voice User Interfaces* (VUIs) or *voice-based conversational interfaces*, following the definition from Porcheron et al. [7], which focuses "specifically on interfaces that are primarily voice-based, in which the user talks to the device and the device answers with a synthesized voice".

Although VUIs began to be widely used with their inclusion in smartphones (Siri for IOS devices, Google Now for Android devices), the appearance of gadgets such as Google Home or Amazon Echo is changing their daily use and pervasiveness. These smart speakers allow users to start some conversational interaction with a voice command expressed in natural language, such as “Alexa, tell me the news”, in the case of the Alexa service from Amazon. Smart speakers have a set of base capabilities such as telling the time and the weather, playing music, reading the news, controlling IoT devices, etc. Also, new capabilities (known as "skills") may be downloaded from vendor stores. This way, other possible user tasks, such as home automation, travel plan, online shopping, alternative information access, etc., may be added by installing third-party skills. Despite the vast ecosystem of skills built by developers, and the chance to extend the base capabilities of the service installing them, there is a limitation among these skills: very few are known and used, so many features remain hidden for the user and wasted (Spallazzo et al. [8]).

Currently, and just mentioning one case, the Alexa Skill repository is organized into categories and offers more than 80.000 skills (in December of 2021[2]), demonstrating the huge growth of this kind of software if we compare the number of skills available at the end of 2017 [3]. The range of skills is very broad, and it is also remarkable that, for some categories, there are more than 4.000 ones (e.g. News). This is not surprising if we consider that, for the same purpose, we have a great variety of web sources and services to achieve it. Consequently, it is straightforward to think about the possibility of creating new kinds of skills using publicly available web content and services (either in the way of Restful APIs [9] or directly parsing and extracting the desired web content).

Beyond this quantitative analysis, a recent study (made with Google Home users) [10] shows that smart speaker users, use skills related to Music, Information, Automation, Smalltalk, Alarm, Weather, Video, Time, Lists, and Others (in order of relevance, i.e., from most used to less-used skills). One more time, we can appreciate that, for most of these categories, there are several web apps counterparts from where end users could read information or complete some business process using a device supporting normal web browsing. Although some skills for automation of IoT devices not have web apps counterparts, it is clear that a broad range of skills could be, or even are, based on existing web contents and functionalities.

The possibility of creating personalized voice commands is also relevant and has already been studied in the context of multi-modal user interfaces [11]. In this regard, it is important to highlight the existence of several automated platforms that allow users without programming knowledge to create voice-based applications (such as skills for Alexa) using visual, drag-and-drop editors. Popular among them are VoiceApps.com [3] and VoiceFlow.com[4]. While these platforms offer some flexibility, they do not allow end users to fully specify the content of the VUI, including dynamic content obtained from the web. This limitation means that end users are restricted to using pre-determined text. In contrast, our approach enables end users to define VUI applications based on existing web content and provides more flexibility regarding the content to include in the VUI.

[2]https://voicebot.ai/2021/01/14/alexa-skill-counts-surpass-80k-in-us-spain-adds-the-most-skills-new-skill-introduction-rate-continues-to-fall-across-countries/
[3]https://voiceapps.com/
[4]https://www.voiceflow.com/

In this paper, we explored how web contents can be extracted, processed, and used as responses by VUI apps (smart speakers apps in particular). Our approach involves a set of tools that let users visually develop these VUI specifications using existing web sites.

2.2 Conversational interfaces and the web

In the context of web browsing, the idea of using a personal assistant already existed fifteen years ago. The concept of Personal Information Assistant (PIA) was introduced in [12] as a "software robot that a user deploys to retrieve targeted data from web sources that he/she is interested in". The authors propose techniques for end users so they will be able to create PIAs using just a web browser and learn how to navigate and extract information.

To automate repetitive web browsing tasks, Borodin et al. proposed an end-user development of voice commands based on web macros [13] which allow users to record actions to be executed and reused later. This approach based on a non-visual browser [14], enables users to interact with web content through voice or text commands, becoming a promising interaction alternative against existing screen readers. End users and developers can collaborate with accessibility metadata (like adding alternative text to images), similar to another work that uses voice interaction to improve Web accessibility [1].. The same authors proposed improving web accessibility in the context of open data portals [15]. Other notable development in voice interaction and the web include Firefox Voice [16], a browser extension that enables users to use their voice to perform various actions.

Several of these approaches arise before the evolution of conversational interface apps and gadgets, such as chatbots, smart speakers, etc. In this regard, the works mentioned above are voice-command approaches oriented to trigger automatic tasks in the context of a web browser instead of being focused on conversational interfaces between apps and users. While these approaches have proven useful, there is still a need for more advanced conversational interfaces that can facilitate access to web content and services interactively. Beyond receiving a voice command and giving a response based on specific web content, the conversational interface may propose an interactive way to deal with that response, explore more from there, etc.

A new approach to automating tasks in the context of web browsing involves using apps called "skills" [17], defined by users through voice commands and multi-modal specifications, annotating some parts of the web using different inputs, combining a Programming By Demonstration (PBD) approach with voice commands. These skills allow users to perform a series of combined tasks, such as navigating to specific websites and obtaining information, by simply invoking them with predetermined voice commands. However, responses to these commands are only provided visually through a pop-up on the current web page.

Although we focus on voice-based conversational interfaces, we see similarities with other conversational interfaces, such as chatbots, which are conversational agents providing access to information and services through everyday language [18]. Chatbots may be used for multiple purposes and from several devices and contexts. In web apps, one of the most common uses is to provide customer service, i.e. to support users by emulating a human conversation, such as in e-commerce apps [19].

Prior research on chatbots for web browsing has focused on conversational web interaction [20] and web browsing [21], both of which rely on HTML annotations defined by app developers. The former is a conceptual paper in which authors describe conversational web interaction, define a set of requirements for bot generation and propose an architecture for fulfilling those requirements based on HTML annotations. In the latter, the authors show an implementation of their prior architecture based on those HTML annotations.

In contrast, our approach allows end users and developers to add annotations to any website on the client side without requiring programming skills.

2.3 Web content extraction

The idea of information extraction we use in this approach is similar to some techniques for web Scraping [22]. A usual method used for information extraction is the annotation of web content. Some web sites already tag their contents allowing other software artifacts (like a web Browser plugin) to process those annotations and improve the interaction with that structured content. A well-known approach for giving some meaning to web data is Microformats [23]. Some approaches take advantage of the underlying meaning given by Microformats, microdata, or JSON-LD, detecting these objects on the web page and allowing users to interact with them in new ways. According to [24], only 5,64% of more than 40 million web sites provide structured data (Microformats, Microdata, RDFa, etc.). This fact raises the importance of empowering external annotations or semantic structure when it is't available.

Some End-User Development approaches arose to empower users to solve their particular needs by themself. For instance, MashMaker [25] allows extracting widgets with properties and later inserting these widgets into other web pages to modify the app. Another work proposes the structuring and abstraction of data models from the client side to create personal web sites that run purely on the client side, i.e. the end user's web browser [26]. SearchAPI [27] allows end users without programming skills to create search APIs by visually selecting the UI parts of web apps search engines. This way, the domain objects from an app can be searched, emulating user interaction. Similar approaches have arisen under the web augmentation technique, which is still a promising technology for end-user development [28]. However, from our knowledge, these approaches are not considered for developing VUI specifications by reusing existing web content.

3 AN APPROACH FOR VUI BASED ON WEB CONTENTS

The base of our development approach for VUIs is three-fold:

(1) A method and tools allowing users to select and define web content blocks. For this purpose, we use web content annotation and abstraction methods by means of visual tools and a simple configuration mechanism. We explain what the content blocks mean later in this section and show several examples in the next. For now, it is enough to see content blocks as abstractions of web content that are materialized with an extraction template.
(2) A way to specify how content blocks can be used in a VUI and how such a VUI could behave. We found out that flowcharts are a suitable way to model the behavior of VUIs; where nodes represent content blocks and connections represent how these blocks are organized and read as a response to the user voice commands. Each set of nodes and connections conforms to a VUI specification.
(3) An interpreter that processes a VUI specification, gets web pages' DOM dynamically, and then answers to the user with the extracted content blocks plus further conversational options configured during the VUI design.

Figure 1 provides an overview of the approach. It's divided into two layers: the VUI Definition Tools and the VUI app Elements. The VUI Definition Tools layer includes the visual tool used for the content extraction and definition processes (SkillMaker box) and the VUI apps Backend. SkillHub is the main component of the VUI app Backend, whose job is to consume the VIU specifications created with SkillMaker. Figure 1 also compares our approach, on the left hand, with the approach used to build a native app, on the right hand (as those built for Alexa[5] or Google Home[6]).

The second layer shows the elements that compose a conversational flow between users and voice-based devices. Both apps, made with our approach and those built as native apps, will connect with the voice cloud service through Restful API communication. This service will be available by the voice-based technology chosen

[5]Alexa SDK - https://developer.amazon.com/en-US/docs/alexa/sdk/alexa-skills-kit-sdks.html
[6]Google Assistant - https://developers.google.com/assistant/sdk

and take care of processing the words spoken by users in requests, mapping to user utterances, and finally delivering the responses in voice format. User utterances will trigger actions called skills (which handle the logic for the next steps of the conversational flow based on the user requests). Both user utterances and skill actions will be defined by developers (through some GUI offered by the Voice-based technology) or by users/developers through our tools.

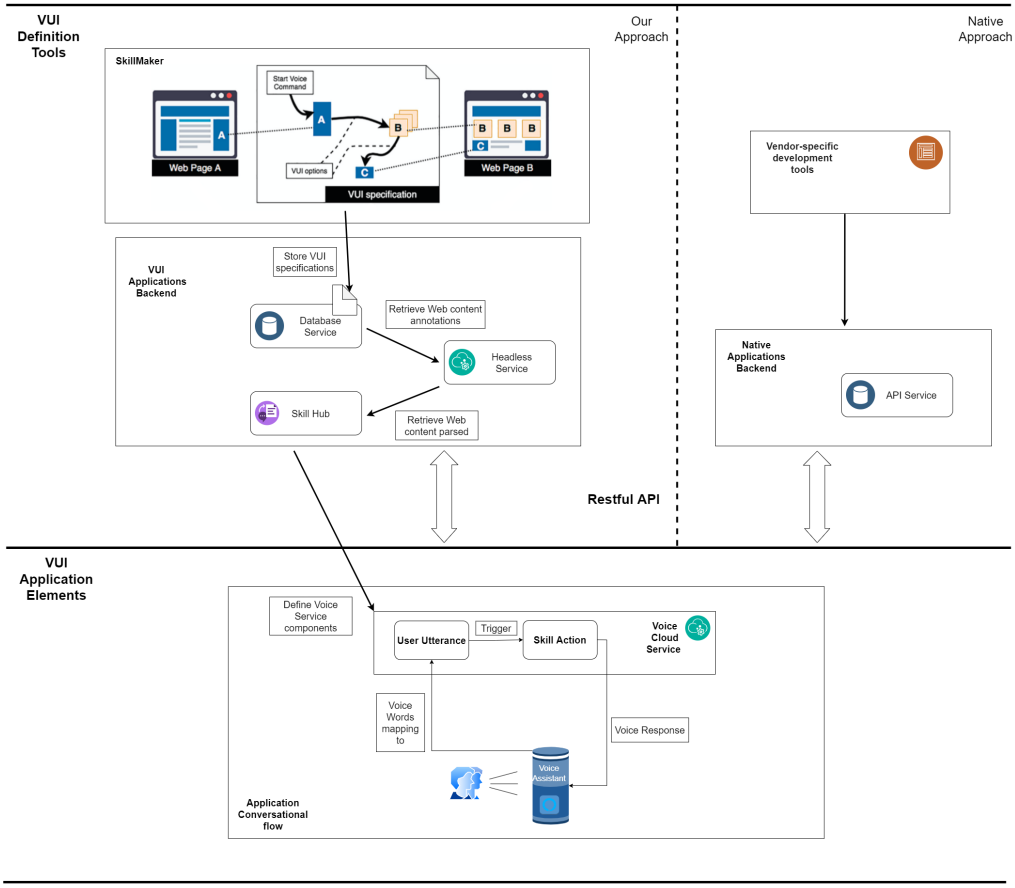

Fig. 1. Overview of the approach

Three elements stand out in the SkillMaker box: web Page A, the VUI specification, and web Page B. The VUI specification includes a set of user-defined contents from both web pages: block A from web Page A and two other content blocks from web Page B, named B (that is, in fact, a collection of related web contents) and C. The main idea is that these content blocks are read consecutively in response to a specific voice command but with some behavioral configuration for both nodes (content blocks) and transitions between them. The customization process will be explained in more detail at the end of the following section.

To support this design activity, we have implemented a web browser extension called SkillMaker as a prototype. This extension allows users to annotate web content and define how to use it in a VUI. To achieve modularity, our approach separates the definition process into two steps: one for defining content blocks (a) and the other for the VUI (b). This way, content blocks can be reused in multiple VUIs.

As a first step in the information extraction process for (a), SkillMaker recognizes some elements from the web page (used to display text) and extracts them from the HTML document itself. Once the page is loaded, all of these elements are made "draggable" to allow them to be extracted.

As a second step in the information extraction process for (a), SkillMaker builds an XPath expression (used to identify the element in the DOM hierarchy) and identifies some properties (either automatically or manually by user action). Using XPath expressions allows us to make content block information accessible to our VUI interpreter at different times, so regardless of any changes to the information the content block will be available for consumption. As a disadvantage, the XPath directions could mutate (due to evolutive or maintenance UI changes) and become unavailable, something we handle by providing feedback from the VUI side and asking users to repeat the content block definition process (a).

Our approach not only involves the VUI definition process but also the consumption of VUI specifications. To support the latter process, we have developed several web services that store the VUI specifications (Database Service) and parse web content annotations (Headless Service), as shown in Fig. 1. In addition, we have created an interpreter called SkillHub, which can process every user-spoken request, obtaining the parsed web content from the Headless Service and returning the text as speech through the voice-based device.

In summary, the basic idea is that a user can create a VUI by defining voice commands, specifying how the VUI will respond by reading content blocks, and adding a specific configuration for each content block. To formalize and simplify the specification of VUIs, we have defined a model; a simplified version of this model is presented in 2. It allows us to verify if a VUI specification is compliant with the interpreter and also to understand the different parts of the VUI specification, enabling the interpreter to know the available commands in a given interaction context when interpreting a skill. Because of this, SkillHub offers navigational menus to allow end users to interact with unknown VUIs.

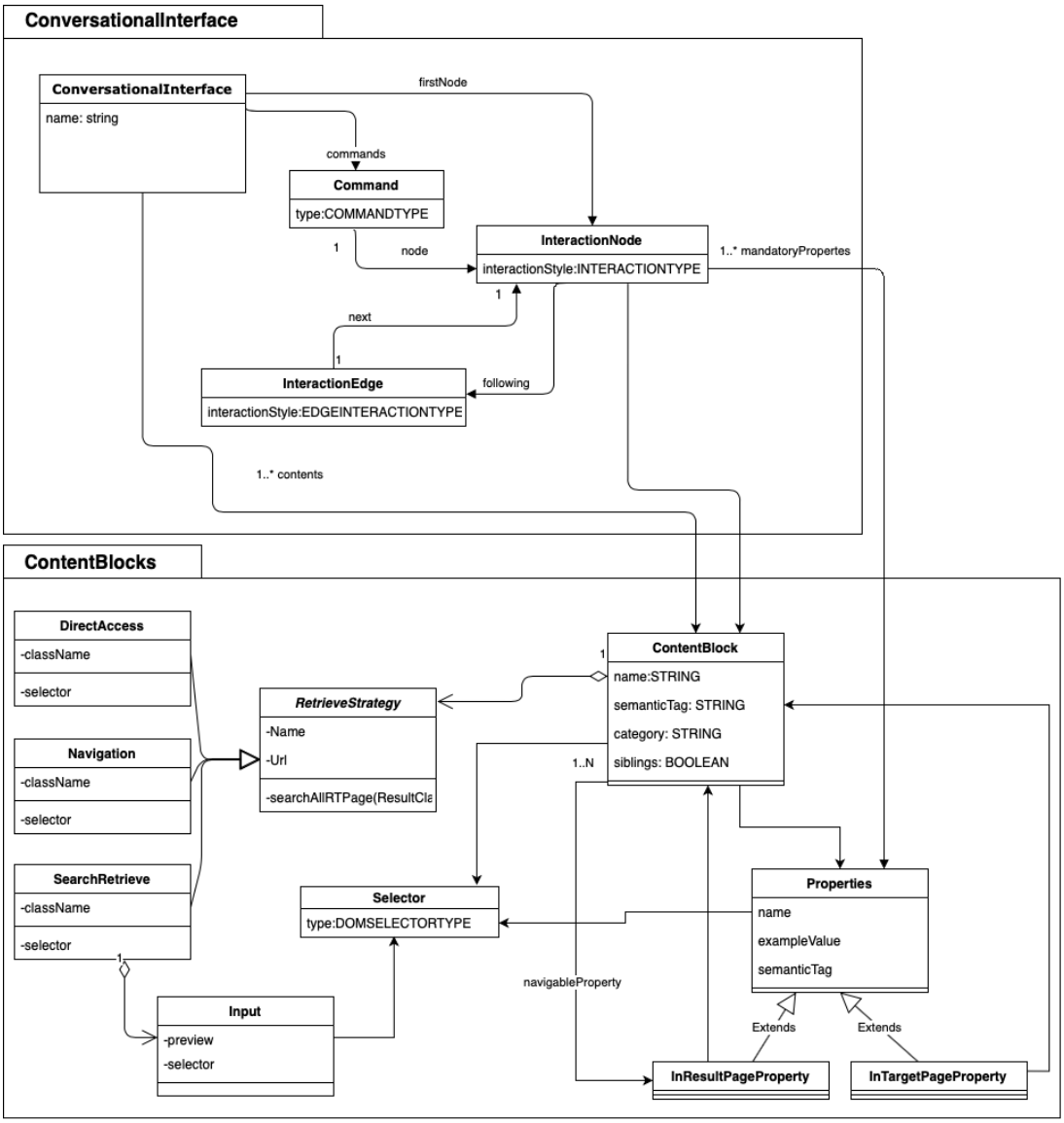

Fig. 2. Model for conversational interfaces based on web contents

The main elements in the "ContentBlocks" package are:

- "ContentBlock" and "Properties": "ContentBlock" represents any web content. Although the approach allows the definition of a content block without semantic annotations, we decided to attach a basic semantic definition with the "title" property to homogenize content blocks when designing the VUI. If desired, the user can define a more complex extraction template with a more detailed semantic structure. A "ContentBlock" can be a single UI element or a set of sibling UI elements; this is possible because the approach allows for various types of DOM selectors.
- "RetrieveStrategy": is another feature of a "ContentBlock" that determines how to access it. We discuss possible strategies for accessing a ContentBlock in the following section.

As for the "ConversationalInterface" package, the most important classes are:

- "Command": this class represents voice commands that initiate an interaction with the VUI.
- "InteractionNode": this class represents an interaction with a particular "ContentBlock"; i.e. it defines how to handle a "ContentBlock" in terms of how to build a response for a voice command.
- "InteractionEdge": sometimes, a response to a particular voice command may require using more than one "ContentBlock"; in this case, "InteractionEdge" allows users to establish connections between the "ContentBlocks" that make up a single VUI response.

Although we do not provide further details in this paper due to space constraints, it's important to note that the "INTERACTIONTYPE" property (which belongs to "InteractionNode") and the "EDGEINTERACTIONTYPE" property (which belongs to "InteractionEdge") offer great flexibility in defining how to respond to a voice command. For example, users can specify which property of a "ContentBlock" to use in the response, using different values of the "INTERACTIONTYPE" property. Also, users can define if the response should offer the remaining properties as more information. Similarly, "EDGEINTERACTIONTYPE" allows users to define different transitions between "ContentBlock" instances, such as aggregating some text before reading the following content block. In the next section, we provide examples of both types of configuration mentioned.

4 RATIONALE, TOOLS AND EXAMPLES

First, we present the rationale behind how web content can be extracted and used to define VUIs. Then, we describe our prototype, called SkillMaker, through several examples. For each example, we first show the tool for defining content blocks and then the editor of the VUI, where these content blocks will be used. The entire environment is deployed as a Chrome browser extension. The prototype has been used to create several examples and is ready for use in a controlled environment.

4.1 Rationale: From web Content to VUIs

This section discusses how web content is abstracted as information objects and then used in VUIs. We propose an abstraction called *content block*, which refers to a specific piece of web content identified within the general layout of a website. We envision three ways to abstract web content:

- One strategy for defining content blocks is based on the individual selection of each part of a web page. For example, imagine a news portal where the main news does not follow a regular presentation pattern. In this case, the user would need to individually select each UI element containing one of the news to create its corresponding content block.
- Content blocks are also defined from a relationship between UI elements that share common features. It's usual for related elements to have a similar structure and presentation, such as the results of a content search or a set of news related to a specific topic. When users select one of these elements, new content blocks are defined to obtain all related elements by comprehension.

- A third way to define content blocks is through semantics. web apps often expose representations of domain objects in their UI, such as news articles, products, etc. It means that on the client side (in the web browser), a simple domain object can be recreated using the attributes presented in the UI. For example, consider a VUI for the IMDB's "coming soon" movies page. It would be interesting to provide this VUI with the ability to understand the website content and extract the main domain concepts: the movie title, release date, genre, description, director, and cast. This way, the conversational interface can be based on the domain, which can be very convenient for users to obtain specific information when interacting in a voice-based conversational style.

In addition to defining an extraction template for content blocks (with or without a semantic domain), another outstanding aspect is how to access a web page DOM element to create them. We envision three ways:

- Direct Access: given a URL (which can be static or based on an API-based URL that allows changing values for specific parameters), it is possible to retrieve the web page. This method helps get the current state of a website that offers frequently updated information, such as news, weather, movies, etc.
- Navigation: when the desired content cannot be accessed using a predefined URL, it is possible to navigate through the links available on a specific web page's URL. This feature also serves to get more information about a content block. For example, reading the details for the main news article may involve following a link that allows users to navigate from the home page of a website to the specific news article's page, whose URL may not be known beforehand. However, although users can access the desired content by automating navigation, a first step using the "Direct Access" method will be required to start this navigation method.
- Web Search: searching specific websites (such as products on Amazon, books on Goodreads, movies on IMDB, etc.) is another way to access content blocks. In these cases, it is expected that the keywords for the search will be obtained from the user's voice interaction. To achieve this, we have previously developed a tool for creating search services by abstracting UI components related to search functionality [27].

4.2 Examples

4.2.1 Main news from different portals. The process of defining a content block for the main news from different portals begins when the user decides to create a content block for the current website. That happens by clicking the button of the SkillMaker web extension from the top browser bar (point 1 in Fig. 3). As a result, the DOM elements from the current web page highlights when the mouse pointer is over them. Once the user selects the option to "Create a default content" and chooses a specific DOM element, they can drag and drop it (point 2 in Fig. 3) into the extraction template definition box (point 3 in Fig. 3).

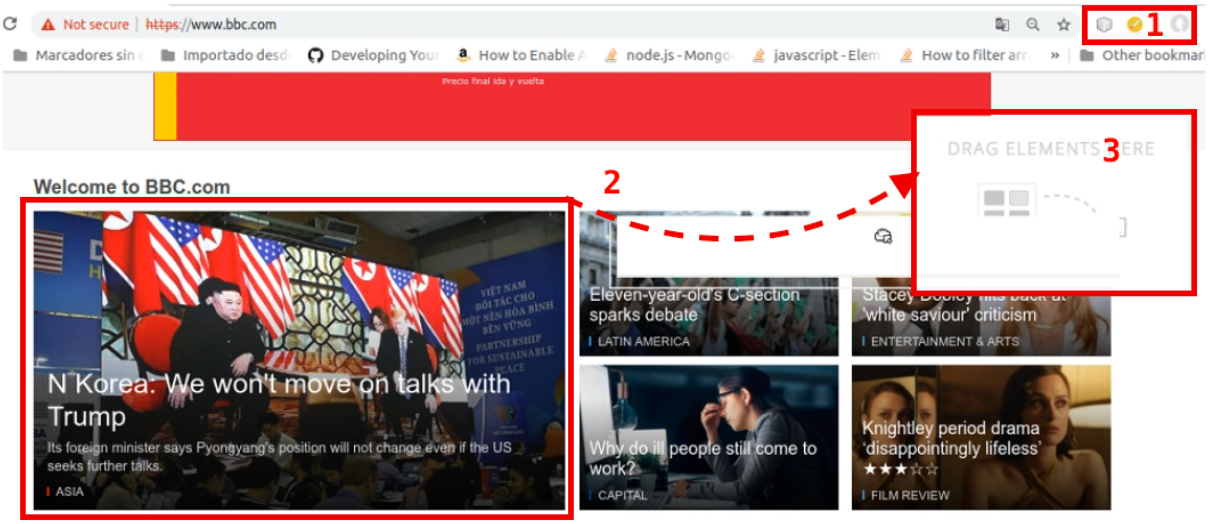

Fig. 3. web content selection for defining a content block

Once got a DOM element, the annotation process begins by adding some semantic features obtained from the sub-DOM parts, as shown in Fig. 4. Fig. 5 shows this in detail, where the confirmation for the title property can be appreciated.

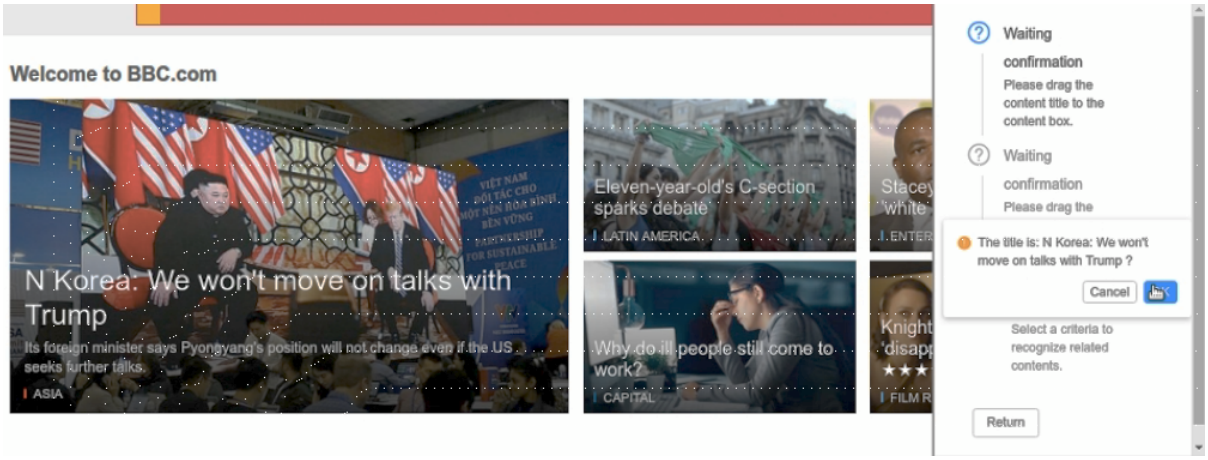

Fig. 4. web content definition

The process continues (as shown in Fig. 6) by asking the user if should include related content to the element previously selected (sibling elements). Fig. 6 also shows that the tool automatically detects the navigation links for the selected DOM element, which allows the user to consider the navigation feature (see "Can navigate?" checkbox in Fig. 7). Fig. 7 shows an editing form for other mandatory properties, such as a category for grouping similar content blocks and the identifier name for the content block.

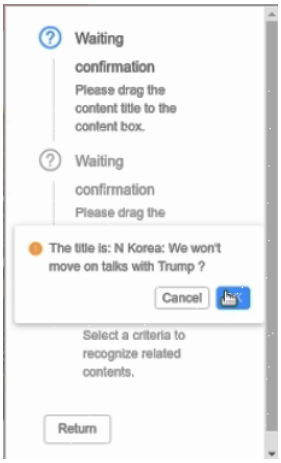

Fig. 5. Semantic attributes edition

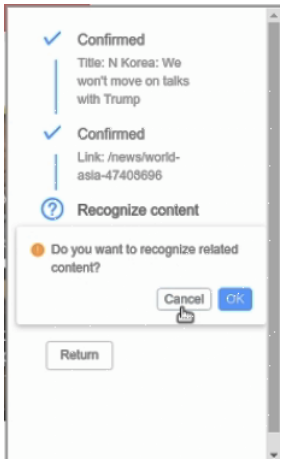

Fig. 6. Related content edition

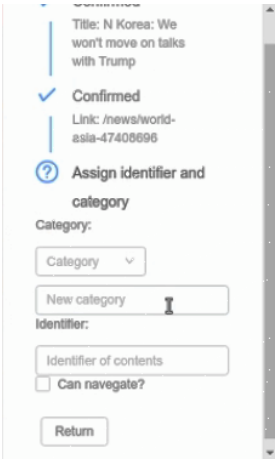

Fig. 7. Final block edition

After the user confirms the creation of this content block, the tool stores it and offers the option to open the VUI editor, as explained later. Alternatively, the user can continue creating content blocks from the same or any other web page.

The VUI editor, shown in Fig. 8, is also deployed as part of the same web browser extension and has access to the content blocks previously defined by the user. The main idea is that content blocks can be dragged and dropped onto the diagram editor's canvas. In this example, the selected content block represents all sibling elements.

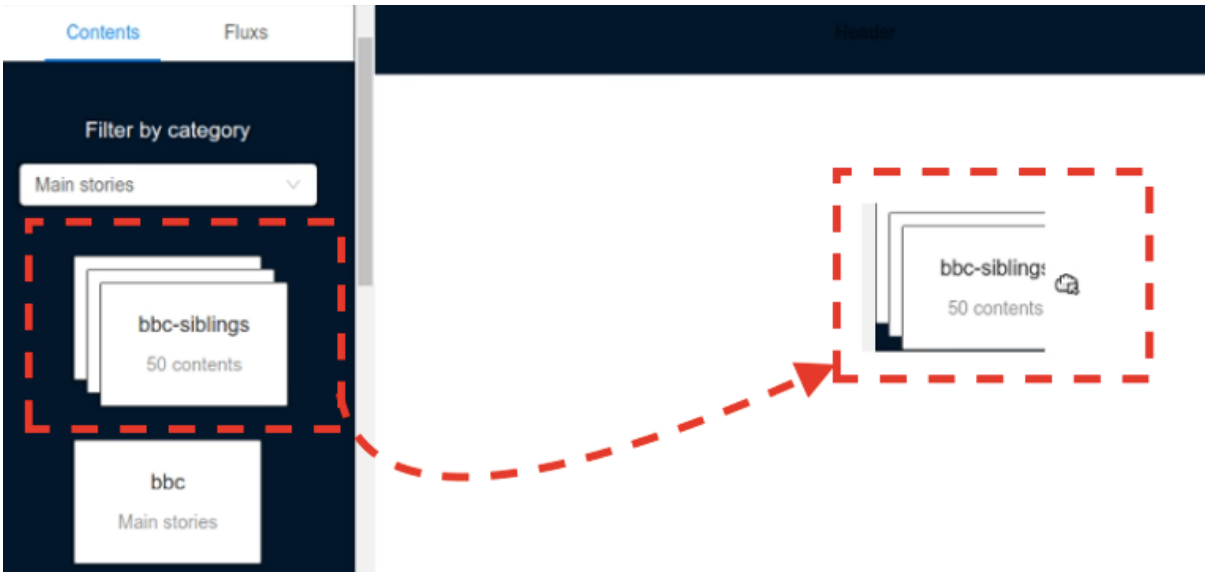

Fig. 8. VUI definition process

After adding several more blocks to the canvas and establishing links between them, the VUI model looks like the one shown in Fig. 9. In this case, the design consists of several content blocks representing the main news from different portals.

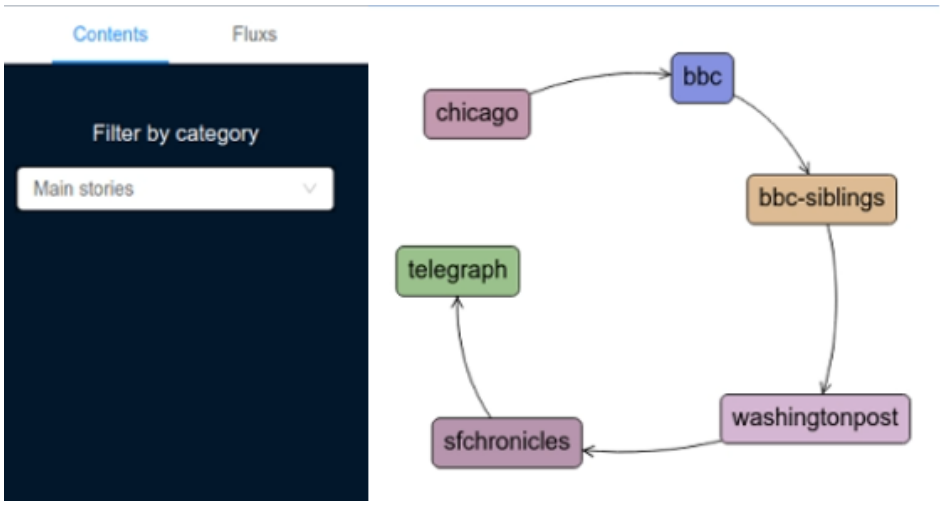

Fig. 9. VUI composed by several main news contents

A sample interaction with this VUI is as follows:

- User voice command: “What are the main stories.”
- VUI: “Chicago police sergeant alleges commander sent officers to ... do you want to listen to the rest of it?” (This response reads the title of the first content from the VUI and then asks the user if he wants to listen to the rest of it.)
- User voice command: “No.”
- “N Korea: We won’t move on talks with Trump... do you want to listen more about this?”
- ...

In the case above, despite the selected content blocks sharing the same category “main stories”, the approach has the capability to filter content blocks according to the category defined for each of them individually, regardless if they are part of the same VUI built or belong to different VUIs. A sample interaction considering this functionality, and according to the news appearing in Fig. 3 is as follows:

- User voice command: “What are the film reviews?”
- VUI: “Knightley period drama ‘disappointingly lifeless’ ... do you want to listen to the rest of it?” (This response reads the title of one of the content blocks defined by the user with the category “film reviews” and then asks the user if he wants to listen to the rest of it.)
- User voice command: “Yes.”
- “The World War Two period drama is ‘pretty to look at, easy to watch, but disappointingly lifeless,’ writes Caryn James... do you want to listen more about this?”
- ...

4.2.2 Weather portal. A second case relies on existing and frequently used skills, such as those in the weather domain. This skill is defined to provide the temperature and humidity in Buenos Aires when the user says: "Weather in Buenos Aires."

Fig. 10. web page highlighting web contents

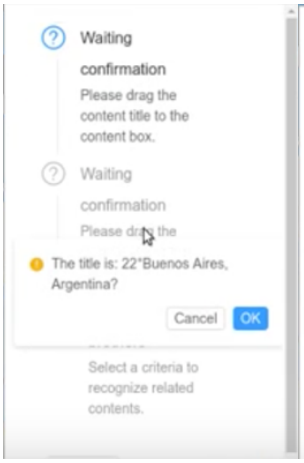

Fig. 11. Temperature block

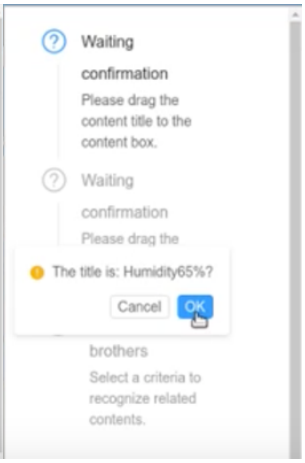

Fig. 12. Humidity block

The skill developed uses information from the "weather.com" website. Figures 10, 11 and 12 show the content block definition process. Two different content blocks are defined: one for getting the temperature and another for obtaining the humidity percentage.

Fig. 13 shows the flow chart created with these content blocks. A possible conversation excerpt is as follows:

- User voice command: "Buenos Aires weather."
- VUI: "The temperature is 22°C in Buenos Aires, Argentina. Humidity is at 65

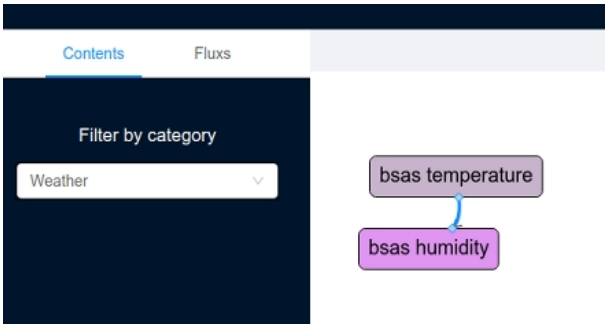

Fig. 13. VUI definition for the weather skill

4.2.3 IMDB coming soon. In this case, we show how content blocks may be specified using a semantic extraction template defined by the user. The example relies on the "Coming Soon" section of the IMDB portal. Since this section presents a navigable list of movies, the semantic properties can be extracted either from the list itself or the target movie web page. The user must select the "Create semantic content" option. After giving an identifier

name to the content block, the user can define several properties by clicking on a DOM element of the web page (Fig. 14). When the user clicks on an element, the tool automatically recognizes the selector that identifies that element in the DOM structure and asks the user to give a property name.

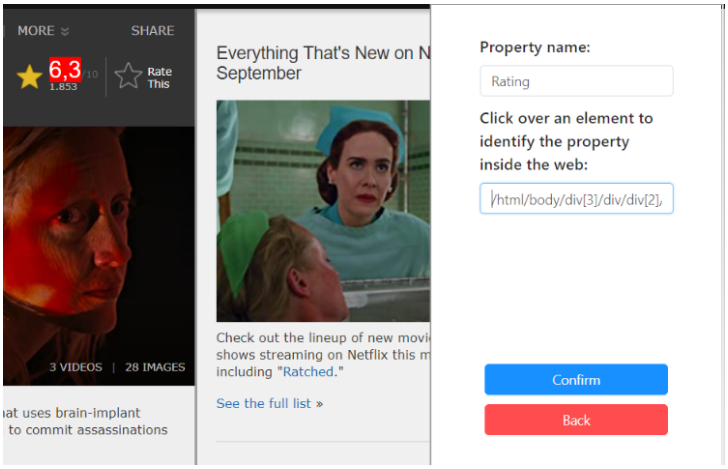

Fig. 14. Selection and definition of a new semantic property

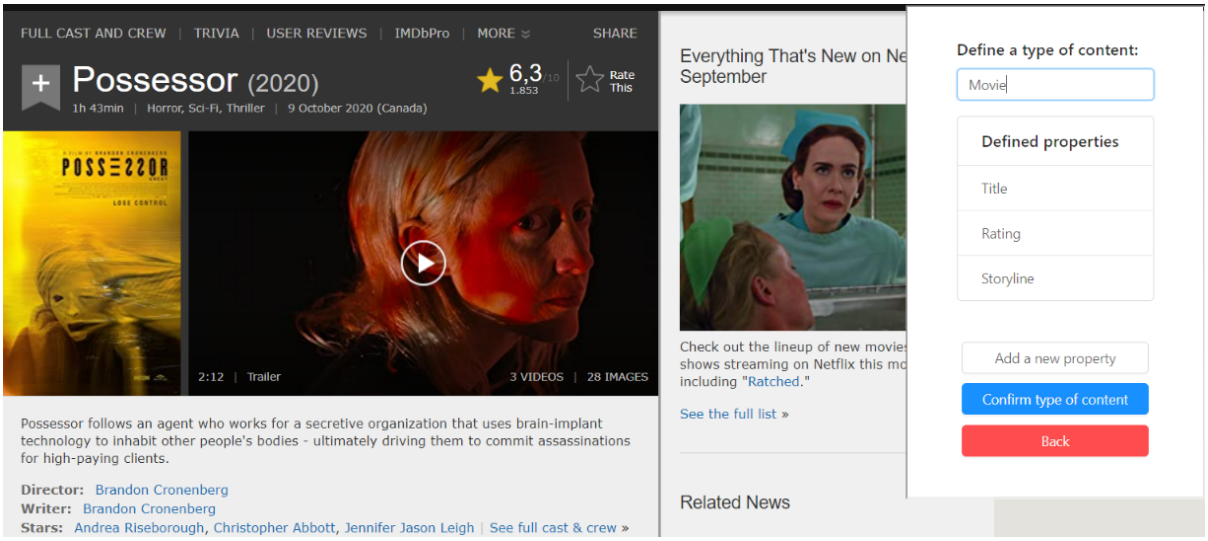

Fig. 15. Definition of content block with semantic structure

In this example, the user decided to define the properties Title, Rating, and Storyline from the Possessor movie web page, as Fig. 15 shows.

As we mentioned early, the VUI editor has access to the content blocks defined by the user (in this case, the VUI model uses the "coming soon" movies, as Fig. 16 shows).

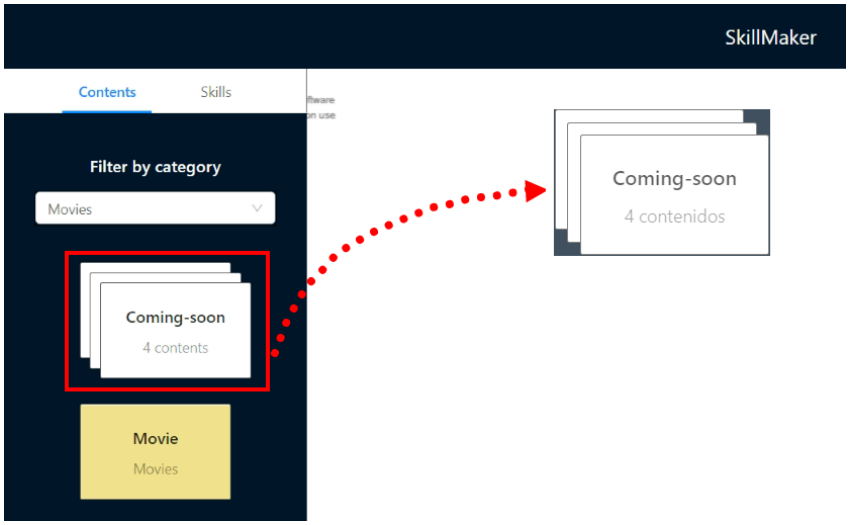

Fig. 16. Differentiation between content blocks defined with predefined properties and with some semantic properties

A sample interaction with this VUI is as follows:

- User voice command: “What movies coming soon?”
- VUI: “Possessor (2020)... do you want to listen to more info related to it? You can ask for the rating and storyline.” (This response reads the title of one of the content blocks from the VUI and then asks the user if he wants to listen to more info related to it, according to the semantic extraction template defined by the user.)
- User voice command: “Storyline.”
- “Possessor follows an agent who works for... do you want to listen more info about this movie?”
- ...

Our solution offers flexibility through the ability to manually edit how content blocks are read, replacing general rules, referred to as the "INTERACTIONTYPE" property in the model. As shown in Fig. 17 the properties Title, Rating, and Storyline (from Fig. 15) are the available options to read in the content blocks. The properties for reading may vary depending on the node the user is setting up.

Fig. 18 shows the form to set up the connections between the content blocks, referred to as the “EDGEINTERACTIONTYPE” property in the model. The options available include: reading a specific text before starting with the content block, and deciding whether to read the incoming content block directly or through user confirmation.

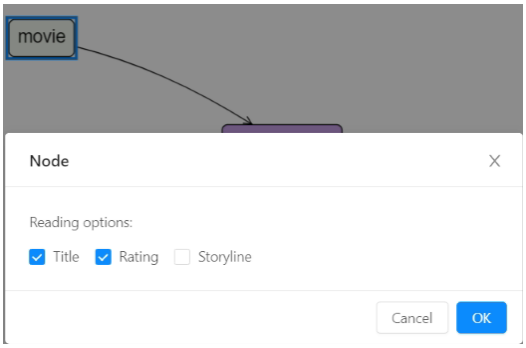

Fig. 17. Editing VUI options for a node

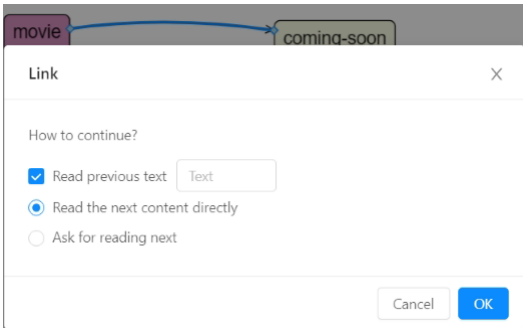

Fig. 18. Editing VUI options for a link

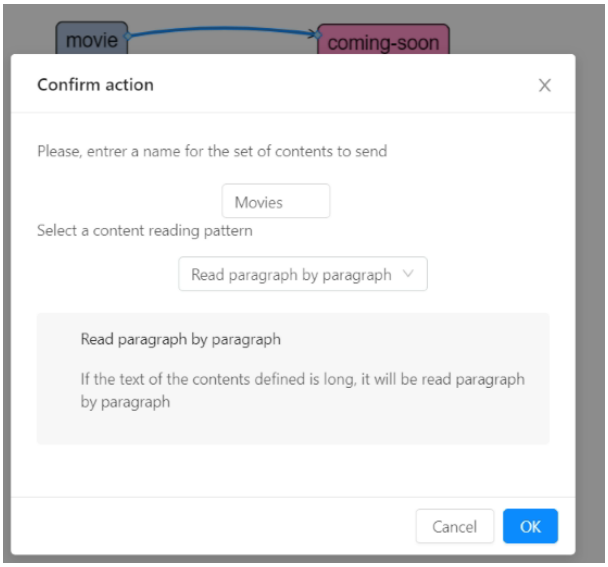

Fig. 19. Setting pattern general rules

When the user clicks on the "Deploy to skill" button, a new modal window (Fig. 19) appears, prompting the user to provide: 1) a name for the skill, which will serve as the voice command for this flow, and 2) the content reading pattern used for all the skill contents.

5 EVALUATION

We conducted an experiment to compare the SkillMaker app user experience with the experience using native apps based on Alexa or Google Assistant SDKs, seeking to determine whether or not there is a significant user experience difference between them. In this section, we report the experiment that aims to answer the following research question: is the user experience equivalent for Native and SkillMaker apps?

The main purpose of SkillMaker is to enable users to define new apps that are not available in any store of smart speakers. However, for this experiment, we decided to mimic existing apps in the store using SkillMaker. This

way, we can verify whether or not the SkillMaker app is similar to a native one that meets the same requirements. The goal of this experiment was not to evaluate the VUI specification process using SkillMaker, but rather to see if SkillMaker apps could provide a similar user experience to native apps.

In the following section, we describe the key elements of the evaluation and present the preliminary results based on guidelines for reporting empirical evaluations by Wohlin [29] and Juristo[30].

5.1 Goal, Research Questions, Hypotheses, and Variables

This experiment aims to *analyze and compare the user experience with SkillMaker and Native apps to determine whether they are equivalent. It was conducted from the perspective of researchers in the context of end users.*.

For the analysis of the research question, we used the System Usability Scale (SUS), a form-based questionnaire containing 10 Likert-style questions. The SUS is a widely used measure of perceived usability, providing a single score that reflects users' overall impression of the app's usability.

We selected a news app and a weather app as case studies and tested native and SkillMaker versions on an Amazon Echo Dot smart speaker to compare the usability of apps targeted at different business domains. For the weather apps, users can ask for the current weather in various cities, as described in Section 4.2. The native version can be found at [7], while the SkillMaker version was discussed in Section 4.2.2.

The news app allows users to inquire about the latest news and guides them through titles and content. The native version can be found at [8], while the SkillMaker version was covered in Section 4.2.1.

5.2 Experiment design

For the experiment, we designed a completely randomized study where subjects were asked to perform a list of tasks on either the native or SkillMaker version of the app.

To begin, we conducted a warm-up phase to familiarize the subjects with the device and the basic functionality of the Alexa service. They were provided with a list of commands to practice with, such as asking for a cooking recipe or searching for information on a famous person using Wikipedia. This way, we allowed them to gain some experience with the technology before starting the experiment. In the next step, subjects interacted with the news and weather apps, using either the native or SkillMaker version, as determined by the randomization process.

We recorded the users' progress to measure the time it took to complete the tasks. Once finished, they must fill out the SUS using Google Forms.

In an experiment, the factor [30] is the provoked variation controlled by the researchers to measure its consequences, and the possible factor values are named treatments. In this study, we controlled the variables or factors (the apps and app development approach) and randomly assigned users to use either the native or the SkillMaker version of the news or weather apps as the treatment. This type of experimental design is called a Complete Randomized Design. The subjects were randomly divided into groups and assigned to use either the native version (control) or SkillMaker (treatment) version of each app (factor). The use of the native or SkillMaker app is the factor tested in the experiment.

To ensure that we have a high level of confidence in the results of our experiment, we need to have a sufficient number of subjects. According to [31], a sample size of nine (9) participants is typically enough to identify 95% of UI issues with an occurrence likelihood of 0.3.

However, as the SkillMaker app is still in its alpha phase, i.e. is still under development, we decided to double the UI bug ratio to .15. To determine the necessary sample size, we rely on the binomial probability theorem [31] $1 - (1 - p)^n$, which calculates the probability for a given problem p occurring in a sample of size n. Based

[7] https://skills-store.amazon.es/deeplink/dp/B07F8WRCN1?deviceType=app&share&refSuffix=ss_copy
[8] https://skills-store.amazon.es/deeplink/dp/B07Z9H2Y5L?deviceType=app&share&refSuffix=ss_copy

on this, we determined that we needed a sample of 20 participants to identify 95% of bugs with an occurrence likelihood of 0.15.

5.3 Experiment subjects

We selected 20 subjects from Argentina for this experiment, including professionals, scientists, students, and government employees. The gender distribution was balanced, with 50% female and 50% male subjects, as shown in Figure 25 inside the Appendix section B.

The group of participants had a range of ages from young adults to seniors (0-23 yo 5%, 23-36 yo 55%, 49-62 yo 35%, and 62-79 yo 5%) and had various levels of education (Secondary 5%, Technician 10%, Undergraduate 30%, Bachelor or superior 55%), as shown in Figure 26.

Figure 27 shows their fields of study (Natural Sciences 5%, Languages 5%, Economic Sciences 5%, Social Sciences 10%, Medicine 15%, General Education 15%, Computer Sciences 20% and Exact Sciences 25%) and the current work activity (Teacher 10%, Student 15%, Retired 20% and Industry 55%).

5.4 Experiment objects

Based on the research questions, we delivered different experimental objects to the subjects:

- A list of warm-up tasks to get familiar with the conversation device.
- A list of activities to perform on the apps under analysis. The activities don't vary according to the app version assigned (native or SkillMaker), so they are the same for everyone.
- A final SUS questionnaire to capture the user assessment of the experience after the evaluation ends. The questionnaire answers are available in A.

In Figure 20, we show an overall schema of the experiment introducing experimental units, metrics, and data sources used during the analysis. The subjects were assigned randomly to one of four possible combinations of Weather and New apps: (Weather-native, News-native), (Weather-native, News-SkillMaker), (Weather-SkillMaker, News-native), and (Weather-SkillMaker, News-SkillMaker). We carefully assigned these combinations so that no uninteresting variable be more influential than another to all alternatives of the factor under examination.

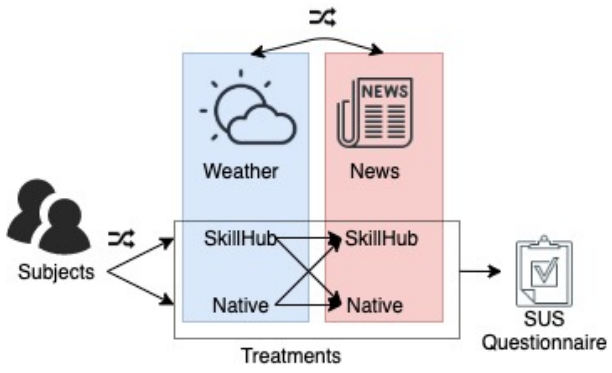

Fig. 20. Overall schema of experiment

5.5 Experiment execution

The experiment protocol was executed the same way by the 20 subjects, end users without experience with the Alexa service. The participants had to complete an experience survey, read the experiment description,

Statistics

Variable	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median	Q3	Maximum
Native news app	10	0	87,00	4,73	14,94	55,00	78,75	95,00	97,50	100,00
SkillHub news app	10	0	88,50	2,39	7,56	77,50	81,88	87,50	94,38	100,00
SkillHub Weather app	10	0	87,75	4,14	13,09	55,00	83,75	91,25	95,63	100,00
Native Weather app	10	0	87,75	3,32	10,50	67,50	81,25	88,75	98,13	100,00

Table 1. Samples descriptive information

study the requirements, and fill out the questionnaire form that asked them to perform some trace tasks. The experiment was carried out using the Amazon Echo Dot device. It was conducted in a controlled environment, with researchers supervising the execution to ensure consistent conditions (facilities layout, infrastructure, and subject isolation) and to prevent any changes to the subjects' responses during the experiment.

5.6 Data processing and reduction

We used both automated and manual processing techniques to analyze the collected results. The data processing phase used the survey responses, including SUS questions and additional feedback, as input (see A).

We used the Minitab tool (version 19.2020.1.0) to analyze the samples obtained for each treatment. The result of this processing was a statistical analysis of the samples using equivalence testing techniques, presented in the next section 5.7.

5.7 Analysis

In Table 1 we present the descriptive information (including mean minimum, maximum, st dev, etc.) for the four app versions: Native news app, SkillMaker news app, SkillMaker Weather app, and Native weather app. The means are similar, from 87, 00 to 88, 50 which denotes a minimal difference in the SUS score.

Based on the mean similarity, we performed equivalence testing. Equivalence testing is a variation of the null hypothesis significance test used to test the difference against two bounds (a range) rather than a value (e.g., 0). The equivalence testing is based on two one-sided tests (TOST) that reject the observation of a Smallest Effect Size of Interest (SESOI).

We show the boxplot charts corresponding to the native and SkillMaker versions of the news app in Fig. 21. Also, we can notice that the native version presents SUS scores scattered in a wider range of values ($55 - 100$) than the SkillMaker version ($77, 50 - 100$).

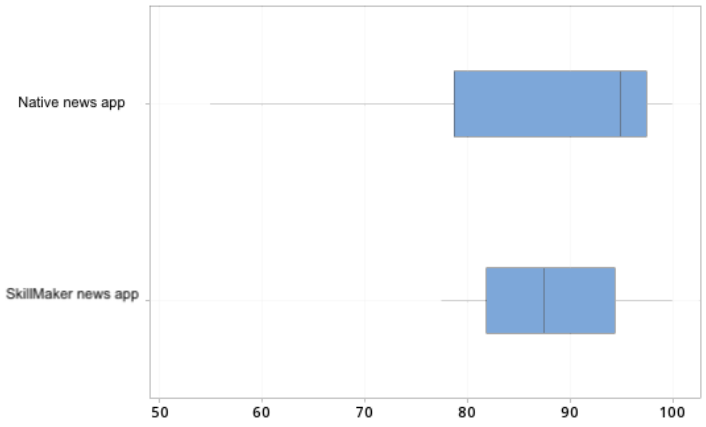

Fig. 21. News App boxplots

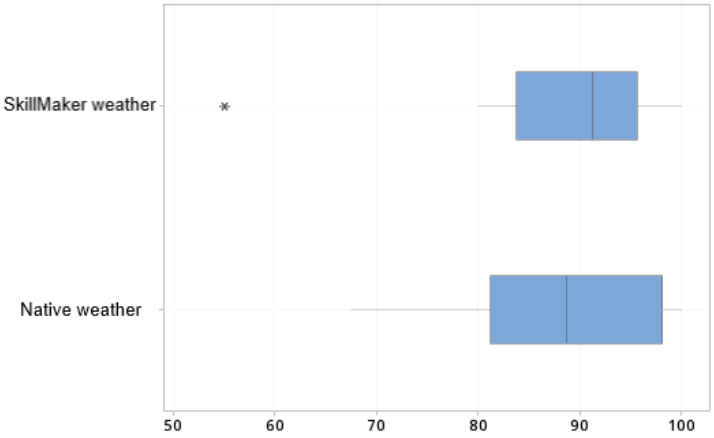

Fig. 22. Weather App boxplots

In Fig. 22, we show the boxplot charts corresponding to the weather app versions. Although there is a lower outlier in the SkillMaker samples, we also see that the SkillMaker version has clustered SUS scores ($80 - 100$) rather than the wider range of values reported by the native version ($67, 50 - 100$).

To answer the research question introduced above, we consider as null hypothesis (H_0) the difference is lower than -11 or greater than 8. As alternative hypothesis H_a, the difference is between -11 and 8 with a standard confidence level (α) of 0.05. For the weather app, the hypotheses testing results in a p-value of 0,048 for Difference ≥ 8 (DF 13, T-value -1,7936) and the same p-value of 0,048 for Difference ≤ -11. We can claim that is an equivalence because the greater of the two P-Values is 0,048. On the other hand, the news app showed that both samples are equivalent using a similar hypothesis testing approach. In this case, the null hypothesis (H_0) is the difference lower than -9.5 or greater than 9.5. As alternative hypothesis H_a, we study the difference between -9.5 and 9.5. The p-value for difference ≥ 9.5 and difference ≤ -9.5 was 0,046 in both cases. Therefore we can claim equivalence. Figures 23 and 24 show the equivalence charts for weather and news apps, respectively.

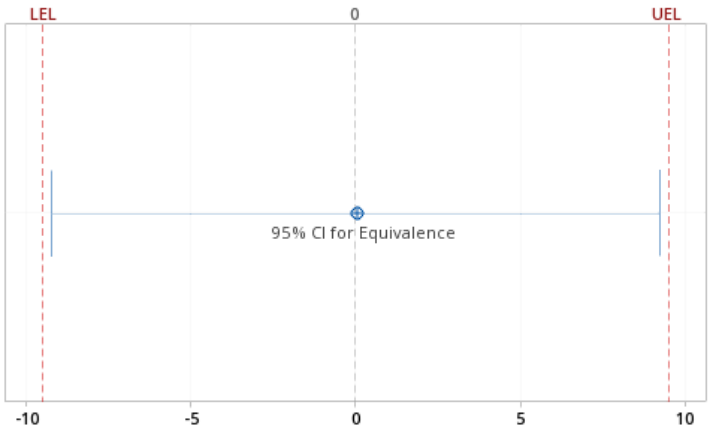

Fig. 23. Weather App equivalence

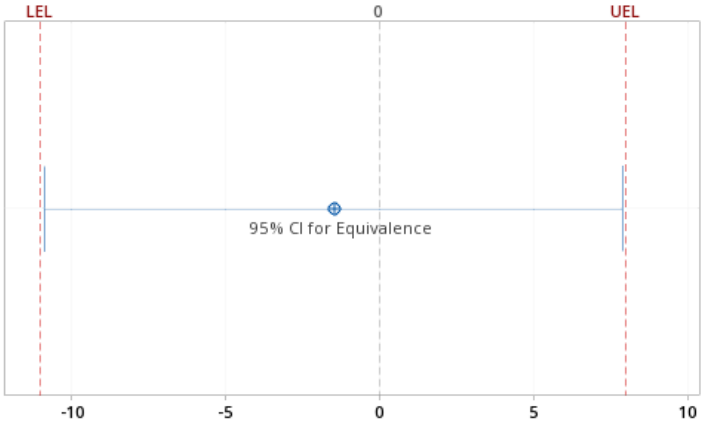

Fig. 24. News App equivalence

Overall, our results suggest that SkillMaker is a viable tool for developing smart-speaker apps with a user experience similar to native solutions. As the sample size is small for a non-parametric test ($n < 20$), the current outcome is preliminary, and further evaluations with a larger group of subjects are required. Regarding the effect size, Cohen's $d = 0.127$ for the news app and 0 for the weather app. This small effect ($d < 0.2$) is expected for equivalence testing as there is no difference in means [32].

6 CONCLUSIONS AND FUTURE WORK

Conversational user interfaces are increasing their presence in users' daily life. In particular, voice-based conversational user interfaces are massively enabled because of the surfacing of new kinds of devices such as smart speakers; despite these devices having been in the market for a few years already play an important role. Despite the process in which web app owners started to migrate or support this kind of interaction for new devices and interaction contexts, there is a huge gap between what users can already do with voice interfaces and regular

access to varied and arbitrary web contents. In fact, many web apps are not still supporting this new kind of interaction. Beyond this, although this gap may shrink soon (for instance, by using artificial intelligence), there are still challenges in personalizing and adapting VUIs, as has been the case with other types of interfaces.

We believe there is a need for an approach to creating VUI apps based on existing web content. In this regard, we have presented an approach based purely on new voice-based virtual assistants, combining old ideas like PIAs with the new kind of requirements around conversational interfaces. The nature of existing approaches makes them very different since our approach involves the design of the conversational interface as a separated model created by users, while for example, the approach proposed by Baez et al. [20] focuses on a conceptual architecture for automatizing the creation of these interfaces. In this regard, our approach allows combining web content from multiple sources because it separates web content access and retrieval from the voice user interface design. This is a remarkable difference with several mentioned approaches that use a complete website as the input of an automatic process and implies web sites' developers' intervention.

In this paper, we presented SkillMaker, a tool that helps address these challenges and create effective VUI apps. The development environment includes tools for defining content blocks (extracting web content) and creating VUIs (organizing them into flow charts that will later be interpreted for answering voice commands).

We conducted an evaluation of our approach by comparing the apps generated with SkillMaker, to the equivalent native apps. The results showed that VUI apps produced with SkillMaker had a similar SUS score to the native apps, with a slightly better result in one of the SkillMaker apps. This proves that SkillMaker has sufficient expressivity to create effective VUI apps without compromising the user experience, despite underlying technical aspects. Although we have not evaluated SkillMaker to provide evidence of the effort required for creating apps, we strongly believe that SkillMaker is more convenient and easier to use than native apps, especially for those users with limited experience. In this sense, we plan to perform new experiments to understand how end users may use our environment. Also, we plan to incorporate new features to SkillMaker, convert it into an end-user development approach, and measure if end users can create their apps.

In the future, we plan to study the use of SkillMaker for creating other types of conversational interfaces, such as chatbots, and explore more automated mechanisms for creating conversational interfaces.

REFERENCES

[1] César González-Mora, Irene Garrigós, Sven Casteleyn, and Sergio Firmenich. A web augmentation framework for accessibility based on voice interaction. In Mária Bieliková, Tommi Mikkonen, and Cesare Pautasso, editors, *Web Engineering - 20th International Conference, ICWE 2020, Helsinki, Finland, June 9-12, 2020, roceedings*, volume 12128 of *Lecture Notes in Computer Science*, pages 547–550. Springer, 2020.

[2] Raphael Menges, Chandan Kumar, Daniel Müller, and Korok Sengupta. Gazetheweb: A gaze-controlled web browser. In Chris Bailey and Voula Gkatzidou, editors, *Proceedings of the 14th Web for All Conference, W4A 2017: The Future of Accessible Work, Perth, Western Australia, Australia, April 2-4, 2017*, pages 25:1–25:2. ACM, 2017.

[3] Ryen W. White. Skill discovery in virtual assistants. *Commun. ACM*, 61(11):106–113, 2018.

[4] Anoja Rajalakshmi and Hamid Shahnasser. Internet of things using node-red and alexa. In *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*, pages 1–4. IEEE, 2017.

[5] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice.* Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2012.

[6] Nadia Elouali, José Rouillard, Xavier Le Pallec, and Jean-Claude Tarby. Multimodal interaction: a survey from model driven engineering and mobile perspectives. *J. Multimodal User Interfaces*, 7(4):351–370, 2013.

[7] Martin Porcheron, Joel E. Fischer, Stuart Reeves, and Sarah Sharples. Voice interfaces in everyday life. In Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox, editors, *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, page 640. ACM, 2018.

[8] Davide Spallazzo, Martina Sciannamè, and Mauro Ceconello. The domestic shape of ai: a reflection on virtual assistants. In *11th Proceedings of Design and Semantics of Form and Movement International Conference (DeSForM) MIT Boston*, pages 52–59, 2019.

[9] Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, Michael Gamon, and Mark J. Encarnación. Building natural language interfaces to web apis. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric

Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 177–186. ACM, 2017.

[10] Frank Bentley, Chris Luvogt, Max Silverman, Rushani Wirasinghe, Brooke White, and Danielle M. Lottridge. Understanding the long-term use of smart speaker assistants. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(3):91:1–91:24, 2018.

[11] Evelyn Kurniawati, Luca Celetto, Nicola Capovilla, and Sapna George. Personalized voice command systems in multi modal user interface. In *2012 IEEE International Conference on Emerging Signal Processing Applications, ESPA 2012, Las Vegas, NV, USA, January 12-14, 2012*, pages 45–47. IEEE, 2012.

[12] Aadesh Gandhre, Prakash Santhanagopalan, Prabhdeep Singh, Darshan Ramavat, I Ramakrishnan, and Hasan Davulcu. Creating and managing personal information assistants via a web browser: The winagent experience. In *Workshop on Information Integration on the Web*, 2004.

[13] Yevgen Borodin. Automation of repetitive web browsing tasks with voice-enabled macros. In Simon Harper and Armando Barreto, editors, *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2008, Halifax, Nova Scotia, Canada, October 13-15, 2008*, pages 307–308. ACM, 2008.

[14] Yevgen Borodin, Faisal Ahmed, Muhammad Asiful Islam, Yury Puzis, Valentyn Melnyk, Song Feng, I. V. Ramakrishnan, and Glenn Dausch. Hearsay: a new generation context-driven multi-modal assistive web browser. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 1233–1236. ACM, 2010.

[15] César González-Mora, Irene Garrigós, Jose-Norberto Mazón, Sven Casteleyn, and Sergio Firmenich. Open data accessibility based on voice commands. In *International Conference on Web Engineering*, pages 456–463. Springer, 2021.

[16] Julia Cambre, Alex C. Williams, Afsaneh Razi, Ian Bicking, Abraham Wallin, Janice Y. Tsai, Chinmay Kulkarni, and Jofish Kaye. Firefox voice: An open and extensible voice assistant built upon the web. In Yoshifumi Kitamura, Aaron Quigley, Katherine Isbister, Takeo Igarashi, Pernille Bjørn, and Steven Mark Drucker, editors, *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama, Japan, May 8-13, 2021*, pages 250:1–250:18. ACM, 2021.

[17] Michael H. Fischer, Giovanni Campagna, Euirim Choi, and Monica S. Lam. Multi-modal end-user programming of web-based virtual assistant skills. *CoRR*, abs/2008.13510, 2020.

[18] Asbjørn Følstad, Theo B. Araujo, Effie Lai-Chong Law, Petter Bae Brandtzaeg, Symeon Papadopoulos, Lea Reis, Marcos Báez, Guy Laban, Patrick McAllister, Carolin Ischen, Rebecca Wald, Fabio Catania, Raphael Meyer von Wolff, Sebastian Hobert, and Ewa Luger. Future directions for chatbot research: an interdisciplinary research agenda. *Computing*, 103(12):2915–2942, 2021.

[19] Eric WT Ngai, Maggie CM Lee, Mei Luo, Patrick SL Chan, and Tenglu Liang. An intelligent knowledge-based chatbot for customer service. *Electronic Commerce Research and Applications*, 50:101098, 2021.

[20] Marcos Báez, Florian Daniel, and Fabio Casati. Conversational web interaction: Proposal of a dialog-based natural language interaction paradigm for the web. In Asbjørn Følstad, Theo B. Araujo, Symeon Papadopoulos, Effie Lai-Chong Law, Ole-Christoffer Granmo, Ewa Luger, and Petter Bae Brandtzæg, editors, *Chatbot Research and Design - Third International Workshop, CONVERSATIONS 2019, Amsterdam, The Netherlands, November 19-20, 2019, Revised Selected Papers*, volume 11970 of *Lecture Notes in Computer Science*, pages 94–110. Springer, 2019.

[21] Pietro Chittò, Marcos Báez, Florian Daniel, and Boualem Benatallah. Automatic generation of chatbots for conversational web browsing. In Gillian Dobbie, Ulrich Frank, Gerti Kappel, Stephen W. Liddle, and Heinrich C. Mayr, editors, *Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings*, volume 12400 of *Lecture Notes in Computer Science*, pages 239–249. Springer, 2020.

[22] Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. Web data extraction, applications and techniques: A survey. *Knowl. Based Syst.*, 70:301–323, 2014.

[23] Rohit Khare and Tantek Çelik. Microformats: a pragmatic path to the semantic web. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 865–866. ACM, 2006.

[24] Christian Bizer, Kai Eckert, Robert Meusel, Hannes Mühleisen, Michael Schuhmacher, and Johanna Völker. Deployment of rdfa, microdata, and microformats on the web - A quantitative analysis. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, volume 8219 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2013.

[25] Robert Ennals and Minos N. Garofalakis. Mashmaker: mashups for the masses. In Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 1116–1118. ACM, 2007.

[26] Sergio Firmenich, Gabriela Bosetti, Gustavo Rossi, and Marco Winckler. End-user software engineering for the personal web: poster. In Sebastián Uchitel, Alessandro Orso, and Martin P. Robillard, editors, *Proceedings of the 39th International Conference on Software*

Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017 - Companion Volume, pages 216–218. IEEE Computer Society, 2017.

[27] Gabriela Bosetti, Alex Tacuri, Ishaya Peni Gambo, Sergio Firmenich, Gustavo Rossi, Marco Winckler, and Alejandro Fernández. ANDES: an approach to embed search services on the web browser. *Comput. Stand. Interfaces*, 82:103633, 2022.

[28] Iñigo Aldalur, Marco Winckler, Oscar Díaz, and Philippe A. Palanque. Web augmentation as a promising technology for end user development. In Fabio Paternò and Volker Wulf, editors, *New Perspectives in End-User Development*, pages 433–459. Springer International Publishing, 2017.

[29] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*, volume 9783642290. 2012.

[30] Natalia Juristo and Ana M Moreno. *Basics of Software Engineering Experimentation*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[31] Robert A. Virzi. Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors*, 34(4):457–468, 1992.

[32] Daniël Lakens. Equivalence tests: A practical primer for t tests, correlations, and meta-analyses. *Social Psychological and Personality Science*, 8(4):355–362, 2017. PMID: 28736600.

A SURVEY SAMPLES

Next you can find the survey responses from 20 subjects related to the SUS questionnaire and additional feedback.

timestamp	I think that I would like to use this system frequently	I found the system unnecessarily complex	I thought the system was easy to use	I think that I would need the support of a technical person to be able to use this system	I found the various functions in this system were well integrated.	I thought there was too much inconsistency in this system	I would imagine that most people would learn to use this system very quickly	I found the system very cumbersome to use	I felt very confident using the system	I needed to learn a lot of things before I could get going with this system	Survey start timestamp	Survey end stamptime	Do you find hard to use the tool? why?	What other uses for the tool you find feasible?	Did you find any issue or bug when using la app?	Do you find any improvement oportunity for the tool?	Is there any feedback about the too you want to share?
11/24/2019 17:47:12	5	1	5	1	4	1	5	2	2	1	5:30:00 PM	5:40:00 PM	No, no lo considero dificil	integración con reconocimiento de imagen	No	Si, la adaptación a un lenguaje "de la calle"	Tiene muchas aplicaciones futuras, especialmente en procesos de enseñanza y
11/24/2019 18:01:28	4	1	4	2	5	1	4	1	4	1	5:50:00 PM	5:56:00 PM	No, se puede manejar fácilmente sin estar muy informado sobre el tema.	Enseñanza.	Si, en la pronunciación.	Si, en la anteriormente mencionada.	No.
11/24/2019 18:17:40	5	1	5	1	3	1	5	1	5	1	6:00:00 AM	6:12:00 AM	No	escuchar música	no, ninguno	no	no
11/24/2019 18:28:51	5	1	5	1	5	1	5	1	5	1	6:18:00 AM	6:24:00 AM	No	Ninguno	No	La forma de invocar podría ser más reducida	No
11/24/2019 18:37:35	5	1	5	1	5	1	5	1	5	1	6:26:00 PM	6:35:00 PM	No	Ninguno	Ninguno	Ninguna	Me pareció muy útil
11/24/2019 18:45:10	5	2	5	1	5	1	5	1	4	2	6:41:00 PM	6:50:00 PM	No	Ninguno	Ninguno	Ninguno	Me parece ideal para estar actualizado
11/24/2019 18:55:33	5	1	5	1	5	1	5	2	5	1	6:55:00 PM	7:03:00 PM	No, una vez que sabes que decir es realmente sencillo.	No sabría decir	No	No se me ocurre	Es una gran herramienta. Soy docente y poder corregir y escuchar las noticias al mismo tiempo sería algo sumamente útil para mi día a día.
11/24/2019 19:03:06	5	1	5	1	4	1	5	1	5	1	7:06:00 PM	7:10:00 PM	No es para nada difícil, sobre todo para quienes habituamos usar dispositivos.	Nada que agregar al lector de contenidos.	No detecte ningún problema.	No realmente.	Fue un experimento sencillo de realizar, no resultó aburrido ni extenso.
11/24/2019 19:30:24	5	1	5	1	5	1	5	1	5	1	7:27:00 PM	7:34:00 PM	No, para nada.	No se me ocurre en éste momento.	Ningún problema detectado durante el experimento.	No	Fue entretenido realizar el experimento, me gustaría poder utilizar la herramienta a diario.
11/24/2019 19:36:24	4	1	5	1	5	1	4	1	5	1	7:50:00 PM	7:59:00 PM	No	Ninguno	No	No	No
11/28/2019 11:29:59	4	1	5	1	4	2	5	2	4	1	11:13:00 AM	11:18:00 AM	No, porque trabajo en el sector de sistemas.	Poder organizar las noticias por temas y solicitar posibles alertas meteorológicas.	No.	No.	Me hubiese gustado probar más funcionalidades orientadas a situaciones del día a día.
11/28/2019 14:25:51	5	4	5	3	5	1	5	1	3	3	2:12:00 PM	2:16:00 PM	No es difícil usar la herramienta con un entrenamiento previo o una explicación básica del uso del dispositivo	Escuchar música	Ninguno	Se podría evitar algunos detalles extras (por ejemplo al informar el sitio desde donde se obtiene el contenido)	Me gustaría utilizarlo en la vida diaria, mientras realizo las actividades del hogar
11/28/2019 19:15:10	5	1	5	2	4	1	5	1	5	1	6:56:00 PM	7:01:00 PM	NO, me resultó sencillo y amigable	me gustaría consumir los contenidos seleccionando los temas de modo aleatorio	NO	ya lo mencioné en respuesta anterior	Es una herramienta innovadora y útil en orden de prioridades: mayores - con discapacidad visual - niños - el resto de la población
12/1/2019 21:19:05	3	2	4	1	4	1	4	1	5	2	9:05:00 PM	9:10:00 PM	No	Que me lea novelas de ficcion	No	No	
12/1/2019 21:19:08	4	1	5	2	4	2	5	4	5	2	9:10:00 PM	9:15:00 PM	No, no es difícil, fue práctica y consisa	Obtener información sin la necesidad de estar al lado de la pantalla	No	Si, el dialecto de alexa	No
12/1/2019 21:19:09	3	1	4	2	5	1	3	2	5	1	9:00:00 PM	9:05:00 PM	No	Obtener información sin la necesidad de escribir	No, ninguno	Si, el vocablo de Alexa	No, ninguno
12/2/2019 19:18:14	5	1	5	2	3	1	4	1	4	2	7:12:00 PM	7:18:00 PM	No, me resultó sencilla ya que la aplicacion misma te iba guiando	Obtener recetas de cocina	No, ninguno	No	Me parece genial para usarla mientras cocino o estoy manejando
12/2/2019 19:23:07	4	2	3	4	4	1	4	1	3	3	7:20:00 PM	7:28:00 PM	No, pero me gustaría tener más ayudas en cada paso	Obtener información de series y películas	No	Como comenté antes, se podrían brindar más ayudas	Me pareció muy util para mis actividades diarias
12/2/2019 19:27:52	4	1	5	1	4	1	4	1	4	1	7:30:00 PM	7:36:00 PM	No, al estar acostumbrado a usar la tecnologia me parecio sencilla	Poder hacer busquedas en google	No	La pronunciacion	No
12/2/2019 19:32:27	5	3	3	4	4	2	3	3	3	4	8:03:00 PM	8:10:00 PM	Al principio me costo entender lo que tenia que decir, pero despues me resulto mas facil	Que te pueda recordar las tareas del dia	No	Me costó diferenciar el texto de la noticia con las palabras que tenia que decir para continuar	Estuvo buena la experiencia, creo que la usaria bastante durante el dia

B EXPERIMENT SUBJECT CHARTS

Next, you can find the demographic data from the 20 subjects from the experiment.

Fig. 25. Genre

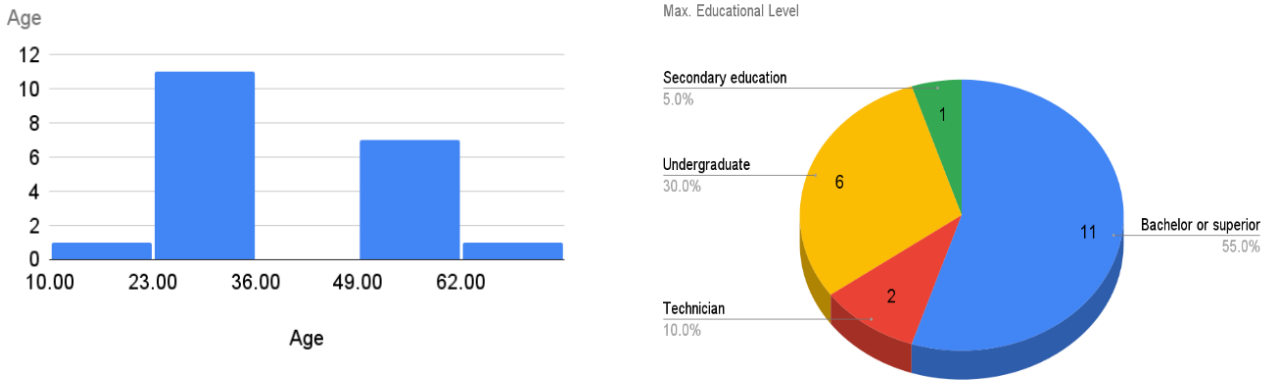

Fig. 26. Age and max educational level

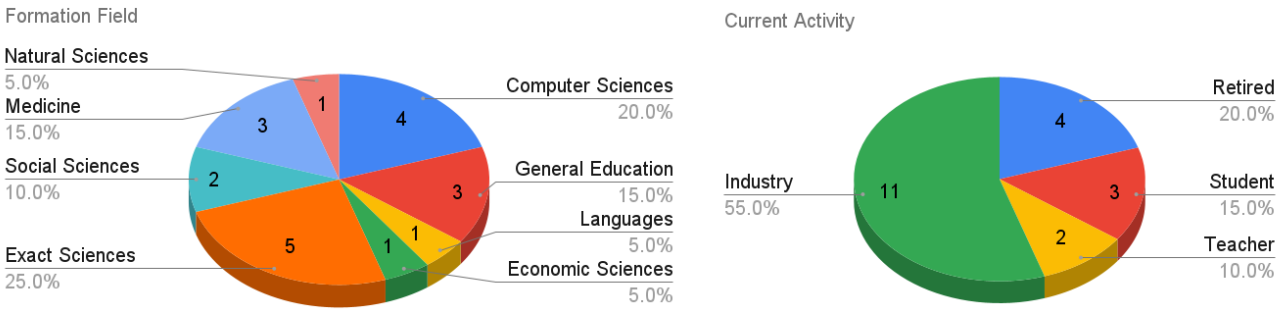

Fig. 27. Formation field and current activity