Conversational Web Interaction: Proposal of a Dialog-Based Natural Language Interaction Paradigm for the Web

Marcos Baez[1(✉)], Florian Daniel[2], and Fabio Casati[1,3]

[1] University of Trento, Trento, Italy
{baez,casati}@disi.unitn.it
[2] Politecnico di Milano, Milan, Italy
florian.daniel@polimi.it
[3] Tomsk Polytechnic University, Tomsk, Russia

Abstract. This paper lays the foundation for a new delivery paradigm for web-accessible content and functionality, i.e., conversational interaction. Instead of asking users to read text, click through links and type on the keyboard, the vision is to enable users to "speak to a website" and to obtain natural language, spoken feedback. The paper describes how state-of-the-art chatbot technology can enable a dialog between the user and the website, proposes a reference architecture for the automated inference of site-specific chatbots able to mediate between the user and the website, and discusses open challenges and research questions. The envisioned, bidirectional dialog paradigm advances current screen reader technology and aims to benefit both regular users in eyes-free usage scenarios as well as visually impaired users in everyday scenarios.

Keywords: Non-visual browsing · Screen readers · Conversational agents

1 Introduction

The current model of the Web is based on a set of basic concepts that evolved over the years to accommodate the complexity of new usage scenarios, but it essentially remains a visual experience: it requires looking, clicking, scrolling, hovering the mouse or performing gestures on a screen. We browse by focusing our attention on the elements that we deem relevant to our main goal. This interaction model does not translate well to visually impaired users and scenarios of true mobility where the visual attention we can provide is limited.

Despite the advancements in assistive technology and the large body of literature on improving the experience for this population [2,5,16,34,46], visually impaired users experience several challenges in accessing the Web [11,14,24,32,44]. The support provided by screen readers and non-visual browsers is essentially based on variations of sequentially reading a webpage from top to bottom, and

A. Følstad et al. (Eds.): CONVERSATIONS 2019, LNCS 11970, pp. 94–110, 2020.
https://doi.org/10.1007/978-3-030-39540-7_7

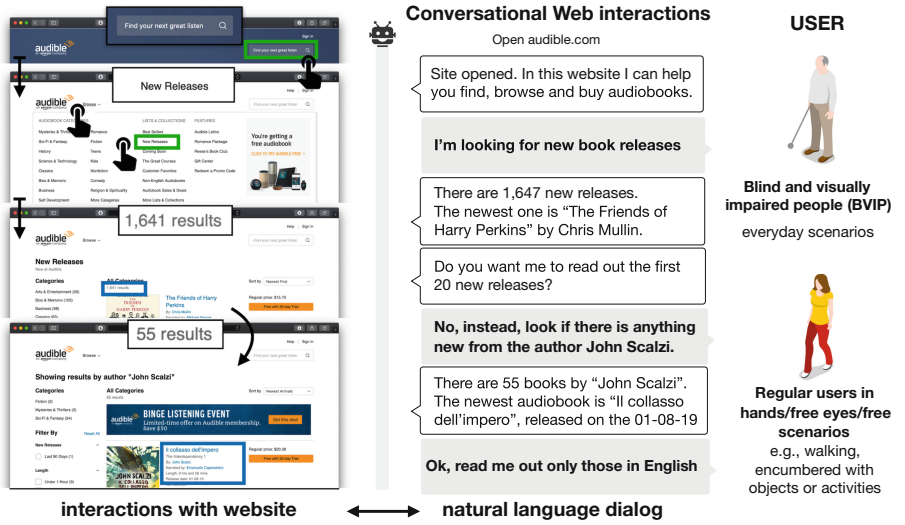

Fig. 1. Example conversational web interaction: the user engages in a dialog with a bot (right), which translates the user's requests into matching interactions with a website (left), enabling the user to browse the site in natural language (written or also spoken).

interacting with keyword shortcuts, which makes the process of translating user goals into a series of steps more time consuming and frustrating [32,37]. Adding to this, visually impaired users have to deal with poorly designed websites with no basic accessibility support [4,27], and the uncertainty of missing out information or not even knowing the source of the problems preventing them from completing their tasks [10]. Even when complying with standards and guidelines [18,46], the experience can be poor and frustrating [37]. In the end, accessibility does not necessarily mean usability.

The same can be said about mobility and attention-demanding scenarios that can lead to what is referred to as *situational impairments* [41], where contextual factors reduce our ability to interact visually. For example, using a mobile phone while walking not only reduces our visual, attention, and motor abilities [15], but can lead to distractions and risky behaviors [23,33]. A survey of more than 1000 adults in the US reported that 60% of the respondents use smartphones also while crossing the street, even if 70% of the total respondents perceived the behavior to be dangerous [25]. It is not surprising, thus, that injuries due to mobile phone distraction from 2004 to 2010 have almost tripled, mirroring that of driver's injuries for the same reason [33]. Browsing the Web while walking or driving are extreme cases but indicative of hands-free, eyes-free and minimal attention scenarios where traditional interaction models don't work.

In this paper we propose a new interaction paradigm for the Web that allows users to access websites by directly expressing their goals in dialog-based interactions with a conversational agent. To illustrate the concept, Fig. 1 shows an example of a goal-oriented conversation for browsing audiobooks, where user

requests in natural language are translated by a conversational agent into automated actions applied to a website. Our approach is based on the idea of opening up the Web to an entire new generation of agents (e.g., Amazon Alexa, Google Assistant, Telegram), which currently need custom-built bots and skills to deliver services to their users. To this end, we introduce a reference model and architecture to allow web developers and content producers to make their websites "bot-friendly" without the need for implementing custom conversation logic, training Natural Language Understanding/Generation (NLU/NLG) models and all the effort that goes into developing new bots.

In the following we give an overview of the current state of the art to then introduce our proposed model and architecture.

2 State of the Art

2.1 Web Accessibility and Models for Non Visual Browsing

Efforts in making the web more accessible span across the development of standards, design guidelines, algorithms and inclusive development processes.

The Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C) contributes with standards that are important in guiding the design of more accessible web pages and applications. Among these, the Web Content Accessibility Guidelines (WCAG) [46] provides recommendations for making content (and structure) accessible to people with disabilities, and are based on W3C technical specifications (e.g., HTML, CSS). The WAI also develops technical specifications such as Accessible Rich Internet Applications (WAI-ARIA) [17], which focus on making web applications – especially those with dynamic content and complex interactions – accessible to users of assistive technologies (e.g., screen readers). Implementing these standards is however in the hands of web developers and content producers.

In order to address the problems of compliance, a whole body of literature has been devoted to developing coping strategies. Automatic approaches have been proposed to facilitate navigation in the absence of the corresponding semantic annotations and structure [2,47] and generation of caption and audio descriptions for multimedia material [38,43]. Collaborative metadata authoring approaches [8,40,42] instead rely on volunteers to "fix" accessibility problems as they find them or upon request, and store the improvements in an external database that can be accessed by others. Other approaches aim to address the problem at the source, helping web developers be aware of accessibility during the design and development process [36].

The need for describing the content in web pages – and not only the structure – emerged with the development of rich snippets and machine processing of websites. Early approaches were based on microformats, with initiatives such as hCard, hCalendar and RDFa [21]. With the introduction of microdata support in HTML5, it became possible to describe the "items" in a page, using HTML properties to define the scope of the items, and key-value pairs to describe their attributes [39]. The absence of common vocabulary to refer to these items led

major search engines (Bing, Google, Yahoo) to the creation of Schema.org [31]. This initiative offers and maintains a common vocabulary (for the most common "items" and their attributes) that simplified the annotation of data in web pages.

The VoiceXML [35] standard by the W3C offers a markup language for voice applications that can reuse the infrastructure developed for the Web. However, it requires developers to build custom applications or custom "chatbots" using the provided markup language. SALT [45] on the other hand is a lightweight markup language created in response to VoiceXML, but that still requires the grammar and logic for handling conversations to be declared in the webpages.

The above model and extensions provide a great foundation for improving accessibility in non visual browsing scenarios. However, they were not designed with conversational agents in mind, and so lack the necessary ingredients to turn websites into a dialog with the user.

2.2 Non Visual Web Browsing

A significant improvement on the linear navigation of screen readers was the introduction of the notion of navigation context. The CSurf [29] browser approach was to capture the context of a link, using structural properties and surrounding content, to identify the most relevant component in the next page, thus avoiding reading out sequentially from the beginning. The same group also worked on HearSay [13], a browser that builds on the idea of contextual browsing, incorporating machine learning and natural language processing to segment and automatically label web components so as to ease the navigation through semantically related content. It also aims at improving web transaction by identifying actionable items (e.g., "add to cart," "checkout") in web pages and making them easily available to the user. With a different take on context, the Sasayaki [48] browser aims at improving on the traditional navigation with a screen reader, by providing supplemental information in the form of page overviews and spatial and contextual cues (called whispers) on a secondary audio channel.

Focusing on performance, other approaches explored speech optimisations and summarisation of web pages. Gerrero and Conçalves [20] experimented with faster text-to-speech and multiple speech channels to speed up the scanning of relevant content by visually impaired users. Experiments showed that faster and concurrent speech significantly improved scanning while maintaining the level of comprehension. Summarisation is another widely explored technique, used for example to create "gists" of web pages (to decide on whether to read sections of a page) [22] and to facilitate non visual skimming [1].

The idea of using spoken commands to interact with the browser has also been explored in the literature. The most recent proponent of this idea is Capti-Speak [5], a speech-enabled screen reader for web browsing that aims at lowering the complexity of managing shortcuts in navigating with screen readers. This solution enables users to utter commands in natural language, which are identified as browsing actions ("press the cart button", "move to the search box") and interpreted in the context of the ongoing dialog. A user study with 20 blind

participants showed that the approach was perceived as more usable and led to better browsing performance compared to a conventional screen reader.

Web automation is another area explored to improve the browsing experience. Voice-enabled macros for repetitive web-browsing have been built on top of the HearSay browser [12], allowing users to record and play their own macros. Bigham et al. [9] introduced the Trailblazer system, which focused on facilitating the process of creating web automation macros, by providing step by step suggestions based on CoScript [28]. The SeEbrowser [30] was designed to provide personalised browsing shortcuts to visually impaired users based on shortcuts useage statistics. Lau et al. [26] instead focused on facilitating the execution of macros, and proposed CoCo, a system for web automation that relies on a conversational interface based on (semi)structured language. The system simplifies the execution of repetitive tasks by allowing users to invoke CoScripter macros or "on the fly" tasks based on the user web history, in natural language.

Another interesting approach to automation is proposed by Ashok et al. [6], where the goal is to free users from performing low level interactions with HTML elements by adding a layer of abstraction on higher level web entities (e.g., "search results", "calendar", "menu", "forms"), and allowing users to execute navigation commands, invoke actions and answer queries. For example, instead of dealing with the complex interaction of a calendar widget, users would directly say "Choose return date 28" on the calendar entity, and the underlying interaction manager would try to execute the appropriate steps.

All of the infrastructure, guidelines and optimisations above were done with the traditional web agent in mind, and while they provide valuable contributions and inspiration for our work, the approaches are not suitable for the conversational web interaction we envisioned in this paper.

3 Conversational Web Interaction

Next, we elaborate on the idea of conversational web interaction, introduce the need for a dedicated conversational agent (a *bot* or *chatbot*), and identify the key requirements that drive our research.

3.1 Concept

Recalling the example conversation in Fig. 1, the interaction paradigm we propose in this paper is based on a bi-directional use of *natural language*. Users either write and read text in a chat window (*written* interaction) or talk and listen (*vocal* interaction). Instead of directly interacting with a target website inside the web browser, they interact with a *bot* that serves as mediator between the user and the website and is able to *translate* back and forth between natural language input/output and website interactions (e.g., reading text, filling form fields, navigating links). While the latter are generally the same across different websites, the bot automatically extracts the necessary *domain-specific vocabulary* and *knowledge about available features* from the target website, e.g.,

it learns that the site allows users to "buy an audio-book" or that it can "read an abstract." Upon a specific instruction by the user, such as "find books by John Grisham," the bot *autonomously interacts with the website on behalf of the user* and presents users only with the final results, preventing them from having to learn how to do so themselves.

Content and functionality of the website can be accessed in a *random access* fashion: just like in any conversation or in visual web browsing, the user can jump from one topic (e.g., reading the summary of a book) to another one (e.g., searching for music), without the need to leaf though or listen to potentially lengthy lists of options. This is different from conventional screen readers and the related works discussed previously, which depend on the structure of the website and require the user either to use the keyboard to walk through options or to utter structure-specific instructions like "go to search box," "enter XYZ," and "submit." In other words, the paradigm is *goal driven*, not structure driven.

3.2 Requirements

The goal of our research is to study how to enable conversational interactions on top of existing websites; the development of generic, stand-alone chatbots is out of the scope of this work. Enabling the described conversational interaction paradigm for the Web requires thus two fundamental ingredients, a *chatbot* mediating between the user and the website and a purposefully designed, *conversation-oriented annotation* of content and functions enabling the bot to get acquainted with the website. The bot provides basic, cross-application conversation support, while the annotation equips the bot with the necessary application-specific knowledge. The core requirements for the development of the bot and the annotation are:

1. *Orientation*: Given the URL of a website, the bot must be able to summarize the content and/or functionalities offered by the website, in order to allow the user to understand what the site offers and to provide for basic access structures. The role of the summary is similar to that of conventional menus or navigation bars in visual browsing. For instance, the bot in Fig. 1 tells the user that it is able to "find, browse and buy audiobooks."
2. *Vocabulary*: As exemplified further in the conversation, the bot should be able to speak the language of the target website. It should not understand and master only generic terminology (e.g., "navigate" or "click"), but also site-specific terminology (e.g., "find a book" or "leave a comment").
3. *Informational vs. transaction tasks*: The bot should not only render the content vocally or textually, which is the basic ingredient for the delivery of informational services (e.g., reading out loud the description of an audiobook – essentially reading content from websites). It must also be able to parse vocal/textual input and forward it to the website, enabling full-fledged, transactional services (e.g., searching for a given author – essentially interacting with websites).

4. *Intents and intent parameters*: The bot must be able to understand the user's intent when speaking or writing, so as to be able to enact suitable actions in response. Intents may be application-agnostic (e.g., fill a form field, read a title) or application-specific (e.g., buy an audiobook, write a review). Intents may further require parameters: searching a book is an intent; searching the book by author "John Scalzi" is an intent equipped with a suitable target.
5. *Action enactment*: As the bot mediates between the user and the website, enacting an action in response to an identified intent means interacting with the website on behalf of the user. The bot must thus be able to mimic user interactions with the website, such as navigating a link or filling a form.
6. *Feedback*: As the user is now interacting with a bot and not with the actual website, it is important that the bot be able to provide feedback on the outcome of actions. For example, a navigation action may fail or a form submission may succeed. These outcomes must be communicated back to the user for confirmation and orientation.
7. *Context*: Natural language conversations, if split into chunks of text, may be ambiguous. It is for instance common to claim that something was cited "out of context" and, hence, may be misleading. It is thus crucial that the bot be able to maintain conversational context and to disambiguate between similar utterances in function of context.
8. *Pro-activity*: Conversational agents are mostly reactive to user requests, yet sometimes pro-active behaviors may be needed. For instance, if content changes dynamically inside a page or if a user is to be guided through a form filling process, the initiative of the conversation may come from the bot instead of from the user. The bot may also pro-actively suggest possible next actions, like in our example.

Satisfying these requirements will require an interplay between the generic bot and the application-specific annotation. For example, only a proper annotation of an application-specific intent will allow the bot to know that it must not only identify the intent (e.g., search author) but also collect a respective parameter (the author name).

3.3 Design Principles

The annotation of the website assumes thus a central role in enabling the envisioned paradigm. In this respect, the assumption underlying our proposal is that, if needed, web pages can be extended with annotations, e.g., standard WAI-ARIA annotations [17] or custom bot annotations, to instruct the bot.

Equipping an existing web page with an annotation to enable conversational interactions implies of course an additional effort to the developer of the website (just like complying with accessibility in general). In order to keep the effort low and effective, the ideas we propose in this paper further build on the following simple design principles:

1. *No need for custom chatbot code.* The goal is to prevent web developers from having to master also chatbot development to equip their websites with

conversational interactions. Developers should thus not have to write any own line of chatbot code.
2. *Separation of conversation model from NLU.* Modern chatbots typically feature AI-powered natural language interpretation requiring proper training before use. We want to prevent developers from having to train themselves the bot and instead want to automate this task and ask developers just to provide the necessary domain-specific vocabulary.
3. *Presentation (NLG) managed by bot.* Developers should not have to worry about how to render outputs (dialog responses), be them vocal or written. The bot should fully take care of this task to harmonise the user experience.
4. *Web developers keep control of the experience.* While we think it is crucial to take over as many tasks as possible, it is however crucial that developers be in control of what happens. Through a sensible annotation of their sites they should be able to control which features of a site to equip with a conversational interface.
5. *Support for social botification.* For those cases in which developers do not provide suitable annotations, it should be possible for interested users to provide them externally and collaboratively and to share them with others for reuse and refinement.

Next we show how we intend to satisfy requirements and design principles with a concrete software architecture.

4 Approach

4.1 Conversation Model

The conversation model driving the envisioned chatbot is a refinement of the conventional *input-intent-action-response* model of modern conversational agents: The user provides an *input* (e.g., "Which audiobooks have been released this week?") that expresses an *intent* of action (e.g., "search for recent audiobooks"). The chatbot interprets the input and extracts the intent – this typically involves the use of a dedicated Natural Language Unit leveraging on AI – to match the intent to an action that allows the bot to meet the user's request (e.g., the bot could navigate to the page with the latest book releases and identify the most recent ones). The execution of this action produces an output that can be used by the bot to generate an informative *response* for the user (e.g., "Five new books have been released this week"). The response may or may not contain a solicitation for the user to provide further input (e.g., "Should I read out loud the titles of the books?").

The refinement consists of two core aspects: (i) the selection of a set of *reactive actions* that are specifically tailored to enabling a natural language conversation with websites and (ii) support for *pro-active actions* enabling the bot to take the initiative in conversations, e.g., in response to updated content inside a page or new content or features appearing in the page. Figure 2 graphically summarizes the key concepts of the target conversation model.

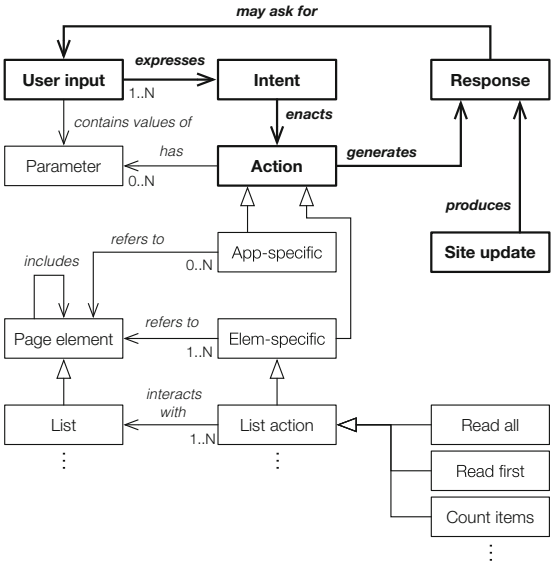

Fig. 2. Model of conversational web interaction. In bold the core conversation logic as interplay between user inputs, actions and responses. Actions are refined into application- and element-specific actions.

The reactive actions are split into two types: *Application-specific* actions express interactions with the website that allow the user to achieve application-specific goals, such as finding or buying an audiobook. *Element-specific* actions refer to presentation elements of a web page and allow the user to interact with the element. Typical elements are forms, form fields, buttons, paragraphs, headers, and similar (identified through suitable HTML tags). Element-specific actions are pre-defined and implemented once for all. Application-specific actions are configured by the developers of the site and refer too to presentation elements to indicate where in a page the respective action can be carried out.

Managing a dialog is now an iteration between application- and element-specific actions: Given an application-specific action, the bot moves the focus of its analysis to the respective presentation element (e.g., a form). Next, it analyzes the content of the element, identifies the presentation elements it comprises and enacts element-specific actions, involving the user when necessary.

4.2 Architecture

The software architecture we propose to reify the above conceptual model and to satisfy the requirements identified earlier is graphically summarized in Fig. 3. The architecture is split into a design phase (top part) and an execution phase (lower part). The key components are: A *botifier*, i.e., a component that is able to parse and analyze a given website identified by its URL and to extract orientation information (requirement R1) and the necessary conversational domain

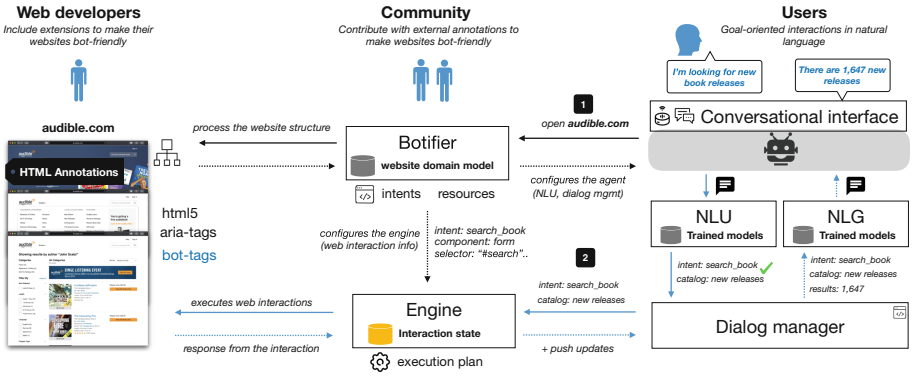

Fig. 3. Reference architecture for conversational web interaction. The number labels correspond to (1) the process of botifying web pages (design time) and (2) the management of user-website dialogs (runtime). In light blue the actors in the ecosystem. (Color figure online)

knowledge (R2). A possibly annotated *website* the user wants to interact with. A *conversational interface* (e.g., a chat window or an Amazon Echo device) with an NLU for input processing (R3, R4) and an NLG for output generation (R6, R8). A *dialog manager* able to manage the conversation with the user, to keep conversational context (R7), and to act on the website on behalf of the user using the *engine*, which implements the necessary logic to mimic user interactions (R5) and to monitor updates (R8) using a headless browser. We discuss website annotation, botification and conversation management in more detail below.

4.3 Content/Function Annotation

The challenge of bot development is training the bot with domain vocabulary and actions. We have seen that there are no ready models or annotations for doing so. The question is thus how to obtain the necessary information.

Microdata[1] allows one to annotate content in web pages with semantics, and initiatives such as Schema.org offer a common vocabulary for popular items on the Web. For example, search results in audible could be annotated with `itemtype=http://schema.org/Audiobook` to describe the type of entity. These annotations could unambiguously describe content and facilitate connecting user requests ("Who is the [narrator]?") to presentation elements. ARIA and HTML5 provide semantic annotations/tags and notifications or errors or updates when new, dynamic content appears. When annotating websites for accessibility, developers are thus already facilitating the automation of web interactions.

There are however no annotations or vocabularies for describing operations, let alone app-specific actions. For example, functionalities like "search", "play"

[1] https://html.spec.whatwg.org/multipage/microdata.html.

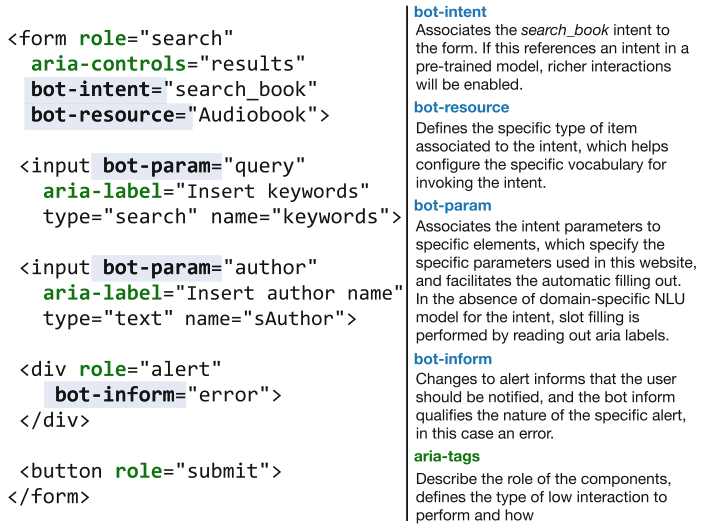

Fig. 4. Concept of joint use of standard ARIA and custom annotations for instructing the bot with domain knowledge (an intent with action).

and "buy" audiobooks are usually listed in menus, but are meant for visual, human consumption. It could thus be an option to introduce specific bot properties, such as "bot-intent" to identify HTML tags (e.g., form, input, link) that match user intents and that the bot can operate on behalf of the user.

Figure 4 provides a feeling of how different annotations could be used together to achieve the task.

4.4 Botification

The *botifier* parses a page's markup and annotations and derives a *domain model*, which configures the bot with the interactions the user can have with the website. Intents and parameters (`bot-intent` and `bot-param` in Fig. 4) define the requests the user can perform and the information that should be provided in order to do so. The vocabulary (named items and properties), in addition to the intents, configures the NLU with the language users can use to express goals.

Enacting user intents during the dialog is then the job of the *engine*. It, too, is configured by the botifier, which binds the execution of actions to information about the mapping between dialog concepts (items, intents and parameters) and the associated presentation components in the webpage. At runtime, the engine supports enacting actions associated to user intents, using the structured information provided by the NLU (e.g., action: "search_book", params: [author : "John Scalzi"]) and operating a headless browser.

While reading out the results of a search, the bot should be able to tell the user what to do with a specific book in that context at run-time (e.g., share, play a sample, go to details). In this respect, the state of the web page provides

context to the interactions that can be allowed during the conversation and that can help the users fulfill their intents. The engine should then be able to provide (and update) the bot with the state of allowed actions that can be used for dialog management.

While the botification process relies on developers providing the proper annotations, the approach can be extended to allow for collaborative metadata authoring efforts such as those seen in the web accessibility community [8,40,42]. In this case, volunteers would provide the domain model, possibly aided by tools and browsers extensions, and store them in a shared repository that can be consulted during botification.

4.5 Conversation Management

The actual *bot* is composed of the conversational interface, the NLU and NLG, and the dialog manger. It manages the bidirectional communication with the user. Natural language understanding is based on pre-trained models on two levels of abstraction: domain-specific and general models.

Domain-specific models could be trained on the most common transactions and queries in certain domains of applications and allow for highly expressive and natural interactions. Training such models would require identifying common, domain-specific intents and run data collection tasks (e.g., crowdsourced [7]) to build a training dataset. This is of course a costly enterprise, and would require efforts at the right level of abstraction, but we can see initiatives similar to Schema.org defining ontologies of intents, and data collection efforts based on volunteers, as a potential direction.

General models may offer a layer of abstraction on top of generic Web components and actions on websites. For example, a general model could be trained for browsing search results or lists of items, with intents such as *read*, *filter*, *sort*, *ask*, supporting general requests like "sort [items] by [param]". Training such general models requires less effort, but would provide for less expressive queries.

Once a user intent is recognized and validated by the NLU, the dialog manager forwards the request to the engine and stores the extracted request information in the conversational context so that it becomes implicit in future interactions during the session. The dialog manager also decides on when to take proactive actions. The output from enacting these actions is then passed through the NLG, which has built-in models for responses in natural language.

The response is then processed to fit the format (and protocol) of the specific bot platform where the session was initiated (e.g., Amazon Alexa, Telegram), which will be in charge of managing the actual interactions with the user in their native conversation modality (text or audio). This requires the integration with existing platforms, with the necessary extensions to provide a proper experience fitting the modality and design language of the target platform.

5 Benefits and Challenges

Conversational web interactions that allow users to express their goals by talking to a website have the potential to change the way we interact with the Web, making it truly *inclusive* and *ubiquitous*. The benefits we identify are:

Improve the Browsing Experience for Visually Impaired Users. The proposed model is a significant improvement over the linear navigation in screen reader technology. Enabling users to express intents directly can reduce the impoverishment of linear transformation of visual information [19] and lower frustration.

Mobile Browsing. A consequence of allowing users to express their intents directly in a goal-oriented spoken dialog is the potential to lower the cognitive load associated with having to perform low-level interactions with websites, making browsing safer and more effective in low-attention situations.

Opening Up the Web to a New Generation of Clients. The proposed model and architecture could enable the now pervasive conversational agents and voice assistants such as Amazon Alexa, Siri, Google Assistant and chatbots, to access websites. This has the potential to improve the browsing experience in mutliple scenarios.

Promoting Adoption of Accessibility Guidelines. We believe making Web accessibility an ingredient of exciting new scenarios for web interactions can boost the adoption of accessibility guidelines and specifications, thus benefiting web accessibility in general.

Platform for Bot Development and Prototyping. We believe this approach is amenable to companies who do not have the budget or capability to develop ad hoc conversational agents to access their content and functionality. This is akin to what happens in web design where the option of developing a mobile app (probably the best approach for a mobile experience) coexists with responsive web designs and other approaches to design or annotate a website so that it can be consumed effectively by mobile users.

The key challenges we see to achieve these benefits are:

Reuse of the Existing Web Infrastructure. Building on the existing infrastructure, such as reusing existing technical specifications and standards for accessibility, should not break the experience for existing users of assistive technology and general users. Understanding what parts to reuse and how is one of the key aspects in materialising a conversational model.

Backward Compatibility, and Trade-Off Between Explicit and Inferred Domain Models. We have discussed an approach that is model-driven as it is important to have a reference model that will also allow developers to be in control of the experience. However, adoption is always a challenge, and so algorithms and heuristics could emerge, in the same way the accessibility community has been coping with this problem over the years.

Organizing Community Efforts. Another reason for following such an architecture is that the way a page is "used" is eventually dictated by the community of users, and we can envision communities of users as well as communities of developers creating "overlays" over a web site to make it "botifiable" and provide the annotations - and training data to map utterances into intents, and possibly even actions. Organizing and enabling such community efforts is another challenge.

Privacy and Security. Managing privacy and security in a context where users express their intents directly is another challenge. We did not address this issue in this paper but it is an area that will need further research.

Understanding how to deliver the experience, in what scenarios, and for what classes of websites it is suited, are all open questions that need to be addressed.

6 Directions for Future Work

The idea of conversational web interaction proposed in this paper opens up a need for further, fundamental research in key areas, including:

- Models, approaches and authoring tools to extend websites with conversational capabilities. This implies working with the underlying technical specifications of the Web (e.g., HTML, CSS, ARIA) to introduce the minimal set of extensions to allow web developers, the community or machine-based approaches to dote websites with conversational capabilities.
- Approaches for deriving conversational interfaces directly from websites. This has to do with the exploration of approaches to turn websites into dialogues with a user: from recognising user requests in natural language to enacting the necessary web interactions in the target website and preparing responses for the specific device and medium.
- Identifying effective design guidelines for conversational web interactions. Building successful experiences would require identifying and validating guidelines and best practices that go beyond general scenarios [3] to inform the design of effective and high-quality conversational web experiences.
- Understanding the impact of conversational web interactions in the target scenarios. This has to do with collecting evidence of the impact of the new paradigm in the accessibility and user experience, and understanding in what scenarios and in what form it can have the biggest impact.

We are currently exploring the technical feasibility through prototyping the various components of the architecture. Preliminary results confirm the feasibility of both general and domain-specific conversation models from suitably annotated HTML markup with a special focus on HTML lists and forms. With this basic infrastructure we plan to run the first exploratory user studies.

Acknowledgement. The study was supported by the Russian Science Foundation (project n. 19-18-00282).

References

1. Ahmed, F., Borodin, Y., Soviak, A., Islam, M., Ramakrishnan, I., Hedgpeth, T.: Accessible skimming: faster screen reading of web pages. In: UIST, pp. 367–378. ACM (2012)
2. Akpinar, M.E., Yeşilada, Y.: Discovering visual elements of web pages and their roles: users' perception. Interact. Comput. **29**(6), 845–867 (2017)
3. Amershi, S., et al.: Guidelines for human-AI interaction. In: Proceedings of CHI 2019, p. 3. ACM (2019)
4. Asakawa, C.: What's the web like if you can't see it? In: W4A, pp. 1–8. ACM (2005)
5. Ashok, V., Borodin, Y., Puzis, Y., Ramakrishnan, I.: Capti-speak: a speech-enabled web screen reader. In: W4A, p. 22. ACM (2015)
6. Ashok, V., Puzis, Y., Borodin, Y., Ramakrishnan, I.: Web screen reading automation assistance using semantic abstraction. In: IUI, pp. 407–418. ACM (2017)
7. Asri, L.E., et al.: Frames: a corpus for adding memory to goal-oriented dialogue systems. arXiv preprint arXiv:1704.00057 (2017)
8. Bigham, J.P.: Accessmonkey: enabling and sharing end user accessibility improvements. ACM SIGACCESS Access. Comput. **89**, 3–6 (2007)
9. Bigham, J.P., Lau, T., Nichols, J.: Trailblazer: enabling blind users to blaze trails through the web. In: IUI, pp. 177–186. ACM (2009)
10. Bigham, J.P., Lin, I., Savage, S.: The effects of not knowing what you don't know on web accessibility for blind web users. In: ACM SIGACCESS. ACM (2017)
11. Billah, S.M., Ashok, V., Porter, D.E., Ramakrishnan, I.: Ubiquitous accessibility for people with visual impairments: are we there yet? In: CHI, pp. 5862–5868. ACM (2017)
12. Borodin, Y.: Automation of repetitive web browsing tasks with voice-enabled macros. In: Proceedings of SIGACCESS 2008, pp. 307–308. ACM (2008)
13. Borodin, Y., et al.: Hearsay: a new generation context-driven multi-modal assistive web browser. In: WWW, pp. 1233–1236. ACM (2010)
14. Christian, K., Kules, B., Shneiderman, B., Youssef, A.: A comparison of voice controlled and mouse controlled web browsing. In: Proceedings of the Fourth International ACM Conference on Assistive Technologies, pp. 72–79. ACM (2000)
15. Conradi, J., Alexander, T.: Analysis of visual performance during the use of mobile devices while walking. In: Harris, D. (ed.) EPCE 2014. LNCS (LNAI), vol. 8532, pp. 133–142. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07515-0_14
16. De Rosa, A., Justice, D.: WebReader: a screen reader for everyone, everywhere. In: Proceedings of the 13th Web for All Conference, p. 10. ACM (2016)
17. Diggs, J., Craig, J., McCarron, S., Cooper, M.: Accessible rich Internet applications (WAI-ARIA) 1.1 (2016). Accessed 24 Apr 2016
18. Fecke, A., Jeleniowski, S., Joisten, M.: Accessible websites for the visually impaired: guidelines for designers. In: Mensch & Computer Workshopband, pp. 419–422 (2015)
19. Giraud, S., Thérouanne, P., Steiner, D.D.: Web accessibility: filtering redundant and irrelevant information improves website usability for blind users. Int. J. Hum Comput Stud. **111**, 23–35 (2018)
20. Guerreiro, J., Gonçalves, D.: Faster text-to-speeches: enhancing blind people's information scanning with faster concurrent speech. In: ACM SIGACCESS, pp. 3–11. ACM (2015)

21. Guha, R.V., Brickley, D., Macbeth, S.: Schema.org: evolution of structured data on the web. Commun. ACM **59**(2), 44–51 (2016)
22. Harper, S., Patel, N.: Gist summaries for visually impaired surfers. In: ACM SIGACCESS Conference on Computers and Accessibility, pp. 90–97. ACM (2005)
23. Hyman Jr., I.E., Boss, S.M., Wise, B.M., McKenzie, K.E., Caggiano, J.M.: Did you see the unicycling clown? Inattentional blindness while walking and talking on a cell phone. Appl. Cogn. Psychol. **24**(5), 597–607 (2010)
24. Inan, F.A., Namin, A.S., Pogrund, R.L., Jones, K.S.: Internet use and cybersecurity concerns of individuals with visual impairments. J. Educ. Technol. Soc. **19**(1), 28–40 (2016)
25. Insurance, L.M.: Pedestrian safety survey (2013). https://bit.ly/2WIDeOM
26. Lau, T., Cerruti, J., Manzato, G., Bengualid, M., Bigham, J.P., Nichols, J.: A conversational interface to web automation. In: UIST, pp. 229–238. ACM (2010)
27. Lazar, J., Allen, A., Kleinman, J., Malarkey, C.: What frustrates screen reader users on the web: a study of 100 blind users. Int. J. Hum.-Comput. Interact. **22**(3), 247–269 (2007)
28. Leshed, G., Haber, E.M., Matthews, T., Lau, T.: CoScripter: automating & sharing how-to knowledge in the enterprise. In: CHI, pp. 1719–1728. ACM (2008)
29. Mahmud, J.U., Borodin, Y., Ramakrishnan, I.: Csurf: a context-driven non-visual web-browser. In: Proceedings of WWW 2007, pp. 31–40. ACM (2007)
30. Michail, S., Christos, K.: Adaptive browsing shortcuts: personalising the user interface of a specialised voice web browser for blind people. In: Data Engineering Workshop, 2007, pp. 818–825. IEEE (2007)
31. Mika, P.: On schema.org and why it matters for the web. IEEE Internet Comput. **19**(4), 52–55 (2015)
32. Murphy, E., Kuber, R., McAllister, G., Strain, P., Yu, W.: An empirical investigation into the difficulties experienced by visually impaired Internet users. Univ. Access Inf. Soc. **7**(1–2), 79–91 (2008)
33. Nasar, J.L., Troyer, D.: Pedestrian injuries due to mobile phone use in public places. Accid. Anal. Prev. **57**, 91–95 (2013)
34. Oney, S., Lundgard, A., Krosnick, R., Nebeling, M., Lasecki, W.S.: Arboretum and arbility: improving web accessibility through a shared browsing architecture. In: UIST, pp. 937–949. ACM (2018)
35. Oshry, M., Auburn, R., Baggia, P., Bodell, M., Burke, D., Burnett, D., et al.: Voice extensible markup language (voicexml) 2.1. w3c recommendation (2007)
36. Plessers, P., et al.: Accessibility: a web engineering approach. In: Proceedings of WWW 2005, pp. 353–362. ACM (2005)
37. Power, C., Freire, A., Petrie, H., Swallow, D.: Guidelines are only half of the story: accessibility problems encountered by blind users on the web. In: Proceedings of CHI 2012, pp. 433–442. ACM (2012)
38. Rohrbach, M., Qiu, W., Titov, I., Thater, S., Pinkal, M., Schiele, B.: Translating video content to natural language descriptions. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 433–440 (2013)
39. Ronallo, J.: Html5 microdata and schema.org. Code4Lib J. (16) (2012)
40. Sato, D., Takagi, H., Kobayashi, M., Kawanaka, S., Asakawa, C.: Exploratory analysis of collaborative web accessibility improvement. ACM TACCESS **3**(2), 5 (2010)
41. Sears, A., Lin, M., Jacko, J., Xiao, Y.: When computers fade: pervasive computing and situationally-induced impairments and disabilities. HCI Int. **2**, 1298–1302 (2003)

42. Takagi, H., Kawanaka, S., Kobayashi, M., Itoh, T., Asakawa, C.: Social accessibility: achieving accessibility through collaborative metadata authoring. In: ACM SIGACCESS Conference on Computers and Accessibility, pp. 193–200. ACM (2008)
43. Tran, K., et al.: Rich image captioning in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 49–56 (2016)
44. Voykinska, V., Azenkot, S., Wu, S., Leshed, G.: How blind people interact with visual content on social networking services. In: CSCW, pp. 1584–1595. ACM (2016)
45. Wang, K.: SALT: a spoken language interface for web-based multimodal dialog systems. In: Seventh International Conference on Spoken Language Processing (2002)
46. WWW-Consortium et al.: Web content accessibility guidelines (WCAG) 2.1 w3c candidate recommendation (2018). Accessed 30 Jan 2018
47. Yesilada, Y.: Web page segmentation: a review (2011)
48. Zhu, S., Sato, D., Takagi, H., Asakawa, C.: Sasayaki: an augmented voice-based web browsing experience. In: Proceedings of SIGACCESS 2010. ACM (2010)