Reagent: Converting Ordinary Webpages into Interactive Software Agents

Matthew Peveler[1*], **Jeffrey O. Kephart**[2] and **Hui Su**[1,2]
[1]Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[2]IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
{pevelm}@rpi.edu, {kephart, huisuibmres}@us.ibm.com

Abstract

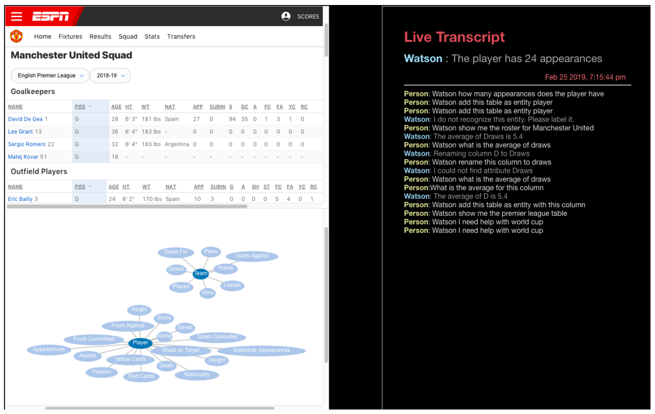
We introduce *Reagent*, a technology that can be used in conjunction with automated speech recognition to allow users to query and manipulate ordinary webpages via speech and pointing. *Reagent* can be used out-of-the-box with third-party websites, as it requires neither special instrumentation from website developers nor special domain knowledge to capture semantically-meaningful mouse interactions with structured elements such as tables and plots. When it is unable to infer mappings between domain vocabulary and visible webpage content on its own, *Reagent* proactively seeks help by engaging in a voice-based interaction with the user.

1 Introduction

Increasingly sophisticated technologies are being developed to enable scientists, business users, and students to interact with AI much as they would with a fellow human. Whether intended for tasks such as decision making [Farrell *et al.*, 2016], data exploration and analysis [Kephart *et al.*, 2018], or learning a language [Allen *et al.*, 2019], these technologies rely upon a mixture of input modalities, including voice, text, gestures, and pointing. To collect semantically-meaningful pointing events resulting from the user's interaction with the display, the common practice has been to build the user interface as a collection of bespoke webpages, each including special instrumentation that responds to mouse events. Multimodal cognitive agents will never become as pervasive as today's less-sophisticated bots unless a less laborious, more scalable approach is developed.

In this paper, we introduce *Reagent*, a technology that solves this scalability problem by capturing semantically-meaningful mouse events from non-instrumented webpages. When used in conjunction with a speech engine that transcribes verbal commands to text, *Reagent* effectively converts ordinary webpages into software agents with which one can interact naturally; in other words, one can query, analyze and manipulate the webpage's content through a combination of speech and pointing (e.g. at a cell in a data table).

[*]Contact Author



Manchester United Squad

Goalkeepers

NAME	POS	AGE	HT	WT	NAT	APP	SUBIN	S	GC	A	FC	FA	YC	RC
David De Gea 1	G	28	6' 3"	181 lbs	Spain	27	0	94	35	0	1	3	1	0
Lee Grant 13	G	36	6' 4"	183 lbs	-	0	0	0	0	0	0	0	0	0
Sergio Romero 22	G	32	6' 4"	183 lbs	Argentina	0	0	0	0	0	0	0	0	0
Matej Kovar 51	G	18	-	-	-	-	-	-	-	-	-	-	-	-

Outfield Players

NAME	POS	AGE	HT	WT	NAT	APP	SUBIN	G	A	SH	ST	FC	FA	YC	RC
Eric Bailly 3	D	24	6' 2"	170 lbs	Spain	10	3	0	0	0	0	5	4	0	1

Live Transcript

Watson : The player has 24 appearances

Feb 25 2019, 7:15:44 pm

Person: Watson how many appearances does the player have
Person: Watson add this table as entity player
Person: Watson add this table as entity player
Watson: I do not recognize this entity. Please label it.
Person: Watson show me the roster for Manchester United
Watson: The average of Draws is 5.4
Person: Watson what is the average of draws
Watson: Renaming column D to Draws
Person: Watson rename this column to draws
Watson: I could not find attribute Draws
Person: Watson what is the average of draws
Person:What is the average for this column
Watson: The average of D is 5.4
Person: Watson add this table as entity with this column
Person: Watson show me the premier league table
Person: Watson I need help with world cup
Person: Watson I need help with world cup

Figure 1: Electron environment containing ESPN page, transcript window, and ontology being built by the human and *Reagent*.

To support the user's desire to issue commands in a familiar domain vocabulary that may not necessarily match the terminology used on the webpage, *Reagent* uses salient structured content on the page to automatically infer mappings between the domain vocabulary and the webpage terminology. When the system fails to find such a mapping, it is able to elicit help from the user to fill these gaps.

The next section overviews some technical details regarding how *Reagent* is implemented and how it is situated within a larger multi-modal AI assistant framework. Then, we briefly describe two aspects of *Reagent* that are illustrated more fully in the accompanying demo video, available at https://youtu.be/qVmY3YmNzGE. First, we demonstrate how — with no prior training or knowledge — one can query and manipulate a table contained within a third-party webpage (SkySports) via speech and pointing. Second, we demonstrate an interaction with an ESPN page, whereby *Reagent* proactively asks the user to help fill in gaps in its ontology, allowing the system to subsequently understand more elaborate queries in the domain vernacular.

2 Technical Details

Reagent, along with the multi-modal AI assistant system in which it is embedded [Kephart *et al.*, 2018], is built upon Electron, a Node.js framework for developing desktop applications from web technologies. As illustrated in Fig. 1, the

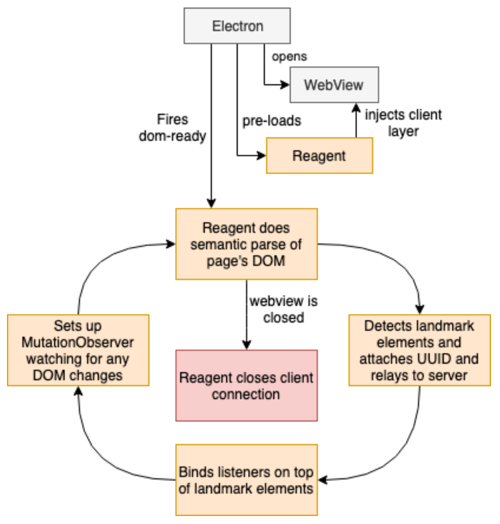

Electron
opens
WebView
injects client layer
Fires dom-ready
pre-loads
Reagent
Reagent does semantic parse of page's DOM
webview is closed
Sets up MutationObserver watching for any DOM changes
Reagent closes client connection
Detects landmark elements and attaches UUID and relays to server
Binds listeners on top of landmark elements

Figure 2: Flow of Reagent system and its integration within Electron and webpages.

system displays on a computer screen a collection of webpages (termed *webviews* in Electron).

When a user opens a webpage, *Reagent* inserts a transparent layer on top of that page's webview. This layer scans the page, looking for meaningful semantic structures (e.g. table, plot). Upon detecting them, it further injects specific JavaScript code to parse the structure and create listeners for meaningful user interactions. In the case of a table, this entails detecting the headers of the table (marking them with a unique ID) and adding a listener to detect whenever a user's cursor enters, leaves, or clicks on any cell in the table. Finally, Reagent creates a *MutationObserver* on the page to detect any potential DOM changes so that it can re-start the page analysis and re-bind itself to any changed or inserted content. This sequence is shown in detail in Figure 2. Interaction between *Reagent* and the webviews happens via a WebSocket; interaction with the rest of the system is via HTTP GET/POST.

Users interact with a web page by pointing to elements (via mouse or other pointing device) and either typing into a chat window or speaking. The system calls Watson Assistant [1] to obtain from the text (or text transcription) an "intent" classification plus a set of entities. Then, it checks whether it can construct from the intent and entities a fully-specified JSON command. If not, the system retrieves from the user interaction event log maintained by *Reagent* the most recent events that could possibly be mapped to the missing parameter(s). The resulting command is then executed, and the result is displayed on the Electron display canvas, possibly accompanied by synthesized speech [Divekar *et al.*, 2018].

Often, webpage developers use abbreviations or synonyms that do not correspond directly to the terms that users would naturally use to interact with those pages. For example, on the ESPN page shown in Fig. 1, the column for "appearances" is labelled "APP". In the course of processing HTML elements, *Reagent* identifies tags that may indicate more human-friendly terms, such as tooltips that reveal explanatory text on hovering over the element, and uses approximate text matching to infer likely associations. *Reagent* can also exploit a large body of accessibility work including the W3C Standard Web Content Accessibility Guidelines[2] or a Voluntary Product Accessibility Template[3] to automatically derive meaningful semantic information in "hidden" attributes. In cases where the system is unable to understand or disambiguate the semantic information, it solicits a definition from the user. For example, if *Reagent* is unable to divine the meaning of "APP" or other columns, the system generates a synthesized voice asking the user to mouse over the unknown columns and state what names they wish to use for them. *Reagent* then stores this information for use during subsequent visits to that webpage, or others with similar content.

3 Building an Ontology

To help demonstrate the capabilities of *Reagent*, we use the system to help build an ontology dealing with football teams and players. To start with, we open a page from SkySports showing the premier league table [4]. After opening it, Reagent detects a table as well as the hidden metadata for some of the columns. However, not all columns have human readable metadata, so the system asks the user to supply the missing domain-specific vocabulary for team attributes. After filling in this information and adding the table as a team entity in the ontology, the user opens a page for a team's roster on ESPN [5]. On this page, there are two tables representing the two types of players: goalkeepers and outfield players. Both of these tables contain abbreviations for the table headers, which *Reagent* is able to understand automatically by exploiting human-readable labels within the HTML structure. This allows a user to add each table directly into the ontology without the system requiring any additional help. Finally, the user verbally asks the system to add a relationship between the added player and team entities. With this more complete ontology, the system is now equipped to answer more complex questions utilizing knowledge across pages and entities.

4 Conclusions

Taking advantage of commonly-occurring structural motifs and human-friendly tagging such as tooltips, *Reagent* makes it easy for developers to create cognitive applications that support voice-based interactions with ordinary webpages. Moreover, *Reagent* readily learns vocabulary by asking questions when it cannot establish a mapping between the user's and webpage's terminology. One potential avenue for further research is to improve the knowledge acquisition process, utilizing prior work on determining salient details and content from less-structured webpages and content [Joshi and Liu, 2009; Sun *et al.*, 2011]. Another path is to automatically determine the domain under investigation, thereby allowing the system to bootstrap itself with knowledge from publicly available datasets such as DBPedia [Auer *et al.*, 2007].

[1]https://www.ibm.com/cloud/watson-assistant/

[2]https://www.w3.org/WAI/standards-guidelines/wcag/

[3]https://www.section508.gov/sell/vpat

[4]https://www.skysports.com/premier-league-table/2018

[5]https://www.espn.com/soccer/team/squad/_/id/360/manchester-united

References

[Allen *et al.*, 2019] David Allen, Rahul Divekar, Jaimie Drozdal, Lilit Balagyozyan, Shuyue Zheng, Ziyi Song, Huang Zou, Jeramey Tyler, Xiangyang Mou, Rui Zhao, Helen Zhou, Jianling Yue, Jeffrey O. Kephart, and Hui Su. The rensselaer mandarin project — a cognitive and immersive language learning environment. In *Proceedings of AAAI 2019*, 2019.

[Auer *et al.*, 2007] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[Divekar *et al.*, 2018] Rahul R. Divekar, Matthew Peveler, Robert Rouhani, Rui Zhao, Jeffrey O. Kephart, David Allen, Kang Wang, Qiang Ji, and Hui Su. CIRA: An architecture for building configurable immersive smart-rooms. In *Advances in Intelligent Systems and Computing*, pages 76–95. Springer International Publishing, nov 2018.

[Farrell *et al.*, 2016] Robert G Farrell, Jonathan Lenchner, Jeffrey O Kephart, Alan Webb, Michael Muller, Thomas Erickson, David Melville, Rachel Bellamy, Daniel Gruen, Jonathan Connell, Danny Soroker, Andy Aaron, Shari Trewin, Maryam Ashoori, Jason Ellis, Brian Gaucher, and Dario Gil. Symbiotic cognitive computing. *AI Magazine*, 37(3):81–93, 2016.

[Joshi and Liu, 2009] Parag Mulendra Joshi and Sam Liu. Web document text and images extraction using dom analysis and natural language processing. In *Proceedings of the 9th ACM Symposium on Document Engineering*, DocEng '09, pages 218–221, New York, NY, USA, 2009. ACM.

[Kephart *et al.*, 2018] Jeffrey O. Kephart, Victor Dibia, Jason Ellis, Biplav Srivastava, Kartik Talamadupula, and Mishal Dholakia. A cognitive assistant for visualizing and analyzing exoplanets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Sun *et al.*, 2011] Fei Sun, Dandan Song, and Lejian Liao. Dom based content extraction via text density. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 245–254, New York, NY, USA, 2011. ACM.