The Conversational Internet: Creating a natural language interface for visually impaired people to converse with the Web

Niki Gomez
Royal London Society for Blind People
Dorton House
Seal, Sevenoaks, TN15 0EB
+44 (0)7971 497420
niki@sloan.mit.edu

Dale Lane
IBM UK
Hursley Park, Winchester
Hampshire, SO21 2JN
+44 (0)7887 928480
dale.lane@uk.ibm.com

Julian Dailly
Royal London Society for Blind People
Dorton House
Seal, Sevenoaks, TN15 0EB
+44 (0)7909 514858
julian.dailly@rlsb.org.uk

ABSTRACT
The Conversational Internet is a working prototype created in collaboration between the Royal London Society for Blind People and IBM UK. It addresses challenges of visually impaired users using screen readers as a method of accessing the Internet.

The Conversational Internet allows users to selectively retrieve information from web pages by asking questions, and aims to provide a conversational interface. The software has some understanding of the contents of the page and is able to describe it to the user, making interactions easier and faster.

IBM created a proof-of-concept using machine learning, natural language processing and speech technologies and are developing it further. The approach could be applied to mainstream markets such as in cars - providing an eyes-free solution for information retrieval that reduces information overload.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *user interfaces*

General Terms
Design, Human Factors

Keywords
Conversational Internet, blind, visually impaired, IBM, RLSB, Everybody Technology, natural language, machine learning, browsing by voice

1. INTRODUCTION
The Conversational Internet is a prototype created by IBM with the Royal London Society for Blind People, a 175 year old charity in the UK.

The aim of the Conversational Internet is to make access to the Internet for visually impaired users much easier: a faster and more precise experience using more intelligent software. We aim to find a better solution than screen readers to access the Web.

RLSB's beneficiaries reported that screen readers were incredibly slow and had a poor signal-to-noise ratio, often taking several minutes to read a single page. This is due to the fact that they read out every word and element that is on the page.

The user has to understand what the page means, combining the series of elements that are read out into a mental model of the page, remembering what is where and what their options are. In addition to being slow, as an interaction model this introduces a significant cognitive burden.

RLSB approached IBM to explore how this could be improved upon. A project was formed to build a demonstration of a Conversational Internet: a system able to retrieve information from web pages by allowing the user to ask questions, in a conversational-style approach.

The aim was to shift some of the cognitive burden from the user to software, reducing the information that a user needs to hear and remember in order to interact effectively with a web page.

2. CONVERSATIONAL INTERNET
2.1 Functionality
IBM created a proof-of-concept that demonstrated the value of two improvements to the traditional screenreader approach.

2.1.1 Understanding the web page
Rather than read everything that is on a web page, the software attempts to interpret the page, both structurally and semantically. Important elements are identified to the user, together with an indication of possible next steps.

This included:

- identifying the type of website
  e.g. “This looks like a news site”, “This looks like a retail site”
- identifying the available navigation methods
  e.g. “There are a number of menus”, “There is a search box”, “There are several links to stories”
- identifying a call-to-action or instruction to the user
  e.g. “There is a form for entering comments”, “There is a form for entering a postcode”

The result was that, on visiting a text-heavy site such as BBC News, rather than spend several minutes reading out every element on the page, the software informs the user that they are on what looks like a news site, and what their options are.

2.1.2 Understanding the user's intention

Rather than require the user to translate their intention (e.g. clicking on a particular menu item) into the series of physical actions necessary to achieve it (e.g. combination of tab / arrow / Enter key presses), the software attempts to understand a request provided in natural language, and perform the necessary actions.

This included:

- Retrieving information
  e.g. "What is the top story?", "What are the menu items?", "What are the headlines?", "What are the most popular stories?", "Read me the story"
  The requested information is found and read out, without needing the user to listen to everything else.
- Performing mouse actions
  e.g. "Take me to Politics", "Go to the Olympics story"
  The appropriate link/button/page element is identified and clicked upon, without needing the user to know where on the page it is, or have to navigate to it.
- Performing keyboard actions
  e.g. entering search terms, comments, etc.
  The appropriate text box on the page is identified, and the requested text is submitted without needing the user to know where on the page it is, or type in their query.

Rather than require users to learn an extensive set of commands, the user is free to describe their request in natural language which the software attempts to recognise.

The user is able to extend the vocabulary of the system by teaching it new synonyms for entities and actions that it knows. When an unknown term is used, the system uses language resources to look for synonyms of the term already in it's lexicon. The user is asked to confirm the interpretation of the unknown word before it is added to the system.

2.1.3 Speech interface

The proof-of-concept offers two modes of input: keyboard or speech. Speech recognition allows for a completely eyes-free mode of operation, with the only physical interaction required being button presses to start and stop the system listening for input. Responses are read out using text-to-speech. The messages are also displayed on-screen alongside the page.

2.2 Project status

The project has produced a functioning prototype which is continuing to be developed further. Training to date has led to the proof-of-concept being most effective at article-based sites, such as news sites, blogs, and encyclopedias.

2.3 Implementation

2.3.1 Conversational Internet client

For the purposes of the proof-of-concept, the client was implemented in JavaScript as an extension to the Firefox web browser [1]. This allowed the client to use Firefox's existing rendering engine, reducing work involved in interpreting the raw HTML of web pages. It also avoided needing to interpret JavaScript or other dynamic elements. The client looks at the final state of the page, as rendered by Firefox.

This also means that pages requiring the user to log in, or have some other cookie-maintained state, are easier to support than it might've been using a server-only implementation.

When the user visits a new web page, the client captures the current state of the page DOM. It calculates attributes of the page elements, such as their absolute position, using jQuery [2]. The client ensures that every element on the page is uniquely identifiable, adding IDs to any page element that does not already have one, so that responses from the server can instruct the client where to click or type with an agreed frame of reference.

This information and the user's id, is submitted to a REST web service provided by the Conversational Internet server.

Once processed by the server, the client informs the user that the system is ready to receive questions, with an audible beep. The user can then submit questions or instructions, which are sent to the server using REST HTTP requests.

As all of the interaction between client and server uses REST over HTTP, it is possible to implement future clients, such as for smartphone or in-car platforms, without changes to the server.

The client also includes a training mode, used to gather machine learning data. This was used during development – by navigating to new web pages in Firefox, developers could mark which parts of the page contained menus, comment boxes, headlines, search boxes, etc. This provided valuable training and test data for the development of the server components.

2.3.2 Conversational Internet server

The server is implemented in Java, hosting a number of web services. The functionality of the services themselves are implemented using Apache UIMA [3] – a framework and OASIS [4] standard for building text analytics applications.

UIMA hosts pipelines of discrete annotators, each of which uses a different strategy to try and identify meaning behind areas of the web page the user is viewing.

There are two main pipelines in the server. The first is used to understand the web page.

As described earlier, when the user visits a web page, the Conversational Internet client submits it to the server for analysis. The contents of the web page – including the text contents, the DOM structure of the HTML markup including any semantic tags used, and other metadata gathered by the client – go through a UIMA analysis engine. It passes through several annotators, each of which is looking for something different in the page.

By itself, no one of these annotators is fool-proof. But they're not stand-alone applications. As elements in a pipeline, contributing a strategy to a collection of results, they're each useful indicators. Each of the annotators in each stage of analysis contributes it's own metadata, building up a picture of what the page contains.

Several annotators were implemented during development of the proof-of-concept, and these were grouped into four categories.

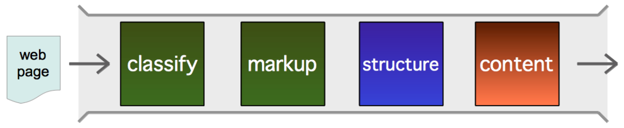

Figure 1: Understanding the web page

2.3.2.1 Classify

What type of page is this? Subsequent annotators in the pipeline will want to treat a news site differently to a retail site, so the first phase of analysis attempts to classify the site, giving a useful steer to later annotators.

The two main strategies used in these annotators are:

1) Machine learning classifiers (e.g. Bayesian) which use the contents of the page to suggest the type of site.

2) Whitelists of known domains from Internet directories

2.3.2.2 Markup

What can we infer from markup used in the page?

For example, have known semantic HTML tags been used, identifying a section of the page containing a menu or a title?

These were not found to be in common use on web pages during development, however for pages which include them, they proved to be a useful signal, so some of the annotators look for known semantic tags.

2.3.2.3 Structure

What does the structure of the page layout suggest? Using machine learning, the system is trained to recognise useful elements, based on structural features such as location, size, type, colour, class names, parent and child elements, names, labels, etc.

For example, the system was trained by manually classifying which element of a web page contained the web site's search box. Repeating this for a large number of pages from many websites created a model which can be applied to previously unseen websites, giving an estimation of which element in the page is most likely to be the site search box.

Similarly, a model was trained to identify which part of a page contains a menu. Signals such as a series of links, arranged vertically or horizontally, with similar font styles and sizes, placed near an edge of a page, and many more were user. This allowed for an effective model to be created, able to identify the menu on most previously unseen sites.

Previous work, e.g. Boilerpipe [5], showed the potential of machine learning models to extract the main textual contents of a webpage, using signals such as the number of words per paragraph. We found that we were able to build models capable of extract several other elements common to most web pages (e.g. comment boxes, search boxes, menus, links, etc.)

The machine learning models were created and trained using SPSS Modeler [6]. This offered a simple way to try out a large number of different modeling algorithms, including classification, clustering, regression and more, with minimal additional development effort. SPSS offers a desktop client UI for exploring the effectiveness of the different algorithms, and the impact on accuracy of the different features that we tried. It allowed experimenting as different features and/or algorithms could be enabled/disabled, showing the result this had on the test data set. The models used varied between the different structural elements that were being looked for, however most made use of logistic regression, bayesian network and CHAID techniques.

At runtime, this is hosted in an SPSS server that the UIMA Java code can send requests to. This was fast enough that many ML models could be used during the page analysis annotators.

2.3.2.4 Content

What does the text of the page tell us? Using natural language processing, the prototype identifies semantic meaning behind some of the content.

This is a focus area for future development. The proof-of-concept demonstrates potential benefits of this by looking for common patterns of calls-to-action, such as instructions to fill out forms.

These patterns were created using LanguageWare [7], a tooling environment for building semantic text matching applications. The patterns developed were exported into UIMA-compliant annotators which could be included in the page analysis pipeline.

The output from this first pipeline is a CAS (Common Analysis Structure) file, which is stored ready to receive questions and instructions about it. The CAS is an XML structure made up of the original web page contents, together with all of the annotations metadata that were added by the many annotators in the page analysis pipeline.

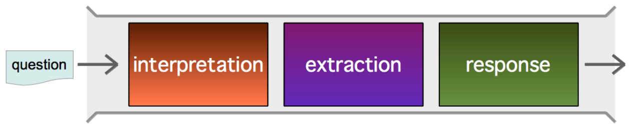

Figure 2: Responding to commands

The second pipeline is used to respond to user commands. Several annotators were implemented during the development of the proof-of-concept, which were grouped into three main phases.

2.3.2.5 Interpretation

The user's request, submitted in natural language, enters a second UIMA analysis engine. LanguageWare rules are used to map the provided request/command to the nearest likely match amongst commands that the Conversational Internet supports.

This uses a number of common natural language processing techniques, such as part-of-speech tagging and entity recognition, to try and interpret the user's request.

If an entity or action in the request is not recognised, language resources such as WordNet [8] are used to attempt to resolve them. If the unknown word is a known synonym of something already in the system's lexicon, and the part-of-speech tagging suggests that the word would fit, this is offered to the user as a possible interpretation. If the user accepts the interpretation, the unknown word is added into a user-extension vocabulary.

Context is a focus area for future development. Currently, commands are interpreted with minimal reference to the analysed page, and conversational context is limited to a few simple patterns, such as remembering the last question returned by the server (allowing the user to answer yes/no to some questions, or choose an item from a list that the server offered).

2.3.2.6 Extraction

Once the user's intent is identified, the system refers to the analysis output from the first pipeline.

If the user asked for a specific piece of information, the output helps identify this info. It could be a selected passage from the page: a sentence or phrase. It could be the outcome of one or more analyses - such as identifying the type of website, or the navigation methods the page makes available.

If the user asked to perform a task, such as follow a hyperlink, the output helps identify the link to use.

The serialized CAS from the page analysis pipeline is retrieved from the cache or disk storage (depending on the time since analysis). The interpretation of the user's command informs which annotations are retrieved from the analysed page CAS.

2.3.2.7 Response

If the user requested information, it is read out, selectively reading out information from the page that the user wants, not the entire page contents.

The outcome is returned using text-to-speech. Speech services are provided by Nuance NDev [9], which supplies web services APIs for both the speech recognition used for input, and the text-to-speech used to read out responses to the user.

If the user described an action, this is performed and confirmed. E.g. the system can navigate to a page and say where it has gone.

3. IMPORTANCE

The prototype is the first step in finding a solution for all websites to be accessible in a new and smarter way to what currently exists.

This solution targets the blind and visually impaired (VI) market initially as they are the 'hardest to please'. It could be adapted to the mainstream market. The natural language interface means it could target web illiterate users for whom learning to access the web is a problem. It could also solve the need for many users to delve through the vast amount of information on the web and provide an 'eyes-free' and 'hands-free' solution, enabling multi-tasking- accessing the web whilst driving, jogging cooking, etc. It can act as a Web assistant.

The market for voice technology is growing in both the mainstream and VI markets. In the last twelve months more than three major voice assistant products have launched including Siri, Evi, Nuance's Nina and Google's Now.

We believe the market is ready and that there are various business models available to making all websites conversationally-ready.

4. USER FEEDBACK

A formal usability study has not yet been carried out, however feedback through informal user acceptance reviews by visually impaired users was an important driver during the development of the prototype.

This included:

- interviews to identify current approaches for getting information from the Internet,
- setting tasks to complete using the proof-of-concept and observing how successful they were,
- interviews after they used the tool to obtain feedback about their experience.

Reviews of later versions of the prototype were very positive, and showed that visually impaired users were able to complete most of their common information retrieval tasks faster than using their current strategies.

5. CONCLUSIONS

Although still an early prototype, the Conversational Internet proof-of-concept demonstrates that the basic idea has potential and that the UIMA architecture and application of machine learning and natural language processing is a good fit for the challenges facing screen reader users.

In terms of demand for this product, we see the following trends as being important:

- an aging population with increased accessibility needs;
- greater accessibility legislation and
- growing wider market for voice technologies due to speed, mobility and comfort.

Our solution means that either the entire Web can be tagged and described or this can happen for a selection of websites, leading to a two-tier Web.

The Conversational Internet will be of great use for the Virtual Voice Assistants and Voice controlled Search Engine market as well as voice apps.

6. REFERENCES

[1] Mozilla Firefox
http://www.mozilla.org/en-US/firefox/

[2] jQuery
http://jquery.com

[3] UIMA – Apache Unstructured Information Management Architecture
http://uima.apache.org

[4] OASIS - Organization for the Advancement of Structured Information Standards
https://www.oasis-open.org

[5] Boilerpipe
http://code.google.com/p/boilerpipe/

[6] IBM SPSS Modeler
http://ibm.com/software/analytics/spss/products/modeler/

[7] LanguageWare
http://ibm.com/software/globalization/topics/languageware/

[8] WordNet
http://wordnet.princeton.edu

[9] Nuance NDev
http://dragonmobile.nuancemobiledeveloper.com/

7. FURTHER INFORMATION

[1] Walkthrough and explanation of the prototype
http://youtu.be/uS6oquJdgbw

[2] Recording of an early iteration of the proof-of-concept
http://youtu.be/tSGyPCcO-bY

[3] Video outlining the challenges of screen readers
http://vimeo.com/33399797

[4] Short interview with one of the VI users who provided feedback during development
http://www.bbc.co.uk/news/technology-19606004
(starting from 1min 47secs)