# Domain Specific Newsbots

## Live Automated Reporting Systems involving Natural Language Communication

Al Johri
The Washington Post
1301 K Street NW
Washington, DC, USA
al.johri@gmail.com

Eui-Hong (Sam) Han
The Washington Post
1301 K Street NW
Washington, DC, USA
sam.han@washpost.com

Dhrumil Mehta
Harvard Kennedy School Ash Center
79 John F. Kennedy Street,
Mailbox 74
Cambridge, MA
dhrumil_mehta@hks.harvard.edu

## ABSTRACT

Media outlets are currently in the nascent stages of adopting bots and automated journalism for the purposes of news. The Washington Post has chosen to invest heavily in exploring these new avenues of storytelling by creating bots for Facebook Messenger and Amazon Echo as well as using natural language generation to write articles based on data. As this technology has matured within our newsroom, we have abstracted a general structure for domain-specific bots that report live data which we call Heliograf. This paper will outline the structure of Heliograf and discuss the lessons we have learned as well as the challenges we foresee as this technology matures.

## CCS Concepts

•**Computing methodologies** → *Natural language generation;* •**Applied computing** → *Publishing;*

## Keywords

automated journalism; automated reporting; robot journalism; bot journalism; facebook messenger; Alexa; amazon echo; natural language generation; nlg; bots

## 1. INTRODUCTION

If 2015 was the year of automated journalism, 2016 is the year of bots. While Siri, Cortana, Alexa, and whatever Google will inevitably name their voice assistant, battle their way for a permanent place in our lives, it's clear that conversational bots are here to stay. The question that remains is how will journalism define its place in the world of bots.

Automated journalism is a vast landscape loosely defined by the idea of using "algorithms to automatically generate news from structured data" [7]. Amid the sensationalist news coverage of robot journalists next in line for the Pultizer [8], soon to write 90 percent of online news content [5], or the end of human reporters as we know it [2], we lose sight of exactly what automated reporting is and what it means for a news organization today.

We should think of automated journalism as we have come to think of interactive graphics, another medium through which we can disseminate news and tell stories. When used well, just as an interactive graphic can improve the experience of consuming news by highlighting certain information or helping the reader discover the story that is told with data, automated journalism can enhance the experience of consuming news by presenting the information that is relevant, timely, and specific to a reader or listener.

In this paper we will focus on a subset of automated journalism that involves natural language communication otherwise referred to as bot journalism. These applications are usually referred to as conversational bots and can be implemented in text based mediums such as Slack and Facebook messenger or voice based mediums such as Amazon's Alexa. We will specifically discuss two conversational bots that we have created at the Washington Post, our internal primary elections bot and our Olympics bot.

We chose to focus on these two live conversational bots because they report data that is changing over time and use different natural language generation techniques to generate stories live as opposed to simply delivering templated content generated offline. The former, our first domain-specific conversational bot, reported the results of the primary elections in real time as they unfold. The latter, a more mature bot, built on the lessons learned from the first, and reported medal results from the 2016 olympic games as they are played. As we moved from our first forray into natural language generation to building more domain specific bots, we have abstracted the general structure into a system we call Heliograf.

In section 2 of this paper, we will discuss the current sucessful manifestations of automated journalism and at various levels of complexity. In section 3 we will introduce Heliograf and its structure. In Section 4, we will discuss practical lessons we have learned in implemetation of bot technology both in a technical sense and in terms of newsroom processes involving collaborations between journalists and engineers.

## 2. BACKGROUND

### 2.1 Natural Language Generation

One of the key components to automated reporting is nat-



*CJ2016 September 30–October 1, 2016, Stanford, CA, USA*



DOI: xx.xxx/xxx_x

ural language generation or NLG. At the simplest level, NLG is just a set of conditional "if statements" and templates resembling "mad-libs", a popular fill-in-the blanks word game. Some of the best examples of automated reporting in the industry use this type of simple templating. One such example is "The Best and Worst Places to Grow Up: How Your Area Compares" from the Upshot. The story "uses pre-assigned blocks of text and follows specific rules for how to assemble paragraphs based on the available data" [11].

Building off of this idea, Automated Insights offers a platform called Wordsmith that enables journalists to easily create such template based stories. One of the most popular examples of this is the AP's quarterly earnings reports powered by Wordsmith. More recently, AP now automatically generates stories about minor league baseball games also using the Wordsmith platform [9]. Technology such as Wordsmith helps keep the editorial control of stories in the hands of journalists and editors and reduces the friction in the back and forth between engineering and the newsroom.

While simple templating has created some amazing pieces, natural language generation technology has come a lot farther. Narrative Science offers a product named Quill that features "Advanced Natural Language Generation." Quill has the ability to generate multiple permutations of sentences, paragraphs, and stories overall from a single set of data. One of the key features of Quill is the ability to generate a single story in multiple tones or from multiple angles [1]. An example of this is their generation of minor league baseball game stories that can play up a player's achievements while playing down their losses. Narrative Science made a big splash in the news industry when it began generating company earning previews for Forbes [3]. In addition, Narrative Science helped Propublica generate stories for their "Opportunity Gap" project [10].

Going from template based approaches to the technology that powers Quill is a pretty big jump. A stepping stone along the way for natural language generation is grammar based methods. This type of natural language generation system has encoded within it the grammatical rules of a particular language. Rather than specifying an exact text-based template, complete with punctuation, verb tenses, etc., the person building the bot can specify the syntactic structure elements in a sentence (nouns, verbs, etc) which the system can then construct into prose using its grammar rules. We will explore this method further when discussing the Washington Post Olympics Bot.

## 2.2 Live Automated Reporting Systems

One additional consideration in building bots is whether the underlying data is unchanging or is being modified in real-time as events occur. Some great examples of automated reporting systems that use real-time data include the "Homicide Report" [4] and "Quakebot" [13] from the Los Angeles Times. Similar to the previous simple templating examples, Ken Schwencke, developer of these systems, said that the "underlying code is 'embarrassingly simple,' as it merely extracts numbers from a database and composes basic news stories from pre-written text modules" [7]. However, these pieces are fundamentally different in both their technological architecture and the editorial process.

Accessing an updating external data source for telling stories in real time (often through an external API) poses new challenges such as monitoring and validating changes in the data. Stories generated in real-time are often published immediately as the data changes, sometimes without human oversight. Editorially, this means that a lot more trust goes into the data source which fills the variables into a template. Technically, this means a lot more validations to ensure that incoming data is what is expected and fluctuations out of the ordinary are flagged for human review.

## 2.3 Two Way Communication

While many of the examples we have discussed involve algorithms that publish content, the examples we discuss in this paper involve two-way communication with a user. Newsrooms, including our own, are still in the experimental phases of automated journalism involving two-way communication and have implemented these "news bots" with mixed success. The first generation of newsbots have faced many issues including "glitches, slow response time, and excessive notifications" [12]. These newsbots have often been clunky, slow, and unnatural to communicate with. Taking the leap and being willing to experiment with new technologies is crucial to keeping journalism relevant in today's world; however as journalists we should be thinking not just about getting ourselves on the newest and latest platform, but about how those technologies can best be used to add value to our readers.

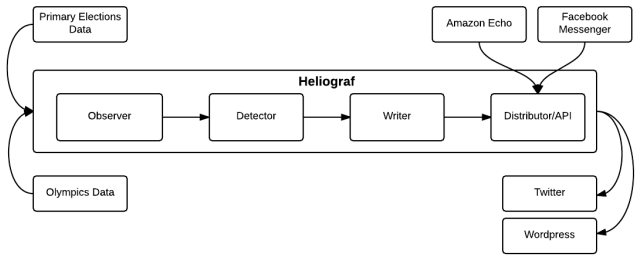
In its current state of maturity, artificial intelligence technologies tend to do much better in a domain restricted setting than in a general setting. This makes two-way conversational bots a great medium through which to communicate complex and vast datasets. We have found that our more successful ventures into bot-journalism have involved bots specific to domains such as the US elections or the 2016 olympic games. By focusing on a specific domain, journalists and engineers can put more effort towards building a system that has a depth of knowledge about a particular topic and can better anticipate all the different ways in which a user might want to learn about that topic to provide a relevant and engaging answer to their inquiry.

# 3. HELIOGRAF

At the Washington Post, we made our first forray into bot journalism with the primary elections. This was an internal attempt at automated story telling using a live stream of data. The goal was to design a bot that (A) reacts to triggers based on time and changes in the data, (B) generates different types of stories, and (C) outputs to different distribution channels. The bot posted an update whenever races were called, the last polls closed in a state, or if a certain amount of time had lapsed without any trigger occuring. It is capable of posting a continuous stream of events (eg. twitter) or updating a single "living story" as events unfold. We made the bot accessible through two-way communication platforms such as Amazon Echo and Slack as well as being able to broadcast one-way updates to Twitter and update a live story in Wordpress.

Our experiments with the primary elections bot resulted in one large, blocking, monolithic application. We learned that many of the pieces of this application can be broken down to small re-usable components. For the 2016 Summer Olympics, we sought to fix many of these issues by creating Heliograf, a generalized architecture of microservices. Heliograf enables us to quickly create bots for large, complex news events and support new distribution channels that keep

Figure 1: Heliograf Architecture




popping up everyday. Heliograf consists of four main parts: an observer, detector, writer, and distributor.

## 3.1 Observer

The observer (or crawler) is an asynchronous application that monitors REST endpoints for any changes to data and caches the most recent version. In the primary elections the observer monitored one endpoint per race, while in the Olympics it monitored one endpoint per discipline (sport). In some applications, this will be unecessary as a webhook may be provided, but most data sources in the wild tend to be RESTful.

Given RESTful endpoints, the observer's polling delay is where you define how "realtime" your stories will really be. Starting the conversation with the newsroom for how to define polling delays and cache invalidation tends to start with "instantaneous" and we haggle our way up from there. There is significant tension between the current infrastructures used to widely distribute data (REST) and the needs of automated and bot reporting. Furthermore, because of the sheer speed at which the observer needs to poll data, its advisable to share the same cache between other users of the same data sources (i.e. graphics) to prevent rate limitting.

## 3.2 Detector

The detector is where the event detection logic lies. We define triggers that create events. Some of the greatest challenges we faced with our primary elections bot was in defining the triggers, especially for mediums that support one-way push updates. Unlike the Olympics or the general election, the number of primaries per day can vary grealy with some days having a single race and other days with more than a dozen (e.g. super tuesday). Defining triggers that work in both of these settings can prove problematic. In the Olympics, its easy to define a trigger such as "medal results announced". This trigger only needs to take into account the current and previous state of the data. In the primaries, while we could just trigger when races are called, because the number of races per day varies, the number of times it triggers will vary greatly with somedays triggering far too frequently and other days not triggering enough. To complicate things further, when races start to get close, we want to increase the frequency of these triggers. Without a consistent stream of clearly defined events like medal results announced or races called, the bot not only needs to tell a story but also in a sense "find" a story within the data. Is the race getting close? Is there an upset from polling expectations? This requires having stateful triggers as the bot needs to remember how it characterized the race in the past in order to determine how the race is characterized in the present. All of this extra logic is with the goal of keeping the user informed of the current status of a race. The trick is to do it without being too overbearing.

To give a specific example, a trigger could be when the percentage of precincts reporting goes from 0 to some value greater than 0. The corresponding event would be that the votes have started to come in. In the primary elections, we sent out tweets as various events occurred throughout a race. For example, when the votes started coming in, a race was close, the precincts reporting increased by ten percent, a race was called, one hundred percent of precincts had reported results, etc. Because the event detection logic is far simpler in the Olympics (a medal was awarded every time the data changed), we chose to skip the detector.

## 3.3 Writer

The writer is one of the most interesting components of Heliograf. In the primaries, we used simple templating to produce output for all of our distribution channels. This strategy proved extremely fragile because of the endless number of permutations that would have to be written to account for all the different events that could occur in any given order and the grammatical nuances of reporting each one. While 140 character tweets are easy enough, writing templates that talk about two things in context of one another, in this case the republican and democratic races in a particular state, tends to get rather complicated. We quickly found that no assumptions could be made in the realm of elections: you can't assume that each state has two races, you can't assume that they will both be primaries or caucuses, you can't assume they will be on the same day, you can't assume that the polls will close at the same time, and a myriad other complications that complicate sentence formation. Each of these ended up being yet another if statement within our templates. More specifically, when talking about the democratic and republican races in context of each other, one needs to account for the number of permutations for each state of the race (i.e. race has not started, polls have closed, race is called). This would result in an if statement for both races have not started, one race hasn't started and the other race has begun, one race is on a different day while the other has already been called, etc.

While complicated logic is only natural (and really ideal, as it creates more nuanced and human sounding stories), we need to have the right abstractions and tools to create these templates. Whether you use jinja2, handlebars, nunjucks, or whatever new fancy templating language comes out next, you'll run into these issues of needing to combine independent clauses in some situations and keeping them separate in others, or dealing with conventions of punctuation when working with complex clauses, and various other grammatical errors.

For the Olympics, we decided to take a fresh look at this problem. Natural langauge generation generally consists of 3 components: document planning, microplanning, and surface realization. Document planning is "deciding which information will appear in the output text" and "how chunks of content should be grouped in a document". Microplanning is "deciding which specific words should be used to express the content" and "deciding which expressions should be used to refer to entities". And last, surface realization "uses the rules of grammar to convert abstract representations of sen-

Figure 2: Sentence Syntactic Structure

```
{:verb=>"win",
 :prepositional_phrases=>
  [{:preposition=>"with",
    :rest=>{:noun=>"time", :determiner=>"a"}},
   {:preposition=>"of",
    :rest=>{"27 minutes 30.42 seconds"}}],
 :complements=>[],
 :subject=>
  {:noun=>"Mohamed Farah",
   :prepositional_phrase=>
     {:preposition=>"of", :rest=>"Britain"}},
 :object=>
     "track and field gold in the men's 10,000m"}
```

tences into actual text". This last component is where we ran into most of our issues with elections. We decided to use a library called SimpleNLG originally developed at the University of Aberdeen's Department of Computing Science. [6]

This library enabled us to create a recursive hash that represents a sentence's syntactic structure and feed it to the SimpleNLG realization engine (see Figure 2).

This sentence would be "realized" as: "Britain's Mohamed Farah wins track and field gold in the men's 10,000m with a time of 27 minutes 30.42 seconds, beating Galen Rupp of the United States by 0.48 of a second. Kenenisa Bekele of Ethiopia wins bronze with a time of 27:32.44."

Looking forward to the general elections, we are working towards a system that combines these SimpleNLG sentence hashes with the ease of simple templating for journalists. By breaking apart a story into subcomponents, we allow small pieces of stories to be reused for bots without having to generate separate templates per distribution channel. In addition, utilizing SimpleNLG, we can easily change the tense of a sentence depending on the medium.

## 3.4 Distributor

The final piece of the bot is distributing the generated text into our various channels.

### *3.4.1 One Way*

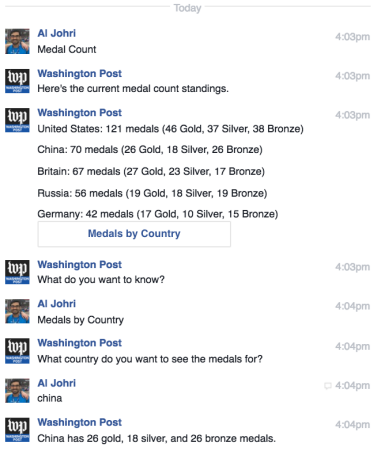
Distribution to one-way channels is straightforward. The distributor acts as a wrapper around the various APIs such as Twitter, Slack, Wordpress, etc. In more high frequency applications, it can form a queue to prevent rate limiting.

### *3.4.2 Two Way*

For two way channels, the distributor is an API that can be hit by Alexa or Facebook Messenger. Two-way communication can be more complicated becasue each channel may have different features that it supports. For example Facebook messenger, shown in Figure 3, will hit the API with the user's input as an atomic request, while other text-based platforms will support a user session with a users previous interactions as well. Voice-based distribution platforms introduce additional complications which we will discuss in section 4.3 because they take on the difficult task of speech recognition within the platform and provide results in a structure specific to the platform.

# 4. DISCUSSION

Figure 3: Facebook Messenger




## 4.1 Lessons from Implementation

Olympics and elections were good domains for deploying two-way communication interfaces because both involved a large amount of information from which the bot could surface only the relevant or timely information upon the user's request. The bots worked particularly well to supplement the work of journalists. For example, the Olympics bot posted medal events to the Washington Post Olympics liveblog, guiding the conversation for the live-blogging human journalists and allowing them to focus on the important analysis for the reader.

Structuring the knowledge in the raw data and building out templates for natural language generation was done as a collaboration between engineers, domain experts, and beat reporters who covered the Olympics, which resulted in a better experience for users. We encoded domain specific information such as if the event split into two medal rounds or one, what metric was used to score an event, and other formal and informal "verbage" for an event as well as event hierarchies. This allowed a user who asked for information about the 100-meter dash, to also be able to query other kinds of races.

We also found that extensive user testing was key with 2-way communication. A user can ask the system questions in many different ways. The system must be designed to handle all the intuitive ways a user might ask for certain information and prompt the user to understand the limitations of the application so that they do not get frustrated by asking for something it cannot provide.

We built these lessons from the deployment and implementation phase into our workflow for Heliograf. We are building out a user interface for Heliograf where journalists and engineers work together to build out templates for automated reporting. The user interface allows journalists to start working with the concepts of templating and conditional statements, and allows engineers to collaborate with journalists to determine how the knowledge will be structured and communicated.

## 4.2 Building for multiple Platforms

Building domain specific bots is a huge lift, so naturally we want to get the bot in front of as many users and on as many different platforms as possible. Each distribution channel

has different requirements for how a user communicates with it and in turn has different technical limitations. Attempting to develop for all platforms could compromise the quality of the product too much or require too many resources or too much time to execute. The key question then becomes which platform to develop for first. This is important because this decision will inevitably determine the structure of the bot. The platform we develop for first also can lead to a decrease in quality of the products on other platforms if support for those platforms is more of an afterthought than a decision embeded into the structure of the bot.

One key difference is between voice and text interfaces. Text based interfaces such as slack and facebook messenger will generally provide to the bot the exact input of the user, while voice-based interfaces such as Alexa generally take on the task of understanding the user's spoken language and converting it to something usable for the bot's developer. Machine understanding of spoken language works much better when many users provide examples of the same phrases, so rather than simply providing the bot with a transcript of the user's spoken language, Amazon has developed a system by which the bot developer defines "Intents" of the user and then creates templates for things the user can say that map to each intnet.

For example, if we wanted to allow for a user to query "What was Usain Bolt's time in the 100 meter dash", a text based interface could use entity extraction algorithms to identify "Usain Bolt" as a person and the 100 meter dash as an event and quickly return the relevant results. Using Alexa however, we would have to encode all the different ways that a user might ask that question and map them to an "Intent". Building for Alexa first then meant that we did not support that functionality.

### 4.3 Future Work

Despite the compelling technical and journalistic reasons to build domain specific bots that provide rich interaction and valuable information to the user, there is still a push by many newsrooms for general purpose bots that serve a similar purpose to a site's homepage or search bar, as an entry point to discover content generated by the news outlet. This is partly because news outlets are eager to experiment, to have a brand presence on these new platforms and to not get left behind. The platforms themselves also align incentives in such a way that makes having several domain specific bots less disirable for brand visibility than having one heavily promoted bot. Domain specific bots require a lot of effort, and sometimes only last as long as the news event that they intend to cover. In the short term, especially as this technology is nascent and the number of users are limited, this can mean putting a lot of effort into something that is not widely used and has an expiration date. In the long run however, these technologies can be reused for recurring news events and will improve with each iteration.

## 5. CONCLUSION

Heliograf's infrastructure is the result of the ongoing maturation of conversational bot technology in our newsroom. We have found that domain specific bots provide value to readers and newsrooms by allowing newsrooms to cover vast amounts of information that may otherwise not get coverage and by delivering relevant, timely, and personalized news to the reader. As automated journalism matures, many more newsrooms will go from their first few experimental bots to making automated journalism a regular part of their storytelling toolbelt. In the future we expect to see many different architectural styles and bot generation tools. We intend continue to improve the Heliograf platform and plan to utitlize it to make bots that cover news in various different knowledge domains.

## 6. REFERENCES


[1] System and Method for using Data and Angles to Automatically Generate a Narrative Story, January 2013. [Online; accessed 22-July-2016].

[2] AI is already making inroads into journalism but could it win a Pulitzer? http://www.aljazeera.com/programmes/listeningpost/2015/11/robot-journalism-human-reporters-151115095033120.html, November 2015. [Online; accessed 22-July-2016].

[3] Company Earning Previews. http://www.forbes.com/sites/narrativescience/#3ffcdea54f72, 2015. [Online; accessed 22-July-2016].

[4] Shevette Lavonne McGowan, 48. http://homicide.latimes.com/post/shevette-lavonne-mcgowan/, July 2016. [Online; accessed 22-July-2016].

[5] T. Adams. And the Pulitzer goes toâĂę a computer. https://www.theguardian.com/technology/2015/jun/28/computer-writing-journalism-artificial-intelligence, June 2015. [Online; accessed 22-July-2016].

[6] A. Gatt and E. Reiter. SimpleNLG: A realisation engine for practical applications. Technical report, Proceedings of ENLG-2009, 2009.

[7] A. Graefe. Guide to Automated Journalism. http://towcenter.org/research/guide-to-automated-journalism/, January 2016. [Online; accessed 22-July-2016].

[8] J. Holmes. AI is already making inroads into journalism but could it win a Pulitzer? https://www.theguardian.com/media/2016/apr/03/artificla-intelligence-robot-reporter-pulitzer-prize, April 2016. [Online; accessed 22-July-2016].

[9] J. Holmes. AP expands Minor League Baseball coverage. http://www.ap.org/Content/Press-Release/2016/AP-expands-Minor-League-Baseball-coverage, June 2016. [Online; accessed 22-July-2016].

[10] S. Klein. How To Edit 52,000 Stories at Once. http://homicide.latimes.com/post/shevette-lavonne-mcgowan/, January 2013. [Online; accessed 22-July-2016].

[11] C. Lecompte. Guide to Automated Journalism. http://niemanreports.org/articles/automation-in-the-newsroom/, September 2015. [Online; accessed 22-July-2016].

[12] J. Lichterman. Remember Facebook Messenger bots? The Washington Post just launched one (with a few bugs). http://www.niemanlab.org/2016/07/remember-facebook-messenger-bots-the-washington-post-just-launch
July 2016. [Online; accessed 22-July-2016].

[13] Quakebot. Earthquake: 3.9 quake strikes near Brawley, Calif. http://www.latimes.com/local/lanow/la-me-earthquakesa-earthquake-39-quake-strikes-near-brawley-calif-
html, June 2016. [Online; accessed 22-July-2016].