Council Chatbots

Example Shared Conversational AI Architecture

1. Introduction

The Example Shared Architecture describes the possible implementation of a conversational Artificial Intelligence platform that is shared between several councils. This report recommends how those councils might host, test and manage the platform.

This document is one of a group of reports resulting from the discovery research project “Can chatbots and AI help solve service design problems?”, in collaboration with 13 English councils.

All key project deliverables outline our findings in detail - please refer to our individual reports for more focused insights and information:

- ROI Analysis and Market Summary | April 2019 | Council chatbots | Torchbox
- Technology Landscape Review | April 2019 | Council chatbots | Torchbox
- User Research Summary Report | April 2019 | Council chatbots | Torchbox
- Case Studies | April 2019 | Council chatbots | Torchbox
- Project Summary Report | April 2019 | Council chatbots | Torchbox

A blog has been published by the project lead, Neil Lawrence of Oxford City Council. To read articles covering each stage of the project please visit the blog:

- https://localdigitalchatbots.github.io

Contents

1. Introduction 2

2. Overall Design 4

2.1 Example architecture diagram 5

3. Agreed Domain NLU Model 6

4. Testing Suite and Results 7

5. Centralised hosted NLU System 7

6. Rules and Conversational Flow Engine 9

7. Standard Domain Flows 10

8. API Gateway and Webservice integration approach 11

9. Log store 11

10. Advanced Analytics Engine 12

11. Content Management System 12

12. Hosted, Embeddable or Deployable Webpage 13

13. Channel Architecture 14

13.1 Normalised internal message format 14

13.2 Web Chat Widget 15

13.3 SMS 16

13.4 Facebook Messenger 16

14. Components not excluded from the diagram 17

14.1 Human Chat Management system 17

14.2 RPA 17

14.3 IVR integration 17

2. Overall Design

Each council could purchase a local system to develop their own, individual conversational Artificial Intelligence (AI) system. In this scenario knowledge becomes siloed, regional variations in training quality become high, and costs overall are much higher. We can see this sort of individual approach has resulted in a wide variety of quality and styles of websites across the councils participating in this project. Even for areas where the information provided is nearly identical in subject (if different in content), a user's online experience would still be very different moving from one side of the country to another.

An individual approach also disadvantages smaller councils with less buying power and fewer resources to optimise their systems. We can see this clearly in the data from the study; smaller councils have recorded roughly £6.00 per call, while larger councils record roughly £2.00 per call.

As we start to venture into conversational AI we can approach this differently to how the adoption of web technologies occured. Instead we can choose to collaboratively purchase and train a centralised system.

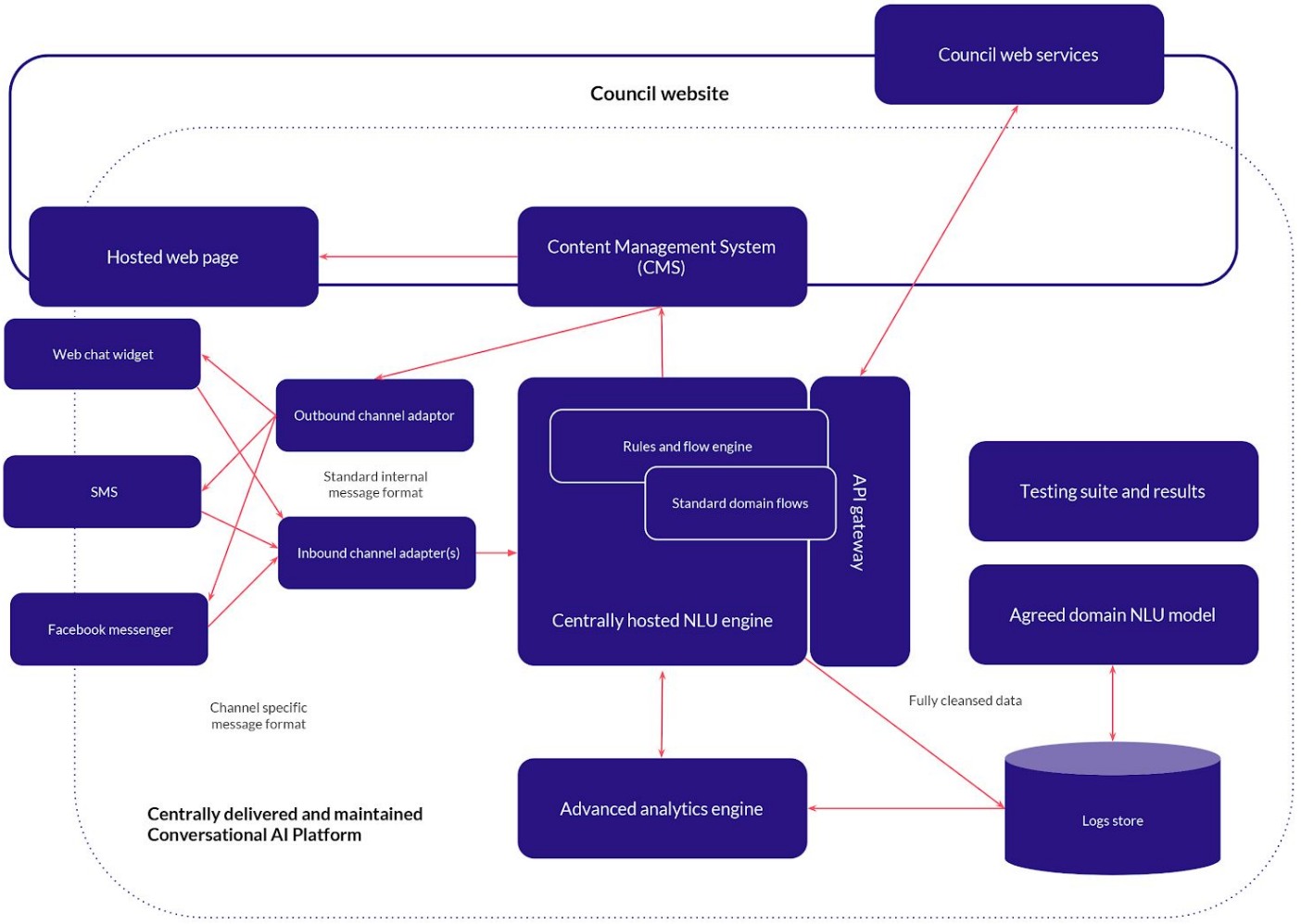
A centralised system would allow the councils to negotiate as a unit and purchase a more secure, more scalable system, with lower usage charges, and premium features which improve the user experience. Knowledge can be shared, high-quality Natural Language Understanding (NLU) continually maintained, and the costs for each individual council can be much lower.

The primary challenge in doing this is not a technical one, it is an organisational one. Councils must agree to co-operate and then collaborate in building an overall model for a research area that can adapt to the variety of ways of handling the chosen area, as well as adapt to the variety of terms used across England for that area.

If we can achieve that level of collaboration, what sort of system might we build to support a first, common conversational AI technology? How would that also allow councils to broaden out from there to build other collaborative or individual areas of conversational AI expertise?

The diagram below represents an example model that could be investigated to tackle this problem, with a section explaining the suggestion for each component within it.

2.1 Example architecture diagram



3. Agreed Domain NLU Model

The first step in agreeing the 'domain NLU model' would be to define the functions within the chosen area and produce the necessary NLU model. This model would need to be broad enough to cover the variety of regional accents and methods of service delivery for this area. For instance: dealing with bins in urban areas with many users in apartments who have shared bins is very different from a rural area, where each collection may be far apart. Council users in Lancashire may describe their bin needs differently from those in Brighton, however those needs, regardless of terminology or delivery method are by-and-large the same.

The intentions of users that would be supported within the NLU domain would need to be defined, and what represents a satisfactory outcome needs to be established. For instance, is giving a piece of information enough, or is a backend integration required to fetch the status of something the user is interested in, or to submit information on their behalf. A model of the responses needing to be configured for each possible user intention and the webservices necessary to successfully achieve that response would be necessary

Having established the scope of the domain NLU model, training examples supporting how users within each region speak about that domain could be sourced, in order to train the NLU system to better understand their requirements.

The domain NLU model would be used to train a single conversational AI platform, but should be stored in a generic way that could be accessible to, or be repurposed for, other platforms. Each of the leading conversational AI platforms is trained using a very similar set of data, and the storage of this data outside of any single tool would avoid being locked into any single provider.

4. Testing Suite and Results

As part of producing the NLU domain model the project could also produce a set of training data and test suites that measure the delivered accuracy and precision of the model using the chosen NLU platform. This measurement would be a valuable benchmarking asset in its own right, with which any potential technical solution in the domain area could be evaluated. It would set a standard for how to evaluate other chatbots in other council NLU domains and show how to provide test data sets and matching test suites in a transparent way. As technology in this area rapidly develops and the number of providers increases, individual councils would have a clear domain-specific benchmark and a quality standard for transparency to compare each possible solution or provider against.

5. Centralised hosted NLU System

This component would provide the intent classification (*what* action the user wants to achieve) and the named entity extraction (*the things* that the user wants to carry out the action upon).

Hosting the system could be public, cloud based hosting using a proprietary provider, or a privately hosted open source solution. Both the open source providers and the closed cloud based platform providers typically provide different service levels; those aimed at single chatbot functions for small organisations, and those aimed at larger organisations that need to host many segregated chatbots for different functions. The premium service levels tend to have greater security, high-availability features (for instance multi-region deployments to prevent failure if any single region were to go down, and so reduce latency within each geographical area), and data segregation options as

well as being setup to support a much wider scale of development than the standard levels.

For instance: the hosting system might have backend log storage on segregated virtual databases, rather than separated by user within a shared database. Segregated virtual databases provide a slight increase in security and allows an enterprise client to copy or reuse the whole database between projects and environments, rather than query only their user section. This also allows easier maintenance, many test and development environments, and the segregation of production and test data.

Some also have end user-facing and council staff managing functionality only available in the premium versions, for instance the ability to generate answers from stores of documentation without explicit configuration is typically only in the premium paid for tiers. (Long tail style functionality.)

Hosting a high quality, highly-available, automatically-scaling system for a large number of different chatbot deployments is a complex undertaking. Cloud-based providers regard conversational AI as one of the pivotal applications to encourage organisations to overcome their reluctance to move processing and hosting onto the cloud. Whilst an entirely open-source version of a centralised platform is attractive, it should not be regarded necessarily as the cheaper option, since the cost to establish this system at scale for multiple councils would be significant. Utilising an existing cloud-based proprietary platform built for this sort of scenario could be cheaper and provide a higher quality solution. Leading cloud-based providers are continuing to invest heavily in AI so the rate of change is fast and competition is high. Utilising one of the leading providers should help to future proof the chosen solution. However, by storing the training and testing data as Open Source items, mastered outside of any one cloud solution, many of the ideals of the Government Digital Service open source guidance can be maintained.

A detailed study of the two options: between keeping the domain model, training and test data open source and deploying in a cloud hosted model, or keeping the entire system open source and deploying in a custom, private cloud hosted model should be conducted.

6. Rules and Conversational Flow Engine

In implementing any chatbot technology, a decision would need to be made whether to use the chosen conversational AI platform's rules and conversation configuration methods or use an external solution.

If the chosen platform's configuration method is used whilst the underlying domain NLU model is highly transferable between technology providers, the example flows corresponding to the NLU model would still be largely platform dependent. So the understanding of what the user is saying could be moved easily between different platforms, but the control of the conversation responding to that would be more difficult to move.

If an external, open source rules engine and conversational flow configuration method was used then both this and the domain NLU model could be reused with any providers' platform.

Despite the perceived advantages of being tech agnostic, we recommend adopting the chosen platform's in-built solution for rules and conversational flow configuration. These platforms are rapidly iterating, and working closely with the supplier and technical community supporting them is highly necessary. Sticking closely to the recommended build patterns for each platform means that, while skills are still scarce, they are easier to find and onboard than using a custom rules and conversation configuration method. It also helps future proof the platform, as suppliers (generally) will support upgrades to technology built using their inbuilt/recommended tools, but if a fully external custom rules

engine and configuration method is used, this is unlikely to be considered as technology providers plan their upgrades. Creating a fully open source rules engine with good flow configuration tooling is a also a large and costly undertaking.

7. Standard Domain Flows

Within the platform selected, alongside the agreed domain model and trained NLU, a set of typical conversational flows to discuss and respond to each intent should be built and designed using the chosen rules and conversational flow engine (see above). The flows should then be matched up to a minimum set of integrations required for each flow for the chatbot to respond. An individual council should be able to select any or all of the flows to implement depending on the services, and integrations they are able to provide.

For example, the result of designing and building these flows would mean that; for the intent #replace_bin_lid there is not only a trained NLU model which understands the user request matches that intent, but also there is a pre-trained conversation to discuss the bin lid, ask the user for any relevant details, give example responses and there is a specification for all the configuration that must be done, and the webservices that must be provided to use this effectively (i.e to actually order the new bin lid for the user).

A council could therefore deploy the full standard functionality just by configuring the provided items: simply configure the standard responses and connected appropriate webservices without needing to conduct any additional NLU training or conversation flow design.

A council wanting to build new chatbots in other NLU domains would also have access to a very high quality system and examples of good quality implementations on which to base their work

8. API Gateway and Webservice integration approach

A standard model for configuring the call to external webservices from the NLU platform is needed, as the form of the webservices may vary from council to council for the same underlying flow.

An API Gateway that would securely connect to external web services and both transform different connection methods and returned fields for each council into a standard format for the NLU system, would enable a standard model.

9. Log store

Once the system is released to the public, it's necessary to rapidly iterate and train the system based on real user interactions with and responses to the bot. Training sets (of real user interactions) must be fully cleansed of all user data, so they can be published in an open source way, and placed within an automatic 'DevOps' pipeline, which moves changes when they have been tested automatically through development, test and production environments.

It is therefore necessary to store a large volume of user interactions in a secure and separate way to the development and test systems, whereby only a small number of specialised users have access to conduct detailed analytics and produce the cleansed training sets which can then be distributed through many hosted deployments.

A dedicated log storage function is required. Most of the platforms - as described in the Technology Landscape Review - provide an option which may

be suitable, and allow this segregation of the production system logs to a small number of users. This is more common in the providers providing automatic integration points to advanced analytics engines.

10. Advanced Analytics Engine

The leading conversational AI platforms tend to provide a variety of basic analytics and dashboards for monitoring the performance of the chatbot. Whilst this may be sufficient for determining basic usage and performance, they generally aren't enough for the detailed evaluation of user reactions, or for analysing in detail the performance of the conversational flows or NLU system.

A separate advanced analytics engine should be connected to the log store and NLU system. Most of the leading platforms provide recommendations for, or integrations to, advanced analytics engines.

11. Content Management System

One of the major benefits of a centralised approach would be for councils of all sizes to have access to an advanced conversational AI platform for building chatbots from scratch when they wished, but also to be able to implement pre-built domain functionality without requiring local conversational AI or Data Science skills.

To be able to make changes to pre-built content without dedicated conversational AI skills or access to the supporting platforms, a separated Content Management System (CMS) would be needed. Most of the conversational AI systems have a basic CMS that is closely coupled to the AI

functionality, however; they tend to be not well suited to maintaining multiple versions of responses across multiple organisations or formats.

Segregating the CMS functionality to use a modern 'headless' CMS system would allow individual councils to customise rapidly all responses of the bot, fully brand it to their needs, and to have a full content development and editorial control process segregated from the technical implementation.

A 'headless' CMS system is one which can be configured and used via API to support content models which aren't page centric, i.e. doesn't assume the content will be used only to support a webpage, but supports content models which work in other forms, such as chatbot utterances. A 'headless' CMS also typically supports a traditional page centric model using the same data via a user interface, therefore councils can have a single headless CMS each which serves both website content and chatbot utterance needs.

A segregated system, where the CMS is housed separately to the AI platform and connected via an API, can be used to ensure that answers supplied by a chatbot across a council's website, mobile app, or other integrated channel, remain aligned. This is very important to ensure that a user doesn't receive different answers depending on the channel they visit. In areas that may have quick updates or rapidly changing content, for instance bin collection days after bad weather, it's important that there is a simple method for changing content and knowing it is replicated across all the places that content is displayed.

12. Hosted, Embeddable or Deployable Webpage

To help ensure there is consistency across council webpages and that chatbot functions are matched to the right data in the CMS system, as well as minimise the changes required to a council website design to implement web chat

widgets, an embeddable, hosted, or otherwise easily deployable webpage could be centrally provided as part of the system. This would be pre-integrated with the CMS to provide a high quality, modern webpage with branded, matching information across the chatbot and website, just by a configuration within the CMS. An embedded or deployable webpage would also automatically provide the webchat widget needed to send information from and to the chatbot while being connected to fully configured modern web analytics so that the mixture of usage across the website and chatbot widget could be tracked. Alternatively, councils could design their own webpage that pulls content from the provided headless CMS (or their own)and only use the provided page as an example of what features they might want to support.

13. Channel Architecture

13.1 Normalised internal message format

The conversational AI system will need to use a standard message format that defines how to process text and, if supported as part of the centralised project, other media or rich message components such as emojis, images, videos, gifs, buttons, option pickers, carousels, and URLs. Unfortunately the different channels a council might want to use to interact with their users (e.g. social media, website or app) use differing message formats. This means an internal message format must be defined to 'normalise' the messages received and sent, and channel-specific adaptors need to be implemented to convert the message to and from the channel-specific format into the defined internal message format.

Each adaptor needs to change the message to both undersatnd an use the content in the best way on any given channel. Some channels support only basic text like SMS, therefore buttons and images need to be replaced with numeric options and alternative text. Other channels support rich content like option

pickers, visual cards, gifs or emojis. A single piece of content needs to be reusable across them all in the best way for that channel to maximise the value from the system.

Having trained the system to support text only channels and a rich messaging channel, expanding to other channels which lie somewhere between is relatively quick, typically requiring only a new channel adaptor and not a fundamental rework of the underlying system.

For the initial system it's recommended to plan for three channels. A rich web channel using a chat widget, a text only channel requiring no smart device or app install like SMS, and a channel which uses a free social platform user interface and is close to any council social media presence.

Pre-built versions of all of these channel widgets or channel user interfaces are available, which allow customisation of the widget's look and feel and so removes the need to build a custom one. For example, the Facebook messenger phone and web app already exist and are customisable.[1] 'Out of the box' Web Chat widgets are also readily available. Whilst a custom user interface for all of these could be rebuilt using the underlying APIs or messaging standard, it is not recommended not to invest in rebuilding the end channel user interfaces, but instead invest in training the conversational AI system to utilise them in the most effective way.

For an initial launch, the example system could support the following channels:

- Web Chat Widget
- SMS
- Facebook Messenger

Each channel is described further below.

---

1 https://blog.messengerdevelopers.com/https-blog-messengerdevelopers-com-how-to-customize-the-customer-chat-plugin-336b6b60ca3

13.2 Web Chat Widget

A modern webchat widget from an external hosted source which supports rich content and can be easily implemented on any webpage using a small amount of javascript and customisation. For example, here is the Javascript to implement the Intercom.io webchat widget and is similar to the way Facebook, or LivePerson or many other hosted configurable UIs are added to websites.

```
<script>

  var APP_ID = "APP_ID";

 window.intercomSettings = {

    app_id: APP_ID

  };

</script>
```

```
<script>(function(){var w=window;var ic=w.Intercom;if(typeof ic==="function"){ic('reattach_activator');ic('update',w.intercomSettings);}else{var d=document;var i=function(){i.c(arguments);};i.q=[];i.c=function(args){i.q.push(args);};w.Intercom=i;var l=function(){var s=d.createElement('script');s.type='text/javascript';s.async=true;s.src='https://widget.intercom.io/widget/' + APP_ID;var x=d.getElementsByTagName('script')[0];x.parentNode.insertBefore(s,x);};if(w.attachEvent){w.attachEvent('onload',l);}else{w.addEventListener('load',l,false);}}})();</script>
```

13.3 SMS

SMS doesn't require a smartphone or an app installation. Supporting it in initial design ensures that full natural languages are supported, and that the system can be operated using text only. Whilst a number that typically falls within a users messaging allowance can be provided so it doesn't cost the user to message the council, each SMS incurs a charge on top of the NLU system usage for the council to send each response. Central procurement of SMS in bulk

provides for discounts and cost efficiencies based on both volume and for committed monthly minimums.

13.4 Facebook Messenger

Council services are widely discussed on a variety of social media platforms. Supporting at least one of these platforms brings AI-enabled advice nearer to where people discuss the problems they face. These channels don't charge per usage and support rich messaging. Many phone users will have the required apps pre-installed allowing for instant usage with rich messaging features.

14. Components not excluded from the diagram

A variety of other components can be useful when considering a conversational AI platform that were not included on the example architecture diagram. When looking at next stages implementing some of the following may be of use:

14.1 Human Chat Management system

If the bot is going to have the option to handover within the chat to human user, it's recommended to include a human chat centre management system. The bot should be trained to summarise and handover to a human operative when a bot interaction is not successful or not the best solution for that user or topic.

How necessary and what type of human handover is required depends on the domain of knowledge tackled and the coverage and quality of the bot training.

14.2 RPA

If a council has multiple manual steps without webservices a Robotic Process Automation (RPA) system may be useful for recording and replacing necessary human interactions and wrapping them in a suitable webservice.

14.3 IVR integration

To achieve high levels of call deflection in general it will be required to integrate into any existing call centre interactive voice response (IVR) system to either make users aware of the bot service, or redirect them to the bot service depending on how firmly the council wishes to steer the user.