Architecture of a Scalable, Secure and Resilient Translation Platform for Multilingual News Media

Susie Coleman[♢], Andrew Secker[♢], Rachel Bawden[♣], Barry Haddow[♣], Alexandra Birch[♣]
[♢]British Broadcasting Corporation, UK
[♣]School of Informatics, University of Edinburgh, UK
susie.coleman@bbc.co.uk, andrew.secker@bbc.co.uk, bhaddow@inf.ed.ac.uk, {rachel.bawden, a.birch}ed.ac.uk

Abstract

This paper presents an example architecture for a scalable, secure and resilient Machine Translation (MT) platform, using components available via Amazon Web Services (AWS). It is increasingly common for a single news organisation to publish and monitor news sources in multiple languages. A growth in news sources makes this increasingly challenging and time-consuming but MT can help automate some aspects of this process. Building a translation service provides a single integration point for news room tools that use translation technology allowing MT models to be integrated into a system once, rather than each time the translation technology is needed. By using a range of services provided by AWS, it is possible to architect a platform where multiple pre-existing technologies are combined to build a solution, as opposed to developing software from scratch for deployment on a single virtual machine. This increases the speed at which a platform can be developed and allows the use of well-maintained services. However, a single service also provides challenges. It is key to consider how the platform will scale when handling many users and how to ensure the platform is resilient.

Keywords: machine translation, AWS, platform, news media

1. Introduction

1.1. Media Context for NLP services

News does not only break in one language, and many modern large news media organisations seek to publish their material in multiple languages. These large news media organisations are, therefore, often working in a multilingual space. The BBC in the UK publishes news content in 40 languages and gathers news in over 100. The BBC distributes content on multiple platforms: radio, 24-hour TV and online video, audio and text content. The BBC's flagship Arabic and Persian services operate 24-hour TV news channels while other language services, including Kyrgyz, French, Russian, Ukrainian, Pashto, Burmese, Hausa and Tamil, also broadcast daily TV news bulletins via regional partner stations. All foreign language services publish news online. This is extremely important to promoting the reach of the news published by these world services, especially to under-served audiences.

Publication of news in multiple languages comes under the general category of *content creation*. With increasing language reach comes increasing demands on journalists' time. One way in which efficient use is made of journalistic endeavour is the republication of news originally authored in one language into another. An underused but key element to supporting journalists undertaking this task is Machine-Assisted Translation (MAT). With the appropriate user interfaces provided to support the translation step, a journalist is able to take a news story or script, in the case of an audio or video report, and quickly translate the original text using an automated technique. This translation is then manually edited to ensure it is in a state which is of sufficient quality. No matter how good the Machine Translation (MT), this will always be an important step for media organisations such as the BBC where quality is paramount; the reversioned content is usually prepared to be published in a different geographic region from where it originated, and therefore local knowledge, assumed geographical or cultural knowledge and colloquialisms must be expunged or explained in the translated copy.

The second application area, that of *news gathering* in multiple languages is supported via *media monitoring*, the (predominantly manual) monitoring of the world's media across video, audio, printed and online sources. In the current workflow, expert monitors and journalists have to perform a lot of manual work to keep up with broadcast and social media streams of data. Considering the huge growth in the number of streams of data that could potentially be monitored, the current processes fail to scale adequately. It is becoming imperative for technology to be used in this process to automate tasks, such as translation, in order to free monitors and journalists to perform more journalistic tasks that cannot be achieved with technology.

MT, the core of MAT, is an increasingly important technology for supporting communication in a globalised world and the use cases above illustrate how the News Media is an ideal candidate for promotion of efficiency through the use of MT.

However there exist significant gaps in the language pair coverage when considering supporting the BBC's multilingual news-gathering and dissemination operations. The European Union's Horizon 2020 research and innovation GoURMET project (Birch et al., 2019) aims to significantly improve the robustness and applicability of Neural MT (NMT) for low-resource language pairs and domains. Tangible outcomes of this process include the production of NMT models in 16 different language pairs. The BBC and Deutsche Welle (DW), the media partners on the project, will then use these in tools for journalists.

The outputs of the project will be field-tested at partners BBC and DW by inclusion in tools and prototypes (Secker et al., 2019b), and evaluation of the translation's utility in these real-world situations. A formal data-driven evaluation

Proceedings of the 1st International Workshop on Language Technology Platforms (IWLTP 2020), pages 16–21
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

will also be undertaken (Secker et al., 2019a).

1.2. Related work

The SUMMA project[1] (Scalable Understanding of Multilingual MediA), ran from 2016 to 2019. The aim of SUMMA was to significantly improve media monitoring by creating a platform to automate the ingest and analysis of live news media. SUMMA integrated stream-based media processing tools (including speech recognition and machine translation) with deep language understanding capabilities (including named entity relation extraction and semantic parsing), and was implemented in use cases at the BBC and DW. Content was ingested in eight non-English languages plus English. All ingested content was translated into English for the purposes of analysis and as such, translation formed a key part of the process. SUMMA built evidence of the need for automated translation technologies to support news-gathering in modern media organisations, motivating GoURMET. The SUMMA platform was also built around micro-services but it was designed to be easy to install locally. It was deployable on the cloud but did not use native cloud scaling capacity as standard practices have evolved since.

The Elitr project[2], another EU project which is currently running, is building the European Live Translator platform (Franceschini et al., 2020). The aim of this project is to create an automatic translation and subtitling system for meetings and conferences. The platform used in Elitr builds on a platform developed by PerVoice[3], and also in an earlier EU project EU-Bridge[4]. The Elitr platform is optimised for real-time transcription and translation, and the transmission of audio and video data in addition to text. Due to the more demanding communication requirements, the Elitr platform has a custom data transmission protocol, and a C API which all components must implement.

1.3. Translation platform

In order to make the translation models available for use in such prototype tools, the BBC created a single platform in which they can be hosted, run and accessed. This single platform can then support numerous prototype tools, across multiple project partners, and with the correct provision around security and mediation of access by 3rd parties. The advantages of locating the translation technology in one place and then mediating access onto that via a service is this provides a single point for maintenance and updates. In contrast, if each prototype (tool, experience, etc.) has the translation technology integrated, an update to the translation technology (whether that be an improvement to the translation models themselves, a bug fix, a security improvement, etc.) must be undertaken numerous times.

There are a number of requirements for a translation platform that would not be present, or present in a different form, if the translation technology were built into each individual prototype or tool.

- **Scalability.** Since the platform now provides a single point of access, the platform must scale in response to the number of requests made from a variable number of tools and their users. Scaling must be automatic and reactive based on incoming request load.
- **Resilience.** With the platform now representing a single point of failure, it must be robust with automatic detection of failure and the ability to seamlessly move the servicing of incoming requests away from a failed process in a manner such that the end user sees no break in service.
- **Security.** As the translation service is accessed by multiple users, access must be secured to authorised parties and thus parties must identify themselves when submitting requests. It is also prudent to ensure that no one party can overload the platform with requests. While the service is required to scale, resources are not infinite and as such it is reasonable for the service to gracefully decline an unreasonable rate of requests from a single party in order to maintain a reasonable level of service for others.
- **Continuity**. It must be straightforward to apply updates to the service and these should not result in a noticeable break in responsiveness from the user's perspective.
- **Standard access**. External input and output to the translation modules occurs via a well defined API.

The effort required to create the above platform, successfully addressing the above considerations in the implementation, are enormous. Cloud computing platforms have developed around the need for institutions to create and maintain such platforms. Amazon Web Services (AWS) has emerged as a platform offering a diverse selection of cloud-based tools. Presented here is the architecture of the translation platform, implemented by the BBC on behalf of the GoURMET project and built entirely on the AWS platform. Whilst AWS is perhaps most known as a provider of Virtual Machines (VMs) via the EC2 product,[5] the development of a monolithic system for deployment on one or more VMs still requires considerable software development effort. In contrast the platform described herein combines a set of individual AWS products such that each manages a separate facet of the platform. Software development work is therefore minimised and limited to the setup of each AWS component and software engineering required to allow components to communicate.

The remainder of this paper describes the architecture of the platform. First the overall architecture of the platform is presented, then the implementation and customisation details of each major component are detailed. Finally, considerations around security, scalability of the platform and the deployment of MT models is covered.

[1]Funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 688139.

[2]Funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 825460

[3]http://www.pervoice.com/en/

[4]https://www.eu-bridge.eu/

[5]https://aws.amazon.com/ec2/

2. System Architecture of the Translation Platform

This platform is hosted on AWS and uses pre-existing services and components to create an architecture which is secure, robust and able to scale. By using the services that AWS offers it is possible to build and host a platform that combines existing components to build a solution rather than building the platform from scratch. This increases the speed of development and allows technology to be used that is already well tested and documented.

The other strength of AWS is the AWS Cloudformation[6] service which allows infrastructure to be defined using Cloudformation templates. A Cloudformation template is an example of infrastructure as code. By defining a template, a record of infrastructure is created that can be version controlled, provides visibility of the architecture and allows the system to be easily recreated from scratch.

There are multiple cloud services available that could have been used to implement this Architecture including Google Cloud Platform[7] and Microsoft Azure[8]. When choosing a platform cost, efficiency and reliability of services as well as the level of experience a development team has with a specific platform should all be considered. At the BBC AWS is the dominant provider of cloud computing services so when building a new service AWS is the default choice. As developers at the BBC are most familiar with AWS using it for new projects allows faster development times and more efficient debugging. However, it is still important to assess the feasibility of other platforms to ensure there is not a compelling reason to switch to an alternative.

The architecture required to fulfil a translation request is shown in Figure 1. A translation request will be initially handled by AWS API Gateway, which is the user facing part of the architecture. The request is then passed to an AWS Lambda, which acts as a bridge to a AWS Load Balancer, which will route traffic to the correct translation model running in AWS ECS (Elastic Container Service). The model in ECS will perform the translation and the response travels back up the stack to be served to a user by API Gateway. A more in depth explanation of the roles of the specific components is outlined in the following subsections as well as the user facing interface to the platform.

2.1. User Facing Interface

The platform is exposed via a RESTful API. The purpose of an API (Application Programmer Interface) is to expose a resource to developers to allow services and applications to make use of that resource (De, 2017). In the case of GoURMET, that resource is MT models. The goal of the API is to provide a consistent and logical interface that abstracts away from the specifics of how the MT models are implemented.

The API accepts and returns JSON objects, which is enforced by the Content Type HTTP Header. To translate text, a POST request is made to the API where the body of the request is a JSON object that specifies the source language, target language and text to translate. The text to translate must only use UTF-8 characters and can be multi-line text providing that it is escaped appropriately to still be valid JSON.

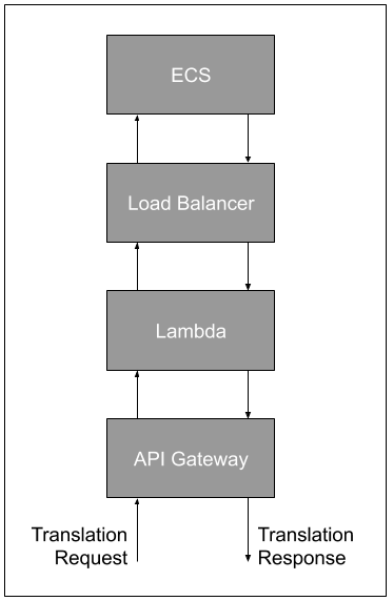

Figure 1: Architecture to fulfil a translation request

2.2. AWS API Gateway

API Gateway[9] is an AWS managed service for creating APIs. The service is used to manage exposure of the MT models to the public internet as outlined in the previous section. In this case, the API is a REST API where the interface is defined using Swagger.[10] As API Gateway is a managed service, it is easy to dynamically scale the API depending on traffic, and the service already implements multiple features for security, resilience and API life-cycle. This makes development of the user facing interface far quicker than starting from scratch.

2.3. AWS Lambda

AWS Lambda[11] offers serverless technology, which removes the need to maintain a server in order to run code. This makes it ideal for running short-lived tasks. A Lambda is created only when there is a need to execute the code and destroyed when that need no longer exists. This is good for both cost and dynamic scaling of services.

In the translation platform, the role of the Lambda is to route traffic from API Gateway to the Load Balancer.[12] This allows the Load Balancer to live within a private network and not be exposed to the public internet. This means

[6] https://aws.amazon.com/cloudformation/
[7] https://cloud.google.com/
[8] https://azure.microsoft.com
[9] https://aws.amazon.com/api-gateway/
[10] https://swagger.io/resources/open-api/
[11] https://aws.amazon.com/lambda/
[12] https://aws.amazon.com/elasticloadbalancing/

that traffic to the Load Balancer and, by extension, the MT models is controlled and managed via API Gateway.

2.4. AWS Load Balancer

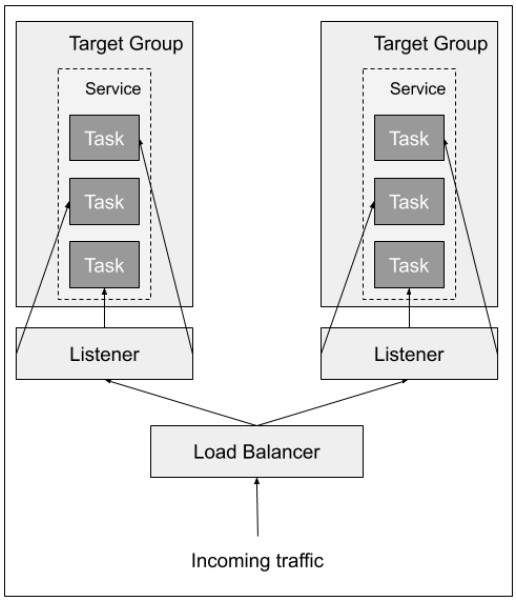

Figure 2: Routing traffic with a Load Balancer

The Load Balancer functions at the application layer of the OSI (Open Systems Interconnection) model. It listens for traffic on specific ports and will route traffic to a Target Group based on the port number as shown in Figure 2. The Target Group is made up of services that can receive the traffic. In this case, these are AWS ECS Tasks within an ECS Service. The Load Balancer will balance incoming traffic across all tasks within the Target Group.

2.5. AWS ECS - Elastic Container Service

The role of ECS[13] in the system is to run containers that contain the MT models.

All MT models are delivered as Docker images, which are definitions of how to create a container. This definition includes, but is not limited to, the operating system, programs installed, ports exposed, environment variables and file system. Containers provide an isolated environment for an application to run in, as defined in the Docker image. Multiple containers can run on a single physical machine to allow an efficient sharing of resources. ECS is a service to manage how containers run as well as the infrastructure they will run on.

The specific architecture of ECS is shown in Figure 3. An AWS Cluster has been created which defines the infrastructure the containers will run on. In this specific instance, AWS Fargate is used as it removes the requirement to manage EC2 instances. An AWS Task Definition has been created for each MT Docker image. The Task Definition defines the properties of a container. This includes but is not limited to which Docker image to use and where to pull the image from, how much CPU power and memory to allocate the container and any AWS IAM Roles[14] the container needs. Containers created using the Task Definitions are referred to as Tasks in AWS. The Tasks have been created within a Service. The Service maintains a specified number of instances of a Task and allows for the number of Tasks to be scaled up or down according to load on the system. The Service also health checks Tasks and destroys and replaces unhealthy ones.

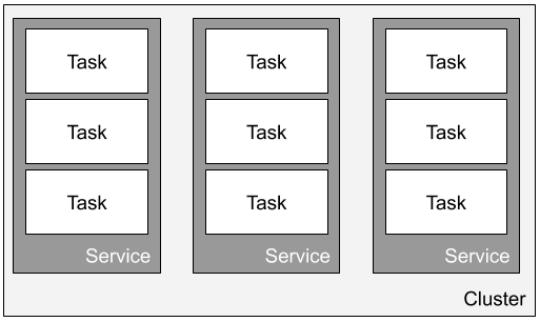

Figure 3: Architecture of an ECS Cluster

3. Security, Access Management and Request Rate Limiting

Security is important for any service available on the public internet, as the service is vulnerable to attacks from malicious users. In the case of an MT platform there is no sensitive information that could be exposed if an API key was to become compromised, therefore the biggest risks stem from DDoS (Distributed Denial of Service) attacks. A DDoS attack overwhelms a system with requests to stop it being able to handle legitimate requests. When using AWS, it is important to also consider the financial costs that can be caused by an insecure service. In this case, the scalable nature of the architecture would make it possible to require the services to use a large number of additional resources to handle malicious traffic increases, if the service is not properly secured.

The user facing API is secured:

- Using HTTPS: All traffic is served over HTTPS by default with API Gateway managing the certificate.
- Using API Keys
- Using Usage Plans: Usage plans are tied to specific API keys and add a throttling limit and quota limit.
- Sanitising Input: Ensure required request inputs are included and that the body matches the JSON schema and request model.

[13] https://aws.amazon.com/ecs/

[14] https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html

- Limiting payload size: API Gateway has an upper payload limit of 10 MB[15]. This allows more than enough capacity to handle translation of news articles whilst setting an upper limit to prevent malicious attacks where the system is overloaded by requests with large payloads.

AWS's API Gateway service was one reason to favour AWS when building this platform. API Gateway provides all of the security features used to secure the API. This means that the in-depth knowledge needed to implement security features is not required as they do not need to be implemented from scratch. Furthermore, API Gateway is widely used in industry and is actively maintained which means that security flaws are detected early and patched quickly.

4. Scalability

It is important to have a service that can scale dynamically in response to changes in volume of traffic. There are two points of scalability in architecture shown in Figure 1.
The first is using ECS. A Service determines the number of instances created from a specific Task Definition that are running at any one time, and this allows the translation platform to scale up to accommodate more traffic. The Load Balancer will distribute the traffic amongst the growing number of Tasks available to fulfil the requests.
The other point of scalability is the Lambda. Serverless technology is designed to be flexible, as it is not the responsibility of the Lambda creator to define the hardware it will run on or to ensure sufficient compute resources are available for it to run. As a result, Lambdas can be automatically initiated as needed to handle increases in traffic without the need to predict traffic spikes and provision hardware to handle these. The API Gateway is able to handle large amounts of traffic hitting the translation platform and using Lambdas to fulfil these requests allows the system to handle these traffic increases.
The final consideration regarding scalability is the ability to automate this scaling. AWS provides Cloudwatch Alarms to monitor and automatically respond to changes in the system under monitoring. This allows the translation platform to alert in response to changes in traffic and use of resources. These alerts can be used to handle these changes without manual intervention.

5. Deploying MT Models

A key consideration for this system is how to provide flexibility for creativity in research to allow novel approaches to MT whilst still building a consistent production service. This was achieved by using Docker. A standard template project was agreed for producing a Docker image for each MT model. This template consisted of:

- A Dockerfile
- A simple Python Flask app with a root endpoint and translate endpoint
- An integration Python script

The integration script contains an `init` function that is called when a container is created from the Docker image and a `translate` function that is called whenever a translation is performed. It is the responsibility of anyone implementing an MT model to implement these two functions. This allowed a consistent interface to all translation models whilst still keeping the actual implementation of the model agnostic.
The Docker images created are hosted on AWS using the AWS Elastic Container Registry.[16] Docker images can be hosted as repositories on any service that provides a Docker registry. Registries provide a central place to store images and the ability to only allow authorised users access to those images. Repositories use tags to allow the images to be version controlled. When a Task is created in ECS, the image is pulled from ECR.

6. Summary

Modern large news media organisations exist in a multilingual environment. MT technologies can be used to promote efficiency in such organisations for both news-gathering and publication in multiple languages. In order to support multiple tools which require translation as a fundamental, a platform providing translation as a service is the preferred solution. This paper describes an architecture for the creation of such a platform using components provided by AWS. The requirements for such a platform were described and the tools available from AWS which realise the required functionality were detailed. The platform described herein will form the basis of a selection of tools and prototypes to be tested in the BBC and DW as well as supporting further formal evaluation of the underlying MT systems.

7. Acknowledgements

This work was supported by funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825299 (GoURMET). It was also supported by the UK Engineering and Physical Sciences Research Council (EPSRC) fellowship grant EP/S001271/1 (MTStretch).

8. Bibliographical References

Birch, A., Haddow, B., Titov, I., Barone, A. V. M., Bawden, R., Sánchez-Martínez, F., Forcada, M. L., Esplà-Gomis, M., Sánchez-Cartagena, V., Pérez-Ortiz, J. A., Aziz, W., Secker, A., and van der Kreeft, P. (2019). Global under-resourced media translation (GoURMET). In *Proceedings of Machine Translation Summit XVII Volume 2: Translator, Project and User Tracks*, pages 122–122, Dublin, Ireland, August. European Association for Machine Translation.

De, B. (2017). *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. Apress.

Franceschini, D., Canton, C., Simonini, I., Schweinfurth, A., Glott, A., Stüker, S., Nguyen, T.-S., Schneider, F., Ha, T.-L., Bojar, O., Sagar, S., Macháček, D., and Smrž, O. (2020). Removing European Language Barriers with

[15] `https://docs.aws.amazon.com/apigateway/latest/developerguide/limits.html`

[16] `https://aws.amazon.com/ecr/`

Innovative Machine Translation Technology. In *Proceedings of the 1st International Workshop on Language Technology Platforms*.

Secker, A., Birch, A., van der Kreeft, P., and Sánchez-Martínez, F. (2019a). GoURMET deliverable D5.1 – evaluation plan.

Secker, A., Wall, J., van der Kreeft, P., and Coleman, S. (2019b). GoURMET deliverable D5.2 – use cases and requirements.