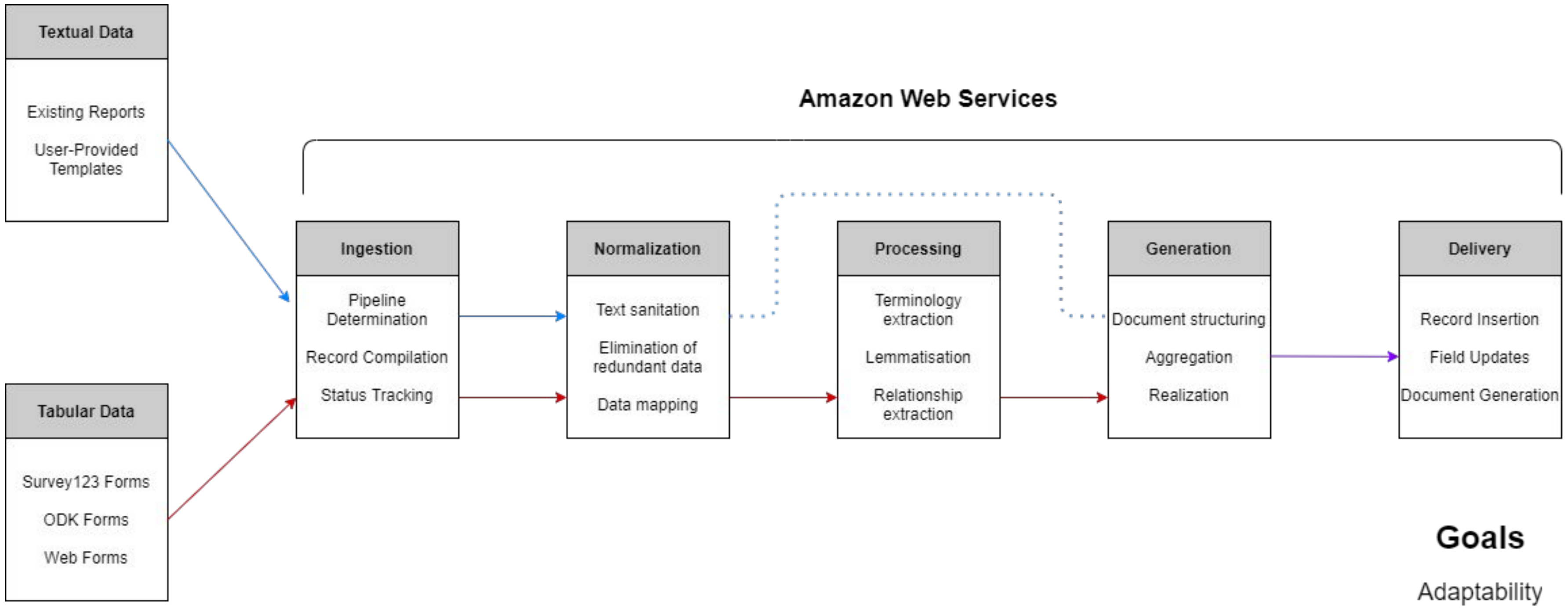
Overview of NLG, NLP Development

Agnes Folga

Primary Objectives

- Create an end-to-end solution that ingests tabular data and returns complete, project-specific sentences.
- 3 main goals:
  - Adaptability: codebase must be easy to expand and update
  - Reusability: codebase must be able to handle different use cases
  - Modularity: codebase needs to have independent components that are easily replaced

Initial Needs Analysis



Goals

Adaptability

Modularity

Scalability

Getting Started

- To build our solution, we needed a robust dataset. We chose to use artifact data collected by the Albuquerque office.
  - We obtained a copy of their field data and a report saved on a Word document and began working on a script to extract artifact information
  - We also pulled tabular data from their PostgreSQL database

Field Data Extraction

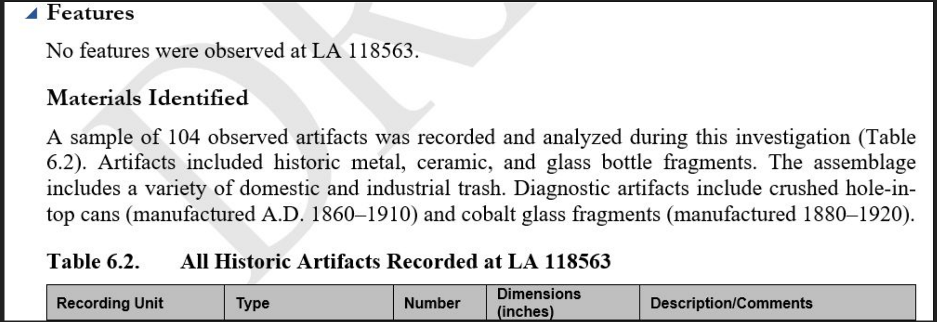
- Queried the Albuquerque database using PostgreSQL and was able to extract artifact data
- Only looked at prehistoric artifact data
- Only used list fields (not fields where users could type a response)

Report Data Extraction

Created a Python script to extract “Materials identified” section out of each site

Used Docx library, and had to separate by section headings

All data extracted was written to a json file



Features

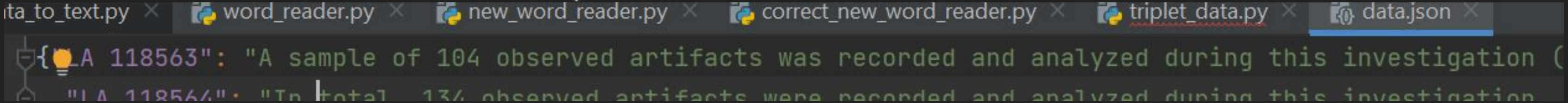
No features were observed at LA 118563.

Materials Identified

A sample of 104 observed artifacts was recorded and analyzed during this investigation (Table 6.2). Artifacts included historic metal, ceramic, and glass bottle fragments. The assemblage includes a variety of domestic and industrial trash. Diagnostic artifacts include crushed hole-in-top cans (manufactured A.D. 1860–1910) and cobalt glass fragments (manufactured 1880–1920).

Table 6.2. All Historic Artifacts Recorded at LA 118563

Recording Unit	Type	Number	Dimensions (inches)	Description/Comments

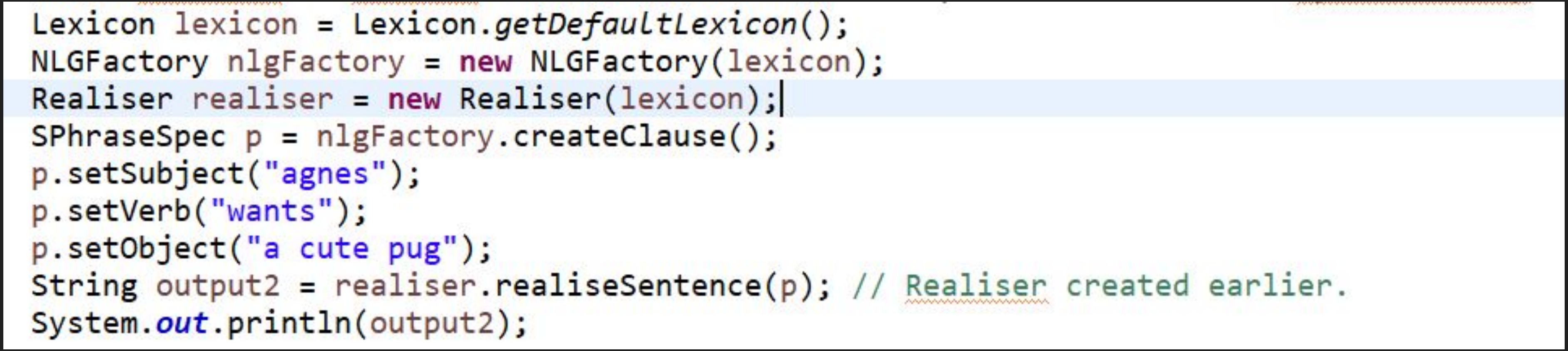


```
{"LA 118563": "A sample of 104 observed artifacts was recorded and analyzed during this investigation (
  "LA 118564": "In total, 134 observed artifacts were recorded and analyzed during this investigation
```

University of Aberdeen: SimpleNLG

- A Java library for NLG
- Found a GitHub repository with examples and documentation
- Is a realiser : Generates texts from a syntactic form
  - Each part of a sentence needs to be explicitly declared in the code.
  - allows the developer to predetermine things like conjugation, pluralization

Example of SimpleNLG



```
Lexicon lexicon = Lexicon.getDefaultLexicon();
NLGFactory nlgFactory = new NLGFactory(lexicon);
Realiser realiser = new Realiser(lexicon);
SPhraseSpec p = nlgFactory.createClause();
p.setSubject("agnes");
p.setVerb("wants");
p.setObject("a cute pug");
String output2 = realiser.realiseSentence(p); // Realiser created earlier.
System.out.println(output2);
```

Output:“Agnes wants a cute pug.”

Downsides of SimpleNLG

- question arose how to use the API in order to fit our needs
- Issue: rule-based (not Machine Learning based)
  - they've written code that explicitly defines proper grammar and syntax
- Rule based: no training, would have to explicitly create conditionals for all variations of your data

OpenAI: GPT-2

- pretrained unsupervised transformer based model
- trained to generate text by predicting the next word in a sequence of tokens
  - Also does translation, text summarization, answer prediction

Downsides of GPT2 (OpenAI)

- 2 main issues:
  - the model is closed source, so we couldn't add to it or see how it works
  - it was geared towards text-to-text, so we couldn't feed it a collection of values.
- GPT-2 is a general purpose AI. It is not intended to be customized for specific tasks and needs (decoder only)
- when you give a piece of text to GPT-2, the decoder infers what parts of the texts are important and what kind of output you want out. So if you ask "Is an orange a fruit?" GPT-2 determines which words are important ("is","orange","fruit") and then based on its training data decides what kind of response you want back. ("Yes, an orange is a fruit.")

Our selected model

- Google T5 model: a transformer based model that uses text-to-text
- Lot of documentation, high BLEU score
    - Tokenization: splitting the words into tokens
    - A BLEU score is a quality metric assigned to a text which has been translated by a Machine Translation engine
- Pretrained on Common Crawl's C4 Corpus
- Code base: https://github.com/MathewAlexander/T5_nlg

Differences between GPT-2, T5

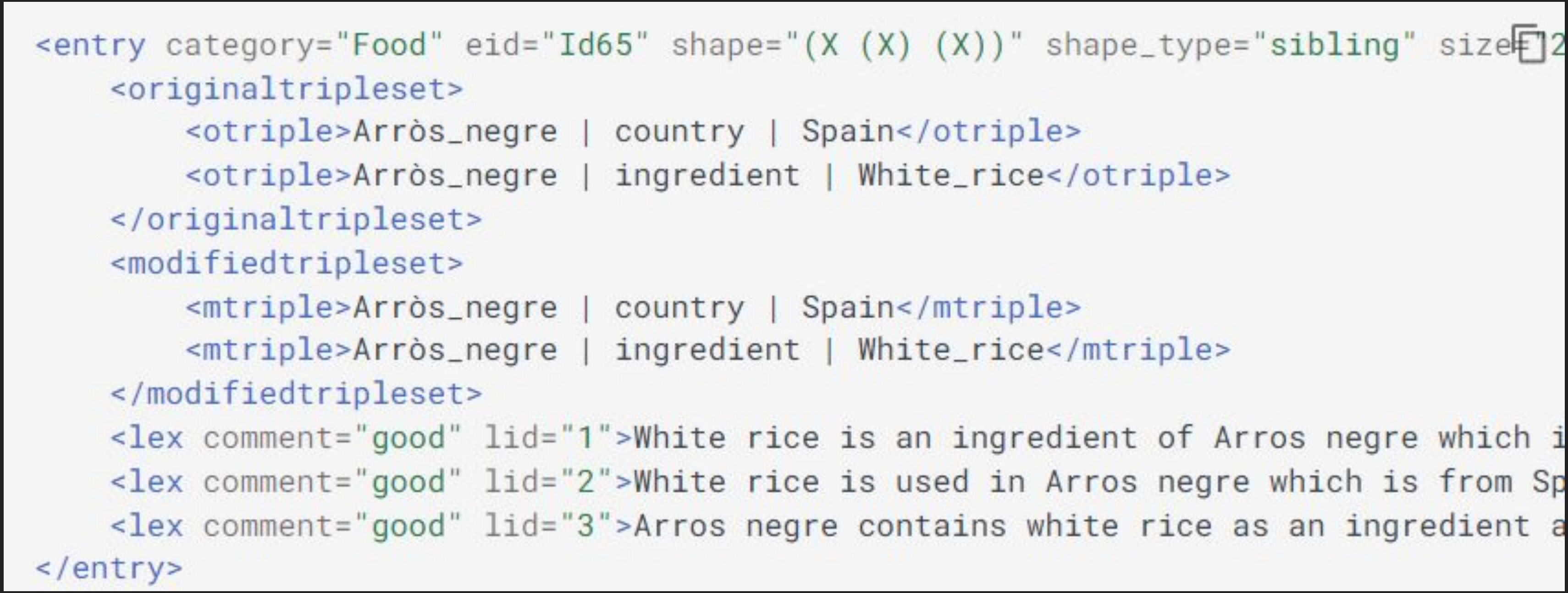
- GPT-2 is decoder only, T5 is encoder and decoder.
- T5 includes a encoder, which is what we uses our triples. The encoder lets us tell the model what words are important when we send it data and how we expect the data to come out.

WebNLG

- In our research into the T5 model, we found a simple example code called Quick NLG that we adapted to our own purposes. Quick NLG is what uses the WebNLG corpus which is derived from DBPedia.
- WebNLG is an ongoing coding competition that challenges teams to create NLG solutions that are easily adaptable for web use.   The author of Quick NLG likely used the WebNLG corpus because it gives readers an example of how to format and load their own training data.
    - Don’t need to use WebNLG once you understand the formatting

WebNLG corpus

- WebNLG corpus is comprised of .xml files each containing a triple, triple information, and example sentence



```xml
<entry category="Food" eid="Id65" shape="(X (X) (X))" shape_type="sibling" size="2
    <originaltripleset>
        <otriple>Arròs_negre | country | Spain</otriple>
        <otriple>Arròs_negre | ingredient | White_rice</otriple>
    </originaltripleset>
    <modifiedtripleset>
        <mtriple>Arròs_negre | country | Spain</mtriple>
        <mtriple>Arròs_negre | ingredient | White_rice</mtriple>
    </modifiedtripleset>
    <lex comment="good" lid="1">White rice is an ingredient of Arros negre which i
    <lex comment="good" lid="2">White rice is used in Arros negre which is from Sp
    <lex comment="good" lid="3">Arros negre contains white rice as an ingredient a
</entry>
```

Huggingface

- an AI community that maintains the libraries used in our code, as well as all wrapper functions for virtually every other natural language model out there
- It also provides different versions of popular models, often trained on different sizes of datasets

Data-to-Text Training With a T5 Model

- 2 types of data needed: example sentence, triple
- example sentence shows the model what a correct use of the input values looks like
- if our input values were “orange” and “fruit”, we might use an example sentence like: “An orange is a fruit”.

Triple Creation

- Triple: establishes the relationship between the input values as they exist in the sample sentence
    - Composed of 3 components: subject, predicate, and object
- If our example sentence is “An orange is a fruit” then:
    - Subject = orange, object = fruit
    - orange | plantType | fruit
        - Predicates can be anything the trainer desires but should generally clarify the relationship between the subject and the object

More Complex Sentences, Triple

- "An orange is a fruit. Fruit grows on trees."
    - Contains two related but grammatically separate ideas, so two triples are needed (separated by double ampersands).
    - orange | plantType | fruit && fruit | grownOn | tree

- “An orange is a type of citrus fruit and citrus fruits grow on trees.”

    -More complicated

    -Citrus used multiple times, fruit goes from being an object to a subject

orange | plantType | fruit &&

orange | plantType | citrus &&

fruit | plantType | citrus &&

fruit | grownOn | tree

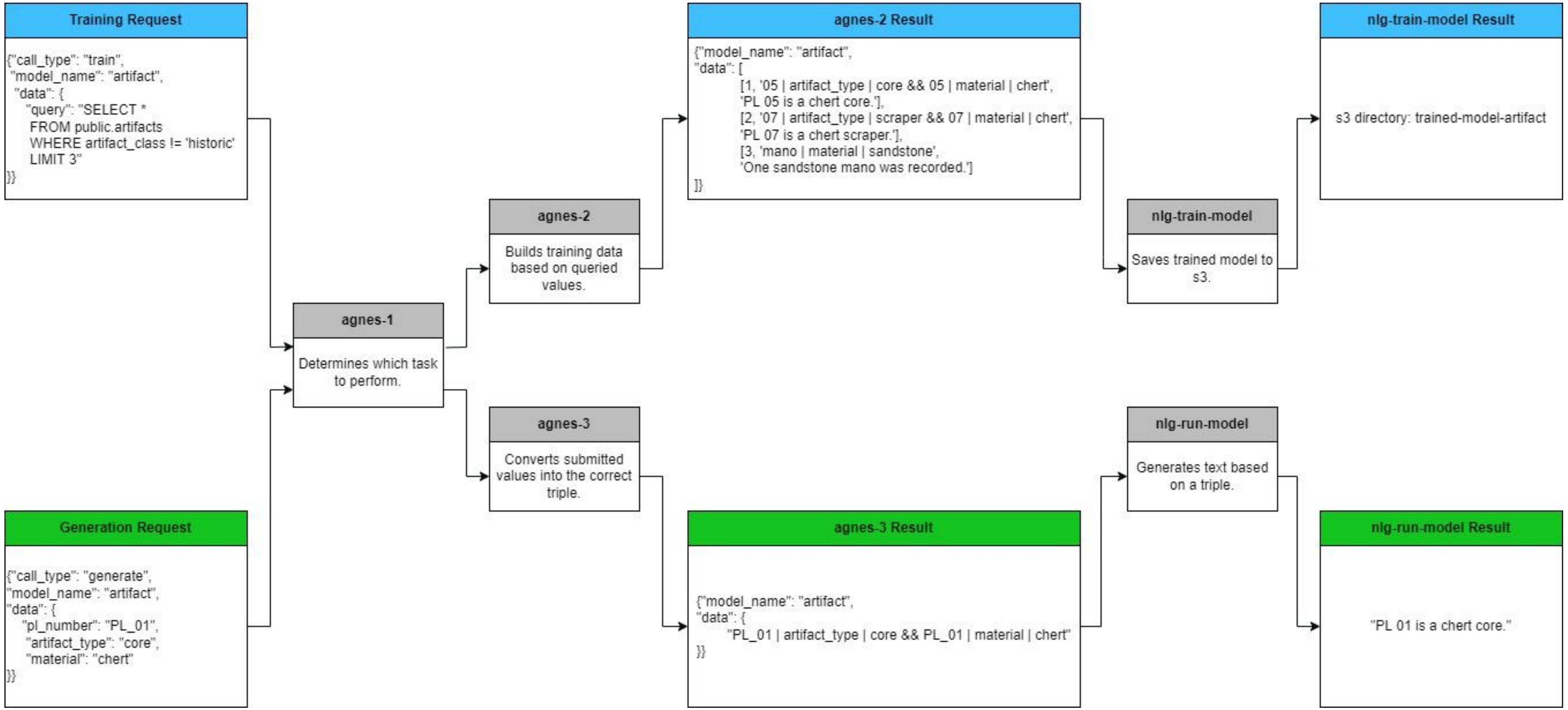
Introduction to AWS Lambda

- Want to create a whole microservice that will be able to process Artifact and new types of data
- make it easy to create modular solutions and because we can make our code available to a variety of applications easily using an API
- AWS Lambda = serverless (kind of does the work for us)
  - Do not need to configure any machines in order to run the code we use
  - Has an ECR (elastic container registry), lets us maintain a repo of all your docker images (use cases)

Docker

- open-source project for automating the deployment of applications as portable, self-sufficient containers that can run on the cloud or on-premises
- We don’t have to compile everything we need for the model, prebuilt image of lambda version of python
- Docker: allows us to bypass the AWS size limitation (container), can store all our containers in the cloud

Introduction to Pipeline


Training Request
agnes-2 Result
nlg-train-model Result
agnes-2
Builds training data based on queried values.
nlg-train-model
Saves trained model to s3.
agnes-1
Determines which task to perform.
agnes-3
Converts submitted values into the correct triple.
nlg-run-model
Generates text based on a triple.
Generation Request
agnes-3 Result
nlg-run-model Result

Training Request

```
{"call_type": "train",
 "model_name": "artifact",
  "data": {
    "query": "SELECT *
     FROM public.artifacts
     WHERE artifact_class != 'historic'
     LIMIT 3"
}}
```

agnes-2 Result

```
{"model_name": "artifact",
"data": [
        [1, '05 | artifact_type | core && 05 | material | chert',
        'PL 05 is a chert core.'],
        [2, '07 | artifact_type | scraper && 07 | material | chert',
        'PL 07 is a chert scraper.'],
        [3, 'mano | material | sandstone',
        'One sandstone mano was recorded.']
]}
```

nlg-train-model Result

s3 directory: trained-model-artifact

Generation Request

```
{"call_type": "generate",
"model_name": "artifact",
"data": {
   "pl_number": "PL_01",
   "artifact_type": "core",
   "material": "chert"
}}
```

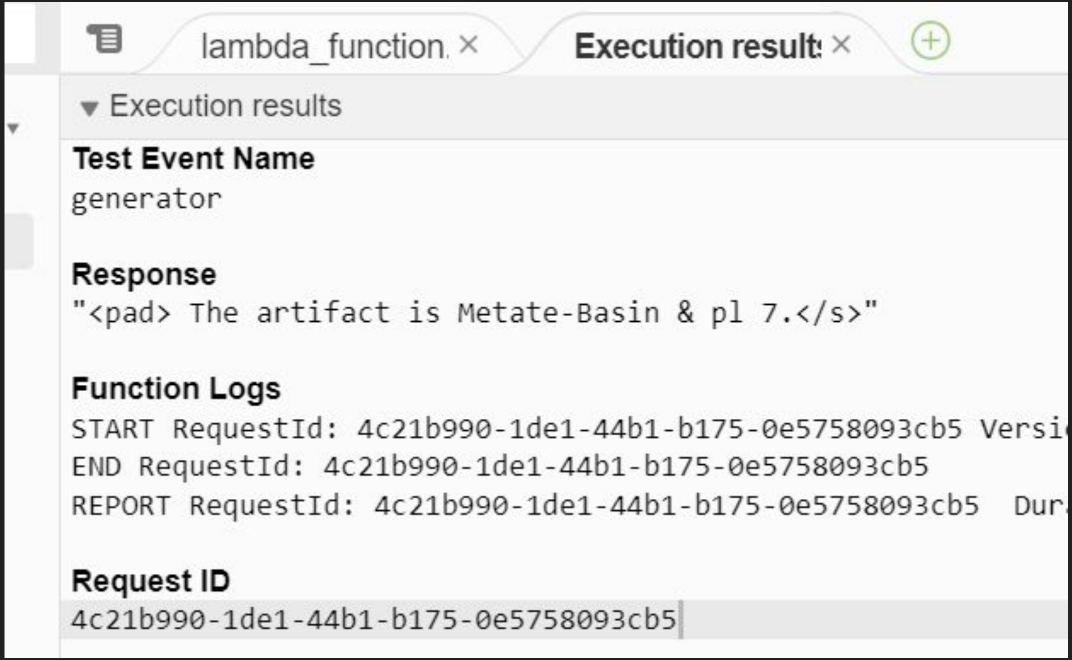
agnes-3 Result

```
{"model_name": "artifact",
"data": {
      "PL_01 | artifact_type | core && PL_01 | material | chert"
}}
```

nlg-run-model Result

"PL 01 is a chert core."

Demo AWS



Test Event Name
```
generator
```

Response
```
"<pad> The artifact is Metate-Basin & pl 7.</s>"
```

Function Logs
```
START RequestId: 4c21b990-1de1-44b1-b175-0e5758093cb5 Versi
END RequestId: 4c21b990-1de1-44b1-b175-0e5758093cb5
REPORT RequestId: 4c21b990-1de1-44b1-b175-0e5758093cb5  Dur
```

Request ID
```
4c21b990-1de1-44b1-b175-0e5758093cb5
```

Next Steps

- Hope to automate the process of loading docker images, creating project-specific sentence generation microservices
- Ways to improve accuracy of our models
- Insert generated text into templates
- Adding Marketing's Letter Proposals

Thank you

Adrian for his guidance and patience through the duration of my internship

Gabe

Any questions?