5.7 TEXT MODERATION MODEL

5.7.1 INTRODUCTION

The objective of text moderation model is to detect any text that contains curse words or inappropriate content (e.g., Hateful speech, violence).
The model is based on BERT (Bidirectional Encoder Representations from Transformers) which is a Google AI Language research paper that introduces a bidirectional training of transformers that enable transfer learning for different NLP (Natural language processing) tasks.

5.7.2 SEQUENCE-TO-SEQUENCE MODEL

Sequence-to-sequence models are deep learning models that are used mainly in NLP tasks like machine translation, text summarization, such models were used by Google translate engine from ~2017. Seq2Seq model basically takes sequence as input and outputs another sequence as output having input based by an encoder which passes its output to a decoder outputting the output of the model, for example if used in machine translation input will be the given language and output will be the translated text of the targeted language.

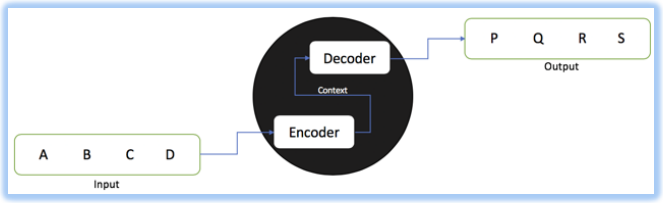

Figure 1: Seq2Seq Model Diagram

The encoder and the decoder in seq2seq model are basically a collection RNN (Recurrent neural network) cells, each of these cells have two inputs the first input is the current word while the second one is the previous word, the output is then received and also preserved to be used as input to the next layer.

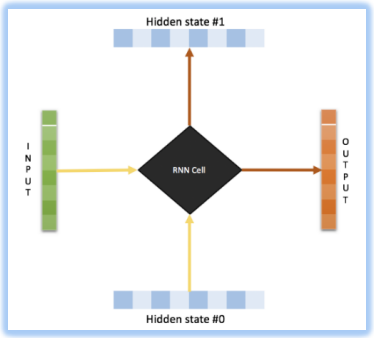

Figure 2: RNN Cells

5.7.3 PROBLEM WITH SEQ2SEQ MODEL

The encoder process the sequence generating a final representation (embedding layer) which is sent then to the decoder to process and to generate the output, after each output is processed a hidden layer is preserved to be used in the upcoming sequence, this architecture has a major drawback which is the output sequence depends mainly on the information received from the hidden state in the final output of the encoder, making it harder for the model to be able to predict correctly longer sentences; As the encoder outputs a single vector which is a bottle neck as there is a high probability that initial information of sequence-sentence- will be lost at the end of the sentence meaning that the initial context is lost.

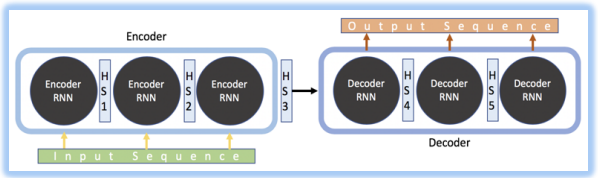

Figure 3: Encoder-Decoder with hidden state

5.7.4 ATTENTION MODELS

Attention models is an increment over the traditional seq2seq model since it addresses the seq2seq model's main drawback which is performance decreases drastically as the size of the sequence -sentence- increases. Attention models address these issues by utilizing the intermediate encoder to build up the context final vectors required which is sent after to the decoder to generate the output sequence -sentence-.

For example, in the following case, when using seq2seq model HS3 is the only state -vector- that is passed to the decoder layer neglecting HS1 and HS2, while when using attention model all of the three state HS1,2,3 are passed to the decoder layer which make the probability of losing initial context lower.

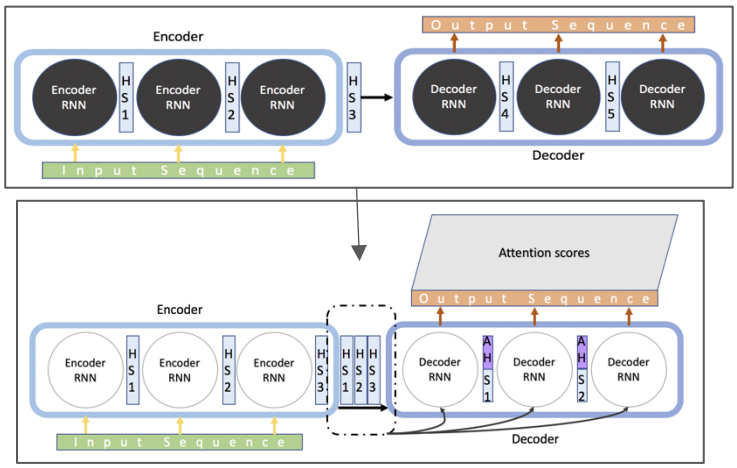

Figure 4: Attention Model

The attention model is able to use multiple hidden states by using some additional operations:

A. Creating context vector step is done at every time step (t) using a weighted sum of the input states to create a single vector.
B. After creation of context vector is created it is concatenated with the hidden layer of each of the encoder RNN cells replacing the old hidden states with new states named attention states (AS).
C. The alignment model it is a simple Neural Network that is trained initially with seq2seq model, this model takes in encoder hidden states of input and the decoder hidden attention states of previous output together and uses softmax activation function to deduce how much input matches previous output to map between input and output in other words which parts of the input is most important to predict which parts of outputs.

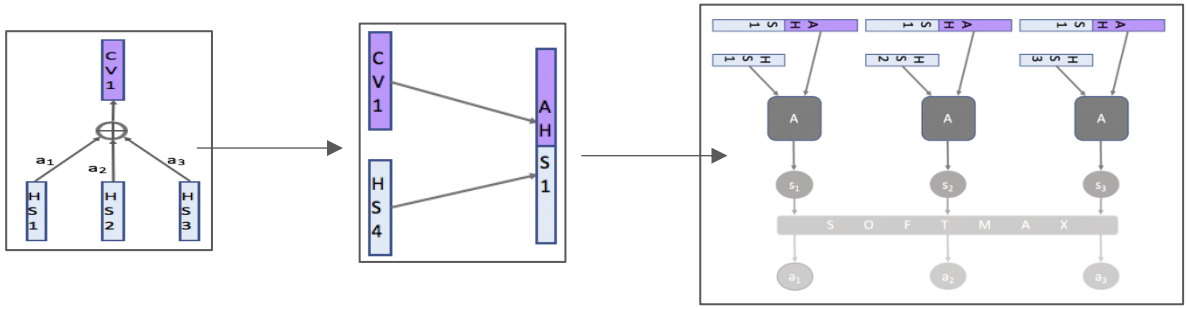

Figure 5: Attention Model Inner Workings

5.7.5 PROBLEM WITH ATTENTION SEQ2SEQ MODELS

The main drawback of attention based seq2seq models is that it uses hugely RNN -recurrent neural networks- which is used to remember how sequences is inputted to the model and their arrangement.

5.7.4 TRANSFORMER MODELS

Transformers model is an increment over attention seq2seq model as it introduces a way to use attention techniques without using RNN.

Transformers offers Encoder-Decoder architecture replacing the RNN cells with multi-attention building blocks and normalization blocks which is a combination of scaled dot product operation and concatenating to previous part then linear transformation, also since no usage of RNN positional encodings is used to indicate the relative position of sequence's parts

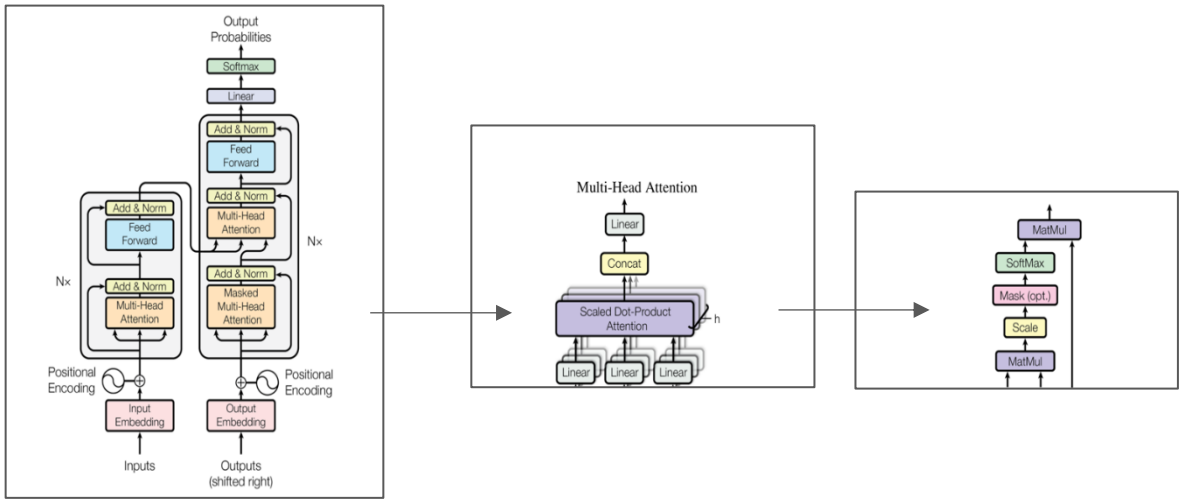

Figure 6: Transformer Model

Scaled Dot Product Attention consist of:

1. Matrix multiplication between vector representation of one word in the sequence, vector representations of all the words in the sequence.
2. Scaling of result from matrix multiplication.
3. Masking of result from scaling.
4. Softmax activation function on mask.
5. Matrix multiplication of softmax output and the vector representations of all the words in the sequence.

5.7.5 BERT MODEL

Bert is a Transformer based model, introduced by a paper published by Google AI team, Bert model use different training strategy than the other models to enhance the ability of a language model.

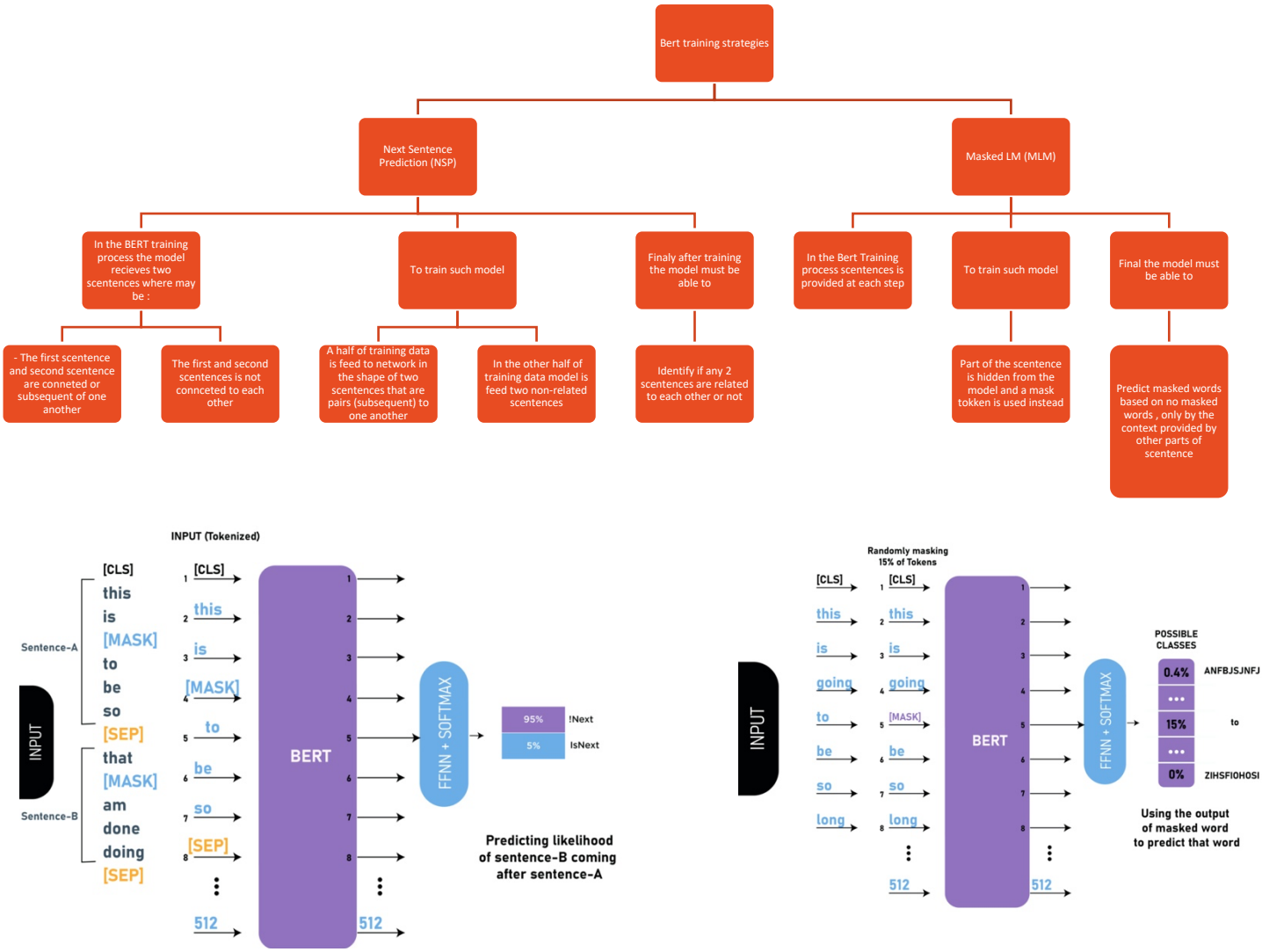

Figure 7: Bert Model Explanation

5.7.6 USING BERT FOR TEXT MODERATION

Using Bert (mBERT) for text moderation is considered an application on binary classification models as text is classified either to be inappropriate or appropriate.

5.7.7 BERT MULTILINGUAL MODELS

Multilingual BERT (mBERT) is a distribution that was released back with the original Bert, having the corpus of more than 100 languages including English, French, Spanish. The mBERT was trained on Wikipedia feeds with a shared vocabulary across all languages.

To help in reducing the imbalance between the different languages the training data was oversampled for small-content languages and under sampled for more content languages

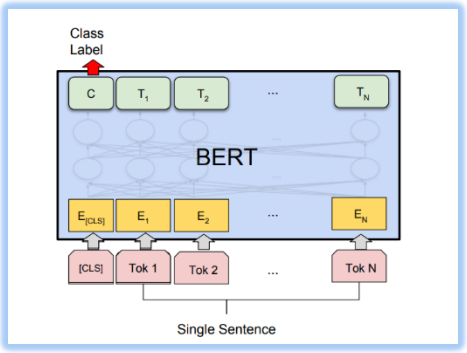

Figure 8: MBert Model

5.7.8 MODEL ARCHITECTURE

The model contains 3 Layers:

- An input Layer (Where the text content enters to the model).
- Bert Layer (The core part of the model)
- Dense output layer (contains one node to provide predictions and uses sigmoid activation function to provide a prediction between 0 and 1 only).

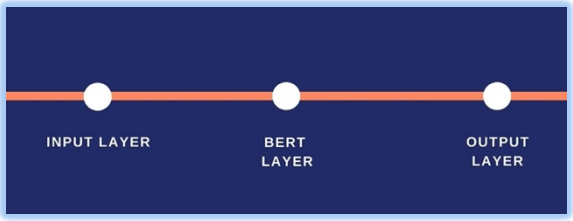

Figure 9: Text-Moderation Model Architecture

5.7.9 SIMPLIFIED SEQUENCE DIAGRAM FOR MODEL

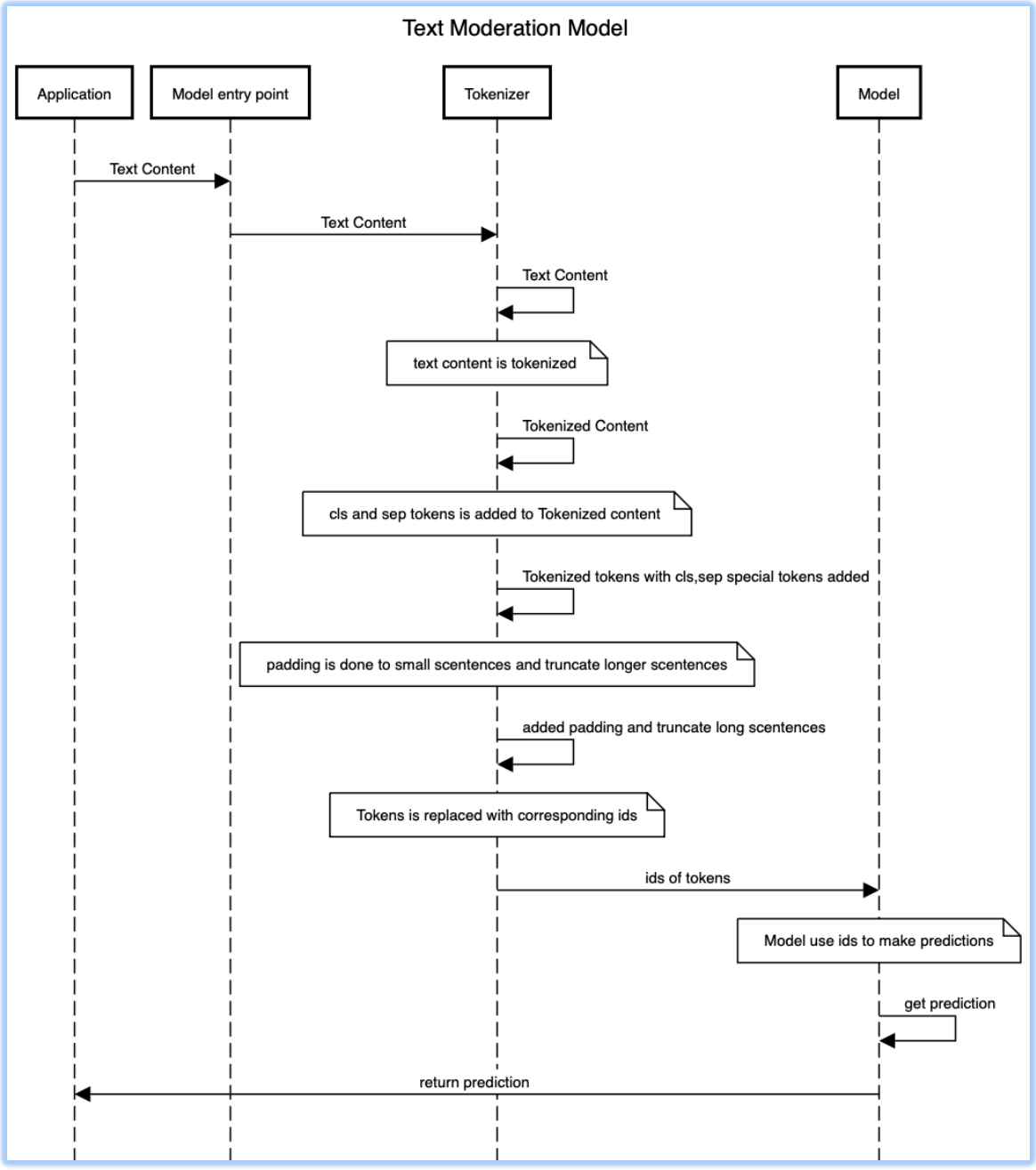

Figure 10: Text Moderation Model Sequence Diagram

5.8.11 MODEL INTERFERENCE

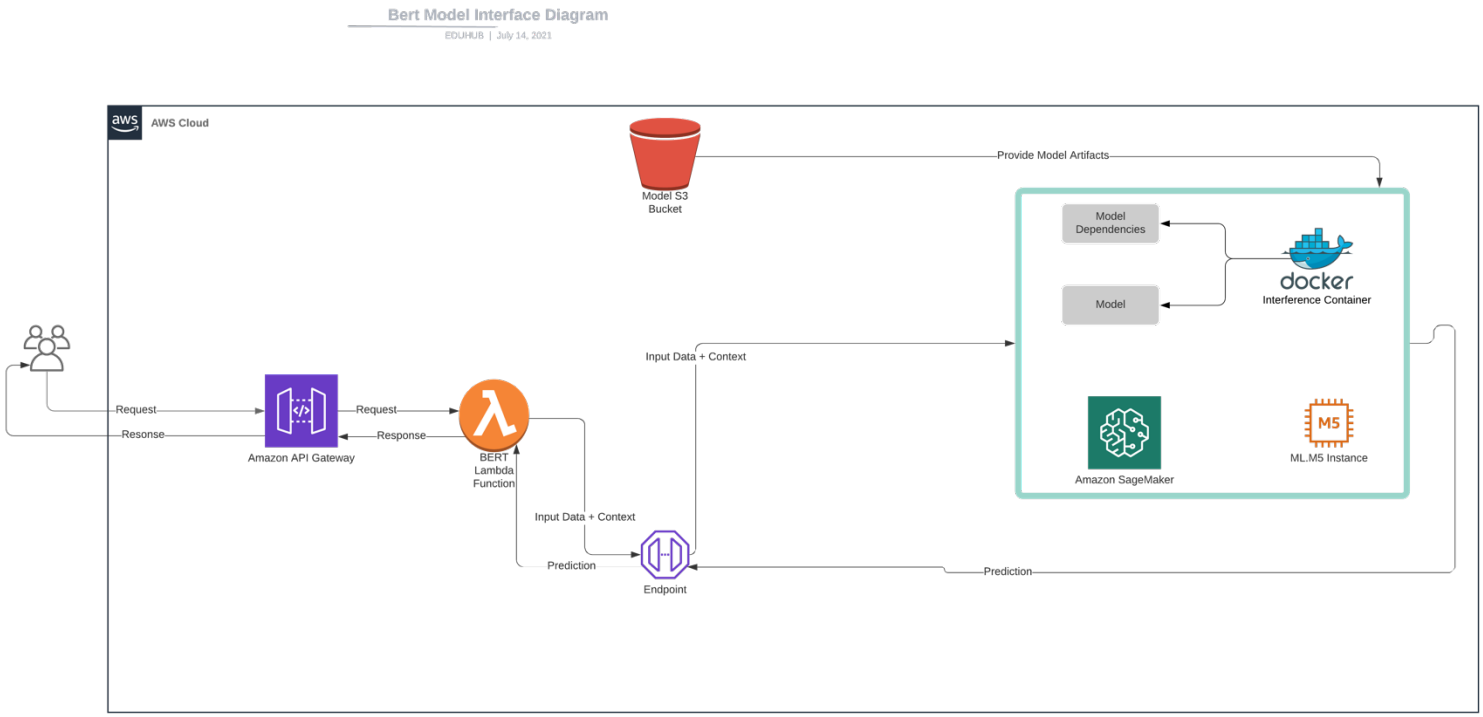

Figure 11 Bert Model interference architecture

The text Moderation prediction pipeline is fully deployed on AWS (Amazon Web Services) the deployment architecture can be briefly described as following :

1. The application / user send his text to be classified against the model through sending a request (REST request) to AWS's API GATEWAY
2. API GATEWAY triggers AWS Lambda function which takes the string (text) from the request body and invoke the model's endpoint
3. The endpoints invoked triggers AWS SageMaker which is hosting the interference image of the model and originally have access to the model artifacts which is placed original on AWS S3 bucket
4. Model predict and classifies the text and returns the prediction in HTTP form
5. AWS Lambda function takes the response from the model endpoint and parse it as int first then as string /text and forwards it as response to the request as string of result :

- 1 for inappropriate text
- 0 for appropriate text

6. Result is propagated back to user/application

5.8.12 MODEL DEPLYMENT ON SAGEMAKER DETAILS

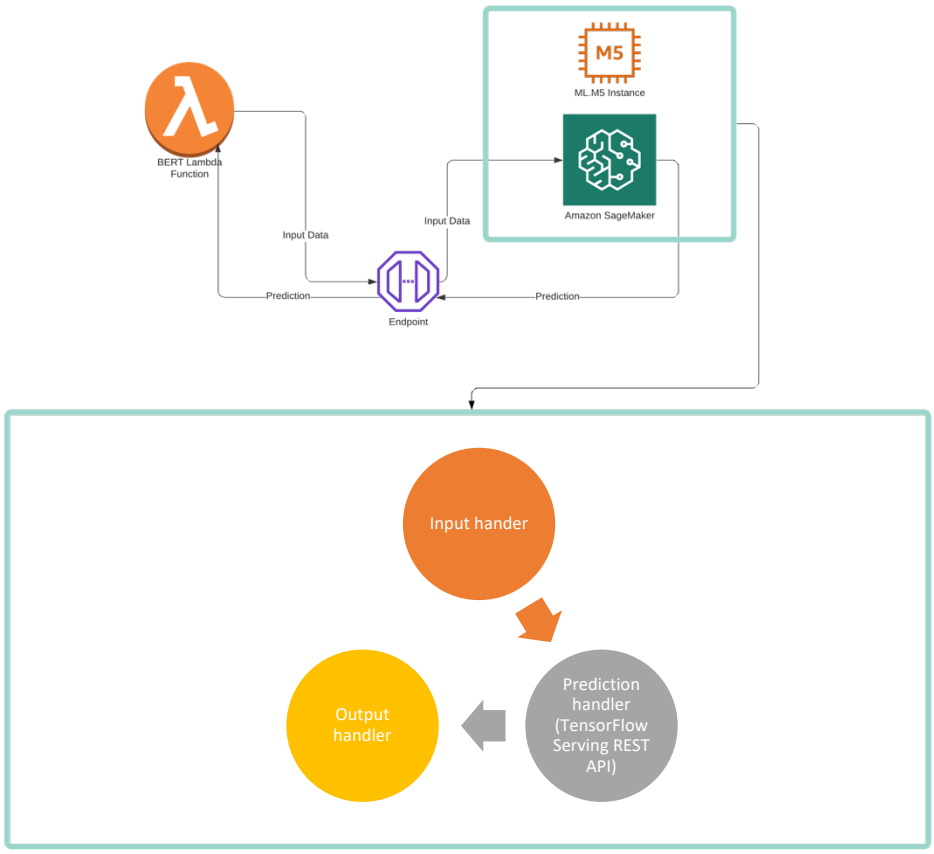

Figure 12 Bert Model SageMaker Model Deployment Details

Amazon SageMaker depends on docker containers in text moderation case it is TensorFlow container , TensorFlow container depends on 3 Parts (2 optional parts but utilized in text moderation container) :

1- Input Handler : Pre-process request input before it is sent to TensorFlow Serving REST API

- It is used in text moderation container to take string input and cast it into numpy list (python data structure)
- Encode the data using the tokenizer

- Make data in Json Format

2 – Prediction Handler : This part takes input from (Input Handler) as json file format make predictions against model and return predictions to output handler

- Used to predict result from input handler json like tokenized input data

3- Output Handler : Post-process TensorFlow Serving output before it is returned to the lambda function

- Return prediction and data type
- If no response raise error

5.8.13 (SEMESTER 10) MEETING NON-FUNCTIONAL REQUIRMENTS

In this section of model documentation model is tested against non-functional requirements as :

- Scalability
- Performance
- Reliability and Availability

5.8.13.1 SCALABILLITY

All of the used components in interference are scalable and durable , Used components are :

1 – API GATEWAY :

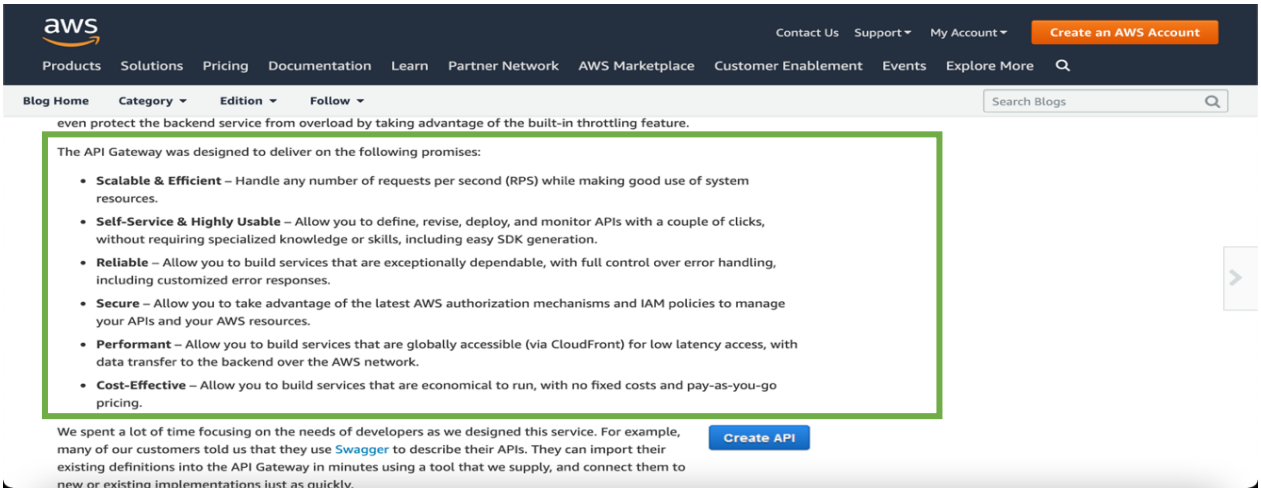

Figure 13 API Gateway scalability

2– Lambda Function :

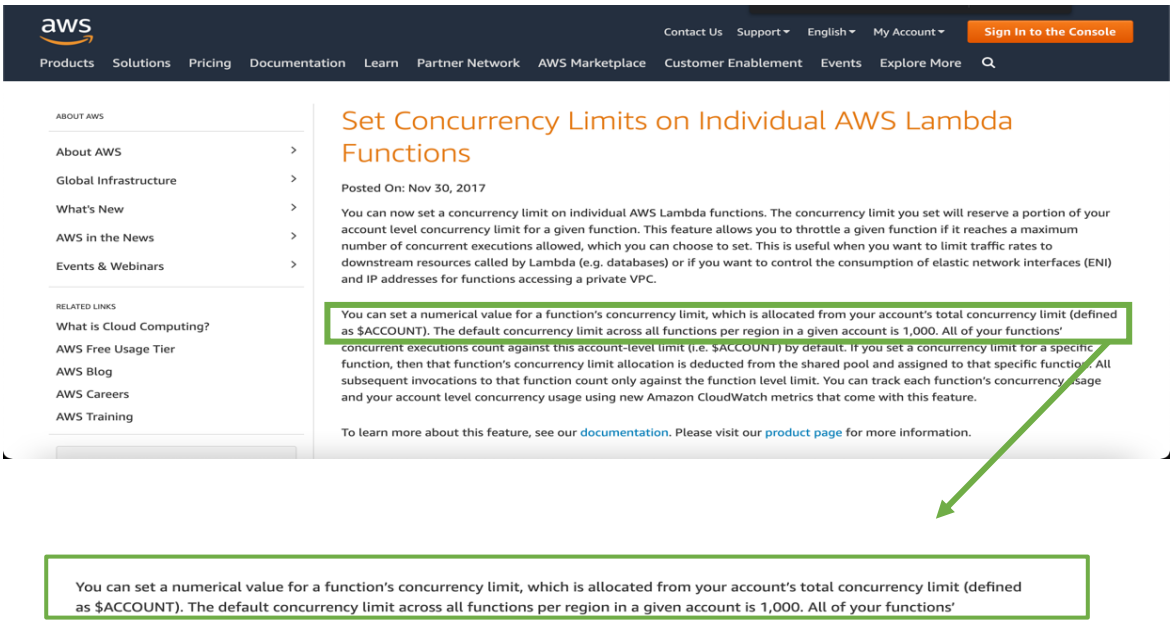

Figure 14 Lambda Function scalability

3– SageMaker Models :

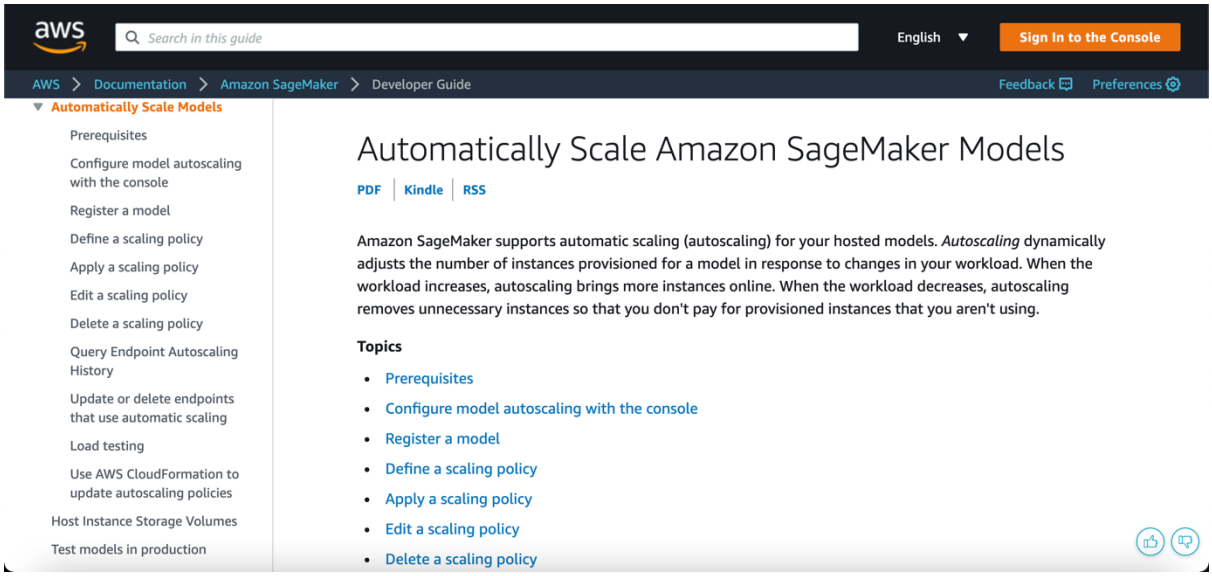

Figure 15 SageMaker Models scalability

4- S3 Buckets :

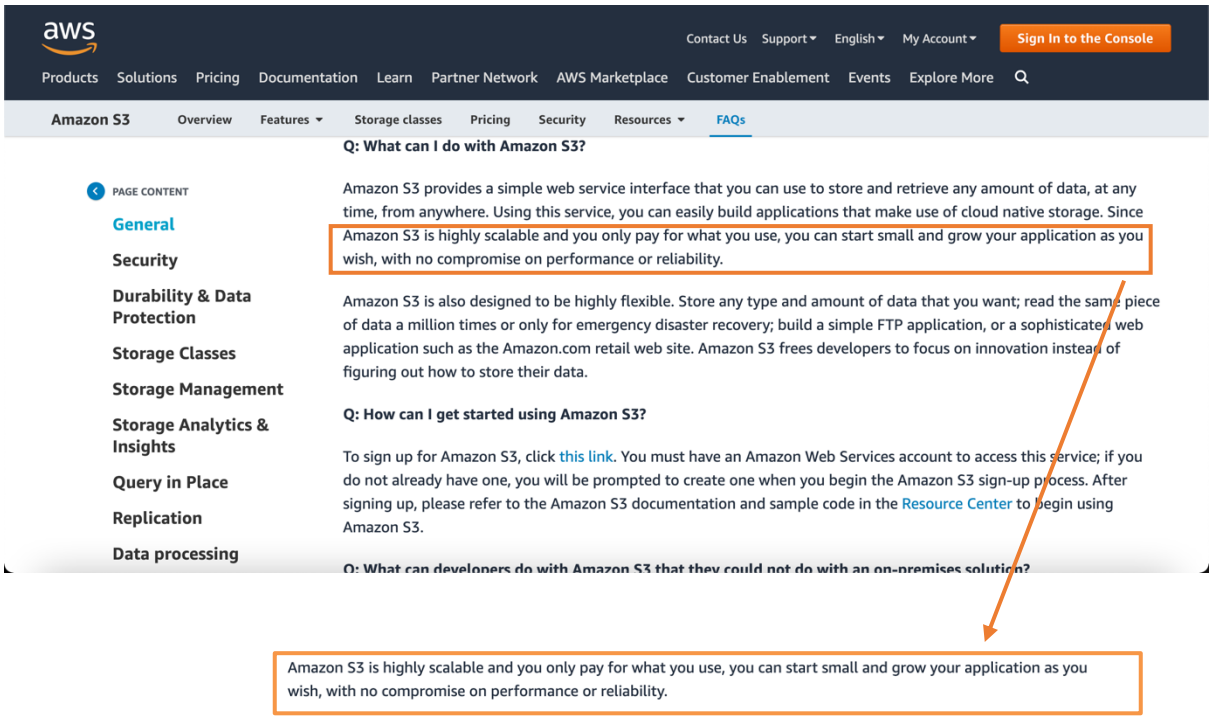

Figure 16 S3 Buckets scalability

5.8.13.2 PERFORMANCE

Performance in text moderation model has been maximized by handling encoding process (Tokenizing of text) inside the more powerful TensorFlow container inside the input handler function instead of having to do that inside the less powerful lambda function , also packages such as numpy package has been moved to the container itself to reduce lambda launching time. After previous modifications the whole process takes averge of 1.5 second

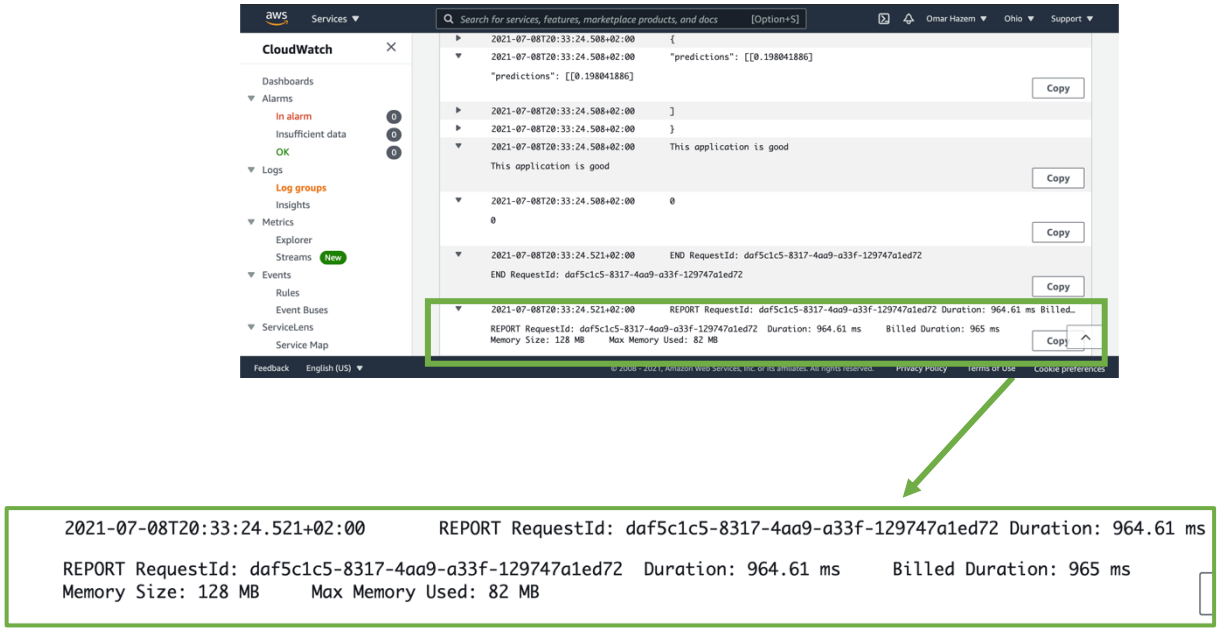

Figure 17 Text Moderation Performance

5.8.13.3 RELIABILITY AND AVAILABILITY

Reliability :The reliability goal for cloud vendors is "five nines" (99.999 percent)[According to cloud outage figures for 2014]a handful of vendors achieved that goal while Amazon Web Services came the closest among the largest cloud providers.

Figure 18 Availability of AWS

Availability : AWS serves its user with 25 Launched Regions Each with multiple Availability Zones (AZ's) 81 Availability Zones , 245 Countries and Territories Served , 108 Direct Connect Locations , 230+ Points of Presence , 218+ Edge Locations and 12 Regional Edge Caches

5.8.14 TEST CASE

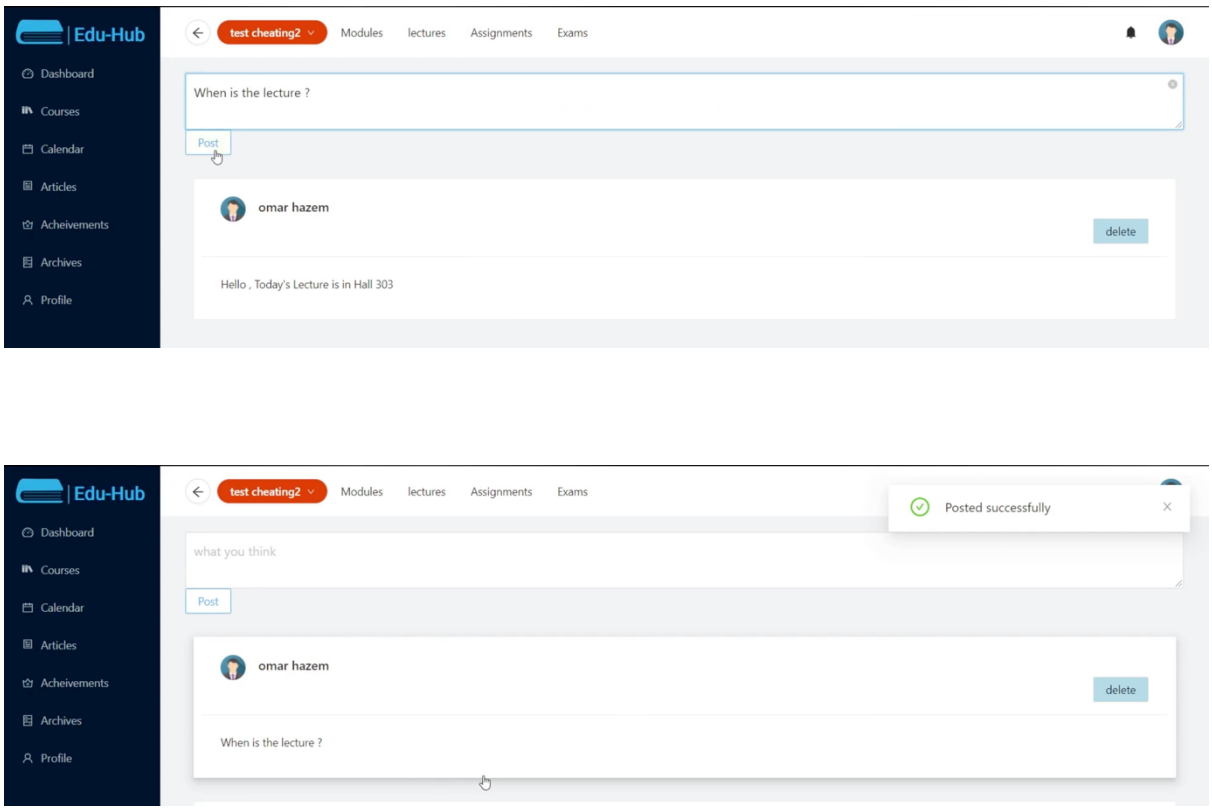

Figure 19 Client Side Text Moderation

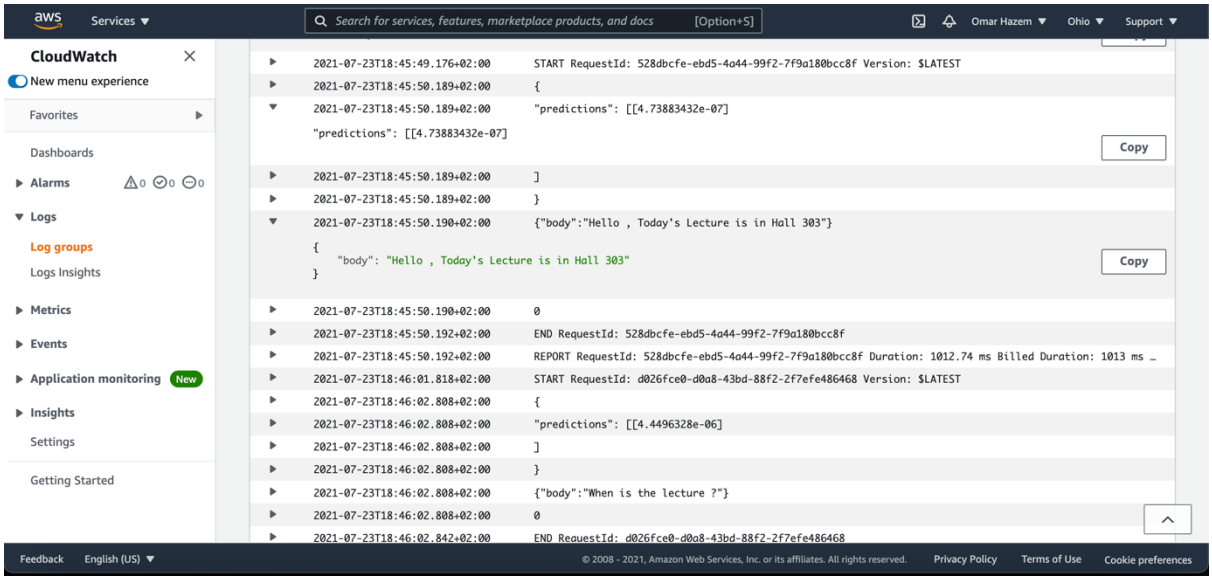

Figure 20 Server side (AWS) text moderation logs

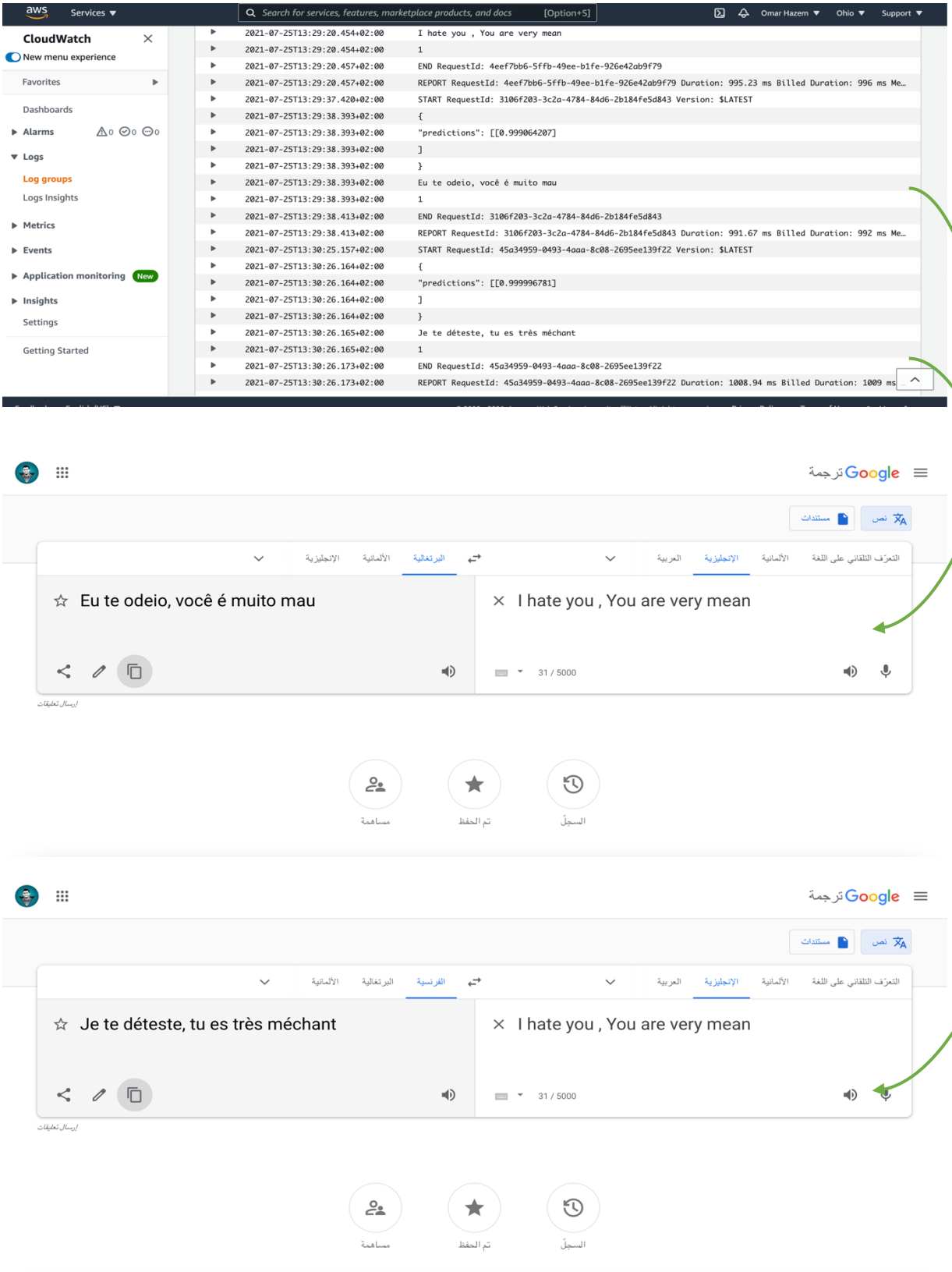

Figure 21 Non-English Text Moderation Example