

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.

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.

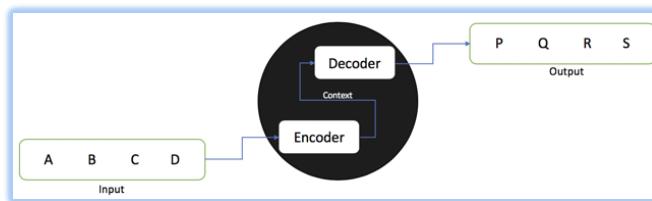


Figure 1: Seq2Seq Model Diagram

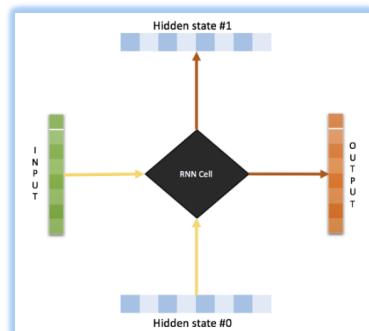


Figure 2: RNN Cells

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.

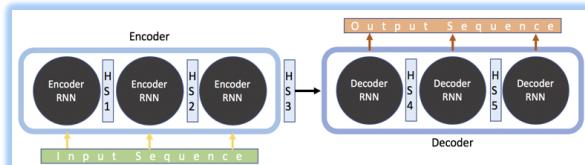


Figure 3: Encoder-Decoder with hidden state

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.

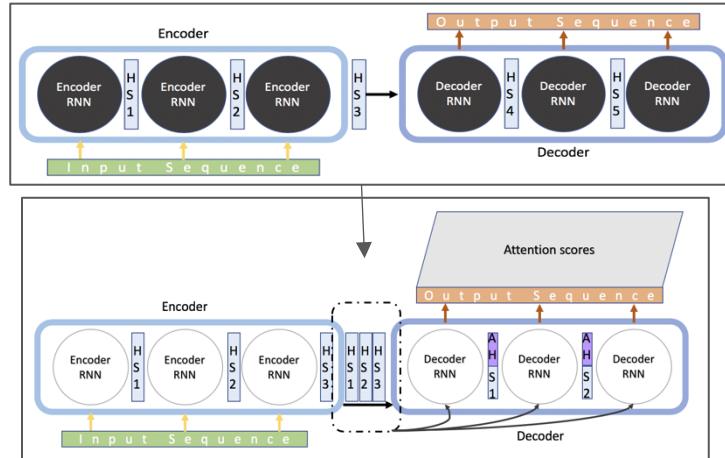


Figure 4: Attention Model

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

- Creating context vector step is done at every time step (t) using a weighted sum of the input states to create a single vector.
- 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).
- 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.

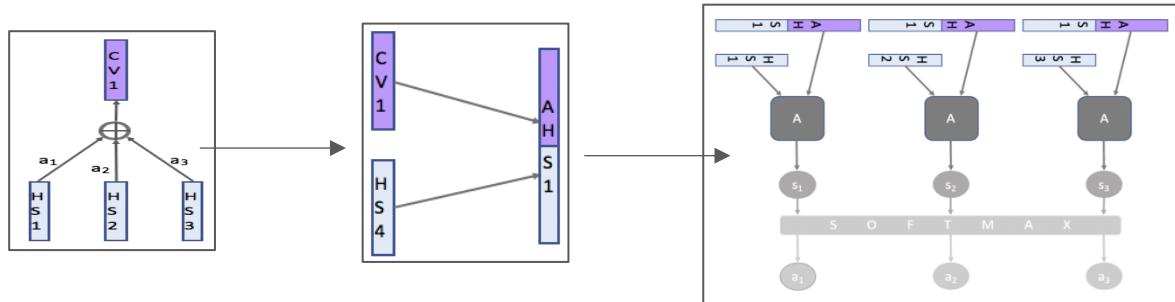


Figure 5: Attention Model Inner Workings

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

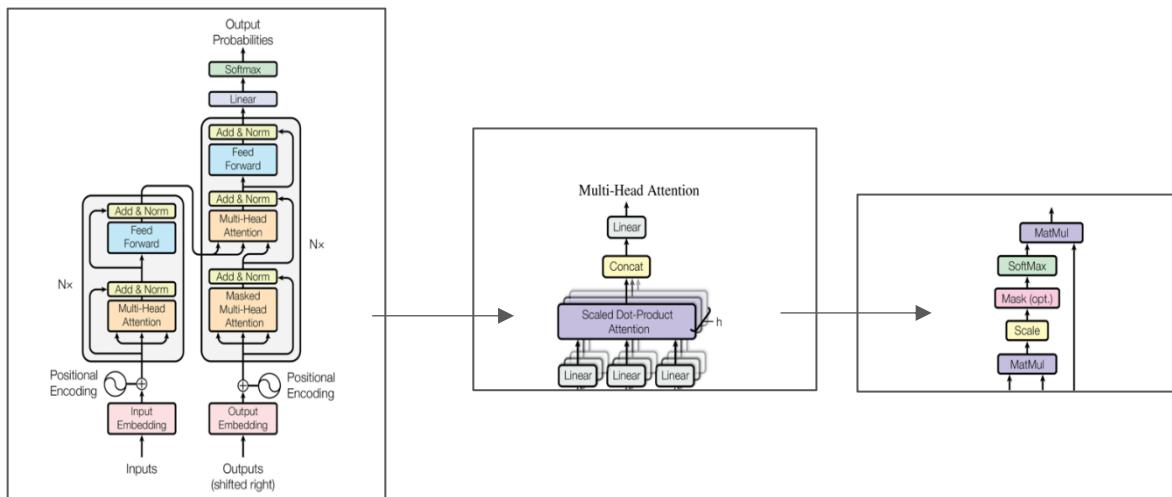


Figure 6: Transformer Model

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.

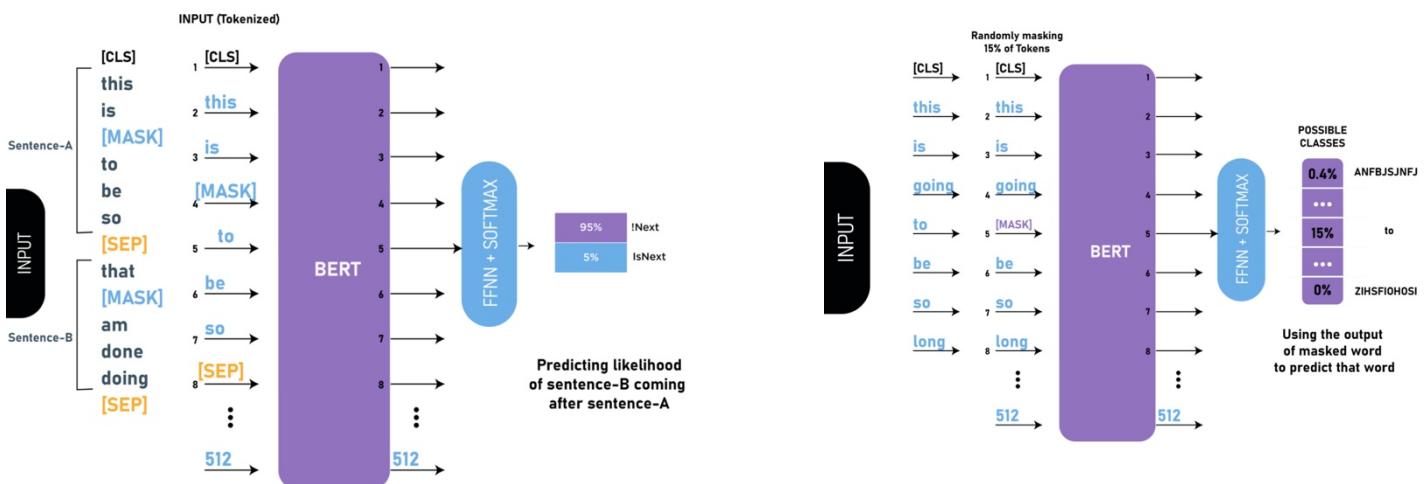
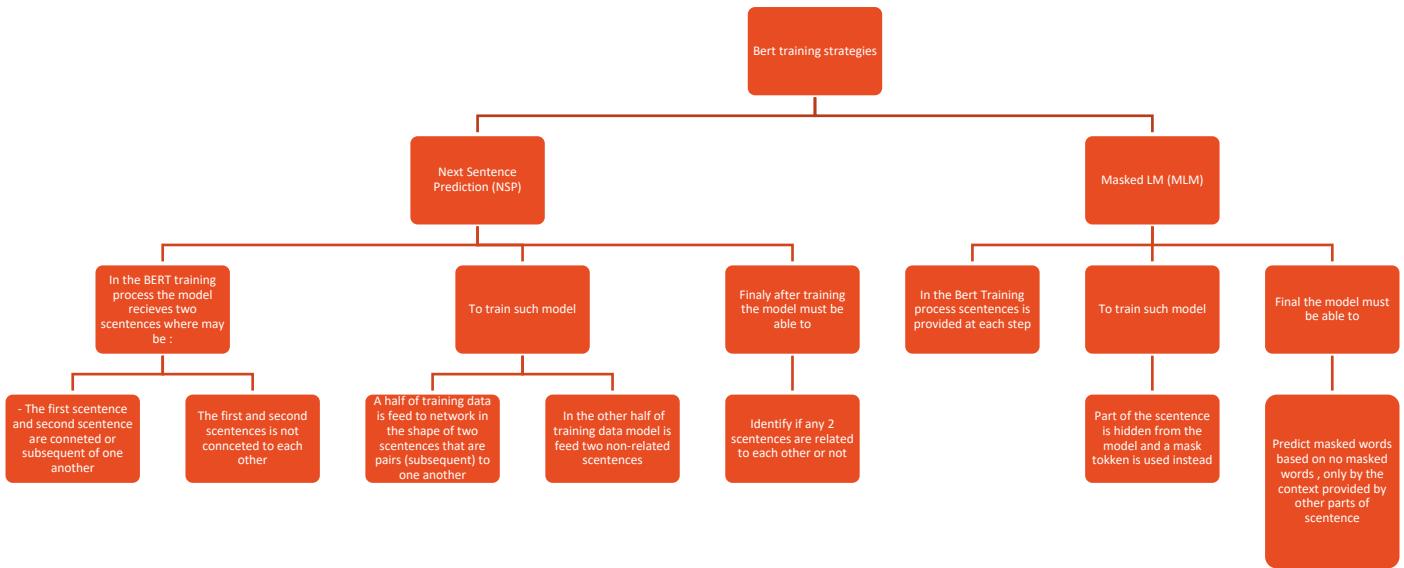


Figure 7: Bert Model Explanation

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

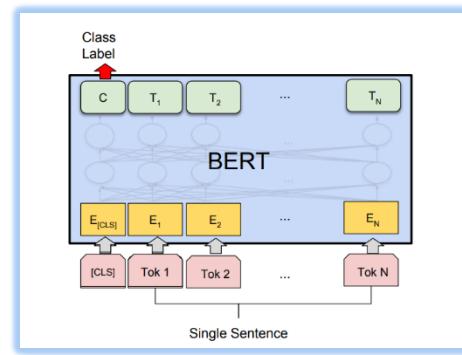


Figure 8: MBert Model

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).

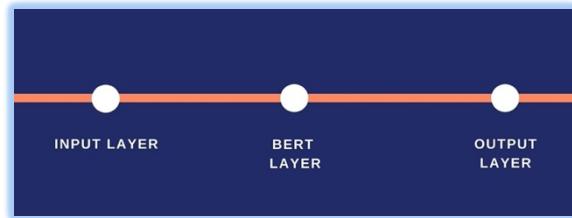


Figure 9: Text-Moderation Model Architecture

5.7.9 SIMPLIFIED SEQUENCE DIAGRAM FOR MODEL

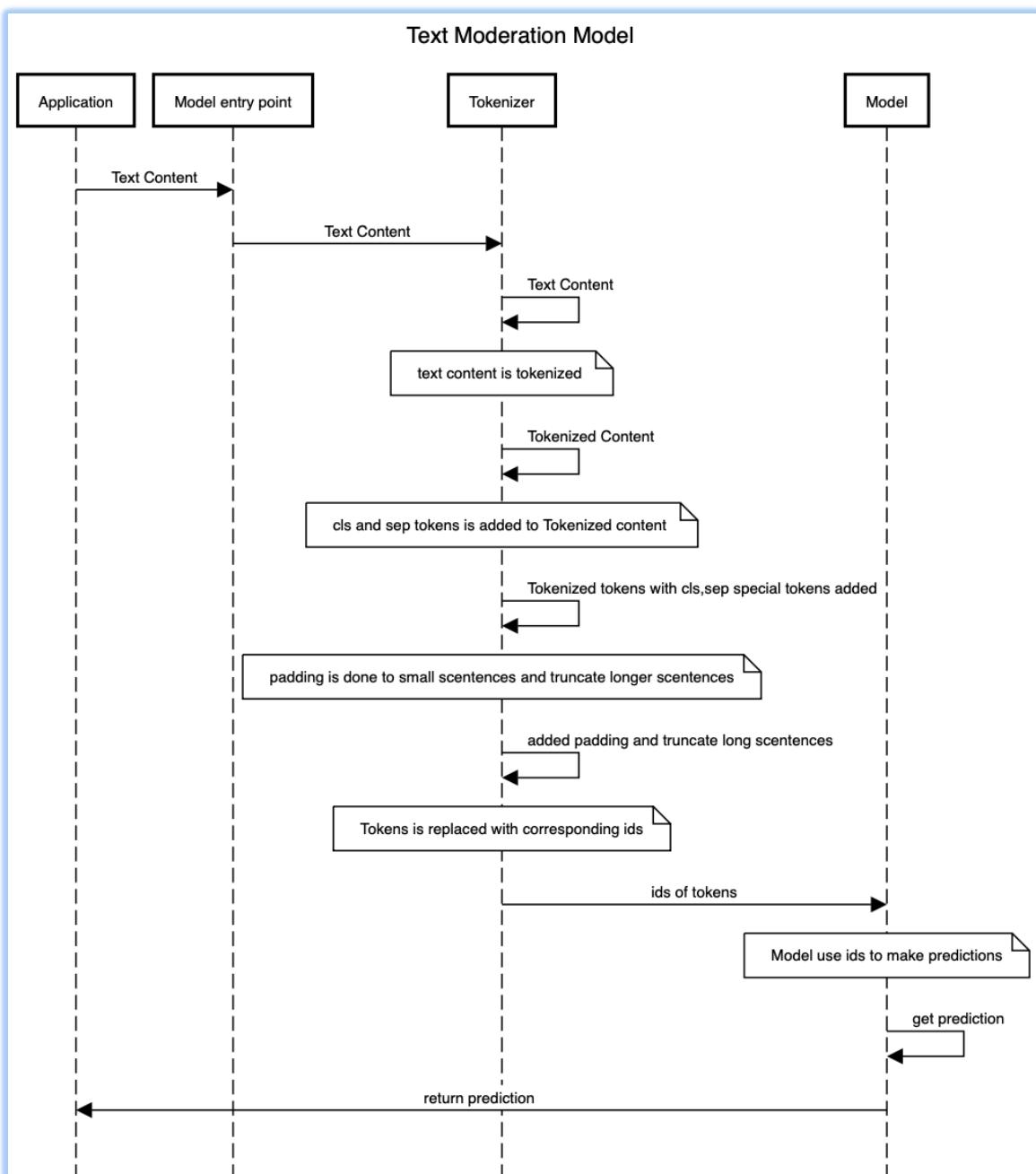


Figure 10: Text Moderation Model Sequence Diagram

5.8.11 MODEL INTERFERENCE

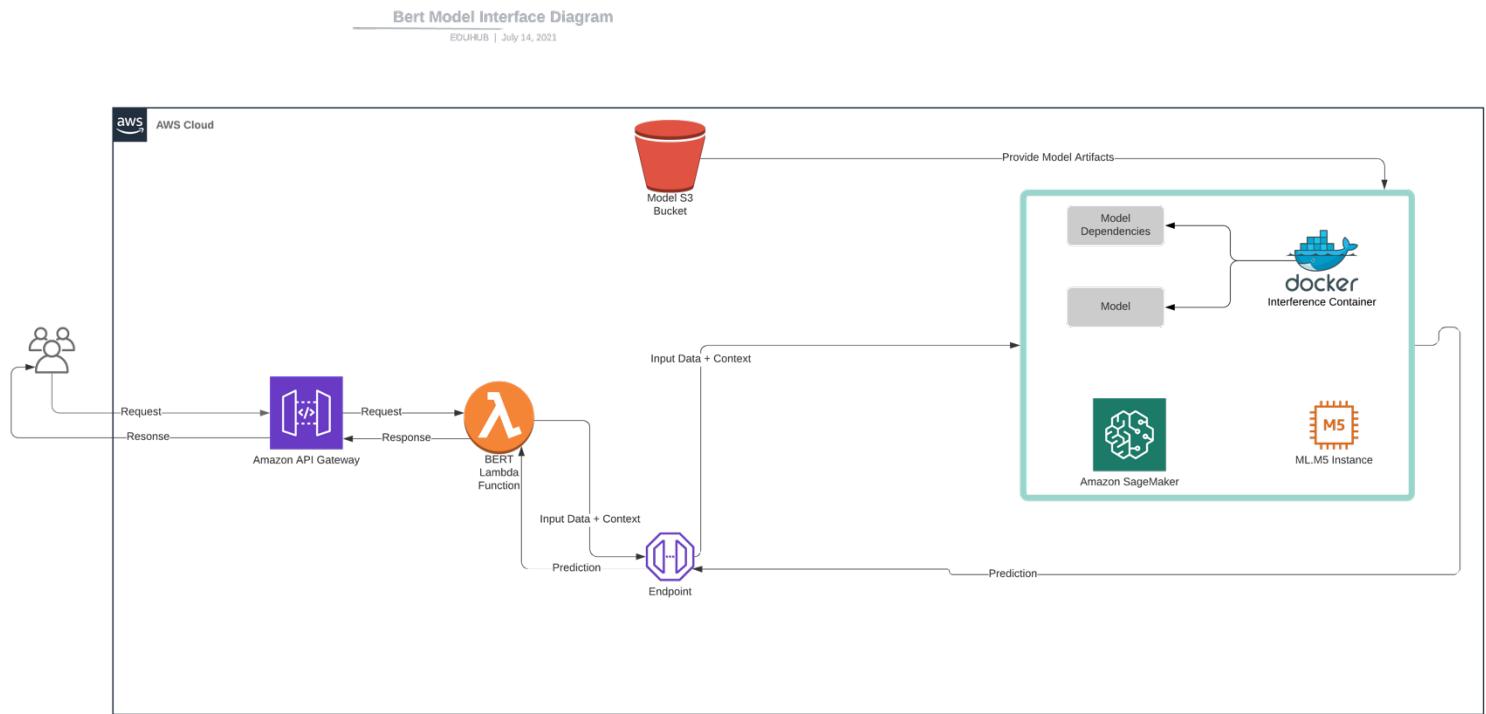


Figure 11 Bert Model interference architecture

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 DEPLOYMENT ON SAGEMAKER DETAILS

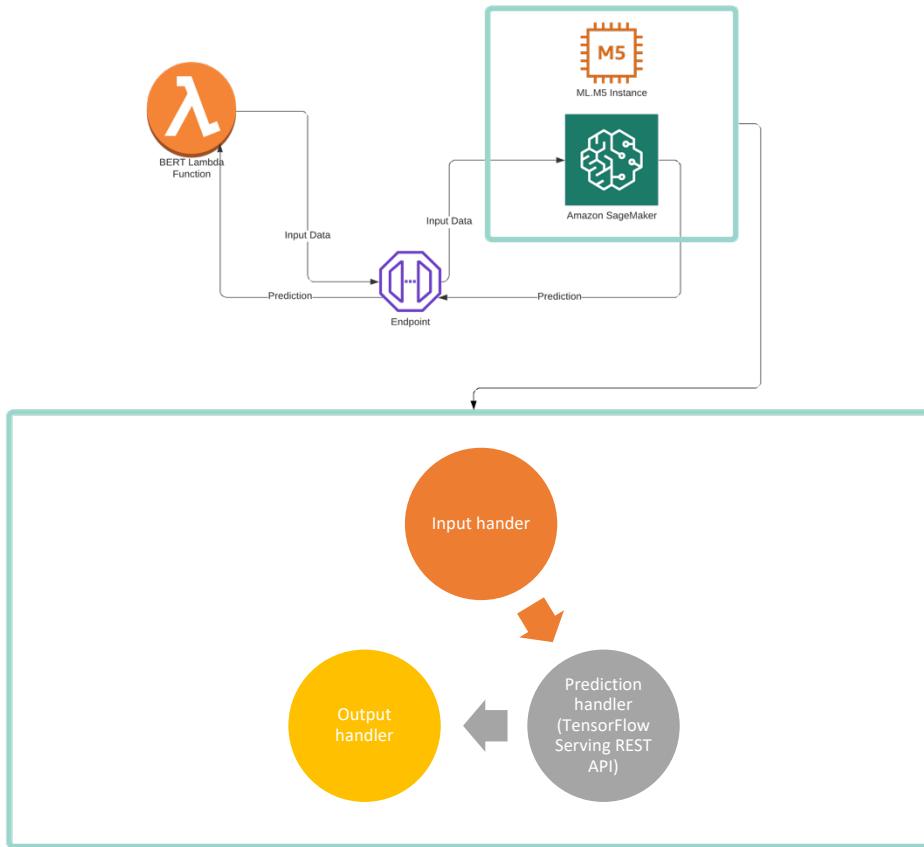


Figure 12 Bert Model SageMaker Model Deployment Details

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 REQUIREMENTS

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

- Scalability
- Performance
- Reliability and Availability

5.8.13.1 SCALABILITY

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

1 – API GATEWAY :

The screenshot shows a blog post titled "Scalability" on the AWS API Gateway blog. The post discusses the design promises of the API Gateway. A callout box highlights the following promises:

- Scalable & Efficient – Handle any number of requests per second (RPS) while making good use of system resources.
- Self-Service & Highly Usable – Allow you to define, revise, deploy, and monitor APIs with a couple of clicks, without requiring specialized knowledge or skills, including easy SDK generation.
- Reliable – Allow you to build services that are exceptionally dependable, with full control over error handling, including customized error responses.
- Secure – Allow you to take advantage of the latest AWS authorization mechanisms and IAM policies to manage your APIs and your AWS resources.
- Performant – Allow you to build services that are globally accessible (via CloudFront) for low latency access, with data transfer to the backend over the AWS network.
- Cost-Effective – Allow you to build services that are economical to run, with no fixed costs and pay-as-you-go pricing.

Below the promises, there is a note about Swagger integration and a "Create API" button.

Figure 13 API Gateway scalability

2– Lambda Function :

The screenshot shows a web browser displaying an AWS Lambda article. The header includes the AWS logo and navigation links like Contact Us, Support, English, My Account, and Sign In to the Console. The main content is titled "Set Concurrency Limits on Individual AWS Lambda Functions" and was posted on Nov 30, 2017. It explains how to set a concurrency limit for individual functions, noting that the default limit is 1,000. A callout box highlights that functions share a pool of concurrent executions. Another callout box provides a link to documentation and a product page. At the bottom left, there's a summary of the feature.

Set Concurrency Limits on Individual AWS Lambda Functions

Posted On: Nov 30, 2017

You can now set a concurrency limit on individual AWS Lambda functions. The concurrency limit you set will reserve a portion of your account level concurrency limit for a given function. This feature allows you to throttle a given function if it reaches a maximum number of concurrent executions allowed, which you can choose to set. This is useful when you want to limit traffic rates to downstream resources called by Lambda (e.g. databases) or if you want to control the consumption of elastic network interfaces (ENI) and IP addresses for functions accessing a private VPC.

You can set a numerical value for a function's concurrency limit, which is allocated from your account's total concurrency limit (defined as \$ACCOUNT). The default concurrency limit across all functions per region in a given account is 1,000. All of your functions' concurrent executions count against this account-level limit (i.e. \$ACCOUNT) by default. If you set a concurrency limit for a specific function, then that function's concurrency limit allocation is deducted from the shared pool and assigned to that specific function. All subsequent invocations to that function count only against the function level limit. You can track each function's concurrency usage and your account level concurrency usage using new Amazon CloudWatch metrics that come with this feature.

To learn more about this feature, see our [documentation](#). Please visit our [product page](#) for more information.

You can set a numerical value for a function's concurrency limit, which is allocated from your account's total concurrency limit (defined as \$ACCOUNT). The default concurrency limit across all functions per region in a given account is 1,000. All of your functions'

Figure 14 Lambda Function scalability

3– SageMaker Models :

The screenshot shows the AWS SageMaker Developer Guide. The top navigation bar includes the AWS logo, a search bar, language selection (English), and a 'Sign In to the Console' button. Below the header, the breadcrumb trail shows: AWS > Documentation > Amazon SageMaker > Developer Guide > Automatically Scale Models. On the right, there are 'Feedback' and 'Preferences' links. The main content area has a title 'Automatically Scale Amazon SageMaker Models' and a sub-section 'Topics' with a list of items.

- Prerequisites
- Configure model autoscaling with the console
- Register a model
- Define a scaling policy
- Apply a scaling policy
- Edit a scaling policy
- Delete a scaling policy
- Query Endpoint Autoscaling History
- Update or delete endpoints that use automatic scaling
- Load testing
- Use AWS CloudFormation to update autoscaling policies
- Host Instance Storage Volumes
- Test models in production

Figure 15 SageMaker Models scalability

4- S3 Buckets :

The screenshot shows the AWS S3 service page. The top navigation bar includes the AWS logo, a search bar, and a 'Sign In to the Console' button. Below the header, the breadcrumb trail shows: Products > Solutions > Pricing > Documentation > Learn > Partner Network > AWS Marketplace > Customer Enablement > Events > Explore More > Q. The main content area has a sidebar with categories like General, Security, Durability & Data Protection, Storage Classes, Storage Management, Storage Analytics & Insights, Query in Place, Replication, and Data processing. The main content area includes a question 'Q: What can I do with Amazon S3?' and its answer, which is highlighted with an orange box. Another question 'Q: How can I get started using Amazon S3?' and its answer are also shown. A third question 'Q: What can developers do with Amazon S3 that they could not do with an on-premises solution?' and its answer are at the bottom, also highlighted with an orange box. A red arrow points from the scalability text in Figure 15 to the scalability text in Figure 16.

Q: What can I do with Amazon S3?

Amazon S3 provides a simple web service interface that you can use to store and retrieve any amount of data, at any time, from anywhere. Using this service, you can easily build applications that make use of cloud native storage. Since Amazon S3 is highly scalable and you only pay for what you use, you can start small and grow your application as you wish, with no compromise on performance or reliability.

Q: How can I get started using Amazon S3?

To sign up for Amazon S3, click [this link](#). You must have an Amazon Web Services account to access this service; if you do not already have one, you will be prompted to create one when you begin the Amazon S3 sign-up process. After signing up, please refer to the Amazon S3 documentation and sample code in the [Resource Center](#) to begin using Amazon S3.

Q: What can developers do with Amazon S3 that they could not do with an on-premises solution?

Amazon S3 is highly scalable and you only pay for what you use, you can start small and grow your application as you wish, with no compromise on performance or reliability.

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 average of 1.5 second

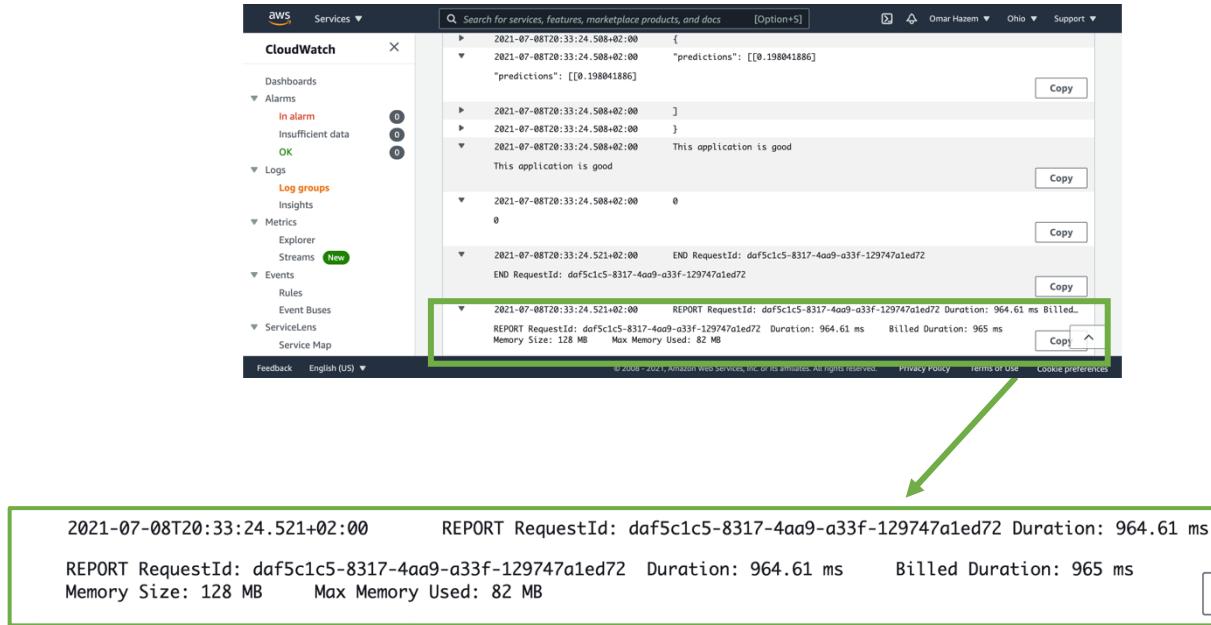


Figure 17 Text Moderation Performance

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.

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



Figure 18 Availability of AWS

5.8.14 TEST CASE

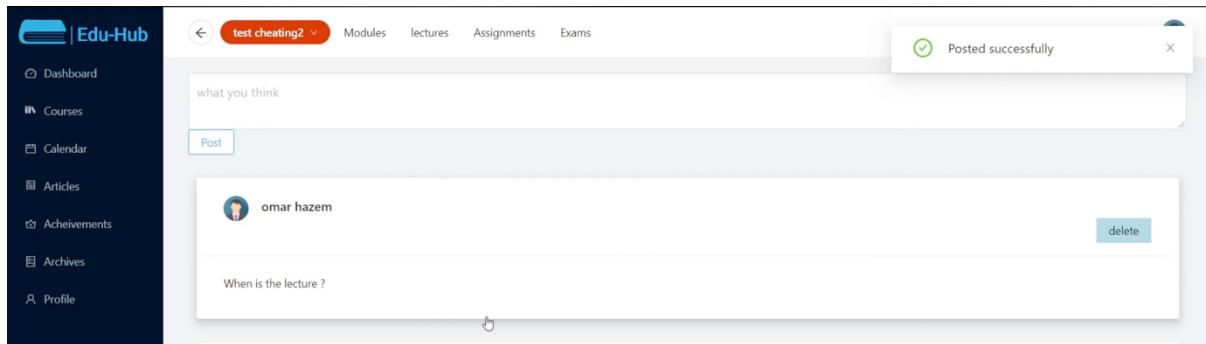
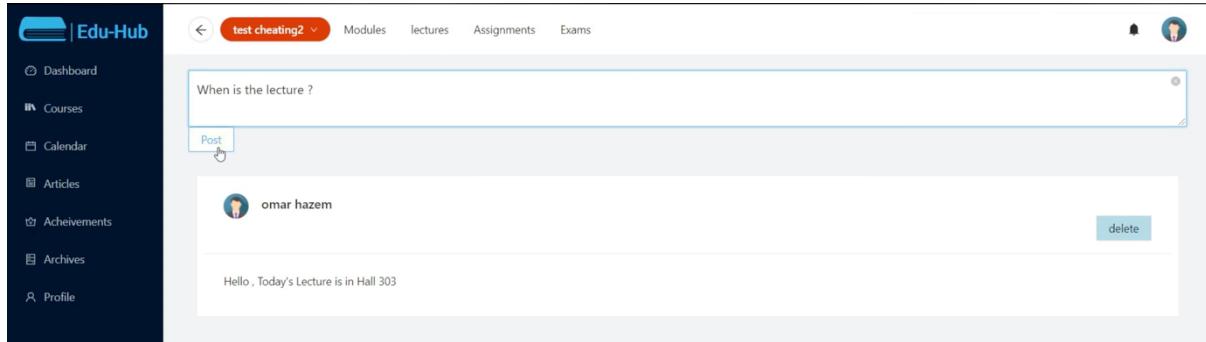


Figure 19 Client Side Text Moderation

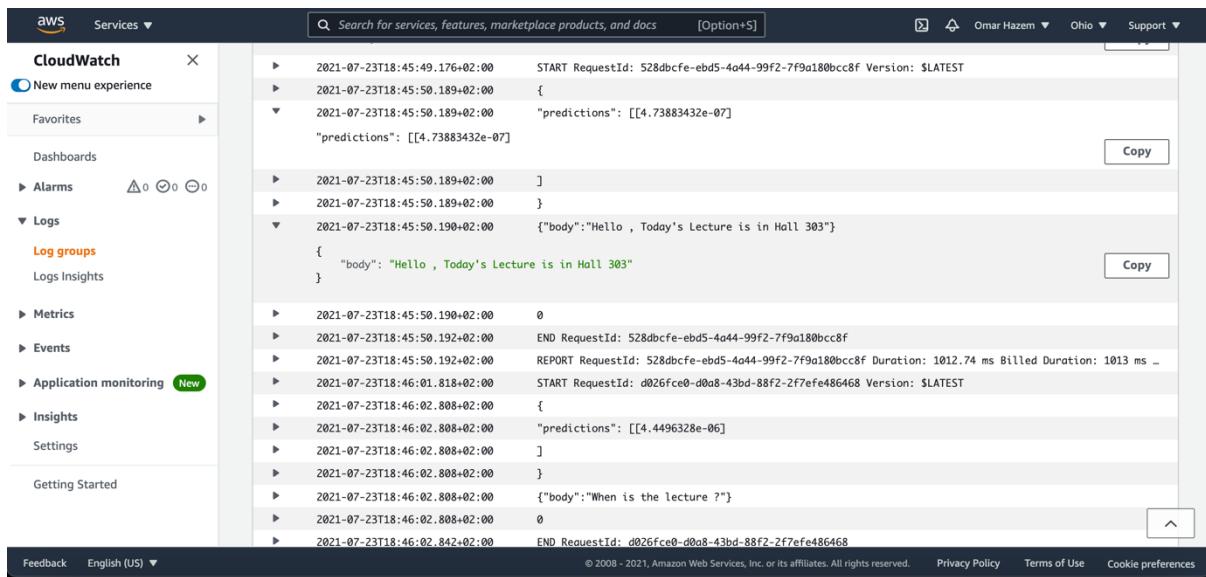


Figure 20 Server side (AWS) text moderation logs

CloudWatch Services ▾

Log groups

Logs Insights

Metrics

Events

Application monitoring **New**

Insights

Settings

Getting Started

Search for services, features, marketplace products, and docs [Option+S]

Omar Hazem ▾ Ohio ▾ Support ▾

2021-07-25T13:29:20.454+02:00 I hate you , You are very mean

2021-07-25T13:29:20.454+02:00 1

2021-07-25T13:29:20.457+02:00 END RequestId: 4eef7bb6-5ffb-49ee-b1fe-926e42ab9f79

2021-07-25T13:29:20.457+02:00 REPORT RequestId: 4eef7bb6-5ffb-49ee-b1fe-926e42ab9f79 Duration: 995.23 ms Billed Duration: 996 ms Method: POST

2021-07-25T13:29:37.426+02:00 START RequestId: 3106f203-3c2a-4784-84d6-6b184fe5d843 Version: \$LATEST

2021-07-25T13:29:38.393+02:00 {

2021-07-25T13:29:38.393+02:00 "predictions": [[0.999064207]]

2021-07-25T13:29:38.393+02:00]

2021-07-25T13:29:38.393+02:00 }

2021-07-25T13:29:38.393+02:00 Eu te odeio, você é muito mau

2021-07-25T13:29:38.393+02:00 1

2021-07-25T13:29:38.413+02:00 END RequestId: 3106f203-3c2a-4784-84d6-6b184fe5d843

2021-07-25T13:29:38.413+02:00 REPORT RequestId: 3106f203-3c2a-4784-84d6-6b184fe5d843 Duration: 991.67 ms Billed Duration: 992 ms Method: POST

2021-07-25T13:30:25.157+02:00 START RequestId: 45a34959-0493-4aaa-8c08-2695ee139f22 Version: \$LATEST

2021-07-25T13:30:26.164+02:00 {

2021-07-25T13:30:26.164+02:00 "predictions": [[0.999996781]]

2021-07-25T13:30:26.164+02:00]

2021-07-25T13:30:26.164+02:00 }

2021-07-25T13:30:26.165+02:00 Je te déteste, tu es très méchant

2021-07-25T13:30:26.165+02:00 1

2021-07-25T13:30:26.173+02:00 END RequestId: 45a34959-0493-4aaa-8c08-2695ee139f22

2021-07-25T13:30:26.173+02:00 REPORT RequestId: 45a34959-0493-4aaa-8c08-2695ee139f22 Duration: 1008.94 ms Billed Duration: 1009 ms

ترجمة Google

مستندات نص

الإنجليزية العربية

البرتغالية

الإنجليزية العربية

الإنجليزية الألمانية

الإنجليزية البرتغالية

☆ Eu te odeio, você é muito mau × I hate you , You are very mean

مساهمة تم الحفظ السجل

إرسال تعليقات

ترجمة Google

مستندات نص

الإنجليزية العربية

الإنجليزية

الإنجليزية الفرنسية

الإنجليزية العربية

الإنجليزية

☆ Je te déteste, tu es très méchant × I hate you , You are very mean

مساهمة تم الحفظ السجل

إرسال تعليقات

Figure 21 Non-English Text Moderation Example