**NLP based matching of Job Descriptions with Skills and Profiles using Amazon SageMaker**

GUS Education has a Job Portal, used by recruiters and potential candidates and professionals. Recruiters use the job portal to upload job notifications with descriptions. Candidates use the job portal to create their profile with their latest skills. They were facing a challenge in mapping the right job description to the candidate based on their skills.

The customer requirement was that a candidate visiting a prospective Job notification should be able to see the skills they satisfy, and their skill gaps with respect to the job notification. This would help in mapping the right candidates to the right jobs, which enables savings in both candidate’s and recruiter efforts leading to a better user experience. This also reduces the cycle time to fulfill vacant position. The mapping of the job description to the right set of technical skills was a difficult problem to solve and ML based solution was recommended for this.

**Use case overview**

MIND discussed the problem with the customer and after analyzing the available data, it was determined that an NLP based ML solution would fit the business problem. Solution flow proposed consisted of the following steps,

* Pre-processing of job descriptions data - i.e. removing special characters, tokenize each word, remove HTML tags, case conversion.
* Labelling of Job descriptions data to skills using EMSI API - This provided the labelled data for job descriptions for initial training until the model was created. We get Labelled data with start char and End char position information of tagged data.
* Pre-Processing for model training
  + Removing data points where NER tagging is empty.
  + Converting NER tagging to BIO/IOB format (short for inside, outside, beginning) is a common tagging format for tagging tokens.
  + Extracting features from existing pre-trained BERT base cased model.
  + Dividing the data into 90% training and 10% testing data.
* Model Building: ML model was trained (NER) using Blazing Text and BERT.
  + With the available data, the model is trained on BERT for 25 epochs along with validation of the model.
  + Hyper parameters were tuned iteratively to optimize the model.

Diagram

Description automatically generated

Fig 1. State Machine in Step Function for Retraining and Deployment

* Model Deployment: Deployed the ML model using Flask API, the API takes job description as an input and outputs Technical Skills.

**Solution overview**

* **Amazon SageMaker**
* Amazon SageMaker is used to create and manage Jupyter notebooks that were used to prepare and process data and to train and deploy the machine learning models.
* Amazon SageMaker High Power GPU Instance used for training of BERT Model and is optimized using Amazon SageMaker Hyper Parameter Optimization Service
* Final Model is deployed on Amazon SageMaker using Flask API

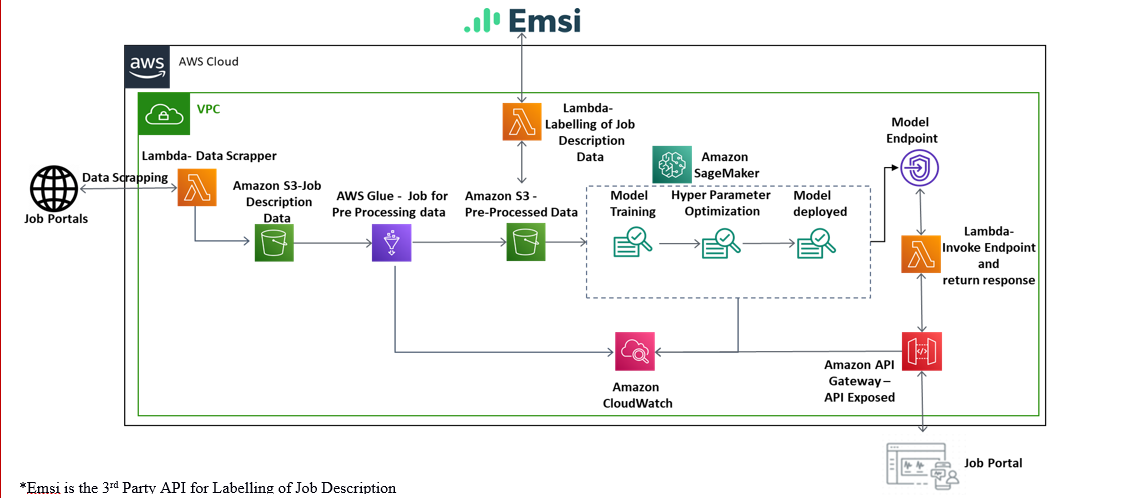


Fig 2. Architecture Diagram

* **AWS Lambda to handle the backend API calls**

It helped to initialize and validate the input and acted as the backend of the whole task. AWS Lambda lets us run code without provisioning or managing servers. Also, it helped to connect with various AWS API’s to acquire various insights from the inputs.

* **Amazon S3 to store CSV raw documents**

It is an object storage service that offers industry-leading scalability, data availability, security, and performance.

* **AWS API Gateway**

It helped in exposing the model endpoint. This API will be used in backend of the job Portal for enhancing user experience.

* **AWS CloudWatch**

It helped in monitor all the metrics related to model training, model endpoint and API logs.

**Methodology**

1. Data Scrapping & Storing to AWS S3
2. Primary Data Exploration
3. Clearing Issues, observations, Queries, pending items mentioned in attached Excel
4. Data Exploration - with Selected key Features
5. Creating a Job for preprocessing of the Job Description including:
   * Removing special characters
   * Tokenize each word
   * Remove HTML tags
   * Make everything to lowercase
6. JD Scraped data from Job portal is fed into EMSI API Skill extractor to be labelled data (technical skills) extractor
7. Data Modelling –Multiple Iterations

We had unstructured data (Job description text) with us, we evaluated below methods to extract skills:

* + Blazing Text
  + Pre-trained NER language models ([Elmo](https://allennlp.org/elmo) / [BERT](https://arxiv.org/abs/1810.04805))

1. Training the Model

A screenshot of a computer

Description automatically generated with medium confidence

Fig 3. Model Training Completed

A picture containing text, screenshot, indoor

Description automatically generated

Fig 4. Model Created Successfully

A screenshot of a computer

Description automatically generated with medium confidence

Fig 5. Endpoint Configuration

Graphical user interface, text, application

Description automatically generated

Fig 6. Model Deployed Successfully

1. Test Model / Model Accuracy
2. Optimized the model using Hyper Parameter Optimization
3. Deployed the ML model using Flask API, the API takes job description as an input and outputs Technical Skills.

Graphical user interface, text, application, email

Description automatically generated

Fig 7. Successfully test Model through Postman

Graphical user interface, text, application, email

Description automatically generated

Fig 8. Model Deployment Image in Amazon ECR

**Additional Considerations**

We selected BERT Model because of the following reasons,

* Got the Best accuracy in BERT Model
* BERT handles out of vocabulary words very easily unlike static embedding
* BERT understands the context of a sentence.
* We can train BERT on pre-train model released by Google.
* State of the art model on benchmark compared to others.
* BERT handles context by reading the data from left to right and right to left. For e.g. in”We are a Microsoft certified organization”– Microsoft Certified is not a skills, BERT understand this perfectly.

BERT model is trained on large corpus of data and it takes less training time. Since it’s a state-of-the-art model, it provides better results. The below test cases make sure the deployment is robust.

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Test Data** | **Expected Result** | **Actual Result** |
| Verify response when a valid Job Description is passed to model API | Job Description | Named Entity from Job Description | Named Entity from Job Description |
| Evaluated Model on Evaluation Data Set of Job Description | Evaluation Data Set | Eval\_Loss,  Precision,  Recall,  F1\_score | 'eval\_loss': 0.22,  'precision': 0.80,  'recall': 0.81,  'f1\_score': 0.80 |

**Conclusion**

In this post, we showed how to use AWS services to successfully create a solution with all the customizations and hyper parameters tuning by training a model with which we were able to extract the Technical skills with up to 80% accuracy. We saved up to 60% of human effort which was used to manually enter Technical skills. Also, we saved up to 90% of training cost by using spot instances for training.