ENIGMA

Knowledge-Based Question Answering System project
Final report

17.7.2023

Contents

1 Executive Summary

This is the final report for EnigmaAI's delivery of the Knowledge-Based Question Answering System Service for SingularityNET's AI Marketplace which was submitted as a proposal on February 12th, 2023. In the introduction, we provide a short explanation of the project and its goals. Then we explain the algorithm, its purpose, and its theoretical basis in information theory.

2 Introduction

This service is an extension to an already running service on EnigmaAI Platform, It is intended to work as a knowledge-based question-answering system, i.e., the customer would submit an inquiry, the system then matches the meaning of the user's question with the most similar question from an embeddings database which it was trained on using Semantic Similarity task and returns the ID of that question. The original service required that a separate model would be deployed for each new user limiting the scalability of the service making it very expensive and unreachable for small businesses. To address this, we propose a solution that incorporates two key components: the use of a new training method called Adapters and Parameter Efficient Tuning [2] for the BERT [1] model. To implement this solution, our team consists of a diverse group of data scientists and MLOps experts.

The service pipeline allows the user to fine-tune the model on his/her data and then efficiently deploy it on the cloud (AWS Sagemaker). Onboarding this service onto the SingularityNET platform offers numerous benefits. SingularityNET is renowned for connecting various AI services, fostering mutual enhancement and inspiration. The service can be utilized by end users or integrated into other services on the platform. This collaboration contributes to the growth of both the service and the platform. Moreover, the platform's community can contribute to improving the technical capabilities of the model and expanding its user base.

The service is licensed under the MIT license, providing flexibility and openness to users. In terms of marketing and competition, the chatbot market in the Middle East and Africa (MEA) remains largely untapped. The company possesses a competitive advantage in understanding and addressing the unique literacy challenges faced by customers in Sudan. Various use cases across different sectors have been validated, showcasing positive outcomes such as sales automation and business process transformation. Moving forward, the company aims to enhance the user experience and interface of its platform, while launching a comprehensive marketing campaign to emphasize the value of chatbots for business owners.

In conclusion, our efforts have culminated in the development of a collaborative and democratized AI ecosystem. This ecosystem harnesses the power of AI technologies to tackle complex challenges and foster innovative solutions across diverse domains.

[width=10cm]figures/Deep fund.drawio (3)

Figure 1: Pipeline

3 Methodology

In this section, we describe the pipeline in our approach to fine-tuning Language models for different downstream tasks, and API

3.1 Data collection

The user data is initially organized into different classes, represented in a table format as shown below:

Question	***Group***	***Id***
I need to exchange Japanese Yen for British Po...	Forgin currency exchange	5
Your services are awful.	complaint	2
Do you have a branch in Halfa Al Jadida?	Branch location	9
Where can I find your phone number?	Phone number + number not working	3
What amount is deposited to open the account?	Account opening	6
Are there any internship opportunities for stu...	Job application	13
Im inquiring about investment deposits.	Investment deposit	11
Thank you for your help.	Thanks	12
I want to know the location of your branch in ...	Branch location	9
I reside in another country, can I apply for a...	Account opening from abroad	7

Table 1: Questions and Groups

To prepare the dataset for training using the triplet loss function, we perform a preprocessing step. This involves transforming the data from the class format into the anchor, positive, and negative format. The anchor represents the reference item, the positive represents a similar item, and the negative represents a dissimilar item. This format is required for training our model's embeddings.

Additionally , we also apply a form of dynamic under-sampling on the generated triplets. This step serve two purposes: i) Balance the data so that all classes are represented equally as the anchor , and ii) Reduce the size of the training set to regularize and reduce the computational resources for the training step.

The processed dataset, after applying the preprocessing step, is shown in the table below:

Table 2: Processed Dataset

	Anchor	**Positive**	**Negative**
439	Excellent Thank you	Thank you very much. I received it yesterday and completed my transaction smoothly. May Allah reward you with goodness.	Do you have any job openings?
62	You guys are annoying	Aren't you afraid of God or what?	I want to travel quickly.
108	Your phone 6100 is not working. Do you have another phone?	The number is not available.	Hello, I want to open an account.
197	What is required to open a foreign account?	I want to ask about opening an account with you. How much does it cost?	I want to ask about financing.
126	I want to speak to someone from customer service.	The number is not available.	I want to ask about opening an account with you. How much does it cost?
4	Hello	How are you?	The phone numbers on your website are not working.
69	Why don't you mind your own business?	Annoying, may you be annoyed.	What is the price of Euro?

In this processed dataset, each row corresponds to a triplet of items: the anchor, positive, and negative. The anchor column contains the reference statements or questions, the positive column contains similar statements or questions, and the negative column contains dissimilar statements or questions. This format allows us to train our model to learn meaningful embeddings that capture the semantic similarities and differences between statements or questions.

3.2 Parameter Efficient Tuning

Once the dataset is prepared, the next step is to fine-tune the model for the classification task. In our approach, we utilize LoRA PEFT [3], which allows us to fine-tune only a portion of the model's layers while keeping the remaining layers frozen. This strategy is particularly useful when dealing with small datasets. For such cases, we recommend fine-tuning a small portion of the model for a shorter number of epochs. Conversely, when the dataset is larger, a larger portion of the model can be fine-tuned for a longer duration.

Currently, we leverage the sentence-transformers library. Our test dataset consists of various labels, and we train the model using triplet loss. This loss function adjusts the weights of the model such that sentences with similar meanings end up closer to each other in the embedding space, while sentences with different meanings are placed further apart. Consider a scenario with K classes, where all sentences within the same class are considered positive with respect to the anchor, and negative with respect to sentences from other classes. The model fine-tunes the embeddings to ensure that classes with closer meanings have embeddings that are closer to each other.

The figure below illustrates the resulting embeddings:

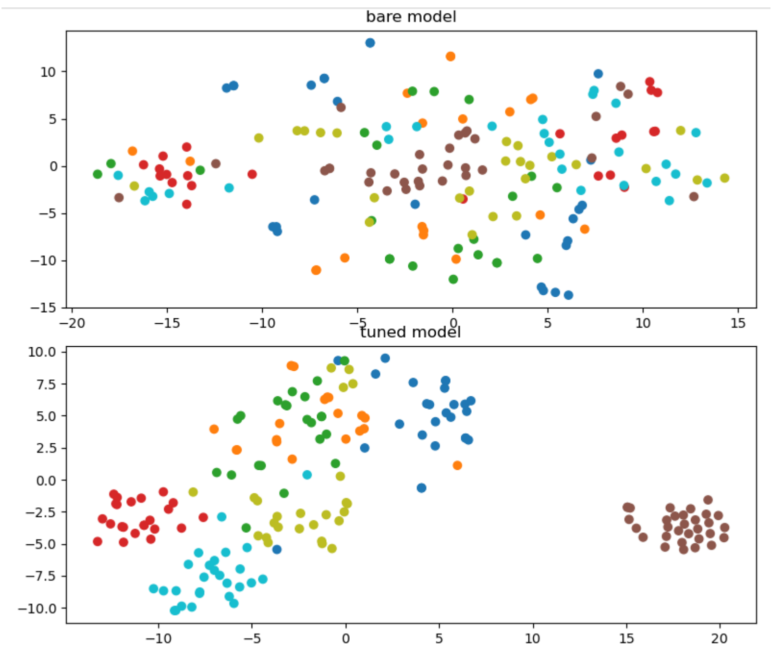


Figure 2: Embeddings Visualization

This visualization demonstrates how the embeddings are arranged in the embedding space, with classes exhibiting closer embeddings when their meanings are similar.

3.3 Deployment

This service was hosted on Amazon Sagemaker as part of AWS services, the deployment process includes creating a machine learning training and deployment pipeline that the user can invoke using a simple API to train a model on his/her custom data and deploy it to a serverless endpoint.

Table 3: Training Parameters

Parameter	Value
Model	MiniLM-L6-v2
Batch Size	16
Epochs	10
Warmup Steps	100

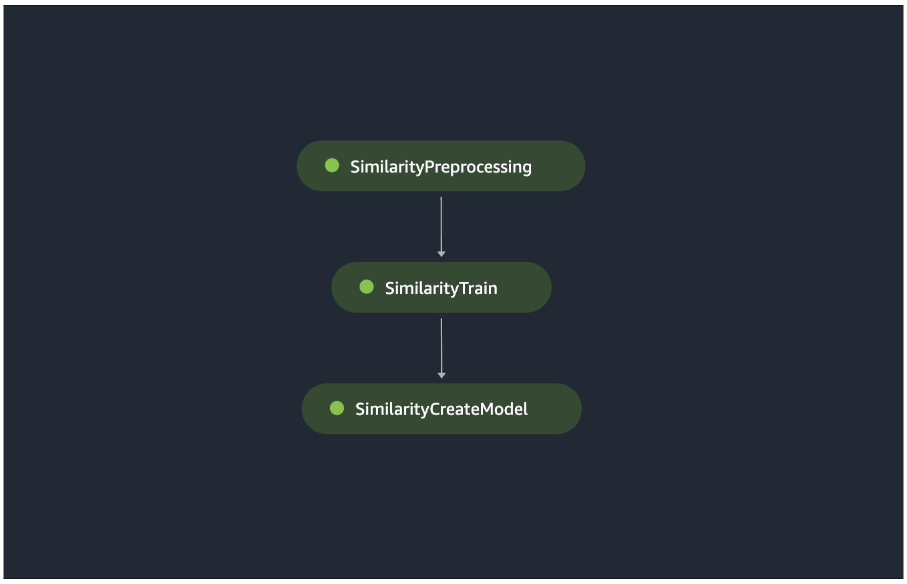


Figure 3: training and deployment pipeline

The pipeline incorporates three steps:

- Data Preprocessing step (SimilarityPreprocessing): where the user data is uploaded, cleaned, and prepared in the triplet loss form described in section3.1.
- Training Step (SimilarityTrain): here the model is trained using the supplied data as described in section 3.2.
- Deploy to Endpoint (SimilarityCreateModel): The model is then deployed to an elastic serverless endpoint accessible to the user through SingularityNET's marketplace API.

3.4 API

TBD

4 Results

4.1 Evaluation Metrics

In order to assess the performance of our model and compare it with existing baseline, we employ several evaluation metrics. We aim to comprehensively analyze and compare the performance of our models, providing insights into its accuracy, interpretability, and computational efficiency.In this subsection, we introduce and briefly explain the following metrics:

Accuracy: Accuracy is a widely used metric in machine learning and classification tasks. It measures the proportion of correctly assigned instances out of the total number of instances in the evaluation dataset. In our case, we evaluate the model's ability to assign input sentences to their corresponding classes based on their similarity in the embedding space. The accuracy score indicates the model's effectiveness in correctly classifying the sentences.

Cosine Similarity Distance: Cosine similarity distance is a metric utilized to quantify the similarity between two vectors in a high-dimensional space. It measures the cosine of the angle between the vectors, resulting in a value between -1 (completely dissimilar) and 1 (completely similar). In the context of our evaluation, cosine similarity distance is employed to measure the similarity between sentence embeddings within a specific class and the distances between different classes. Specifically, we refer to the similarity within a class as intra-similarity and the similarity between different classes as inter-similarity. These distances provide insights into the separability and clustering of the sentence embeddings in the embedding space.

Efficiency Measures: Efficiency measures refer to metrics that assess the computational efficiency and resource requirements of a model. These metrics help evaluate the speed and scalability of the model, particularly in scenarios where real-time processing or limited computational resources are crucial. Examples of efficiency measures include inference time, memory consumption, and processing power utilization.

4.2 Training Dataset

We utilized a custom pre-processed dataset for training purposes. The dataset followed the triplet format (anchor, positive, negative), containing a total of 7112 triplets. The dataset comprised 14 distinct classes, with each class represented by 508 triplets. The average sentence length was 5.5 words, while the maximum sentence length reached 26 words.

4.3 Model

The base model used in our training process was MiniLM-L12-v1. This model is a pretrained sentence-transformers model that maps sentences and paragraphs to a 384-dimensional dense vector space. It is suitable for tasks such as clustering or semantic search. The model was initially trained on a large and diverse dataset consisting of over 1 billion training pairs. The training process employed a contrastive loss function.

To fine-tune the model, we adopted the Parameter Efficient fine-tuning (Peft) technique. Peft enables the fine-tuning of large models with a limited number of parameters. Specifically, we employed the LoRa technique, a Peft variant that fine-tunes a small set of Low Rank matrices inserted in different layers of the model. We used a rank of 64 for the Low Rank matrices, focusing on the value, query, key, and dense layers. This resulted in a model with 5 million trainable parameters, equivalent to 13.5% of the original model's parameters.

4.4 Training Details

The model was trained for 10 epochs with a batch size of 16. We employed the AdamW optimizer with a learning rate of 1e-5. The training process utilized the triplet loss function, with a margin of 1.0 and a norm degree of 2. The maximum sentence length considered during training was 200 words. The training was conducted on a single GPU (Nvidia Tesla T4), and the entire process took approximately 2 hours and 30 minutes.

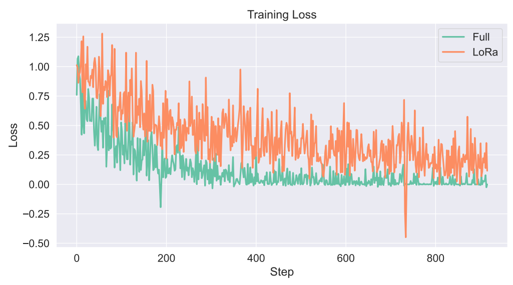
4.5 Results and Analysis

After training the model and evaluating its performance, we obtained the following results:

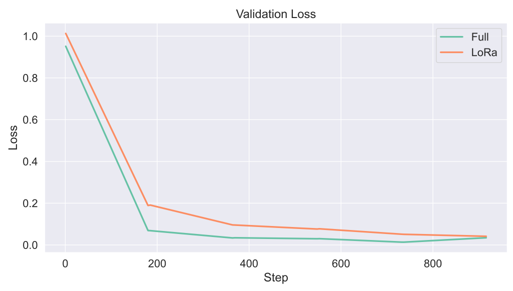
Accuracy: The accuracy of the model on the evaluation dataset was measured to be 94.55% intra-similarity and the similarity between different classes as inter-similarity

Metric	**LoRa**	**Full**
# Trainable Parameters	675,840	23,389,056
Size of Final Weights in MB	2.73	90.9
Training Time in Minutes	6.2	15.4
Memory Used for Training in GB	7.1	14.2
Evaluation Accuracy	96.15	96.15

Table 4: Efficiency Comparison



(a) Train Loss



(b) Validation Loss

Figure 4: Train and Validation Loss for Full Fine-tuning vs LoRa Fine-tuning

Inter-Similarity Distance The average inter-similarity metric was found to be 0.2366. This metric measures the average cosine similarity distance between sentence embeddings within the same class. A lower value (min:-1) implies that sentences belonging to the same class are more similar to each other in the embedding space, indicating that the model successfully captures the intra-class similarity.

Intra-Similarity Distance:The average intra-similarity distance metric was calculated to be 0.8399. This metric quantifies the average cosine similarity distance between sentence embeddings from different classes. A higher value (max:1) indicates a higher degree of separability between different classes in the embedding space, suggesting that the model is effective in distinguishing between different sentence categories.

4.5.1 Efficiency Evaluation

We were able to achieve significant gains in efficiency without compromising the performance quality by employing Parameter Efficient Fine-Tuning (PEFT) methods along with the triplet loss on pre-trained sentence similarity models. Specifically, the use of the LoRa variant of PEFT allowed us to reduce the number of trainable parameters from 3 million to 150,000. This reduction not only saved memory but also significantly reduced the training time. With the PEFT approach, we only required 7GB of memory and 6 minutes to achieve 96% classification accuracy on the validation split of our new and custom dataset. For a side-by-side comparison of these measures, refer to Table 4.

Furthermore, in Figure 4, we provide a visual comparison between full fine-tuning and efficient fine-tuning in terms of the evaluation loss and the intra-similarity distance. It is evident from the plot that while there is a small difference in performance between the two methods, both achieve the same final accuracy, as shown in the above table.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[3] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.