Advances in Informatics, Management and Technology in Healthcare
J. Mantas et al. (Eds.)

doi:10.3233/SHTI220714

Inference Time of a CamemBERT Deep Learning Model for Sentiment Analysis of COVID Vaccines on Twitter

Guillaume GUERDOUX[a], Théophile Tiffet[b] and Cedric BOUSQUET[b,c,1]
[a] *Geegz, Paris, France*
[b] *Unit of Public health, University hospital of Saint-Etienne, France*
[c] *Sorbonne Université, Université Sorbonne Paris Nord, INSERM, Laboratoire d'Informatique Médicale et d'Ingénierie des Connaissances pour la e-Santé, LIMICS, 75006, Paris, France*

Abstract. In previous work, we implemented a deep learning model with CamemBERT and PyTorch, and built a microservices architecture using the TorchServe serving library. Without TorchServe, inference time was three times faster when the model was loaded once in memory compared when the model was loaded each time. The preloaded model without TorchServe presented comparable inference time with the TorchServe instance. However, using a PyTorch preloaded model in a web application without TorchServe would necessitate to implement functionalities already present in TorchServe.

Keywords. Artificial Intelligence, MLOps, COVID-19, Social Media, Vaccines

1. Introduction

In previous work, we implemented a deep learning model for predicting the opinion of Twitter users on COVID vaccines before they were available on the market [1] with PyTorch and CamemBERT [2]. Using pretrained embeddings leads to larger number of parameters in the model which has an impact on inference time. For example, a simple model based on a Bi-LSTM network led to inference 434 times faster than using BERT [3]. Therefore, it is necessary to evaluate how recent models based on Transformers may perform after they are trained. We present a preliminary study comparing inference times with or without TorchServe, a library for serving PyTorch models.

2. Methods

We used an Elastic Cloud Compute instance on Amazon Web Services: a g4dn.xlarge with a Tesla T4 NVidia graphic card. Inference times were measured with and without using TorchServe. When the model was not deployed on TorchServe, we compared inference time when the is preloaded in memory (optimized approach), and when the

[1] Corresponding Author, Dr Cedric Bousquet, SSPIM, Bâtiment CIM42, chemin de la Marandière, Hôpital Nord, 42055 Saint Etienne, France; E-mail: cedric.bousquet@chu-st-etienne.fr.

model is loaded each time we make a new prediction (naïve approach). A different number of tweets (100 vs. 1000) was used to evaluate the impact of the load.

3. Results

The mean inference time by tweet (third column) is described in Table 1 according to the number of tweets (first column), and model loading (second column) with and without TorchServe (optimized vs. naïve approach). Without TorchServe, inference time was three times faster when the model was loaded once in memory compared when the model was loaded each time. The preloaded model without TorchServe presented comparable inference time with the TorchServe instance.

Table 1. Mean inference time by tweet with or without Torchserve

Number of tweets	Model loading	Mean Inf. Time by tweet (ms)
100	Optimized approach	18,70
100	Naïve approach	66,70
1000	Optimized approach	13,00
100	TorchServe	17,00
1000	TorchServe	15,40

4. Discussion

The naïve approach consists in reloading the neural network model in memory at each call. This is what would be done in a classic web application (without memory persistence) where each API call is independent. The optimized approach consists in manually preloading the neural network in memory and therefore make it available for each future call. Although performances were comparable with and without TorchServe, the optimized approach with preloading of the neural network has some shortcoming.

First, the handling of the neural network is firmly coupled to the code of the application. On the contrary TorchServe, by proposing an API for inference, can be deployed on a server with a graphics processing unit, while having our web application and user interface on another server, in a microservices architecture. Second, this architecture is highly scalable and can easily be grown to fit our requirements. Finally, TorchServe is built to manage calls in parallel, a feature that the user will have to implement in the optimized version, which amounts to redeveloping TorchServe.

References

[1] Dupuy-Zini A, Audeh B, Gagneux-Brunon A, Bousquet C. Users' Reactions on Announced Vaccines against COVID-19 Before Marketing in France: Analysis of Twitter posts. medRxiv. 2022 Jan 1. 2022.02.14.22268832. doi: https://doi.org/10.1101/2022.02.14.22268832.

[2] Martin L, Muller B, Suárez PJ, Dupont Y, Romary L, de La Clergerie ÉV, Seddah D, Sagot B. CamemBERT: a tasty French language model. arXiv preprint arXiv:1911.03894. 2019 Nov 10. arXiv preprint. 2019 arXiv:1911.03894. 2019.

[3] Tang R, Lu Y, Liu L, Mou L, Vechtomova O, Lin J. Distilling task-specific knowledge from bert into simple neural networks. arXiv preprint arXiv:1903.12136. 2019 Mar 28. arXiv preprint. 2019 arXiv:1903.12136.