Ingénierie des Systèmes d'Information
Vol. 29, No. 4, August, 2024, pp. 1357-1364
Journal homepage: http://iieta.org/journals/isi

PyTorch and TensorFlow Performance Evaluation in Big data Recommendation System

Hoger K. Omar[1,2*], Mondher Frikha[3], Alaa Khalil Jumaa[4]

[1] ENETCOM, ATISP Research Lab, University of Sfax, Sfax, Tunisia
[2] Department of Computer Science, College of Computer Science and Information Technology, University of Kirkuk, Kirkuk, Iraq
[3] Department of Electronics, National School of Electronics and Telecommunications of Sfax, University of Sfax, Sfax, Tunisia
[4] Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani 46001, Iraq

Corresponding Author Email: hogeromar@uokirkuk.edu.iq

https://doi.org/10.18280/isi.290411

Received: 27 February 2024
Revised: 6 June 2024
Accepted: 30 July 2024
Available online: 21 August 2024

Keywords:
big data, deep learning, NLP, PyTorch, recommendation system, TensorFlow, text mining

ABSTRACT

Over the past decade, there has been a renewed interest in Artificial Intelligence and its diverse frameworks. This research proposed a big data recommendation system that leverages two prominent deep learning frameworks which are PyTorch and TensorFlow to enhance collaborative filtering (CF) outcomes. The goal is to recommend more relevant products to users by incorporating textual review comments and combining various attributes. The study explores two recommendation system approaches within each framework. In the PyTorch framework, two collaborative filtering models were developed: one without incorporating user review text and the other with the inclusion of text from user reviews. Similarly, for the TensorFlow framework, two recommendation models were created: one without utilizing user review text, and the other with the integration of text reviews. The dataset, sourced from the Amazon website, comprised over 600,000 ratings and reviews. The outcomes showed significant enhancements by employing the proposed text-based method in both PyTorch and TensorFlow frameworks and addressed challenges such as dynamic preferences and data sparsity. Finally, the study provides a comparative analysis of the advantages and drawbacks of each utilized framework.

1. INTRODUCTION

The term big data refers to an extensive volume of data that surpasses the capabilities of traditional data management methods. Primary sources for generating big data include conventional business systems, the Internet, social networks, and the Internet of Things [1]. This substantial increase in data provides a significant opportunity for scholars and analysts to gain a profound understanding of user opinions and interests [2]. Therefore, recommender systems (RS) can address the challenge of information overload in big data scenarios by utilizing user historical data sourced from social networks. In data-driven markets, digital platforms, and competitors frequently leverage proficient recommendation engines to offer valuable suggestions. Consequently, recommendation engines have emerged as potent tools for internet companies [3, 4]. Fundamentally, recommendation systems are software applications that endeavor to recommend the most fitting items to individual users. This is achieved by predicting their interest in an item based on pertinent information about similar items [5].

Conventional RS can be constructed by developing a model using typical data, which encompasses user, item, and user preferences such as ratings [6]. These stated attributes in the input data may not provide sufficient information for achieving exact outcomes. Relying on single form input data lacks the necessary details and is susceptible to noisy data. Therefore, it is more favorable to incorporate supplementary information into RS to increase accuracy and mitigate potential inaccuracies associated with limited data [7]. Thus, this study recommends the inclusion of text data to enhance accuracy. When dealing with large text data, it is essential to employ Natural Language Processing (NLP) techniques as a first stage for proficient data examination. The significance of NLP has been explored and determined to be essential for the effective processing of large-scale data. NLP utilizes various methods to interpret uncertainties in human language and contains many techniques such as textual summarization, and voice recognition [8]. Essentially, the recommendation system employs four different filtering types specifically collaborative, content-based, demographic, and hybrid [9]. CF models evaluate similarities between various users by investigating their ratings, and then they make predictions for new recommendations based on the relationships between users [10]. The two primary categories of CF algorithms are memory-based CF and model-based CF. In memory-based CF, historical behavioral data is employed to identify potential connections between users and items, forming the basis for generating recommendations. This type of CF can be further classified into two subcategories: user-based CF (UCF) and

item-based CF (ICF). Multiple models have been devised for both UCF and ICF. UCF models concentrate on identifying items favored by similar users and predicting the ratings a target user might assign to those items. Conversely, ICF models ascertain the similarity between items and utilize the ratings provided by the target user to recommend items with high similarity. Model-based CF, also known as a learning based model, entails defining a parameter model to characterize the user-item relationship and subsequently optimizing these model parameters. Various machine learning algorithms, including matrix factorization, neural networks, and clustering algorithms, are grounded in the model-based CF approach [11, 12]. Recommendation systems confront a range of hurdles and challenges, encompassing issues such as accuracy, sparsity, dynamic preferences, cold-start problems, scalability, and various other related challenges [13]. Moreover, traditional collaborative filtering methods grapple with these challenges, particularly the issue of data sparsity. This challenge arises because each user only provides ratings for a restricted number of available items [14, 15]. Furthermore, the user's preferences are distinct and suffer from dynamic changes in response to the characteristics of the interaction. This uniqueness and dynamism highlight the need for adaptive systems that can effectively capture and accommodate evolving user preferences. Developing mechanisms to continuously monitor and adapt to these changing preferences is crucial for enhancing the overall user experience and ensuring that recommendations remain relevant and personalized over time [16]. Hence, to address these problems and mitigate the dynamic preference changes and data sparsity issues, employing the proposed text-based methods of this study emerges as a viable and effective solution.

In recent times, scholars have started employing Deep Learning (DL) with its frameworks are used to enhance the capabilities of many fields. This work tests two DL frameworks which are PyTorch and TensorFlow in big data collaborative filtering recommendation systems. DL models with varied operational rules, such as Boltzmann machines, stacked auto encoders, and Multi Layer Perceptron, are applied to acquire characteristic representations among users and items [17]. A deep neural network (DNN) is chosen because of its effectiveness in handling scenarios with a high level of complexity or a large volume of training data [18]. The existence of online reviews and comments on various platforms underscores the common use and impact of text as a significant medium for users to convey their opinions. Considering that text mirrors users' needs and emotions, the integration of text mining techniques stands as a powerful means to considerably improve the accuracy of recommendation systems [19]. Predicting user preferences solely through the use of numerical data such as rating can pose significant barriers creating challenges in accurately capturing the full spectrum of user preferences [20]. Hence, numerous researchers have incorporated the use of auxiliary data and the integration of information from diverse attributes, recognized as effective solutions for addressing challenges related to accuracy and sparse data [21]. Therefore, including user reviews in this study as input could be advantageous, as it provides extra information about the product, aiding in more accurate recommendations. Extracting meaningful details from the review text also allows for the creation of a more comprehensive user-item feature matrix that takes into consideration both the ratings and textual information.

The core contributions of this work can be summarized as follows:

-Combine advanced text-mining and NLP methods with two famous deep learning frameworks: PyTorch and TensorFlow to find an effective recommendation system model.

-Evaluating and comparing the effectiveness of the big data recommendation system in both PyTorch and TensorFlow deep learning frameworks.

The structure of this article is outlined as follows: Section 2 explores a thorough examination of the current literature concerning recommendation systems, PyTorch, and TensorFlow frameworks. Section 3 delivers a complete clarification of the deep learning frameworks used, specifically PyTorch and TensorFlow. In Section 4, a comprehensive depiction of the proposed system architecture is provided. Section 5 showcases the experimental results, and lastly, Section 6 shows the conclusion of the work.

2. RELATED WORK

In recent times, a considerable number of scientific articles have been published on subjects such as the PyTorch framework, TensorFlow framework, big data, and recommendation systems. These articles have introduced diverse methodologies for constructing proficient models. This section will concentrate on examining the most esteemed contributions within these domains.

Omar et al. [22] proposed an innovative big data RS that improves CF outcomes by integrating NLP methods and managing multiple-attributes. It developed two big data RS models using ML algorithm. Both models employed the Alternating Least Squares (ALS) algorithmic rule within the Spark big data context. The first model did not utilize NLP methods, whereas the second model incorporated advanced NLP methods by analyzing user-review-comments. A dataset comprising more than 3 million ratings and reviews, totaling 3.1 GB, was collected from the Amazon website. Novac et al. [23] presented an analysis of critical aspects that emerge during the development of neural network applications. The objective is to assess whether the selection of a library can influence the overall performance of the system, whether during training or design. Initially, they compared two widely used neural network libraries, PyTorch and TensorFlow. The goal is to select a set of measures that can effectively showcase the advantages and disadvantages of each library under consideration. Omar et al. [24] Suggested a prototype RS hosted on the cloud designed to manage substantial big data volumes. This system utilizes matrix factorization through three distinct methods: singular value decomposition (SVD), ALS implemented with Spark, and DNN leveraging TensorFlow. Through the optimization of algorithms and parameters in the ALS and DNN methodologies, the study effectively tackled the issue of handling extensive collaborative filtering datasets. The outcomes from these two methods surpassed the performance of conventional techniques while maintaining a satisfactory level of computational efficiency. Liang et al. [25] proposed a novel recommendation system named deepLTRS, leveraging a deep generative latent model known as variational auto-encoder (VAE). The incorporation of topic modeling in the review segment is achieved through the application of the ProdLDA algorithm to enhance the quality of topics. The numerical

experiments conducted on both simulated and real-world datasets, provide evidence of the effectiveness of their proposed model. Mahon et al. [26] Conducted scalability assessments on various prominent deep learning frameworks (TensorFlow, Keras, MXNet, and PyTorch) using state-of-the-art High Performance Computing (HPC) resources. Their goal was to compare the efficiency of diverse implementations across various hardware architectures, including Central Processing Units (CPUs) and Graphics Processing Units (GPUs). Nassar et al. [27] proposed a novel collaborative filtering recommender system that incorporates multiple criteria by combining deep neural network and matrix factorization techniques. The model comprises two distinct components. The first component utilizes a fused model of deep neural network and matrix factorization to estimate ratings for various criteria. The second component employs a deep neural network to predict the overall rating. Chirodea et al. [28] presented a comparison between the PyTorch and TensorFlow environments utilized for defining neural networks. The aim is to investigate whether the choice of a library impacts the overall performance of the system during both the training and design phases. This involves analyzing the processes involved in creating a neural network, measuring outcomes, and monitoring its evolution over epochs. Subsequently, advantages and disadvantages are extracted from their obtained results. Osman et al. [29] used a sentiment based model with added contextual data to tackle domain sensitivity. This implies that their new approach combines the sentiment based model with contextual data. The test aimed to assess the efficiency of the standard rating model, sentiment model, and contextual data model. The outcomes showed that the proposed contextual information sentiment based model performed better than the traditional CF method in terms of recommendation accuracy. Sharma et al. [30] utilized natural language processing technologies along with CNN to predict similar products. In the analysis and recommendation process, particular emphasis is placed on the product's text title as a significant attribute. CNN is employed to create a feature vector from the product images, which is then integrated with other vectors for making predictions. For recommendations, a comparison of the distance between vectors for all products is conducted, suggesting those with the least distance. Abbasi-Moud et al. [31] proposed a personalized tourism RS that tailors suggestions according to individual preferences. User reviews sourced from tourism-centric social networks are leveraged as a valuable data repository for extracting user preferences. These reviews undergo preprocessing, semantic grouping, and sentiment analysis to discern travelers' preferences. Likewise, reviews gathered from all users regarding specific attractions are employed to select the primary characteristics of these points of interest. In the end, the RS proposed semantically compares a user's preferences with the characteristics of attractions to suggest the most fitting options.

3. PYTORCH AND TENSORFLOW FRAMEWORKS

PyTorch, launched in 2016, is grounded in the principle of maintaining a simple API that can be effortlessly customized and updated to align with the latest developments in the field of artificial intelligence [32]. Overall, PyTorch serves as a comprehensive framework for both ML and DL. With its user-friendly interface, distributed training capabilities, and a rich ecosystem of tools and libraries, PyTorch facilitates seamless and versatile experimentation, as well as efficient production [33]. Additionally, PyTorch, recognized as an emerging Python library and framework, executes effective GPU-based tensor computations and streamlines the construction of neural architectures by offering robust mechanisms for automatic gradient computation. A distinctive feature of PyTorch is its flexible design, inherently accommodating the creation of dynamic neural networks [34]. Moreover, deep learning frameworks that are graph-based such as PyTorch typically feature an automatic differentiation engine. This engine preserves data dependencies in the form of a graph and offers an API that allows users to invoke algorithms for exploring the mathematical operations history and computing derivatives with a single line of code [35].

On the other hand, TensorFlow is a flexible and scalable software library designed for numerical computations utilizing dataflow graphs. This library, along with its associated tools, allows users to efficiently program, train, and deploy neural networks and other machine learning models. The core algorithms of TensorFlow are written in highly optimized C++ and CUDA (Compute Unified Device Architecture), which is a parallel computing platform and API developed by NVIDIA. TensorFlow provides application programming interfaces (APIs) in multiple languages, and the Python API stands out as the most complete and reliable option. Moreover, one of the benefits of TensorFlow is its integration with the Keras package [36, 37]. TF supports deep neural networks and a diverse range of machine learning techniques, along with powerful computational capabilities for managing various datasets. It also incorporates unique features for efficient memory management and data optimization. Additionally, TF is compatible with various operating systems, such as Ubuntu, macOS, and Windows, allowing users the flexibility to install and run TensorFlow with either CPU or GPU support [38]. However, TensorFlow offers a TensorBoard visualization API. This tool can exhibit the input and output tensor variables at each node. Additionally, it can reveal the dependencies between tensor operations through edges [39].

4. PROPOSED RECOMMENDATION DESIGN

This section introduced the design of a book recommendation system for big data that employs two deep learning frameworks along with natural language processing. The proposed framework includes both PyTorch and TensorFlow, as illustrated in Figure 1, outlining the overall system steps. It is clear that the framework involves nine distinct steps:

Step 1: Google Colab Pro Plus which is known as a paid Jupyter notebook platform that operates entirely in the cloud, has been utilized in this work.

Step 2: Acquiring the dataset, which is about 1 gigabyte and includes over 650,000 rows of user reviews and ratings obtained from the Amazon website which enables the researchers to use the data for scientific intentions.

Step 3: Generally, the original data may comprise many irrelevant attributes that do not impact the efficiency of the recommendation system. To expedite the processing time, all these non-essential attributes are removed.

Step 4: To enhance the accuracy of the utilized models while dealing with unstructured datasets, various preprocessing techniques have been implemented. These

encompass tasks such as data cleaning, data normalization, feature extraction, etc.

Step 5: The proposed study encompasses two deep learning frameworks. The first one utilizes the PyTorch framework, while the second employs the TensorFlow framework. In this phase, the essential environments and dependencies for both approaches were installed to build the models.

Step 6: Both frameworks employed a DNN algorithm with a batch size of 64 and 10 epochs. However, the training dataset size is 80 % and the remaining is used for the testing set. In the case of PyTorch, two models were proposed: the first model excluded NLP methods, while the second model integrated NLP methods, considering the texts written by users along with the other relevant attributes such as user ID and product ID. The same procedure was mirrored for the TensorFlow, one model incorporating NLP techniques and the other without. The integration of the texts in both frameworks has been done by using the column of the text comments in the dataset with the other traditional columns which are always used in the recommendation system. Because the mentioned text column consists of rich information that helps the system to recommend more accurate items to the users.

Step 7: After the processing phase, each framework produced two recommendation lists, each corresponding to a specific model. Additionally, it mentions that a total of four recommendation models were constructed.

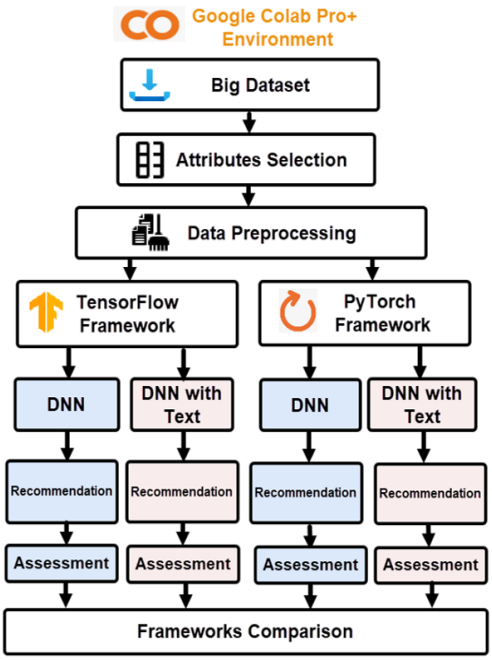

Figure 1. The proposed recommendation system using both PyTorch and TensorFlow frameworks

Step 8: At this point in the process, an individual evaluation is conducted for each list of recommendations. This involves a comprehensive examination of the suggested item selections within each framework to gauge their effectiveness and relevance. The aim is to assess the quality and appropriateness of the recommendations provided by each specific model. This thorough evaluation contributes to the overall analysis of the recommendation system's performance and its ability to meet the desired criteria for accuracy and user satisfaction.

Step 9: A comprehensive assessment was performed to scrutinize and compare the outcomes produced by the models within each framework. This involved a meticulous examination of the recommendations generated by the models in PyTorch and a parallel analysis of the models implemented in TensorFlow. The purpose of this comparative analysis was to discern any disparities in performance, strengths, or weaknesses between the two deep learning frameworks. By conducting a thorough examination of these aspects, valuable insights were gained into the relative effectiveness and nuances of the recommendation models, contributing to a more informed understanding of their respective capabilities and potential areas for improvement.

5. RESULT ANALYSIS AND COMPARISON

In the next subsections, the details of the experimentations and outcomes are introduced. Subsection 5.1 displays the system environment However, subsection 5.2 provides an explanation of the dataset. Finally, subsection 5.3 demonstrates the analysis and comparison of the results.

5.1 System environment

The integrated development environment (IDE) utilized throughout this project was Google Colab Pro Plus, featuring a Graphical Processing Unit and Tensor Processing Unit run-time environment. This paid Jupyter Notebook platform operates entirely in the cloud, eliminating the need for any setup. Leveraging Google Colab enables users to write and accomplish code, preserve and share their studies, and get influential calculating resources directly through their web browsers [40].

This Framework is projected for scripting and executing the Python program codes, offering three designs: a free version and two premium options named Colab Pro and Colab Pro Plus. The preference for Colab Pro Plus in this context is due to its provision of additional resources essential for this work, along with a more generous usage-limit compared to the free plan. The maximum usage limits for Colab Pro Plus VMware are 52 GB RAM, 1TB data storage, and P100- T4-V100 GPU usage with 24 hours of session time [41].

5.2 Dataset explanation

In this work, actual data was utilized to validate the efficacy of the proposed approaches. The dataset was sourced from the Amazon website, specifically focusing on books. Figure 2 present the screenshot of the dataset.

customer_id	product_id	star_rating	product_title	review_headline	review_body
41012519	871138344	2.0	The Bureau and the Mole: The Unmasking of Robe...	I Was Disappointed	To say that this is a brief study of the Hanss...
51999478	1576104877	4.0	RHCE Linux Exam Cram: Exam: RH-302	Good overview	The book provides a decent overview I passed t...
31771263	811817660	5.0	Joie Warner's No-Cook Pasta Sauces	No-Cook Pasta Sauces	Best book of recipes I've found for truly good...

Figure 2. A screenshot of the dataset

Also, the dataset is Demonstrated in a Tab Separated Values File (TSV) type, the dataset includes diverse attributes such as customer ID, product ID, star ratings, product title, review headlines, and review body. Analyzing this extensive dataset effectively required the application of feature-extraction and various NLP methods to reach satisfactory outcomes. The dataset itself is substantial, comprising over six hundred thousand user ratings, with a file size of approximately one gigabyte, encompassing 650,000 rows.

5.3 Results

As previously noted, the proposed study is partitioned into two deep learning frameworks: PyTorch and TensorFlow. The deep learning algorithm employed is the DNN algorithm, and the data are divided randomly, with 80% allocated for training and 20% for testing in both frameworks. It's essential to highlight that all tuning procedures such as the number of the epochs, batch size, and any other relevant parameters have been implemented consistently and fairly for both PyTorch and TensorFlow frameworks. In the PyTorch framework, two collaborative filtering recommendation system models were constructed. The initial model excluded the use of text, while the second model incorporated it by considering the review text provided by users. A parallel approach was undertaken for the TensorFlow framework, where two models were developed, one without and the other with the integration of the text. Both frameworks were examined to assess the impact of incorporating the user's textual review on recommendation accuracy. For each model within the frameworks, key metrics such as RMSE, MAE, and execution time were computed. These metrics serve to assess the performance of the RS and provide insights into its accuracy and efficiency. The subsequent sections will present the detailed results of the PyTorch and TensorFlow frameworks, along with a concise comparative analysis between them in two tables.

5.3.1 PyTorch and TensorFlow results

Two distinct DNN recommendation models were built utilizing the PyTorch and TensorFlow frameworks. One model integrated the text, while the other model refrained from using texts by employing only the traditional deep neural network algorithm. Old-style systems only rely on standard data such as user, item, and rating which may fall short in accuracy. To solve this restriction and enhance correctness, incorporating additional information, particularly text data becomes vital. Using NLP methods is imperative for the efficient analysis of huge text data. The DNN text-based models in both frameworks involved the training of collaborative filtering on attributes containing user-item-rating triplets and user text reviews.

In general, all the DNN and DNN Text-Based models in both frameworks underwent experimentation with five different types of optimizers: Stochastic Gradient Descent (SGD), Root Mean Squared Propagation (RMSprop), Adaptive Gradient Algorithm (Adagrad), Adaptive Moment Estimation (Adam), and AdaMax. Consequently, Adam was identified as the optimal parameter among all the mentioned optimizers. It yielded the best results and achieved the highest scores in both frameworks. The results from both frameworks demonstrate that the DNN text-based model surpassed the DNN model that did not incorporate texts across all the metrics employed. Fortunately, there was a reduction in both RMSE and MAE in the text-based models. There are several reasons for performance improvement in models with NLP techniques such as Data Interpretation, Extraction of meaningful features from the text, and Understanding user preferences deeply. These differences can significantly impact the practical application of recommendation systems and increase the relevancy of the overall system. Table 1 demonstrates the results of the frameworks for all the proposed models.

Table 1. Results of the frameworks for all the constructed models

Framework	Type of Methods	RMSE	MAE	Time in Min.
PyTorch	DNN	0.44	0.30	17.56
	DNN with Text-based	0.32	0.20	30.11
TensorFlow	DNN	0.68	0.58	13.17
	DNN with Text-based	0.35	0.24	21.43

5.3.2 Comparison between the frameworks

After implementing both approaches, it becomes imperative to compare them to identify the optimal model and framework. These comparisons focus on time efficiency and other metrics such as RMSE and MAE. The findings reveal that text-based models in both frameworks outperform those without text, albeit with a longer computational time for model building. Moreover, the models utilizing the PyTorch package exhibit slightly greater accuracy than their TensorFlow counterparts in both text-based and traditional models.

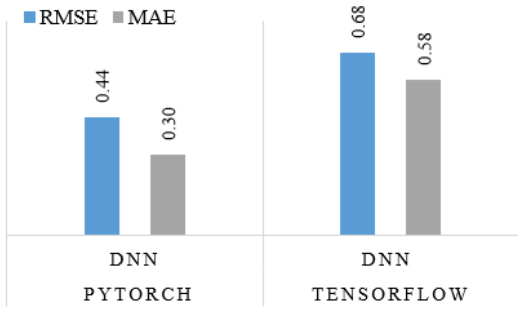

Figure 3. Frameworks comparison on DNN models

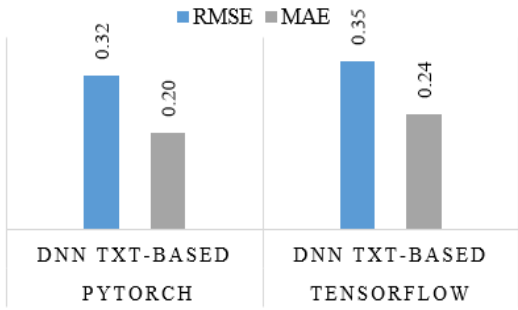

Figure 4. Framework comparison on DNN text-based

However, model building in PyTorch takes more time, consumes more GPU resources, and requires more memory than in TensorFlow. In summary, the results suggest that integrating text in recommendation systems for big data can lead to significant improvements in accuracy, effectively addressing challenges such as dynamic preferences and data sparsity associated with CF. Figure 3 compares the results of DNN models between PyTorch and TensorFlow, while Figure 4 contrasts the results of DNN text-based models in both

frameworks. On the other hand, Figure 5 illustrates the variations in execution time among the models in the frameworks.

When it comes to dealing with big data, both PyTorch and TensorFlow have strengths and considerations that may influence the choice of framework. Table 2 presents a comparison of their advantages and limitations in general. On the other hand, Table 3 demonstrates their advantages and limitations in the context of handling big data. Both PyTorch and TensorFlow can manage big data, but TensorFlow distinguishes itself with its robust ecosystem, comprehensive tools, and substantial support for distributed computing in large-scale environments. The decision to opt for PyTorch or TensorFlow in big data projects hinges on particular requirements, the team's familiarity with the frameworks, and the preferences of the development team.

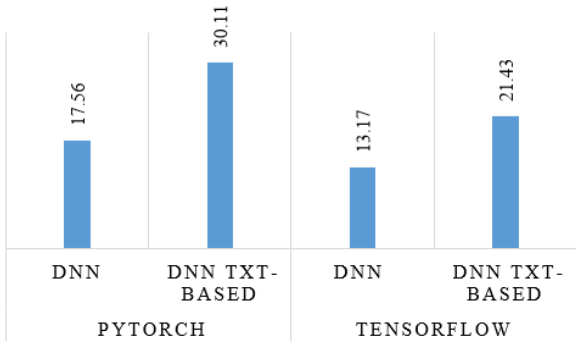

Figure 5. Frameworks execution time

Table 2. General advantages and limitations of the frameworks

Criteria	PyTorch	TensorFlow
Ease of use	Known for its dynamic computational graph, making it more intuitive and Pythonic. Easier to debug and understand	Has a steeper learning curve due to its static computational graph but is well-suited for production environments
Flexibility	Offers dynamic computational graphs, making it more flexible for dynamic models and experimentation.	Primarily uses static computational graphs, which can be less flexible for certain dynamic models
Community support	Growing community with a focus on research and experimentation. Extensive support in the research community	Larger and more established community with widespread industry adoption, providing extensive resources and libraries
Deployment	Generally considered better for research and experimentation. Deployment can be simpler with the use of the TorchScript	Strong support for deployment in production environments, especially with TensorFlow Serving and TensorFlow Lite
Visualization	Limited visualization tools, but integration with libraries such as Matplotlib is common	TensorBoard provides comprehensive visualization tools for monitoring and debugging.
Popularity	Gaining popularity, especially in the research community	Widely popular and extensively used in both research and industry
Ecosystem	A growing ecosystem with various extensions and libraries	A mature and extensive ecosystem with a wide range of tools, libraries, and community-contributed projects

Table 3. Advantages and limitations of the frameworks in the context of textual big data

Criteria	PyTorch	TensorFlow
Coding style	More complex than TensorFlow	Easy to write codes with less complexity
Text processing	Supports text processing efficiently but it has some complexity with the preprocessing	Supports text processing efficiently without any complexity in the preprocessing phase
Using optimization Algo.	It has some complexity	Less complex
Ease of use in big data	PyTorch has a relatively straightforward API, making it accessible for researchers and practitioners, but may require additional setup for distributed computing in big data	TensorFlow's comprehensive ecosystem includes tools such as TensorFlow Extended (TFX) and TensorFlow Data Validation (TFDV) for end-to-end deployment, making it a strong choice for large-scale data
Integration with big data tools	PyTorch has integration options with big data tools, but may not be as seamlessly integrated as TensorFlow with tools for data processing	TensorFlow has better integration with various big data tools and frameworks, making it well-suited for end-to-end big data pipelines and deployments
Their results in our system	Slightly better than TensorFlow	Somewhat less accurate
Their time performance in our system	Needed a slightly extended period of computational time	Required a shorter amount of time than PyTorch

6. CONCLUSION

This research introduces a recommendation system for big data, utilizing two prominent deep learning frameworks which are PyTorch and TensorFlow to enhance collaborative filtering outcomes. The primary objective is to provide users with more relevant product recommendations and examining multiple deep learning frameworks. Hence, to test the performance of the frameworks, the study built two recommendation system models within each framework. In the PyTorch framework methodology, two collaborative filtering models were created using DNN algorithms, one with incorporating user textual review data and another model without the text. Similarly, for the TensorFlow framework, two recommendation models were created, one with integrating text data and one without. The dataset, sourced from the Amazon website, included over 600,000 records. The results showed significant enhancements when applying the suggested text-based models in both PyTorch and TensorFlow frameworks, effectively addressing challenges related to dynamic preferences and data sparsity in

collaborative filtering recommendation. Additionally, a comparative analysis of the advantages and drawbacks of each framework was provided in the results section. While both PyTorch and TensorFlow can handle big data, TensorFlow often stands out due to its robust ecosystem, comprehensive tools, and extensive support for distributed computing in large-scale environments. However, TensorFlow is noted for its ease of use in implementing optimization algorithms. On the other hand, PyTorch exhibits slightly higher accuracy than TensorFlow in both proposed models. But, model construction in PyTorch requires slightly more time. Ultimately, the choice between PyTorch and TensorFlow for big data projects depends on specific requirements such as the familiarity of the development team, and their preferences. Future work will concentrate on using a bigger dataset with applying a few optimization algorithms such as Optuna, Swarm, Bee, and Bat to gain more accurate recommendation system.

REFERENCES

[1] Omar, H.K., Jumaa, A.K. (2022). Distributed big data analysis using spark parallel data processing. Bulletin of Electrical Engineering and Informatics, 11(3): 1505-1515. https://doi.org/10.11591/eei.v11i3.3187

[2] Baker, M.R., Utku, A. (2023). Unraveling user perceptions and biases: A comparative study of ML and DL models for exploring twitter sentiments towards ChatGPT. Journal of Engineering Research. https://doi.org/10.1016/j.jer.2023.11.023

[3] Asaithambi, S.P.R., Venkatraman, R., Venkatraman, S. (2023). A thematic travel recommendation system using an augmented big data analytical model. Technologies, 11(1): 28. https://doi.org/10.3390/technologies11010028

[4] Xiao, X., Wen, J., Zhou, W., Luo, F., Gao, M., Zeng, J. (2022). Multi-interaction fusion collaborative filtering for social recommendation. Expert Systems with Applications, 205: 117610. https://doi.org/10.1016/j.eswa.2022.117610

[5] Xing, S., Liu, F., Zhao, X., Li, T. (2018). Points-of-interest recommendation based on convolution matrix factorization. Applied Intelligence, 48: 2458-2469. https://doi.org/10.1007/s10489-017-1103-0

[6] Zheng, Y., Wang, D. (2022). Multi-criteria ranking: Next generation of multi-criteria recommendation framework. IEEE Access, 10: 90715-90725. https://doi.org/10.1109/ACCESS.2022.3201821

[7] Tian, Z., Pan, L., Yin, P., Wang, R. (2021). Information fusion-based deep neural attentive matrix factorization recommendation. Algorithms, 14(10): 281. https://doi.org/10.3390/a14100281

[8] Pais, S., Cordeiro, J., Jamil, M.L. (2022). NLP-based platform as a service: a brief review. Journal of big data, 9(1): 54. https://doi.org/10.1186/s40537-022-00603-5

[9] Afoudi, Y., Lazaar, M., Al Achhab, M. (2021). Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network. Simulation Modelling Practice and Theory, 113: 102375. https://doi.org/10.1016/j.simpat.2021.102375

[10] Mu, Y., Wu, Y. (2023). Multimodal movie recommendation system using deep learning. Mathematics, 11(4): 895. https://doi.org/10.3390/math11040895

[11] Aljunid, M.F., Huchaiah, M.D. (2021). An efficient hybrid recommendation model based on collaborative filtering recommender systems. CAAI Transactions on Intelligence Technology, 6(4): 480-492. https://doi.org/10.1049/cit2.12048

[12] Mandal, S., Maiti, A. (2020). Explicit feedback meet with implicit feedback in GPMF: A generalized probabilistic matrix factorization model for recommendation. Applied Intelligence, 50(6): 1955-1978. https://doi.org/10.1007/s10489-020-01643-1

[13] Batmaz, Z., Yurekli, A., Bilge, A., Kaleli, C. (2019). A review on deep learning for recommender systems: challenges and remedies. Artificial Intelligence Review, 52: 1-37. https://doi.org/10.1007/s10462-018-9654-y

[14] Zhang, W., Liu, F., Xu, D., Jiang, L. (2019). Recommendation system in social networks with topical attention and probabilistic matrix factorization. PloS One, 14(10): e0223967. https://doi.org/10.1371/journal.pone.0223967

[15] Ma, M., Cao, Q., Liu, X. (2022). A graph convolution collaborative filtering integrating social relations recommendation method. Applied Sciences, 12(22): 11653. https://doi.org/10.3390/app122211653

[16] Maneechote, N., Maneeroj, S. (2022). Explainable recommendation via personalized features on dynamic preference interactions. IEEE Access, 10: 116326-116343. https://doi.org/10.1109/ACCESS.2022.3219076

[17] Wang, R., Wu, Z., Lou, J., Jiang, Y. (2022). Attention-based dynamic user modeling and deep collaborative filtering recommendation. Expert Systems with Applications, 188: 116036. https://doi.org/10.1016/j.eswa.2021.116036

[18] Zhang, S., Yao, L., Sun, A., Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR), 52(1): 5. https://doi.org/10.1145/3285029

[19] Li, Q., Li, S., Zhang, S., Hu, J., Hu, J. (2019). A review of text corpus-based tourism big data mining. Applied Sciences, 9(16): 3300. https://doi.org/10.3390/app9163300

[20] Choi, S.M., Lee, D., Jang, K., Park, C., Lee, S. (2023). Improving data sparsity in recommender systems using matrix regeneration with item features. Mathematics, 11(2): 292. https://doi.org/10.3390/math11020292

[21] Peng, M., Xie, Q., Wang, H., Zhang, Y., Tian, G. (2018). Bayesian sparse topical coding. IEEE Transactions on Knowledge and Data Engineering, 31(6): 1080-1093. https://doi.org/10.1109/TKDE.2018.2847707

[22] Omar, H.K., Frikha, M., Jumaa, A. K. (2024). Improving big data recommendation system performance using NLP techniques with multi attributes. Informatica, 48(5): 63-70. https://doi.org/10.31449/inf.v48i5.5255

[23] Novac, O.C., Chirodea, M.C., Novac, C.M., Bizon, N., Oproescu, M., Stan, O.P., Gordan, C.E. (2022). Analysis of the application efficiency of TensorFlow and PyTorch in convolutional neural network. Sensors, 22(22): 8872. https://doi.org/10.3390/s22228872

[24] Omar, H.K., Frikha, M., Jumaa, A.K. (2023). big data cloud-based recommendation system using NLP techniques with machine and deep learning. TELKOMNIKA (Telecommunication Computing Electronics and Control), 21(5): 1076-1083. http://doi.org/10.12928/telkomnika.v21i5.24889

[25] Liang, D., Corneli, M., Bouveyron, C., Latouche, P. (2021). DeepLTRS: A deep latent recommender system based on user ratings and reviews. Pattern Recognition Letters, 152: 267-274. https://doi.org/10.1016/j.patrec.2021.10.022

[26] Mahon, S., Varrette, S., Plugaru, V., Pinel, F., Bouvry, P. (2020). Performance analysis of distributed and scalable deep learning. In 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Melbourne, VIC, Australia, pp. 760-766. https://doi.org/10.1109/CCGrid49817.2020.00-13

[27] Nassar, N., Jafar, A., Rahhal, Y. (2020). Multi-criteria collaborative filtering recommender by fusing deep neural network and matrix factorization. Journal of big data, 7: 34. https://doi.org/10.1186/s40537-020-00309-6

[28] Chirodea, M.C., Novac, O.C., Novac, C.M., Bizon, N., Oproescu, M., Gordan, C.E. (2021). Comparison of tensorflow and pytorch in convolutional neural network-based applications. In 2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, pp. 1-6. https://doi.org/10.1109/ECAI52376.2021.9515098

[29] Osman, N.A., Mohd Noah, S.A., Darwich, M., Mohd, M. (2021). Integrating contextual sentiment analysis in collaborative recommender systems. Plos One, 16(3): e0248695. https://doi.org/10.1371/journal.pone.0248695

[30] Sharma, A.K., Bajpai, B., Adhvaryu, R., Pankajkumar, S.D., Gordhanbhai, P.P., Kumar, A. (2023). An efficient approach of product recommendation system using NLP technique. Materials Today: Proceedings, 80: 3730-3743. https://doi.org/10.1016/j.matpr.2021.07.371

[31] Abbasi-Moud, Z., Vahdat-Nejad, H., Sadri, J. (2021). Tourism recommendation system based on semantic clustering and sentiment analysis. Expert Systems with Applications, 167: 114324. https://doi.org/10.1016/j.eswa.2020.114324

[32] Paszke, A., Gross, S., Massa, F., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems, 32.

[33] PyTorch. (2024). https://pytorch.org/.

[34] Ravanelli, M., Parcollet, T., Bengio, Y. (2019). The pytorch-kaldi speech recognition toolkit. In ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, pp. 6465-6469. https://doi.org/10.1109/ICASSP.2019.8683713

[35] Gao, X., Ramezanghorbani, F., Isayev, O., Smith, J. S., Roitberg, A.E. (2020). TorchANI: A free and open source PyTorch-based deep learning implementation of the ANI neural network potentials. Journal of Chemical Information and Modeling, 60(7): 3408-3415.

[36] Pang, B., Nijkamp, E., Wu, Y.N. (2020). Deep learning with tensorflow: A review. Journal of Educational and Behavioral Statistics, 45(2): 227-248. https://doi.org/10.3102/1076998619872761

[37] Grattarola, D., Alippi, C. (2021). Graph neural networks in tensorflow and keras with spektral [application notes]. IEEE Computational Intelligence Magazine, 16(1): 99-106. https://doi.org/10.1109/MCI.2020.3039072

[38] Chockwanich, N., Visoottiviseth, V. (2019). Intrusion detection by deep learning with tensorflow. In 2019 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang, Korea (South), pp. 654-659. https://doi.org/10.23919/ICACT.2019.8701969

[39] Li, Y., Zhao, Z., Luo, Y., Qiu, Z. (2020). Real-time pattern-recognition of GPR images with YOLO v3 implemented by tensorflow. Sensors, 20(22): 6476. https://doi.org/10.3390/s20226476

[40] Gunawan, T.S., Ashraf, A., Riza, B.S., Haryanto, E.V., Rosnelly, R., Kartiwi, M., Janin, Z. (2020). Development of video-based emotion recognition using deep learning with Google Colab. TELKOMNIKA (Telecommunication Computing Electronics and Control), 18(5): 2463-2471. http://doi.org/10.12928/telkomnika.v18i5.16717

[41] Colab research. Google. https://colab.research.google.com/signup/pricing?utm_source=dialog&utm_medium=link&utm_campaign=settings_page.