Decentralized P2P databases for collaborative learning in AI applications

Quinten van Eijs Delft University of Technology The Netherlands Johan Pouwelse Delft University of Technology The Netherlands

***Abstract*—**

***Index Terms*—Decentralized learning, P2P, Information Retrieval.**

I. Introduction

Recommender systems leverage the power of Artificial Intelligence (AI) to predict user preferences and suggest relevant items. This technology plays a crucial role in various online platforms, from suggesting video's on YouTube to recommending personalized content on TikTok. By analyzing user data like past purchases, browsing history, and engagement metrics, AI algorithms within recommender systems create a personalized profile for each user. This profile then fuels content recommendations, aiming to surface items that are most likely to capture user interest and keep them engaged.

The current generation of AI is fully centralised. Only huge Big Tech companies can afford the enormous cost of datacenters filled with scarse AI training hardware. We prove the viability of a radically different AI model. We show that machine learning without any point of control is possibly. One of the Internet's defining features is its lack of any single point of technical, political, or economic control. We present experimental results of a machine learning architecture without any central control. [4]

Decentralized learning paradigms are emerging as an alternative due to their superior privacy, security, and scalability. However, existing decentralized learning methods are still immature. Federated learning is still struggling to provide the performance offered by centralised LLM-based approaches . Yet, federated learning has some of the same drawbacks of centralised learning. Federated learning is still based on central control. The defining feature of The Internet of fully decentralised control can *not* be achieve with the federated learning model[3].

1) **Enabling collaborative learning:** Developing robust communication protocols and efficient algorithms for nodes to share data and model updates, fostering the collective refinement of knowledge and leading to better, more consistent models across the network.
2) **Facilitating dynamic embedding learning:** Implementing mechanisms for nodes to dynamically adjust vector representations based on their local data and interactions, resulting in contextually relevant and personalized models that adapt to the evolving data landscape.
3) **Ensuring privacy-preserving learning:** Exploring privacy-preserving techniques like federated learning and homomorphic encryption to allow nodes to contribute to the learning process without directly sharing sensitive data, thereby addressing privacy concerns inherent in decentralized learning.
4) **Robustness to node failure:** PeerAI gracefully handles the failure of neighboring nodes.

By addressing these challenges we aim to unlock the true potential of P2P vector databases for decentralized learning, paving the way for a more secure, scalable, and privacy-conscious future of AI and ML applications.

II. Problem Description

While content retrieval systems on platforms like Youtube and TikTok offer personalized recommendations and a seemingly endless stream of content, they are not without limitations. These limitations stem from their dependence on centralized control and infrastructure. Some of the key challenges include:

Privacy Concerns: Centralized storage of user data raises privacy issues. User behavior, search history, and watch time are all collected and analyzed to personalize recommendations. This data can be vulnerable to breaches and may be used in unforeseen ways, leading to a lack of transparency and control for users.

Bias and Algorithmic Injustice: Algorithms that power these systems can be biased based on the data they are trained on and the goals of the platform. This can lead to the filtering out of diverse viewpoints and the promotion of content that reinforces existing biases. Users may end up trapped in "echo chambers" where they are only exposed to content that confirms their existing beliefs.

Scalability and Performance: Centralized systems face challenges in scaling to accommodate the growing volume of data and users. As the number of users and content items increases, the system may struggle to provide timely and relevant recommendations. This can result in slower response times and decreased user satisfaction.

Limited Control for Users: Users have little control over how their data is used or how content is recommended. The algorithms operate as a black box, making it difficult for users to understand why specific content is being surfaced. This lack of control can be frustrating for users who may not appreciate the recommendations they receive.

These limitations highlight the need for alternative approaches to content retrieval systems. Decentralized models that prioritize user privacy and data control are emerging as potential solutions for a more secure and equitable future of online content consumption.

PeerAI is a decentralized learning paradigm that leverages peer-to-peer (P2P) vector databases and dynamic embeddings to address the challenges of centralized content retrieval systems. By enabling collaborative learning, facilitating dynamic embedding updates, and ensuring privacy-preserving mechanisms, PeerAI aims to provide a more secure, scalable, and privacy-conscious alternative to traditional content retrieval systems.

III. Background and Related Works

IV. Design of PeerAI

We now present the design of PeerAI and discuss the system architecture, data partitioning, communication protocols, and consistency management mechanisms. We also describe the dynamic embedding learning algorithm and its role in collaborative learning within the P2P network.

A. System Model and Assumptions

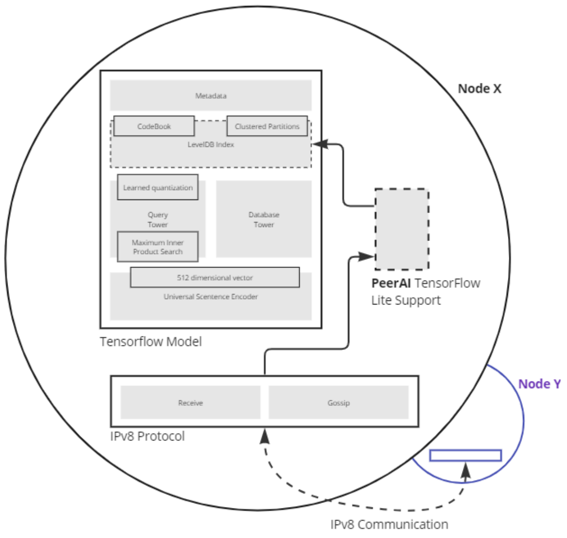

Fig. 1. The PeerAI system model. The system consists of three components: the P2P network, the model, and the collaborative learning algorithm. The P2P network is responsible for sharing the data, the model is responsible for encoding the data into a vector space, and the collaborative learning algorithm is responsible for updating the embedding model based on the data received from the P2P network.

The System model consists of three differente components. The first component is the P2P network, which is responsible for sharing the data. The second component is the model, which is responsible for encoding the data into a vector space. The third component is the collaborative learning algorithm, which is responsible for updating the embedding model based on the data received from the P2P network.

B. P2P IPV8 network

The IPV8 network is a decentralized peer-to-peer network that enables secure communication between nodes.

C. Tflite model

1) Universal Scente Encoder: The Universal Sentence Encoder (USE) transforms text into high-dimensional vectors suitable for various natural language processing tasks like text classification, semantic similarity measurement, clustering, and more. Trained and optimized for processing text longer than individual words, such as sentences, phrases, or short paragraphs, the USE is versatile across a spectrum of natural language understanding tasks. Its training corpus encompasses diverse data sources and tasks, facilitating adaptability to various text-related tasks. The model accepts variable-length English text as input and produces a fixed-length 512-dimensional vector as output. In practice, it demonstrates effectiveness in tasks like semantic similarity evaluation, exemplified through its application in the STS benchmark, with detailed results available in an accompanying notebook. Built upon a Deep Averaging Network (DAN) encoder architecture, the USE stands apart from traditional word-level embedding models by training on tasks that necessitate understanding the meaning of word sequences rather than individual words. For further exploration of text embeddings, refer to the TensorFlow Embeddings documentation.[7]

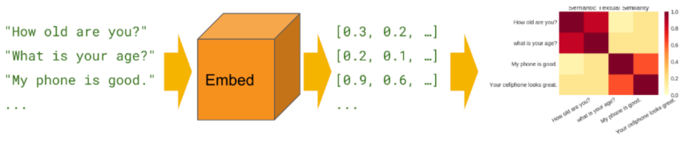

Fig. 2. Universal scentence embedder

2) SCaNN: The SCaNN algorithm is a scalable and efficient algorithm for similarity search in high-dimensional vector spaces. It is based on Product Quantization (PQ) and Asymmetric Hashing, which compresses the database embeddings into a compact form for fast retrieval. SCaNN is designed to work with large-scale datasets and is optimized for both CPU and GPU architectures. It is capable of handling millions of embeddings and can perform similarity search in milliseconds. SCaNN is particularly well-suited for applications that require fast and accurate retrieval of similar items, such as recommendation systems, search engines, and content-based filtering.[2]

D. *PeerAI TFLITE SUPPORT*

Dynamic embeddings are vector representations that can adapt to changes in the underlying data distribution. By updating the embeddings based on the local data and interactions of each node, dynamic embeddings can capture context-specific information and improve the performance of machine learning models. In the context of PeerAI, dynamic embeddings enable nodes to personalize their models and contribute to the collaborative learning process. The collaborative learning algorithm leverages the dynamic embeddings to refine the global model and ensure consistency across the network.
Everything should be accissible from the Tribler SuperApp Kotlin implementation.

V. Implementation and Experimental Evaluation

We now describe our implementation and present the experimental evaluation of PeerAI. We first discuss the system architecture, data partitioning, and communication protocols. We then present the dynamic embedding learning algorithm and evaluate its performance. Finally, we discuss the results of our experiments and compare them with existing approaches.

A. *Implementation Details*

1) PeerAI TFLite Support: We implement the PeerAI system using the TensorFlow Lite framework, which enables the deployment of machine learning models on mobile and IoT devices. TensorFlow Lite provides a lightweight solution for running machine learning models on edge devices with limited computational resources. We leverage the TensorFlow Lite framework to interact with the pre-trained model that transforms text into high-dimensional vectors suitable for various natural language processing tasks. The framework is written in C++ and uses Bazel [1] to build cross-Platform and is supported on Java, C++ (WIP), and Swift (WIP).

2) Tensorflow Lite Model: Dataset used for training for the pre-trained model.

Dataset	Model Params	Entries	Size
PandaCD [5]	$\sigma = 30$, $\omega = 6$	700	500KB
Spotify and Youtube [6]	$\sigma = 140$, $\omega = 6$	20.000	4MB
NPI* Spotify and Youtube [6]	$\sigma = 140$, $\omega = 6$	2.600.000	37MB
YouTube-Commons [8]	$\sigma = 1450$, $\omega = 3$	2.000.000	400MB?

TABLE I
Overview of different pre-trained models all using the same embedding model. MPI non perfect inserts are used for the NPI* Spotify and Youtube dataset which consists of an inbalanced tree and thus a exploded partition.

Metadata is not bound to the model, but it is to the SuperApp kotlin implementation therefore the metadata is stored as parsed JSON including a title, author and a youtube video id.

We utilize the following parameters for all models:

- $distance\_measure$
- $dimensions\_per\_block$
- $anisotropic\_quantization\_threshold$

In ScaNN PQ is used to compress the database embeddings, but not the query embedding. Which is called it Asymmetric Hashing.

dimensions_per_block How many dimensions in each PQ block. If the embedding vector dimensionality is a multiple of this value, there will be. $\frac{number\_of\_dimensions}{dimensions\_per\_block}$ PQ blocks.
Otherwise, the last block will be the remainder. For example, if a vector has 12 dimensions, and **dimensions_per_block** is 2, then there will be 6 2-dimension blocks. However, if the vector has 13 dimensions and **dimensions_per_block is still** 2, there will be 6 2-dimension blocks and one 1-dimension block.

If this value is set, we will penalize the quantization error that's parallel to the original vector differently than the orthogonal error. A generally recommended value for this parameter is 0.2.

When measuring distance, we opt for the **dot_product** method to assess the disparity between two embedding vectors. It's essential to highlight that we derive the negative dot product value to adhere to the principle that smaller is closer. Regarding the dataset structure, it undergoes segmentation into σ partitions, roughly corresponding to the square root of the dataset size. During retrieval, the search encompasses σ of these partitions, constituting approximately between 0.15% and 3% of the dataset. For space optimization, we implement quantization for floating-point embeddings, converting them into int8 values of the same dimension.

B. *Experimental Evaluation*

C. *Content Retrieval Performance*

In app screenshots.

D. *Non perfect insert experiment breeaking the partition*

We evaluate the performance of the PeerAI system using the Spotify and Youtube dataset. We compare the performance of the system with and without the use of non-perfect inserts. The non-perfect inserts are used to simulate an imbalanced tree structure in the dataset, which can lead to an exploded partition. We measure the retrieval accuracy and efficiency of the system under different conditions and analyze the impact of non-perfect inserts on the search performance.

E. *Non perfect insert versus pretrained model*

We evaluate the performance of the PeerAI system using the Spotify and Youtube dataset. We compare the performance of the system with and without the use of non-perfect inserts. The non-perfect inserts are used to simulate an imbalanced tree structure in the dataset, which can lead to an exploded partition. We measure the retrieval accuracy and efficiency of the system under different conditions and analyze the impact of non-perfect inserts on the search performance.

VI. Evaluation and Experiments

- Design and conduct experiments comparing the proposed system with existing approaches.
- Evaluate search accuracy, efficiency, and personalization based on real-world datasets.
- Analyze the impact of dynamic embeddings and collaborative learning on search performance.

A. *Vector Space*

Indexing Youtube and TikTok: Explore the feasibility of indexing content from these platforms using P2P vector databases. Consider potential challenges related to copyright, privacy, and data access limitations. Recommending 20k Closely Related Items: Design and evaluate algorithms that recommend highly relevant content based on a user's personalized model and collaborative learning within the P2P network. Analyze the accuracy and diversity of recommendations compared to traditional approaches.

- Pretrained 20k item model including 140 partitions and 128 embedded asymmetric hashes
- Inference Speed
- Impact of different embedding methods and inpact on clusters. Such as sentence encoder vs weighted song features.
- cluster visualization

B. *Investigate the use of various distance metrics and similarity search algorithms for enhanced retrieval accuracy.*

C. *Benchmark against traditional or static vector databases database?*

D. *(CPU/GPU) Performance*

Personalized Model on Smartphones: Investigate the implementation of personalized models on individual smartphones using local data and collaborative learning within a P2P network. Evaluate the impact of limited resources and privacy constraints.

- Gossip 1 item per second with multiple connected peers.
- Loss function calculating new centroids given k-mean and codebook update of hashes. Non-Perfect update show practical implementation but does not converge the loss function since partitions are expanded.

VII. Conclusion and Future Work

- Summarize the key findings and contributions.
- Discuss limitations and potential future research directions.
- Highlight the broader implications of P2P vector databases and dynamic embeddings for personalized content retrieval and other applications.

Current setup has no block/filter on given other nodes such that all data and given meta data is learned by peers even unwanted metadata. This is out of scope for the given research.

References

[1] *Bazel*. URL: https://bazel.build/.

[2] Ruiqi Guo et al. "Accelerating Large-Scale Inference with Anisotropic Vector Quantization". In: *International Conference on Machine Learning*. 2020. URL: https://arxiv.org/abs/1908.10396.

[3] Enrique Tomás Martínez Beltrán et al. "Decentralized Federated Learning: Fundamentals, State of the Art, Frameworks, Trends, and Challenges". In: *IEEE Communications Surveys & Tutorials* 25.4 (2023). ISSN: 2373-745X. DOI: 10.1109/comst.2023.3315746. URL: http://dx.doi.org/10.1109/COMST.2023.3315746.

[4] Mark Nottingham. *RFC 9518: Centralization, Decentralization, and internet standards*. URL: https://datatracker.ietf.org/doc/rfc9518/.

[5] *pandacd-dataset*. URL: https://pandacd.io/.

[6] *spotify-youtube-dataset*. URL: https://www.kaggle.com/datasets/salvatorerastelli/spotify-and-youtube.

[7] *tensorflow*. URL: https://tensorflow.org/.

[8] *youtube-commons-dataset*. URL: https://huggingface.co/datasets/PleIAs/YouTube-Commons.