Evaluating Recommendation Systems Using the Power of Embeddings

Jonathan Gomes Selman[1], Robert Friel[2], Bogdan Gheorghe[3], Derek Austin[4] and Atindriyo Sanyal[5,*,†]

[1]Galileo Technologies Inc., 525 Brannan St, Suite 410, San Francisco CA 94107

Abstract

This paper addresses challenges faced by Recommendation Systems by introducing novel techniques that utilize embeddings and data centric algorithms. It highlights the limitations of current evaluation metrics and proposes an ideal embeddings management system with automatic ingestion and generation, serving real-time and batch embeddings through separate datastores. The paper also introduces three new metrics: semantic coverage, out-of-distribution detection, and representativeness, which provide a more comprehensive evaluation of recommendation systems, aiming to enhance user satisfaction and engagement.

Keywords

Embedding Stores, Semantic Drift, Coverage, Representativeness

1. Introduction

Recommendation Systems have become an integral part of most software applications, offering enhanced user experiences and improved engagement. However, these systems still face challenges such as cold start issues and long tail problems. This paper aims to address these challenges by introducing novel techniques that leverage embeddings, along with advanced algorithms, to evaluate and enhance Recommendation Systems.

Initially, we highlight the limitations of current standardized metrics used for evaluation, including MAP@k, MAR@k, Coverage, and Personalization. While these metrics have been widely employed, they fail to capture the complete picture of system performance.

Next, we delve into the components of an ideal embeddings management system. We propose an approach that involves automatic ingestion and generation of embeddings, combined with a

EvalRS'23: a rounded evaluation of recommender systems, August 07, 2023, Long Beach, CA

*Corresponding author.

†These authors contributed equally.

jonathan@rungalileo.io (J. G. Selman); rob@rungalileo.io (R. Friel); bogdan@rungalileo.io (B. Gheorghe); derek@rungalileo.io (D. Austin); atin@rungalileo.io (A. Sanyal)

https://www.linkedin.com/in/jonathan-gomes-selman-5144b0112/ (J. G. Selman); https://www.linkedin.com/in/robertfriel/ (R. Friel); https://www.linkedin.com/in/bogdangheorghe/ (B. Gheorghe); https://www.linkedin.com/in/derekaustin22/ (D. Austin); https://www.linkedin.com/in/atinsanyal/ (A. Sanyal)

0000-0002-0877-7063 (J. G. Selman); 0000-0002-0877-7063 (R. Friel); 0000-0002-0877-7063 (B. Gheorghe); 0000-0002-0877-7063 (D. Austin); 0000-0002-0877-7063 (A. Sanyal)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

dual datastore system. This system efficiently serves real-time embeddings through an online datastore, while batch embeddings are handled by an offline datastore.

Furthermore, we discuss three metrics that leverage embeddings to provide a more comprehensive evaluation of recommendation systems. These metrics include semantic coverage, which assesses the system's ability to recommend diverse items with similar meaning; out-of-distribution detection, which identifies recommendations outside the system's usual scope; and representativeness, which measures the system's ability to capture the users' preferences accurately.

By incorporating these innovative techniques, we aim to enhance the evaluation process of Recommendation Systems, enabling a more accurate assessment of their performance and ultimately improving user satisfaction and engagement.

2. Evaluation Metrics for Recommendation Systems

Recommendation Systems today rely on standard metrics to assert the effectiveness of the recommendations such as MAP@k, MAR@k, Coverage and Personalization Measures. But the limitations of these metrics are the same as that of evaluating model performance on metrics like P/R curves and F1 Scores. The main limitation is that these metrics are idempotent (in context of a single (or few) models' training processes) and rely heavily on the sanity of the training and validation sets, which are often laden with errors, noise and imbalance.

3. Components of an Embeddings Management System

Embeddings are learned representations of data, typically seen in Text and Computer Vision systems where each piece of text or an image has an associated embedding. The power of embeddings comes from the fact that they capture hidden features of the item in question (text or image), and allows a user to treat them like representational vectors i.e. do vector math on these and derive second order meaning in datasets.

In a typical scenario, embeddings are derived from a Neural Network that processes the training data. These embeddings are generated by pre-featurizing the data before it is fed into the model. However, in the context of recommendation systems, embeddings often originate from external processes as third-party entities. These embeddings, known as Entity Embeddings, play a crucial role in recommendation systems as they represent the ”entities” involved in recommendations, such as users and items.

These Entity Embeddings hold great importance in recommendation systems, as they enable the system to understand the relationships and interactions between different entities. By capturing the essence of these entities, the embeddings facilitate accurate and effective recommendations.

Therefore, Entity Embeddings get treated as bespoke features in recommendation systems as they are generated outside of the RecSys model, and incorporated directly into models at training and inference runtime. These embeddings bridge the gap between entities, allowing the recommendation system to make informed and relevant suggestions to users based on their preferences and behaviors. [1]

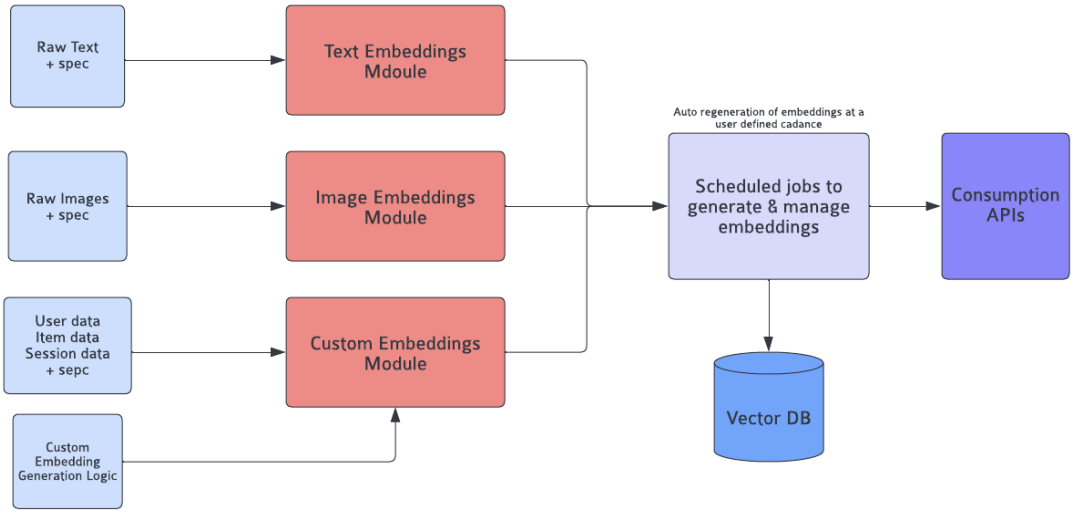

Figure 1: Embeddings Management System

Below we outline the 3 major components of a robust evaluation framework for Recommendation Systems:

3.1. Auto ingestion and Embedding Generation

The distinction between text and image embeddings and embeddings generated in Recommendation Systems is the latter embeddings evolve over time (sometimes in a matter of minutes). For example, say Airbnb creates ”component embeddings” from user interactions of different components on their application (homes, rentals, experiences etc.), embeddings for change and evolve as new users interact with those components. Apart from that, embeddings can also be extracted from raw text present in their systems (descriptions of listings) etc. Hence the first key component of an embedding management system is to allow for easy auto-ingestion and generation of embeddings from different sources, as outlined in Figure 1.

3.2. A dual data store system for real-time and offline consumption

Embeddings are consumed either by offline systems e.g. training models as well as various real-time systems. This is because a robust recommendation system need not rely only on the output of a machine learning model, but can also leverage embeddings for online rules. [2]

For example, to solve the cold start problem in Recommendation Systems, the system can leverage static embeddings from raw text and images to make personalize recommendations for a new user, a new listing, a new restaurant etc. Here’s the outline of such a dual datastore system for embedding ecosystems:

Here, the online vector DB provides low latency access to real-time embeddings generated minutes (potentially seconds) ago. Such embeddings can be used to capture real-time changes in user’s interactions or any entity whose embeddings are a function of real-time changes. The

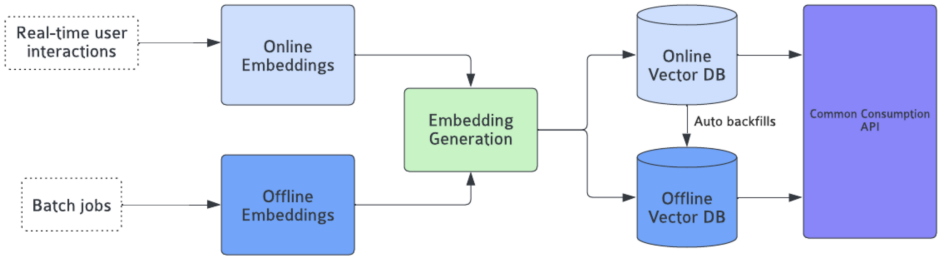

Figure 2: Managing Real-time and Offline Generated Embeddings in a single system

offline store generates embeddings from non-realtime information and stores them in an offline vector database. The distinction between the different types of embeddings should be abstracted from the user and made available via a common consumption API.

3.3. Framework for RecSys Evaluation using Embeddings

3.3.1. Semantic User<>Item Drift

This section dives deeper into leveraging drift to capture new user/item interactions. In machine learning we generally view data drift as data - e.g. production data - differing from the data used to train a model - i.e. coming from a different underlying distribution. There are many factors that can lead to dataset drift and several ways that drift can manifest. Broadly there are two main categories of data drift: 1) virtual drift (co-variate shift) and 2) concept drift.

Detecting drift using embeddings: Using an embedding based, non-parametric nearest neighbor algorithm one can easily detect out of distribution (OOD) data - i.e. drifted and out of coverage data. Differentiating algorithm characteristics include:

A. Embedding Based: Leverage hierarchical, semantic structure encoded in neural network embeddings - particularly realized through working with (large) pre-trained models, e.g. large language models (LLMs)

B. Non Parametric: does not impose any distributional assumptions on the underlying embedding space, providing simplicity, flexibility, and generality

C. Interpretability: the general simplicity of nearest neighbor based algorithms provides easy interpretability

In the context of Recommendation systems, embeddings that are drifted from a provided baseline allude to new users with new shopping patterns not leading to a purchase, or new items that are not getting enough coverage. The diagram below illustrates clusters of user and item embeddings overlaid on top of each other, suggesting regular traffic patterns (Figure 3 - Left), and drifted traffic patterns (Figure 3 - Right)

Figure 3 (Right) leverages embeddings based drift detection to surface users and items (represented by the embedding points) far apart in space from other closely correlated clusters. These embedding points represent new users and items, that will likely suffer from the Cold Start problem. Some corrective actions a practitioner may take includes adding this new data to

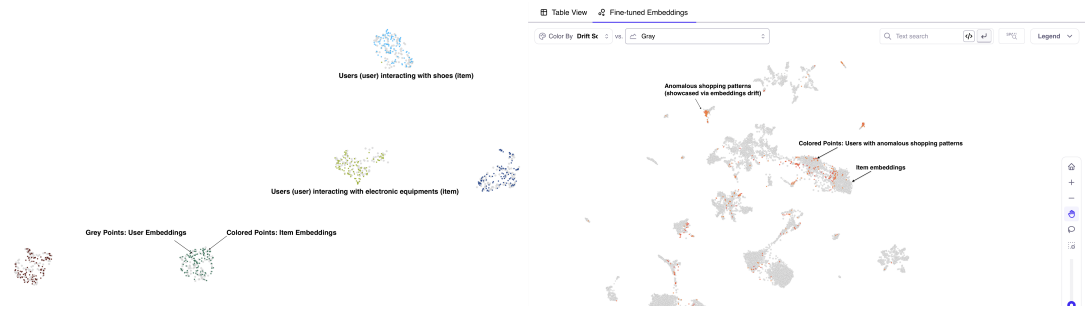

Figure 3: Overlay of User and Item Embeddings for capturing representation (Left) and Drifted shopping patterns detected using Embeddings Drift (Right)

their training and evaluation datasets and using embedding distance measurements to find the ”nearest dense clusters” i.e. representing likely users for new items, and likely items for new users.

3.3.2. User<>Item - semantic coverage

A key metric to capture personalization and solve the long tail problem in Recommendation Systems is to monitor coverage. Similar to drift, coverage can be easily measured using baseline embeddings. In coverage, the idea is to overlay user embeddings with the ones for items, and measure the extent of the overlay - which we term as coverage [3]. There's an underlying assumption here that the user and item embeddings have a shared latent space. But this method is a great improvement over measuring the classic ”coverage” metric, which purely relies on the output of the machine learning model (i.e. whether an item from training was recommended in the test set or not).

As shown in the figure below, colored embedding points refer to items that user's (grey dots) typically wouldn't interact with, and hence will likely not show up in other users' recommendation.

3.3.3. Out-of-distribution User<>Item Interactions

There's 2 forms of outliers – anomalies, and global outliers. In the context of Recommendation systems, anomalies refer to slightly drifted behavior (e.g. restaurants with a new fusion cuisine , a new style of jacket in the marketplace etc.). Often such items might still have coverage, but they are different enough to make the models perform poorly, and hence necessitate adding new training data.

Global outliers on the other hand are drastically new (and sparse) data points (either users or items) that are likely one-offs and can be safely ignored or removed from the training/evaluation data without affecting the overall performance of the recommendation system.

3.3.4. Representativeness in Data Selection

Achieving an optimal level of personalization can be efficiently accomplished by juxtaposing two sets of embeddings, one for users and another for items, and applying a representational algorithm on top. The objective is to curate an ideal collection of user-item pairs to be included in both the training and evaluation datasets.

By leveraging embeddings, we gain a potent and sophisticated approach to selecting user-item pairs for integration into our machine learning workflows. This methodology aids in making informed decisions regarding which data should be included or excluded. Consequently, it helps mitigate data blind spots that can lead to anomalous recommendations.

This embedding-based selection process allows for a more comprehensive and precise understanding of user-item relationships, enhancing the accuracy and relevance of recommendations. By effectively managing the inclusion of user-item pairs in the training and evaluation datasets, we can optimize the personalization level and minimize the occurrence of suboptimal recommendations.

4. Conclusion

While Recommendation Systems have become essential for enhancing user experiences and engagement in software systems, they still face challenges such as cold start issues and long tail problems. This paper has addressed these challenges and augmented existing methodologies of evaluating RecSys models by introducing innovative techniques that utilize embeddings and advanced algorithms to assess the robustness of recommendation systems. We have highlighted the limitations of standard evaluation metrics and proposed a comprehensive approach for embedding management, including automatic ingestion and generation, along with a dual datastore system for real-time and batch embeddings. Moreover, we have presented three novel metrics—semantic coverage, out of distribution detection, and representativeness—that leverage embeddings to provide a more comprehensive evaluation of recommendation systems. By adopting these techniques and metrics, we can advance the effectiveness and efficiency of Recommendation Systems, ensuring improved recommendations and user satisfaction in diverse application domains.

5. Acknowledgments

Thanks to the broader Galileo engineering team for their extensive work on deriving data insights from Embeddings and building them into real-life practical systems.

References

[1] P. J. Chia, J. Tagliabue, F. Bianchi, C. He, B. Ko, Beyond ndcg: Behavioral testing of recommender systems with reclist, WWW '22 Companion, Association for Computing Machinery, New York, NY, USA, 2022, p. 99–104. URL: https://doi.org/10.1145/3487553.3524215. doi:10.1145/3487553.3524215.

[2] L. Orr, A. Sanyal, X. Ling, K. Goel, M. Leszczynski, Managing ml pipelines: feature stores and the coming wave of embedding ecosystems, arXiv preprint arXiv:2108.05053 (2021).
[3] A. Sanyal, V. Chatterji, N. Vyas, B. Epstein, N. Demir, A. Corletti, Fix your models by fixing your datasets, arXiv preprint arXiv:2112.07844 (2021).