HEMR: Hypergraph Embeddings for Music Recommendation

Antonino Ferraro[1], Antonio Galli[1], Valerio La Gatta[1], Vincenzo Moscato[1], Marco Postiglione[1], Giancarlo Sperlì[1] and Flora Amato[1]

[1]*Department of Electrical Engineering and Information Technology, University of Naples Federico II, Italy*

Abstract

With the increasing number of multimedia streaming platforms, it has become essential to provide advanced recommendation systems to support users in browsing the vast amount of multimedia data according to their preferences and needs. The key challenge is to model entities and their complex relationships, such as users' listening patterns, song features, and artists' releases. This paper represents an extended abstract of a recent work describing of a novel framework, namely Hypergraph Embeddings for Music Recommendation (HEMR), which leverages hypergraph data structures along with modern graph machine learning techniques for song recommendation. The hypergraph data model is used to represent complex interactions between users and songs, while embedding techniques help to infer user-song similarities via a vector mapping. Our experiments demonstrate HEMR's effectiveness and efficiency compared to state-of-the-art music recommender systems, especially in cold start problem scenarios. Therefore, our system is a promising solution to embed within a music streaming platform to enhance users' satisfaction.

Keywords

Hypergraph, Graph Embedding, Recommender Systems, Information Filtering.

1. Introduction

During the last decade, recommendation systems have become increasingly pervasive across various industries, spanning multimedia streaming platforms [1], and personal career matching [2]. The Big Data era has created new opportunities and challenges for recommender systems, allowing them to exploit a wider range of information to improve their performance without user interaction. For instance, a recommender system can suggest songs based on the user's listening history, and propose new artists that make similar music to the ones already listened to.

State-of-the-art music recommenders (e.g. [3, 4, 5]) retrieve features from raw audio data, lyrics, user information and exploit the behavioural patterns of user communities. The challenge is to maximize the value of all available information, particularly with new users. The use

SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy

antonino.ferraro@unina.it (A. Ferraro); antonio.galli@unina.it (A. Galli); valerio.lagatta@unina.it (V. L. Gatta); vincenzo.moscato@unina.it (V. Moscato); marco.postiglione@unina.it (M. Postiglione); giancarlo.sperli@unina.it (G. Sperlì); flora.amato@unina.it (F. Amato)

0000-0002-1326-0325 (A. Ferraro); 0000-0001-9911-1517 (A. Galli); 0000-0002-0877-7063 (V. L. Gatta); 0000-0002-0877-7063 (V. Moscato); 0000-0002-0877-7063 (M. Postiglione); 0000-0003-4033-3777 (G. Sperlì); 0000-0002-5128-5558 (F. Amato)

of graph and hyper-graph structures in data analytics have been widely studied due to their capability to represent complex and diverse information [6, 7, 8, 9, 10]. Graphs are a natural data structure to represent the heterogeneous information of users, songs, artists and their interactions [11], and hypergraph data structures can improve the overall recommendation process by modeling complex interconnections between more than two nodes [12, 13, 14]. For instance, consider the scenario where two users have recently listened to the same song with the tag "summer" but have different musical preferences (i.e. one likes metal, and the other likes rock). A simple graph representation cannot fully capture the relationship between the users, which may lead to inaccurate music recommendations. Instead, a hypergraph model can incorporate the relationship between the user, song and tag, and provide more meaningful suggestions by recognising that the core of the relationship is not the musical genre but the relaxed mood associated with being on holiday. Thus, hypergraph-based approaches may be more effective in generating accurate recommendations[15] by better understanding the nuanced relationships between users, songs and their associated metadata.

This paper is an extended abstarct of a recent proposal [16], in which the authors introduce *Hypergraph Embeddings for Music Recommendation (HEMR)*, a novel framework for music recommendation based on the *hypergraph* data model. HEMR utilizes hypergraph embedding techniques to calculate a user-song affinity score based on the information provided by hypergraphs, enabling the system to select the most suitable songs for each user. This represents the first attempt to utilize the benefits of hypergraph data structure and graph embedding methods in the music recommendation domain. By using the hypergraph data structure, the learnt representation is more effective due to the enrichment of the structural information of the graph [17].

Our experiments on the *Million Song Dataset* [18] show that HEMR outperforms several state-of-the-art recommendation baselines. In addition, we find that not only do the hypergraph embeddings are effective in representing the complex relations between entities (users, songs, artists, tags) but they also guarantee a significant improvement in performance with respect to the other methods, especially on top recommendations and cold-start scenarios. We release our code on github[1].

The paper is organized as follows: Section **??** describes related work for music recommendation and graph embedding techniques. Section 2 presents HEMR, along with the hyper-graph based data model on which it is based. Then, in Section 3 we evaluate our framework w.r.t. three state-of-the-art music recommendation techniques. Finally, Section 4 discusses conclusions and possible future works to enhance HEMR's capabilities.

2. Methodology

The overall workflow behind the proposed music recommender system is divided in the following steps: (i) *Hypergraph Data Modeling*: data is modeled and stored in an hypergraph-based structure; (ii) *Embedding Generation*: random walks and vertex embeddings are generated from the hypergraph data model; (iii) *Recommendation Generation*: a top-K song recommandation is generated for each user.

[1] https://github.com/picuslab/HEMR

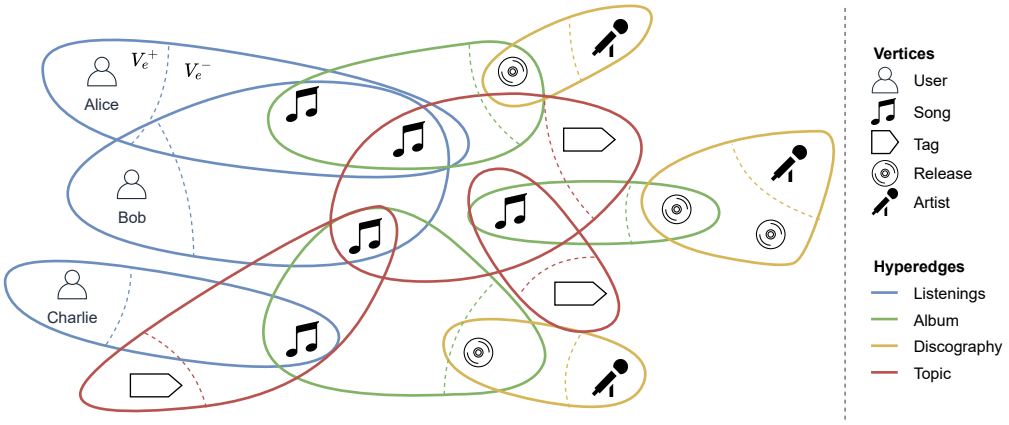

Figure 1: An example of the proposed Hypergraph data model in which three users listen to four songs to which different metadata are associated as well as releases, tags and artists.

We detail each stage of the framework in the next sections.

2.1. Hypergraph model

Aiming to interconnect all the information embedded in the data sources, information is modeled through an hypergraph data structure which incorporates relationships among various entities (e.g., users, songs, artists).

In order for a music recommender system to work, we should at least provide songs and user data. We could define relationships between users and songs by means of a graph data structure. Nevertheless, we can exploit the generalization power of hypergraphs expliciting the relationships between users and songs as higher degree hyperedges.

The hypergraph $\mathcal{H}$ is defined as a collection of vertices $\mathcal{V}$ and hyperedges $\mathcal{E}$. The vertex set $\mathcal{V}$ is basically composed by the following entities:

- *Songs (S):* the set of songs;
- *Users (U):* the set of persons or organizations constituting the particular social community;
- *Artists (A):* the set of artists who released songs embedded in the structure;
- *Releases (R):* the set of releases;
- *Tags (T):* the set of tags (e.g. rock, pop, female vocalist, love).

Several types of relationships can be identified between the above-mentioned entities. We define the following hyperedges:

- *Listenings:* the relationship between a user and all the songs which he has listened to.
- *Album:* the relationship between a release and all the songs which it contains.

- *Discography:* the relationship between an artist and all the releases he has published.
- *Topic:* the relationship between a tag and all its related songs.

An example of the resulting hypergraph is shown in Figure 1. While there are infinite possibilities on how to model music data in an hypergraph, the chosen configuration might potentially impact the functioning of the whole system: for example, generating higher degree hyperedges or increasing the number of vertices might require longer random walks or a bigger embedding size in order to properly extract the context information from the hypergraph, occasionally resulting into performance degradation and longer execution times.

2.2. Embedding generator

2.2.1. Random walks generator

The generation of random walks on the hypergraph $\mathcal{H}$ is the first step of HEMR embedding generation procedure. The algorithm used in this work is a slightly modified version of that proposed in [19]. In particular, for each vertex v, we randomly choose one of its hyperedges $e \in \mathcal{E}$. Then, we evaluate the probability p of jumping to another hyperedge, which is inversely proportional to the degree of the hyperedge $degree(e)$:

$$p_e = min\Big(\frac{\alpha}{degree(e)} + \beta, 1\Big), \tag{1}$$

where $\alpha, \beta \geq 0$ are tunable parameters.

This allows the algorithm to explore more deeply the hyperedges with more vertices while avoiding getting stuck in a loop inside of smaller hyperedges. Then, based on the probability p, the current hyperedge could either remain unchanged or switch to another hyperedge in which the current vertex is. Finally, we randomly choose the new vertex to be added in the random walk from the current hyperedge.

Vertices are iteratively added to each walk. The *length* of the walk (number of iterations) is specified by the user: the bigger the length, the deeper the neighborhood of each vertex is explored by the algorithm. While deeper walks allow to take count of more information to compute embeddings, they imply a loss in efficiency performance, since the computational complexity of the algorithm is $O(|\mathcal{V}| \cdot length)$, revealing that increasing the length too much might penalize running times.

Tuning the α and β parameters will also allow to regulate how the algorithm traverses the hypergraph: higher α values will reduce the ability of high-degree hyperedges to force the algorithm into exploring vertexes that belong to them, allowing it to switch the current hyperedge more frequently; meanwhile, higher β values will have the same result, but without the dependency from the current hyperedge degree.

2.2.2. Word2Vec

We leveraged the skip-gram implementation of *Word2Vec* [20], consisting of a single hidden-layer neural network, to predict the *context words* of a given target word, which are essentially the closest vertices in the hypergraph. The entire vocabulary (i.e., all $|\mathcal{V}|$ nodes) of the hypergraph,

i.e., words, were used as the model's vocabulary. As a result, each node was represented as a one-hot encoded vector of size $|\mathcal{V}|$, wherein the i-th value is set to 1 if the corresponding position represents the i-th node; otherwise, it is set to 0.

Since the model takes in input a single one-hot encoded word, the input layer of the network has $|\mathcal{V}|$ neurons, among which only the one which is associate to the input word will be activated. The output layer of the network will also have $|\mathcal{V}|$ neurons, one for each possible context node.

2.2.3. Training process

Initially, the model is trained by identifying *node pairings*, i.e. pairs of nodes occurring adjacent in random walks, indicating contextual connections between these nodes. During the training process, a "context window" w is defined. All nodes in a random walk that are within the context window of a target node v form a *word pairing* with v, referred to as context words for the target. Larger context windows enable the model to comprehend weaker contextual connections between nodes. In comparison to traditional graph embedding techniques, hypergraph data structures benefit from larger context windows so that the model can grasp broader vertex relationships.

We use a *softmax* function to compute the probability of each vertex v_j in the hypergraph, $j \in \{1, 2, ..., |\mathcal{V}|\}$, to be a context node of the input vertex $v_i \in \mathcal{V}$:

$$y_j = p(v_j|v_i) = \frac{exp(u_j)}{\sum_{k=1}^{|\mathcal{V}|} exp(u_k)}, \tag{2}$$

where u_j is the output of the last layer of the word2vec network, obtained by multiplying the one-hot encoded vector of the input vertex $\hat{v}_i$ with the corresponding weights in the embedding matrix W_i and context matrix W'_O:

$$u_j = {W'_o}^T {W_i}^T \hat{v}_i \tag{3}$$

It is worth to note that Word2Vec utilizes the *hierarchical softmax* method to enhance the computational efficiency of (softmax) evaluation by representing the probabilities of each node via a binary tree structure, which becomes especially significant in high-dimensional vocabularies, i.e. larger number of nodes.

Finally, lhe loss function, which seeks to maximize the probability of predicting the context nodes for a target node during training, can be expressed as the average negative log likelihood across all training set nodes.

2.3. Recommendation generator

For recommendation purposes, embeddings undergo z-score normalization to establish a common range and scaling of features before cosine similarities between each user's embedding and every item embedding are calculated. In general, the cosine similarity score, gauges the degree of similarity between two vectors in the embedding space. For our use case, it represents the proximity of a song to a user's music taste; values closer to 1 indicate a better match, those closer to 0 suggest orthogonality, and negative values close to -1 represent a poor match. Eventually, once the score has been evaluated for each user/item pair, we can return the *top K recommendations* for each user by selecting the K songs with the highest scores.

Table 1
Hypergraph's nodes and hyper-edges

Node	# entries
Songs	10,880
Users	918,725
Artists	3,754
Releases	8,966
Tags	156,159

Hyperedge	# entries
Listening	7,443,520
Album	8,966
Discography	3,754
Topic	156,159

3. Experiments

3.1. Experimental setup

We acquired data about songs and users from the large-scale *Million Song Dataset* [18]. In particular, we collected: (i) a subset of users, (ii) a subset of songs and (iii) songs' metadata such as the artist, the album and some tags assigned by users (e.g. genre). Then, we extracted the relationships described in Section 2.1 to model the hyperedges. Table 1 shows the overall statistics of the dataset.

For our experiments we divided the dataset into training (80%), validation (10%) and testing (10%) sets and dropped all users with less than 10 recordings. All the experiments have been run on *Kaggle Notebooks*. We used a virtual machine in CPU mode (4 CPU cores and 16GB RAM) for most of the implementation.

As other standard recommendation systems, we evaluate the performance with *recall@K* and *average precision@K* metrics, with $K \in \{5, 10, 20, 50, 100\}$. While recall quantifies how similar the recommendations are w.r.t. the ground truth data, the average precision scores the quality of the HEMR's ranking results.

Finally, we compared HEMR results with the following state-of-the-art recommender systems: (i) *Random Walks with Restarts (RWR)* [21]; (ii) *ItemRank* [22]; (iii) *CAME* [4]; (iv) *HRM* [23]; (v) *Deep Multimodal* [24].

3.2. Results

3.2.1. Comparison with baselines

Table 2 displays the Recall@k and MAP@k evaluation results for all the baselines under comparisons, with k values varying from 5 to 1000. Although RWR algorithm performs the best for k values of 50 and 100, our proposed algorithm achieves significantly superior Recall and MAP performance on top recommendations with k values of 5, 10, and 20 (with $\rho < 0.001$). From a user perspective, recommendations in these top spots hold most value. For instance, we notice that our algorithm yields around 3 times better performance in Recall@5 and approximately 4 times more in MAP@5 compared to the second-best baseline. The embeddings computation can further be ameliorated by including additional data such as the number of listenings or

Table 2
Comparison of our model with different baselines (Bold indicates the best results. $*$ indicates statistical significance at $p = 0.001$ compared to the second best)

Model	Recall@k					MAP@k				
	$k = 5$	$k = 10$	$k = 20$	$k = 50$	$k = 100$	$k = 5$	$k = 10$	$k = 20$	$k = 50$	$k = 100$
RWR [21]	0.008	0.058	0.183	**0.401***	**0.565***	0.008	0.058	0.180	**0.378***	**0.540***
ItemRank [22]	0.000	0.002	0.006	0.030	0.093	0.000	0.002	0.006	0.030	0.091
Deep Multimodal [24]	0.011	0.025	0.043	0.082	0.0.111	0.001	0.001	0.002	0.003	0.009
HRM [23]	0.045	0.059	0.075	0.100	0.126	0.028	0.030	0.031	0.319	0.323
CAME [4]	0.027	0.046	0.074	0.135	0.204	0.026	0.043	0.072	0.128	0.195
HEMR (Ours)	**0.134***	**0.185***	**0.250***	0.331	0.385	**0.125***	**0.170***	**0.221***	0.310	0.372

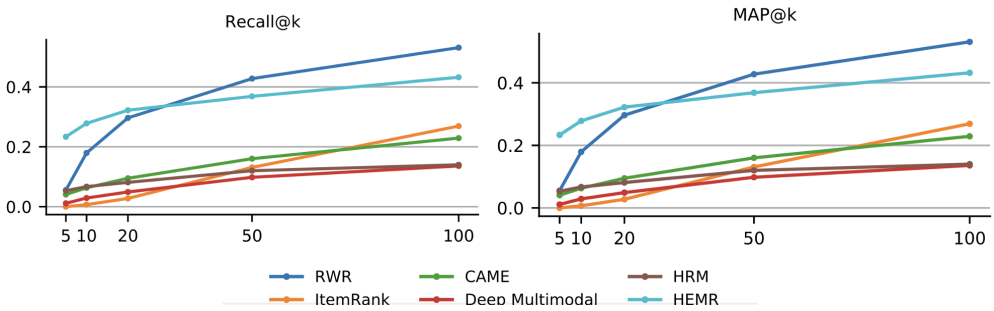

Figure 2: Cold-start experiment results: Recall and MAP scores on test set users with 5 or fewer listenings in the corresponding training set

content information, such as lyrics and raw audio. This would expectedly help us narrow the performance gap with RWR for higher ranked instances, such as those in the top 50 or top 100 category.

3.2.2. Cold-start evaluation

In order to assess baseline and overall system performance in cold-start situations, we evaluated their effectiveness on test users who had listened to five or fewer songs in the training set. Figure 2 depicts the results, which exhibit that our proposed approach is particularly effective in cold-start scenarios. For top recommendations, our approach offers superior quality than other baselines.

4. Conclusion & Future Work

This paper proposes a hypergraph model for music recommendation, which effectively embeds and interconnects all information related to songs, artists, and users. The proposed embedding approach has shown great promise in achieving a high-quality recommendation that includes both context and content information. In our experiments, it has outperformed other state-of-the-art techniques in terms of recall, average precision, and efficiency. Despite the promising results, using the hypergraph model still represents a relatively new direction, and

there is still room for improvement. For example, instant recommendations for new items require embedding calculation, and content-based information such as lyrics and raw audio can significantly contribute to more representative embeddings. Time-based information could also be incorporated in future research to be able to give greater weight to recently listened songs, which better capture the user's mood. Furthermore, our findings suggest that the application of hypergraph data embeddings also holds promise in domains beyond music recommendation such as movies, restaurants, etc. The system outputs both user and item embeddings, which could be used for other applications such as user clustering based on their proximity to each other to form groups of people with similar tastes (community detection).

5. Acknowledgements

We acknowledge financial support from the PNRR project "Future Artificial Intelligence Research (FAIR)" – CUP E63C22002150007

References

[1] A. Fatemeh, J. N. Nima, Recommender systems: A systematic review of the state of the art literature and suggestions for future research, Kybernetes 47 (2018) 985–1017.

[2] A. Barducci, S. Iannaccone, V. La Gatta, V. Moscato, G. Sperlì, S. Zavota, An end-to-end framework for information extraction from italian resumes, Expert Systems with Applications 210 (2022) 118487. URL: https://www.sciencedirect.com/science/article/pii/S095741742201572X. doi:https://doi.org/10.1016/j.eswa.2022.118487.

[3] A. van den Oord, S. Dieleman, B. Schrauwen, Deep content-based music recommendation, in: C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 26, Curran Associates, Inc., 2013, pp. 2643–2651. URL: https://proceedings.neurips.cc/paper/2013/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf.

[4] D. Wang, X. Zhang, D. Yu, G. Xu, S. Deng, Came: content-and context-aware music embedding for recommendation, IEEE transactions on neural networks and learning systems (2020).

[5] V. Moscato, A. Picariello, G. Sperli, An emotional recommender system for music, IEEE Intelligent Systems (2020) 1–1. doi:10.1109/MIS.2020.3026000.

[6] V. La Gatta, V. Moscato, M. Postiglione, G. Sperlì, An Epidemiological Neural network exploiting Dynamic Graph Structured Data applied to the COVID-19 outbreak, IEEE Transactions on Big Data (2020). doi:10.1109/TBDATA.2020.3032755.

[7] D. D'Auria, V. Moscato, M. Postiglione, G. Romito, G. Sperlí, Improving graph embeddings via entity linking: A case study on italian clinical notes, Intelligent Systems with Applications 17 (2023) 200161. URL: https://www.sciencedirect.com/science/article/pii/S2667305322000989. doi:https://doi.org/10.1016/j.iswa.2022.200161.

[8] M. Postiglione, Towards an italian healthcare knowledge graph, in: Similarity Search and Applications: 14th International Conference, SISAP 2021, Dortmund, Germany, September 29–October 1, 2021, Proceedings 14, Springer, 2021, pp. 387–394.

[9] A. De Santo, A. Galli, V. Moscato, G. Sperlì, A deep learning approach for semi-supervised community detection in online social networks, Knowledge-Based Systems 229 (2021) 107345. URL: https://www.sciencedirect.com/science/article/pii/S0950705121006079. doi:https://doi.org/10.1016/j.knosys.2021.107345.

[10] A. Ferraro, V. Moscato, G. Sperlì, Deep learning-based community detection approach on multimedia social networks, Applied Sciences 11 (2021) 11447.

[11] S. Wu, W. Zhang, F. Sun, B. Cui, Graph neural networks in recommender systems: A survey, 2020. arXiv:2011.02260.

[12] Y. Zhang, N. Wang, Y. Chen, C. Zou, H. Wan, X. Zhao, Y. Gao, Hypergraph label propagation network, Proc. AAAI Conf. Artificial Intelligence 34 (2020) 6885–6892. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6170. doi:10.1609/aaai.v34i04.6170.

[13] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, X. Zhang, Self-supervised hypergraph convolutional networks for session-based recommendation, CoRR abs/2012.06852 (2020). URL: https://arxiv.org/abs/2012.06852. arXiv:2012.06852.

[14] J. Wang, K. Ding, L. Hong, H. Liu, J. Caverlee, Next-Item Recommendation with Sequential Hypergraphs, Association for Computing Machinery, New York, NY, USA, 2020, p. 1101–1110. URL: https://doi.org/10.1145/3397271.3401133.

[15] F. Amato, F. Moscato, V. Moscato, F. Pascale, A. Picariello, An agent-based approach for recommending cultural tours, Pattern Recognition Letters 131 (2020) 341–347.

[16] V. La Gatta, V. Moscato, M. Pennone, M. Postiglione, G. Sperlí, Music recommendation via hypergraph embedding, IEEE Transactions on Neural Networks and Learning Systems (2022).

[17] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, X. He, Music recommendation by unified hypergraph: Combining social media information and music content, MM'10 - Proc. ACM Multimedia 2010 International Conference (2010) 391–400. doi:10.1145/1873951.1874005.

[18] T. Bertin-Mahieux, D. Ellis, B. Whitman, P. Lamere, The million song dataset., Proceedings of the 12th Int. Conference on Music Information Retrieval (ISMIR 2011) (2011) 591–596.

[19] J. Payne, Deep hyperedges: a framework for transductive and inductive learning on hypergraphs (2019). arXiv:1910.02633.

[20] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space (2013). arXiv:1301.3781.

[21] I. Konstas, V. Stathopoulos, J. Jose, On social networks and collaborative recommendation, Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (2009).

[22] M. Gori, A. Pucci, Itemrank: A random-walk based scoring algorithm for recommender engines, in: IJCAI, 2007.

[23] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, X. Cheng, Learning hierarchical representation model for nextbasket recommendation, in: Proc. 38th Int. ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 403–412.

[24] S. Oramas, O. Nieto, M. Sordo, X. Serra, A deep multimodal approach for cold-start music recommendation, Proc. 2nd Workshop on Deep Learning for Recommender Systems (2017). URL: http://dx.doi.org/10.1145/3125486.3125492. doi:10.1145/3125486.3125492.