MULTI-OBJECTIVE HYPER-PARAMETER OPTIMIZATION OF BEHAVIORAL SONG EMBEDDINGS

Massimo Quadrana
Apple
mquadrana@apple.com

Antoine Larreche-Mouly
Apple
alarreche@apple.com

Matthias Mauch
Apple
mmauch@apple.com

ABSTRACT

Song embeddings are a key component of most music recommendation engines. In this work, we study the hyper-parameter optimization of behavioral song embeddings based on Word2Vec on a selection of downstream tasks, namely next-song recommendation, false neighbor rejection, and artist and genre clustering. We present new optimization objectives and metrics to monitor the effects of hyper-parameter optimization. We show that single-objective optimization can cause side effects on the non optimized metrics and propose a simple multi-objective optimization to mitigate these effects. We find that next-song recommendation quality of Word2Vec is anti-correlated with song popularity, and we show how song embedding optimization can balance performance across different popularity levels. We then show potential positive downstream effects on the task of play prediction. Finally, we provide useful insights on the effects of training dataset scale by testing hyper-parameter optimization on an industry-scale dataset.

1. INTRODUCTION

Modern Recommendation Systems (RS) rely on embedding vectors to represent the latent user and item factors [1]. They can be employed for several downstream applications, ranging from song recommendation in radio stations or playlists [2–7], search [8, 9], tagging [10], to the generation of artist and genre representations for annotation and recommendation tasks [11–13]. Embeddings are usually generated in the early stages of complex RS pipelines, often through self-supervised methods like Word2Vec [14], which come with default hyper-parameters tuned on non-RS tasks (e.g., NLP).

Practitioners and researchers in the field hence need feasible optimization objectives and reliable metrics to compare against when optimizing embeddings. Recent research shows that hyper-parameter optimization can significantly improve recommendation quality [15,16]. While prior work mainly focuses on recommendation tasks, it is worth considering other tasks like false positive rejection and clustering during optimization.

Additionally, embedding spaces are used as a source of knowledge and interpretation either through visualization tools [17, 18], or by using relationships between items from a statistical standpoint or by leveraging information outside the input data [19–22]. We believe that music-RS practitioners and researchers could benefit from a deeper understanding of the behavior of song embedding optimization in relation to important factors such as song popularity. This could assist them in handling some emerging algorithmic biases like the popularity bias [23–25].

1.1 Contributions

In this work, we provide a strong framework within which it is possible to monitor the performance of embedding models and evaluate the potential of the embedding model to adapt to new tasks. Our work presents a general methodology that can be applied to any embedding system and not only to Word2Vec. The main contributions are:

- We define metrics and optimization objectives for three relevant tasks: next-song recommendation, false neighbor rejection, and genre and artist clustering.
- We demonstrate experimentally that single-objective optimization can have negative side effects on the non optimized metrics and propose a multi-objective optimization approach to combine recommendation and clustering objectives effectively.
- We show that next-song recommendation quality and song popularity are anti-correlated and reveal that song embedding optimization can balance performance across different popularity levels.
- We show the potential positive downstream effects on the task of play prediction revealing that the benefits of song embedding optimization extend beyond the tasks considered during optimization.
- Finally, we study embedding optimization at scale on an internal dataset of billions of listening events and show that increasing training dataset size allows for better configurations at the expense of longer optimization times.

2. METHOD

We consider song embeddings based on Word2Vec. This model was originally created to represent words in an

English corpus through a self-supervised shallow neural network trained to learn dense vector representations of the words from sentences based on the surrounding words [14]. The same principle is now commonly used in recommender systems to compute item embeddings from user interaction sequences like site sessions or playlists [25–28].

We study song embedding optimization with respect to four main tasks: next-song prediction, false neighbor rejection, artist clustering, and genre clustering. We define in this section the task and metrics that we used both for embedding optimization and evaluation of its effects on the resulting embedding space.

2.1 Next-Song Prediction

We choose next-song prediction as our primary target task. Next-item prediction is a common recommendation task whose goal is to predict the next item the user will interact with given some context [29]. In music recommenders, this often translates to predicting the next song given the history of songs played by the user [30]. To ensure the contextual relevance of recommendations, the past play history is often limited to the past few played songs or sessions, although more flexible solutions that look beyond a fixed horizon exist [31].

Since our main goal is not to build the most accurate next-song predictor, but rather to measure effects of optimization on the embedding space as directly as possible, we simplified next-song prediction to the extreme case of predicting the next played song based only on the *immediately preceding* song. For each song in the evaluation set (the *target* song henceforth), we consider the song the user played before it as the *query* song. For each (query, target) pair, we simply retrieve the top-100 exact nearest neighbors of the query and compute the average HitRate and Normalized Discounted Cumulative Gain (NDCG) on the top-100 ranked neighbors. HitRate is the fraction of times the target song is contained in the nearest neighbor of the query, while NDCG also accounts for the rank of the correct next-song within the predictions [32].

In order to ensure that improvements are due to true generalisation and not to overfitting, we split the evaluation into in-set and out-of-set. In *in-set evaluation*, we mask the last song in every training sequence and use second-to-last song as the query to compute next-song prediction metrics. In *out-of-set evaluation*, we hold-out the whole play sequences in the validation and test sets, and use every ordered pair of songs therein contained to compute the next-song prediction metrics. We use the average in-set and out-of-set HitRate and NDCG values in our experiments. The precise definition of a sequence varies between our experimental datasets and will be discussed in Section 3.

2.2 False Neighbor Rejection

Since high quality nearest neighbors are essential to providing a good recommendations, we are interested in evaluating the effectiveness of filtering out *spurious* song neighbors, i.e., songs that are in the closest neighborhood of another song merely by chance and not due to real behavioral or metadata similarity.

Query song	Hard Negative Neighbors
Paradise - Coldplay	Snowman - Sia Immigrant Song - Led Zeppelin Basket Case - Green Day
Smells Like Teen Spirit - Nirvana	Power - Kanye West Ob-La-Di, Ob-La-Da - The Beatles Rock Your Body - Justin Timberlake
Carry On Wayward Son - Kansas	Natural - Imagine Dragons Rap God - Eminen It Ain't Me - Kygo & Selena Gomez

Table 1: Examples of hard negative neighbors in Stream.

To this end we define the task of False Neighbor Rejection. We used a chi-squared X^2 test to identify false neighbors in the play sequence dataset [33]. The X^2 test compares the observed co-occurrence frequencies of every pair of songs in the listening sequences against the expected co-occurrence frequencies in case of independence. It is thus suitable to detect pairs of songs that appear in sequence by chance, and not because they are strongly related due to behavioral factors or metadata. In practice, we compute the X^2 coefficient for each unordered bigram of songs in the play sequence dataset. We consider as a *hard negative neighbor* of a song any other song having high co-occurrence but chi-squared statistic below a significance threshold [1] .

One limitation of this approach is that it requires a large number of events to be able to find frequently co-occurring song pairs by chance. We were therefore able to run this analysis only on the Stream dataset, from which we retrieved 5M hard-negative pairs, with an average of 92 hard-negatives per song. Some examples of hard negative neighbors are shown in Table 1. We define the HardNeg metric as the proportion of hard negatives for the top-100 nearest neighbors of every song (the smaller the better). We will use it as a safeguard metric agains undesired side effects of song embedding optimization.

2.3 Artist and Genre Clustering

We are interested in measuring how strongly classes such as artists and genres cluster together in a given embedding space. We introduce here the concept of Local Genre Coherence of the embedding space as the average fraction of songs in nearest neighbors set having the same primary genre as the query song. We similarly define the Local Artist Coherence as the average fraction of nearest neighbors belonging to the same artist as the query song. For example, an embedding space has Local Genre Coherence $= 0.5$ if on average 50% of the nearest neighbors of each song have its same primary genre.

Computing Local Coherence metrics is a costly operation since it requires us to inspect the nearest neighbors of every embedded song. We instead propose to use as *proxy metric* during optimization the much cheaper Caliński-

[1] We empirically chose 10^{-7} times the sum of all song occurrences in the dataset as threshold.

Dataset	Stream	Stream-1%	LFM-1b
Songs	66M	17M	31M
Events	255B	2.6B	1B
Sequences	0.9B	9.3M	120K
Sequence length: mean	277	276	9044
Sequence length: 25th percentile	16	16	1138
Sequence length: median	29	27	3930
Sequence length: 75th percentile	58	57	16155

Table 2: The statistics of the datasets.

Parameter	Default	Range	Description
d	100	$[25, 200]$	Embedding vector dimension
L	5	$[1, 40]$	Sliding window max length
α	0.75	$[-1.0, 1.0]$	Negative sampling exponent
N	5	$[1, 100]$	Number of negative samples
λ	0.025	$[0.001, 0.1]$	Initial learning rate

Table 3: Default Word2Vec hyper-parameters and their respective optimization ranges.

Harabasz index or Variance Ratio Criterion (VRC) [34] over artist and genre clusters in the embedding space. The VRC is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all genre clusters, where dispersion is the sum of squared distances. Higher VRC values correspond to better separation among clusters, which is a desirable property because it reduces the chances of uncontrolled, cross-artist and cross-genre pollution in recommendations.

We study the optimization of embedding spaces with respect to clustering metrics and their relation to recommendation quality in Section 4.3 and 4.4.

3. DATASETS

We run our experiments on two datasets. The first is a large-scale proprietary dataset of anonymized streaming listening sequences and playlists. We call this dataset Stream. The second is the LFM-1b dataset which contains 1B time-stamped listening events collected from Last.fm [35].

The main statistics of each dataset are shown in Table 2. The sequence length distribution differs significantly between the two datasets. Stream contains large numbers of both long listening suquences and short playlists (the median length is 58, but mean is 277), while LFM-1b contains mainly long listening sequences (the median length is 3930). This will have an impact on the optimal hyper-parameters discovered for each dataset.

In order to compute artist and genre clustering metrics we need song-level artist and genre annotations. For the Stream dataset, each song in is mapped to the corresponding primary genre and artist through an internal auto-tagging pipeline. LFM-1b already comes with song to artist mappings, however song-level genre annotations are not directly available. We hence mapped each song to the *first* Freebase artist genre from the LFM-1b User Genre Profile dataset [36]. We are aware of the noise introduced by this approximation, yet we believe it provides a useful contribution towards the reproducibility of our experiments.

We partition both datasets by randomly sampling sequences without replacement with proportions 98/1/1 for training, validation and test respectively. However, Stream still contains 7500 times more sequences than LFM-1b, which makes optimization at full-scale impractical. We hence downsample the sequences in the dataset using a 1% rate. We will refer to this subsampled dataset to as Stream-1%. We will investigate how to scale the optimization up to the full dataset in more detail in Section 4.7.

4. EXPERIMENTS

We report here the optimization of song embeddings for the tasks defined in the previous section. We optimize the hyper-parameters of skip-gram Word2Vec by running Bayesian Hyper-Parameter Optimization (HPO) [37], initialized with 10 iterations of Random Search [38] before running Bayesian Search until convergence. Similarly to previous work, we constrained all training times to be approximately equal to the default Word2Vec configuration [16]. This ensures a fair comparison among trials, and prevents the optimizer from discovering configurations that are impractical to train at large scales.

4.1 Background on Word2Vec

Both variants of Word2Vec, Skipgram and the Continuous Bag of Words (CBOW), are self-supervised shallow neural network models trained by minimizing the categorical cross-entropy loss with approximation softmax [14]. Here we consider Skipgram with negative sampling for its superior computational efficiency [39].

The hyper-parameters of Word2Vec, their defaults and optimization ranges are detailed in Table 3. In short, d is the embedding size, L is the maximum window length, α controls the negative sampling (uniform sampling for $\alpha = 0$, popularity sampling $\alpha > 0$, inverse popularity sampling for $\alpha < 0$), N is the number of negative samples used to approximate the softmax and λ is the learning rate.

4.2 Optimizing for Next-Song Recommendation

We first analyze the optimization of next-song recommendation quality by running HPO with the HitRate objective. In line with previous works, we observe significant improvements with respect to the default configurations on both the tested datasets (second line of Table 4). On Stream-1%, HitRate improves by 5% and NDCG by 7%, while HardNeg reduces drastically by 40%. The reduction in HardNeg ensures that the improvement in recommendation accuracy does not comes at the expense of more false positive neighbors. On LFM-1b we similarly observe +19% HitRate and +31% NDCG. The results on this task are in line with previous findings [15, 16].

We are now also able to monitor the effects of optimization on Genre and Artist Clustering metrics. While, $\text{VRC}_{\text{Genre}}$ and $\text{VRC}_{\text{Artist}}$ slightly increase on Stream, they both reduce on LFM-1b. This suggests that recommendation and clustering metrics may be slightly anti-correlated

		Stream-1%					LFM-1b			
Opt. Type	Objective	HitRate	NDCG	HardNeg	$\text{VRC}_{\text{Genre}}$	$\text{VRC}_{\text{Artist}}$	HitRate	NDCG	$\text{VRC}_{\text{Genre}}$	$\text{VRC}_{\text{Artist}}$
N/A	N/A	0.3538	0.1378	0.0180	927	3.373	0.3771	0.1562	520	4.141
Single-obj	HitRate	$0.3725^{\uparrow}$	$0.1482^{\uparrow}$	$0.0108^{\uparrow}$	1033	4.437	$0.4492^{\uparrow}$	$0.2050^{\uparrow}$	382	3.457
	$\text{VRC}_{\text{Genre}}$	0.3000	0.1218	0.0146	2006	5.521	0.3776	0.1609	1293	8.603
	$\text{VRC}_{\text{Artist}}$	0.3402	0.1336	0.0175	1397	**34.347**	0.3532	0.1488	**1444**	**94.476**
Multi-obj	$\lambda_{\text{Genre}}(0.01)$	$\mathbf{0.3772}^{\uparrow}$	$\mathbf{0.1586}^{\Uparrow}$	$0.0084^{\uparrow}$	1495	5.216	$0.4428^{\uparrow}$	$\mathbf{0.2092}^{\uparrow}$	623	5.955
	$\lambda_{\text{Genre}}(0.1)$	$0.3742^{\uparrow}$	$0.1515^{\Uparrow}$	$0.0096^{\uparrow}$	1617	5.771	$0.4067^{\uparrow}$	$0.1698^{\uparrow}$	997	7.298
	$\lambda_{\text{Artist}}(0.01)$	$0.3699^{\uparrow}$	$0.1520^{\Uparrow}$	$\mathbf{0.0057}^{\Uparrow}$	1166	4.580	$0.4458^{\uparrow}$	$0.1883^{\uparrow}$	487	5.469
	$\lambda_{\text{Artist}}(0.1)$	0.3331	0.1332	$0.0093^{\uparrow}$	**2233**	7.422	$\mathbf{0.4537}^{\uparrow}$	$0.1998^{\uparrow}$	619	6.022

Table 4: Results of hyper-parameter optimization on both datasets. The first line reports the metrics for the default configuration. Best results are in **bold**. For HR/NDCG/HardNeg only: $^{\uparrow}$ and $^{\Uparrow}$ denote stat. sig. improvement over the default and the best single-objective configurations respectively (paired t-test at $p < 0.01$ with Bonferroni correction).

for this dataset. We will investigate this effect further in Section 4.3 and 4.4.

4.3 Optimising for Genre and Artist Clustering

We now analyze the optimization of embeddings with respect to the Local Genre and Artist Coherence by using the Variance Ratio Criterion as a proxy objective.

We first show that VRC is a suitable proxy by comparing it against Local Coherence metrics on Stream-1%. We computed Local Genre Coherence on 500 songs with at least 10k plays selected using stratified sampling across the top-10 played genres and using 50 nearest neighbors per song. Figure 1 (left) shows strong positive correlation between average Local Genre Coherence and $\text{VRC}_{\text{Genre}}$ of 5 embedding spaces generated with different hyper-parameters. We similarly computed Local Artist Coherence by sampling 125 artists having at least 25 songs using stratified sampling by artist popularity to account for popularity biases. We then average artist coherence score over 5 songs per artist and 50 nearest neighbors per song. Figure 1 (right) shows positive correlation between Local Artist Coherence and $\text{VRC}_{\text{Artist}}$ of 5 embedding spaces generated with different hyper-parameters.

Table 4 shows that the optimal configuration for genre clustering on Stream-1% doubles the $\text{VRC}_{\text{Genre}}$ score compared to the default hyper-parameters, but with a significant reduction in terms of HitRate and NDCG. The next-song recommendation quality is thus badly affected by the myopic optimization of genre coherence. This effect is slightly less evident for artist clustering optimization, in which we were able to obtain 10 fold greater $\text{VRC}_{\text{Artist}}$ with negligible decrease in HitRate and NDCG. In both cases, HardNeg is not significantly affected.

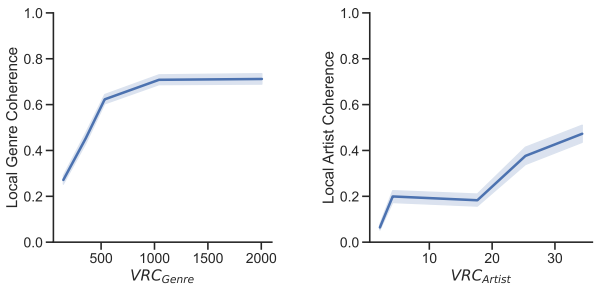

Figure 1: Local Coherence for genres (left) and artists (right) on Stream-1%. Means and 95% Confidence Intervals are shown.

We observe similar effects on LFM-1b, where significant improvements in $\text{VRC}_{\text{Genre}}$ and $\text{VRC}_{\text{Artist}}$ correspond to non significant improvement (genre clustering) or significant deterioration (artist clustering) of next-song recommendation metrics.

For both datasets, the optimal configurations have significantly worse next-song recommendation quality than the optimal one found by single-objective next-song recommendation. This suggests that optimizing embeddings for clustering alone can seriously harm the recommendation quality. In the next section we tackle the problem of optimizing both objectives simultaneously.

4.4 Multi-objective Optimization

High quality next-song recommendation and genre/artist clustering are both desirable properties of the embedding space but, as we have just observed, optimizing for a single objective can harm others. We investigate here the simultaneous optimization of both objectives through Multi-Objective Hyper-parameter Optimization (MOHPO).

The simplest approach to MOHPO is scalarization, which transforms a multi-objective goal into a single-objective one. Examples of scalarization are the weighted sum, Tchebycheff or ϵ-constraint approaches [40]. Advanced MOHPO techniques extend evolutionary algorithms and Bayesian Optimization to explicitly handle different trade-offs between the multiple objectives [40]. The choice of the optimal MOHPO method is beyond the scope of this work, hence we opted for Bayesian Optimization with scalarization since it fits easily into our existing optimization framework.

We focus on the joint optimization of HitRate and $\text{VRC}_{\text{Genre}}$. The same reasoning holds for HitRate and $\text{VRC}_{\text{Artist}}$ and it is omitted for space reasons. Since these metrics have widely different scales, we first define the relative improvement in $\text{VRC}_{\text{Genre}}$ of a new hyper-parameter

configuration t with respect to the default configuration:

$$\text{rVRC}_{\text{Genre}}{}^{(t)} = \frac{\text{VRC}_{\text{Genre}}{}^{(t)} - \text{VRC}_{\text{Genre}}{}^{(def)}}{\text{VRC}_{\text{Genre}}{}^{(def)}} \quad (1)$$

where $\text{VRC}_{\text{Genre}}{}^{(def)}$ is the $\text{VRC}_{\text{Genre}}$ for the default configuration. The new scalarized objective is the simple convex combination of the two objectives:

$$\lambda_{\text{Genre}}(\alpha) = \alpha\text{HitRate}^{(t)} + (1-\alpha)\text{rVRC}_{\text{Genre}}{}^{(t)} \quad (2)$$

where α controls the relative weight of next-song recommendation and genre clustering objectives in the optimization.

We tried values of $\alpha = 0.01$ and 0.1. Results shown in Table 4 highlight the effectiveness of this approach on Stream-1%. While there is not a single solution that performs best on all metrics, the configuration found with objective $\lambda_{\text{Genre}}(0.01)$ *dominates* the best solution discovered by next-song optimization on all metrics, with statistically better NDCG (+7%). The configuration found with objective $\lambda_{\text{Artist}}(0.01)$ also achieves statistically better HardNeg, with a reduction of 68%, at the expense of lower clustering scores. In both cases increasing the value of α to 0.1 we can effectively trade some next-song prediction accuracy for better genre and artist clustering quality. The optimal setup depends on the final application.

Similarly, on LFM-1b we observe that the best next-song recommendation strategies are found through multi-objective optimization, although no solution entirely dominates any single-objective this time. This fact can be partly attributed to the noisy genre annotations we have available for this dataset.

These results highlight the effectiveness of combining recommendation and clustering objectives, which can mutually benefit from each other if combined properly.

4.5 Insights on Song Popularity

The effects of popularity on recommendations are subject of intense research activity within the research community [23–25]. We contribute here by studying the effects of next-song recommendation HPO on (query, target) song pairs belonging to various buckets of popularity.

We first categorize songs into buckets of popularity, each of which comprises 20% of the total listening events in the dataset, being 0 the smallest bucket with most popular songs and 4 the largest bucket containing the least popular ones [2]. We randomly sample 1k songs per bucket pair and the aggregate HitRate based on query *and* target bucket. For the sake of space, we analyze the Stream dataset only here. The results on LFM-1b are available in the Supplementary Material.

Figure 2a shows the values of HitRate by bucket pair for the default configuration of Word2Vec. We first observe that the recommendation quality is localized to the same or nearest popularity bucket to which the query song belongs. As the query and target begin to differ in popularity, HitRate drops. We clearly see that HitRate is anti-correlated with popularity. One possible explanation is that popular songs by definition give less information about the user tastes and hence have larger sets of plausible "next-songs". On the other hand, the less popular songs often belong to taste "niches" and are thus easier to model.

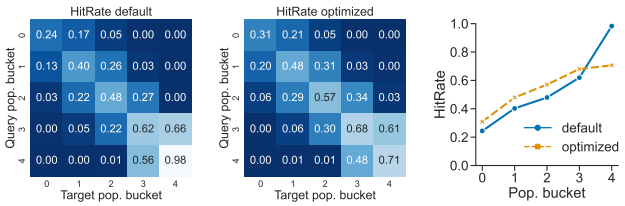

Figure 2: Query-target popularity analysis for Stream: HitRate by popularity bucket for the default configuration (a) and the optimized one (b), and HitRate for song pairs in the same popularity bucket (c). Bucket 0 contains the most popular songs.

Figure 2b shows the values of HitRate by bucket pair for embeddings for the optimal $\lambda_{\text{Genre}}(0.01)$ multi-objective configuration. The behavior of localized performance and anti-correlation observed using the default configuration is still visible. However, we notice a significant difference between the two configurations across the main diagonal. We have highlighted this difference in Figure 2c. We can see that tuning brings significant gains in HitRate at all buckets except for the least popular songs in 4. On both datasets [3] optimization seems to balance recommendation accuracy more across popularity buckets, which is a desirable effect.

4.6 Insights on Play Prediction

As song embeddings are usually used as inputs to other pipelines, we wanted to explore whether embedding optimization had beneficial effects on a different task from the one it was originally run on. For the sake of this experiment, we considered the problem of Play Prediction in seeded radio and autoplay. In seeded radio, the user selects a *seed*, e.g., an artist or a song, and the recommender generates an endless sequence of songs that are related to that seed. Similarly, autoplay generates a stream of music starting from the last played song in an album or a playlist. Seeded radio and autoplay are two prominent product features that allow users to generate streams of songs that are fully machine-learning driven. In both cases, the user organically selects only the seed or the collection from which to start the stream of music, and everything else is left to the recommendation algorithm to decide.

In this context, we study the reaction that users have on the *first* recommended song, since it is where the transition from organic to algorithmic selection happens. A bad listening experience at this point could easily induce users to stop listening entirely. We thus consider the task of pre-

[2] The songs that account for the top-20% of all plays end in bucket 0, the ones for the top-20% to 40% end in bucket 1, etc.

[3] Figures for LFM-1b omitted to fit within space limits.

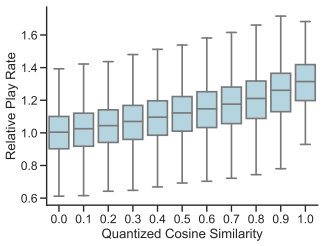

Figure 3: Visualization of the positive correlation between Relative Play Rate and cosine similarity on default embeddings (similarities quantized to the first decimal). Play Rates are scaled relative to cosine similarity $= 0.0$ to preserve business sensitive values.

	Default	Single-obj	Multi-obj
All pairs	0.2055	0.2140	0.2304
Frequent pairs	0.3335	0.3480	0.3717

Table 5: Pearson's correlation between Play Rate and cosine similarity for default and optimized embeddings.

dicting whether the first song generates a play event that lasts at least 30 seconds. We call this task Play Prediction.

Building an accurate predictor is beyond the scope of this paper. We instead measure the correlation between the Play Rate, i.e., the average number of times one organic play is followed by a successful algorithmic play, and the cosine similarity between the embeddings of the recommended and the seed songs. We use Word2Vec embeddings trained on the full Stream dataset. We compare default hyper-parameter configuration to the single-objective and multi-objective configurations having the highest HitRate on this dataset. We compute Play Rates on a proprietary dataset composed of 4.2M song pairs extracted from 184M listening sessions.

Figure 3 shows that there is a strong positive correlation between Play Rate and cosine similarity between embeddings. We observe positive correlation for both the default and optimized embedding spaces (Table 5). The correlation is stronger for the most frequent pairs (≥ 100 occurrences). However, the correlation is stronger for the optimized embeddings than the default ones, $+4\%$ for the best single-objective configuration and $+12\%$ for the best multi-objective configuration. This suggests that embedding optimization can have beneficial effects on tasks different from the one it was initially designed to address. Furthermore, it provides additional evidence on the superiority multi-objective optimization over single-objective one.

4.7 Optimization at Scale

To the best of our knowledge, there exists little literature on hyper-parameter optimization over massive datasets with billions of sequences and events. Previous work on the topic either optimize directly on the full dataset, which is unfeasible at scales like our Stream dataset, or consider a *fixed* subsample rate [16]. However, it is unclear which subsample rate would lead to results that are representative of performance at full-scale.

Sampling	Opt. time	HitRate	NDCG	HardNeg	$\text{VRC}_{\text{Genre}}$	$\text{VRC}_{\text{Artist}}$
N/A	N/A	0.3079	0.1217	0.0126	7229	3.266
1%	28h	0.3432	0.1378	0.0109	7367	3.780
2%	47h	0.3499	0.1421	0.0063	6886	3.374
5%	83h	0.3729	0.1506	0.0167	10410	3.992

Table 6: Results of multiple scale HPO on Stream, with the total optimization time and the best metrics obtained full scale (first line refers to the default configuration). Better performance is obtained at the largest subsample rates at the expense of longer optimization times. All pairwise comparisons on HitRate, NDCG and HardNeg are stat. sig. at $p < 0.01$ using Bonferroni correction, except for HardNeg between default and 1% sampling.

We explore this aspect by running Hyper-Parameter Optimization at increasing training dataset scales. The best hyper-parameters found at each data scale are then used to train the model at full-scale. We limit training time to stay within 25% more the time of training runs with the default hyper-parameters. This allows us to identify the best representative subsample rate while keeping optimization times within reasonable limits.

We split the Stream dataset into 4 overlapping training sets containing 1%, 2%, 5% and 98% of randomly selected sequences respectively, We run Bayesian hyper-parameter search on the first 3 splits and use the best hyper-parameters from each run to train Word2Vec at full 98% scale. For simplicity, we consider just single-objective HitRate optimization. Table 6 shows that there is significant correspondence between the sampling rates used during optimization and the final performance on the full scale dataset. The optimal configuration found at 5% scale is by far the best when evaluated at full-scale on all metrics. However, the total optimization time increases significantly when larger subsamples are used.

5. CONCLUSIONS

In this paper we analyzed the offline optimization of song embeddings through the lens of the tasks they are often employed on downstream. We proposed an effective way to optimize Word2Vec hyper-parameters on recommendation and clustering tasks jointly, with substantial benefits over their respective single-objective optimization variants. We investigated the interactions between next-song recommendation and song popularity. Our results suggest that careful optimization has the desirable property of balancing algorithm performance across popularity buckets. We showed the potential positive effects of optimization on the downstream task of Play Prediction, which provided further evidence on the superiority of multi-objective optimization over single-objective one. Finally, we investigated the effects of optimization at scale, which is particularly relevant for industrial applications.

As future work, we plan to validate these insights with online A/B testing. Our approach can be extended to other tasks, like diversity and fairness, and to the latest findings in Multi-Objective Hyper-parameter Optimization [40,41].

6. AKNOWLEDGEMENTS

We would like to thank Matt Jockers and all the members of the Music Machine Learning team for their help in improving and proofreading this work.

7. REFERENCES

[1] A. Pal, C. Eksombatchai, Y. Zhou, B. Zhao, C. Rosenberg, and J. Leskovec, "Pinnersage: Multi-modal user embedding framework for recommendations at pinterest," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2311–2320. [Online]. Available: https://doi.org/10.1145/3394486.3403280

[2] G. Bonnin and D. Jannach, "Automated generation of music playlists: Survey and experiments," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, pp. 1–35, 2014.

[3] C.-W. Chen, P. Lamere, M. Schedl, and H. Zamani, "Recsys challenge 2018: Automatic music playlist continuation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ser. RecSys '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 527–528. [Online]. Available: https://doi.org/10.1145/3240323.3240342

[4] A. Patwari, N. Kong, J. Wang, U. Gargi, M. Covell, and A. Jansen, "Semantically meaningful attributes from co-listen embeddings for playlist exploration and expansion," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, October 11-16, 2020*, J. Cumming, J. H. Lee, B. McFee, M. Schedl, J. Devaney, C. McKay, E. Zangerle, and T. de Reuse, Eds., 2020, pp. 527–533. [Online]. Available: http://archives.ismir.net/ismir2020/paper/000125.pdf

[5] R. T. Irene, C. Borrelli, M. Zanoni, M. Buccoli, and A. Sarti, "Automatic playlist generation using convolutional neural networks and recurrent neural networks," in *2019 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[6] C. Hansen, R. Mehrotra, C. Hansen, B. Brost, L. Maystre, and M. Lalmas, "Shifting consumption towards diverse content on music streaming platforms," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ser. WSDM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 238–246. [Online]. Available: https://doi.org/10.1145/3437963.3441775

[7] J. L. Moore, S. Chen, T. Joachims, and D. Turnbull, "Learning to embed songs and tags for playlist prediction," in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, F. Gouyon, P. Herrera, L. G. Martins, and M. Müller, Eds. FEUP Edições, 2012, pp. 349–354. [Online]. Available: http://ismir2012.ismir.net/event/papers/349-ismir-2012.pdf

[8] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal metric learning for tag-based music retrieval," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 2021, pp. 591–595. [Online]. Available: https://doi.org/10.1109/ICASSP39728.2021.9413514

[9] S. Doh, J. Lee, and J. Nam, "Million song search: Web interface for semantic music search using musical word embedding," in *International Society for Music Information Retrieval Conference, ISMIR 2021*. International Society for Music Information Retrieval, 2021.

[10] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched music representations with multiple cross-modal contrastive learning," *IEEE Signal Processing Letters*, vol. 28, pp. 733–737, 2021.

[11] K. Chen, B. Liang, X. Ma, and M. Gu, "Learning audio embeddings with user listening data for content-based music recommendation," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3015–3019.

[12] F. Korzeniowski, S. Oramas, and F. Gouyon, "Artist similarity using graph neural networks," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. L. 0001, A. L. 0001, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., 2021, pp. 350–357. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000043.pdf

[13] E. V. Epure, G. Salha, and R. Hennequin, "Multilingual music genre embeddings for effective cross-lingual music item annotation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*. Montreal, Canada: ISMIR, Oct. 2020, pp. 803–810. [Online]. Available: https://doi.org/10.5281/zenodo.4245556

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[15] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier, "Word2vec applied to recommendation: Hyperparameters matter," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ser. RecSys

'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 352–356. [Online]. Available: https://doi.org/10.1145/3240323.3240377

[16] B. P. Chamberlain, E. Rossi, D. Shiebler, S. Sedhain, and M. M. Bronstein, "Tuning word2vec for large scale recommendation systems," *Fourteenth ACM Conference on Recommender Systems*, Sep 2020. [Online]. Available: http://dx.doi.org/10.1145/3383313.3418486

[17] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, "Embedding projector: Interactive visualization and interpretation of embeddings," 2016. [Online]. Available: https://arxiv.org/abs/1611.05469

[18] P. Tovstogan, X. Serra, and D. Bogdanov, "Web interface for exploration of latent and tag spaces in music auto-tagging," in *Proceedings of the 37 th International Conference on Machine Learning; 2020 Jul 13-18; Vienna, Austria.[Vienna]: ICML; 2020.[3 p.]*. ICML, 2020.

[19] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 238–247. [Online]. Available: https://aclanthology.org/P14-1023

[20] J. Andreas and D. Klein, "How much do word embeddings encode about syntax?" in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 822–827. [Online]. Available: https://aclanthology.org/P14-2133

[21] Y. Chen, B. Perozzi, R. Al-rfou, and S. Skiena, "The expressive power of word embeddings," in *In: ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

[22] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele, "Evaluation of output embeddings for fine-grained image classification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2015.7298911

[23] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," in *The thirty-second international flairs conference*, 2019.

[24] D. Kowald, M. Schedl, and E. Lex, "The unfairness of popularity bias in music recommendation: A reproducibility study," in *Advances in Information Retrieval*, J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, Eds. Cham: Springer International Publishing, 2020, pp. 35–42.

[25] A. Vall, M. Quadrana, M. Schedl, and G. Widmer, "Order, context and popularity bias in next-song recommendations," *International Journal of Multimedia Information Retrieval*, vol. 8, no. 2, pp. 101–113, 2019.

[26] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2016, pp. 1–6.

[27] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1809–1818. [Online]. Available: https://doi.org/10.1145/2783258.2788627

[28] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 130–137. [Online]. Available: https://doi.org/10.1145/3109859.3109896

[29] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Comput. Surv.*, vol. 51, no. 4, jul 2018. [Online]. Available: https://doi.org/10.1145/3190616

[30] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fams.2019.00044

[31] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, and M. Lalmas, *Contextual and Sequential User Embeddings for Large-Scale Music Recommendation*. New York, NY, USA: Association for Computing Machinery, 2020, p. 53–62. [Online]. Available: https://doi.org/10.1145/3383313.3412248

[32] E. M. Voorhees and D. K. Harman, *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press, 2005.

[33] C. Manning and H. Schutze, *Foundations of statistical natural language processing*. MIT press, 1999.

[34] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/03610927408827101

[35] M. Schedl, "The lfm-1b dataset for music retrieval and recommendation," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 103–110. [Online]. Available: https://doi.org/10.1145/2911996.2912004

[36] M. Schedl and B. Ferwerda, "Large-scale analysis of group-specific music genre taste from collaborative tags," in *2017 IEEE International Symposium on Multimedia (ISM)*, 2017, pp. 479–482.

[37] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[38] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[39] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12. Madison, WI, USA: Omnipress, 2012, p. 419–426.

[40] F. Karl, T. Pielok, J. Moosbauer, F. Pfisterer, S. Coors, M. Binder, L. Schneider, J. Thomas, J. Richter, M. Lang, E. C. Garrido-Merchán, J. Branke, and B. Bischl, "Multi-objective hyperparameter optimization – an overview," 2022. [Online]. Available: https://arxiv.org/abs/2206.07438

[41] M. Abdolshah, A. Shilton, S. Rana, S. Gupta, and S. Venkatesh, "Multi-objective bayesian optimisation with preferences over objectives," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/a7b7e4b27722574c611fe91476a50238-Paper.pdf