code{4}lib JOURNAL

Issue 42, 2018-11-08 ISSN 1940-5758

Automated Playlist Continuation with Apache PredictionIO

The Minrva project team, a software development research group based at the University of Illinois Library, developed a data-focused recommender system to participate in the creative track of the 2018 ACM RecSys Challenge, which focused on music recommendation. We describe here the large-scale data processing the Minrva team researched and developed for foundational reconciliation of the Million Playlist Dataset using external authority data on the web (e.g. VIAF, WikiData). The secondary focus of [illegible]ating and adapting the processing tools that support data reconciliation. This paper reports on the playlist enrichment process, indexing, and subsequent recommendation model developed for the music recommendation challenge.

by Jim Hahn

1. Introduction

The purpose of this paper is to report the results of the Minrva team's (https://minrvaproject.org) development of a data-focused recommender system for the 2018 ACM RecSys Challenge [1]. The guiding contention of our team's work was that while algorithm development, ensemble construction [2], and selection [3] are vital elements of playlist completion, we viewed the primary problem of playlist completion as one of data reconciliation. In this paper, we describe the data processing that the Minrva team researched and developed for foundational reconciliation of the Million Playlist Dataset (MPD) [4] using external data sources on the web. The secondary focus of the research was the processing tools that support data reconciliation.

Machine learning is of interest to many data-intensive fields and is a growing topic in multiple applied computing domains in addition to powering innovations in music recommendation systems (MRS) [5]. As an example, digital library environments are no exception to the machine learning trend for recommender systems [6]. Digital libraries traditionally have focused on data preservation and access. Digital preservation seeks first to manage and describe data, but more recently, the field has sought to add value to data, much like the Minrva team's approach to enriching the Spotify MPD. The Minrva team, a software development research group based at the University of Illinois Library sought to reconcile a portion of the Spotify MPD with the Virtual International Authority File or VIAF, a large dataset available openly that library professionals have curated for many years. The VIAF provides an authoritative source of linked data for named entities, such as the names of artists from the Spotify MPD [7]. The field of digital librarianship's focus on data curation and e-science [8] has generated an underlying premise for data intensive research of the re-use of datasets over time and adding value to data while also supporting their preservation. Overall, the ACM RecSys Challenge offered a rich area in which to put data curation and accompanying reconciliation techniques from digital librarianship into practice, including the enrichment and transformation of the MPD's large corpus (over 2 million unique tracks).

Together with machine learning's noted popularity in many different applied computing domains, the parallel development of machine learning tools, such as common algorithms, is now available within software frameworks that are implemented easily. With this growing integration of machine learning software into big data tools, such as MLlib in Apache Spark in general, and the Alternating Least Squares algorithm for recommendations in particular, the Minrva project team was able to reuse existing machine learning algorithms, frameworks, and data models for the playlist continuation task. Specifically, the Alternating Least Squares implicit feedback approach [9] incorporated into Apache Spark was used to generate Minrva's recommendation model for the ACM RecSys Challenge 2018. Before moving on to challenge metrics and tasks, we explore the anonymized playlist metadata provided by Spotify within the MPD and the accompanying song metadata.

1.1 Million Playlist Dataset (MPD) Metadata Elements

The playlist metadata that were made available by Spotify were encoded in JSON and used associative arrays to convey higher level details about the playlist. As shown below, the playlist incorporates song data in a hierarchical structure. Track data are the second level of data with the playlist data inherited by each song in the list. Metadata in the playlist element field included: the name of the playlist (free text); whether it was collaboratively developed (true/false); a playlist id (numeric), the last time it was modified (numeric), the number of tracks in the playlist (numeric), the number of albums in the playlist (numeric), and the number of followers (numeric).

```
"playlists": [
  {
    "name": "example",
    "collaborative": "",
    "pid": ,
```

```
    "modified_at": ,
    "num_tracks": ,
    "num_albums": ,
    "num_followers": ,
    "tracks": [
      {
        "pos": 0,
        "artist_name": "",
        "track_uri": "spotify:track:" ,
        "artist_uri": "spotify:artist:" ,
        "track_name": "",
        "album_uri": "spotify:album:",
        "duration_ms": ,
        "album_name": ""
      }
      ...
    ]
  }
  ...
]
```

Metadata elements which Spotify provided as track data included: the position of the track in the playlist; the name of the artist (free text); a track uri (spotify controlled identifier); a track name (free text); an album uri (spotify controlled identifier); duration of the track (milliseconds); the name of the album (free text). It should be noted that these are the elements that were provided to our team after registering for the challenge, we did not see nor did the team consult documentation regarding Spotify data storage, therefore, it could be an approximation or abstraction as to how Spotify stores metadata.

1.2 Baseline Properties of the Million Playlist Dataset

Baseline statistics of the MPD were provided in [1]. The values underscored several properties of the MPD reporting the average playlist length (or tracks) as 66.35; 17,381 unique normalized playlist titles; 92,944 unique playlist titles; 295,860 unique artists; 734,684 unique albums; 2,262,292 unique tracks; 66,346,428 tracks; which made up the 1,000,000 playlists [1].

2. Background

Music recommendation in the RecSys Challenge focuses on the task of playlist continuation when given a set of seed tracks with a playlist name [1]. For the Spotify MPD RecSys Challenge, the task included the call to "…develop a system for the task of automatic playlist continuation. Given a set of playlist features, participants' systems shall generate a list of recommended tracks that can be added to that playlist, thereby 'continuing' the playlist" [4]. Metrics for the challenge included traditional information retrieval metrics, such as a precision metric, together with two additional metrics that were unique to the Spotify Recommendation Challenge. These are delineated in Table 1.

Metric	Definition
R-precision	R-precision is the number of relevant tracks retrieved divided by the number of relevant tracks known (i.e., the number of tracks withheld).
Normalized discounted cumulative gain (NDCG)	Discounted cumulative gain (DCG) measures the ranking quality of the tracks recommended, and increases when relevant tracks are placed higher in the list. Normalized DCG (NDCG) is determined by calculating the DCG and dividing it by the ideal DCG in which the recommended tracks are ranked perfectly.
Recommended Songs clicks	Recommended Songs clicks is the number of refreshes needed before a relevant track is encountered.

Table 1: Metrics of the RecSys Challenge 2018 [10].

The challenge area is described more precisely as a challenge to complete ten different types of playlists using the MPD as training data. In the ten focus areas, the recommender system must generate 500 recommended tracks in decreasing order of confidence when provided a variation in either a) seed tracks and b) playlist name or c) just a playlist name [1]. One of the most challenging aspects of this challenge for our team was to improve our baseline metrics for playlists when only a playlist name is provided with no seed tracks.

The challenge allowed for both non-academia (industry researchers) and academia researchers to participate. Only the research teams from academia (verified by our @illinois.edu email addresses, presumably) such as the Minrva team, were able to access the Spotify MPD in its entirety [10]. Non-academia or industry researchers were able to join a majority academic research team and have access to a subset of the Spotify MPD. These steps were taken to better ensure user privacy [10].

3. Methods

As our overall goal was to enrich the MPD, we sought to use a baseline entry in the main track as a comparison to the reconciled MPD entry in the creative track. The creative track is the portion of the RecSys Challenge that allows integration of external datasets to improve recommendations. The schema used in our model are described in section 3.4, model training, but we review first several machine learning methods of our recommendation engine (section 3.1) together with data transformation and data enrichment processing and tools (sections 3.2 and 3.3 respectively).

3.1 Recommendation Engine: Apache PredictionIO

The recommendation engine used the Similar Product Template [11] in Apache PredictionIO. The Similar Product Template was designed to perform machine learning with the Alternating Least Squares (ALS) algorithm, as implemented in Spark MLlib. Matrix factorization techniques, the basis for the Minrva team's algorithm, were proven successful as a recommender algorithm for the Netflix prize [12]. The data load scripts used to import items, users, and views are available from the Minrva team's public GitHub repository [13].

The PredictionIO system provides several bootstrapping advantages for participating in the RecSys 2018 Challenge. The Apache project webpage lists the advantages of the PredictionIO framework on the PredictionIO website as an "…open source Machine Learning Server built on top of a state-of-the-art open source stack for developers and data scientists to create predictive engines for any machine learning task. It lets you quickly build and deploy an engine as a web service on production with customizable templates" [14]. The open source stack is a "full machine learning stack," which includes the following interlocking set of tools as a framework for recommendation processing: Apache Spark, MLlib, HBase, Spray, and Elasticsearch [14].

3.2 Data Transform Tools: OpenRefine

OpenRefine is a fundamental tool for data transformation and reconciliation tasks in targeted areas of enrichment [15]. OpenRefine generally is marketed as a data cleaning tool to use with "messy data"—however, we found that its transformation and enrichment functionality were critical to our approach. As an example of its transformation affordances, the OpenRefine project helped us to format the MPD corpus from a set of JSON files into CSV files by a simple import/export command from the web app interface. The need for CSV became apparent to us, as we intended to store data in a relational model using PostgreSQL and use relational modeling to create associations in the dataset using external data. The relational model has been a profound bedrock of computing since the advent of the modern relational database era. Therefore, one of the team's first processing tasks was loading the MPD into a relational model that could be enriched and transformed further. The PostgreSQL database proved to be suited well to this approach and provided data persistence overall in the recommendation system. The Minrva team focused on the artist name with which to reconcile to related artists using VIAF. The plugin to OpenRefine that made this reconciliation possible was the open source project Conciliator, which is available on GitHub to use within OpenRefine [16].

3.3 Data Enrichment: Conciliator

According to the Conciliator project webpage, the software is, "…a growing collection of OpenRefine reconciliation services, as well as a Java framework for creating them. A reconciliation service tries to match variant text (usually names of things) to standard IDs for the entity represented by that text." VIAF reconciliation was undertaken through Conciliator functionality that we used on our server together with OpenRefine, which also ran on our development server for reconciliation. The Conciliator tool is packaged in such a way that it can be run as a standalone Spring project on our system.

Conciliator was used specifically to generate new rows of data for entities that were related to the free text of a named artist. This is a several step process in which the Conciliator tool looks for a name match in the VIAF dataset first, and then looks for related artists. When a named entity, such as the artist, can be matched in VIAF, the best match (a high confidence score) of a related artist is associated in our dataset. One of the motivations for doing this with the free text for name artists stems from the need to create additional connections among tracks. The more connections among the data we could provide, the better our model training process. Furthermore, and perhaps more critically for machine learning, it was our team's observation that controlled identifiers were available for musical tracks, the album, and the artist. Like a controlled vocabulary, controlled identifiers are a preferred element in model training, as they will be less noisy for a machine learning process, and in theory, are assigned uniformly while adhering to a standard syntax and semantics. Other cultural heritage projects have suggested controlled vocabulary reconciliation to achieve greater value from existing named entities [17]. We note that controlled identifiers and vocabularies stand in stark contrast to the free text elements in the MPD. Our team theorized that these data points may produce unintended noise for machine learning model development when used in combination with other free text from the project, including the playlist name.

We note also the unique challenges posed by playlist names, which are valuable free text, but contain no corresponding controlled id. The data fields and schema used to store data from the MPD challenge are summarized below, in which Table 2 is the main track schema with entities, while Table 3 delineates the schema for the creative track entry. Any entity without a trailing _uri represents a potential noisy data point for the model during training, as it is an uncontrolled vocabulary and may represent free text or other words

that the model will have to interpret during training. Note that in the main track, we used most of the relevant entities available, including all the non-controlled vocabulary. In contrast, in the creative track, we attempted to minimize the amount of textual data, while at the same time including the enriched, reconciled related artists vocabulary.

track_uri	modified_at, playlist_name, album_uri, artist_uri, track_name, album_name, artist_name

Table 2: Main Track Schema [18].

track_uri	playlist_name, album_uri, artist_uri, artist_name, related_artist

Table 3: Creative Track Schema [18].

3.4 Model Training: Automating Playlist Continuation with ALS Algorithms

PreditictionIO's similar product template implements the Alternating Least Squares algorithm and can use the implicit training set of users who have viewed (included in a playlist) products (playlist tracks). Once the datasets have been loaded into a persistent data store, such as PostgreSQL, the engine is trained with a simple train command. Spark uses as many CPU cores as the cluster on which it is running, and depending on the size of the dataset, may require large amounts of RAM to complete the training. Therefore, model training was completed using two different types of system resources with varying CPUs, RAM, and storage depending on the challenge track. The maximum amount of system RAM described below was used for each respective model training process. For the Spotify corpus of over two million items, training time was approximately 2-4 hours. For the main track, system resources included:

- Ubuntu 18.04 LTS (GNU/Linux 4.15.0-20-generic x86_64)
- java-8-openjdk-amd64
- 128GB RAM, 24 CPU Cores, 2.5 TB SSD
- psql (10.4 (Ubuntu 10.4-0ubuntu0.18.04))

The creative track system was comprised of the following resources:

- Ubuntu 18.04 LTS (GNU/Linux 4.15.0-20-generic x86_64)
- java-8-openjdk-amd64
- 300GB RAM, 16 CPU Cores, 340GB SSD
- psql (10.4 (Ubuntu 10.4-0ubuntu0.18.04))

3.5 Playlists without Seed Tracks

As noted earlier, we employed two different strategies to generate recommendations when a challenge set included a playlist name with no seed tracks. In the main track, we simply used the top 30 tracks from the training data paired with the playlist name to generate recommendations. For the creative track, we used a reverse indexing approach to infer likely tracks from the data available via a playlist name. Tracks from the training data that had used the same or a similar playlist name as the challenge set were identified; then, those tracks were used to generate recommendations. The system used the playlist name as a category in the query pattern described above. Note that challenge data were used to create approximately half of this reverse index, as the creative track allowed the use of public datasets.

4. Results

After training the model using the imported data described in Tables 2 and 3, the system was then able to make recommendations for playlist completion. Throughout the duration of the RecSys challenge, submissions were evaluated against half of the challenge set, and for this portion of the challenge, the ALS-based recommendation system earned the following scores.

Metric	Minrva Score
R-precision	0.056733
NDCG	0.085486
Clicks	14.0374

Table 4: Baseline/Main Track Challenge Results [19].

Metric	Minrva Score
R-precision	0.058689
NDCG	0.090218
Clicks	13.263

Table 5: Focus area/Creative Track Challenge Results [20].

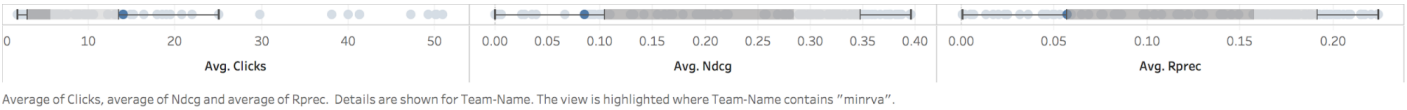

Figure 1. Average aggregate scores for all teams in the Main Track [19].
(click to enlarge)

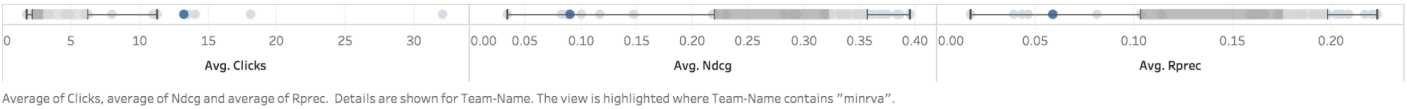

Figure 2. Average aggregate scores for all teams in the Creative Track [20].
(click to enlarge)

5. Discussion and conclusion

Overall, the Minrva team's entry exhibited improvement from the baseline entry in the main track compared to the enriched MPD with the VIAF artist name associations in the creative track. This finding indicates that creative approaches to playlist continuation with external datasets hold promise for improved retrieval. The single algorithm employed for this task contributed to the lower ranking in the challenge overall, as the Minrva team's entry did not reach the top 10 team entries and at the end of the challenge was in the lower tier of all entries. The main track had 113 final team submissions, and the creative track had 33 team submissions [1].

Revisiting our original hypothesis with respect to machine learning software and tools, it seems that considering algorithm development and data processing equally may have yielded better results overall. Enriching data with external data sources can be time consuming if the data endpoints are accessed by web services. This is attributable largely to request limitations. There were additional areas of enrichment work that were identified near the end of the challenge, in that our team became aware of a wealth of resources available as OpenRefine add-ons for a variety of reconciliation tasks related to linked data [21]. The Minrva team is interested in pursuing future data enrichment work to supplement the VIAF data that was integrated into our dataset. To this end, the team will explore reconciliation processing with WikiData. WikiData model exploration and experimentation through OpenRefine is time consuming because it uses web-based end-points for reconciliation. However, there are graph traversing procedures that may prove quite valuable for linked data in WikiData corpus reconciliation [22]. Text mining approaches [23] in playlist continuation were underdeveloped in our project, and further development of the reverse index of playlist names is an area for future research. Identifying the tools and corresponding data processing steps took precedence over algorithm development for this entry, which resulted in a lower ranking overall.

Integration of two-stage machine learning and ensemble approaches would be valuable and desirable in the next step of our research for the Apache PredictionIO framework for playlist continuation. A RecSys 2018 paper summarized the methodologies from the top 10 scoring teams in the creative and main tracks and found approaches that utilized ensemble entries were markers of high performing teams noting — ".. most approaches ensemble the results obtained by several well-known methods, including matrix factorization models, neighborhood-based collaborative filtering models, basic information retrieval techniques, and learning to rank models. The results show that the models work best when a sufficient number of tracks per playlist is provided and they are randomly selected from the playlist (as opposed to the sequential order from the beginning of the playlist)" [24]. The winning (top scoring) entries generally had in common a strategy that employed two-stages machine learning. We note that several teams in the top 10 entries incorporated two-stage machine learning with matrix factorization of the type used in our system and that collaborative filtering, when used in ensemble approaches, was a viable part of high scoring playlist continuation methodologies.

A final unique contribution to the challenge we note from our work is that in an analysis of the top 10 scoring teams, the creative track scores did not outperform the main track scores [25]. Our results indicated the opposite — our final creative track scores with VIAF reconciliation of related artist names provided an improved score in all metrics as compared to the baseline main track entries. We emphasize this finding to advance our research concern with reconciliation-based recommendation approaches. The method improves baseline scores and add valuable context to recommender systems. Data reconciliation processes given equal importance in ensemble machine learning could support contextual richness overall in providing recommendations. Therefore, a future step of

our recommender system development is to implement two-stage machine learning paired with data reconciliation processes, which our results show is a vital component for information retrieval metrics in general and machine learning based automated playlist continuation specifically.

Acknowledgements

The author sincerely appreciates the research contributions of software developers Nathaniel Ryckman & Benjamin Ryckman.

Further reading

Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys '18): https://dl.acm.org/citation.cfm?id=3267471

About the author

Jim Hahn is an Associate Professor in the Undergraduate Library at the University of Illinois (Urbana-Champaign, Urbana, IL USA). His research into technology-enhanced learning has led to many software development projects within library settings and provides unique insights into new student's expectations and needs and helps inform the work that he does as the Orientation Services and Environments Librarian for undergraduate students at the University of Illinois. He founded and manages the Minrva project (https://minrvaproject.org) and currently serves as project PI for a software development grant funded through the EBSCO/FOLIO innovation challenge, the aims of which are focused on incorporating portions of the Minrva mobile app Wayfinding utility functions into the open source FOLIO codebase. Email: jimhahn@illinois.edu

Notes

[1] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. 2018. RecSys Challenge 2018: Automatic Music Playlist Continuation. In Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3240323.3240342

[2] Peter Romov and Evgeny Sokolov. 2015. RecSys Challenge 2015: Ensemble learning with categorical features. In Proceedings of the 2015 International ACM Recommender Systems Challenge (RecSys '15 Challenge). doi: 10.1145/2813448.2813510

[3] Marco Tulio Ribeiro, Anisio Lacerda, Adriano Veloso, and Nivio Ziviani. 2012. Pareto-efficient hybridization for multi-objective recommender systems. In Proceedings of the 6th ACM conference on Recommender systems (RecSys'12), 19-26. doi: 10.1145/2365952.2365962

[4] Million Playlist Dataset, official website hosted at https://recsys-challenge.spotify.com/

[5] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahhi. 2018. Current challenges and visions in music recommender systems. International Journal of Multimedia Information Retrieval, 7, 2, 95-116. doi: 10.1007/s13735-018-0154-2

[6] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. 2016. Research paper recommender systems: A literature survey. International Journal on Digital Libraries, 17, 4, 1–34. doi: 10.1007/s00799-015-0156-0

[7] VIAF (Virtual International Authority File) 2018. https://viaf.org/

[8] Samuelle Carlson and Ben Anderson. 2007. What are data? The many kinds of data and their implications for data re-use. Journal of Computer-Mediated Communication, 12, 635-651. doi: 10.1111/j.1083-6101.2007.00342.x

[9] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for Implicit Feedback Datasets. In 2008 Eighth IEEE International Conference on Data Mining, 263-272. doi: 10.1109/ICDM.2008.22

[10] Spotify RecSys Challenge. 2018. Challenge Rules. https://recsys-challenge.spotify.com/rules

[11] PredictionIO similar product engine template (Scala-based parallelized engine). https://github.com/apache/predictionio-template-similar-product

[12] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. Computer, 42, 8, 30-37. doi: 10.1109/MC.2009.263

[13] jimfhahn/RecSys-Challenge-2018-creative-track. 2018. https://github.com/jimfhahn/RecSys-Challenge-2018-creative-track/tree/master/load-data

[14] PredictionIO. 2018. http://predictionio.apache.org/

[15] OpenRefine. 2018. http://openrefine.org/

[16] codeforkjeff/conciliator. 2018. https://github.com/codeforkjeff/conciliator

[17] Seth van Hooland, Ruben Verborgh, Max De Wilde, Johannes Hercher, Erik Mannens, and Rik Van de Walle. 2013. Evaluating the success of vocabulary reconciliation for cultural heritage collections. Journal of the American Society for Information Science and Technology, 64, 3, 464-479. doi: 10.1002/asi.22763

[18] See full data load details for the main track entry here: https://github.com/jimfhahn/RecSys-Challenge-2018-main-track/blob/master/load-data/import-items.py and the data load details for the creative track entry here: https://github.com/jimfhahn/RecSys-Challenge-2018-creative-track/blob/master/load-data/import-items.py

[19] Main Track Ranking. 2018. https://recsys-challenge.spotify.com/leaderboard

[20] Creative Track Ranking. 2018.https://recsys-challenge.spotify.com/leaderboard/creative

[21] Christina Harlow. 2015. Data munging tools in preparation for RDF: Catmandu and LODRefine. Code4Lib Journal, 30. https://journal.code4lib.org/articles/11013

[22] Wikidata:Tools/OpenRefine. 2018. https://www.wikidata.org/wiki/Wikidata:Tools/OpenRefine

[23] Andreu Vall, Hamid Eghbal-zadeh, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. 2017. Music playlist continuation by learning from hand-curated examples and song features. In Proceedings of DLRS 2017, Como, Italy. doi: 10.1145/3125486.3125494

[24] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. 2018. An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. https://arxiv.org/pdf/1810.01520.pdf