Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра Системного Программирования

Соболев Артём Александрович

Разработка и анализ системы формирования плейлистов музыки

Курсовая работа

Научный руководитель:
Данильченко А.

Санкт-Петербург
2014

Оглавление

Введение

С распространением широкого доступа к сети интернет появилось множество сервисов, предлагающих персонализированное обслуживание. Такой подход позволяет лучше учесть интересы пользователя и порекомендовать ему наиболее интересные товары или услуги.

В области музыкальных сервисов задачей является составление плейлиста, релевантного интересам пользователя. При этом хотелось бы учесть, в том числе, и настроение пользователя, как и контекст, задаваемый текущим треком.

Одним из подходов к персонализации и составлению персонального списка воспроизведения являются рекомендательные системы. Как и в любом другом разделе машинного обучения, сначала строится математическая модель, а потом производится её "обучение", т.е. настройка параметров по данным.

В случае рекомендации музыки данными является история прослушиваний пользователя, а также отношения между треками. Интересными нам могут являться отношения трек-исполнитель, трек-жанр, трек-альбом и подобные. Учёт такой таксономии позволяет нам не только обогатить нашу модель чуть более общими взаимосвязями, но и частично побороть некоторые трудности рекомендательных систем.

Большой проблемой рекомендательных систем является проблема "холодного старта". Как правило, для новых объектов (треков в нашем случае) мы не располагаем какими-либо данными об их потреблении, что затрудняет их получение, ведь мы не можем побудить пользователя обратить на них своё внимание.

Таксономия позволяет решить данную проблему в некоторых случаях. Так, мы можем оценить заинтересованность пользователя в определённом исполнителе, жанре или альбоме и предлагать ему похожие треки, даже если никто их никогда не слушал.

Ещё одним подходом к проблеме "холодного старта" является перенос знаний. Даже если мы не обладаем какой-либо историей, мы можем попробовать воспользоваться уже существующими данными. Например, в случае музыкальных рекомендаций можно использовать данные тематических интернет радио-станций.

1. Постановка задачи

Задачей данной работы является реализация существующей [3] модели музыкальных рекомендаций, её модификация и сравнение результатов.

1. Реализовать модель из статьи
2. Модифицировать модель
3. Подобрать параметры модели
4. Оценить обе модели
5. Провести сравнение полученных результатов

В рассматриваемой статье [3] упомянуто 2 модели: station-based и station-less. Согласно статье, вторая модель демонстрирует лучшие результаты, поэтому было решено взять её за основу.

2. Базовая модель

В статье исследователей из Yahoo [3] описывается использование интернет-радиостанций для "раскрутки" рекомендательной системы. Модель строится вокруг вероятностного распределения на треках для станций и учитывает как отношение трек-автор, так и контекст, задаваемый предыдущими треками.

2.1. Матричное разложение

Модель основывается на идее матричного разложения [4]. Обозначим за s какого-либо пользователя (в нашем случае это станция), а i - рекомендуемый объект (в нашем случае – треки). Введём понятие соответствия пользователя и объекта:

$$r_{si} = \mu + b_i + b_s + p_s^T q_i$$

Где $b_i, b_s \in \mathbb{R}$ - скалярные поправки для объектов и пользователей, а $p_s, q_i \in \mathbb{R}^{dim}$ – латентные вектора, характеризующие пользователей и объекты, чьё скалярное произведение служит мерой сходства. Данные коэффициенты подбираются во время обучения, а dim является гиперпараметром модели, задаваемым до обучения.

Основываясь на данном понятии, введём распределение вероятности на треках для пользователя s:

$$P_s(t) = \frac{\exp(r_{st})}{\sum_j \exp(r_{sj})}$$

2.2. Stationless модель

В предположении тематических станций естественно предположить, что треки одной и той же радиостанции довольно похожи, т.е. станция описывается средним вектором её треков. Тогда мы получаем stationless модель, для которой сходство записывается как

$$r_{si} = b_i + q_i^{(1)} \cdot \Big(\frac{1}{\sqrt{|P_s|}} \sum_{j \in P_s} q_j^{(2)} \Big)$$

Здесь P_s – список воспроизведения станции s, а верхний индекс у латентного вектора означает роль вектора: $q_j^{(2)}$ используется для описания профиля станции.

2.3. Временной контекст

Контекстом в stationless модели является средний профиль треков, проигранных за последние w минут. Это число также является гиперпараметром системы и задаётся до обучения. В нашей реализации было выбрано 30 минут.

Теперь отношение сходства является также и функцией текущего времени.

$$r_{si;t} = b_i + q_i^{(1)} \cdot \Big(\frac{1}{\sqrt{|P_s|}} \sum_{j \in P_s} q_j^{(2)} + \frac{1}{\sqrt{|P_{s;[t-w,t)}|}} \sum_{j \in P_{s;[t-w,t)}} q_j^{(3)} \Big)$$

2.4. Отношение трек-исполнитель

Для моделирования таксономии каждый параметр трека представляется суммой собственного параметра трека и соответствующего параметра его исполнителя.

$$b_i = b_{t(i)} + b_{a(i)}$$

$$q_i = q_{t(i)}^{(k)} + q_{a(i)}^{(k)}$$

Тем самым, для новых треков мы можем использовать параметры их исполнителя.

2.5. Обучение

Обучение производится методом максимального правдоподобия.

$$L(\{b\}, \{q^{(1)}, q^{(2)}, q^{(3)}\}) = \prod_{s \in S} \prod_{(i,t) \in P_s} \frac{\exp(r_{si;t})}{\sum_{j \in I} \exp(r_{sj;t})}$$

На практике проще работать с логарифмом функции правдоподобия

$$LL(\{b\}, \{q^{(1)}, q^{(2)}, q^{(3)}\}) = \sum_{s \in S} \sum_{(i,t) \in P_s} \ln \frac{\exp(r_{si;t})}{\sum_{j \in I} \exp(r_{sj;t})}$$

Для оптимизации мы будем использовать метод стохастического градиентного спуска. Однако, на этом пути имеется проблема: градиент оптимизируемой функции

$$\frac{\partial}{\partial \theta} \Big(\ln \frac{\exp(r_{si;t})}{\sum_{j \in I} \exp(r_{sj;t})} \Big) = \frac{\partial r_{si;t}}{\partial \theta} - \sum_{j \in I} P_{s;t}(j) \frac{\partial r_{sj;t}}{\partial \theta}$$

включает в себя вычисление среднего градиента сходства по всем имеющимся трекам, что является вычислительно затратной задачей. Для преодоления этой проблемы используется техника Importance Sampling [2], заключающаяся в приближении целевого распределения аналогичным с меньшим носителем.

Другой проблемой является численная неустойчивость операции exp. Для её решения мы обрезаем все вектора при выходе за границы отрезка $[-1, 1]$.

2.6. Регуляризация

В целях контроля сложности модели мы вводим штраф за отклонение латентных параметров от нуля с некоторым весом:

$$LL(\{b\}, \{q^{(1)}, q^{(2)}, q^{(3)}\}) = \sum_{s \in S} \sum_{(i,t) \in P_s} \ln \frac{\exp(r_{si;t})}{\sum_{j \in I} \exp(r_{sj;t})} - \lambda \sum_{k \in \{1,2,3\}} \sum_{i \in I} (\|q_i^{(k)}\|_2^2)$$

3. Модификация

Основной задачей данной работы являлась проверка работоспособности иной схемы таксономизации. Вместо задания специальных векторов, представляющих исполнителей, мы задаём априорное нормальное распределение, штрафующее вектора треков за отклонение от заданного среднего:

$$q_i^{(k)} \sim \mathcal{N}(p_{a(i)}, \hat{\alpha} I)$$

$$b_i \sim \mathcal{N}(c_{a(i)}, \hat{\alpha})$$

где $p_{a(i)}$ - среднее значение латентных векторов для исполнителя трека i.

Это приводит нас к следующей целевой функции:

$$LL(\{b\}, \{q^{(1)}, q^{(2)}, q^{(3)}\}) = \sum_{s \in S} \sum_{(i,t) \in P_s} \left(\ln \frac{\exp(r_{si;t})}{\sum_{j \in I} \exp(r_{sj;t})} - \alpha \|b_i - c_{a(i)}\|_2^2 - \alpha \sum_{k \in \{1,2,3\}} \|q_i^{(k)} - p_{a(i)}^{(k)}\|_2^2 \right)$$

Для оптимизации данной функции мы так же используем стохастический градиентный спуск, однако, поскольку нам необходимо поддерживать вектора $p_{a(i)}$ средними векторами треков соответствующего исполнителя, мы вводим дополнительный предварительный шаг, заключающийся в вычислении данных векторов на каждой итерации.

4. Реализация и подбор параметров

Основной вычислительный модуль был реализован на C++ в целях максимальной производительности. По сравнению с наивной реализацией, было произведено несколько кеширующих оптимизаций, позволивших значительно ускорить процесс по сравнению с оригинальной работой.

Частой проблемой при реализации алгоритмов машинного обучения являются вычислительные ошибки вроде неправильно вычисленного градиента. Для предотвращения подобных ошибок (особенно на стадии оптимизации) было

4.1. Подбор параметров

Наша модель (как оригинальная, так и модифицированная) обладает несколькими свободными параметрами, который исследователь должен задать сам. Для подбора этих параметров нами был выделен отдельный тестовый набор и произведён поиск по сетке.

Поскольку приложение является CPU-bound, не имеет большого смысла в запуске большего количества процессов, чем доступно ядер. Однако, в нашем распоряжении оказалось несколько компьютеров с суммарно 10 ядрами, чем было решено воспользоваться. Для этого использовалась программа GNU Parallel [1], позволяющая запускать задачи параллельно на нескольких машинах при наличие ssh-доступа к ним. Все необходимые данные были упакованы в shar архивы, представляющие из себя самораспаковывающиеся архивы вместе с некоторым дополнительным кодом для запуска процессов.

5. Тестирование

Для оценки результатов была выбрана метрика AUC@k, чей алгоритм вычисления выглядит следующим образом:

1. Берётся случайный список воспроизведения
2. Из него изымается последний трек
3. Выбирается случайные k треков, не присутствующих в данном списке
4. Полученные $k + 1$ ранжируются в соответствие с вероятностью их появления
5. Результатом считается доля верно отранжированных пар

Для получения более стабильных результатов мы считаем эту метрику на нескольких плейлистах, а результат усредняем. Также, для того, чтобы убедиться, что наша модель способна отражать действительно важные взаимосвязи, а не только выделять наиболее популярные треки, мы выбираем k треков не с равномерным распределением, а с вероятностью, пропорциональной популярности трека.

5.1. Наборы данных

Для сравнения результатов нами был использован оригинальный набор данных, полученный исследователями Yahoo путём обхода нескольких тысяч наиболее популярных интернет-радиостанций. К нему были добавлены два других набора, полученных из истории воспроизведений на радиостанциях Санкт-Петербурга и Москвы: Piter.FM и Moskva.FM, соответственно.

Набор данных для обучения

	Плейлистов	Воспроизведений	Треков	Исполнителей
Piter.FM	9 813	2 401 227	72 406	2 1689
Moska.FM	52 028	12 732 422	139 046	34 059
Yahoo R9	2 651	5 927 199	164 256	54 474

Для оценки результатов часть исходных списков воспроизведения была выделена в отдельный, тестовый набор данных.

6. Результаты

Модель была обучена на нескольких наборах данных, результаты представлены в таблице:

	Оригинал	Модификация
Piter.FM	0.77521	**0.80827**
Moska.FM	**0.84987**	0.79064
Yahoo R9	0.82065	**0.85725**

Для сравнения двух моделей использовался критерий Уилкоксона, показавший статистическую значимость всех предъявленных результатов.

Для наглядности была обучена двухмерная модель, на которой были выделены треки только трёх наиболее популярных исполнителей.

Видно, что точки группируются в заметные кластеры, а центроиды более-менее рассеяны.

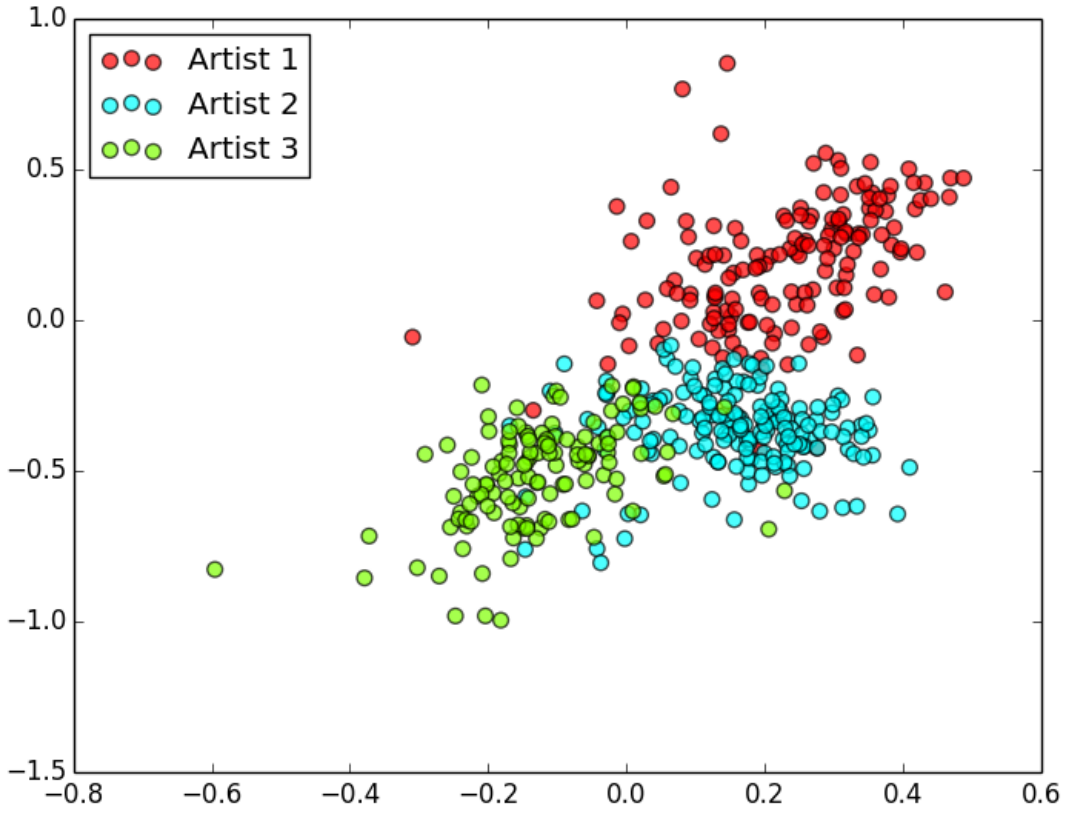

Рис. 1: Треки трёх наиболее популярных исполнителей

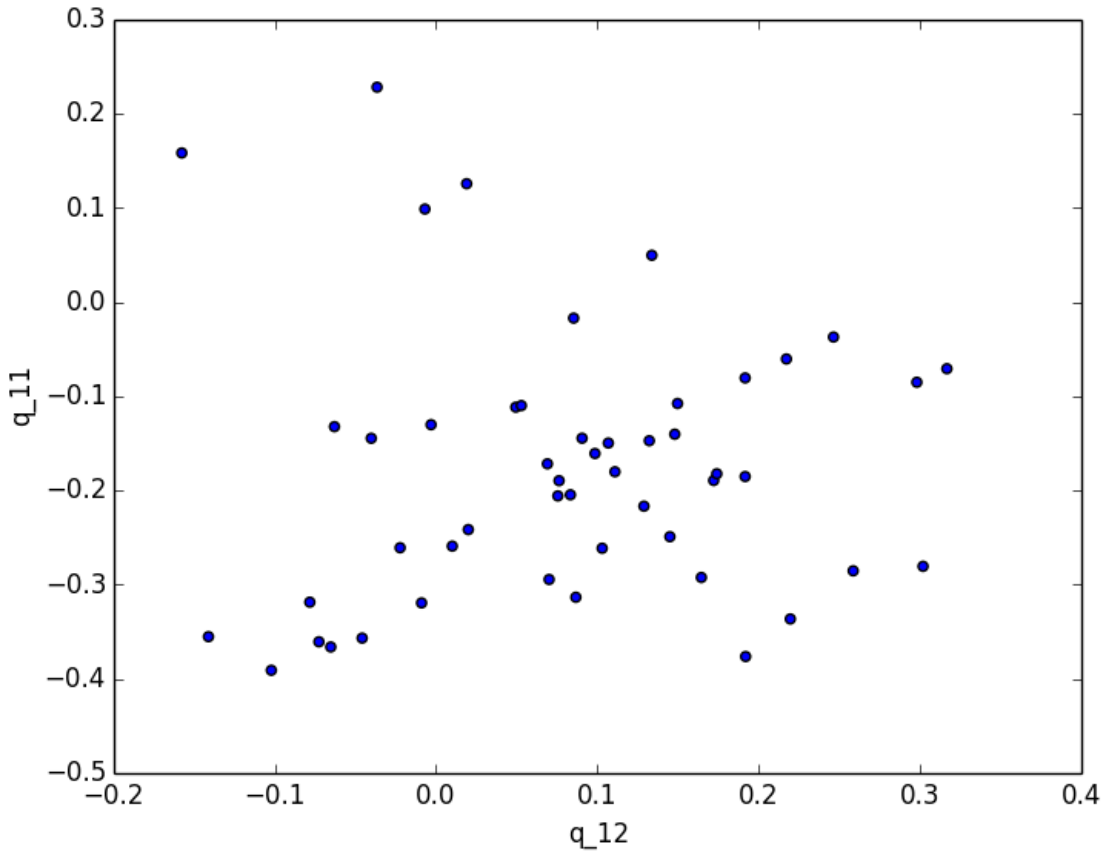

Рис. 2: Центроиды 50 наиболее популярных исполнителей

Заключение

В ходе данной работы было исследовано множество публикаций, реализована и проанализирована рекомендательная система в контексте музыкальных рекомендаций. Рассмотренная модификация показала статистически значимое превосходство в некоторых случаях, однако, на других данных она уступает оригинальной модели. Полученные результаты являются интересными, но всё же требуют дальнейшей работы.

Список литературы

[1] GNU Parallel - GNU Project - Free Software Foundation // GNU Parallel. — http://www.gnu.org/software/parallel/.

[2] Murphy Kevin P. Machine Learning: a Probabilistic Perspective. — MIT Press, 2012.

[3] O. Somekh N. Aizenberg Y. Koren. Build Your Own Music Recommender by Modeling Internet Radio Streams. — 2012.

[4] Y. Koren R. M. Bell, Volinsky C. Matrix factorization techniques for recommender systems. — 2009.