

Генерация плейлиста по выбранным трекам

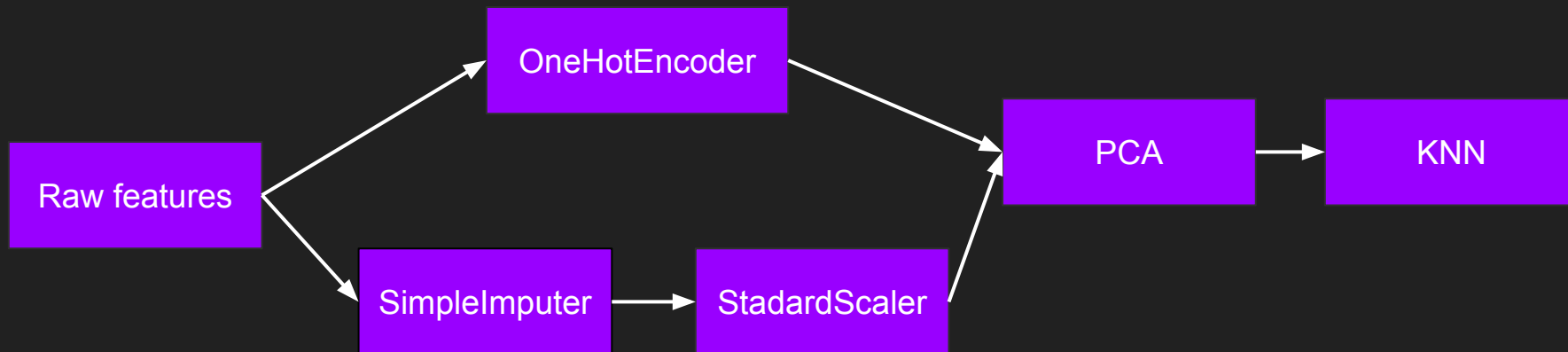
Чекпоинт 5

Костров Вячеслав
Ганыч Даниил
Сараев Никита

Исходное состояние ML части

В исходном виде ML часть проекта представляла собой:

1. Пайплайн предобработки мета-информации: OneHotEncoding категориальных фичей, нормировка числовых фичей и отбор 141 финальных признаков методом главных компонент.
2. Поиск ближайших соседей на предобработанных признаках.



Классические ML подходы

Исходя из специфики решаемой нами задачи, число подходящих методов классического машинного обучения крайне ограничено и, по сути, сводится к:

1. Поиску соседей (KNN, ANN и т.д).
2. Решению задачи ранжирования, где входной трек - документ, а все наши треки - база документов (LambdaRank, YetiRank и т.д)

В силу отсутствия разметки в формате трек-рекомендации в общем доступе, реализация второго класса моделей становится невозможной, так как прибегнуть к помощи ассессоров мы не можем. Первый же класс моделей уже был протестирован и реализован в проекте.

Основные направления работы

Изучение дополнительных источников данных

Необходимо оценить возможность и ресурсозатраты для сбора дополнительной разметки или обогащения существующих данных.

Эксперименты с Metric Learning подходами

Провести эксперименты с популярными подходами в Metric Learning:

1. Парные лоссы: TripletLoss
2. Классификационные лоссы: ArcFace

Извлечение эмбеддингов из mp3 формата

Провести сравнение существующих подходов к получению аудио-эмбеддингов.

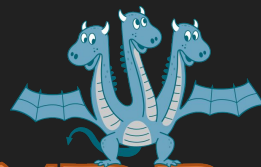
Оценить применимость к нашей задаче, обращая особое внимание на скорость и качество работы.

Сетап для экспериментов

Зачем: эксперименты с Deep Learning моделями часто имеют множество степеней свободы (структура самих моделей, большое количество лоссов, learning rate и др.), поэтому мы сразу решили придумать структуру для хранения и логирования всех экспериментов.

Что сделали:

- в репозитории создали директорию **research**, в которой храним реализации всех моделей;
- настроили использование конфигов с помощью **Hydra**;
- настроили использование **Lightning** в связке с **PyTorch** для поддержания общей структуры кода;
- создали S3 бакет для MLflow, который подняли на отдельной VM в YC для логирования всех экспериментов;
- реализовали CLI интерфейс для запуска экспериментов, описали процесс в **research/README.md**.



HYDRA

mlflow

Сетап для экспериментов

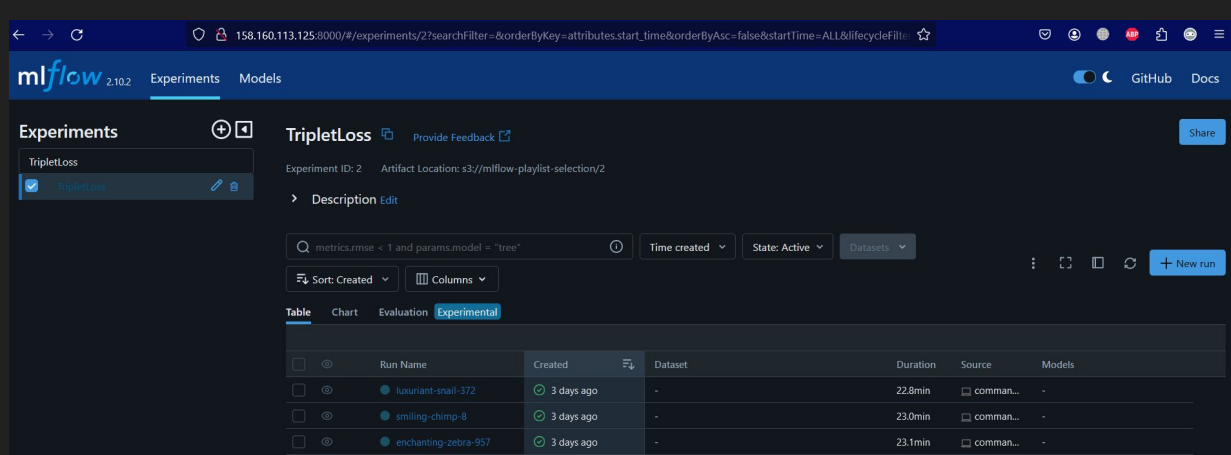


Рис 1. Интерфейс mlflow

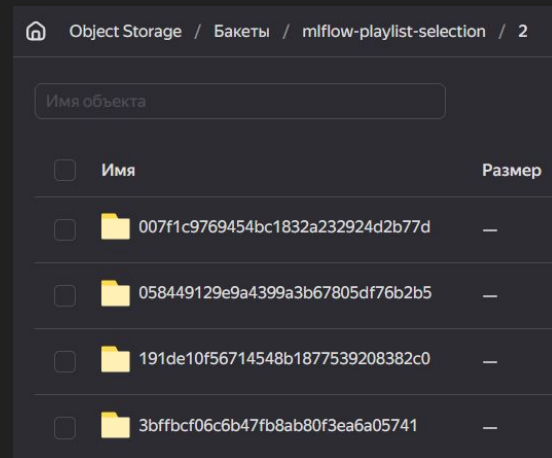


Рис 2. Бакет в YC S3

Эмбеддинги

Мотивация для исследования:

Полученные эмбеддинги других моделей можно использовать в нашей модели с кастомным Metric Learning лоссом.

Какие модели рассматривались:

Первоначально в исследование входили модели, которые умеют работать с сырыми .wav аудио файлами. Важным условием было наличие предобученной модели и возможность простого получения эмбеддингов без дополнительного переобучения.

По такой логике мы не рассматривали модели [VGGish](#), [Earworm](#) и [CLMR](#), которые потенциально были нам интересны, но не имели готовой модели.

В данной итерации исследования эмбеддингов были рассмотрены модели [EnCodec](#), [Wav2Clip](#), [Wav2Vec2](#) и [Wav2Vec2-BERT](#).

Сравнение эмбеддингов

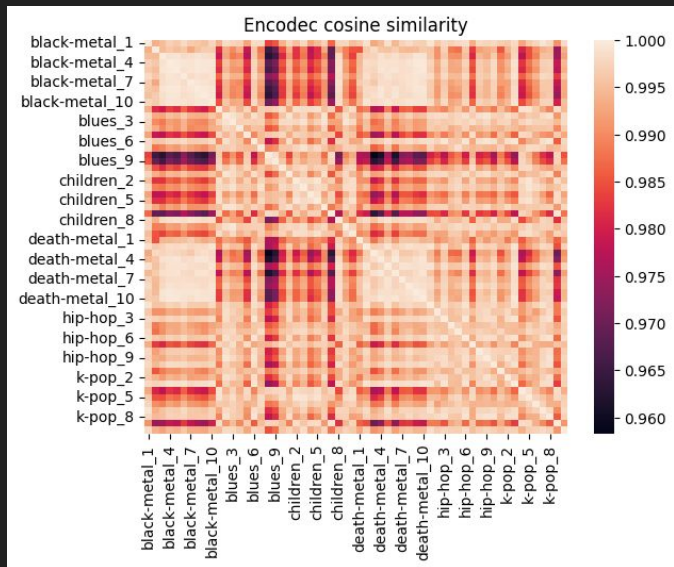
Название модели	Описание модели	Основные параметры	Скорость обработки 30 сек	Размер выходного эмбеддинга
EnCodec	Аудиокодек для сжатия .wav файлов без потери качества.	<ul style="list-style-type: none">• sample_rate - в каких частотах будет обрабатываться аудиодорожка в Гц - 24к или 48к (в зависимости от выбора модели) .• channels - параметр для определения количества каналов аудио - 1 для моно и 2 для стерео. Также как и sample_rate влияет на длину эмбеддинга.• target_bandwidth - целевая пропускная способность дорожки (измеряется в kbps). Обычно используется 6.0 .	В среднем 7.87 sec	[128, n_codes] n_codes - количество фреймов в 1 секунде. Для 30 секунд вышло [128, 2250]
Wav2Clip	Нейронная сеть для генерации изображений по звуку.	Предобученная модель не требует дополнительной спецификации параметров.	В среднем 2.28 sec	[n_channels, 512] n_channels - количество каналов аудио файла (нужен pooling для финального использования)

Сравнение эмбеддингов

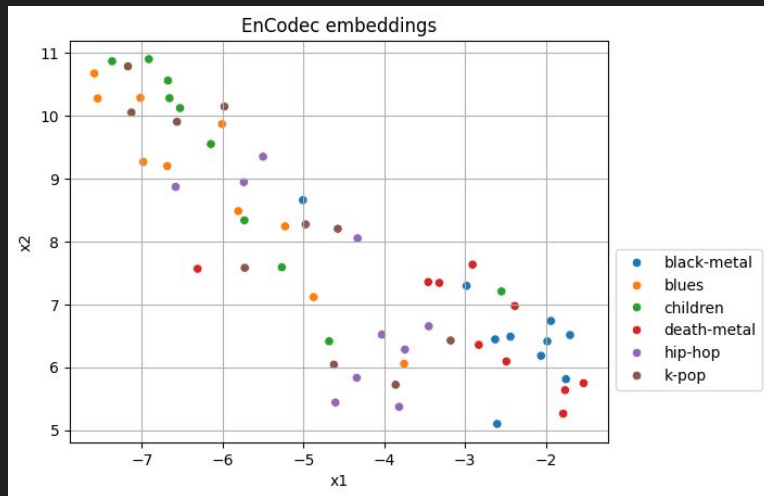
Название модели	Описание модели	Основные параметры	Скорость обработки 30 сек	Размер выходного эмбеддинга
Wav2Vec2	Нейронная сеть для распознавания речи, использовалась <i>facebook/wav2vec2-large-xlsr-53</i>	<ul style="list-style-type: none">• feature_size - размер исходящего эмбеддинга.• sampling_rate - тоже, что и sample_rate (модель обучена на аудио в 16кГц).	В среднем 16.1 sec	[feature_size, 1499, 512]
Wav2Vec2-BERT	Нейронная сеть для распознавания речи с использованием BERT, использовалась <i>hf-audio/wav2vec2-bert-CV16-en</i>	<ul style="list-style-type: none">• feature_size - размер исходящего эмбеддинга.• sampling_rate - тоже, что и sample_rate (модель обучена на аудио в 16кГц).	В среднем 55.4 sec	[feature_size, 1499, 160]

Сравнение эмбеддингов - EnCodec

ScatterPlot двух компонент эмбеддингов, полученных при помощи TSNE



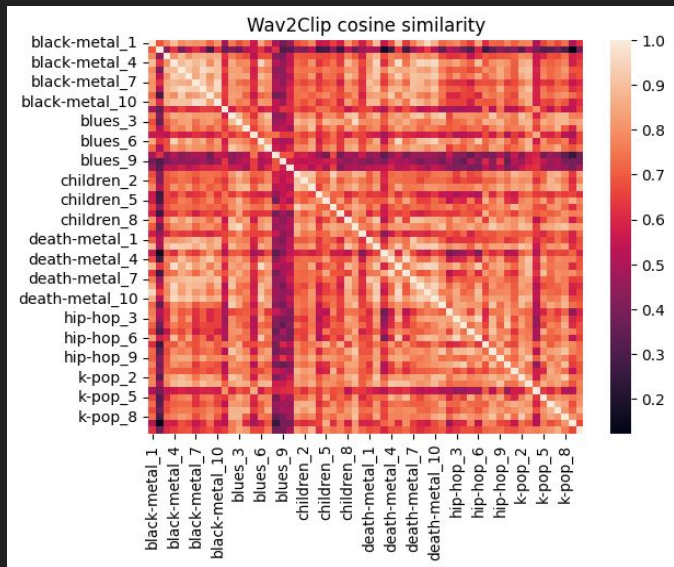
HeatMap косинусной близости для треков разных жанров



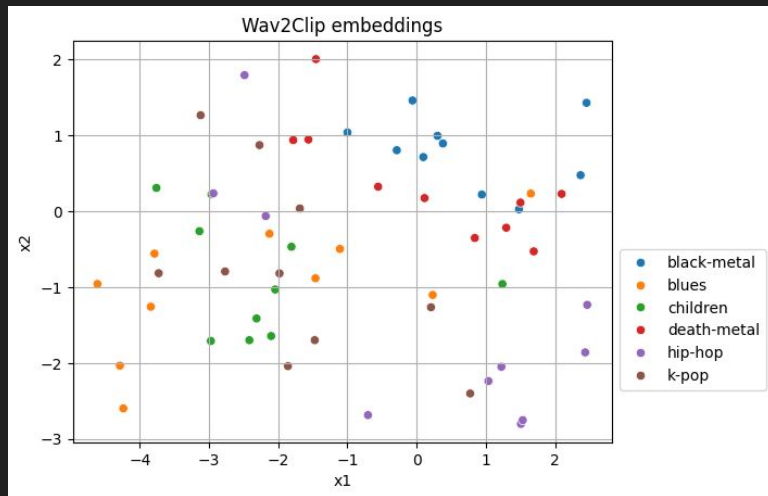
Эмбеддинги EnCodec'a не показывают явного различия между песнями различных жанров, если смотреть на метрику косинусной близости - скорее всего информация о жанрах неявно присутствует в эмбеддингах модели. Однако, если эмбеддинги сжать при помощи TSNE до 2 компонент, можно заметить, что некоторые близкие жанры - black-metal и death-metal находятся довольно близко друг к другу. Но в целом нет четких границ между облаками точек песен разных жанров.

Сравнение эмбеддингов - Wav2Clip

ScatterPlot двух компонент эмбеддингов, полученных при помощи TSNE



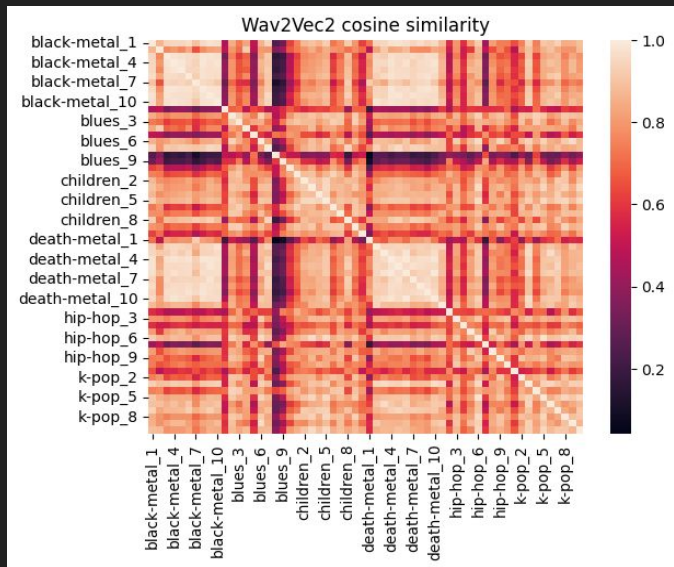
HeatMap косинусной близости для треков разных жанров



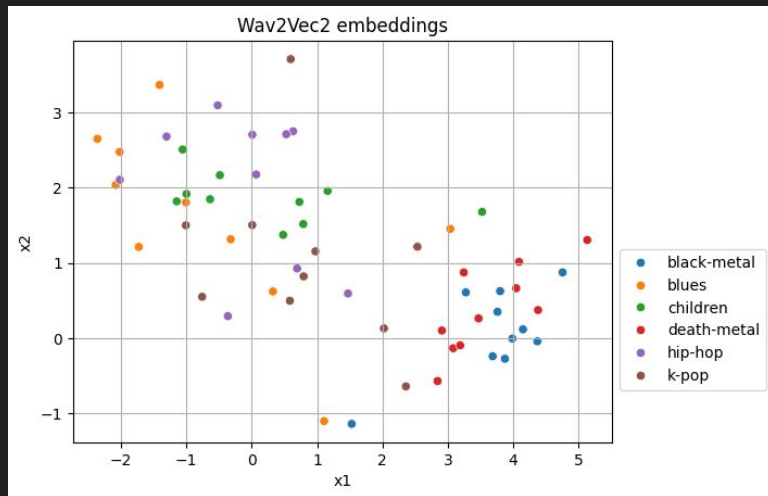
Эмбеддинги Wav2Clip по косинусной близости показывает более четкие результаты - песни одного жанра имеют больший коэффициент по данной метрике, чем эмбеддинги разных. Тоже можно сказать и про облака точек эмбеддингов после TSNE, заметны четкие границы между жанрами, в том числе облака точек black и death metal находятся рядом.

Сравнение эмбеддингов - Wav2Vec2

ScatterPlot двух компонент эмбеддингов, полученных при помощи TSNE



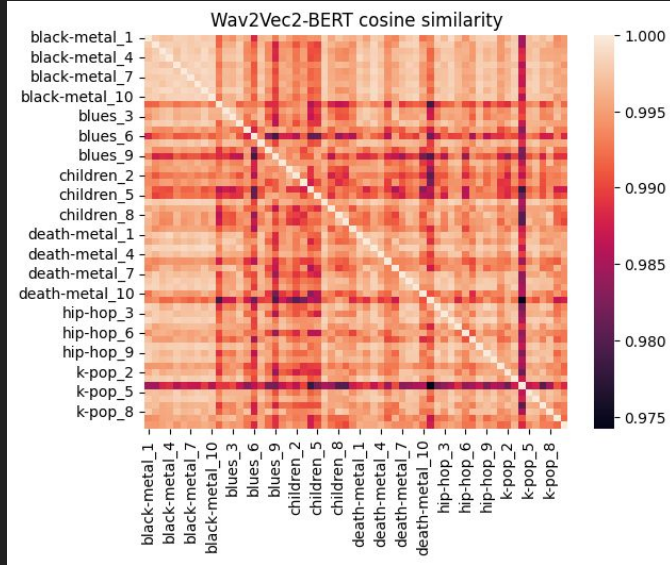
HeatMap косинусной близости для треков разных жанров



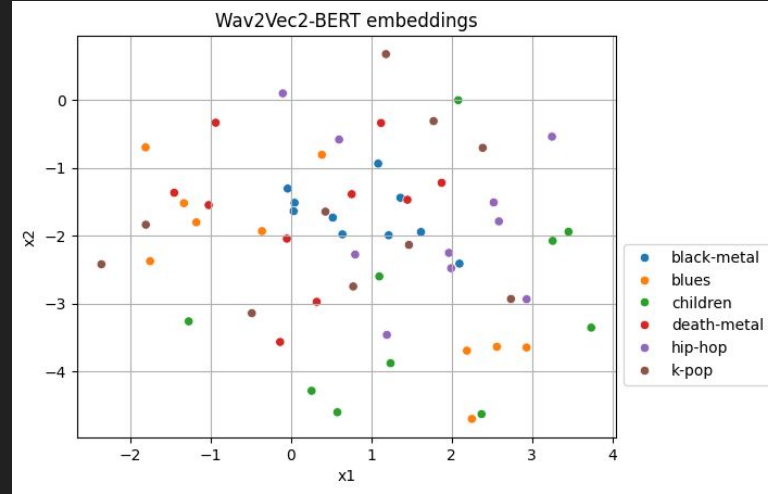
Эмбеддинги Wav2Vec2 по косинусной близости видят разницы между треками одного и различных жанров, по крайней мере лучше, чем EnCodec. Вообще эмбеддинги данной модели часто используют в моделях семантического анализа речи, и можно предположить, что треки разных жанров она также пытается разделить по эмоциональному окрасу, из за чего на обеих картинках более агрессивные жанры - black и death metal находятся близко, а более спокойные и безобидные - k-pop, children и blues перемешаны.

Сравнение эмбеддингов - Wav2Vec2-BERT

ScatterPlot двух компонент эмбеддингов, полученных при помощи TSNE



HeatMap косинусной близости для треков разных жанров



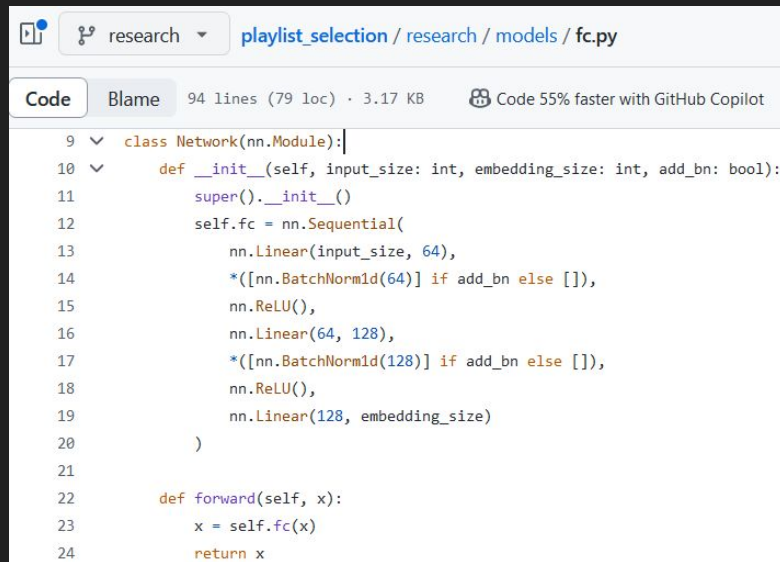
Эмбеддинги Wav2Vec-BERT показывают не лучший результат в различии жанров, судя по косинусной близости и графику выделенных компонент при помощи TSNE. Модель в целом не видит различия между wybranными треками, выделяя только отдельные, которые сильно отличны по звучанию от остальных. Вероятнее всего, эта модель вовсе не приспособлена для обработки аудио музыкального формата.

Metric Learning

Основной задачей Metric Learning является построение модели, позволяющей разделить объекты на основе некоторой метрики/расстояния.

В нашем случае мы хотим, чтобы приближались треки с похожими характеристиками по жанру, году выпуска и исполнителю.

Вся “магия” в этом подходе происходит в голове сети, на чем и был фокус внимания, поэтому в качестве бэкбона была использована простая MLP сеть из трех слоев.



The screenshot shows a code editor interface with a file path `playlist_selection / research / models / fc.py`. The code defines a `Network` class that inherits from `nn.Module`. The `__init__` method takes `input_size`, `embedding_size`, and `add_bn` as arguments. It initializes a `self.fc` attribute as a `nn.Sequential` block containing three layers: a `nn.Linear` layer with `input_size` and 64 units, a `nn.BatchNorm1d(64)` layer (if `add_bn` is true), a `nn.ReLU()` layer, another `nn.Linear` layer with 64 and 128 units, another `nn.BatchNorm1d(128)` layer (if `add_bn` is true), another `nn.ReLU()` layer, and a final `nn.Linear` layer with 128 units and `embedding_size`. The `forward` method takes an input `x`, passes it through `self.fc`, and returns the result.

```
9 class Network(nn.Module):  
10     def __init__(self, input_size: int, embedding_size: int, add_bn: bool):  
11         super().__init__()  
12         self.fc = nn.Sequential(  
13             nn.Linear(input_size, 64),  
14             *([nn.BatchNorm1d(64)] if add_bn else []),  
15             nn.ReLU(),  
16             nn.Linear(64, 128),  
17             *([nn.BatchNorm1d(128)] if add_bn else []),  
18             nn.ReLU(),  
19             nn.Linear(128, embedding_size)  
20         )  
21  
22     def forward(self, x):  
23         x = self.fc(x)  
24         return x
```

Оценка качества

Для оценки качества моделей была придумана своя метрика, считающаяся следующим образом:

Для каждого жанра

1. Выбираем случайный трек указанного жанра (якорь)
2. Сэмплируем N треков аналогичного жанра и N треков из остальных жанров
3. Для якоря считаем среднее евклидово и косинусное расстояние до объектов своего и чужого жанра

Отдельно для евклидова и косинусного расстояния считаем долю жанров, для которых внутрижанровое расстояние меньше, чем внежанровое.

При этом надо понимать, что в экспериментах использовались аудио-фичи из меты, а не аудио-эмбединги, поэтому значения могут быть сильно занижены, по сравнению с финальной версией.

Triplet Loss

Идея: для каждого объекта (anchor/A) находим релевантный (positive/P) и нерелевантный (negative/N) объекты и пытаемся “сблизить” anchor с positive и “отдалить” anchor от negative.

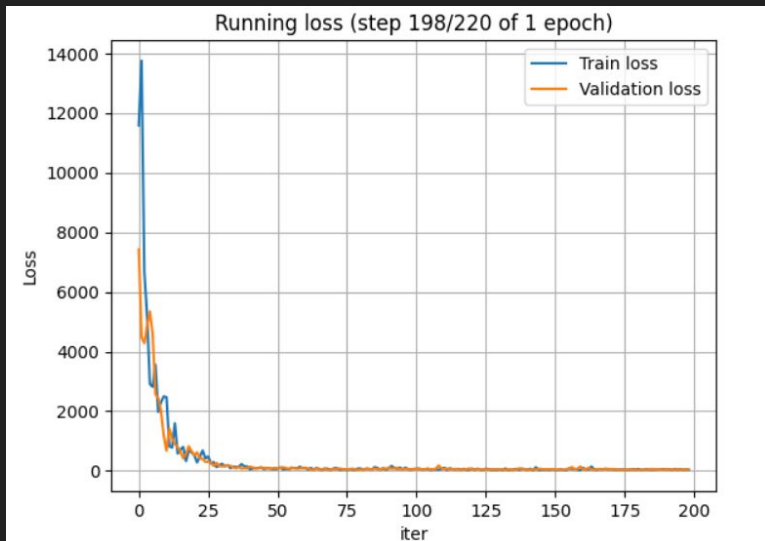


Рис 4. Кривые обучения

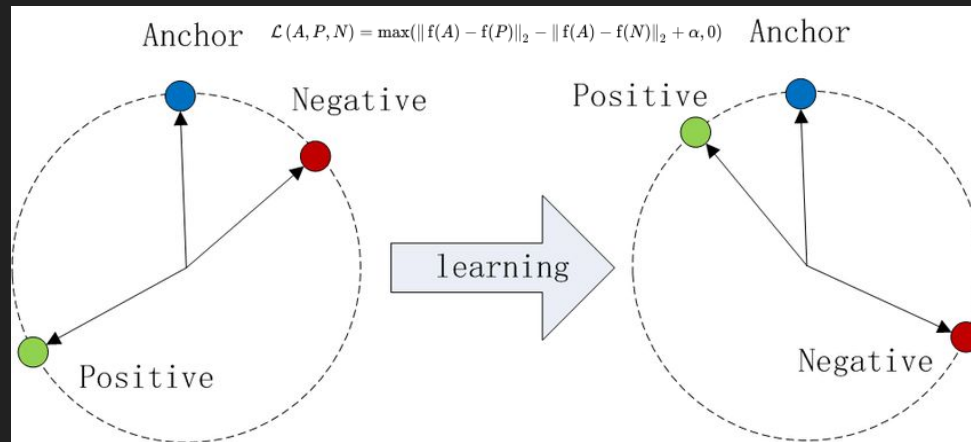


Рис 3. Иллюстрация и формула для Triplet Loss

Сэмплинг positive и negative объектов осуществлялся с помощью самописной оценки релевантности, которая учитывает пересечение жанров, исполнителей, а также близость по году выхода. **Итог:** на выходе получаем достаточно хорошие эмбединги.

```
===== example 1 =====
anchor - New N3on ['Playboi Carti'] ['atl hip hop', 'plugg', 'pluggnb'] cosine_sim - 1.0 euclidian - 0.0
positive - Antidote ['Travis Scott'] ['hip hop', 'rap', 'slap house'] cosine_sim - 0.951 euclidian - 220.261
negative - Admit It!! ['Say Anything'] ['emo', 'neon pop punk', 'pop punk'] cosine_sim - 0.891 euclidian - 344.539
```

Рис 5. Пример A/P/N объектов

ArcFace

Идея

Основной идеей данного метода является добавление отступа m к классическому CrossEntropyLoss, для сосредоточения объектов одного класса возле своего центроида, с отступом хотя бы m от центров других кластеров.

Описание экспериментов

В качестве фичей использовались аудио-фичи из меты, поэтому надо понимать, что финальные метрики будут сильно лучше. Таргет - комбинация жанра и декады выпуска. Отдельно стоит заметить, что качество сильно зависит от параметров модели `scale` и `margin`.

1. Softmax loss:

$$L = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^n e^{W_j^T x_i + b_j}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{s \cos \theta_{y_i} + b_{y_i}}}{e^{s \cos \theta_{y_i} + b_{y_i}} + \sum_{j=1, j \neq y_i}^n e^{s \cos \theta_j + b_j}}$$

3. Additive margin:

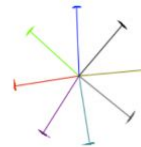
$$L = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{s \cos(\theta_{y_i} + m) + b_{y_i}}}{e^{s \cos(\theta_{y_i} + m) + b_{y_i}} + \sum_{j=1, j \neq y_i}^n e^{s \cos \theta_j + b_j}}$$

2. Transform

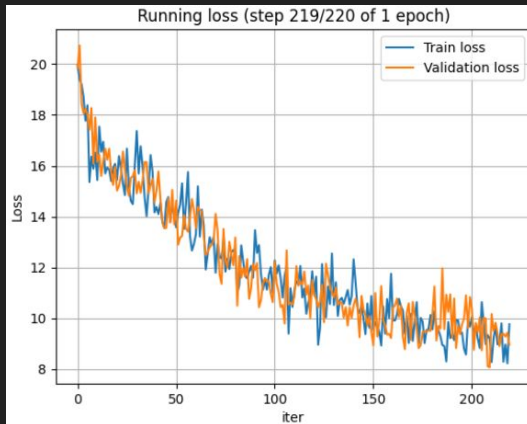
$$W_j^T x_i = \|W_j\| \|x_i\| \cos \theta_j$$



(a) Softmax



(b) ArcFace



Результаты

В результате получаем относительно неплохую модель, однако немного уступающую по качеству модели с TripletLoss.

Значение метрики по жанру составило 0.66 для евклидового расстояния и 0.69 для синусного

Дополнительные данные

Как возможное “дешевое” обогащение собранных данных, может использоваться [Million Song Dataset](#), в котором содержится крайне похожая на нашу мета информация для миллиона треков, выпущенных до 2012 года. При этом имеется возможность помимо мета информации достать и аудио дорожки треков.

Что касается разметки, ожидаемо ничего пригодного к использованию в открытом доступе не было найдено, самое похожее на разметку для ранжирования - User-Track матрицы прослушиваний, однако они зачастую обезличены и немногочисленны, не говоря уже о необходимости строить рекомендательную модель для извлечения отранжированного списка для каждого трека.

Выводы

В итоге по каждому из запланированных пунктов была проведена существенная работа:

1. Проанализированы модели для построения классических аудио-эмбеддингов, и выявлены подходящие нам кандидаты.
2. Проведены эксперименты с подходами Metric Learning, на данный момент чуть лучше себя показывает модель с TripletLoss.
3. Найден потенциальный источник для более легкого обогащения базы треков, для которого, как минимум, можно будет избежать блокировок по API.

Спасибо за внимание