Pontifícia Universidade Católica
do Rio de Janeiro

**Matheus Manhães Masson**

# Cold Start em Recomendação de Músicas Utilizando Deep Learning

**Dissertação de Mestrado**

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática do Centro Técnico e Científico da PUC-Rio.

Orientador: Prof. Ruy Luiz Milidiú

Rio de Janeiro
Agosto de 2016

**Matheus Manhães Masson**

# Cold Start em Recomendação de Músicas Utilizando Deep Learning

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

**Prof. Ruy Luiz Milidiú**
Orientador
Departamento de Informática – PUC-Rio

**Profª. Simone Diniz Junqueira Barbosa**
Departamento de Informática – PUC-Rio

**Prof. Edward Hermann Haeusler**
Departamento de Informática – PUC-Rio

**Prof. Márcio da Silveira Carvalho**
Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 23 de Agosto de 2016

**Matheus Manhães Masson**

Recebeu seu título de Bacharel em Sistemas de Informação pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) em 2011. Suas principais áreas de interesse são Machine Learning e Engenharia de Software.



Ficha Catalográfica

Masson, Matheus Manhães

Cold Start em recomendação de Músicas utilizando Deep Learning / Matheus Manhães Masson ; orientador: Ruy Luiz Milidiú. – 2016.
61f. : il. (color.) ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2016.
Inclui referências bibliográficas

1. Informática – Teses. 2. Deep Learning. 3. Aprendizado de Máquina 4. Algoritmos. 5. Recomendação. 6. Cold Start. I. Milidiú, Ruy Luiz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD:004

## Agradecimentos

Aos meus Pais, ao meu Irmão e amigos, que sempre estiveram do meu lado acreditando e me dando forças para continuar e finalizar mais esta etapa.

Ao meu orientador Professor Professor Ruy Luiz Milidiú pelo estímulo e parceria para a realização deste trabalho.

Ao CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus pais, pela educação, atenção e carinho de todas as horas.

A NVIDIA que apoiou e forneceu auxilio ao projeto.

Aos professores que participaram da Comissão examinadora.

A todos os professores e funcionários do Departamento pelos ensinamentos e pela ajuda.

A todos os amigos e familiares que de alguma forma me estimularam ou me ajudaram.

## Resumo


Manhães Masson, Matheus; Milidiú, Ruy Luiz. **Cold Start em recomendação de Músicas utilizando Deep Learning**. Rio de Janeiro, 2016. 61p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Sistemas de recomendação são utilizados para oferecer informações ou Produtos para Usuários ao aprender o Perfil de seus Usuários de forma automática utilizando técnicas de *Machine Learning*. Normalmente, esses Sistemas se baseiam em dados previamente coletados sobre seus Produtos e Usuários. Quando não existem dados prévios esses Sistemas não funcionam – esse problema é chamado de *Cold Start Problem*. Este trabalho é focado no problema do *Cold Start* que afeta a qualidade das recomendações e a impossibilidade de recomendar Músicas Novas por métodos tradicionalmente utilizados. Para isso, utilizamos *Deep Learning* com o Áudio das Músicas e assim extraimos características úteis para recomendar. A partir de Fatores Latentes obtidos por uma Fatoração de Matriz, uma Rede Neural Convolucional é treinada para aprender esses Fatores das Músicas utilizando o Áudio delas. Com isso a rede pode ser utilizada para predizer Fatores Latentes das Músicas, utilizando somente o Áudio, sem necessidade de dados prévios. Essa se torna uma solução viável para o *Cold Start Problem*.O resultado mostra que essa é uma solução funcional para o problema ainda que não alcance as melhores métricas de métodos tradicionais. A Rede Convolucional treinada consegue aprender a partir do Áudio e predizer os Fatores. Sendo assim o resultado permite recomendar Músicas Novas e ainda pode incrementar recomendações utilizando métodos Híbridos.





## Palavras-chave

Recomendação; Deep Learning; Fatoração de Matrizes; Algoritmos, Aprendizado de Máquina;Cold Start Problem; Música.

## Abstract

Manhães Masson, Matheus; Milidiú, Ruy Luiz (Advisor). **Cold Start in Music recommendation using Deep Learning.** Rio de Janeiro, 2016. 61p. MSc. Dissertation– Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Recommender systems are used to provide information or Products for users by learning the profile of their users automatically using Machine Learning techniques. Typically these Systems are based on previously collected data on their products and users. When there are no previous data these Systems do not work, this problem is called Cold Start Problem.This work is focused on the Cold Start Problem that affects the quality of the recommendations and the failure to recommend New Songs by methods traditionally used.For this solution we use Deep Learning with Audio of the Songs and thus extract useful features to recommend. From Latent Factors obtained by Matrix Factorization a Convolutional Neural network is trained to learn these factors using the Audio. Thus the network can be used to predict Latent Factors of Songs using only the Audio without the need for previous data. This becomes a viable solution to the Cold Start Problem. The result shows that this is a workable solution to the Problem even if they did not reach the best metrics of traditional methods. The Convolutional Network trained learns from the Audio and predicts factors. Thus the result allows to recommend New Songs and may even increase recommendations using Hybrid methods.



## Keywords

Recommendation; Deep Learning; Matrix Factorization; Machine Learning; Algorithms, Cold Start Problem; Music.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# 1 Introdução

Cada vez mais o uso de aplicativos de Música digital sob demanda se torna realidade entre os consumidores. Aplicativos como o Spotify [1]tem enfrentado desafios na recomendação de novas músicas para Usuários. Os mecanismos mais comuns usados para recomendar Músicas se baseiam em dados sobre a Música e dados colaborativos de Usuários. Essa forma de recomendação encontra problemas ao se deparar com músicas novas ou com baixa popularidade por não ter dados suficientes para recomendar acaba deixando-as fora das recomendações.

O problema de recomendação de músicas novas é chamado de *Cold Start Problem*. Para efeitos de análise dos resultados, experimentos de recomendação são relatados em conjuntos de Músicas onde se tem dados prévios, chamado de *Warm Start.* A proposta é verificar resultados em conjuntos de Músicas que simulem o *Cold Start* comparando com conjuntos de *Warm Start*, ambos no Conjunto de Teste.

A proposta em questão é utilizar caracteristicas extraídas pelo áudio das músicas utilizando uma Rede Neural Convolucional. Uma Fatoração de Matriz é executada inicialmente para ser utilizada como entrada da Rede como no Artigo (Aaron van den Oord 2014) que serviu como base para o Projeto. O áudio é associado a cada fator de cada Música para criar o Conjunto de entrada da Rede. No passo seguinte, são separados os Conjuntos de Treino e Teste. O Áudio é cortado para conter sempre 6 segundos cada para que o treino fosse mais rápido e por ter se mostrado suficiente para identificar a Música. Esse Áudio é transformado em imagens resultantes por meio da transformação dos áudios em Espectrogramas. Essa forma possibilita o uso de Redes Convolucionais para aprender baseada nos Espectros como Imagens.

Após a Rede ser treinada, executamos o Teste. As imagens pertencentes ao Conjunto de Músicas do Teste passa pela rede para extrair o resultado na forma de Fatores Latentes, que são características extraídas do áudio em forma de vetor, por isso Latentes. Com esses fatores resultantes é feita a análise das recomendações,

1 Spotify – Empresa que oferece serviços de Música Digital - "*Spotify - Music for Everyone.*", 22 Ag. 2016.

utilizando o Conjunto de Teste e os Fatores dos Usuários anteriormente obtidos pela Fatoração de Matriz.

Essa tarefa tem sido estudada por meio de técnicas de Machine Learning. Uma técnica recente, que apresenta resultados interessantes, usa o Áudio para recomendação (Aaron van den Oord 2014). A proposta em questão apresenta um resultado relatado com a métrica Mean Average Precision( MAP) de 0.0067, em um conjunto de dados simulados no estado Warm Start treinada com 382.410 Músicas. O referido trabalho não relata resultados finais em um conjunto Cold Start.

Na presente proposta, é apresentado um estudo sobre as técnicas e etapas para a recomendação baseada em Áudio. A mesma é baseada no uso das técnicas de Fatoração de Matriz e Deep Convolutional Neural Networks. São propostos dois experimentos: o primeiro com um Conjunto Pequeno de Músicas mais Populares e outro com todos os dados do *Dataset* que foi formado a partir do *EchoNest Dataset*.

Uma boa recomendação traz vantagens para os aplicativos, que passam a vender mais músicas, e para o Usuário que deseja conhecer músicas mais próximas possíveis ao seu gosto. Além disso, se torna necessária pela grande quantidade de dados presentes nos bancos de dados de empresas relacionadas a Músicas Digitais. A quantidade impossibilita essa tarefa ser feita de forma manual.

O objetivo do projeto é encontrar soluções para o problema do *Cold Start*, que sejam compatíveis com modelos de predição, possibilitando a comparação de resultados. Com isso, apresentar uma forma viável para recomendação em casos de Músicas Novas, que anteriormente eram ignoradas por métodos tradicionais como a Fatoração de Matriz.

Este trabalho está organizado da forma descrita a seguir. No Capítulo 2, detalhamos o problema de Cold Start em sistemas de Recomendação, enfatizando o caso especial de Recomendação de Músicas. No Capítulo 3, descrevemos a metodologia proposta, baseada na utilização do Áudio para determinar os Fatores Latentes para recomendação das Músicas. No Capítulo 4, reportamos os resultados experimentais obtidos com a metodologia proposta aplicada ao *Echo Nest Taste Profile Dataset*. No Capítulo 5, sumarizamos as conclusões deste trabalho e apontamos direções futuras de extensão do mesmo.

# 2 Técnicas de Recomendação

A tarefa de Recomendação pode ser definida como a forma de apresentar para um Usuário um ítem que ele não conheça, embasado no conhecimento do seu perfil, com o objetivo de apresentar um ítem que ele possa gostar.

Um cenário comum de recomendação são aplicações em *e-commerce*, que utilizam essa técnica para mostrar produtos mais próximos ao perfil do consumidor, com interesse na venda desses produtos. Ou ainda, uma página de notícias que guarda informações sobre as notícias que o Usuário costuma acessar e passa a destacar notícias que parecem ser do interesse dele.

Nesse estudo, o foco é em um tipo de recomendação específico chamado de recomendação baseada em conteúdo, que será definida na sessão 2.2 com mais detalhes e pode ser mais aprofundada com a leitura do Artigo (Castillo 2007) sobre esse tipo de recomendação.

A tarefa de recomendação é desejável para que se possa apresentar opções de itens a usuários que facilitem o consumo. O objetivo é apresentar opções personalizadas que sejam compatíveis com o perfil do usuário.

A recomendação pode ser utilizada em diversas situações e aplicada a diferentes tipos de itens, como filmes, produtos em sites de compra, livros, música entre outros.

Existem diversas técnicas de recomendação conhecidas. Essas técnicas podem ser agrupadas em dois tipos, filtragem colaborativa e recomendação baseada em conteúdo.



## 2.1 Filtragem Colaborativa

A Filtragem Colaborativa á uma técnica que se baseia nos dados dos Usuários que normalmente são informados explicitamente pelos mesmos. Tem o nome de colaborativa justamente por utilizar dos dados de todos os usuários para definir as recomendações para um deles. Sendo assim, todos colaboram na recomendação.

Quando o usuário não informa sua preferência explicitamente, podem ser inferidas preferências a partir de dados de consumo do Usuário – no nosso caso, quantas vezes ele ouviu cada Música. Essa abordagem se chama de Filtragem Colaborativa com *feedback* implícito (Yifan Hu 2008).

A filtragem colaborativa utiliza dados pré-existentes sobre usuários e itens. Ele necessita de dados que definam as características de usuários e itens. Os dados podem ser obtidos de diversas formas. Podem ter sido cadastrados por empresas que precisaram gastar recursos com a análise dos itens e definição de características suficientes para definir os mesmos. Podem também ter sido fornecidas de forma gratuita e colaborativa por usuários, como por exemplo uma avaliação de filmes. Outra forma de obtenção é a utilização de dados de consumo, que não precisam ser cadastrados de forma explícita. Dados de consumo são informações implícitas que podem ser utilizadas para definir preferências. No caso da Música pode ser utilizado o contador de consumo de Músicas por Usuário.

Em alguns casos esses dados podem conter erros. Quando são definidas características para representar os itens, estas não necessariamente são suficientes. Quando pessoas analisam itens ou fornecem informações sobre preferências podem gerar dados incorretos por diversos motivos, como falta de atenção, digitação incorreta entre outros. Essa possível existência de erros não invalida as técnicas de filtragem colaborativa, muitas vezes mesmo contendo erros o resultado da recomendação é suficiente para ser utilizado.

Como elas se baseiam em dados existentes, sofrem alguns problemas impeditivos para alguns casos de recomendação. O principal é o chamado *Cold Start*, o problema encontrado com itens novos onde ainda não existem informações de consumo sobre os mesmos (Aaron van den Oord 2014).

Em muitos casos, também ocorrem problemas na recomendação de itens pouco acessados, pois muitas dessas técnicas se baseiam em aprendizado supervisionado sobre grandes conjuntos de dados onde a aparição dos itens precisa ocorrer muitas vezes para que seja possível definir suas características.

Nesses casos, itens pouco consumidos acabam ganhando pouca importância nas recomendações, se tornando difíceis de recomendar mesmo para usuários que deveriam recebê-los como recomendação.

Algumas técnicas desse tipo amplamente utilizadas são a Fatoração de Matriz, Conjunto de Palavras (bag-of-words), Vizinhos próximos (*Nearest Neighbors*), dentre outras. Elas diferem na abordagem de resolução do problema e no tipo de dados que precisam como entrada. Eu optei pela Fatoração de Matriz pois meus dados inicais contém poucos atributos, somente o consumo do Usuário, as outras técnicas citadas precisam de mais atributos para que funcionem. Essa técnica foi proposta em 2009, será detalhada na sessão 4.1 e pode ser mais aprofundado seu entendimento pelo Artigo (Y. Koren 2009).

## 2.2 Recomendação Baseada em Conteúdo

Se baseia em dados de conteúdo onde o objetivo é extrair aspectos do conteúdo que definam suas características ou ainda o uso de conteúdo obtido manualmente por algum processo definido. Este pode não necessitar de dados cadastrados previamente se o conteúdo for utilizado com extração automática de caracteristicas, esses dados são obtidos diretamente pelo conteúdo do item a recomendar. Foi proposta essa técnica por (Foote n.d.) inicialmente.

Essa técnica com foco em obtenção automática não apresenta os problemas da filtragem colaborativa pois não sofrem influência humana no processo de construção das características. Também não sofre o problema do Cold Start, pois itens novos podem ser recomendados pelas características extraídas. Itens pouco consumidos não sofrem desvantagem pois os dados de consumo não são utilizados e suas características são extraídas de forma equivalente a itens muito consumidos.

O desafio desse tipo de recomendação é extrair do conteúdo características relevantes, o que na maioria das vezes é uma tarefa difícil. Além disso, existem características que definem os itens que não são possíveis de extrair do conteúdo. No caso da música por exemplo, não existe como extrair dados de localidade, datas, ou outras características que não estão no áudio.

As técnicas de recomendação baseada em conteúdo precisam encontrar formas de extrair suas características relevantes para recomendação. Essas formas de extração precisam ser padronizadas para que possam ser aplicadas ao mesmo tipo de conteúdo gerando resultados semelhantes.

No caso do áudio, existem duas principais formas de extrair essas características. A primeira consiste na aplicação de técnicas de engenharia de áudio. O propósito é utilizar conhecimentos sobre manipulação de áudio definidos previamente na construção do treino e tentar encontrar as características com as técnicas. A segunda forma consiste em extrair as características utilizando redes neurais que aprendam a identificar essas características sem que seja necessária a definição prévia de nenhuma técnica de extração.

# 3 Cold Start

O *Cold Start* é um problema importante a ser tratado tanto para as empresas que visam lucrar com boas recomendações como para os Usuários que recebem recomendações mais rapidamente. Sempre quando surgem Músicas novas a infomação mais informativa é o Áudio. Como nenhum Usuário ainda não ouviu não se sabe o tipo de público, ou seja, a preferência dos Usuários opor ela. Isso se chama *Cold Start*. O primeiro a tentar resolver esse problema foi (Aaron van den Oord 2014). Dado o fato de que se tem uma informação valiosa que é o Áudio, com seu conteúdo pode ser possível extrair características boas para recomendar para o público certo mesmo sendo nova. Músicas novas surgem todos os dias, recomenda-las o mais rápido e adequadamente resulta em um ganho para as empresas de Música Digital. Além disso, algumas vezes Músicas pouco populares acabam com recomendações pobres por terem baixa frequencia de consumo. Nesses casos, a incorporação de características do Áudio pode melhorar os resultados destas também.

Recomendar pelo Áudio significa extrair características do Áudio aliadas as preferencias dos Usuários. O uso do Deep Learning permite extrair características a partir do áudio, estas podem ser relacionadas as preferências dos Usuários. O

Deep Learning consiste em uma técnica de Aprendizado de Máquina baseada em Redes Neurais Profundas, ou seja, com muitas camadas. Essa técnica foi proposta em 2012 por (A. Krizhevsky 2012) e será detalhada no capítulo 8. Permite encontrar essas características sem indicar quais inicialmente. Isso resulta em características obtidas do Áudio que seriam muito difíceis de identificar manualmente com técnicas de engenharia de atributos de Áudio.

# 4 Fatores Latentes baseados no Áudio

Foram realizados dois experimentos. O primeiro com um conjunto menor de dados somente com as Músicas mais populares e o segundo com o Conjunto Completo.

A escolha da técnica deve levar em consideração alguns aspectos relevantes. O tipo dos itens a serem recomendados, os dados existentes e a quantidade de dados são aspectos importantes.

A ideia de uma boa recomendação não é somente atingir uma boa métrica no teste. Recomendações onde o resultado são itens populares não necessariamente são recomendações inteligentes, menos óbvias, como já disse (Aaron van den Oord 2014) em seu artigo. apesar de muitas vezes acertar muito. Nesse caso, não seria necessário um algoritmo de inteligência artificial para aprender a recomendar. Uma forma, por exemplo, seria sempre recomendar os itens mais populares. Uma boa recomendação precisa conseguir ser interessante para o usuário e não óbvia como por exemplo recomendar as Músicas populares. Ser uma boa recomendação significa surpreender o Usuário com uma Música que ele goste, nunca tenha ouvido e aprove a recomendação por ser coerente com o seu perfil.

Para atingir boas recomendações pode ser preciso uma junção de técnicas e resultados com o intuito de compensar a falha de cada uma, utilizando outras técnicas em conjunto. Como exemplo, o Spotify, que é atualmente a maior empresa de Serviços de Música digital por aplicativo, utiliza uma gama de algoritmos de recomendação que resultam em seu produto final, utilizando tanto técnicas de filtragem colaborativa como recomendação baseada em conteúdo.

Dentre as técnicas de recomendação a Fatoração de Matriz (Y. Koren 2009) é uma abordagem muito utilizada. Dentre suas principais vantagens estão os bons resultados referentes a qualidade apresentada em cálculos de métricas apropriadas, a necessidade de poucos dados e o espaço de armazenamento necessário.

Ela se baseia em dados previamente coletados. São necessários dados dos Usuários e Das Músicas. A escolha dos atributos varia de acordo com o algoritmo e necessidade. Em comum todas precisam de dados referentes a preferência dos

Usuários pelas Músicas. Essa preferência pode ter sido explicitamente indicada, como em avaliações feitas pelos Usuários ou implicitamente inferida através de dados conhecidos que possam indicar preferencia.

O principio da técnica é que dentre algumas características das Músicas pode-se obter a preferencia dos Usuários pelas mesmas. Assim descobre-se a preferencia do Usuário. Recomenda-se então Músicas que se enquadram nessas características que o Usuário não tenha ouvido.

Pode-se utilizar uma coleta de características manualmente para cada Música. Nesse caso, o fator que dificulta é o tempo necessário para pessoas avaliarem as Músicas e categorizarem e ainda a precisão dessa avaliação. Como alternativa, essas características podem ser calculadas pela fatoração, o que torna o processo mais rápido e preciso.

Existem diversas variações de algoritmos para Fatoração. O que mais diferencia todas é se os dados utilizados são implícitos ou explícitos. Dados explícitos significam que foram explicitamente forneceidas pelo Usuário, como uma avaliação de um filme por exemplo. Já os dados implícitos são dados que se revelam a partir de dados existentes do Usuário de forma implícita, como no meu caso a quantidade de vezes que um Usuário ouviu uma Música revela o quanto ele gosta. Em nosso caso, utilizemos uma técnica baseada em fatoração implícita para realizar os experimentos com os dados exatos do conjunto inicial. Existem técnicas baseadas em conceitos estatísticos para transformar dados implícitos em explícitos, isso pode ser útil em certos casos.

Dados implícitos acarretam em ter somente exemplos positivos no conjunto. No nosso caso eu tenho somente as Músicas que ele ouviu e assumo de forma implícita uma preferencia positiva. Não se tem exemplos de preferencias negativas, ou seja, indicação do que o Usuário não gosta. Isso além de influenciar na técnica a ser adotada também pode ser relevante para a escolha da métrica de avaliação dos resultados.

Inicialmente optamos pela técnica chamada de Weighted Matrix Factorization. A escolha é principalmente para permitir replicar os experimentos de (Aaron van den Oord 2014). É uma técnica baseada no uso de dados implícitos onde

leva-se em consideração um peso chamado de fator de confiança para os dados. Esse peso é considerado para indicar a influencia de um dado implícito na consideração da preferencia do Usuário. Em nosso caso, a quantidade de vezes que um Usuário ouviu cada Música foi utilizado para indicar esse fator de confiança. Quanto mais vezes ele ouviu, maior a confiança na preferencia do usuário pela Música.

## 4.1 Fatoração de Matrizes

Como entrada são utilizados os dados de consumo do Dataset. Cada tripla (Usuário, Música, Consumo) representa um exemplo. Os exemplos estão separados nos conjuntos de treino e teste. É importante observar que para a fatoração funcionar os Usuários e as Músicas necessariamente devem estar presentes nos dois conjuntos ao menos uma vez. Ela não consegue recomendar o que não viu no treino – o caso do Cold Start. Para ilustrar o funcionamento da Fatoração segue uma representação na Figura 2.

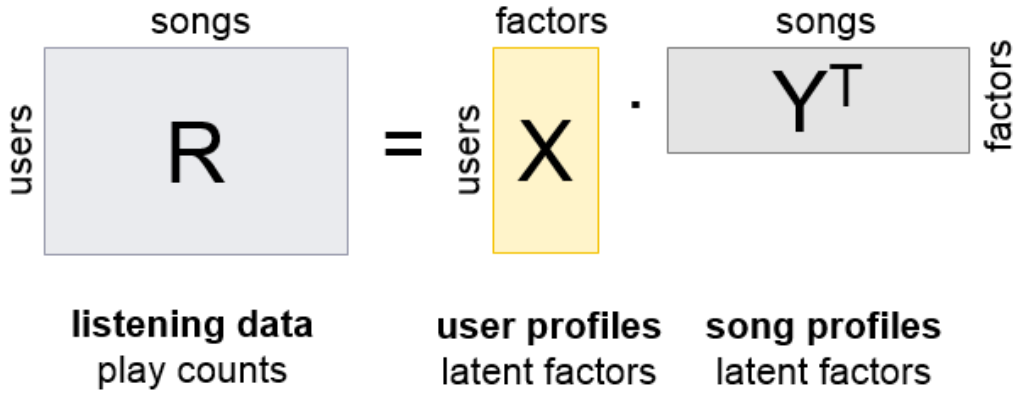


**Figura 1 -** Fatoração de Matriz

Em seguida, essas triplas precisam ser transformadas em uma matriz para o treino ser realizado. A Matriz é definida de forma esparsa, onde cada linha representa um Usuário e cada Coluna uma Música. Os elementos da Matriz são os dados de consumo. A Matriz é considerada esparsa, pois a maior parte não está preenchida. Considerando que somente os dados de consumo conhecidos estão presentes, cada Usuário na Matriz tem em sua linha somente as Músicas que ouviu.

O objetivo final do Algoritmo é preencher as células inicialmente vazias. Assim que preenchida pode-se recomendar os elementos aprendidos. A premissa é de que dado um vetor, que represente características da Música e outro que represente a preferencia do Usuário por essas características, pode-se obter um valor que representa a preferência do Usuário pela Música. Para obter esse valor, basta multiplicar os dois vetores. Sendo assim, a fatoração deve obter esses vetores para cada um. Recomenda-se as Músicas onde esses valores estão mais altos e que o Usuário não tenha ouvido.

Na prática, existe um vetor para cada Usuário e um para cada Música. Quando multiplicados, preenchem uma célula da Matriz. O aprendizado se baseia nos elementos que estão preenchidos. Quanto mais perto deles esses valores chegarem melhor a recomendação.

Esses vetores são inicializados aleatóriamente, com valores pequenos e multiplicados. Atualiza-se os vetores para que cheguem mais perto dos valores conhecidos. Devemos escolher o tamanho desses vetores inicialmente, essa escolha influencia no resultado. Por consequência, ocorre uma redução no espaço necessário, pois podemos armazenar em seguida somente os vetores.



## 4.2 Aprendizado

A função objetivo da fatoração minimiza o erro quadrático entre o que foi predito e recomendado e o que realmente o Usuário ouviu e estava no conjunto de teste retirando as Músicas preditas para o Usuário que estavam no Treino pois estas ele viu no treino então seria uma contagem injusta. Além disso é multiplicado ao erro uma constante c que representa a confiança da predição, quanto mais o usuário ouviu a Música maior essa consante. Ao final de cada somatório dos erros é acresentado um somatório de regularização vizando evitar overfitting do treino. A função matemática se encontra representada abaixo na Figura 2.

$$\min_{x_*, y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left( \sum_u ||x_u||^2 + \sum_i ||y_i||^2 \right)$$

**Figura 2 -** Função Objetivo

Onde P(u,i) significa o valor real do consumo do Usuário que é na realidade a quantidade de vezes que o usuário ouviu a Música. X representa os fatores das Músicas e Y os fatores dos Usuários. Somando em seguida uma regularização para evitar overfitting no treino.

O objetivo é minimizar o erro quadrático das predições para cada Usuário. Durante esse processo são obtidos os fatores das Músicas e Usuários.

Para esse experimento, utilizamos vetores de características de tamanho cinquenta.

Utilizamos a métrica *Mean Average Precision* (map), representada na Figura 3. Essa métrica permite avaliar o resultado de recomendação por ranking. Quanto maior o resultado melhor a recomendação está funcionando. A MAP é calculada utilizando a fórmula

$$ap@n = \sum_{k=1}^{n} P(k)/min(m,n)$$

$$MAP@n = \sum_{i=1}^{N} ap@n_i/N$$

**Figura 3 -** Mean Average Precision

- **P(k)** é a precisão calculada para uma recomendação **k**. Essa precisão é calculada pelo número de acertos do Conjunto de teste comparada a predição do modelo.
- **m** corresponde ao número de músicas que o usuário prefere
- **n** corresponde ao número de recomendações

Para ilustrar esses vetores de Fatores Latentes, podemos utilizar uma biblioteca que calcule a distância entre eles e plote o resultado em um espaço latente com duas dimensões como forma de verificar se os Fatores estão fazendo sentido quanto a suas proximidades. Nesse caso, ao visualizar esse espaço o que ocorre são

agrupamentos em *Clusters*. Músicas parecidas são exibidas de forma próxima. Nesse caso eu considero parecidas as que são mais próximas no espaço vetorial. Isso não necessariamente significa que são parecidas pelo áudio e sim pelas preferências encontradas na Fatoração que usou perfis de Usuário para calcular os vetores das Músicas. Segue abaixo na figura 5, um exemplo de visualização desse espaço latente. As Músicas foram agrupadas em 4 generos diferentes cada um com uma cor. Cada Música ocupa um espaço nesse mapa e quanto mais próximos os vetores das Músicas mais próximas elas se apresentam no mapa. O espaço preto na figura representam Músicas que não se encaixaram em nenhum dos 4 clusters.

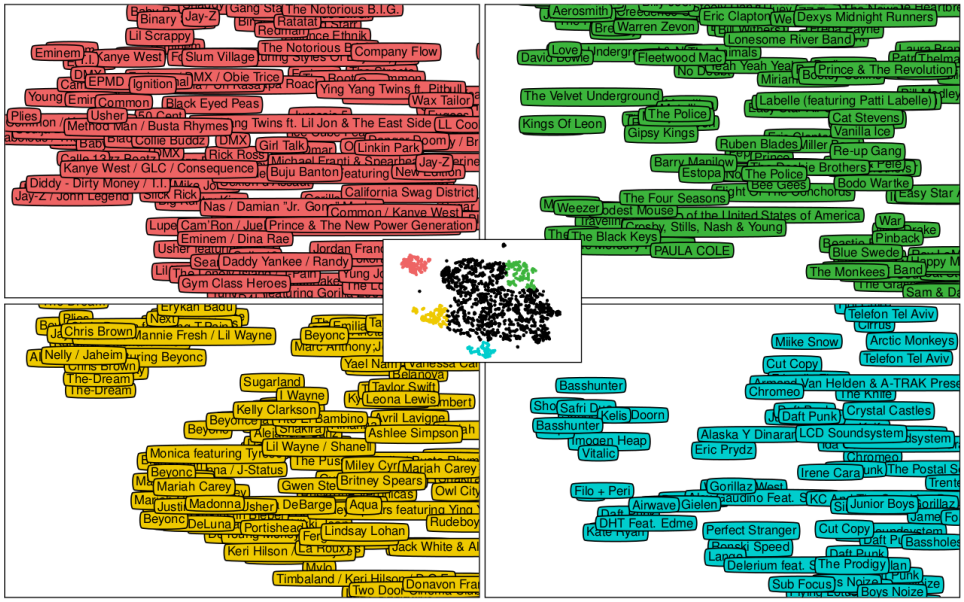


**Figura 4 -** Espaço Latente

# 5 Cold Start usando Áudio

A tarefa é realizada seguindo alguns passos de forma sequencial. Os detalhes das tarefas mais complexas são explicitados mais à frente em tópicos separados. O que se encontra abaixo é uma sequência de execuções que explicitam todos os passos para que se tenha uma visão do processo inteiro.

1. Transformação do dataset inicial em uma matriz esparsa carregada em memória que contém os dados de consumo.
2. Execução do treino da Fatoração de Matriz utilizando a matriz de consumo em memória, para obter vetores de características latentes relativas aos usuários e às músicas.
3. Para cada áudio obter um espectrograma representativo.
4. Com o conjunto de espectrogramas das músicas associados aos vetores de características obtidos pela fatoração, treinar uma Rede Neural Convolucional para que possa predizer esses vetores a partir dos Espectrogramas.
5. Testar a qualidade das recomendações utilizando o conjunto de teste para os resultados da Fatoração e para os resultados da Rede Neural.
6. Testar a qualidade das recomendações para o conjunto Cold Start
7. Apresentar os resultados dos testes.

# 6 Conjunto de Dados

O *Echo Nest Taste Profile Dataset* foi escolhido na época por ser o maior Dataset público de preferencias de Usuários. Ainda assim nem todas as Músicas, contidas em forma de identificadores, puderam ser obtidas pois o Dataset estava desatualizado. Mesmo assim foi possível utilizar por conseguir a maior parte das Músicas dele permitindo executar a tarefa proposta com os dados obtidos de forma suficiente. Alem disso esse foi o mesmo Dataset utilizado por (Aaron van den Oord 2014) em seu artigo. Isso permitiu comparar os resultados.

O conjunto inicial consiste em um arquivo texto contendo 3 colunas por linha que representam o identificador do Usuário no *Echo Nest*, o identificador da Música no *Echo Nest* e um contador de consumo do Usuário que significa quantas vezes o Usuário ouviu a Música. Ele contém cerca de 47 milhões de exemplos de consumo e ocupa 2gb no HD. Esse arquivo não pode ser aberto por qualquer editor de texto pelo tamanho. Para ler foi utilizada a Biblioteca *Pandas* (P. D. Team 2008) do Python que facilita esse processo. Foram realizados dois experimentos descritos abaixo diferenciados principalmente pela quantidade de dados. Exemplos do *Echo Nest* que consiste em 7 linhas de consumo de um mesmo Usuário:

| Usuário | Música | Consumo |
|---|---|---|
| b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOAKIMP12A8C130995 | 1 |
| b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOAPDEY12A81C210A9 | 1 |
| b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOBFNSP12AF72A0E22 | 1 |
| b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOBFOVM12A58A7D494 | 1 |
| b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOBNZDC12A6D4FC103 | 1 |
| b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOBSUJE12A6D4F8CF5 | 2 |
| b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOBVFZR12A6D4F8AE3 | 1 |

**Tabela 1 -** Dataset Inicial do Echo Nest

Os dados contidos no *Dataset* foram coletados pela Empresa *Echo Nest*. Representam informações de consumo de música por usuários em um Sistema Real. Os dados que poderiam identificar os usuários não estão no Dataset. A proposta do Dataset é fornecer informações sobre cada usuário contendo as músicas que foram ouvidas e quantas vezes foram ouvidas (Consumo).

A construção do *Dataset* foi feita pela empresa chamada *Echo Nest* e teve o propósito de fornecer esses dados para pesquisas acadêmicas de forma a contribuir com o avanço em tarefas relacionadas a música. O *Dataset* completo contém:

- 1.019.318 usuários distintos
- 384.546 músicas distintas
- 48.373.586 usuário-música-contador (Consumo)

## 6.1 Particionamento do *Dataset*

Para simular situações de recomendações reais alguns procedimentos de tratamento dos dados foi necessário. Primeiramente a divisão das linhas para formar os Conjuntos de Treino, Teste e Cold. Foram selecionadas randomicamente linhas de cada Usuário para o Teste e o restante ficou para o Treino. Após essa divisão o Teste foi dividido também em duas partes, o conjunto *Warm Start* e *Cold Start.* A diferença entre eles é que no *Warm Start* todas as Músicas presentes devem estar também no Treino, ao contrário do *Cold Start* onde foram separadas Músicas exclusivas para esse Conjunto. O teste com o Cold é possível pois a métrica de qualidade é sobre recomendação e não sobre acertar os fatores latentes.

### 6.1.1 Conjunto Pequeno

Subconjunto dos dados contendo as 12.910 músicas mais populares e 27.997 usuários. Esse modelo foi construído com o intuito de realizar uma experiência inicial reduzida da tarefa e verificar a influência da popularidade das músicas para recomendação. Esse *Dataset* resultante foi subdividido nas seguintes partes:

- 12.910 Músicas
- 27.997 Usuários
- 3.272.506 Triplas de Consumo
- 11.044 Músicas no Conjunto de Treino
- 1.859 Músicas no Conjunto de Teste
- 1.866 Músicas no Conjunto Cold

### 6.1.2 Conjunto Completo

Modelo contendo todos os dados do *Dataset* obtido. Utilizado para validar a tarefa em um *Dataset* de escala fidedigna ao problema apresentado. Seu particionamento foi da seguinte forma:

- 243.948 Músicas
- 151.028 Usuários
- 14.408.456 Triplas
- 214.685 Músicas no Treino
- 30.000 Músicas no Teste
- 29.263 Músicas no Cold

Devido à Fatoração necessitar de dados de consumo dos usuários e das músicas, todas as músicas do teste devem estar no treino. Diferente do *Cold Start* que foi separado para ser testado ao final e não podem aparecer no treino e no teste.

A Fatoração em resumo consiste em multiplicação de matrizes. Quando uma linha ou coluna está zerada a multiplicação dessa linha não irá revelar nenhuma informação. Por isso que todos os dados do teste devem estar no treino para a Fatoraçao. Isso ocasona o problema do *Cold Start*. O procedimento mais detalhado da Fatoração será descrito abaixo em uma sessão exclusiva. A seguir os detalhes sobre hardware e ferramentas de desenvolvimento utilizadas.



# 7 Infraestrutura Computacional

Para que todo o processo possa ser executado em um tempo hábil é necessário um Hardware atualizado, principalemente com relação a GPU onde ocorre o processamento da Rede Neural. Nesse Capítulo é detalhado o processo de obtenção do dataset final. A obtenção do Áudio e ainda todos os dados técnicos necessários utilizados no Projeto. Além dos dados técnicos é apresentada a Modelagem do Projeto. Esse é o principal Capítulo para quem quiser reproduzir o experimento.

Segue a descrição dos principais componentes da Máquina utilizada:

- CPU: Intel Core  i7-4790
- GPU: Nvidia GTX-980 com 4gb de memória
- Memória ram 32gb

## 7.1 APIs Utilizadas

Como o Dataset inicial só contém informações de identificação foi necessário encontrar APIs que fornecem dados adicionais principalmente o *preview* do Áudio das Músicas. O detalhamento do uso das APIs:

### 7.1.1 Echo Nest API

Utilizada para obter dados sobre os áudios presentes no Dataset inicial. Ela contém além dos identificadores do EchoNest[2], identificadores para o 7Digital e Spotify. Com esses identificadores pude usar outras APIs para obter os áudios. Com isso pude cruzar esses identificadores para manter cada Música de forma Única no Banco.

Após obter o preview das Músicas pelas APIs foi necessário filtrar as triplas de consumo para utilizar somente as que foram possíveis obter o Áudio. Pois somente as Músicas com Áudio eu poderia utilizar no Deep Learning. Filtrei Usuários com ao menos 40 informações de Consumo e Músicas com ao menos 20 Usuários diferentes que tenha ouvido. A seguir um exemplo em Json de retorno da API do EchoNest. Nesse exemplo observa-se o <foreign_id> como identificador da música no 7Digital e o que mais interessa que é o link para o download do áudio de exemplo está em <preview_url>. Com esse link pode-se fazer uma requisção para executar o download. Caso esse identificador nã exista eu passo a procurer pelas outras API's o áudio.

2 EchoNest – Empresa que oferece Músicas Digitais - "*The Echo Nest industry's*", 22 Ag. 2016

```
{
 "response": {
    "status": {
       "version": "4.2",
       "code": 0,
       "message": "Success"
    },
    "songs": [{
       "title": "Karma Police",
       "artist_name": "Radiohead",
       "tracks": [{
          "catalog": "7digital-US",
          "foreign_id": "7digital-US:track:2748611",
          "release_image":
"http://cdn.7static.com/static/img/sleeveart/00/002/577/0000257700_200.jp
g",
          "id": "TRKGPQB128F4252E52",
          "preview_url":
"http://previews.7digital.com/clips/34/2748611.clip.mp3"
       }
    }]
```


### 7.1.2 7Digital API

Utilizada para obter informações das Músicas pelos identificadores anteriormente obtidos pelo Echo Nest. Alguns não pude obter pois já não existiam mais então precisei utilizar outras APIs de outras Empresas. A partir do identificador da Música no 7Digital obtido pela API do Echo Nest foi possível encontrar a Música pela API do 7Digital. Com isso temos informação do nome do Artista em <artist><name>, e algumas vezes também pode-se obter o *preview* do áudio em <preview_url> quando já não obtida anteriormente pela API do EchoNest. Existem limitações relacionadas a quantidade de requisições que podem ser feitas a API por segundo, logo algumas vezes a API retorna erro. Então tive que checar diversas vezes.

Exemplo de retorno da API na próxima página:

```
<track id="3153161">
   <title>Island Stomp</title>
   <version />
   <artist id="60288">
      <name>Michel Camilo</name>
      <appearsAs>Michel Camilo</appearsAs>
      <slug>michel-camilo</slug>
      <image>http://artwork-cdn.7static.com/static/img/artistimages/00/000/602/0000060288_150.jpg</image>
      <isPlaceholderImage>true</isPlaceholderImage>
   </artist>
   <trackNumber>1</trackNumber>
   <duration>320</duration>
   <explicitContent>false</explicitContent>
   <isrc>USSM18900815</isrc>
   <type>audio</type>
   <release id="282442">
      <title>On Fire</title>
      <version />
      <type>Album</type>
      <barcode>074644529524</barcode>
      <slug>on-fire</slug>
      <artist id="60288">
         <name>Michel Camilo</name>
         <appearsAs>Michel Camilo</appearsAs>
         <slug>michel-camilo</slug>
         <image>http://artwork-cdn.7static.com/static/img/artistimages/00/000/602/0000060288_150.jpg</image>
         <isPlaceholderImage>true</isPlaceholderImage>
      </artist>
      <image>http://artwork-cdn.7static.com/static/img/sleeveart/00/002/824/0000282442_50.jpg</image>
      <label id="1085">
         <name>Epic</name>
      </label>
      <licensor id="2">
         <name>Sony Music</name>
      </licensor>
   </release>
   <discNumber>1</discNumber>
   <number>1</number>
   <download> <!-- Optional: You should only see this element if you are requesting with download usageType --
>
      <releaseDate>1989-10-02T23:00:00Z</releaseDate>
      <previewDate>1989-10-02T00:00:00Z</previewDate>
      <packages>
         <package id="2">
            <description>Standard</description>
            <price>
               <currencyCode>GBP</currencyCode>
               <sevendigitalPrice xsi:nil="true" />
               <recommendedRetailPrice>0.99</recommendedRetailPrice>
            </price>
            <formats>
               <format id="17">
                  <description>MP3 320</description>
               </format>
               <format id="33">
                  <description>AAC 320</description>
               </format>
            </formats>
         </package>
      </packages>
   </download>
   <subscriptionStreaming> <!-- Optional: You should only see this element if you are requesting with
subscriptionstreaming usageType -->
      <releaseDate>1989-10-02T23:00:00Z</releaseDate>
   </subscriptionStreaming>
   <adSupportedStreaming> <!-- Optional: You should only see this element if you are requesting with
adsupportedstreaming usageType -->
      <releaseDate>1989-10-02T23:00:00Z</releaseDate>
   </adSupportedStreaming>
</track>
```

### 7.1.3 Spotify API

Utilizado para obter informações sobre as Músicas, Artistas e obter outros *previews* que faltavam. Da mesma forma que o 7Digital [3]eu pude obter os previews pelos identificadores do Spotify coletados anteriormente pelo Echo Nest. Esse é o


```
{
 "album" : {
  "album_type" : "single",
  "external_urls" : {
   "spotify" : "https://open.spotify.com/album/3X33e7UII5loqrEgauOKEC"
  },
  "href" : "https://api.spotify.com/v1/albums/3X33e7UII5loqrEgauOKEC",
  "id" : "3X33e7UII5loqrEgauOKEC",
  "images" : [ {
   "height" : 640,
   "url" : "https://i.scdn.co/image/c171113a197828a6ee8017d1ede2e78c9a7df654",
   "width" : 640
  }, {
   "height" : 300,
   "url" : "https://i.scdn.co/image/de50cbd4f0e62be8d4ffd11c7d6f3c59d964bdb6",
   "width" : 300
  }, {
   "height" : 64,
   "url" : "https://i.scdn.co/image/5a18558e39d542ec9d71345daafe89d5862aa67a",
   "width" : 64
  } ],
  "name" : "Timber",
  "type" : "album",
  "uri" : "spotify:album:3X33e7UII5loqrEgauOKEC"
 },
 "artists" : [ {
  "external_urls" : {
   "spotify" : "https://open.spotify.com/artist/0TnOYISbd1XYRBk9myaseg"
  },
  "href" : "https://api.spotify.com/v1/artists/0TnOYISbd1XYRBk9myaseg",
  "id" : "0TnOYISbd1XYRBk9myaseg",
  "name" : "Pitbull",
  "type" : "artist",
  "uri" : "spotify:artist:0TnOYISbd1XYRBk9myaseg"
 }, {
  "external_urls" : {
   "spotify" : "https://open.spotify.com/artist/6LqNN22kT3074XbTVUrhzX"
  },
  "href" : "https://api.spotify.com/v1/artists/6LqNN22kT3074XbTVUrhzX",
  "id" : "6LqNN22kT3074XbTVUrhzX",
  "name" : "Kesha",
  "type" : "artist",
  "uri" : "spotify:artist:6LqNN22kT3074XbTVUrhzX"
 } ],
 "disc_number" : 1,
 "duration_ms" : 204053,
 "explicit" : false,
 "external_ids" : {
  "isrc" : "USRC11301695"
 },
 "external_urls" : {
```


[3] 7Digital – Gravadora de Músicas - "7Digital Gravadora de Música", 22 Ag. 2016.

caso onde eu ainda não consegui obter pelo 7digital então tento pelo Spotify. O Áudio está presente pelo link em <external_urls>. Exemplo:

### 7.1.4 Apple API

Utilizado da mesma forma que o Spotify, para obter previews de Músicas que faltavam. O problema aqui foi não encontrar correlação dos identificadores da

{"wrapperType":"track",
"kind":"song",
"artistId":909253,
"collectionId":120954021,
"trackId":120954025,
"artistName":"Jack Johnson",
"collectionName":"Sing-a-Longs and Lullabies for the Film Curious George",
"trackName":"Upside Down",
"collectionCensoredName":"Sing-a-Longs and Lullabies for the Film Curious George",
"trackCensoredName":"Upside Down",

"artistViewUrl":"https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewArtist?id=909253",

"collectionViewUrl":"https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewAlbum?i=120954025&id=120954021&s=143441",

"trackViewUrl":"https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewAlbum?i=120954025&id=120954021&s=143441",

"previewUrl":"http://a1099.itunes.apple.com/r10/Music/f9/54/43/mzi.gqvqlvcq.aac.p.m4p",
"artworkUrl60":"http://a1.itunes.apple.com/r10/Music/3b/6a/33/mzi.qzdqwsel.60x60-50.jpg",

"artworkUrl100":"http://a1.itunes.apple.com/r10/Music/3b/6a/33/mzi.qzdqwsel.100x100-75.jpg",
"collectionPrice":10.99,
"trackPrice":0.99,
"collectionExplicitness":"notExplicit",
"trackExplicitness":"notExplicit",
"discCount":1,

Apple [4]com os do Echo Nest então precisei utilizar a funcionalidade de Busca da Apple. A API permite buscar uma música pelo nome dela e retorna informações sobre a mesma. Assim pude comparar pelo nome da Música e do Artista provido pelo Echo Nest para assim achar a Música. O preview do áudio está na parte "previewUrl" retornado por json pela Apple. Exemplo de retorno da API em json:

## 7.2 Software e Banco de Dados

Para armazenar os dados coletados, filtrar, marcar a separação dos conjuntos de Treino, Teste e Cold foi utilizado o Banco de dados Sqlite3. Isso facilita manter armazenada uma configuração de um experimento e acessar os dados de forma relacional diferente do arquivo texto inicial do EchoNest. Nele, armazenamos somente a parte que foi utilizada pela Fatoração e pela Rede. Dessa forma, a filtragem do Conjunto Inicial só precisou ser realizada uma vez. É necessário o uso de um Banco de Dados para facilitar a separação dos conjuntos e análise dos dados.

Python como Linguagem de programação é usada em todo o projeto. A escolha do Python é favorecida por existirem inúmeras bibliotecas excelentes para desenvolvimento de Machine Learning. Apesar de ser uma linguagem interpretada de scripts, o que teoricamente a torna lenta para o uso de Machine Learning. Porém, ela contém Bibliotecas que resolvem isso ao gerar arquivos compilados em C a partir de código em Python. O principal exemplo é o uso do *Theano* que permite criar funções e variáveis para serem armazenadas e executadas na GPU, além de compilar uma única vez as funções, o que torna o processamento extremamente paralelo e eficiente.



4 Apple – Empresa vendedora de Hardware e Software - "*Apple Inc*.", 22 Ag. 2016.

## 7.3 Modelo de dados

O Sqlite é um banco prático pois não depende de instalação, é em forma de um arquivo e é utilizado localmente. Isso torna as operações nesse banco fáceis e rápidas.

O Banco é composto por quatro Tabelas: Musica, Artista, Usuario, e UsuarioMusica. A Tabela Musica guarda os identificadores nas API's relativas, o nome, uma referência ao Artista e um identificador local. A Tabela Artista guarda o nome dele, os identificadores nas API's relacionadas e um identificador local. A Tabela Usuario guarda somente o identificador EchoNest deste Usuário e um identificador local. A Tabela UsuarioMusia é um relacionamento entre Musica e Usuario, ela é a mais importante pois somente ela é utilizada de fato dentro da execução dos Algoritmos, as outras servem como informações adicionais para análises e segmentação. Ela contém o identificador local da Música, o identificador local do Usuário, a quantidade de vezes que  o Usuário ouviu a Música e em qual conjunto ela foi segmentada, ou Treino ou Teste ou Cold representadas como atributos booleanos. Além disso contém o caminho (*path*) onde o áudio se encontra fisicamente no HD. Segue abaixo, uma imagem (Figura 1) do Modelo Relacional criado.

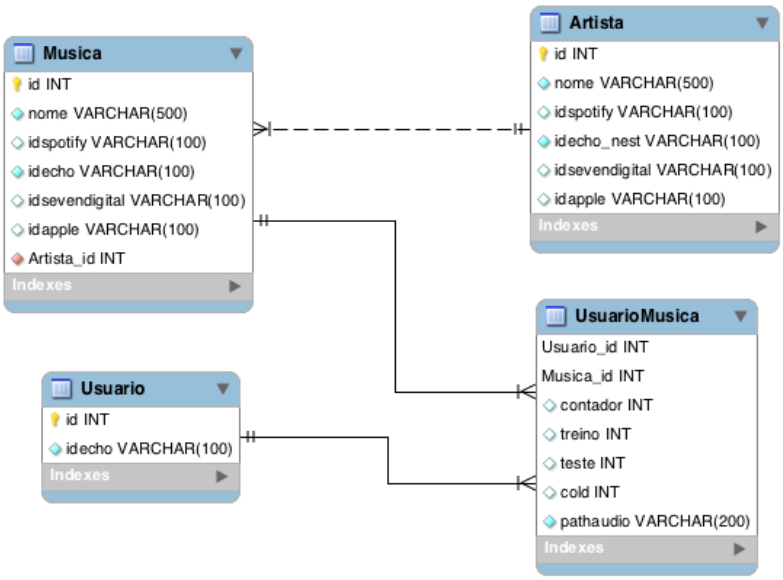


**Figura 5 -** Modelo ER

## 7.4 Bibliotecas do Python utilizadas

A biblioteca *Matplotlib* (Hunter 2007) é utilizada para gerar gráficos em formato de imagens. Permitiu acompanhar o aprendizado pelos Gráficos.

A biblioteca matemática *Theano* (T. D. Team 2016), que compila códigos em Python, que devem ser na linguagem que ele aceita de forma Matemática, compila o Código em C para acelerar a computação e permite escolher a GPU para processar, utilizando o CUDA.

A biblioteca Lasagne, baseada em *Theano* (T. D. Team 2016), que facilita a implementação de Redes Neurais.

A biblioteca Nolearn, baseada no *Lasagne* (L. contributors 2016), que abstrai ainda mais um nível de implementação de Redes Neurais.

As bibliotecas matemáticas *Scipy* (Jones E 2001) e *Numpy* (Developers 2016), que podem compilar o código em C, otimizam o desempenho e fornecem estruturas e funções para facilitar o desenvolvimento de códigos principalmente Matemáticos.

A biblioteca *Scikit-learn* (2011Scikit-learn Corp.), criada para Machine Learning, oferece inúmeras funções para auxiliar o desenvolvimento de aplicações em Machine Learning. Desde Algoritmos de treino completos e normalização até cálculos de métricas.

A biblioteca publica *H5py* (A. C. contributors 2006), que permite salvar objetos python em arquivo de forma eficiente de guardar e recuperar.

A biblioteca *Pickle* (Pickle - Python Software Foundation 2016), cuja utilidade é parecida com a do *H5py* (A. C. contributors 2006), mas suporta mais tipos de objetos Python, em compensação é mais lenta.

## 7.5 *Preview* de Músicas

Para a obtenção das músicas do *Echo Nest Dataset* foram utilizadas APIs públicas de três empresas: *7Digital, Spotify e Apple*. Foram obtidos 243.948 áudios de cerca de 30 segundos cada e em seguida cortados em 6 segundos cada pois observou-se ser sufiente para aprender e porque quanto maior o áudio maior a imagem representativa de espectrograma utilizada na rede neural o que torna mais lento o treino. São previews livres de direitos autorais pois não contém o Áudio completo, autorizados para uso livremente.

Para a obtenção dos Áudios foi executada um passo-a-passo sequencial. Primeiramente obtem-se a entidade Musica pelo EchoNest. Com isso pode-se ou não já conseguir o áudio. Em seguida com o identificador *7digital* ou *Spotify* podem ser obtidos os Áudios que faltaram, essa relação de identificadores veio da primeira requisição a API do *7Digital*. Por último ainda foram tentadas obter mais áudios pela API da *Apple* que não teve uma relação direta de identificadores com nenhuma API anterior então foi necessário utilizar uma estratégia de busca na API da *Apple* pelo nome da Música e nome do Artista.

## 7.6 Etapas da Obtenção

1. Separados os identificadores de música do *Echo Nest*
2. Obtenção das chaves OAuth2 da API para desenvolvedores do *Echo Nest*. Essa permissão possibilita efetuar até 86.400 requisições POST por dia. Essas requisições retornam dados sobre a música como o título, artista e chave externa de identificação das mesmas no *Spotify* e no *7Digital*.
3. Ao utilizar as APIs públicas do *Spotify* e do *7Digital*, tentou-se obter o preview das músicas do dataset do *Echo Nest*. Até então, foram obtidas 190.000 previews de músicas.
4. Poucas empresas fornecem APIs para obtenção de preview. Sendo assim, para tentar obter mais músicas e estender o dataset, usou-se o webservice da *Apple* de busca de informações sobre músicas que fornece um link para acesso ao preview dado um nome de música e artista. Assim, conseguiu-se mais 47.810 previews, resultando em um dataset final com 243.948 trechos de músicas.

## 7.7 Tratamento de Dados

O Dataset inicial continha cerca de 48 milhões de informações de consumo. Esse dataset foi reduzido, resultando apenas em todas as informações de consumo relativas às músicas das quais foram obtidas o preview. Levando em conta que informações de consumo onde não se tem o *preview* são irrelevantes para a tarefa.

Devido ao tamanho do Dataset e dos passos para a obtenção de todos os dados, uma simples interação tratando sequencialmente cada informação de consumo seria custosa, demorando cerca de 2 meses para ser realizada. Utilizou-se a biblioteca *Pandas* do Python que permitiu a leitura de todo o dataset e tratamento das informações em cerca de 2 horas. Contudo, a demora para a obtenção dos áudios relativo ao limite de requisições pelas APIs continuou existindo, totalizando cerca de 3 meses para a obtenção dos *previews*.

# 8 Experimentos

Nesse Capítulo, são relatados os detalhes dos experimentos realizados bem como o funcionamento das Redes Neurais Convolucionais.

As redes neurais convolucionais (CNN´s) foram utilizadas recentemente para melhorar o estado da arte em problemas de reconhecimento de fala, e classificação de imagens em grande escala, com uma grande taxa de acerto. Essa evolução é principalmente devido a três fatores:

- Usando unidades retificadoras lineares (Relus), em vez de sigmoides não-lineares leva a convergência mais rápida e reduz o problema do "escape gradiente" que persiste em redes neurais tradicionais com muitas camadas.

- Paralelização é usada para acelerar o processo, de modo que modelos maiores possam ser treinados em uma razoável quantidade de tempo. Foi utilizada a biblioteca *Theano* do *Python* para tirar vantagem da aceleração da GPU que utiliza internamente a biblioteca CUDA da *Nvidia*[5].

- Uma grande quantidade de dados de treinamento é necessária para ser capaz de implementar modelos grandes com muitos parâmetros. O Dataset inicial contém dados de treinamento suficiente para ser capaz de treinar grandes modelos de forma eficaz.



5 Nvidia – Empresa fabricante de Placas de Vídeo - "*Nvidia Corp*.", 22 Ag. 2016.

## 8.1 Convolutional Neural Networks

Redes Neurais Convolucionais são uma implementação especifica do Conceito de Redes Neurais de Múltiplas Camadas (*Deep Learning*). Elas têm se provado capazes de extrair características de imagens com uma grande taxa de acerto, com qualquer tipo de imagem em problemas de Classificação e Regressão. A proposta do uso desse algoritmo é verificar a possibilidade de extrair características dos Áudios relevantes para a recomendação. Para isso é necessário transformar os Áudios em Imagens (Espectrogramas) e tentar aproveitar essa qualidade desse tipo de algoritmo para a tarefa proposta.

As CNNs (Figura 6) permitem que características intermediárias possam ser compartilhadas entre diversos fatores, e sua estrutura hierárquica composta por camadas alternadas de extração de características e de reagrupamento, *pooling*, permitem operar em múltiplas escalas de tempo.

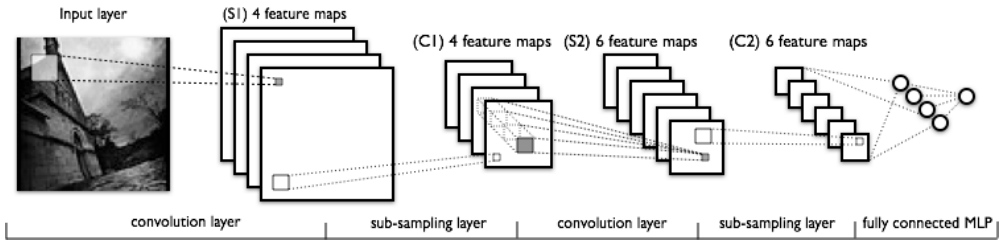


**Figura 6 -** CONVNET

Essas redes possuem muitos parâmetros de aprendizagem e exigem uma grande quantidade de processamento para ajustar todos os parâmetros. O que tornou viável o treino dessas redes foi a possibilidade do uso de Placas Graficas para o processamento do Treino.

A tarefa proposta é encontrar os Fatores Latentes, tendo sido modelada como uma tarefa de Regressão, via aprendizado supervisionado. Para cada imagem, no Treino, a rede recebe em conjunto os Fatores Latentes que deve aprender. A Rede é desenvolvida por camadas sequenciais onde no meu caso utilizei Camadas de Convolução, Pooling e Dense Layers e Dropout Layer. Cada uma possui particularidades que contribuem para o aprendizado. O Áudio entra pela primeira camada e percorre a rede sequencialmente pelas camadas até o fim. A seguir, apresentamos a definição do que faz cada tipo dessas camadas.

### 8.1.1 Camada de Convolução

A camada de convolução tem como objetivo aplicar filtros na imagem para extrair características. Esses filtros são treinados para que se encontre as características desejadas na construção da rede. A ativação dos filtros determina as características que são obtidas pela rede.

A primeira camada obtém características da imagem de baixo nível como bordas e linhas. A seguir, as camadas subsequentes obtêm características de mais alto nível. Quanto maior a profundidade da rede, mais níveis hierárquicos de características são reconhecidos. Gênero musical por exemplo pode ser considerado uma característica de alto nível hierárquico, portanto precisa de muitas camadas para ser aprendido.

Para ilustrar a convolução, segue abaixo uma imagem representativa na Figura 7:



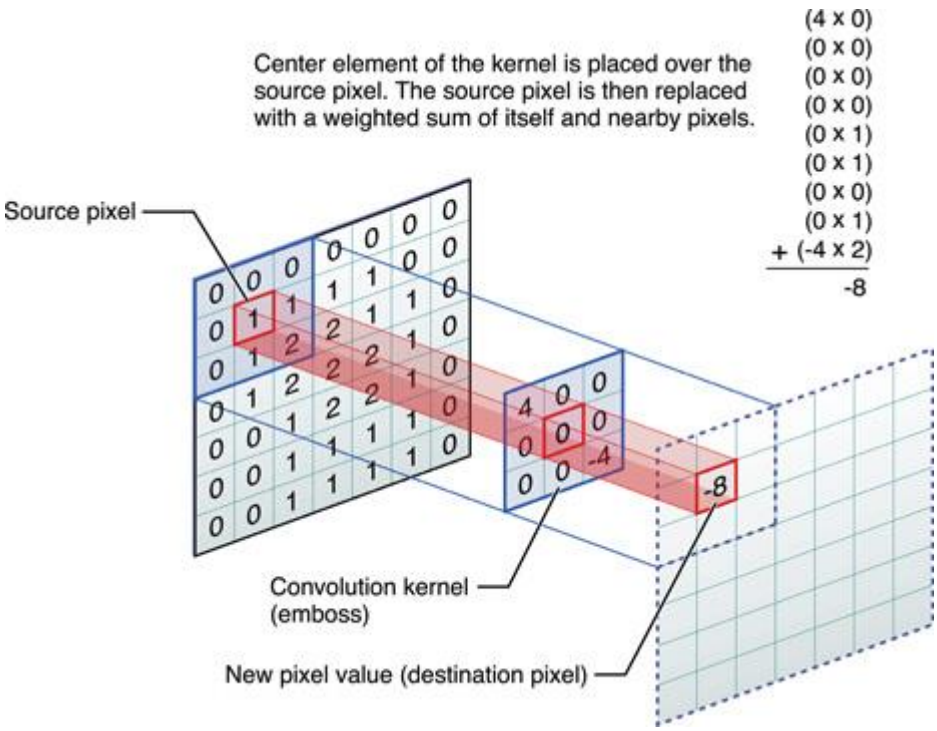


**Figura 7 -** Convolução

As camadas de covolução possuem parâmetros chamados de Kernels. Os Kernels são associados a cada parte da imagem. Os kernels são parâmetros

ajustáveis que servem para aprender as características da imagem de forma independente de onde essas características apareçam na imagem.

### 8.1.2 Camada de Pooling

Usadas para reduzir a variância, camadas de pooling calculam o valor máximo ou médio de uma característica particular ao longo de uma região da imagem. Isto irá assegurar que o mesmo resultado será obtido, mesmo quando as características de imagem têm pequenas alterações. Esta é uma operação importante para a classificação de objetos e de detecção. Ilustrada na Figura 8.

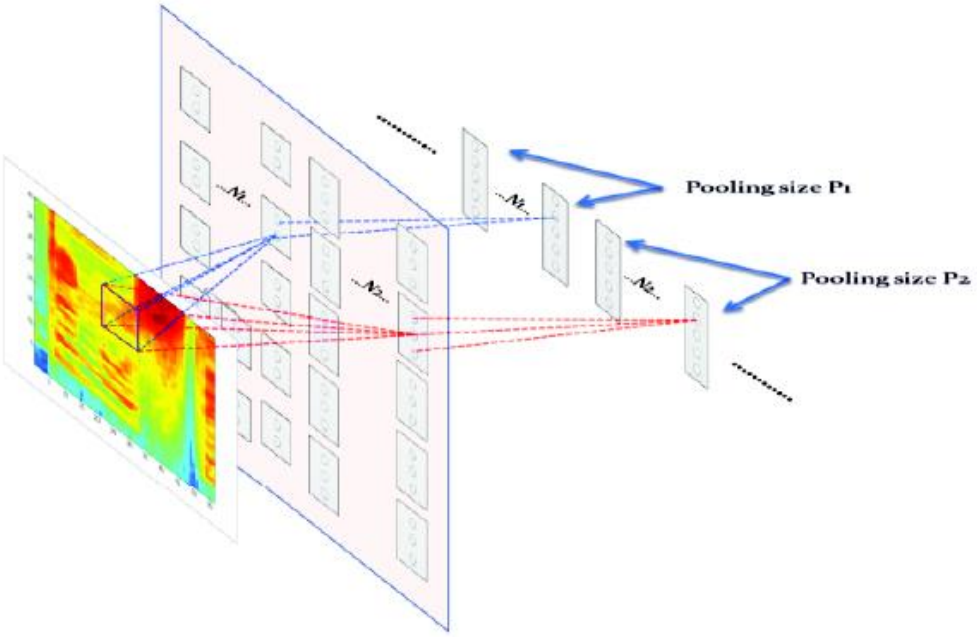


**Figura 8 -** Pooling

### 8.1.3 Dropout

O Dropout serve como uma forma de regularização, ou seja, ele intencionalmente desativa neurônios da rede com o intuito de reajustar as saídas das camadas. Podem ser utilizados após qualquer camada da rede. Com isso ele permite efetuar de forma simplificada uma média de valores regularizando a saída da camada.

### 8.1.4 Dense Layer

Nas Camadas Densas são aplicados o conceito de *Backpropagation.* Nessas camadas o principal objetivo é relacionar os parâmetros aprendidos pela rede com a saída esperada e ajustar os parâmetros para corrigir erros. Funcionam como uma função onde existe uma entrada que são os parâmetros da Rede e uma Saída que deve ser a saída esperada inicialmente. Por ser supervisionado ele compara a saída dessa função com a esperada e corrige os parâmetros da Rede para que se adequem.

## 8.2 Normalização

A Normalizacao consiste em um processo a ser realizado nos dados de entrada da Rede antes do processamento. O objetivo desse passo é generalizer o modelo encurtando a variação dos valores de entrada ajustando valores muito altos ou muito baixos.

Esse processo pode ser aplicado tanto nas imagens quanto nas *labels* de entrada da rede. No caso das imagens ele centraliza na origem nivelando os pixels da imagem por um valor médio. Já no caso das *labels* onde nesse projeto são os Vetores Latentes, a Normalizacao tem o papel de colocar cada elemento dentro de um intervalo fixo entre -1 e 1.

A formula da normalização das labels é:

$$X' = \frac{\mathrm{X} - \min(\mathrm{X})}{\max(X) - \min(X)}$$

Onde X é o valor original e min(x) e max(x) são os valores mínimos e máximos possíveis para X respectivamente.

O objetivo dessa etapa é tornar o Treino mais rápido e com uma maior qualidade, pois as funções aplicadas nas camadas aprendem melhor em intervalos pequenos.

## 8.3 Translação

A translação é uma etapa opcional que consiste em aumentar a quantidade de dados movendo os pixels das imagens em ambos os eixos. Isso pode ser utilizado para aumentar o tamanho do Conjunto de Treino e também deixar o modelo mais robusto e com menos chance de *overfitting*. Além da translação existem outras formas de pre-processsamento em Imagens que podem ser uteis no treino mas não irei aboradar nesse Projeto. Cito aqui somente a Translação pois no meu caso dos Espectrogramas foi a única que usei por ser uma boa prática com objetivo de trazer uma melhoria no aprendizado, não cheguei a medir a diferença de qualidade sem usar a translação.

## 8.4 Funções de ativação

A função de ativação é executada em cada neurônio. Dada uma entrada ao neurônio essa entrada passa pela função que tem como saída um número entre -1 e 1. Essa saída é passada para neurônios das camadas seguintes como entrada, ao aprender a entrada de cada neurônio a rede aprende a tarefa na etapa de treino.

Uma das funções de ativação dos neurônios mais utilizadas para Redes Convolucionais é a RLU pois apresenta um ótimo resultado em uma escala entre 0 e 1 de forma rápida. No modelo proposto essa função se saiu bem mas a melhor função testada foi a Tangente Hiperbolica ou *tanh* como termo mais conhecido ela está representada na figura 9 abaixo. Essa analise foi feita testando o uso das

funções no modelo e checando o Conjunto de teste além da verificação da velocidade e rapidez de aprendizagem.

A *tanh* é uma função não linear com a seguinte fórmula:

$$f(x) = \tan(x) = \frac{2}{1 + e^{-2x}} - 1$$

Onde tan(x) é a tangente calculada para o valor x. Os resultados sempre ficam entre -1 e 1.

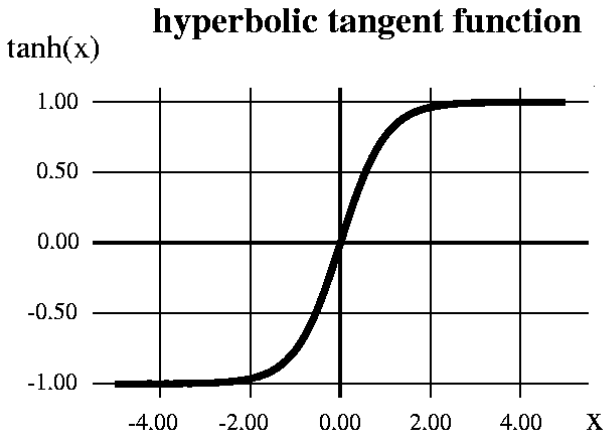


**Figura 9 -** Tanh

## 8.5 Arquitetura da Rede Utilizada

A variação na arquitetura, quantidade de neurônios, tamanho dos filtros na parametrização das camadas e tipos de camadas influencia diretamente no resultado do aprendizado. No projeto diversas arquiteturas foram testadas até chegar em uma com um bom resultado. Como foram muitos testes de camadas eu não citarei cada um deles, o que aprendi de mais importante é que quanto mais profunda, ou seja, quanto mais camadas mais a rede aprende. A arquitetura a seguir contem 1.497.394 parâmetros de aprendizado e 19 camadas incluindo a entrada. Segue abaixo a arquitetura da rede utilizada:

1. (InputLayer, {'shape': (datasetsize, 1, 128, 128)}),
2. (Conv2DLayerFast, {'num_filters': 32, 'filter_size': (3, 3)}),
3. (Conv2DLayerFast, {'num_filters': 32, 'filter_size': (3, 3)}),
4. (Conv2DLayerFast, {'num_filters': 32, 'filter_size': (3, 3)}),
5. (Conv2DLayerFast, {'num_filters': 32, 'filter_size': (3, 3)}),
6. (Conv2DLayerFast, {'num_filters': 32, 'filter_size': (3, 3)}),
7. (MaxPool2DLayerFast, {'pool_size': (2, 2)}),
8. (Conv2DLayerFast, {'num_filters': 64, 'filter_size': (3, 3)}),
9. (Conv2DLayerFast, {'num_filters': 64, 'filter_size': (3, 3)}),
10. (Conv2DLayerFast, {'num_filters': 64, 'filter_size': (3, 3)}),
11. (MaxPool2DLayerFast, {'pool_size': (2, 2)}),
12. (Conv2DLayerFast, {'num_filters': 128, 'filter_size': (3, 3)}),
13. (Conv2DLayerFast, {'num_filters': 128, 'filter_size': (3, 3)}),
14. (Conv2DLayerFast, {'num_filters': 128, 'filter_size': (3, 3)}),
15. (MaxPool2DLayerFast, {'pool_size': (2, 2)}),
16. (DenseLayer, {'name': 'dense1', 'num_units': 64}),
17. (DropoutLayer, {}),
18. (DenseLayer, {'name': 'dense2', 'num_units': 64}),
19. (DenseLayer, {'name': 'output', 'num_units': 50, 'nonlinearity': tanh}),

Ao longo do treino os parâmetros da Rede são atualizados ao final de cada época. O gráfico a seguir na Figura 10 demonstra o andamento da aprendizagem e é útil para checar se a Rede realmente está aprendendo. Na horizontal está representado o tempo, na vertical o erro e a curva presente no gráfico demonstra o quanto a execução errou ao longo tempo. Logo o ideal é que seja uma curva sempre decrescente.

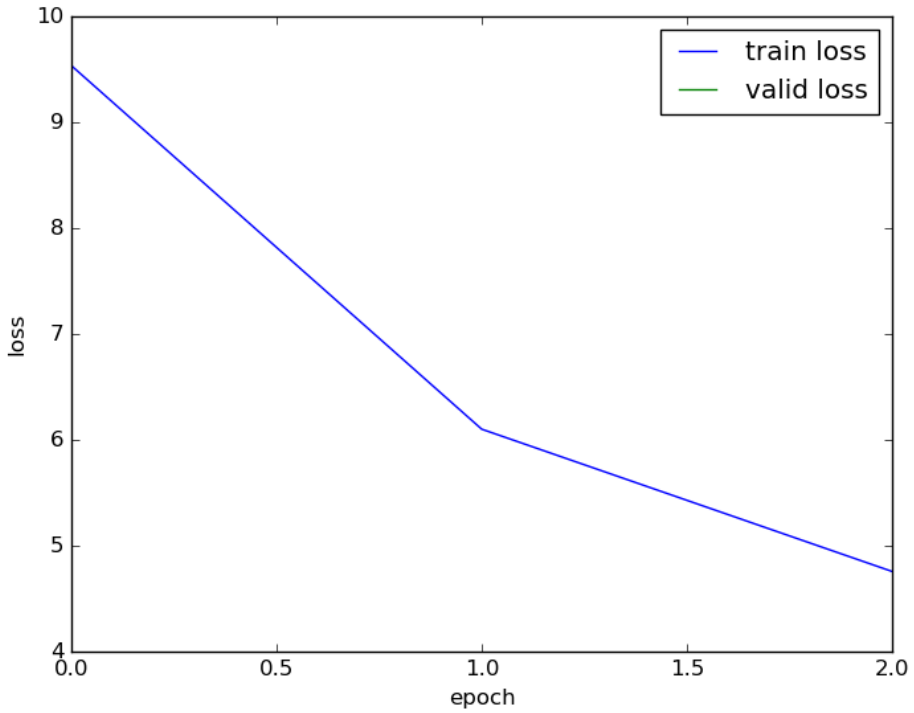


**Figura 10 -** Gráfico do aprendizado

Para ilustrar uma rede com um desempenho insatisfatório por exemplo temos:

1. **(InputLayer, {'shape': (datasetsize, 1, 128, 128)}),**
2. **(MaxPool2DLayerFast, {'pool_size': (2, 2)}),**
3. **(Conv2DLayerFast, {'num_filters': 128, 'filter_size': (3, 3)}),**
4. **(MaxPool2DLayerFast, {'pool_size': (2, 2)}),**
5. **(DenseLayer, {'name': 'output', 'num_units': 50, 'nonlinearity': tanh}),**

Ao executar a visualização do gráfico de aprendizado pode-se ver claramente que a Rede logo para de aprender. Isso se deve ao fato da rede ter poucas camadas de covolução e muitas de Pooling para essas imagens com dimensões altas. E ainda ter pouca profundidade, o que ocasiona aprender somente caracteristicas básicas, assim não consegue aprender as caracteristicas Latentes que são caracteristicas mais abstratas necessitando o uso de uma rede mais profunda. A seguir um gráfico que demonstra esse processo de aprendizagem incorreto para a solução na Figura 11:

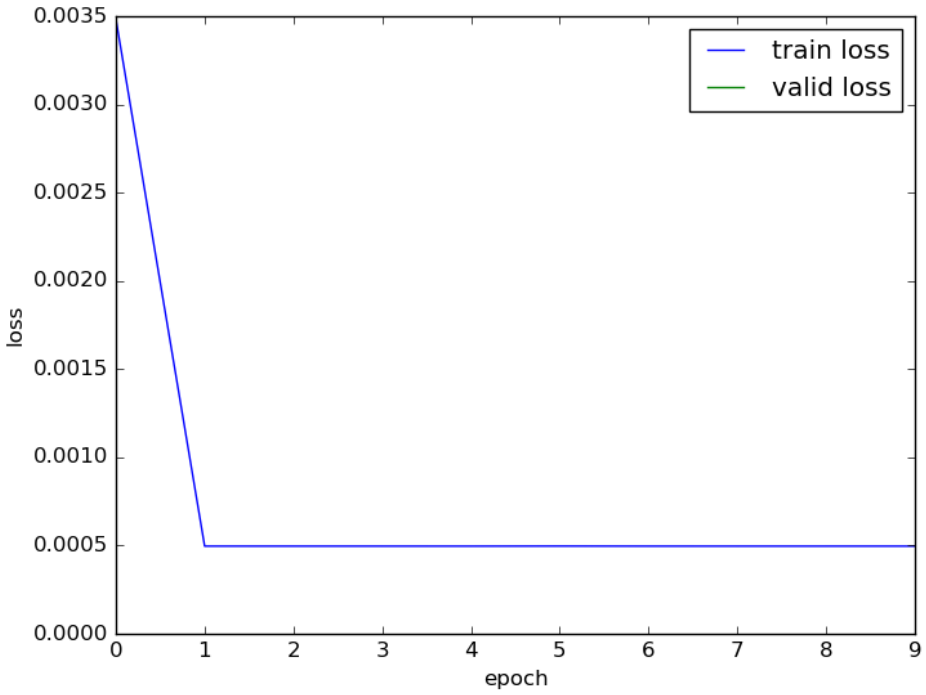


**Figura 11 -** Gráfico de Aprenizagem 2

Ao analizar o gráfico conclui-se que em momento inicial a rede para de aprender, logo a arquitetura nesse caso não está adequada.

## 8.6 Exemplo de execução

Ao executar utilizando a Biblioteca Nolearn no início já é mostrada a arquitetura, parâmetros, capacidade, dentre outras características. Ela fornece

informações sobre cada camada, como capacidade, filtros e ainda mostra se existe algum possível problema de arquitetura colorindo camadas quando elas tendem a não aprender.

## 8.7 Treino da Rede

No Conjunto Completo, a rede demorou cerca de 8 horas para treinar. Foram 80 épocas cada uma, passando por todo o Conjunto de Treino.

Para o pré-processamento foi utilizada a Biblioteca *Scikit-Learn* do Python. As características obtidas pela fatoração são colocadas em uma escala entre -1 e 1.

Em seguida, cada Vetor de fatores Latentes das Músicas é associado aos seus respectivos espectrogramas para formar a entrada da Rede.

A função de ativação das camadas é a Tangente. Para atualização dos parâmetros utilizamos o Stochastic Gradient Descent (SGD), através do *Backpropagation Algorithm*. Outras técnicas de atualização não foram testadas pois o autor do artigo Base (Aaron van den Oord 2014) utilizou essas e para obter o mesmo experimento utilizou-se essas. Além disso O SGD é considerado o melhor para esse tipo de tarefa. Ao aplicar os filtros nas Camadas de Convolução a rede aprende quais filtros devem ser ativados para corresponder a entrada.



O aprendizado dos filtros revela as partes significativas do Áudio úteis para recomendação. A Biblioteca Nolearn permite vizualizar os filtros e ainda a dificuldade na aprendizagem de características nas imagens para cada camada.

## 8.8 Transformação de Áudio em Imagem (Espectrograma)

A transformação do áudio em imagem é uma forma de aproveitar o alto desempenho do treinamento de redes neurais convolucionais em imagens em uma tarefa de recomendação de áudio.

A transformação do áudio é feita pelo cálculo da transformada de Fourier em janelas de frequência. Cada transformada constitui um frame. Os frames sucessivos são concatenados para formar uma matriz que dá origem ao espectrograma do

áudio. Diversos tipos de transformação foram testados, ao final foi utilizado o *Mel Log Spectrogram.* Seu uso se deve a dois fatores, o primeiro para reprodução do artigo base (Aaron van den Oord 2014) e segundo por aapresentar melhores resultados no treino. No meu caso utilizei imagens de 128x128 pixels para cada Música.

No eixo horizontal está representado o tempo, na vertical as frequências e a amplitude é representada pela intensidade de cor presente na imagem.Utilizei o Log Mel Power pois ele apresentou os melhores resultados no treino.

## 8.9 Recomendação

A recomendação se torna possível com o uso dos vetores de características obtidos pelo áudio das músicas.

Para descobrir se uma música é uma boa recomendação para um usuário pode ser calculada a multiplicação entre os vetores da música e do usuário. Se a distância for pequena é uma boa recomendação. Da mesma forma, se a distância entre duas músicas é pequena elas provavelmente são parecidas. Essa distância que me refiro é a multiplicação dos vetores do Usuário pela Música. Isso gera um número que representa a preferência do Usuário pela Música, quanto maior mais provável o Usuário gostar da Música. Quanto a distância entre duas Músicas pode ser calculado pela distância entre os vetores delas, essa distância dos vetores quanto menor mais provável que as Músicas tenham características parecidas.

Em resumo, para predizer a preferência de um usuário por uma Música basta multiplicar seus vetores Latentes:

$$\mathbf{U \times M = P}$$

Onde U representa o vetor Latente de um Usuário. M representa o vetor Latente de uma Música e P a preferência do Usuário pela Música.

Sendo assim, como o Dataset inicial informa exatamente essa informação pode-se comparar o desempenho e qualidade das recomendações geradas pelo Projeto. Os resultados a seguir foram gerados pela métrica *Mean Average Precision* (MAP) descrita anteriormente. Nesse caso quanto maior o valor obtido melhor

foram obtidas as recomendações. O resultado está apresentado para os dois experimentos. Como objetivo a ser alcançado e como referencia foi colocado o melhor resultado de recomendação possível que foi utilizando a MAP no Conjunto de testes com a Fatoração de Matrizes.

# 9 Resultados

Os resultados dos experimentos podem ser analisados de diversas formas. A principal forma é a utilização de métricas definidas matematicamente de forma padronizada. Outra forma por exemplo é testar os resultados com Usuários reais verificando suas opiniões de forma qualitativa. Além disso cada métrica padronizada tende a analisar o resultado por um único aspecto.

Como utilizei um artigo inicialmente como base (Aaron van den Oord 2014) para construir meus experimentos resolvi utilizar a métrica *MAP*, definida aqui anteriormente, para comparar os resultados. Além disso resolvi desenvolver um site que utiliza os resultados dos experimentos para analisar de forma qualitativa a satisfação dos Usuários com as recomendações.

Nesse Capítulo apresento os resultados das métricas e uma demonstração básica do uso do site. Infelizmente não pude calculcar métricas qualitativas quanto a satisfação no site, mas obtive informalmente uma resposta positiva das pessoas que utilizaram. Quanto a métrica *MAP* consta relatado abaixo a obtida dos experimentos comparada com o Estado da Arte que foi obtida pelo artigo de base.

Resultado gerado pela *MAP* para o Conjunto contendo somente uma parte dos dados descritos na sessão 6.1.1, com o foco nas Músicas Populares:

### 9.1.1 Conjunto Pequeno

| **Experimento** | **Treino** | **Teste** *(Warm Start)* | **Novas** *( Cold Start)* |
|---|---|---|---|
| Conjunto Pequeno | 0.0719 | 0.0263 | 0.0246 |
| Estado da Arte | - | 0.0501 | - |
| WMF *(Upper Bound)* | - | 0.2531 | - |

**Tabela 2 -** Resultado Conjunto Pequeno

### 9.1.2 Conjunto Completo

Resultado gerado pela *MAP* para o Conjunto contendo todos os dados que foram obtidos descitos na sessão 6.1.2.

| **Experimento** | **Treino** | **Teste** *(Warm Start)* | **Novas** *( Cold Start)* |
|---|---|---|---|
| Conjunto Completo | 0.0069 | 0.0024 | 0.0011 |
| Estado da Arte | eristiccas | 0.00672 | - |
| WMF *(Upper Bound)* | - | 0.0786 | - |

**Tabela 3 -** Resultado Conjunto Completo

### 9.1.3 Análise do Resultado

O resultado da Métrica MAP revela que o aprendizado somente com o Áudio com intuito de predizer os fatores ainda está fraco em relação a Fatoração de Matriz. Apesar disso deve-se considerar alguns fatores importantes. As carcterísticas que estão sendo aprendidas podem não necessariamente estar no Áudio pois a Fatoração não considera o Áudio, ou seja, características que não estão no Áudio não são aprendidas com uma boa eficácia. Considera-se também que o método permite utilizar somente o Áudio diferente da Fatoração que não recomenda sem informações previas. Com isso para os casos do *Cold Start* esse método se torna muito útil por possibilitar essas recomendações.

## 9.2 Site (Genial Mind)

Ao final dos experimentos avaliamos os resultados de forma qualitativa independente das métricas e checar exemplos de uso do resultado desse Projeto.

Existem diversas formas de uso para os Fatores Latentes resultantes do Projeto. Esses fatores identificam a Música e ainda trazem informações relevantes. A partir deles podem ser criadas ferramentas não somente de recomendação, filtragem por certos fatores por exemplo, distância entre as Músicas, só depende da imaginação de quem for utilizar.

O site Genial Mind foi construído visando demonstrar alguns usos possíveis das técnicas aqui apresentadas. Considerando a parte que efetivamente utiliza a recomendação gerada pelo Projeto o site contém duas ferramentas. Recomendação de Músicas baseada em uma Música e Recomendação de Músicas para um Perfil de Usuário. Todas as imagens relativas a Artistas e informações de conteúdo são providas pelo Spotify através de sua API para desenvolvedores.

Essas ferramentas Utilizam os Fatores do resultado do Projeto para recomendar e a recomendação ‘w feita no servidor que retorna seus resultados para a página. A API do Spotify foi excencial nesse caso para permitir gerar Playlists, e principalmente para possibilitar que o Usuário escute a Música.

Vale ressaltar que o processo todo descrito anteriormente foi executado antes e o site somente utiiza os resultados na forma de Fatores Latentes. Isso facilita o desempenho do site principalmente considerando que não é necessário repetir o processso para cada Usuário. O Site e as ferramentas serão melhor definidas a seguir.

### 9.2.1 Página Inicial

Essa é a página inicial do site que da acesso aos links para as ferramentas (Figura 23).

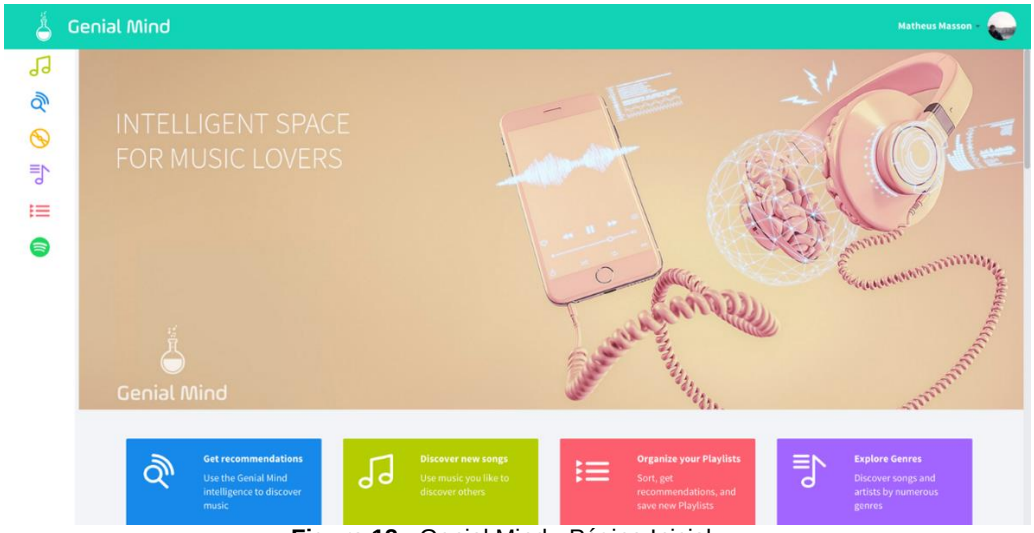


**Figura 12 -** Genial Mind - Página Inicial

### 9.2.2 Recomendação por Música

O objetivo dessa ferramenta é recomendar Músicas a partir de outras. Com isso o Usuário pode descobrir Músicas novas que tem características parecidas com Músicas que ele já conhece e gosta. O passo-apasso da interação do Usuário segue abaixo.

1. O Usuário entra na Página Música a partir do link na página principal (Figura 24)

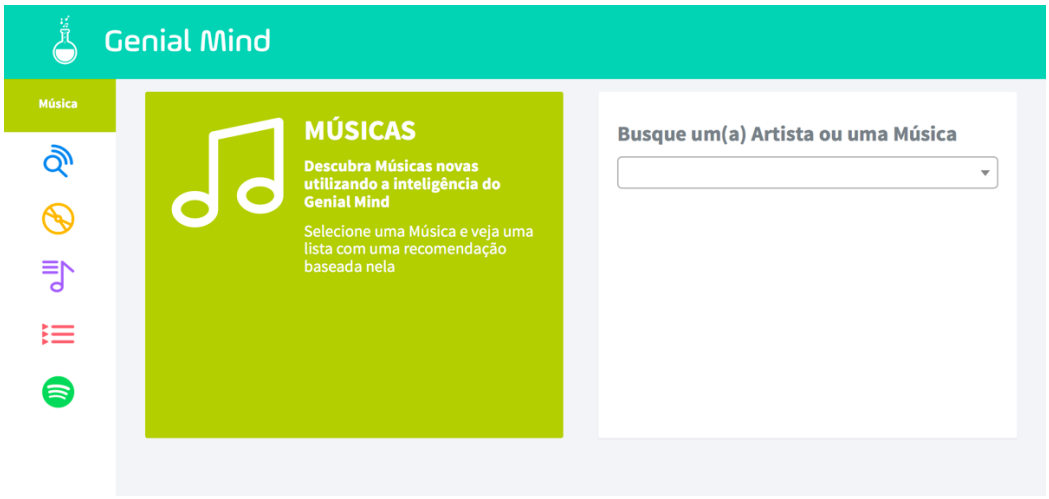


**Figura 13 -** Genial Mind - Resultado Música

2. O Usuário busca por uma Música a partir de um Artista ou diretamente pelo nome (Figura 25).

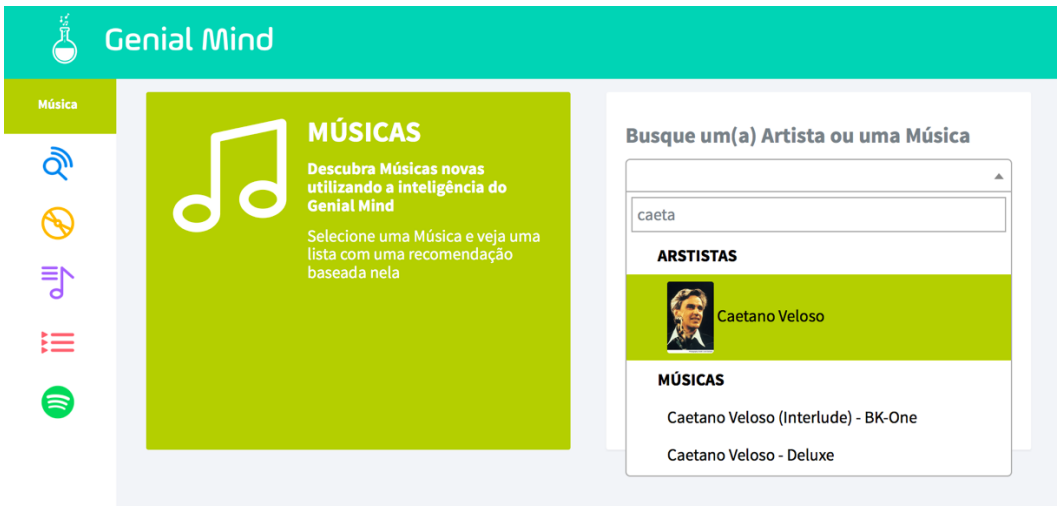


**Figura 14 -** Site, seleção de Música 1

3. O Usuário seleciona a Música que prefere e clica em Buscar (Figura 26).

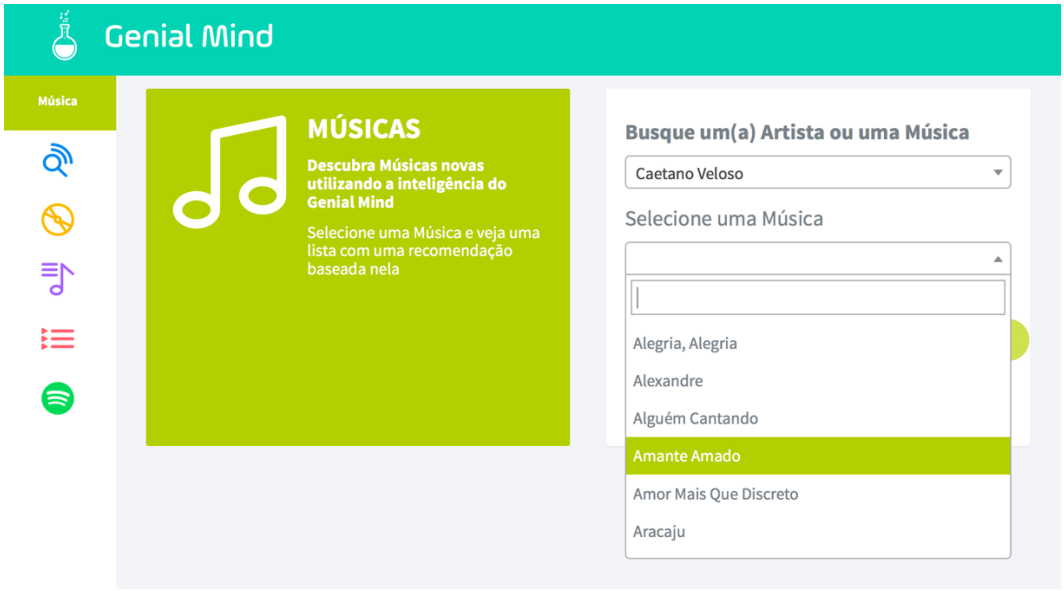


**Figura 15 -** Site, Seleção Música 2

4. O site retorna uma Playlist no formato do Spotify com as Músicas relacionadas (Figura 27).

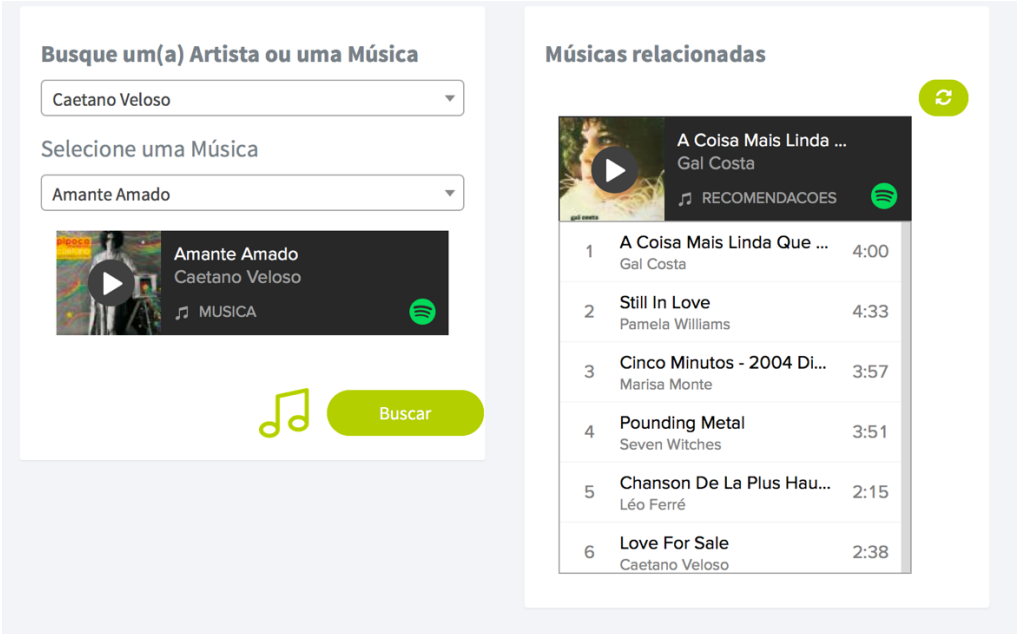


**Figura 16 -** Resultado Músicas

Essa ferramenta funciona da seguinte forma:

1. São armazenados os vetores Latentes de cada Música obtidos pelo Projeto
2. Ao escolher uma Música o site procura o vetor correspondente
3. Utiliza-se em seguida o algoritmo *Nearest Neighbor*
4. O algoritmo retorna as Músicas mais próximas
5. O site então retorna essas Músicas mais próximas em forma de lista para que o Usuário possa ouvir pelo Spotify utilizando a API do Spotify para exibir as Músicas.
6. Nesse caso então o que o site está retornando são as Músicas que tem características parecidas
7. Essa ferramenta não utiliza o resultado do *Cold Start* por ainda gerar resultados fracos, mas no futuro pretendo utilizar
8. Essa Funcionalidade foi colocada como exemplo do que pode ser feito com resultados de recomendação de forma geral

### 9.2.3 Recomendação por Perfil

Nessa parte o site recomenda uma lista de Músicas para o Usuário baseada no Perfil do Usuário obtido pelo Spotify das Músicas que ele já ouviu.

1. O Usuário deve se conectar ao Spotify utilizando o botão de conexão no canto superior direito da Página como na Figura 28.

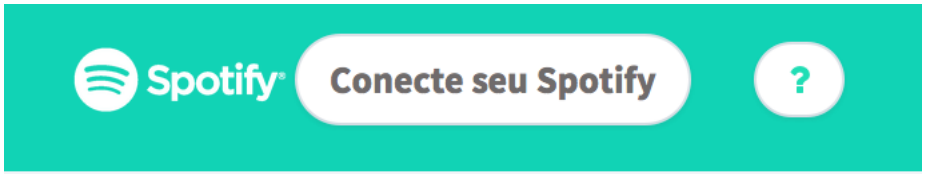


**Figura 17 -** Login Spotify

2. O Usuário é direcionado para a Página de Autorização de acesso aos seus dados pelo Site do Spotify.
3. Após aceitar o termo de consentimento de acesso a seus dados ele é direcionado devolta ao Site que então exibe o botão para gerar as recomendações personalizadas (Figura 29).

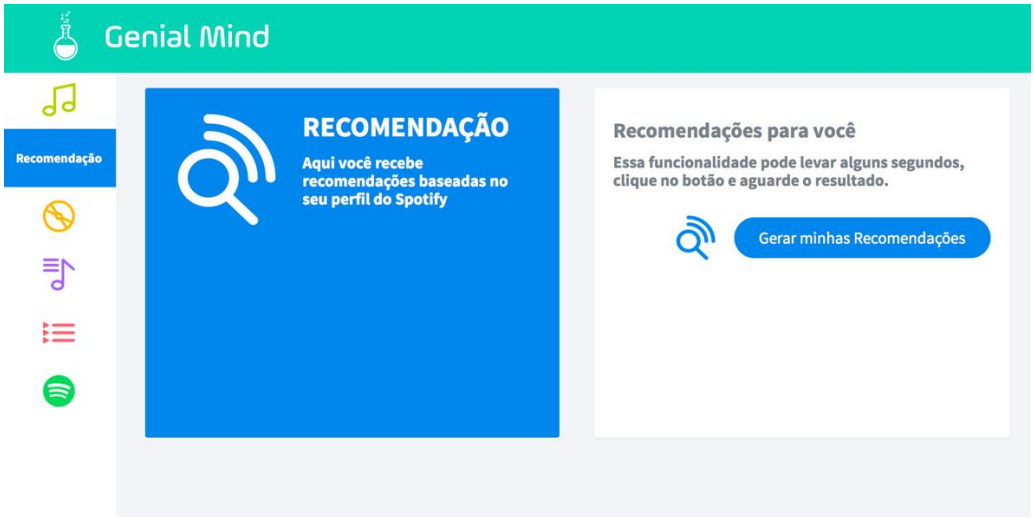


**Figura 18 -** Genial Mind – Recomendação 2

4. O site processa as recomendações baseando-se no Perfil do Usuário dentro do servidor e retorna uma lista de recomendações (Figura 30).

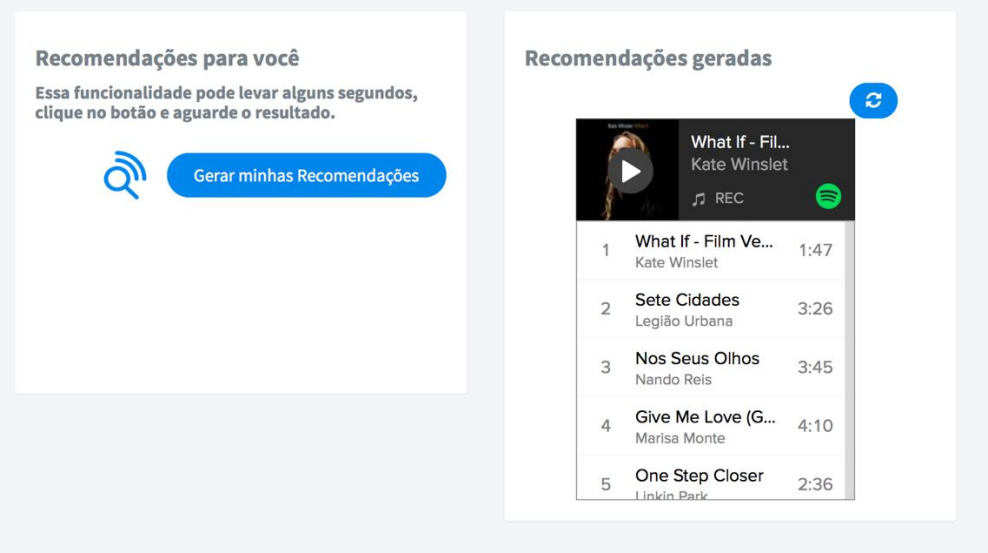


**Figura 19 -** Recomendacao Pronta

Essa ferramenta funciona da seguinte forma:

1. São obtidas as Músicas mais ouvidas e Playlists do Usuário Conectado ao site pela API do Spotify
2. Procura-se no banco de dados do site quais dessas Músicas foram utilizadas no Projeto, ou seja, quais Músicas do Usuário existem em forma de Fatores Latentes. Somente essas são filtradas
3. Em seguida é executada um ciclo de Fatoração de Matriz somente para esse Usuário resultando no vetor Latente do Usuário
4. Utilizando esse Vetor pode-se então recomendar uma lista de Músicas que o Usuário não ouviu

Com a criação desse site é possível demonstrar usos dos experimentos bem como avaliar as recomendações. Além disso, é um site com ferramentas úteis e interessantes para Usuários. Nele, é possível checar de forma pessoal se as recomendações estão retornando Músicas que fazem sentido. Além disso, permite obter avaliação de Usuários reais e comuns que acessam e testam as ferramentas. Essa ferramenta utiliza os fatores obtidos tanto pela fatoração quanto pela rede. Nesse caso não foram utilizados fatores do Cold por ainda terem uma qualidade baixa mas o objetivo é que no futuro isso possa ser feito como no exemplo anterior.

# 10 Conclusão

O resultado obtido demonstra que é possível recomendar somente com o Áudio e ainda propõe uma solução para o *Cold Start*. Apesar da Métrica relatar um resultado ainda longe da Fatoração ela já demonstra que ocorre um aprendizado. Assim sendo, pode-se obter características do Áudio relevantes para recomendação.

A recomendação de música através do áudio é uma tarefa que pode trazer inúmeros benefícios para artistas, usuários, empresas de comercio de músicas digitais. A combinação de resultados obtidos pelo áudio com resultados obtidos por filtragem colaborativa pode trazer ainda melhores resultados.

Os testes permitiram verificar algumas observações interessantes. A rede já conseguiu aprender algumas características dos áudios, características específicas e de nível de abstração baixo. Características como gênero, que pertence a um nível de abstração muito alto, precisam de muitos exemplos e de uma rede com muitas camadas. Como muitas características não puderam ser identificadas isso reflete no resultado baixo pois muitas delas podem refletir as preferências dos usuários testados.

A continuação desse projeto com as propostas apresentadas pode resultar em uma nova forma de recomendar e talvez ainda melhor que a Fatoração.

A seguir, enumeramos cinco aspectos que sugerimos a serem futuramente explorados no tema:

1. Aumentar o número de camadas da rede, permitindo obter características mais abstratas. (Bengio 2009)
2. Utilizar os filtros de alguma camada como os vetores das Músicas e não o resultado da Fatoraçao. (E. Humphrey 2013)
3. Obter mais dados de Músicas e Usuários.
4. Checar outros valores para as constantes das Funções Objetivo.
5. Desenvolver um modelo Hibrido que utilize características da Fatoração e do Áudio em conjunto. (L. M. de Campos 2010)

# 11 Referências bibliográficas

A. Krizhevsky, I. Sutskever, and G. E. Hinton. "**Imagenet classification with deep convolutional neural networks.**" In Advances in Neural Information Processing Systems , 2012.

Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen. "**Deep content-based music recommendation.**" (Electronics and Information Systems department (ELIS) (Ghent University)) 2014.

B. Shao, D. Wang, T. Li, and M. Ogihara. "**Music Recommendation Based on Acoustic Features and User Access Patterns.**" Audio, Speech, and Language Processing, IEEE Transactions on, 2009.

Bengio, Y. "**Learning Deep Architectures for AI.**" Foundations and Trends in Machine Learning, vol. 2, 2009.

Brian McFee, Luke Barrington, and Gert R. G. Lanckriet. "**Learning content similarity for music recom- mendation.**" IEEE Transactions on Audio, Speech & Language Processing, 2008.

Castillo, H. S. Del. "**Hybrid Content-Based Collaborative-Filtering music recommendations.**" Master's thesis, University of Twente, The Netherlands, 2007.

Chen, D. Agarwal and B. C. "**Regression-based latent factor models.**" Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '09, 2009.

Chen, H. C. Chen and A. L. P. "**A Music Recommendation System Based on Music Data Grouping and User Interests.**" Proceedings of the Tenth International Conference on Information and Knowledge Management, 2001.

contributors, Andrew Collette and. "**HDF5 - Hierarchical Data Format 5.**" 2006.

Cook, G. Tzanetakis and P. "**MARSYAS: a framework for audio analysis.**" Org. Sound, vol. 4, 1999.

Developers Team, NumPy. "**A package for scientific computing with Python.**" 2016.

E. Humphrey, J. Bello, and Y. LeCun. "**Feature learning and deep architectures: new directions for music informatics.**" Journal of Intelligent Information Systems, vol. 41, 2013.

E. J. Humphrey, J. P. Bello, and Y. LeCun. "**Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics.**" 13th International Society for Music Information Retrieval Conference, 2012.

Hinton, Laurens Vander Maaten and Geoffrey. "**Visualizing data using t-sne.**" Journal of Machine Learning Research,, 2008: 2579-2605.

Hinton, Vinod Nair and Geoffrey E. "**Rectified linear units improve restricted boltzmann machines.**" Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010.

Jason Weston, Chong Wang, Ron Weiss, and Adam Berenzweig. "**Latent collaborative retrieval.**" ro- ceedings of the 29th international conference on Machine learning, 2012.

Jones E, Oliphant E, Peterson P, et al. "**SciPy: Open Source Scientific Tools for Python.**" 2001.

L. M. de Campos, J. M. Ferna ndez-Luna, J. F. Huete, and M. A. Rueda-Morales. "**Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks.**" nternational Journal of Approximate Reasoning, 2010.

Lanckriet., Brian McFee and Gert R. G. "**Metric learning to rank.**" In Proceedings of the 27 th International Conference on Machine Learning, 2010.

Malcolm Slaney, Kilian Q. Weinberger, and William White. "**Learning a metric for music similarity.**" Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR), 2008.

Nouri, Daniel. "**nolearn: scikit-learn compatible neural network library.**" 2014.

O`.Celma. "**Music Recommendation and Discovery in the Long Tail.**" PhDthesis,UniversitatPompeu Fabra, 2008.

Pauws, M. Tiemann and S. "**Towards ensemble learning for hybrid music recommendation.**" Proceedings of the 2007 ACM conference on Recommender systems, 2007.

Pickle – "**Pickle**", Python Software Foundation. 2016.

Salomon, B. Logan and A. "**A Content-Based music similarity function.**" rep., Cambridge Research Laboratory, 2001.

Sander Dieleman, Benjamin Schrauwen. "**END-TO-END LEARNING FOR MUSIC AUDIO.**" Ghent University, 2014.

Team, PyData Development. "**Pandas Python data librarie.**" 2008.

Team, Theano Development. "**Theano: A Python framework for fast computation of mathematical expressions.**" 2016.

Y. Koren, R. Bell, and C. Volinsky. "**Matrix Factorization Techniques for Recommender Systems.**" Aug de 2009: 30–37.

Yifan Hu, Yehuda Koren, and Chris Volinsky. "**Collaborative filtering for implicit feedback datasets.**" Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, 2008.