Projeto Integrador V: QU Anime, modelo preditivo para qualificação de animes utilizando Multilayer Perceptron

Danilo Mative[1], **Fernando Alves**[1], **Victor Eleuterio**[1]

[1]Departamento de Ciência da Computação – Centro Universitário Senac (SENAC)
São Paulo – SP – Brasil

danilo.amative/eleuteriotrindade@gmail.com, fernandoimp@outlook.com.br

1. Introdução

O site *My Anime List* utiliza a nota de fãs para classificar e ranquear animações tipicamente japonesas entre 0 e 10. Observando as classificações sobre cada animação, levantou-se as seguintes dúvidas: Quais as relações entre as notas atríbuidas? Se faz possível predizer uma nota de acordo com os parâmetros de criação de cada animação?
Buscando obter a resposta, decidimos colher uma série de dados dos animes do site a fim da criação de uma base de dados separada por uma série de parâmetros sobre cada um, incluindo informações sobre a obra em si, como o gênero e fonte, além de características diretas da animação, como estúdio e tempo de duração do episódio.
A partir da base criada, ficou acertada a realização de testes com o algoritmo de inteligência artificial *MultiLayer Perceptron*, com a ideia de relacionar os fatores de criação da animação com a qualidade da nota recebida.

2. Objetivos

De acordo com o site *My Anime List*[My Anime List 2015], as notas das animações devem ser atríbuidas de acordo com o olhar do espectador, levando em consideração todas as emoções obtidas e pontos observados durante a visualização da obra. Acontece que, de acordo com a popularidade da obra, notas por pura empolgação e paixão podem ser atribuídas a mesma, ocultando fatores qualitatórios e dando-nos uma visão muito maior da quantidade de fãs do que da real qualidade da obra.
Com este trabalho, busca-se algo muito além da aplicação do *Multilayer Perceptron*, como também encontrar relações entre as notas, verificando como a variação dos valores se relaciona com a média final definida na base.

3. Materiais e Métodos

Para interação com o software desenvolvido, se fez necessário a utilização de alguns materiais e métodos, tais como uma linguagem padrão para desenvolvimento do algoritmo e equipamento específico para visualização do software.

3.1. Python

O *Python* foi escolhido como linguagem a ser utilizada devido a sua facilidade de adaptação aos códigos e grande número de ferramentas para diversos meios.
Em ênfase, a biblioteca *SciKit Learn* favoreceu demasiadamente a escolha da linguagem, uma vez que de acordo com sua documentação[INRIA and other 2015], tanto a utilização dos algoritmos de *Machine Learning* quanto o teste probabilístico dos mesmos, podiam

ser utilizados de maneira extremamente práticas, sendo somente com o envio da base de dados, ou da base em conjunto com as instâncias de entrada.
Além da quantidade de ferramentas, houve também a verificação da possibilidade de integrar o *Python* a outras linguagens, dado que o método de criação da interface gráfica ainda não estava definido. Com isso, a possibilidade de utilizar o código em conjunto com outras bibliotecas ou ser referenciado por outras linguagens cresceu significativamente.

3.2. Mac e Swift

Definido o *Python* como linguagem principal, verificamos que o mesmo tinha diversas semelhanças ao *swift*, tais como o método de escrita e padrões das entrada. Sendo assim, realizou-se um estudo de caso sobre a integração de ambos, no qual descobrimos ser uma possibilidade viável para aplicação.
Em virtude da fácil manipulação do *swift* para criação de interfaces gráficas e conhecimento pré-estabelecido na equipe para o mesmo, determinamos que sua utilização seria adequada para uma criação rápida e eficaz de entradas interativas com o usuário[Apple Inc.].
Fixada a ideia de utilização da linguagem para o UI, apurou-se um único problema com a escolha: A exclusividade em empregar o programa unicamente em dispositivos desktop ou notebooks da Apple. Após nos questionarmos sobre a situação, observamos que pelas principais funções terem sido empregadas em *Python*, a possibilidade de remanejar o código para outros sistemas poderia não ser um problema se tratada futuramente.

4. Descrição

O projeto foi divido em 4 partes: A coleta de dados utilizando Web Scraping em Python; Manipulação da base CSV para melhor interação com algoritmo; Aplicação do *Multilayer Perceptron* utilizando a biblioteca *SciKit Learn* em *Python*; Criação de interface gráfica para interação com usuário em *Swift*.

4.1. Web Scraping

Ao notar que o *My Anime List* oferecia APIs extremamente limitadas para o envio de dados, ficou acordada a realização de uma coleta por meio de varreduras sobre cada página do site, utilizando técnicas de *Web Scraping* por meio de um algoritmo construído em *Python*.
Para a construção do algoritmo, primeiramente fez-se necessário realizar uma análise do que deveria ser obtido, considerando as ofertas de dados do site e o que seria relevante para a análise.
As páginas do site alvo traziam uma série de informações, cada qual com potenciais diferentes para utilização e análise. Obviamente, por grande quantidade e variedade, algumas informações tiveram de ser descartadas, tais como dubladores e staff geral. Definido os descartes, passou-se a observar parâmetros que traziam mais de uma informação, como o gênero. Como algumas animações apresentavam poucos gêneros e outras mais de cinco, definiu-se que três dos gêneros escolhidos ao acaso dariam uma boa média, sendo uma quantidade nem tão baixa e nem grande demais. Qualquer obra com menos de três gêneros foi descartada para evitar nulos.
De acordo com essas escolhas, partimos para a visualização de casos únicos e de grande relevância, como estúdio, produtora, tempo de episódio, origem e afins. Realizando uma

verificação rápida no site, reparamos uma grande quantidade de estúdios que continham pouquíssimos animes, sendo que alguns possuíam até mesmo uma única animação. Para evitar a sobrecarga de estúdios, definiu-se que seria realizada a inclusão dos 29 principais, marcando qualquer outro como 'Estúdio não listado'.
Definido tudo que deveria ser coletado na busca foi realizada a instalação das bibliotecas *urllib* e *Beautiful Soup 4*, para solicitação da página e ordenação do código respectivamente.
Foram varridas 37565 páginas, uma vez descoberto que o último anime continha o ID 37565. Cada página foi buscada dentro de um *try*, evitando erros de *NotFound* para ID's não existentes. Para cada página localizada e salva, eram lidas as tags que continham a classe *dark_text*, cujo título se referia a informação demonstrada abaixo. Para coletar a informação fora da classe, fez-se necessário ir à tag pai(sem classe), e armazenar o texto das linhas abaixo.
Ao final do processo de leitura das 37 mil páginas, foram coletadas 4247 instâncias, todas inseridas num .txt com ID, status, produtor, estúdio, fonte, 3 gêneros, duração, classificação indicativa, número de episódios e nota separados por vírgula.

4.2. Base de dados

Embora o *Web Scraping* coletasse todas as informações necessárias, notamos que algumas alterações deveriam ser realizadas para melhora da leitura no algoritmo. Primeiro, realizamos o mesmo processo dos estúdios para os gêneros e produtores, mantendo somente aqueles que estavam contidos em grande quantidade de instâncias. Logo após, foram realizadas discretizações em valores de classes que eram definidas em grandes quantidades.
As discretizações foram realizadas de maneira a aglutinar os dados, gerando uma maior concentração e evitando *outliers*. Foram criados intervalos de distribuição para a quantidade de episódios, duração do episódio e nota.
Para a duração, as classes foram separadas em 16-26, 60+, 27-60, 6-15 e 0-5 minutos, tal como para a quantidade de episódios foram separados entre 1, 14-26, 100+, 53-100, 2-6, 7-13 e 27-52 episódios, já a nota, em 21 classes de acordo com a distribuição dos dados.

4.3. Multilayer Perceptron

O MLP(*Multilayer Perceptron*) foi aplicado em *Python* utilizando a biblioteca *SciKit Learn*, que contém diversas aplicações úteis para trabalhar com *Machine Learning*.
Para utilização do MLP, a biblioteca *Pandas* também teve de ser instalada para a leitura e conversão dos dados, uma vez que haviam formatos específicos para o uso em algoritmo, assim como o *Numpy* e *Pyplot* para funções numéricas do *SciKit*.
As principais funções utilizadas em código foram as contidas no módulo *MLPClassifier* dentro do diretório de redes neurais do *SciKit*, sendo elas: *fit, score, predict_proba e predict*.
Para conseguir utilizar as funções referidas, se fez necessário antes criar a rede neural *MLPClassifier*, que podia ser gerada de diferentes formas de acordo com os parâmetros enviados ao construtor. Após análise do tipo de dado a ser tratado, concluiu-se o envio dos seguintes parâmetros ao construtor: *hidden_layer_sizes=(10,15)*, representando o número de neurônios da camada oculta com o número de camadas menos 2 e tamanho, *solver='sgd'*, para utilização método de descida gradiente estocástica(método numérico usado em otimização para encontrar um mínimo local de uma função por meio de um esquema iterativo), *max_iter=600*, que é o número máximo de iterações caso o *solver* não

tenha chego em uma convergência, e *random_state=1*, para utilizar métodos randômicos para cada iteração.
Fit foi utilizada de modo a definir as classes alvos na matriz de dados, recebendo a matriz de dados e a coluna de notas da base de dados. Exemplo: *mlp.fit(df_x,df_y)*, onde df_x é a matriz e df_y os dados alvo.
Predict foi uma das principais funções utilizadas, na qual se era realizado a previsão das instâncias inseridas, sendo enviado um vetor de instância com todos os parâmetros descritivos.
Score foi instanciada como *mlp.score(df_x,df_y)*, utilizando a rede *mlp* pré-definida e enviando a matriz e dados alvos, a fim de retornar a precisão média nos dados e rótulos de teste fornecidos.
Predict_proba gerou as estimativas de probabilidade relativas a base, tendo como dado de entrada a base completa.

4.4. Interface gráfica

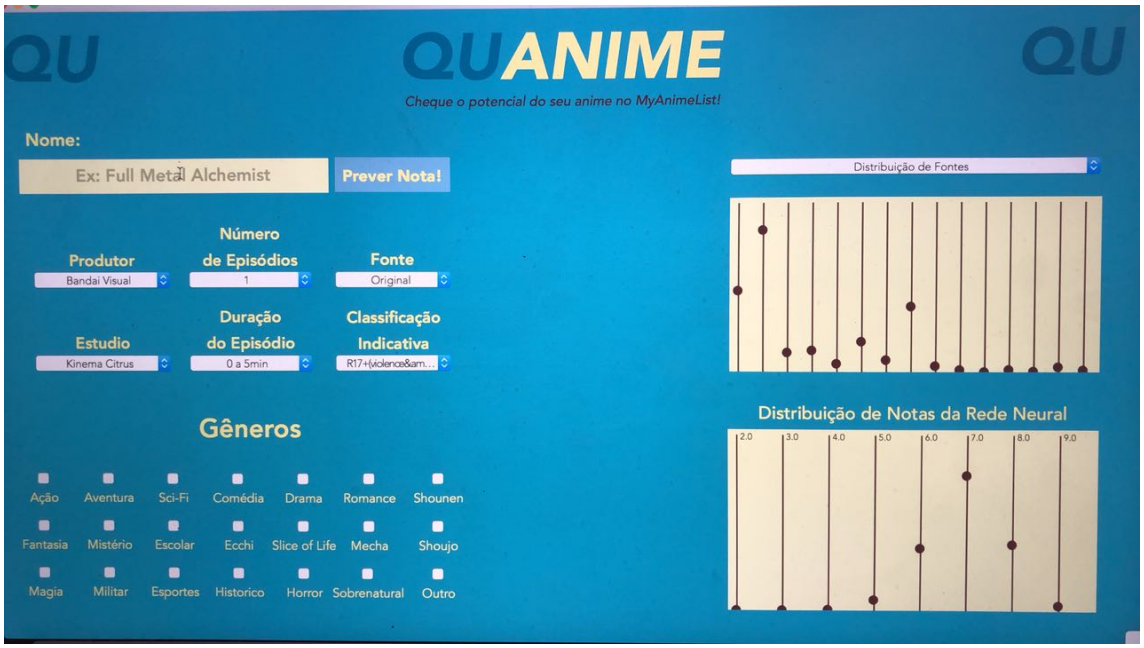

Figura 1. Interface

A interface gráfica mostrada na figura acima tem como principal a entrada de dados para o cálculo da nota que está localizada à esquerda da imagem.
O primeiro elemento da interface é uma caixa de texto onde se referencia o nome de determinado anime. Logo abaixo, estão caixas de listas suspensas onde se escolhe os parâmetros para buscar compatibilidade com os dados da base e assim estipular uma nota ao anime. Vale lembrar que as opções das caixas suspensas são os intervalos de distribuição comentados na seção *Base de Dados*. Ao final temos caixas selecionáveis para parametrizar os gêneros, o número de gêneros não pode ser superior a 3 (três) dado melhor previsibilidade da base.
No lado direito da interface temos as informações sobre a base de dados em forma de

gráficos, no qual o primeiro local onde estão dispostos os gráficos pode se selecionar diversas distribuições, como por exemplo a distribuição dos produtores, fontes e etc. No segundo local, o gráfico é exclusivamente para a distribuição das notas, dada a importância de informar ao usuário onde se concentra o maior número de notas e entender o funcionamento do que se está sendo buscado.
A nota prevista está localizada no centro da interface, juntamente com o nome do anime, onde a mesma é automaticamente atualizada caso haja uma outra previsão.

5. Resultados

O objetivo de prever a nota do anime em função da base de dados no *My Anime List* foi alcançado, porém, o resultado obtido não foi totalmente satisfatório, constando no valor de 57,87%.
A precisão final do algoritmo não foi alcançada de maneira esperada pelo grupo, tendo em vista que mesmo com alterações visando melhorias por meio da junção de intervalos na base de dados, o problema persistiu, mostrando apenas pequenos aumentos na precisão.
Outra obtenção deste projeto foi a própria base de dados, que foi criada e aprimorada conforme realizada a implementação do algoritmo. De acordo com as manipulações realizadas, notamos a real importância de se ter dados de qualidade, no qual somente classificações realmente úteis são utilizadas, evitando ao máximo o uso de dados desnecessários. Relativo a base, podemos ainda citar que não só estudos quanto a nota podem ser realizados com as instâncias colhidas, como também estudos relativos a qualquer outra classe do conjunto, que podem ser especuladas se trocadas de lugar com a nota.
Com base nos pontos ressaltados e na informação colhida, verificamos que por mais que a precisão tenha retornado um valor relativamente baixo, pela nota ser um intervalo e pelos valores flutuarem de acordo com o tempo, a precisão de 57,87% foi consideravelmente boa, desde que feita uma melhoria no algoritmo visando uma checagem que estude o caso dos resultados estarem dentro de um valor próximo de aceitação.
Cabe ainda ressaltar que para um análise realmente precisa da base de dados utilizando o *Multilayer Perceptron*, seriam necessários alguns meses ou anos, para que novas temporadas de animes fossem lançadas e testadas de acordo com variações da base de acordo com a flutuação das notas já existentes.

6. Conclusão

De acordo com todas as considerações realizadas até aqui, chegamos a conclusão de que as notas de animes podem sim ser especuladas por meio de algoritmos de inteligência artificial, porém, para que se tenha uma aproximação maior da possível nota, seria necessário a visualização dos dados com o decorrer do tempo utilizando novos animes em lançamento como entrada, além de possíveis melhoras relativas aos intervalos da classe final.
Considerando que a criação de intervalos facilitou a predição dos resultados, seria válido realizar novas tentativas de predição alterando o intervalo de classes como duração de episódio ou mesmo a nota final. Uma outra possibilidade interessante seria a visualização dos resultados da base utilizando outros algoritmos de inteligência artificial, como *K-Nearest-Neighbor*, que considera apenas uma certa quantidade de instâncias vizinhas próximas para a predição.
Muito além de tudo dito, consideramos que este tipo de trabalho seria demasiado interessante se utilizado em sites detentores da base de notas, como o próprio *My anime list*,

trazendo aos usuários experiências da qualidade do que está sendo apresentado, podendo até mesmo ser relacionado à espera de outros usuários em relação a nova animação.
Levando em consideração todas as observações, totalizamos que as notas tem sim relação com as características de cada anime, nos mostrando que todos os testes realizados tiveram determinada importância, trazendo-nos não só ideias de possíveis aplicações do software, como também um punhado de possíveis melhorias e novos testes em potencial para serem analisados com a base formada.

Referências

Apple Inc. Technical note tn2328: Changes to embedding python using xcode 5.0. Access date: 10 jun. 2018.

INRIA and other (2015). Scikit-learn: machine learning in python. Access date: 17 may. 2018.

My Anime List (2015). Myanimelist.net - anime and mangá database and community.