Uma estratégia de aprendizado supervisionado para recomendação de puzzles em um jogo casual

Paulo de Freitas Coleta Neto[1]* Túlio Braga Moreira Pinto[2]† Wagner Meira Jr.[1]‡ Luiz Chaimowicz[1]§

[1]Universidade Federal de Minas Gerais, DCC, Brasil
[2]Centro Federal de Educação Tecnológica de Minas Gerais, DECOM, Brasil

RESUMO

Há uma tendência crescente na oferta de editores de fases e ferramentas para criação de conteúdo por usuários de jogos eletrônicos que funcionam como mecanismos de extensão do jogo e permitem uma maior longevidade. Porém a adoção desses mecanismos pode resultar em uma experiência negativa aos usuários, causada pela dificuldade que os jogadores podem ter em achar conteúdo que lhe agradem, criado por outros jogadores.

Recomendar conteúdo apropriado para os jogadores, ou seja, facilitar ao usuário encontrar aquilo que goste, é o desafio tratado neste artigo. Em particular, propomos uma estratégia de aprendizado supervisionado para recomendar fases aos jogadores através da previsão das avaliações dos usuários. O interessante do sistema de recomendação é a personalização inerente, ou seja, cada jogador recebe uma recomendação individual.

Para validação da estratégia e mensuração da sua performance, coletamos as avaliações de fases por usuários do jogo Mr. Square, um puzzle casual para dispositivos móveis, durante o período de 5 meses. Nossos resultados indicam que podemos aumentar o percentual de fases avaliadas positivamente, e, consequentemente, aumentar a satisfação dos usuários para 96,3%, enquanto a estratégia da implementação atual do Mr. Square alcança apenas 70,1%.

Palavras-chave: Recommendation, Classification, User Content, Casual Game

1 INTRODUÇÃO

Um dos maiores, mais trabalhosos e importantes desafios do desenvolvimento de jogos é como manter o jogador ativo e interessado no jogo. Uma possível solução para este desafio é o uso de conteúdo criado por usuários Jasek [6]. Esta alternativa baseia-se na ideia de que a criação de conteúdo pelos próprios usuários tende a aumentar a longevidade do jogo. Este conteúdo pode variar desde a inclusão de novas fases, o que é comum em jogos com editores de *levels*, à criação de cosméticos, que são elementos sem influência na mecânica do jogo, chegando até as modificações capazes de criar modos de jogos ou até mesmo novos jogos. Alguns chegam a depender totalmente deste conteúdo, como visto em Graft [3].

Em Jasek [6] também são apontados alguns desafios e desvantagens de se utilizar o conteúdo criado por usuários. Dentre eles, destaca-se a possibilidade de baixa proporção de conteúdo de qualidade quando comparado à quantidade total de conteúdo criado. Esta questão potencializa o desafio de motivar os jogadores a consumirem os conteúdos criados por outros usuários.

É comum em jogos com conteúdo criado por usuários existir algum sistema de avaliação dos itens criados, seja por sistemas de notas, estrelas ou até mesmo *Likes* e *Dislikes*, representando respectivamente aprovação e reprovação. Tais sistemas podem servir para encontrar os itens mais aprovados pela comunidade do jogo, porém a aprovação geral do conteúdo não significa a aprovação de um indivíduo sobre o mesmo item, demandando uma estratégia diferente para o problema.

Segundo Pazzani e Billsus [8] uma estratégia para selecionar conteúdo relevante a cada usuário é utilizar sistemas de recomendação, que podem ser uma abordagem para solucionar o problema da baixa fração de bom conteúdo criado pelos usuários, apontando por Jasek [6]. Em sistemas de recomendação baseados no conteúdo é construída uma representação dos itens a serem recomendados e do usuário. Utilizando dados rotulados com as avaliações dos usuários sobre os itens, aprende-se um modelo de classificação para estimar o quanto um usuário pode gostar de determinado item, como em Pazzani e Billsus [8].

Neste artigo, nós propomos a implementação de um sistema de recomendação de fases para um jogo *puzzle* casual utilizando classificadores para prever a avaliação das fases feitas pelos usuários. Para avaliar o uso de sistemas de recomendação utilizamos uma base de dados contendo avaliações de conteúdo criados por usuários do jogo *Mr. Square*, um *puzzle mobile*, criado pela *Ludic Side Game Studio*[1].

Nossa contribuição é demonstrar a viabilidade do uso dos sistemas de recomendação para solucionar o problema da baixa fração de bons conteúdos criados pelos usuários. Esta estratégia ajudará os desenvolvedores de jogos a lidar com o desafio, citado por Jasek [6], de motivar os usuários a continuar consumindo este conteúdo criado por outros usuários.

2 TRABALHOS RELACIONADOS

Pequisas envolvendo sistemas de recomendações existem para diversas finalidades. Um exemplo é o trabalho de Linden et al. [7], que utiliza uma variante da técnica *Collaborative Filtering* para criar recomendações personalizadas de produtos para cada usuário em uma loja virtual. Um outro trabalho é Davidson et al. [2] no qual um sistema de recomendação é criado e avaliado para a plataforma de vídeos Youtube.

Também foram feitos trabalhos para a recomendação de conteúdo relacionados a jogos eletrônicos como o Sifa et al. [9] e Skocir et al. [10]. No primeiro trabalho, os autores recomendam novos jogos para um determinado usuário, utilizando os dados da plataforma *Steam*, voltada para jogos em computadores. Nesse são comparados modelos de classificação utilizando a vizinhança dos itens, os jogos, e *Factor Oriented Model*. Skocir et al. [10] cria um modelo próprio de recomendação chamado de *MARS, Multi-Agent Recommendation System*, que leva em conta o perfil da habilidade do usuário pelo histórico em jogos anteriores. Nosso trabalho se diferencia dos anteriores, pois iremos recomendar fases, os *puzzles*, dentro de um único jogo, enquanto a proposta deles é recomendar diferentes jogos dentro de uma plataforma.

*e-mail: paulo.freitas@dcc.ufmg.br
†e-mail: tuliobragam@gmail.com
‡e-mail: meira@dcc.ufmg.br
§e-mail: chaimo@dcc.ufmg.br

[1]http://ludicside.com/#portfolio

Uma abordagem diferente é o trabalho realizado por Hicks et al. [5]. Nele os autores lidam com o problema da baixa proporção de fases de boa qualidade em um jogo, o BOTS, no qual jogadores podem criar puzzles para compartilhar com outros usuários. Para solucionar o problema é adicionado ao jogo um processo de autoavaliação das fases criadas, no qual os jogadores precisam resolver o próprio puzzle criado antes de enviá-lo ao servidor. Os autores reportam que este processo gera uma redução nas fases de baixa qualidade criadas pelos usuários.

Comparando com os trabalhos envolvendo sistemas de recomendação citados anteriormente, o nosso trabalho apresenta uma abordagem diferente quanto ao modelo de recomendação construído, baseado no conteúdo, e o tipo de fonte de dados utilizado. Os trabalhos anteriores permitem montar um perfil do jogador, ou usuário, utilizando atributos próprios, disponíveis nas plataformas em que os trabalhos foram realizados, enquanto no nosso trabalho os valores relacionados ao jogador são oriundos das suas avaliações sobre os *puzzles*, o conteúdo a ser recomendado. Dessa forma, nosso trabalho contribui com um método de recomendação no contexto de jogos baseado apenas no conteúdo, sem o uso de dados adicionais sobre jogador, o que minimiza eventuais questionamentos sobre quebra de privacidade.

Enquanto Hicks et al. [5] apresenta um processo para reduzir a proporção de fases de má qualidade, nosso trabalho lida com o mesmo problema de uma maneira diferente, enviando aos usuários as fases que estimamos que serão aprovadas por eles. Esta diferença é nossa vantagem do nosso trabalho, pois não modificamos o conjunto de fases disponivel, e portanto matemos na base de dados fases para todos os gostos respeitando as preferências de cada usuário.

3 RECOMENDAÇÃO DE PUZZLES

A recomendação de *puzzles* dentro de um jogo trata da tarefa de exibir ao jogador os *puzzles* que estimamos que este avaliaria positivamente. O sucesso da tarefa pode ser mensurado pelo aumento da satisfação do usuário, que é definida pelo porcentual de fases recomendadas aos usuários que são avaliadas positivamente.

Realizar tal tarefa é difícil, pois precisamos entender o perfil de cada usuário. Diferentes usuários possuem diferentes critérios de avaliação das fases e, portanto, aprovarão e reprovarão diferentes conjuntos de fases. Um sistema de recomendação deve ser capaz de entender os diferentes grupos de usuários e as diferentes características avaliadas em cada fase.

Na abordagem de sistemas de recomendação baseada em conteúdo, devemos criar uma representação das fases, que é o conteúdo a ser recomendado, e uma representação dos jogadores, que serão os usuários a receber e avaliar as recomendações. Cada par usuário-conteúdo forma uma avaliação do conteúdo pelo usuário. A tarefa de recomendação então passa a ser a de aprender se cada par representa uma avaliação positiva ou negativa, o equivalente ao jogador aprovar ou reprovar a fase que acabou de jogar.

Comumente a representação do perfil do usuário apresentar dados que evidenciem suas preferências e histórico. Já a representação do conteúdo, as fases, é uma tarefa mais complexa, pois depende da maneira como os dados estão disponíveis. Trabalhos como Guyon et al. [4] debatem todo o processo da construção e seleção de métricas para representar itens para a realização de tarefas de aprendizado de máquina.

Após construirmos representações de cada avaliação, no nosso caso representado pelo par jogador e *puzzle*, podemos aplicar um modelo de aprendizado de classificação. Este modelo deverá aprender a classificar a percepção, positiva ou negativa, dos usuários sobre as fases, os *puzzles* criados por outros usuários. Esta tarefa será avaliada pela acurácia na previsão das avaliações, e especificamente pela precisão da classificação das classes positivas, pois tal métrica representa a satisfação média dos usuários.

4 METODOLOGIA

A nossa metodologia consiste em utilizar uma base de dados composta por um conjunto de avaliações de fases feitas por jogadores e realizar um treinamento utilizando o SVM, *Support Vector Machine*, de forma a obter um classificador capaz de prever as avaliações de cada usuário, e que poderá ser utilizado como um recomendador de fases.

Nesta seção discutiremos as tarefas realizadas ao longo do trabalho. Na primeira parte apresentamos o modelo *Support Vector Machine*, utilizado para treinar o classificador. Na segunda parte apresentamos a base de dados utilizada. A seguir apresentamos a construção das métricas, que são as representações dos jogadores e fases. Por último, debatemos sobre a amostragem da base de dados utilizada no treino e a validação dos resultados.

4.1 SVM

Neste trabalho utilizamos o *Support Vector Machine*, SVM, um método de classificação binário baseado na máxima margem linear discriminante, ou seja, tem o objetivo de achar o hiperplano que maximiza a margem ou espaço entre duas classes. Adicionalmente, podemos usar o método conhecido como *Kernel Trick* para achar uma margem discriminante não linear entre as classes que corresponda a um hiperplano em um espaço não linear de alta dimensionalidade, detalhado por Zaki e Wagner Meira [11, cap. 21]. O algoritmo SVM foi escolhido por ser considerado ótimo no problema da classificação binária, como visto em Boser et al. [1].

Por se tratar de um modelo de aprendizado supervisionado, o SVM ajusta a resposta do modelo baseado em um conjunto de dados de treinamento. Então, consideramos o dado de treinamento como um conjunto de n pontos em um espaço de d dimensões, cada ponto com seu rótulo $y_i \in \{+1,-1\}$, representando a qual classe o ponto pertence. Desta forma, definimos o *dataset* como sendo $D = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$.

Um hiperplano em d dimensões é dado pelo conjunto de pontos $\mathbf{x} \in \mathbb{R}^d$ que satisfazem a equação $h(\mathbf{x}) = 0$, na qual $h(\mathbf{x})$ é a função do hiperplano definida a seguir:

$$h(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b = w_1x_1 + w_2x_2 + ... + w_dx_d + b \qquad (1)$$

$\mathbf{w}$ é o um vetor de pesos com d dimensões e b é um valor escalar, chamado de viés. A função do hiperplano $h(\mathbf{x})$ serve como um classificador linear ou discriminante linear, que prevê a classe y para qualquer dado ponto $\mathbf{x}$, de acordo com a regra de decisão abaixo:

$$y = \begin{cases} +1 & \text{if } h(\mathbf{x}) > 0, \\ -1 & \text{if } h(\mathbf{x}) < 0 \end{cases} \qquad (2)$$

Dado o modelo de classificação acima, o SVM é um algoritmo que permite refinar o vetor $\mathbf{w}$, que representa o hiperplano $h((x))$, maximizando a margem de distância entre os pontos $\mathbf{x}$ pertencentes a diferentes classes y. Durante a etapa de treinamento, o algoritmo ajusta o vetor de pesos $\mathbf{w}$, de dimensão d, relativos às *features* de cada item do conjunto de dados de treinamento. Ao final, o que se obtém é um hiperplano $h(x)$ capaz de separar os dados. O algoritmo para a construção do SVM utilizando o *Kernel Trick* é detalhado em Boser et al. [1], neste trabalho foi utilizado um *Kernel* polinomial de terceiro grau.

Encontrar um hiperplano $h(\mathbf{x})$ que separe perfeitamente os pontos de diferentes classes pode ser impossível, caso os pontos não sejam separáveis, ou levar a um modelo muito complexo, também conhecido como *overfitting*. Para evitar estes casos indesejados, foi introduzido no modelo um parâmetro de regularização C empiricamente escolhido para obter o modelo com o menor erro de validação.

$$\text{Função objetivo: } \min_{\mathbf{w},b,\xi_i} \left\{ \frac{\|w\|^2}{2} + C\sum_{i=1}^{n} \xi_i^k \right\} \qquad (3)$$

$$\text{Restrições Lineares: } y_i(w^t x_i + b) \geq 1 - \xi_i, \forall x_i \in \mathbf{D} \qquad \xi_i \geq 0, \forall x_i \in \mathbf{D} \tag{4}$$

Na equação 3 o termo $\sum_{i=1}^{n} \xi_i^k$ representa os erros de classificação durante o treino, enquanto o termo $\frac{\|w\|^2}{2}$ representa o compromisso em maximizar a margem. O parâmetro C controla o *trade-off* entre maximizar a margem e minimizar os erros durante o treino. A constante k define o tipo de perda será utilizada, quando $k = 1$ utilizamos a *hinge loss* e quando $k = 2$ usamos a *quadratic loss*, conforme Zaki e Wagner Meira [11, cap. 21].

4.2 Base de Dados

Para avaliar nossos resultados utilizamos a base de dados do jogo *Mr. Square*, da *Ludic Side Game Studio*. O jogo é um *puzzle*, para dispositivos móveis, no qual usuários podem submeter novas fases, além das fases padrões criadas pelos desenvolvedores do jogo. Ao jogar as fases criadas por outros jogadores, o jogador recebe um conjunto aleatório de fases e poderá avaliar, positiva ou negativamente, cada fase após encontrar a solução para ela.

A base de dados utilizada é composta por um *log* contendo as avaliações, positivas e negativas, das fases por usuários do jogo; e por detalhes sobre cada fase como tamanho do *puzzle*, posição inicial do jogador, posição dos obstáculos, qual a solução, quantidade total de avaliações positivas e negativas.

O *log* de avaliações foi coletado no período de novembro de 2015 até março de 2016. O *log* é composto por 646.838 avaliações de fases, de um total de 457.823 fases criadas e 70.323 jogadores ativos durante o período de coleta. Destas avaliações, 453.445 são positivas, restando 193.393 avaliações negativas. Pela proporção entre avaliações positivas e negativas, temos que a atual aprovação média dos usuários ao conteúdo criado por outros jogadores é de 70,1%.

4.3 Métricas

Na abordagem dos Sistemas de Recomendação Baseados no Conteúdo, é necessário criar representações para o usuário e para o conteúdo. Nossa base de dados fornece, de forma estruturada, dados sobre todas as fases criadas pelos usuários até a data final da coleta. Junto com os dados de cada fase também temos o conjunto de avaliações, positivas ou negativas, enviadas por jogadores sobre as fases.

Para a representação de cada *puzzle*, nós elaboramos um conjunto de 11 métricas, sendo elas o tamanho do *puzzle*, dividido em largura e altura, o número de movimentos da solução e os 8 diferentes tipos dos elementos presentes na fase. Estes elementos são específicos do jogo e representam os obstáculos da fase.

Cada jogador é representado pelo histórico de avaliações. As métricas deste histórico são os valores médios das métricas das fases, separando entre as avaliadas positiva e negativamente. Portanto, 22 é o total de métricas para representar cada jogador. E para representar as possíveis avaliações basta agregar as métricas do jogador com as das fases, assim temos 33 métricas para cada avaliação.

Durante a criação das métricas para os *puzzles* foram detectadas 4.960 fases com configurações inválidas. Estas fases foram ignoradas no nosso modelo de dados, ficando assim um total de 452.863 fases. Devido às fases removidas da base de dados, também foram removidas 8.499 avaliações que envolviam as fases inválidas.

4.4 Amostragem

O processo de construção do modelo de classificação é uma operação custosa, em termos de processamento e tempo. Um dos fatores que influenciam diretamente o tempo gasto para treinar o modelo é o tamanho da base de dados utilizada. Para poder otimizar os parametros do modelo SVM, foi construída uma amostra da base completa utilizando 50 mil registros.

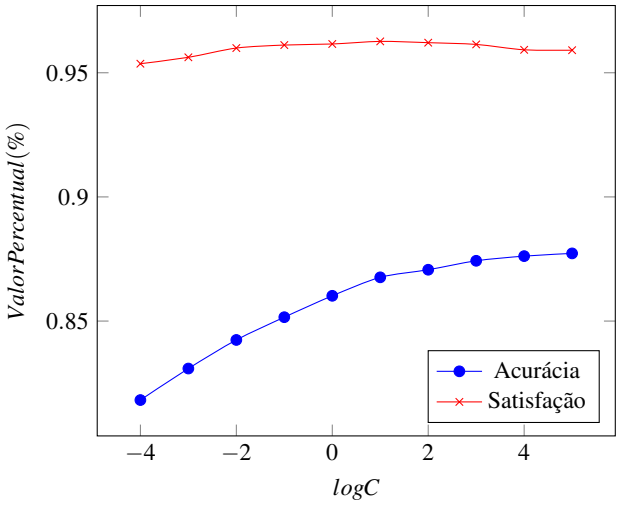

Figura 1: Acurácia e satisfação dos usuários por valor de C

Este processo pode comprometer o resultado do modelo geral, caso a base amostrada não seja uma representação satisfatória da base original. Uma maneira de medir o quão bem a base amostrada representa a original é comparando os valores de variância de cada uma das métricas nas diferentes bases de dados.

Nós construímos 10 amostras diferentes e, para cada uma, calculamos as variâncias de cada métrica. Para cada métrica calculamos a razão entre a variância da métrica da base original dividida pelo valor medido na base da amostra. A média das razões das diferentes métricas entre as diferentes bases é 1,0006, e o desvio padrão encontrado foi de 0,0039. Tais valores mostram que diferentes amostragens representam a base de dados adequadamente, tornando assim aceitável o uso das amostras no processo de aprendizado. Realizando o cálculo de intervalo de confiança, a razão entre a variância da base de dados e a variância da amostra tem 99% de certeza de estar dentro do intervalo entre 0,997 e 1,005.

5 Avaliação dos Resultados

O processo de avaliação do modelo de classificação foi dividido em dois passos: o primeiro é reservado à otimização dos parâmetros do modelo de aprendizado a ser construído; no segundo são avaliados os valores encontrados para o melhor modelo, seguindo um método de validação cruzada para estimar um intervalo de confiança.

Para o processo de otimização dos parâmetros foi variado o valor do parâmetro C do modelo SVM. Para cada classificador construído, são avaliados os valores encontrados para a acurácia do modelo dada pelo total de previsões corretas sobre o total de previsões realizadas; e a precisão da classe positiva, que é dado pelo total de itens previstos como positivos que de fato foram avaliados positivamente pelos usuários. A precisão da classe positiva é importante para o nosso trabalho, pois ela representa a satisfação do usuário caso o modelo de recomendação seja adotado.

A base de dados utilizada é uma amostra de 50 mil registros, separados em 80% para o treino e 20% para a validação. Os valores de C assumiram valores de potências de 2 que variaram de 2^{-4} até o 2^5. Os resultados são apresentados na Figura 1. Não apresentamos valores de C maiores, pois a construção do modelo não convergiu em tempo hábil.

Pela Figura 1 podemos perceber que, com o aumento do valor de C, também aumenta a acurácia do modelo, entretanto a precisão

Tabela 1: Validação K-Fold

Métrica	Média	Desvio Padrão	Intervalo de Confiança	
Acurácia	86,14%	0,436597959%	85,69% até	86,59%
Precisão +	96,28%	0,2765327346%	95,99% até	96,56%

da classe positiva, que representa a satisfação do usuário, começa a diminuir a partir do valor de C igual 2, portanto este será o valor usado nos próximos passos para o parâmetro C.

Para avaliar o modelo aprendido e estimar um intervalo de confiança, usamos o parâmetro escolhido no passo anterior. Realizamos uma validação cruzada utilizando a técnica K-Dobras, ou *K-Fold*, utilizando 10 como o valor de K, que representa o número de dobras.

A validação em K-Dobras, *K-Fold Validation*, consiste em dividir a base de dados em K partições de tamanhos iguais e treinar K diferentes modelos. Cada um dos K diferentes modelos será treinado com K-1 partições e validado utilizando a partição restante.

Com os resultados de cada um dos K modelos construídos, podemos estimar o valor médio da acurácia do modelo, utilizando a média entre os valores encontrados, e também calculamos o valor do desvio padrão. Com o valor médio, desvio padrão e o número de amostras utilizadas podemos realizar o cálculo do intervalo de confiança do nosso modelo, como detalhado em Zaki e Wagner Meira [11, cap. 12].

No nosso trabalho utilizamos 10 partições e calculamos o intervalo de confiança de 99%. Os valores encontrados podem ser vistos na tabela 1. Na primeira linha encontramos os valores para a acurácia do modelo, que representa a taxa de acerto para a previsão de todas as classes envolvidas. Na segunda linha encontramos os valores para a previsão da classe positiva.

Nossos resultados mostram que o modelo alcança boa acurácia, 86,14%, e boa precisão da classe positiva, 96,28%. Estas métricas também apresentam bons intervalos de confiança, mostrando a robustez do modelo. Apesar disso, acreditamos que melhores valores de acurácia poderiam ser alcançados com um processo de treinamento mais longo.

6 Aplicabilidade

O estudo realizado teve como foco um ambiente *offline* do jogo *Mr. Square*, com quantidade de fases estática, através da utilização de um histórico de avaliações. Apresentamos aqui uma discussão sobre a sua aplicabilidade em um ambiente dinâmico.

Para garantir a acurácia do modelo, é interessante atualizar a representação do usuário a cada avaliação feita, enquanto que a representação de cada fase é criada durante a submissão da mesma. Além disso, uma forma interessante de avaliar a necessidade de treinar novamente o modelo é utilizar como base a variância do conjunto de dados. Sendo assim, a cada novo treinamento, deve-se ser armazenado o valor da variância do conjunto de dados gerador do modelo. Posteriormente, através de um tarefa recorrente diária (*job*), é possível calcular a variância com base no conjunto de dados atualizado. Caso a variância relativa ao último modelo treinado e a variância relativa aos dados atuais possuam uma diferença maior que um limite preestabelecido, significa que um novo treinamento de modelo deve ser iniciado. Uma queda na acurácia também pode ser usada como indicador de que devemos treinar o modelo novamente.

Podemos afirmar que esta metodologia evita que os pesos do modelo sejam recalculados sem necessidade, poupando recursos operacionais do jogo, visto que o cálculo da variância é uma tarefa simples e barata. Além disso, ela garante que o treinamento seja refeito quando novos dados de avaliação das fases, novas fases criadas por usuários ou mesmo novas métricas lançados pelos desenvolvedores comecem a impactar nos resultados da predição de avaliações, mantendo o modelo consistente com os dados do momento.

7 Conclusão e Trabalhos Futuros

Neste trabalho, utilizamos um classificador SVM para realizar a tarefa de recomendação de conteúdo criado por usuários. Nossos resultados mostram que, ao aplicar esta estratégia, é possível aumentar a satisfação dos jogadores, dado pelo porcentual de avaliações positivas. Com a base de dados utilizada, tivemos um aumento da satisfação dos usuários de 70,1% para 96,3%. Por último, debatemos a aplicabilidade da estratégia e demonstramos que a mesma é viável e aplicável em ambientes reais.

Nosso trabalho não ataca diretamente o problema da baixa fração de bom conteúdo, porém abordamos o problema encontrando o conteúdo que agrada cada usuário individualmente. Não restringir a diversidade das fases é uma grande vantagem desta abordagem em comparação aos métodos de filtragem de conteúdo.

Para validar a estratégia apresentada aqui, um trabalho futuro é replicar o modelo com um conjunto de dados de um jogo diferente. Neste trabalho também não abordamos como os jogadores evoluem ao longo do tempo de jogo. Um trabalho futuro poderia abordar a possibilidade do usuário mudar suas preferências, seja ignorando as avaliações antigas ou priorizando as avaliações mais recentes.

Apesar dos bons valores encontrados para satisfação do usuário, nossos resultados não obtiveram o melhor modelo em relação à acurácia total, devido à preocupação em manter altos valores porcentuais de precisão da classe positiva. Trabalhos futuros deveriam atuar com foco em diminuir a quantidade de falsos negativos, avaliações que são positivas e que foram previstas como negativas.

Referências

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[2] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[3] K. Graft. User-generated content: When game players become developers, October 2012. Disponível em: http://www.gamasutra.com/view/news/179493/Usergenerated_content_When_game_players_become_developers.php. Acessado em: 27 de maio de 2016.

[4] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[5] A. Hicks, V. Cateté, and T. Barnes. Part of the game: Changing level creation to identify and filter low quality user-generated levels. In *Proceedings of the International Conference on the Foundations of Digital Games*, 2014.

[6] P. Jasek. User generated content for video games. Technical report, Department of Computer Science Aalborg University, January 2014. 9th semester project report of the Software Development program.

[7] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003.

[8] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[9] R. Sifa, C. Bauckhage, and A. Drachen. *Archetypal Game Recommender Systems*, pages 45–56. CEUR-WS, 2014.

[10] P. Skocir, L. Marusic, M. Marusic, and A. Petric. The mars - a multi-agent recommendation system for games on mobile phones. In *Proceedings of the 6th KES International Conference on Agent and Multi-Agent Systems: Technologies and Applications*, KES-AMSTA'12, pages 104–113, Berlin, Heidelberg, 2012. Springer-Verlag.

[11] M. J. Zaki and J. Wagner Meira. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, May 2014.