Investigando o uso de informações temporais para o desempenho de filtros colaborativos e cadeias de Markov na recomendação de aplicativos móveis

Gabriel T. P. Coimbra[1], **Raissa P. P. M. Souza**[1],
Fabrício A. Silva[1], **Thais R. M. B Silva**[1]

[1]Instituto de Ciências Exatas e Tecnológicas – Universidade Federal de Viçosa (UFV)
Florestal – MG – Brasil

{gabriel.coimbra,raissa.polyana,fabricio.silva,thais.braga}@ufv.br

Resumo. *Com o uso crescente de dispositivos móveis, mais aplicações estão sendo desenvolvidas e publicadas para satisfazer as necessidades do consumidor. Com muitas opções, os usuários podem ficar com dificuldades com as possibilidades de selecionar um aplicativo móvel para atender às suas necessidades. Uma solução comum para este problema é desenvolver sistemas de recomendação com o objetivo de, com base nas necessidades do usuário, recomendar aplicativos que provavelmente os usuários instalariam por conta própria. Neste trabalho, usamos informações cronológicas das sequências de instalação de 14 mil usuários para melhorar algoritmos tradicionalmente utilizados para sistemas de recomendação. Nossos resultados mostram um aumento em média de até 16% na revocação para os filtros colaborativos e 36% para a Cadeia de Markov, em comparação com os mesmos modelos não utilizando informações cronológicas.*

1. Introdução

Com a popularização de dispositivos móveis, é perceptível um aumento no número de possibilidades para instalação de aplicativos. Somente na *Google Play Store* ™ existem 2,56 milhões de aplicativos que podem ser instalados em aparelhos com sistema operacional *Android*. Com esse volume, a escolha de qual aplicativo instalar pode se tornar mais onerosa para os usuários. Alguns desses aplicativos são muito populares e utilizados pela maioria dos usuários de *smartphones* (e.g., *Facebook*, *Instagram*). Porém, outros pertencem a contextos específicos (e.g., diagnóstico de falhas em automóveis, contabilidade) e não são facilmente recomendados pelas lojas. Essas situações, em conjunto com a quantidade de aplicativos, levam os sistemas recomendativos para aplicativos a serem de suma importância, pois permitem que os usuários encontrem o aplicativo mais apropriado.

Um método para resolver esse problema é a recomendação personalizada baseando-se em informações relevantes para cada usuário. O problema de recomendação de aplicativos é caracterizado por, dadas informações sobre o usuário e seu contexto, prever quais aplicativos tem maior probabilidade de serem de interesse do usuário. Porém, mesmo com uma quantidade razoável de dados, pesquisas nessa área relatam a dificuldade de obter uma alta precisão na recomendação [Cheng et al. 2018, Souza et al. 2020, Peng et al. 2018, Yin et al. 2017, Xu et al. 2018]. Alguns motivos são a grande quantidade de aplicativos, tornando o conjunto de dados esparso, ou aplicativos com propósitos de nicho sendo instalados em situações específicas, que é uma situação difícil de prever.

Neste trabalho, consideramos a hipótese de que as informações temporais de instalações de aplicativos são importantes para a recomendação pela tendência dos usuários a instalarem aplicativos com propósitos semelhantes ou complementares em sequência. Esse comportamento pode ser explicado pela necessidade de testar diversos aplicativos que prometem os mesmos benefícios (e.g., jogos com o mesmo tema), por aplicativos que são complementares (e.g., comércio eletrônico e *cashback*) ou pela lembrança de algum aplicativo após a instalação de outro. Por isso, o presente trabalho tem como objetivo investigar essa hipótese utilizando algoritmos que podem conseguir capturar essa tendência de instalação cronológica de aplicativos.

Para alcançar esse objetivo, são utilizados duas estratégias para a recomendação: Cadeias de Markov (CM) e Filtros Colaborativos (FC). São comparados os resultados dessas soluções base com a proposta do trabalho que adiciona a informação cronológica. Foram usados dados reais de 14 mil usuários coletados por um período de 3 meses. Os resultados mostram que foi possível melhorar a precisão e a revocação para a recomendação de aplicativos, indicando que a informação temporal é um fator relevante.

Esse artigo está organizado da seguinte forma. Na seção 2, apresentamos os principais trabalhos relacionados. Na seção 3 exploramos os dados utilizados em nosso trabalho. Na seção 4 apresentamos a construção dos modelos de aprendizado e a sua utilização para recomendação. A seção, 5 apresenta as estatísticas de desempenho dos modelos e discute os resultados obtidos. Por fim, na seção 6, apresentamos as conclusões e possíveis extensões deste trabalho.

2. Trabalhos Relacionados

Alguns algoritmos tradicionais de sistemas recomendativos são utilizados para a recomendação de aplicativos. O artigo de [Cheng et al. 2018] utiliza Cadeias de Markov e *Latent Dirichlet Allocation* (LDA), propondo um modelo estatístico unificado para capturar três contextos motivadores para instalação de aplicativos: curto prazo, padrões de co-instalação e escolha aleatória. Os contextos de curto prazo são capturados utilizando as Cadeias de Markov. Para detectar padrões de co-instalação ao longo prazo, um algoritmo baseado em LDA foi utilizado. Já o terceiro modelo descrito baseia-se no fato de que nem todas as tendências seriam capturadas pelos dois algoritmos anteriores, por isso uma distribuição multinomial é utilizada para explicar essas situações. Por fim, os três modelos são integrados utilizando o Modelo oculto de Markov (*Hidden Markov Models*), que permite inferir qual modelo tem mais potencial de acerto para determinadas situações.

A utilização do LDA também é feita por [Frey et al. 2017] para extrair características dos aplicativos com base em suas descrições e nomes, ao invés de detectar padrões de co-instalação ao longo prazo utilizando as sequências de instalações. Além do LDA, são utilizadas características dos usuários como sexo e idade obtidas com o uso de questionários. Essas variáveis foram utilizadas em conjunto com os tópicos do LDA para capturar as tendências de interesses dos aplicativos. Como as relações entre as variáveis e os tópicos extraídos podem não ser lineares, o algoritmo de Floresta Aleatória foi utilizado para permitir capturar essas relações complexas. Assim como em [Cheng et al. 2018], acredita-se que o LDA pode assimilar padrões de instalação ao longo prazo por agregar aplicativos com tópicos comuns. Apesar disso, informações temporais

não são utilizadas explicitamente.

Várias informações podem ser utilizadas para tentar prever a recomendação além da similaridade entre aplicativos. O trabalho de [Peng et al. 2018] propõe um algoritmo de filtros colaborativos baseado em fatoração de matrizes que considera características dos aplicativos e sua relação com as permissões no sistema operacional. Uma vantagem dessa abordagem é considerar preferências de privacidade de cada usuário, não sendo necessário obter outras informações sigilosas. Nesse artigo, uma modelagem de risco sugere que aplicativos com funcionalidades similares tem *scores* de risco de segurança distintos, e existe uma preferência dos usuários por aplicativos com o menor risco. Portanto, é calculado um risco para cada aplicativo que é utilizado para a construção da fatoração de matrizes e realizar a recomendação. Além de [Peng et al. 2018], [Yin et al. 2017] também propõe a utilização de preferências das permissões e características dos aplicativos. As características são representadas com a utilização de vetores latentes que modelam os interesses do usuário com base nas palavras presentes em descrições dos aplicativos. Com isso, é possível recomendar aplicativos que sejam do interesse do usuário e respeitem sua privacidade.

Além da utilização de filtros colaborativos, [Xu et al. 2018] propõe um método que pode gerar recomendações considerando funcionalidades dos aplicativos ao invés de considerar somente a similaridade, como métodos baseados em filtros colaborativos fazem. As funcionalidades dos aplicativos são extraídas utilizando suas descrições nas lojas onde são disponibilizados. É proposto um método de ranqueamento baseado em grafos que, de forma similar ao *PageRank*, é capaz de considerar as necessidades existentes e futuras de cada usuário.

O trabalho feito por [Souza et al. 2020] utiliza filtros colaborativos para entender a similaridade de aplicativos e a preferência de usuários por aplicativos específicos. Nesse trabalho, a abordagem item-item com o conceito para os filtros colaborativos foi utilizada devido ao custo assintótico viabilizar escalar o modelo para um grande volume de usuários. Ao contrário de [Cheng et al. 2018] e do presente trabalho, [Souza et al. 2020] não usa o histórico de instalação ou a sequência cronológica, mas apenas os aplicativos instalados atualmente em cada *smartphone*. O trabalho procede criando dois modelos, denominados *ANCESTOR* e uma solução *baseline ALBERTA*, em que a primeira utiliza informações demográficas além de interesses semelhantes de aplicativos. As informação demográficas são incluídas em uma coluna da matriz usuário-aplicativo, indicando o pertencimento de um usuário a um grupo demográfico. Dessa forma, a vizinhança de cada usuário, além de considerar aplicativos instalados em conjunto, também considera informações demográficas.

Até onde sabemos, com exceção de [Cheng et al. 2018], a utilização de informações cronológicas para a recomendação de aplicativos não é explorada na literatura. O trabalho de [Cheng et al. 2018] procura reunir capacidade preditiva de três algoritmos diferentes, cada um com o potencial de capturar tendências de curto, longo prazo e tendências aleatórias. Porém o método utilizado para a construção da Cadeia de Markov não foca em permitir esse algoritmo diferenciar padrões de instalação com a utilização da distância temporal. Os outros trabalhos utilizam outras informações não exploradas aqui, como a avaliação de risco dos aplicativos, informações demográficas ou a elaboração de algoritmos mais sofisticados. Nosso trabalho investiga como as informações cronológicas

podem melhorar o desempenho de sistemas recomendativos para aumentar o peso de instalações em determinadas distâncias temporais.

3. Dados Utilizados

Os dados utilizados no presente trabalho foram fornecidos por uma empresa privada mediante assinatura de termo de confidencialidade. Os dados representam os aplicativos instalados nos dispositivos móveis de milhares de usuários, que são coletados diariamente, possibilitando a construção da sequência de instalações. É importante notar que, quando mais de um aplicativo era instalado no mesmo dia, o método de coleta não pode determinar a ordem de instalação desses aplicativos.

Para assegurar que a ordem de instalação em um mesmo dia não afete os resultados por algum viés, como a ordenação dessa sequência pelo identificador do aplicativo, as sequências de instalações em um mesmo dia foram reordenadas de forma aleatória. Essa reordenação é importante pois, em instalações em um mesmo dia, não temos uma sequência cronológica, mas é possível que outro critério tenha sido utilizado na coleta das instalações que pode enviesar o algoritmo.

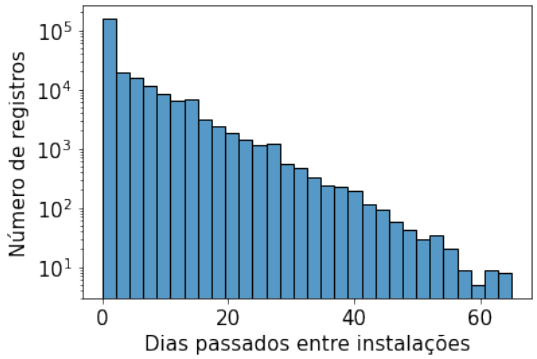
A figura 1(a) exibe a quantidade de dias passados entre uma instalação e outra consecutiva para todos usuários. Como pode ser visto, para muitos pares de instalações (44% para ser mais específico) não há ordenação cronológica pois os aplicativos foram instalados no mesmo dia. Porém, a figura 1(b) ilustra que na maioria desses dias uma pequena quantidade de aplicativos é instalada. Portanto, quando as sequências são reordenadas, aplicativos que estavam próximos na sequência de instalação continuariam dessa forma após a reordenação, não afetando de forma significativa os resultados. Além disso, na figura 1(a) percebemos que os usuários são ativos para instalar novos aplicativos, instalando em média um aplicativo a cada três dias. Isso corrobora a importância de sistemas recomendativos para aplicativos.

Para excluir aplicativos cuja análise não será relevante e melhorar a recomendação, decidimos não considerar aplicativos com mais de dez mil instalações e com menos de quatro. Esses valores representam o percentil 1 e 99 dos aplicativos mais instalados e menos instalados. Esse filtro é importante, pois existem aplicativos incomuns que podem estar em fase de desenvolvimento ou que já entraram em desuso e sua recomendação muitas vezes é desnecessária. Já a recomendação para os aplicativos com alto número de instalação também não é considerada útil para os objetivos desse trabalho.

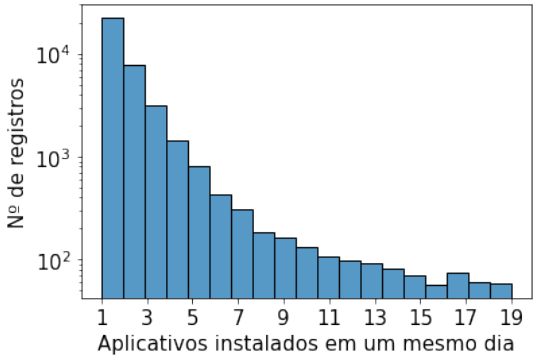
O método da coleta teve como consequência erros em que alguns dias, alguns usuários tinham um número muito baixo ou muito alto de instalações. Para corrigir isso, para cada dia de cada usuário foi utilizada a fórmula $\frac{i-\mu}{\sigma}$ para normalizar, sendo μ a média da quantidade de aplicativos instalados por dia para todos usuários, e σ o desvio padrão. Se o resultado dessa normalização para algum dia de um usuário for maior que três desvios da média, as instalações desse dia são desconsideradas. Além disso, filtramos usuários com somente um dia de instalação, pois nesse caso não teríamos informação cronológica, visto que em um mesmo dia não sabemos a ordem de instalação.

Outro pré-processamento dos dados realizado foi filtrar usuários que não tenham ao menos 10 aplicativos instalados, sendo necessário que tenham sido instalados em um mínimo de 4 dias distintos para considerarmos somente usuários que possam fornecer

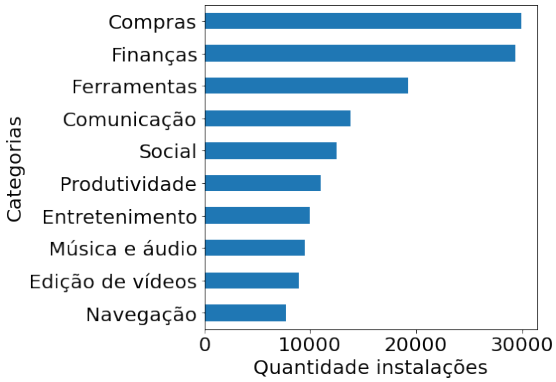
informações sequenciais de instalação. Estes filtros resultaram em um conjunto de dados com 14.660 usuários válidos, 13.329 aplicativos distintos e 244.002 instalações.



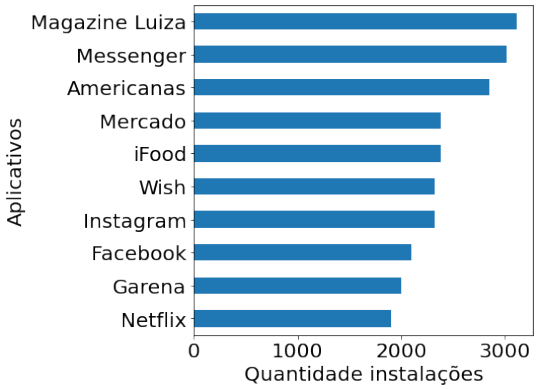
(a) Distribuição da quantidade de pausas entre instalações em dias para todos usuários.



(b) Distribuição da quantidade de aplicativos instalados em sequência em um mesmo dia para todos usuários.



(c) Quantidade de instalações das 10 Categorias mais populares.



(d) Quantidade de instalações Top 10 aplicativos mais populares.

Figura 1. Caracterização dos dados

As figuras 1(c) e 1(d) representam a distribuição de novas instalações entre as categorias e aplicativos no conjunto de dados. As instalações de categorias são mais concentradas em aplicativos financeiros, assim como é perceptível uma prevalência maior dessa categoria nos 10 aplicativos mais populares. Vale destacar que, se um aplicativo já está instalado no *smartphone* do usuário no início da coleta, ele não é contabilizado aqui.

4. Métodos

Nesta seção são descritos detalhes de implementação dos modelos para a recomendação de aplicativos.

4.1. Cadeia de Markov (CM) e Cadeia de Markov com Ajuste de Tempo (CMAT)

A Cadeia de Markov (CM) foi gerada de forma idêntica a [Cheng et al. 2018], assim como a sua utilização para a recomendação de aplicativos. A CM é um algoritmo que permite inferir as tendências de instalação de um aplicativo com base na frequência com que os mesmos são instalados em alguma sequência. Com a forma proposta em [Cheng et al. 2018],

a sequência de instalação de aplicativos permite gerar múltiplas Cadeias de Transição de Markov para uma mesma sequência de um usuário.

Seja a sequência de instalação $(app_1, app_2, ..., app_{n-1})$. Modelar a probabilidade condicional de instalação do próximo aplicativo app_n dado os $n-1$ aplicativos anteriores levaria a problemas de escalabilidade devido ao aumento exponencial no número de possibilidades e aumento na esparsidade do problema. Ao invés disso, são geradas k matrizes para modelar o fator de instalação no curto prazo. Na proposta de [Cheng et al. 2018], cada cadeia é representada por uma matriz de dimensões $N_{apps} \times N_{apps}$ sendo N_{apps} o número de aplicativos. A solução pode ser composta de k matrizes M_k, cada uma utilizando uma sequência de instalação derivada da original. A constante k indica qual aplicativo será o próximo para compor o par (origem,destino) de instalação. Quando $k = 1$, por exemplo, a sequência original é utilizada sem modificações. Dada a sequência de instalação (app_1, app_2, app_3), são incrementadas as entradas $M_{[app_1,app_2]}$ e $M_{[app_2,app_3]}$ na matriz M_1. Porém, quando $k = 2$, deve ser considerado somente o par (app_1, app_3) na matriz M_2, ignorando o aplicativo entre a origem e o destino.

Para a recomendação, são considerados os k últimos aplicativos da sequência I_u. Com isso, são somados k vetores de dimensão $1 \times N_{apps}$ extraídos das posições $M^k_{[|I_u|-k]}$. Para obter as probabilidades de instalação, normalizamos o vetor resultante V ao dividir cada posição por k. Com isso, a posição V_{app_j} indicará a probabilidade de instalar o app_j após a sequência $(app_{|I_u|-k}, ..., app_{|I_u|-2}, app_{|I_u|-1}, app_{|I_u|})$.

Em nosso trabalho, propomos uma versão modificada da Cadeia de Markov nomeada Cadeia de Markov com Ajuste de Tempo (CMAT) que pode ter um melhor desempenho ao prever os próximos s aplicativos a serem instalados sem ser necessário aumentar a constante k e assim evitar um possível sobre-ajustamento do modelo. Para isso, considera-se o conjunto de sequências de instalações I composto de $|I|$ sequências ordenadas cronologicamente I_u para cada usuário u. Cada sequência I_u possui os aplicativos que foram instalados pelo usuário u. Considerando esse conjunto de dados, definimos a função ω na equação 1. Essa função tem como objetivo calcular a importância de um par de instalação $(app_{origem}, app_{destino})$ com base na distância entre ambos aplicativos na sequência de instalação (i.e., quantos aplicativos foram instalados entre o aplicativo de origem e de destino). Após o cálculo, o valor resultante será adicionado na Cadeia de Markov na posição $M_{[app_{origem},app_{destino}]}$.

$$\omega(I_u, app_{origem}, app_{destino}) = \begin{cases} w_1 & \text{se } d \leq t_0 \\ w_2 & \text{se } t_0 < d \leq t_1 \\ 0 & \text{senão} \end{cases} \tag{1}$$

$$t_1 \geq t_0 \; \forall (t_0, t_1) \in (\mathbb{N}^+_* \times \mathbb{N}^+_*) \wedge (w_1, w_2) \in (\mathbb{R}^+_* \times \mathbb{R}^+_*),$$

$$\text{onde } d = dist(I_u, app_{origem}, app_{destino})$$

A função $dist$ calcula a quantidade de aplicativos entre app_{origem} e $app_{destino}$ na sequência I_u, podendo resultar em um valor negativo caso $app_{destino}$ preceda app_{origem} na sequência I_u, ou zero caso um aplicativo ou ambos não estejam na sequência. Seja o conjunto de aplicativos A_{t_0} que encontram-se a uma distância menor ou igual a t_0 de app_{origem} e os aplicativos A_{t_1} cuja distância de app_{origem} é maior que t_0 e menor ou igual à t_1. A

motivação da função ω é a premissa de que os aplicativos do conjunto A_{t_0} terão uma tendência maior de serem instalados do que os aplicativos do conjunto A_{t_1} que, por sua vez, terão uma probabilidade maior que os próximos aplicativos com uma distância maior que t_1 de app_{origem}. Dois pesos, w_1 e w_2, são aplicados nos intervalos de 0 até t_0 e de t_0 até t_1 ao invés de um peso definido de forma contínua. Isso permite que a mesma importância seja dada a aplicativos que estejam a distâncias muito próximas, e também previne o sobre-ajustamento passível de acontecer caso uma fórmula contínua fosse usada para calcular esse peso. Pois, nesse caso, cada par de instalação a uma distância diferente poderia ter um peso distinto, mesmo que a influência não seja significativamente diferente. Com isso, cada entrada da matriz de transição de markov $M_{[app_{origem},app_{destino}]}$ será calculada com a seguinte fórmula:

$$M_{[app_{origem},app_{destino}]} = \sum_{u}^{\|I\|} \sum_{m}^{\|APP\|} \sum_{n}^{\|APP\|} \omega(I_u, app_{origem}, app_{destino})$$

Devido à natureza sequencial do problema, os últimos s aplicativos da sequência I_u são separados para teste do modelo como feito em [Cheng et al. 2018]. Com isso, a previsão é feita ao calcular o *score* obtido da Cadeia de Markov para cada aplicativo. Para isso utilizamos os $p \in L$ últimos aplicativos da sequência I_u. A previsão se dará pelos Top-10 aplicativos com maior *score* definido pela seguinte fórmula:

$$Score(L, app_{destino}) = \sum_{p}^{\|L\|} \sum_{app_{destino}}^{\|APP\|} M_{[p,app_{destino}]}$$

4.2. Filtro Colaborativo

A implementação do Filtro Colaborativo (FC) baseou-se no *framework LightFM* [Kula 2015] versão 1.16. Essa é uma implementação em *Python* que utiliza um modelo híbrido entre dados colaborativos, em que os usuários e itens são representados por vetores latentes, e em conteúdo em que os fatores latentes são representados por combinações lineares de *embeddings* que descrevem o conteúdo de cada item ou usuário. Esse modelo baseado em conteúdo explica usuários e itens por seus fatores latentes que são aprendidos utilizando gradiente de descendência estocástico para minimizar a função de perda entre as recomendações fornecidas pelo modelo e os dados de treinamento.

Existem algumas vantagens dessa abordagem em relação a filtros colaborativos baseados em memória ou modelo somente. Em primeiro lugar, o desempenho em situações *cold start* (i.e., onde se tem poucos dados sobre novos usuários ou aplicativos) e também em cenários de baixa densidade (i.e., poucas relações entre usuários e aplicativos) são tão bons quanto modelos puramente baseados em conteúdo. Em segundo lugar, quando dados colaborativos são abundantes, no caso desse trabalho, *LightFM* tem o desempenho próximo a métodos que utilizam somente a fatorização de matrizes.

O *LightFM* utiliza representação baseada em interações entre itens e usuários para aprendizado dos fatores latentes. Esse *framework* utiliza dois conjuntos de dados, sendo

o primeiro conjunto formado por tuplas que representam interações entre usuários e itens. Essas tuplas tem a forma (u, app_i, w) sendo u um usuário e app_i um aplicativo e w um peso atribuído a essa interação. Na versão base, o peso w é sempre positivo e tem valor 1, para indicar a instalação do aplicativo app_i pelo usuário u. A representação de itens e usuários são combinações lineares de seus vetores latentes de características dos mesmos. A combinação linear é aprendida através da maximização da função objetivo composta por uma função de custo do produto interno entre interações entre usuários e itens [Kula 2015]. Obtivemos melhores resultados utilizando a função de custo WARP *(Weighted Approximate-Rank Pairwise)* [Weston et al. 2011]. A função WARP permite considerar um ranqueamento das N melhores escolhas de recomendação em seu custo e permite trabalhar com pesos somente positivos.

Além da Cadeia de Markov, propomos uma versão modificada dos filtros colaborativos utilizando o *framework ligthfm*. Denominamos essa versão Filtros Colaborativos com Ajuste de Tempo (FCAT). Nessa versão, utilizando informações temporais, o peso w é calculado pela fórmula $w(app_i) = 1/t_i$ onde t_i é o número de dias passados desde o início da sequência I_u para o usuário u até a instalação do i-ésimo aplicativo. Com essa fórmula, aplicativos que foram instalados mais recentemente terão um peso menor. Ou seja, atribuiremos um peso maior a aplicativos que tenham a tendência a serem instalados mais rapidamente, em média, pelos usuários. Para que isso funcione, partimos da premissa de que o início da coleta dos registros dos usuários tenha começado de forma aproximadamente simultânea. Uma forma de garantir isso seria considerar instalações de todos os usuários a partir de uma data fixa. Além disso, é importante que essa data determinada seja mais recente para permitir ao modelo capturar tendências atuais de popularização de aplicativos.

O método proposto de Filtros Colaborativos com Ajuste de Tempo (FCAT) é diferente do que encontramos na literatura usando tempo para considerar o *concept-drifting* [Ding and Li 2005]. Este conceito consiste no comportamento em que aplicativos instalados mais recentemente são mais importantes para explicar a mudança dos interesses dos usuários ao longo do tempo.

5. Resultados

Nesta seção serão apresentados os resultados da avaliação dos modelos de aprendizado utilizando as métricas de precisão e revocação.

Os hiper-parâmetros de ambos algoritmos foram otimizados utilizando uma plataforma de metaheurísticas [de Rosa et al. 2019]. A meta-heurística utilizada foi Algoritmos Genéticos com 25 agentes e 20 iterações na CM e 10 indivíduos e 10 iterações no filtro colaborativo. O objetivo de minimização foi $1 - F1Score(Precisao\#3, Recall\#3)$ em que o valor $s = 3$ foi utilizado por representar um balanceamento entre da predição entre curto (i.e., $s = 1$) e médio prazo (i.e., $s = 5$). Os hiper-parâmetros da meta-heurística utilizados foram os definidos por padrão na ferramenta, sendo eles 25% de probabilidade de mutação, 50% de *crossover* e 75% de seleção. Nas tabelas 1,2 e 3 informamos os valores encontrados pelo algoritmo genético para os hiper-parâmetros. Os hiper-parâmetros para a Cadeia de Markov sem ajuste de tempo (e.g., k) não são informados, pois são os mesmos utilizados por [Cheng et al. 2018].

De forma similar a [Cheng et al. 2018], utilizamos os últimos aplicativos instala-

Tabela 1. CMAT

Hiper-parâmetro	Valor
t_0	3
t_1	2
p	5
w_1	56
w_2	50

Tabela 2. FC

Hiper-parâmetro	Valor
Fatores latentes	157
α	0.095
ρ	0.5
ϵ	0.009
Epochs	28

Tabela 3. FCAT

Hiper-parâmetro	Valor
Fatores latentes	63
α	0.0320
ρ	0.5
ϵ	0.0008
Epochs	12

Tabela 4. Hiper-parâmetros encontrados pela meta-heurística de Algoritmo Genético.

dos na sequência do usuário para testar o desempenho dos modelos de recomendação. Por isso, foi necessário filtrar usuários cuja sequência de instalação é menor que o necessário para o treinamento e teste dos modelos. A quantidade de aplicativos usados para teste em cada sequência é denominado s. E, para cada valor de s, é necessário excluir do conjunto de treinamento sequências menores que o necessário para o treino. Na Cadeia de Markov, as sequências muito curtas são as menores que $s + 2$, pois é necessário no mínimo um par de aplicativos na sequência de instalação. Para os filtros colaborativos, sequências muito curtas são menores que $s + 1$, pois basta a associação de um aplicativo com um usuário. Após esse passo, modelos distintos são treinados também para cada valor de s em ambos algoritmos (e.g., Cadeia de Markov e Filtros Colaborativos).

Cada algoritmo de recomendação calcula a probabilidade de um usuário instalar um aplicativo. Para cada usuário, foram separados os s últimos aplicativos da sequência de instalação para medir o desempenho dos modelos. As seguintes fórmulas foram utilizadas para medir a precisão (2) e a revocação (3).

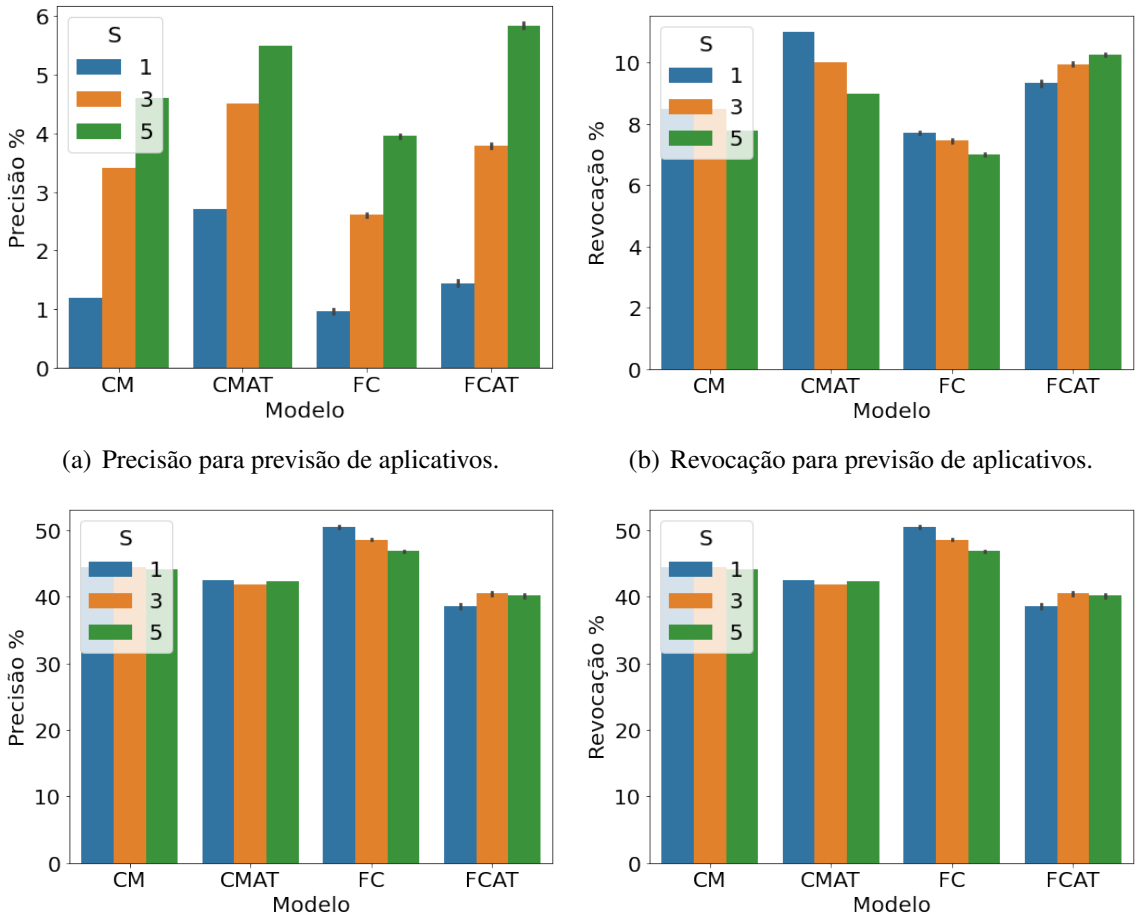
$$Precisao\#s = \frac{1}{|U|} \sum_{u}^{U} \frac{|U_s \cap U_n|}{n} \tag{2}$$

$$Revocacao\#s = \frac{1}{|U|} \sum_{u}^{U} \frac{|U_s \cap U_n|}{s} \tag{3}$$

Onde U representa o conjunto de usuários e $s = \{1, 3, 5\}$ e $n = 10$, U_s são os s últimos aplicativos usados para teste da sequência de instalação do usuário u e U_n são os n aplicativos com maior probabilidade calculada pelos modelos de serem instalados. O valor de n é o mesmo utilizado em [Cheng et al. 2018], onde utilizamos os 10 aplicativos com maior *score* para recomendação.

Devido à natureza não determinística na recomendação da ferramenta *LightFM*, para os resultados das figuras 2(d) e 2(c) as barras representam uma média de 10 execuções e o desvio padrão também é representado graficamente. Utilizando o teste-t constatamos que as diferenças entre as médias de precisão e revocação para base e algoritmos com informação cronológica tem significância estatística (valor-p) de no máximo 0.0001 em todas configurações do valor de s.

A Figura 2 apresenta os resultados. Em geral, pode-se perceber que a incorporação de dados temporais na Cadeia de Markov tem um maior potencial de melhora em relação



(a) Precisão para previsão de aplicativos.

(b) Revocação para previsão de aplicativos.

(c) Precisão para previsão de categorias.

(d) Revocação para previsão de categorias.

Figura 2. Resultados dos modelos para previsão de categorias e aplicativo utilizando métricas de precisão e revocação para os contexto $s = 1$, $s = 3$ e $s = 5$. Comparação de CM (Cadeia de Markov), CMAT (Cadeia de Markov com Ajuste de Tempo), Filtro Colaborativo (FC) e Filtro Colaborativo com Ajuste de Tempo (FCAT).

a base, se comparados com os filtros colaborativos utilizando os métodos investigados. Isso se deve à natureza das cadeias de Markov em tratar de sequência de eventos.

As figuras 2(a) e 2(b) ilustram a precisão e revocação para a recomendação de aplicativos em todas possibilidades do valor de s. No geral, vemos que a precisão tende a aumentar com o aumento de s, enquanto a revocação diminui; isso é justificado pelas equações 2 e 3, pois o numerador permanece o mesmo mas a interseção entre os conjuntos de recomendação (U_n) e teste (U_s) tende a aumentar. Apesar de ser utilizado uma menor proporção da sequência de instalação para treinamento em valores maiores de s (e.g., $s = 5$), é maior a probabilidade de que o aplicativo recomendado seja um dos cinco próximos que o imediatamente próximo em relação ao último aplicativo disponível para treinamento. Tanto a revocação quanto a precisão são baixas em relação ao máximo possível, mas isso também é observado em outros trabalhos [Cheng et al. 2018, Souza et al. 2020, Peng et al. 2018] e é devido ao grande número de aplicativos e a complexidade inerente ao problema.

Para a Cadeia de Markov, a melhora em relação à base foi maior quanto menor o valor de s. Acreditamos que isso se deve ao aumento das motivações de instalações no longo prazo e não capturados pelo modelo em comparação com tendências de curto/médio prazo. É verossímil que as motivações para instalação de aplicativos com alguma característica similar diminuam com o passar do tempo. Isso também é observado para os resultados do filtro colaborativo, mas talvez devido a um menor aproveitamento da informação cronológica no algoritmo, a melhora em relação à base não foi significativa para podermos ver a mesma tendência tão claramente. Contudo, podemos observar que ao invés da revocação ter uma tendência descendente em outras solução, a revocação tem uma tendência ascendente no FCAT. Uma explicação possível para esse fenômeno pode ser o fato da utilização de informações temporais em um algoritmo que, por sua natureza, não tem a característica de considerar o tempo entre as instalação, ao contrário da CM.

Para as figuras 2(c) e 2(d), mostramos os resultados ao tentar prever qual categoria de aplicativo será instalada utilizando os mesmos modelos treinados para as previsões de aplicativos. Sendo a $precisso\#1$ para $s = 1$, o resultado esperado da solução utilizando uma escolha aleatória de aplicativos será $\frac{n}{|APP|} \approx 6.8 \times 10^{-4}$ e $\frac{n}{|CAT|} \approx 0.26$ para categorias, sendo $|CAT|$ a quantidade de categorias (38) e n o número de recomendações (10). Com isso, o *lift* considerando a precisão do CMAT ($precisao\#1 = 0.027$), em relação à escolha aleatória, é 39 vezes maior para aplicativos. Enquanto que para categorias é menos expressiva, sendo somente 1.6 vezes maior, considerando a precisão do CMAT ($precisao\#1 = 0.42$) para categorias.

A solução de escolha aleatória consiste em considerar a chance uniforme de recomendar qualquer aplicativo ou categoria. O desbalanceamento de instalações de categorias e aplicativos é uma possível causa dessa discrepância nos resultados. No total existem 38 categorias, sendo que a categoria mais popular recebe aproximadamente 12% das instalações enquanto a quinta categoria mais comum tem perto de 5% instalações. Já para os aplicativos essa diferença está de 1.2% do aplicativo mais popular para 0.9% para o quinto mais popular. Esse desbalanceamento pode explicar os melhores resultados na *baseline* observados na previsão de categorias nas figuras. Além disso, é possível notar uma ligeira piora no desempenho dos modelos ao prever categorias devido à inclusão de informações temporais indicando que talvez a inclusão dessas informações na previsão de categorias não seja tão eficiente quanto para a previsão de aplicativos. Uma possível explicação para esse resultado seja que o treinamento dos modelos foram utilizando sequências de aplicativos e não de categorias.

6. Conclusão e Trabalhos Futuros

No presente trabalho, utilizamos informações cronológicas de instalação para investigar a melhora do desempenho de modelos para recomendação de aplicativos. De forma geral, é perceptível uma melhora da inclusão dessas informações nos algoritmos utilizando Cadeia de Markov e Filtros Colaborativos. Em trabalhos futuros pretendemos aplicar essa técnica em outros conjuntos de dados em contextos variados e analisar o treinamento utilizando categoria.Além disso, pretende-se comparar as soluções atuais com outros algoritmos mais conhecidos na literatura como SVD++, fatoração de matrizes e redes neurais.

Referências

Cheng, V. C., Chen, L., Cheung, W. K., and Fok, C.-k. (2018). A heterogeneous hidden markov model for mobile app recommendation. *Knowledge and Information Systems*, 57(1):207–228.

de Rosa, G. H., Rodrigues, D., and Papa, J. P. (2019). Opytimizer: A nature-inspired python optimizer. *arXiv preprint arXiv:1912.13002*.

Ding, Y. and Li, X. (2005). Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492.

Frey, R. M., Xu, R., Ammendola, C., Moling, O., Giglio, G., and Ilic, A. (2017). Mobile recommendations based on interest prediction from consumer's installed apps–insights from a large-scale field study. *Information Systems*, 71:152–163.

Kula, M. (2015). Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*.

Peng, M., Zeng, G., Sun, Z., Huang, J., Wang, H., and Tian, G. (2018). Personalized app recommendation based on app permissions. *World Wide Web*, 21(1):89–104.

Souza, R., Santos, L., Silva, M., Silva, F., and Silva, T. (2020). Impacto do uso de informações demográficas para a recomendação de aplicativos móveis. In *Anais do XII Simpósio Brasileiro de Computação Ubíqua e Pervasiva*, pages 111–120. SBC.

Weston, J., Bengio, S., and Usunier, N. (2011). Wsabie: Scaling up to large vocabulary image annotation.

Xu, X., Dutta, K., Datta, A., and Ge, C. (2018). Identifying functional aspects from user reviews for functionality-based mobile app recommendation. *Journal of the Association for Information Science and Technology*, 69(2):242–255.

Yin, H., Chen, L., Wang, W., Du, X., Nguyen, Q. V. H., and Zhou, X. (2017). Mobi-sage: A sparse additive generative model for mobile app recommendation. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 75–78. IEEE.