UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

IGOR CESCON DE MOURA

Identificando Usuários em Contas Compartilhadas através de Affinity Propagation

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Profa. Dra. Renata Galante
Coorientador: Prof. Dr. Weverton Luís da Costa Cordeiro

Porto Alegre
2019

RESUMO

Muitos dos sistemas de recomendação dependem de contas de usuários para identificar seus usuários e predizer itens que eles possam se interessar. O problema é que, não é incomum ter usuários que compartilham de uma mesma conta (e.g., familiares e amigos), seja para dividir o preço de um serviço ou por pura conveniência. Por essa razão, aprender a identificar um único usuário pelo histórico da conta pode levar a sugestões imprecisas. Para endereçar esse problema, alguns estudos já foram feitos, mas todos são focados em domínios específicos. Este artigo propõe um método genérico para identificar os diferentes usuários em contas compartilhadas, baseado em um método de *clusterização* por passagem de mensagens, o *Affinity Propagation*. Experimentos em múltiplas bases de dados, utilizando métricas de performance de *clusterização*, mostram resultados significativos que demonstram que é possível chegar próximo ao número de usuários em contas compartilhadas, e identificar com certa precisão qual é o usuário fazendo uso do sistema para se gerar uma recomendação mais relevante a ele.

Palavras-chave: Contas multiusuário. clusterização. affinity propagation. sistemas de recomendação.

Identifying Users in Shared Accounts using Affinity Propagation

ABSTRACT

Many recommender systems rely on user accounts to identify its users and predict items they may like. The problem is, it's not uncommon to have users who share an account (e.g. families and friends), whether to split the price of a service or for sheer convenience. For this reason, learning to identify a single user from the account history may lead to inaccurate suggestions. To address this issue, some studies were already made, but all focused in specific domains. This paper focuses on a generic method to identify the different users' profiles behind shared accounts, based on a clustering method by passing messages, the *Affinity Propagation*. Experiments on multiple databases, using clustering performance metrics, have shown meaningful results that proved that it's possible to find an approximation to the number of users in a shared account, and identify with some precision who's the user using the system to generate a recommendation relevant to him.

Keywords: Multi-user accounts, clustering, affinity propagation, recommender systems.

LISTA DE ABREVIATURAS E SIGLAS

RI	Rand Index
ARI	Adjusted Rand Index
AMI	Adjusted Mutual Information
FMI	Fowlkes-Mallows Index
SHE-UI	Session-based Heterogeneous graph Embedding for User Identification

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Garantir uma boa experiência de navegação para usuários que navegam online se faz fundamental, e empresas de mídia, entretenimento, e-commerce e varejo estão cada vez mais estudando formas de personalizar a entrega de conteúdo baseado nas preferências desses usuários. O viés deste estudo se faz por meio de Sistemas de Recomendação que buscam formas de auxiliar seus usuários na descoberta de conteúdo e produtos que possam lhes interessar, tornando assim a experiência de navegação mais completa. Muitos desses sistemas fazem uso principalmente de histórico de ações (como itens vistos, comprados ou avaliados) de contas de usuários, assumindo que cada conta pertence a somente um usuário, o que nem sempre é verdadeiro.

Por conveniência, muitas pessoas compartilham contas, seja para centralizar o pagamento dos serviços ou para aliviar o incômodo de trocar de contas em uma máquina compartilhada. Fazendo com que sistemas de recomendação, especialmente os baseados em filtragem colaborativa, tenham sua qualidade degradada (ZHANG et al., 2012). Esses sistemas acabam por recomendar conteúdo que não é de grande interesse de nenhum dos usuários, mas de médio interesse de todos (VERSTREPEN; GOETHALS, 2015).

Dado esse cenário, o desafio de identificar os vários usuários por trás de um conta se torna altamente relevante para a geração de recomendações precisas. Algumas soluções para esse problema já foram estudadas, porém as mesmas se encaixam em sistemas com metadados de domínios específicos (RASTOGI, 2015; SEMBIUM et al., 2018; JIANG et al., 2018) ou baseados em avaliações dos usuários (ZHANG et al., 2012).

Este trabalho propõe e implementa uma solução genérica para identificar os vários usuários em contas compartilhadas para sistemas com poucos metadados e cuja interação do usuário com os itens se dá de forma binária (e.g., compra de um produto ou consumo de um conteúdo). As questões de pesquisa são: (i) É possível utilizar *clusterização* para identificar contar compartilhadas em cenários diversos? e (ii) Qual é a efetividade do uso de *clusterização* para separar usuários? A solução é baseada no trabalho de Jiang et al. (2018) que faz uso de um algoritmo de *clusterização* por passagens de mensagens, o *Affinity Propagation* (FREY; DUECK, 2007). Validamos nossa clusterização considerando três métricas: *Adjusted Rand Index*, *Adjusted Mutual Information* e *Fowlkes-Mallows Index*, que serão apresentadas no decorrer do artigo. Os resultados demonstram que nosso algoritmo pode melhorar sistemas de recomendação, especialmente os por filtragem colaborativa onde se tem pouco conhecimento sobre os itens e a recomendação se dá pela

busca por usuários que possam ter gostos parecidos. Obtivemos MAE próximo de 1,6 e RMSE próximo de 2, no melhor caso, mostrando que o algoritmo consegue prever de forma aproximada a quantidade de usuários na conta. Também obtivemos resultados melhores que sem *clusterização* em um dos cenários, que mostra que há domínios em que a solução pode ser aplicada.

O restante do documento está organizado como segue. O Capítulo 2 apresenta os conceitos principais que embasam este trabalho e uma revisão do estado-da-arte. No Capítulo 3 definimos o problema da identificação de usuários e especificamos o método proposto para solucioná-lo. O Capítulo 4 descreve os experimentos e analisa os resultados. Por fim, o Capítulo 5 apresenta as conclusões e as possibilidades de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Neste capítulo é apresentado o levantamento bibliográfico e os principais conceitos relacionados ao desenvolvimento deste trabalho.

2.1 Introdução de Conceitos

Aqui são apresentados os principais conceitos que se faz necessário para o entendimento do trabalho como um todo. Serão apresentadas a definição de Sistemas de Recomendação, Sessão, Conta e Usuário.

2.1.1 Sistemas de Recomendação

Sistemas de Recomendação são sistemas capazes de predizer quais **itens** são relevantes para um usuário baseados nas preferências dele (RICCI et al., 2010). **Itens** é o termo geral usado para se referir ao que os sistemas de recomendação recomendam (e.g., produtos em um site de e-commerce, vídeos e músicas em sites de mídia). As preferências de um usuário podem ser **explícitas** (e.g., avaliação de um produto) ou **implícitas**, inferidas de alguma interação do usuário com o item (e.g., a compra de um produto, a visualização de uma notícia ou a escuta de uma música).

Sistemas de Recomendação podem ser **baseados em conteúdo**, por **filtragem colaborativa** ou um híbrido dos dois. Os **baseados em conteúdo** buscam itens similares aos que o usuário gostou no passado comparando as características entre eles. Já os sistemas por **filtragem colaborativa** buscam usuários similares baseado na similaridade entre as preferências deles, para então recomendar itens que um dos usuários gostou e o outro ainda não visualizou.

2.1.2 Sessão

Uma **sessão** é uma sequência de requisições de um usuário para a aplicação em um determinado período de tempo (ARLITT, 2000). No contexto desse trabalho, consideramos apenas as requisições que indiquem a preferência de um usuário à um item. Uma sessão tem um início e um fim determinados pela natureza da aplicação, podendo

ser considerado o tempo de inatividade do usuário, a diferença entre os itens em que ele interagiu, a forma com que chegou à aplicação, entre outros.

2.1.3 Conta de Usuário

Contas de usuários são a forma utilizada por aplicações para identificar seus usuários. Uma conta de usuário pode ser uma **conta individual**, ou seja, que pertence a apenas um usuário, ou uma **conta compartilhada**, onde vários usuários fazem uso (JIANG et al., 2018).

Os **usuários** são os indivíduos por trás das contas que esse trabalho busca identificar. Cada usuário tem seus comportamentos e preferências. Sistemas de Recomendação fazem uso das preferências associadas a uma conta, para gerar uma recomendação voltada à conta, que pode conter mais de um usuário. Nesse trabalho usamos as próprias preferências como forma de identificar o real usuário na conta para então gerar uma recomendação voltada ao usuário.

2.2 Método de Clusterização

Clusterização é a ação de agrupar objetos de forma que objetos similares fiquem no mesmo grupo (chamado de *cluster*). Sendo o objetivo do projeto separar sessões de uma conta em diferentes usuários, a forma escolhida para isso é através de métodos de *clusterização* onde cada *cluster* de sessões está associado a um usuário. O método *Affinity Propagation* (FREY; DUECK, 2007) encontra o número de *clusters* automaticamente, sendo estes representados pelo elemento que melhor generaliza todos os elementos dentro do *cluster*. Como é desconhecido pela aplicação o número de usuários por trás de uma conta, o método de *Affinity Propagation* se torna um bom candidato para a separação das sessões.

O *Affinity Propagation* se baseia na troca de mensagens entre itens até que sejam encontrados exemplares que representam cada *cluster*. O algoritmo recebe como entrada as similaridades entre cada item e a cada iteração são passadas dois tipos de mensagens: as *responsabilidades* e as *disponibilidades*, que são calculadas de acordo com as similaridades. Detalhes mais profundos do algoritmo são dados no capítulo 3.

Para calcular a similaridade entre os itens, considerando poucos metadados dispo-

níveis, usamos uma métrica de similaridade bastante comum, a similaridade por cosseno, que leva em conta os itens de cada sessão e quantas vezes eles aparecem. Essa métrica também é explicada em mais detalhes no capítulo 3.

2.3 Trabalhos Relacionados

O trabalho de Zhang et al. (2012) foi um dos primeiros a tentar identificar usuários distintos em contas compartilhadas, em sistemas de recomendação multiusuário. Eles constroem um modelo de contas compostas representadas pela união de subespaços lineares, para isso fazem uso das avaliações dadas pelos usuários aos itens.

Já no trabalho de Verstrepen e Goethals (2015) é proposta uma modificação do algoritmo de recomendação top-n baseado em itens (DESHPANDE; KARYPIS, 2004). Primeiro é demonstrado que, quando usado em contas compartilhadas, esse algoritmo favorece itens de média relevância para todos usuários da conta a itens de alta relevância para cada um dos usuários. A modificação consiste em buscar um subconjunto do conjunto de itens preferidos por uma conta, tal que o *score* entre esse subconjunto e um item candidato a ser recomendado seja máximo. Também é aplicado um fator à equação que calcula o *score* para favorecer conjuntos menores. Ao aplicar esse método a vários itens candidatos a serem recomendados, os com maior relevância para cada um dos usuários da conta terão maior *score*. Porém, esse método não reconhece o usuário corrente e é sugerido que todos os top-n itens sejam recomendados na esperança de que pelo menos um dos itens seja altamente relevante para aquele usuário.

Rastogi (2015) menciona em uma palestra no SIGKDD que lidar com várias pessoas por trás de contas de clientes individuais é um dos desafios pesquisados pela Amazon.com. Sembium et al. (2018) implementaram um sistema que recomenda produtos cujos tamanhos mais se aproximam dos usados pelos usuários (e.g., sapatos). Esse sistema aborda o caso em que um usuário pode corresponder a múltiplos indivíduos.

Mais recentemente, Jiang et al. (2018) propuseram um *framework* para identificar usuários distintos em um contas compartilhadas, porém especificamente para sistemas de conteúdos multimídia (música e vídeo). A abordagem feita se aproveita da grande quantidade de metadados provenientes desses sistemas para construir uma estrutura em grafo de onde se calcula a similaridade entre itens para depois reestruturar entre sessões (que são representadas por listas de itens) de uma conta. Essa similaridade é usada em um algoritmo baseado em *Affinity Propagation*, para então *clusterizar* sessões parecidas.

Considerando cada *cluster* um usuário na conta, dada uma nova sessão, encontra-se o usuário correspondente. A necessidade de um vasto número de metadados para criar o grafo inicial é uma limitação que impede esse método de ser usado em sistemas com poucos metadados. Ainda assim, utilizamos a parte da clusterização como base de nosso trabalho.

Com essa breve revisão da literatura relacionada, pode-se perceber que os trabalhos feitos até então para identificar os vários usuários por trás de uma conta, focam em domínios específicos e não são genéricos o suficiente para serem utilizados em grande parte dos sistemas de recomendação. É importante citar que esse projeto não estará limitado a um cenário específico, ou seja, poderá ser utilizado em uma vasta gama de sistemas com pouco ou nenhum metadado.

2.3.1 SHE-UI - Um framework para modelar preferências de Usuários

Nós colocamos em destaque o trabalho proposto por Jiang et al. (2018), que propõe o *framework* SHE-UI, pois é o que mais se aproxima da a nossa proposta, nós nos baseamos na ideia do uso de *Affinity Propagation* e nos dois passos principais do *framework* para propor nossa arquitetura. SHE-UI consiste em um *framework* para modelar as preferências de usuários em uma conta, e agrupar as sessões por esses usuários, o *Session-based Heterogeneous graph Embedding for User Identification* (SHE-UI).

O SHE-UI ocorre em duas etapas principais: primeiro o histórico de sessões de cada conta de usuário é *clusterizado* utilizando o *Affinity Propagation* para encontrar os *clusters* que representam cada um dos usuários, etapa chamada de UI-Past, e então para uma sessão nova é aplicada a similaridade entre os exemplares dos *clusters* e a sessão para encontrar a qual *cluster* ela pertence, etapa chamada de UI-New. Ambas as etapas requerem uma métrica de similaridade entre as sessões, para ser passada para o *Affinity Propagation* no UI-Past, e para comparar a nova sessão com os exemplares dos *clusters* no UI-New. A métrica proposta por Jiang et al. (2018) requer uma grande quantidade de metadados e de relações entre eles. Com exceção da métrica de similaridade, nosso modelo segue os mesmos passos principais do SHE-UI.

2.4 Considerações Finais

Trouxemos os conceitos principais para o entendimento do trabalho, assim como os trabalhos relacionados. Dos conceitos principais, sistemas de recomendação são os sistemas que mais podem tirar proveito de nossa solução, sessões são os objetos que *clusterizamos* para identificar os usuários que estão por trás de uma conta.

A maioria dos trabalhos que tentam solucionar o problema de identificação de usuários por trás de contas compartilhadas, funcionam apenas em cenários específicos. Em contrapartida, a nossa solução é genérica e feita para funcionar em sistemas escassos de metadados.

3 PROPOSTA DO MÉTODO PARA DETECÇÃO DE USUÁRIOS

Neste capítulo é especificado o método proposto para detecção de usuários distintos em uma conta. Definimos formalmente o problema da identificação de usuários na Seção 3.1. Uma arquitetura para o uso do método com um sistema de recomendação é descrita na Seção 3.2. Em seguida, são explicados os cálculos das similaridades, na Seção 3.3, e o passo do *Affinity Propagation*, na Seção 3.4, usados para os passos de *clusterização* de sessões em usuários distintos e para a identificação do usuário de uma nova sessão.

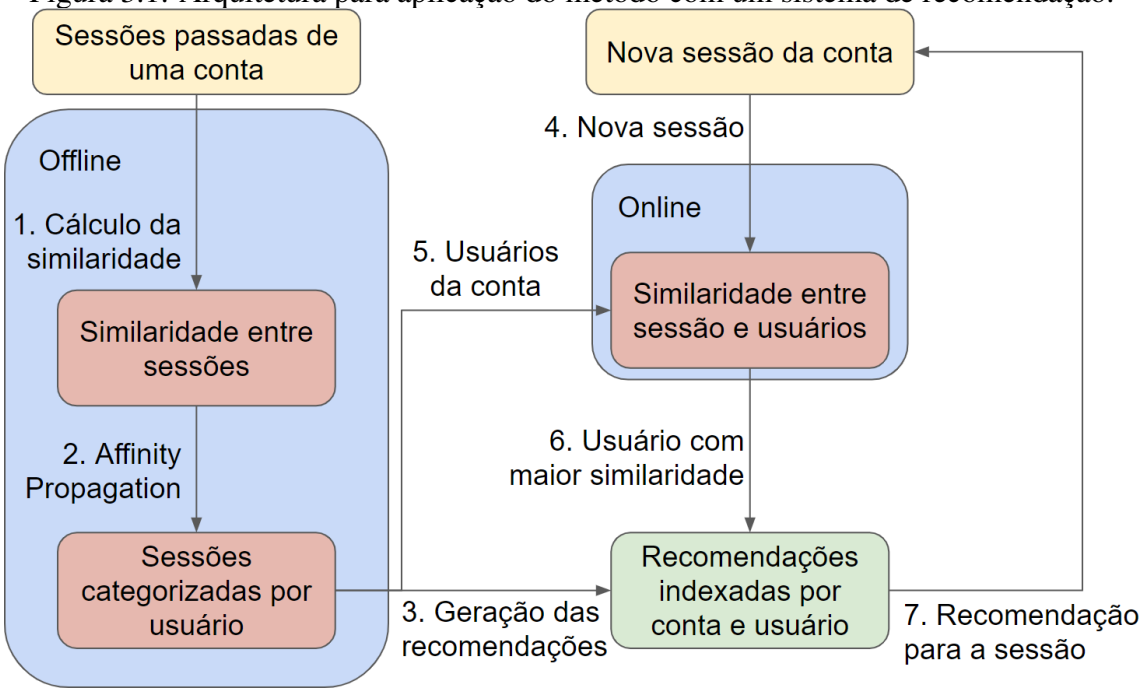
3.1 Definição do Problema

Nesta seção, definimos formalmente o problema de identificar usuários através de sessões, de forma similar a (JIANG et al., 2018). Seja I o conjunto de itens (e.g., produtos, notícias ou músicas). A o conjunto de contas de usuários. Para cada conta $a \in A$, o conjunto de usuários da conta é denotado como $U(a)$, que é previamente desconhecido. Cada conta $a \in A$ contém uma sequência de sessões $S(a)$. Cada sessão $s \in S(a)$ é uma sequência de T_S itens (que um usuário u_S interagiu): $s = \langle i_1, i_2, \ldots, i_{T_S} \rangle \in I^{T_S}$. Assumimos que cada sessão foi gerada por um único usuário. Os dois passos principais do método consistem em:

1. **Identificar Usuários em Sessões Passadas (etapa Offline)**: Dada a sequência de sessões de uma conta $S(a)$, esse passo consiste em *clusterizar* cada sessão $s \in S$ em k *clusters* (i.e., usuários), $C(a) = \left\{c_1^a, c_2^a, \cdots, c_{K_a}^a\right\}$ de forma que as sessões de um mesmo usuário acabem no mesmo *cluster*. A métrica de similaridade entre sessões é baseada nos itens $i \in I$ compartilhados entre elas, e será abordada mais adiante. K_a é desconhecido e estimado pelos dados, sendo que $1 \leq K_a \leq |S(a)|$. Esse passo é demonstrado no Algoritmo 1.
2. **Identificar Usuários de Novas Sessões (etapa Online)**: Dados os usuários $C(a)$ identificados para a conta a, para uma nova sessão $s \notin S(a)$ da conta a, o próximo passo consiste em predizer qual o usuário associado a essa sessão. Baseado nos n primeiros itens de s, queremos identificar o *cluster* c_k^a ao qual s pertence, ou seja, a qual usuário a sessão pertence. Esse passo é demonstrado no Algoritmo 2.

3.2 Arquitetura

Figura 3.1: Arquitetura para aplicação do método com um sistema de recomendação.



Fonte: O Autor

Apresentamos a arquitetura proposta para uso do método com um sistema de recomendação na Figura 3.1. Vemos os dois passos principais do método separados pelas caixas azuis. Para o processamento das sessões passadas, são aplicados os passos da etapa *offline*, descrito pelo Algoritmo 1. Na etapa *online*, demonstrado pelo Algoritmo 2, são passados os dados gerados pela *clusterização offline* e a nova sessão, para gerar uma recomendação voltada para o usuário da sessão. Os passos da arquitetura são os seguintes:

1. **Cálculo da similaridade**: Nesse passo as sessões da conta são comparadas par a par utilizando uma das métricas de similaridade explicadas na seção 3.3.
2. **Affinity Propagation**: A similaridade entre as sessões é passada para o algoritmo *Affinity Propagation* para que seja feita a *clusterização* das mesmas. Mais detalhes desse passo são explicados na seção 3.4.
3. **Geração das recomendações**: Para um algoritmo de recomendação é passado os usuários preditos pela *clusterização* como mais um dado a ser considerado junto do ID da conta. Sendo assim a recomendação é gerada não apenas para a conta, mas também para o usuário.
4. **Nova sessão**: Quando um usuário da conta está em uma nova sessão, esta é passada

para a etapa *online* para ser gerada uma recomendação.

5. **Usuários da conta**: Os usuários gerados pela etapa *offline* são recuperados para serem comparados com a nova sessão. Os usuários são sessões do passado que melhor representam o centro de cada *cluster*, então eles podem ser diretamente comparados com a nova sessão utilizando a mesma métrica de similaridade usada para gerar a *clusterização*.
6. **Usuários com maior similaridade**: Entre os usuários da conta, aquele que der o maior valor de similaridade com a nova sessão será o escolhido para representar o usuário e passado para o sistema de recomendação. Como mostrado no Algoritmo 2, se a nova sessão ainda não tiver uma quantidade significativa de itens, é passado para o sistema de recomendação apenas a conta, sem o usuário.
7. **Recomendação para a sessão**: Com o usuário da conta é gerada uma recomendação para o usuário da nova sessão.

3.3 Cálculo da Similaridade por Cosseno

Para fazer uso do *Affinity Propagation* é necessária uma forma de medir a similaridade entre as sessões. Jiang et al. (2018) sugere uma métrica calculada através do caminhamento por um grafo montada com a relação de itens e metadados. Como não podemos usufruir de metadados foi escolhida uma métrica de similaridade mais simples, a similaridade por cosseno, uma das medidas de similaridade entre itens mais comuns (TAN; STEINBACH; KUMAR, 2006).

Podemos representar as sessões por vetores onde cada espaço do vetor corresponde a um item que o usuário interagiu e o valor é a quantidade de vezes que a interação ocorreu com esse item. Considerando o total de itens em sistemas de e-commerce, mídia, e afins, são poucos os itens que um usuário interage em cada sessão, os vetores das sessões são então esparsos, ou seja, têm poucos elementos diferentes de zero.

A similaridade por cosseno de duas sessões s_1 e s_2 é então definida como:

$$cos(s_1, s_2) = \frac{s_1 \cdot s_2}{\|s_1\|\|s_2\|}$$

A similaridade por cosseno utiliza os itens em comum entre as sessões para medir a similaridade, então é necessário que esses itens se repitam frequentemente de uma sessão para outra. Em casos em que raramente ocorre a interação de um usuário com itens

repetidos, ou seja, para os itens que o usuário interagiu o número de interações com cada um deles é na média próximo de 1, sugere-se utilizar algum agrupamento, no lugar dos itens. Por exemplo, em um site de notícias são raros os casos em que um usuário vê uma mesma notícia mais de uma vez, isso implica que frequentemente a comparação entre duas sessões será igual a zero, então o melhor a se utilizar seria a categoria da notícia no lugar da mesma para o cálculo de similaridade, isso é demonstrado mais adiante no capítulo 4.

O *Affinity Propagation* recebe como entrada as similaridades entre itens. O passo do cálculo dessas similaridades pode ser visto no Algoritmo 1 da linha 1 à 5.

3.4 Affinity Propagation

Seguindo as mesma ideias propostas por Jiang et al. (2018), utilizamos o algoritmo de *Affinity Propagation* para a *clusterização*. Nesta seção explicaremos esse algoritmo em detalhes.

O algoritmo de *Affinity Propagation* funciona através da passagem de mensagens entre itens, que no nosso caso são as sessões. Ele recebe como entrada as similaridades $s(i,k)$ entre cada item i e k (e.g., a similaridade por cosseno), sendo a similaridade de um item como ele mesmo $s(k,k)$ chamada de preferência, sendo que maiores valores de preferência torna o item mais provável de ser escolhido como centro do *cluster*. A cada iteração são passados dois tipos de mensagens: as *responsabilidades* e as *disponibilidades*. A *responsabilidade* $r(i,k)$ representa o quão adequado é o item k para servir como exemplar do item i, comparado com os outros potenciais exemplares. Enquanto que a *disponibilidade* $a(i,k)$ reflete o quão apropriado seria para o item i escolher o item k como exemplar, levando em conta o apoio de outros itens em que o item k deveria ser um exemplar. Inicialmente as *disponibilidades* são iniciadas com zero $a(i,k) = 0$. E as *responsabilidades* são calculadas com a seguinte fórmula:

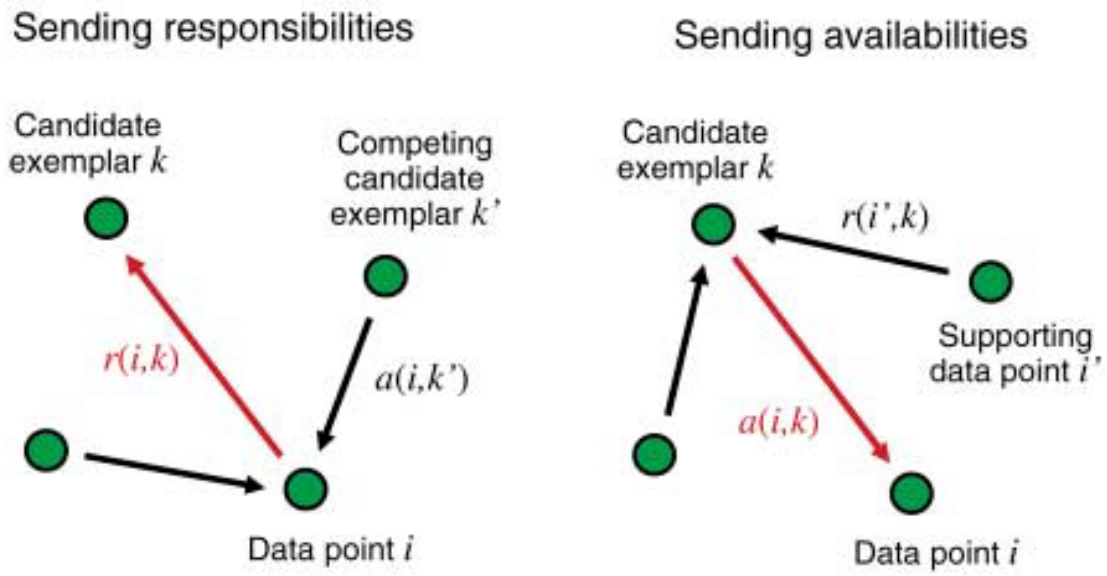
$$r(i,k) \leftarrow s(i,k) - \max_{k' \text{s.t. } k' \neq k} \{a(i,k') + s(i,k')\}$$

Isso faz com que $r(i,k)$ seja inicializado com as preferências dos pontos menos a maior das similaridades. A cada iteração, os valores das *responsabilidades* são atualizados com a fórmula acima e, seguidamente, as *disponibilidades* são atualizadas com base na soma das *responsabilidades*, conforme a seguinte equação:

$$a(i,k) \leftarrow \min\left\{0, r(k,k) + \sum_{i' \text{s.t. } i' \notin \{i,k\}} \max\{0, r(i',k)\}\right\}$$

A Figura 3.2 demonstra a passagem de mensagens entre pontos. Na esquerda, responsabilidade $r(i,k)$ sendo enviada do ponto i para o candidato a exemplar k para indicar o quanto o ponto i favorece o exemplar sobre os outros candidatos. Na direita, disponibilidade $a(i,k)$ sendo enviada do candidato a exemplar k para o ponto i para indicar o quanto ele está disponível para ser o centro de um *cluster* para o ponto i.

Figura 3.2: Passagem de mensagens do *Affinity Propagation*



Fonte: (FREY; DUECK, 2007)

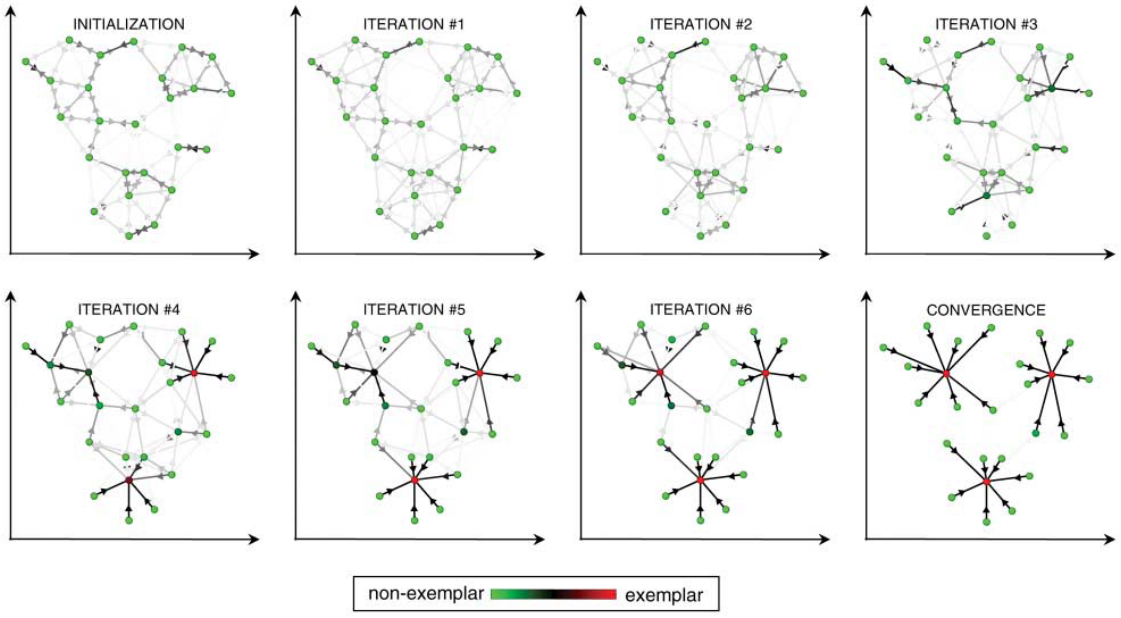
Esses valores são iterativamente atualizados até que tenha se passado um certo número de iterações, que a mudança seja menor que um dado valor ou que as decisões de escolha de representantes se mantenham constantes por um dado número de iterações. Para evitar que durante a atualização dos valores ocorra uma oscilação muito grande, é aplicado um fator de amortecimento λ. Cada mensagem é definida com a multiplicação de λ pelo valor da última iteração mais $1-\lambda$ vezes o valor da iteração atual, sendo λ entre 0 e 1. As equações atualizadas seguem:

$$r(i,k) \leftarrow \lambda r(i,k) + (1-\lambda)\left[s(i,k) - \max_{k' \text{s.t. } k' \neq k}\{a(i,k') + s(i,k')\}\right]$$

$$a(i,k) \leftarrow \lambda a(i,k) + (1-\lambda)\left[\min\left\{0, r(k,k) + \sum_{i' \text{s.t. } i' \notin \{i,k\}} \max\{0, r(i',k)\}\right\}\right]$$

A Figura 3.3 demonstra as iterações do *Affinity Propagation* para pontos em um espaço de duas dimensões, onde a similaridade entre os pontos é o inverso da distância euclidiana. A cor de um ponto indica a atual evidência de que ele é o centro de um *cluster* (exemplar). A opacidade das flechas indica a intensidade das mensagens que representam o quanto o ponto de onde sai a flecha pertencem ao exemplar no outro fim da flecha.

Figura 3.3: Iterações do *Affinity Propagation*



Fonte: (FREY; DUECK, 2007)

Algoritmo 1: Etapa *offline*

input : Conta a, suas sessões $S(a)$ e a função de similaridade f
output: Usuários da conta e sessões categorizadas por usuário

for $i \leftarrow 1$ **to** $|S(a)|$ **do**
  for $j \leftarrow 1$ **to** $|S(a)|$ **do**
    $Similarities_{i,j} \leftarrow f(s_i, s_j)$
  end
end
$C \leftarrow affinityPropagation(Similarities)$
$U \leftarrow C.centers$
for $i \leftarrow 1$ **to** $|S(a)|$ **do**
  $sU_i \leftarrow (s_i, C.labels_i)$
end
return (U, sU)

Algoritmo 2: Etapa *online*

input : Nova sessão s da conta a contendo T_s itens e a função de similaridade f
output: Recomendações para o usuário

if $T_s < minItems$ **then**
  return $RecomendationForAccount(a)$
end
$U \leftarrow GetUsersForAccount(a)$
Inicializa $bestUSim$ e $USim$ com valores nulos
for $u \in U$ **do**
  $USim \leftarrow f(S, c)$
  if $USim > bestUSim$ **then**
    $bestUSim \leftarrow USim$
    $bestU \leftarrow u$
  end
end
if $bestU$ *is null* **then**
  return $RecomendationForAccount(a)$
else
  return $RecommendationForUser(bestU)$
end

3.5 Considerações Finais

Similarmente ao *framework* SHE-UI, nosso método ocorre em duas etapas. A etapa *offline* (chamada de UI-Past no SHE-UI), onde são identificados os diferentes usuários para cada conta. E posteriormente a etapa *online* (chamada de UI-New no SHE-UI), que ocorre no momento em que o usuário está fazendo uso do sistema em uma nova ses-

são, e onde prevemos o usuário baseado nos usuários encontrados no passo *offline*. Nosso método, por padrão, utiliza a similaridade por cosseno, uma métrica mais simples que a do SHE-UI, será demonstrado que isso torna ele menos preciso no capítulo 4, porém ele pode ser aplicado em uma vasta gama de sistemas independentemente do domínio, pois não faz uso de metadados.

4 EXPERIMENTOS

Neste capítulo demonstramos os experimentos realizados sobre três bases de dados distintas. O objetivo é avaliar a performance do algoritmo para *clusterizar* sessões em diferentes cenários. Nossas questões de pesquisa são: (i) é possível utilizar *clustering* para identificar contas compartilhadas em cenários diversos? e (ii) Qual é a efetividade do uso de *clustering* para separar usuários?

Na seção 4.1 descrevemos como foram montados os experimentos, as tecnologias utilizadas e a metodologia. Falamos sobre as bases de dados, suas propriedades e pré-processamentos aplicados, na seção 4.2. Descrevemos as métricas utilizadas para avaliar os experimentos na seção 4.3. For fim, na seção 4.4, mostramos e avaliamos os resultados obtidos em cada uma das bases.

4.1 Configuração dos Experimentos e Metodologia

Os experimentos foram implementados na linguagem *Python* utilizando bibliotecas para computação científica e mineração e análise de dados *NumPy* (OLIPHANT, 2006), *Scikit-learn* (PEDREGOSA et al., 2011) e *Pandas* (MCKINNEY, 2010), e para computação paralela e de alta performance usamos *Numba* (LAM; PITROU; SEIBERT, 2015).

Foram utilizadas três bases de dados: de lista de compras de mercado do Instacart (2017), de cliques em notícias da globo.com[1] e histórico de escuta de músicas do Last.fm[2], cada uma com suas peculiaridades descritas na seção 4.2. Nenhuma das bases possuía dados de contas compartilhadas, então foram criados dados sintéticos juntando contas de usuários de forma similar ao método feito por Verstrepen e Goethals (2015), ou seja, 25% das contas têm 1, 2, 3 e 4 usuários, respectivamente. Apesar dessa abordagem não ser perfeita, Zhang et al. (2012) demonstram que as propriedades de contas compartilhadas sintéticas são muito similares as reais.

Para as três bases foram seguidos os mesmos passos principais. Após o pré-processamento específico de cada base, foram feitas as combinações de contas de usuários e então, para cada conta de usuário gerada, foi aplicada a métrica de similaridade entre as sessões e o algoritmo de *Affinity Propagation* com essas similaridades. As *clusterizações*

[1]Extraído de: <https://www.kaggle.com/gspmoreira/news-portal-user-interactions-by-globocom>
[2]Extraído de: <https://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html>

foram avaliadas pelas três métricas *ARI*, *AMI* e *FMI*, descritas na sessão 4.3, e, para fins de comparação, as mesmas métricas foram aplicadas sem *clusterização*, ou seja, considerando como se as sessões fossem sempre do mesmo usuário. Os resultados obtidos são mostrados na sessão 4.4.

4.2 Descrição e Construção das Bases de Dados

Nesta sessão são descritas as bases de dados usadas. Na subseção 4.2.1 é descrita a base de compras de mercado do Instacart (2017). Descrevemos a base de dados da globo.com na subseção 4.2.2. E por fim, na subseção 4.2.3 temos a descrição da base de dados da Last.fm.

4.2.1 Compras de Mercado do Instacart

Instacart (2017) provê uma base de dados pública de compras de mercado de seus usuários, com um pouco mais de 3 milhões de listas de compras de aproximadamente 200 mil usuários. Cada usuário possui listas de compras e cada lista de compras é formada pelos produtos que foram comprados. Para essa base de dados, cada sessão é a lista de itens comprados para cada ordem de compra. Algumas propriedades dessa base são:

1. Compras de um mesmo produto por um mesmo usuário ocorrem com certa frequência, como mostrado na Figura 4.1.
2. Itens nunca se repetem em uma mesma sessão.

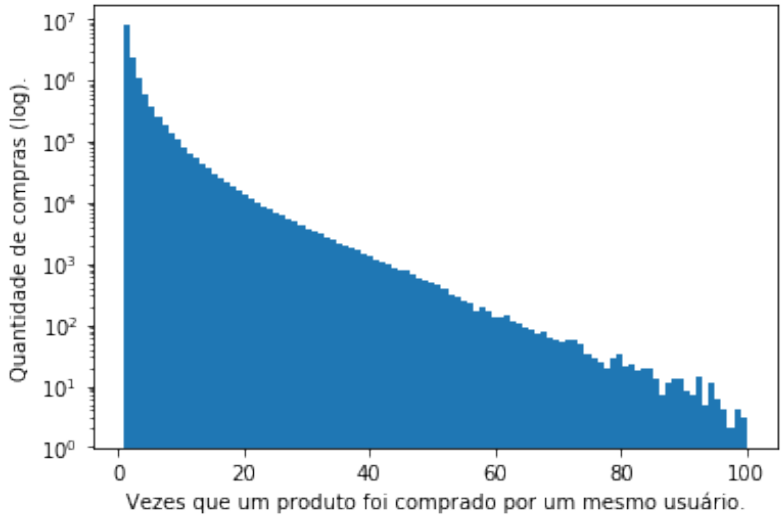
Foram retirados da base os itens comprados menos de dez vezes no total, também as listas com menos de dez itens e os usuários com menos de dez sessões. Os dados antes e depois dos filtros podem ser vistos na Tabela 4.1.

Tabela 4.1: Quantidade de dados do Instacart

	Original	Após o filtro
Itens	49,685	42,750
Listas de compras	3,346,083	980,645
Usuários	206,209	46,704
Contas		18,680

Fonte: O Autor

Figura 4.1: Histograma do número de compras de um produto por um mesmo usuário na base do Instacart (2017) em base logarítmica.



Fonte: O Autor

4.2.2 Cliques em Notícias da globo.com

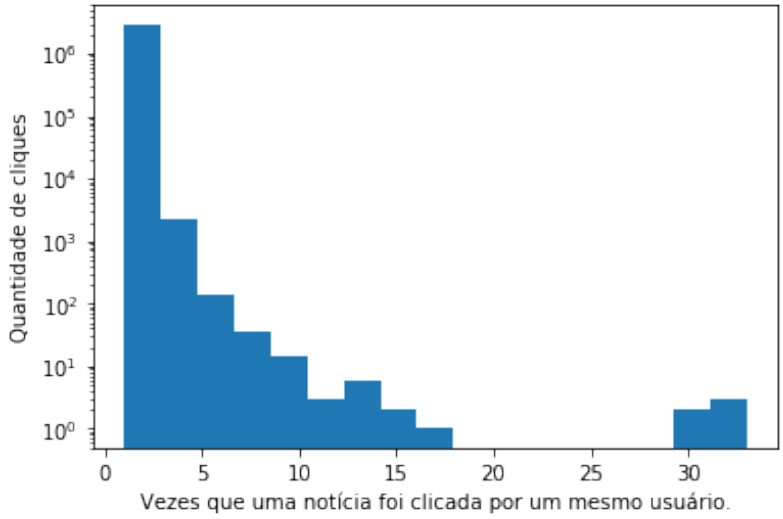
A globo.com provê a base de dados de interações de usuários com sua plataforma que foi utilizada nos trabalhos feitos por Moreira, Ferreira e Cunha (2018) e Moreira, Jannach e Cunha (2019). Essa base conta com dados de mais de 300 mil usuários com quase 3 milhões de interações no total. As interações se restringem a cliques em notícias. Algumas propriedades dessa base são:

1. Sessões são os intervalos de atividade do usuário no site. Cortados a cada 30 min de inatividade.
2. Muitas sessões são curtas (60% das sessões com apenas 2 itens e 21% com 3).
3. Usuários raramente clicam na mesma notícia mais de uma vez, tanto na mesma sessão quanto em outras. Isso é mostrado na Figura 4.2.

Pela terceira propriedade, vemos que sessões de um mesmo usuário terão similaridade baixas por raramente conterem itens em comum. Sendo assim, usamos a categoria das notícias como itens da clusterização.

Foram retiradas as categorias que aparecem menos de dez vezes, o que não causou nenhuma redução significativa na base. Usuários veem poucas notícias por sessão e muitos usuários têm poucas sessões, um corte de notícias com menos de dez itens e usuários com menos de dez sessões nos deixaria apenas com 71 usuários, mas ao mesmo tempo, não conseguimos medir a similaridade entre sessões com poucas notícias. Fazemos então

Figura 4.2: Histograma do número de cliques em uma notícia por um mesmo usuário na base da globo.com em base logarítmica.



Fonte: O Autor

o corte em notícias com menos de quatro itens, e usuários com menos de quatro sessões. Os dados antes e depois dos filtros podem ser vistos na Tabela 4.2.

Tabela 4.2: Quantidade de dados da globo.com

	Original	Após o filtro
Categorias	316	200
Sessões	1,048,594	84,498
Usuários	322,897	12,774
Contas		5,108

Fonte: O Autor

4.2.3 Histórico de Escuta de Músicas do Last.fm

O Last.fm (CELMA, 2010) provê uma base de dados de para recomendação de músicas. Ela contém o histórico de mil usuários desde fevereiro de 2005 à maio de 2009. Algumas propriedades dela são:

1. As sessões são longas (52% delas têm mais que 10 itens).
2. Usuários não repetem muito a mesma música em uma mesma sessão (apenas 9% das sessões possuem músicas repetidas).
3. Usuários repetem músicas entre várias sessões (52% das vezes que um usuário ouviu uma música, ele a ouviu de novo em outra sessão).

Nessa base as músicas não estão separadas por sessão, então criamos a separação

da mesma forma que Jiang et al. (2018): usamos o tempo de 30 minutos de inatividade como divisor entre sessões. Excluímos itens que aparecem menos que dez vezes, sessões com menos de dez itens e usuários com menos de dez sessões.

4.3 Métricas de Avaliação

Para verificar que a quantidade de usuários preditos está próxima as reais, foram utilizadas as métricas *Mean Absolute Error* (MAE) e *Root Mean Squared Error* (RMSE). Para medir a qualidade das *clusterizações* foram usadas as métricas *Adjusted Rand Index* (ARI), *Adjusted Mutual Information* (AMI) e *Fowlkes-Mallows Index* (FMI), explicadas nas seções 4.3.1, 4.3.2 e 4.3.3, respectivamente.

4.3.1 Adjusted Rand Index

O *Adjusted Rand Index* (HUBERT; ARABIE, 1985) é uma métrica que mede a similaridade entre *clusterizações* através do quanto as duas clusterizações concordam entre os pares de itens. Ela é uma métrica derivada da função *Rand Index* (RAND, 1971). Considerando $X = \{X_1, \ldots, X_R\}$ e $Y = \{Y_1, \ldots, Y_C\}$ duas *clusterizações* de um dado conjunto de n objetos $S = \{o_1, \ldots, o_n\}$, *Rand Index* (RI) é definido pela seguinte equação:

$$RI = \frac{a+b}{C_2^{n_{samples}}} = \frac{a+b}{\binom{n_{samples}}{2}}$$

$a = |S^*|$, onde $S^* = \{(o_i, o_j) \,|o_i, o_j \in X_k, o_i, o_j \in Y_l\}$ onde $1 \leq k \leq R, 1 \leq l \leq C$

$b = |S^*|$, onde $S^* = \{(o_i, o_j) \,|o_i \in X_{k_1}, o_j \in X_{k_2}, o_i \in Y_{l_1}, o_j \in Y_{l_2}\}$ onde $1 \leq k_1, k_2 \leq R, 1 \leq l_1, l_2 \leq C$

O valor de a é a contagem de todos os pares de objetos que pertencem ao mesmo *cluster* em X e ao mesmo em Y, ou seja, é a concordância entre as *clusterizações* de que os dois objetos devem pertencer ao mesmo *cluster*. Já o valor de b é, analogamente a a, a contagem de todos os pares de objetos que pertencem a *clusters* diferentes em X e diferentes em Y, ou seja, a concordância entre as *clusterizações* de que os dois objetos devem pertencer a *clusters* diferentes. A soma $a + b$ indica todas as concordâncias das *clusterizações* e, por fim, esse valor é dividido pelo número total de combinações de pares

de itens. Intuitivamente, o RI é a probabilidade das *clusterizações* concordarem em um par de objetos escolhidos aleatoriamente.

O RI não funciona bem para *clusterizações* aleatórias em casos em que o número de *clusters* está próximo ao número de itens. O ARI corrige esse problema. O ARI pode ser definido pela seguinte equação:

$$ARI = \frac{RI - E(RI)}{max(RI) - E(RI)}$$

Onde $E(RI)$ é o valor esperado para o RI e $max(RI)$ é o valor máximo. O ARI varia entre -1 e 1, sendo zero o valor dado a uma clusterização aleatória, e 1 o valor de clusterizações iguais.

4.3.2 Adjusted Mutual Information

O *Adjusted Mutual Information* (VINH; EPPS; BAILEY, 2010) é uma versão corrigida do *Mutual Information* para clusterizações aleatórias. O *Mutual Information* (MI) mede a correlação entre *clusterizações* utilizando a ideia de entropia. Considerando $X = \{X_1, \ldots, X_R\}$ e $Y = \{Y_1, \ldots, Y_C\}$ duas *clusterizações* de um dado conjunto de n objetos $S = \{o_1, \ldots, o_n\}$, o MI é dado pela seguinte equação:

$$MI(X, Y) = \sum_{i=1}^{R} \sum_{j=1}^{C} P(i, j) \log \frac{P(i, j)}{P(i)P'(j)}$$

Onde $P(i)$ é a probabilidade de que se um objeto de S for escolhido aleatoriamente, ele está em X_i. Analogamente, $P'(j)$ é a probabilidade desse objeto estar em Y_i:

$$P(i) = \frac{|X_i|}{n}$$

$$P'(j) = \frac{|Y_j|}{n}$$

E $P(i, j)$ é a probabilidade desse item estar tanto em X_i quanto em Y_j:

$$P(i, j) = \frac{|X_i \cap Y_j|}{}$$

Assim como o RI, o MI não funciona bem para clusterizações aleatórias, o AMI

corrige esse problema de forma similar ao ARI:

$$AMI = \frac{MI - E(MI)}{mean(H(X), H(Y)) - E(MI)}$$

O AMI varia entre 0 e 1, sendo zero o valor dado a uma clusterização aleatória, e 1 o valor de clusterizações iguais.

4.3.3 Fowlkes-Mallows Index

O *Fowlkes-Mallows Index* (MORLINI; ZANI, 2012) é a média geométrica da precisão e revocação para cada par de itens. Considerando $X = \{X_1, \ldots, X_R\}$ e $Y = \{Y_1, \ldots, Y_C\}$ duas *clusterizações* de um dado conjunto de n objetos $S = \{o_1, \ldots, o_n\}$, o *Fowlkes-Mallows Index* (FMI) pode ser descrito pela seguinte equação:

$$FMI = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}}$$

Onde TP são os verdadeiros positivos, i.e., os pares de itens que pertencem ao mesmo *cluster* tanto em X quanto em Y. FP são os falsos positivos, i.e., os pares de itens que pertencem ao mesmo *cluster* em Y, mas *clusters* diferentes em X. E o FN são os falsos negativos, i.e., os pares de itens que pertencem ao mesmo *cluster* em X, mas *clusters* diferentes em Y.

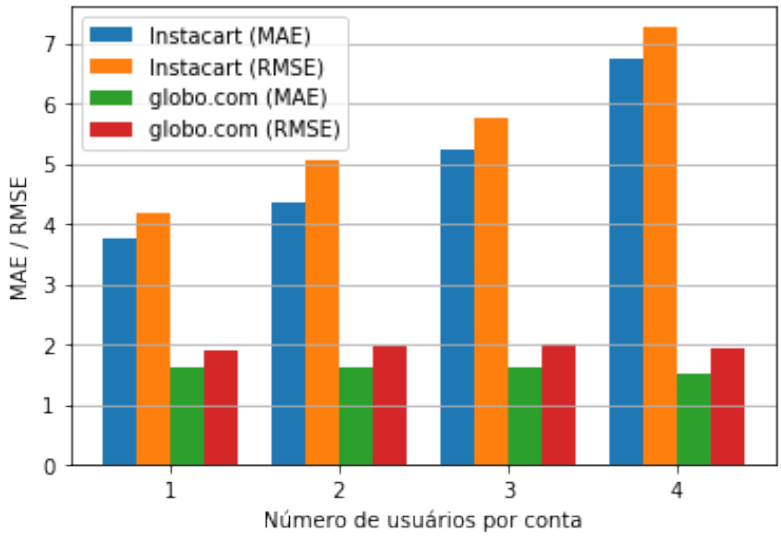
O FMI varia de 0 a 1, sendo zero o valor dado a uma clusterização aleatória, e 1 o valor de clusterizações iguais.

4.4 Descrição dos Experimentos e Análise dos Resultados

Nossas questões de pesquisa são: (i) é possível utilizar *clustering* para identificar contas compartilhadas em cenários diversos? e (ii) Qual é a efetividade do uso de *clustering* para separar usuários. Para avaliar a questão (i) medimos o MAE e RMSE da quantidade de usuários preditos em comparação com o real número de usuários. Os resultados dessas métricas são mostrados na Figura 4.3. Para a questão (ii) medimos o ARI, AMI e FMI das *clusterizações* (representados pelo *AP*) e comparamos com essas métricas medidas sem *clusterização* (representados por *sem AP*). Os resultados são demonstrados nas tabelas 4.3, 4.4 e 4.5.

A primeira coisa a se notar é que o caso sem *clusterização* obteve score máximo para 1 usuário por conta, o que é esperado, visto que há exatamente 1 *cluster* para representar o único usuário da conta. Outro fato é que o caso sem *clusterização* obteve valores altos para o FMI, isso se deve ao fato de que, por estar colocando todos no mesmo *cluster*, deixam de existir falsos positivos, levando ao valor máximo na revocação e deixando apenas a raiz da precisão no FMI.

Figura 4.3: Comparação de MAE e RMSE para cada uma das bases



Fonte: O Autor

Na base dados da globo.com foi onde se obteve os menores MAE e RMSE (visto na Figura 4.3), o que indica que melhor aproximou o número de usuários. Isso é visto também pela Tabela 4.4, que foi a única base de dados onde se obteve resultados levemente significativos para o caso de 1 usuário por conta. Isso se deve ao fato de que usuários possuem poucas sessões, deixando o número máximo de *clusters* que o *Affinity Propagation* pode encontrar muito próximo do número exato de usuários.

A base com os melhores resultados foi a do Instacart, visto na tabela 4.3. Isso mostra como a repetição de itens entre sessões é importante para melhores resultados da métrica de similaridade por cosseno.

Na base do Last.fm se obtiveram os piores resultados, demonstrados pela tabela 4.5. Isso se deve às sessões muito longas. Mesmo que haja repetições de itens entre elas, esses em comparação com o total de itens são poucos, o que torna a similaridade por cosseno muito baixa.

Tabela 4.3: Resultados para o Instacart

	ARI		AMI		FMI	
Usuários por conta	AP	Sem AP	AP	Sem AP	AP	Sem AP
1	0.0000	**1.0000**	0.0000	**1.0000**	0.5131	**1.0000**
2	**0.3623**	0.0000	**0.3762**	0.0000	0.5997	**0.7307**
3	**0.4388**	0.0000	**0.4883**	0.0000	**0.6125**	0.6091
4	**0.4586**	0.0000	**0.5323**	0.0000	**0.6053**	0.5350
Total	**0.3149**	0.2500	**0.3492**	0.2500	0.5826	**0.7187**

Fonte: O Autor

Tabela 4.4: Resultados para globo.com

	ARI		AMI		FMI	
Usuários por conta	AP	Sem AP	AP	Sem AP	AP	Sem AP
1	0.0313	**1.0000**	0.0313	**1.0000**	0.6274	**1.0000**
2	**0.1848**	0.0000	**0.1600**	0.0000	0.5034	**0.6997**
3	**0.1752**	0.0000	**0.1748**	0.0000	0.4102	**0.5781**
4	**0.1707**	0.0000	**0.1853**	0.0000	0.3593	**0.5040**
Total	0.1405	**0.2500**	0.1378	**0.2500**	0.4751	**0.6954**

Fonte: O Autor

Tabela 4.5: Resultados para Last.fm

	ARI		AMI		FMI	
Usuários por conta	AP	Sem AP	AP	Sem AP	AP	Sem AP
1	0.0000	**1.0000**	0.0000	**1.0000**	0.1780	**1.0000**
2	**0.0153**	0.0000	**0.0826**	0.0000	0.1298	**0.8049**
3	**0.0149**	0.0000	**0.1166**	0.0000	0.1166	**0.7050**
4	**0.0146**	0.0000	**0.1391**	0.0000	0.1044	**0.6354**
Total	0.0112	**0.2500**	0.0112	**0.2500**	0.1307	**0.7863**

Fonte: O Autor

5 CONCLUSÃO

Neste trabalho foi proposta uma solução genérica que se utiliza de poucos ou nenhum metadado para encontrar usuários em sessões de contas compartilhadas, baseada no método SHE-UI (JIANG et al., 2018), porém utilizando similaridade por cosseno. A principal motivação do desenvolvimento desse trabalho é encontrar uma forma de melhorar a recomendação em sistemas de e-commerce onde temos poucos metadados (e.g., nome do produto e categoria).

Demonstramos como essa solução pode ser usada com um Sistema de Recomendação e, não apenas limitados a e-commerce, validamos a funcionalidade dessa solução em diversos cenários, de e-commerce, notícias e mídia, apresentando resultados significativos. Obtivemos MAE próximo de 1,6 e RMSE próximo de 2 no melhor caso, mostrando que o algoritmo consegue prever de forma aproximada a quantidade de usuários na conta. Também obtivemos resultados melhores que sem *clusterização* em um dos cenários, que mostra que há domínios em que a solução pode ser aplicada.

Trabalhos futuros podem aplicar a solução proposto por Jiang et al. (2018) nas mesmas bases de dados onde foi aplicado esse trabalho, e fazer um comparativo. Testar a real usabilidade dessa solução em Sistemas de Recomendação, e possivelmente comparar com estratégias de recomendação para grupos. Também, sabendo que o número de usuários em contas compartilhadas é pequeno, é possível analisar diferentes valores para as preferências das sessões para forçar um menor número de *clusters*. Podem ser testadas novas métricas de similaridade, visto que a arquitetura proposta deixa isso em aberto.

REFERÊNCIAS

ARLITT, M. Characterizing web user sessions. **SIGMETRICS Perform. Eval. Rev.**, ACM, New York, NY, USA, v. 28, n. 2, p. 50–63, sep. 2000. ISSN 0163-5999. Available from Internet: <http://doi.acm.org/10.1145/362883.362920>.

CELMA, O. **Music Recommendation and Discovery in the Long Tail**. [S.l.]: Springer, 2010.

DESHPANDE, M.; KARYPIS, G. Item-based top-n recommendation algorithms. **ACM Trans. Inf. Syst.**, ACM, New York, NY, USA, v. 22, n. 1, p. 143–177, jan. 2004. ISSN 1046-8188. Available from Internet: <http://doi.acm.org/10.1145/963770.963776>.

FREY, B. J.; DUECK, D. Clustering by passing messages between data points. **Science**, v. 315, p. 2007, 2007.

HUBERT, L.; ARABIE, P. Comparing partitions. **Journal of Classification**, v. 2, n. 1, p. 193–218, 1985. Available from Internet: <https://EconPapers.repec.org/RePEc:spr:jclass:v:2:y:1985:i:1:p:193-218>.

INSTACART. **The Instacart Online Grocery Shopping Dataset 2017**. 2017. <https://www.instacart.com/datasets/grocery-shopping-2017>. Acessado em: 31/03/2019.

JIANG, J.-Y. et al. Identifying users behind shared accounts in online streaming services. In: **The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval**. New York, NY, USA: ACM, 2018. (SIGIR '18), p. 65–74. ISBN 978-1-4503-5657-2. Available from Internet: <http://doi.acm.org/10.1145/3209978.3210054>.

LAM, S. K.; PITROU, A.; SEIBERT, S. Numba: A llvm-based python jit compiler. In: **Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC**. New York, NY, USA: ACM, 2015. (LLVM '15), p. 7:1–7:6. ISBN 978-1-4503-4005-2. Available from Internet: <http://doi.acm.org/10.1145/2833157.2833162>.

MCKINNEY, W. Data structures for statistical computing in python. In: WALT, S. van der; MILLMAN, J. (Ed.). **Proceedings of the 9th Python in Science Conference**. [S.l.: s.n.], 2010. p. 51 – 56.

MOREIRA, G. de S. P.; FERREIRA, F.; CUNHA, A. M. da. News session-based recommendations using deep neural networks. In: **Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems**. New York, NY, USA: ACM, 2018. (DLRS 2018), p. 15–23. ISBN 978-1-4503-6617-5. Available from Internet: <http://doi.acm.org/10.1145/3270323.3270328>.

MOREIRA, G. de S. P.; JANNACH, D.; CUNHA, A. M. da. Contextual hybrid session-based news recommendation with recurrent neural networks. **CoRR**, abs/1904.10367, 2019. Available from Internet: <http://arxiv.org/abs/1904.10367>.

MORLINI, I.; ZANI, S. Dissimilarity and similarity measures for comparing dendrograms and their applications. **Adv. Data Anal. Classif.**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 6, n. 2, p. 85–105, jul. 2012. ISSN 1862-5347. Available from Internet: <http://dx.doi.org/10.1007/s11634-012-0106-2>.

OLIPHANT, T. E. **A guide to NumPy**. [S.l.]: Trelgol Publishing USA, 2006.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

RAND, W. Objective criteria for the evaluation of clustering methods. **Journal of the American Statistical Association**, v. 66, n. 336, p. 846–850, 1971.

RASTOGI, R. Machine learning @ amazon. In: **Proceedings of the 2Nd IKDD Conference on Data Sciences**. New York, NY, USA: ACM, 2015. (CODS-IKDD '15), p. 2:1–2:1. ISBN 978-1-4503-3616-1. Available from Internet: <http://doi.acm.org/10.1145/2778865.2778867>.

RICCI, F. et al. **Recommender Systems Handbook**. 1st. ed. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN 0387858199, 9780387858197.

SEMBIUM, V. et al. Bayesian models for product size recommendations. In: **Proceedings of the 2018 World Wide Web Conference**. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018. (WWW '18), p. 679–687. ISBN 978-1-4503-5639-8. Available from Internet: <https://doi.org/10.1145/3178876.3186149>.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. [S.l.]: Pearson Education, 2006.

VERSTREPEN, K.; GOETHALS, B. Top-n recommendation for shared accounts. In: **Proceedings of the 9th ACM Conference on Recommender Systems**. New York, NY, USA: ACM, 2015. (RecSys '15), p. 59–66. ISBN 978-1-4503-3692-5. Available from Internet: <http://doi.acm.org/10.1145/2792838.2800170>.

VINH, N. X.; EPPS, J.; BAILEY, J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. **J. Mach. Learn. Res.**, JMLR.org, v. 11, p. 2837–2854, dec. 2010. ISSN 1532-4435. Available from Internet: <http://dl.acm.org/citation.cfm?id=1756006.1953024>.

ZHANG, A. et al. Guess who rated this movie: Identifying users through subspace clustering. **CoRR**, abs/1208.1544, 2012. Available from Internet: <http://arxiv.org/abs/1208.1544>.