UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO
LEANDRO JOSÉ MEYER MACHADO

RESOLUÇÃO DE ENTIDADES PARA SISTEMAS DE RECOMENDAÇÃO

Florianópolis, 2013.2

Leandro José Meyer Machado

RESOLUÇÃO DE ENTIDADES PARA SISTEMAS DE RECOMENDAÇÃO

Esta é uma proposta de Trabalho de Conclusão de Curso para a obtenção do grau de Bacharelado no Curso de Ciências da Computação na Universidade Federal de Santa Catarina

Professor Responsável:

Vitório Bruno Mazzola

Orientador externo:

Tiago Macambira

Banca Examinadora:

Carina Friedrich Dorneles

Renato Fileto

Florianópolis, 2013.2

RESUMO

Com o crescimento da disponibilidade de informações e do surgimento de e-commerces, é comum que cada vez mais estas lojas on-line tentam utilizar essa informação para personalizar seu site e aumentar o número de engajamento e de vendas. Com isso e-commerces estão acoplando Sistemas de Recomendação à sua arquitetura. No entanto, por natureza, catálogos de e-commerces são atualizados constantemente, com vários produtos sendo diariamente adicionados e removidos. Além de lojas menores levarem mais tempo para vencer a inércia do cold-start e adquirir uma quantidade mínima de enventos para gerar boas recomendações.

Neste trabalho é proposto um processo em MapReduce de Resolução de Entidades o qual é acoplado em um Sistema de Recomendação, visando aumentar sua qualidade e cobertura. Os resultados são então avaliados quantitativamente e qualitativamente através de uma ferramenta de teste cego. A ferramenta foi exclusivamente desenvolvida para utilizar na análise qualitativa. Por fim são apresentadas as conclusões sobre as vantagaens e desvantagens de se usar a resolução de entidades para aprimorar as recomendações.

Palavras-chave: sistemas de recomendação, resolução de entidades, deduplicação.

SUMÁRIO

ÍNDICE DE ILUSTRAÇÕES

ÍNDICE DE TABELAS

1 INTRODUÇÃO

Sistemas de recomendação, segundo Ricci(2011), são um conjunto de softwares e técnicas que visam encontrar items que sejam úteis para um usuário dentro de um domínio de items. Este tem suas raízes em diversas disciplinas como teoria da informação, ciências cognitivas e até mesmo marketing. Em meados dos anos 90 surgiram as primeiras ideias para agregar essas partes em sistemas de inteligência coletiva, o que deu o ponta pé inicial para que Sistemas de Recomendação se consolidassem como área de pesquisa (Resnick,1994). Área a qual tem se tornado muito importante nos últimos anos devido a sua aplicabilidade em ajudar usuários a lidar com a sobrecarga de informações e por ser rica em problemas práticos ainda não resolvidos.

Atualmente, Sistemas de Recomendação(SR) são principalmente utilizados para recomendação de produtos para usuários de sites de compra on-line. Basicamente, de acordo com as compras anteriores do usuário, o sistema recomenda produtos similares, visando aumentar as vendas. Empresas como Amazon e Saraiva utilizam esse sistema em seus sites de venda. Grande parte dos SR são baseados em inteligência coletiva, que recomenda itens baseado nas interações dos usuários. No entanto esta técnica gera poucas recomendações e de baixa qualidade até que junte informações suficientes para sair da inércia(Cold Start). Este problema pode acontecer tanto a nível de sistema, no início da sua operação, quanto a nível de produtos e usuários, quando estes são novos no sistema. Adicionalmente, existem lojas que tem alto potencial de rentabilidade, no entanto estas apenas não geram eventos suficientes para que se possa usar a filtragem colaborativa.

Considerando estes problemas, este trabalho visa propor uma forma de identificar objetos similares, deduplicar usuários ou itens, entre diferentes bases de dados, por exemplo lojas on-line. Desta maneira será possível utilizar recomendações já computadas em uma base para itens ou usuários de outra base, e também aumentar a quantidade de eventos de bases pequenas, agregando eventos de objetos deduplicados, para melhorar a qualidade das recomendações. No entanto, o problema de deduplicar objetos é um problema recente e ainda não resolvido da computação. Neste trabalho, será proposto um processo que resolve este problema considerando as retrições de eficiência e escalabilidade associadas as soluções para Big data. Por fim sera realizado uma análise quantitativa e qualitativa dos resultados, esta última, utilizando uma ferramenta de teste cego.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver um processo de resolução de entidades para sistemas de recomendação para melhorar a quantidade e a qualidade das recomendações.

2.2 OBJETIVOS ESPECÍFICOS

a) Propor um processo de resolução de entidades confiável, com baixa taxa de Falso-Positivos.
b) Aprimorar o processo para que o mesmo tenha uma boa cobertura sobre as bases de dados, referente ao percentual de produtos que foram resolvidos.
c) Otimizar o processo proposto para que seja eficiente para rodar em sistemas de grande porte.
d) Implementá-lo num ambiente de Big data e testá-lo com bases de dados de grandes empresas.
e) Expandir o processo proposto para que seja possível resolver produtos entre lojas.

3 PROCEDIMENTOS METODOLÓGICOS

O início do trabalho se dará por um estudo apronfudado sobre a área de sistemas de recomendação, desde a formalização do problema até às soluções utilizadas atualmente. Com o conhecimento na área solidificado, serão analizadas as soluções existentes para o problema da resolução de entidades identificando: seus pontos fortes, suas limitações e a sua aplicabilidade em ambientes Big Data. A partir de então escolher-se-ão as melhores técnicas, as quais serão implementadas para analisar sua qualidade e eficiência, começando pela resolução exata e dentro de uma mesma base de dados. Na sequência, analisar como usar a resolução de entidade em um sismeta de recomendação de forma a aprimorar a cobertura, qualidade e diversidade da recomendações geradas. Prosseguindo, estudar e implementar técnicas aproximadas de resolução e utilizar o problema de casamento estável para atingir a resolução entre diferentes bases de dados. Finalmente, o processo será testado com bases de dados reais de grande escala para analisar sua qualidade e eficiência em ambientes de Big data. Todo o código será feito em Python e os processos rodados no Elastic MapReduce da Amazon.

4 FUNDAMENTAÇÃO TEÓRICA

4.1 SISTEMAS DE RECOMENDAÇÃO

Em muitos mercados on-line, vulgos e-commerces, usuários são apresentados uma imensa quantidade de produtos e informações dos quais podem escolher. No entanto como escolher os poucos produtos que os interessam dentro de centenas, milhares e até milhões de produtos apresentados? Do ponto de vista do e-commerce, como é possível fornecer ao usuário opções de produtos que ele possa gostar, ou que possam complementar uma compra passada, de forma a aumentar as chances de uma nova compra? Visando resolver estes problemas, lojas on-line incorporaram sistemas de recomendação aos seus websites (Resnick e Varian, 1997).

Sistemas de recomendação podem fornecer ao usuário uma lista de produtos que provavelmente o interessa, dado seu histórico de interações com o e-commerce. Ou seja, se um usuário visualizou o livro Harry Potter I, muito provavelmente se interessará pelo Harry Potter II. Ou caso o usuário vá finalizar a compra de uma televisão, talvez se interesse em adicionar um suporte para TV à compra. As recomendações ajudam tanto o usuário a encontrar o que ele procura, quanto a loja on-line a converter um visitante em um comprador.

O crescimento explosivo e a variedade de informação disponível na Web, e a rápida introdução de novos serviços de e-commerce (compra de produtos, comparação, leilões), frequentemente tem sobrecarregado os usuários com informações, os levando, geralmente, a fazer decisões ruins. A grande disponibilidade de opções, ao invés de trazer benefícios, começou a reduzir a qualidade da experiência do usuário na loja on-line. Percebe-se que enquanto é bom ter escolhas, mais escolhas nem sempre é melhor. Na verdade, opções de escolha, com suas implicações de liberdade, autonomia e auto-determinação, pode se tornar excessivo, criando um senso de que liberdade pode se tornar uma tirana causadora de arrependimentos (Schwartz, 2004)

Segundo Ricci(2011), Sistemas de Recomendação tem se provado úteis nos ultimos anos como uma forma valiosa de resolver o problema de sobrecarga de informação. Basicamente, um SR lida com este fenômeno direcionando o usuário para novos, ainda não experimentados, items que podem ser relevantes para a atual tarefa do usuário. Dado uma requisição do usuário, que pode ser devidamente articulada, dependendo da abordagem da recomendação, por contexto e necessidade, o SR gera recomendações usado varios tipos de conhecimento e dados sobre os usuário, itens disponíveis e interações anteriores armazenadas em bancos de dados customizados. O usuário pode então navegar pelas recomendações. Este pode aceitá-las ou não e então prover, imediatamente ou num próximo estagio, um feedback explícito ou implícito. Todas estas ações e *feedbacks* são armazenados no banco de dados do SR e podem então ser usados para gerar novas recomendações para os próximas interações e usuários.

Devido a sua necessidade surgir como consequência do crescimento da Web, o

estudo de sistemas de recomendação é relativamente novo comparado a outras ferramentas e técnicas clássicas de sistemas de informação. Somente na metade da década de 90 os sistemas de recomendação emergiram como uma área de estudo independente (Goldberg, 1992) e nos anos recentes o interesse neles tem crescido dramáticamente, conforme os seguintes fatos indicam:

1. Sistemas de recomendação fazem um papel importante em sites de alto renome como Amazon.com, Netflix, Youtube, IMDb, entre outros. Muitas dessas companias oferecem as recomendações como parte do serviço que proporcionam à seus usuários e continuamente desenvolvem e aperfeiçoam este serviço. Por exemplo a Netflix, o serviço de TV por internet, premiou com 1 milhão de doláres o time que conseguiu desenvolver um sistema de recomendação que recomendasse substancialmente melhor que o sistema deles.

2. Existem conferências e *workshops* especialmente dedicadas à área. Umas das principais referências é o ACM Recommender Systems (RecSys), estabelecido em 2007 e agora o mais reconhecido evento em pesquisa e applicação de tecnologias de recomendação. Complementando, muitas conferências tradicionais nas áreas de teoria da informação, bancos de dados, aprendizado de máquina tem sessões dedicadas a SR's. Algumas que valem comentar: ACM Special Interest Group on Information Retrieval (SIGIR), ACM's Special Interest Group on Management Of Data (SIGMOD) e User Modeling, Adaptation and Personalization (UMAP).

3. Em várias instituições de ensino superior ao redor do mundo existem cursos, tanto de gradução quanto de pós-graduação, que são inteiramente dedicados à sistemas de recomendação. Recentemente um livro foi publicado que introduz à técnicas de recomendação (Jannach 2010).

4. Tiveram diversas edições especiais de jornais academicos cobrindo pesquisa e desenvolvimento no campo de SR. Entre os jornais dedicados estão: AI Communications (2008); IEEE Intelligent Systems (2007); International Journal of Electronic Commerce (2006); International Journal of Computer Science and Applications (2006); ACM Transactions on Computer-Human Interaction (2005); e ACM Transactions on Information Systems(2004).

4.1.1 DEFINIÇÃO DO PROBLEMA DE RECOMENDAÇÃO

Em sua formulação mais comum, o problema de recomendação é reduzido ao problema de estimar notas para itens que ainda não foram vistos pelo usuário, ou os quais o usuário ainda não tenha consciência que existem. Intuitivamente, esta informação tem que ser derivada de notas que o usuário deu para outros itens, ou das notas que outros usuários deram para este item, ou de outras interações entre usuário e produtos. Assim que se conseguir estimar essas possíveis notas para os itens ainda não avaliados, pode-se então recomendar os itens com as maiores notas.

Mais formalmente o problema de recomendação pode ser definido como a seguir. Seja C o conjuntos de todos os usuários e S o conjunto de todos os itens que podem ser recomendados, como livros, filmes, produtos ou restaurantes. A quantidade de itens em S pode variar muito dependendo do contexto, indo de centenas a até milhões em alguns casos, como em lojas de artigos gerais. Da mesma maneira o conjunto U também pode ser imenso, em geral maior que o próprio conjunto S. Seja *u* a função que mede a utilidade do item *s* ao usuário *c*, do tipo *u*:CXS->R, onde R é um conjuto totalmente ordenado (como os inteiros, reais, etc.). Então para cada usuário *c* em C, é desejado escolher um item s' em S que maximize a função de utilidade (Adomavicius, 2005).

A função de utilidade pode ser definida de diversas maneiras, como o quão provavelmente um usuário vá comprar dado item, o quão similar o item é com os itens que o usuário procurar, se ele pode complementar a compra do usuário ou não. Esta pode ser tanto definida diretamente pelo usuário, como em avalições de produto, ou computado por um sistema, como o caso que a função de utilidade é baseda em lucros.

O problema central dos sistemas de recomendação se baseia em que a utilidade u geralmente não é definida em todo o espaço CXS, mas apenas em um subconjunto deste. O que siginifica que a função *u* necesita ser extrapolada para todo o espaço CxS. Às vezes este subconjunto é representado pelos itens que o usuário avaliou, mas em alguns sistemas este subconjunto é vazio e todas as notas tem que ser derivadas de ações implícitas dos usuários. Estas notas podem ser estimadas de diversas maneiras usando métodos de aprendizado de máquina, teoria da aproximação, extração de informação e várias heurísticas (Hill, In Proceedings of CHI'95). Sistemas de recomendação são geralmente classificados de acordo com a sua abordagem para estimar as notas. A seguir serão descritas as categorias nas quais SR's são geralmente classificados.

4.1.2 PRINCIPAIS ABORDAGENS EM SISTEMAS DE RECOMENDAÇÃO

O texto a seguir define diferentes tipos de sistema de recomendação, que variam em seus domínios abordados, suas formas de extrair informação, sua modelagem e, principalmente, no algoritmo de recomendação.

4.1.2.1 BASEADA EM CONTEÚDO (CONTENT-BASED)

Em recomendações baseadas em conteúdo a função de utilidade é estimada baseada na similaridade entre o conteúdo dos items a serem estimados. Por exemplo, em uma aplicação de recomendação de filmes, o SR baseado em conteúdo tenta entender as afinidades entre os filmes que o usuário assistiu e gostou (atores específicos, diretores, gêneros, etc.). Então, baseado nessas informações, ele recomenda filmes que o usuário não assistiu que contenham caracteristícas similares.

A abordagem baseada em conteúdo tem suas raízes na áreas de recuperação de informação e filtragem de informação. Devido aos avanços que tiveram as comunidades de pesquisa em filtragem e recuperação de informação e por causa da importância de aplicações baseadas em texto, como o Google, muitos sistemas baseados em conteúdo focam em recomendar itens que contenham informação textual, como documentos, produtos com descrição e páginas web. A diferença sobre as técnicas tradicionais de recuperação de informação vêm do uso de perfis de usuário que contêm informações sobre as preferências do usuário, gostos e necessidades. A informação do perfil pode ser extraída de forma explícita, através de questionários, ou de forma implícita, sendo aprendido ao longo do tempo através das interações do usuário (Adomavicius, 2005).

Alguns dos problemas desta abordagem é que ela esta estritamente limitada pelas técnicas de análise de conteúdo e às inferências possíveis através da similaridade de conteúdo. Portanto, em ordem de obter um conjunto suficiente de qualidades para definir um perfil, tanto de usuário quanto de itens, as informações têm que estar dispostas de forma que possam ser analisadas automaticamente, ou devem ser definidas manualmente. Também não é possível inferir relações de complementariedade entre itens, dado que a similaridade entre itens é dada unicamente pelas características que eles compartilham.

4.1.2.2 FILTRAGEM COLABORATIVA

Ao contrário da abordagem por conteúdo, sistemas de recomendação colaborativos, ou sistemas de filtragem colaborativa, tentam estimar a utilidade de um item para um dado usuário baseada na utilidade deste item para outros usuários. Em outras palavras, uma das melhores maneiras de saber se você vais gostar de um dado filme, música, ou produto, é perguntar para pessoas com gostos similares aos seus se eles gostaram. A filtragem colaborativa utiliza o histórico de interações dos usuários com os itens para encontrar correlações e fazer inferências de similaridade e complementariedade.

Uma das grandes vantagens deste método é que ele não depende de conteúdos analisáveis por máquina e portanto consegue recomendar itens complexos sem necessáriamente entender do que se trata o item. Utilizando medidas de correlação como o K-vizinhos-próximos, coeficiente de Pearson e até angulo entre vetores, é possível estimar a similaridade entre usuários, com base em suas interações com os itens, e então estimar a utilidade dos itens baseados nos usuários mais parecidos. No entanto, apesar deste modelo baseado em usuários gerar bons resultados em geral, ele não é fácil de escalar devido ao espaço de usuários ser relativamente grande para que possam ser comparados um com o outro e, subsequentemente, seus produtos em comum(Schafer, 2001).

Para contornar este problema da escalabilidade e esparsidade entre usuários, usa-se como alternativa um modelo baseado em similaridade de items, ou filtragem colaborativa baseada em items. Em casos nos quais conjuntos de dados são muito grandes, a filtragem baseada em conteúdo pode render resultados melhores, e permite que muitas das computações possam ser calculadas com antecedência. A técnica se parece muito com o modelo baseado em usuários, no entanto esta computa a similaridade de itens com outros items. Assim, basta recomendar itens similares ao que são de interesse do usuário. A grande diferença aqui é que, apesar de você ter que examinar os mesmos dados, comparações entre items não mudam tão frequentemente quanto comparações entre usuários(Segaran, 2007). Existem prova empíricas de que modelos baseados em itens tem melhor performance que modelos baseados em usuários (Desphande, 2004).

No entanto a filtragem colaborativa também tem seus problemas, independente to modelo utilizado. O principal deles é o problema de *cold-start*, ou “arranque frio”. Como esta abordagem depende basicamente da informação sobre as interações entre usuários e itens, é necessário que haja um quantidade mínima de informações para que o sistema seja capaz de inferir sobre a função de utilidade. Assim até que se junte informações suficientes, o sistema gera poucas recomendações e de baixa qualidade. Este problema pode acontecer tanto a nível geral do sistema, quanto a nível de novos usuários ou itens. Uma das maneiras de contornar esse problema é combinar com técnicas de sistemas baseados em conteúdo(Schain, 2002).

4.2 RESOLUÇÃO DE ENTIDADES

Organizações coletam informações do mundo real sobre entidades que os interessam. A entidade descrita pode ser física, como uma pessoa, ou casa, ou pode ser uma construção lógica, como uma família, rede social ou uma lista de pessoas que gostam de uma música em particular. Uma relação é a representação digital desta coleção. Essa relação é composta de um conjunto de entradas ou tuplas, cada uma pertencendo a uma entidade ou conjunto de entidades do mundo real. Cada tupla pode se referir a uma única entidade, mas cada entidade pode ter uma ou mais tuplas descrevendo ela. Isto é descrito em Hernandez e Stolfo (1998).

Frequentemente é desejado remover entradas duplicadas em uma relação, ou até identificar quando uma entrada esta contido em outra. No entanto isto não é uma tarefa fácil por inúmeras razões, como descrito por Muller e Freitag(2003). Resolução de entidades (Entity Resolution ou ER) é o problema de extrair, comparar e resolver menções de entidades em dados estruturados e desestruturados. Este é um desafio relativamente novo, sendo incluído nas áreas de gerência de dados, extração de informação, aprendizado de máquina, processamento de linguagem natural e estatística. Ironicamente, diversas disciplinas se referem a ER por uma variedade de nomes, incluindo linkagem de registros, deduplicação, resolução de co-referência, consolidação de objetos, incerteza de identidade e enxutamento de bancos de dados.

Devido a recente história da ER, existe uma supreendente diversidade de abordagens - como métodos baseados em regras, classificação por pares, técnicas de clustering e formas de inferência probabilística - e falta de uma liderança teórica. Enquanto isso, na era do Big data, a necessidade por ER de alta qualidade está crescendo. Casa vez mais sistemas são inundados com dados que precisam ser integrados, alinhados e casados para que seja extraído uma melhor utilidade deles. Uma ER rápida e precisa terá grandes implicações em uma variedade de domínios comerciais, científicos e de segurança.

4.2.1 CRITÉRIOS DE COMPARAÇÃO

Para resolver entidades é necessário pelo menos um ou mais critérios para decidir quando dois registros referem ao mesmo objeto. Quando um ou mais critérios são usados, estes necessitam estar conectados entre eles, podendo ser de maneira lógica ou ponderada, por exemplo. Esta relação é o que vai definir uma entidade. Esta relação pode ser, por exemplo, que todas pessoas que moram no mesmo endereço e compartilham o sobrenome se referem a mesma entidade. Grande parte da literatura foca em definir estes critérios baseados em comparação de textos. O que faz sentido já que grande parte dos dados é armazenado em texto.

A seguir, apresenta-se uma breve explicação dos principais critérios encontrados na literatura:

Exact match. O critério mais simples e direto seria uma comparação exata. Quando as tuplas tem o mesmo valor para um atributo em particular, estas são ditas se referirem à mesma entidade ou estão relacionadas. Por exemplo, se uma base de dados contém duas tuplas com nomes identicos e com o mesmo RG, estas podem ser consideradas como referentes à mesma pessoa.

Distance (approximate) match. Outro critério para comparação pode considerar que existe uma função que retorna uma “distância” entre duas tuplas, ou o nível de similaridade entre elas. Nesse caso geralmente se define um limite a partir do qual acima deste os objetos são considerados iguais. Algumas dessas funções são: correlação de Pearson, distância Euclidiana e número de palavras em comum.

Soundex. Duas strings podem ser convertidas para uma classe de representções fonéticas, e essas representações são comparadas entre si. Sarawagi & Bhamidipaty(2002) brevemente descrevem o uso de funções soundex para estabelecer similaridade entre tuplas. Este critério tem uma boa eficácia na teoria quando usada com palavras com diferentes formas de serem escritas, como “Daiani” e “Daiane”, que de outra maneira seriam difíceis de comparar.

TF/IDF. Frequência do termo sobre a frequência inversa nos documentos, é uma medida advinda do Processamento de Fala e Linguagem discutido em Cohen & Richman (2002). A idéia é determinar a frequência de uma string na base e favorecer strings menos frequentes, penalizando as mais comuns. Essa comparação requer conhecimento, ou derivação, da frequência de cada atributo considerado.

Clustering. Como descrito em Malin & Sweeney (2005), um grupo de tuplas com cracterísticas similares pode ser extraído de uma relação. Esse grupo pode ser estabelecido de varias maneiras, como co-presença e localização temporal ou geográfica. Este critério pode servir principalmente como critério auxiliar, por exemplo para reduzir o espaço de procura para a comparação exata.

4.2.2 APLICAÇÃO SOBRE RELAÇÕES

As forma de comparação descritas anteriormente podem ser aplicadas sobre tuplas e seus atributos para verificar se estas são co-referentes. No entanto a maneira como os critérios são aplicados sobre as relações (conjuntos de tuplas) pode variar resultando em diferentes complexidades e áreas de cobertura. Assim que os critérios são selecionados, uma dessas formas de aplicação devem ser escolhida. A seguir algumas dessas aplicações:

Força bruta. Por esta abordagem cada tupla em uma relação é comparada com cada outra tupla. Esta forma é simples de implementar e cobre toda relação, no entanto tem a maior complexidade em termos de tempo de execução($O(n^2)$). É necessario considerar também o tempo de execução do critério de comparação, já que este será rodado para cada tupla. Logo esta forma, apesar de simples, não factível para ser aplicado com Big data.

Janela móvel. Por esta abordagem as tuplas são ordenadas por algum critério, digamos nome, e uma janela de tamanho fixo, digamos *w*, é definida. Todos os pares das primeiras *w* tuplas são comparados de acordo com o critério especificado. Quando todas as tuplas dentro da janela foram foram comparadas com as outras, as próxima *w* tuplas são consideradas. A janela se move progressivamente pela lista de tuplas até que todas tenham sido analisadas. Já que o número de comparações em cada janela é constante, e o número de janelas é linear em respeito ao número de tuplas. O custo desta aplicação é dominado pelo custo de ordenar a relação, que é $O(n\log n)$. Uma aplicacação bem sucedida deste método é descrita em Hernandez & Stolfo (1998).

4.3 PROBLEMA DO CASAMENTO ESTÁVEL

4.3.1 DEFINIÇÃO ORIGINAL

Considere um grafo bipartido G = (U, V, E), onde U e V são os conjuntos de vértices e E o conjunto de arestas. Um *matching* M em G é um subconjunto de E tal que cada vértice aparece apenas uma vez em M. Estas combinações bipartidas são, as vezes, interpretadas como um casamento entre homem e mulher: U e V representam os conjuntos de homem e mulher, respectivamente, e a existência de uma aresta entre *m* em U e *w* em V implica que *m* e *w* são aceitáveis um para outro.

No Problema do Casamento estável (Stable Marriage Problem, ou SM), cada pessoa espressa não apenas a aceitabilidade, mas também uma ordem de preferência pelos membros do sexo oposto. E uma combinação proposta deve satisfazer a condição de estabilidade, que intuitivamente significa que não deve haver um par de homem-mulher no qual os dois preferiam estar casados com outra pessoa. Uma definição mais formal será dado em breve.

O problema foi primeiramente introduzido por Gale e Shapley em 1962 (Gale e Shapley, 1962) e tem recebido atenção de muitos pesquisadores por causa da sua inerente estrutura matemática e possíveis aplicações no mundo real. De fato, a introdução deste problema por Gale e Shapley foi motivado pela distribuição de estudantes de medicina para hospitais nos EUA, mais conhecido como National Resident Matching Program. Também, a noção de estabilidade é um conceito central em teoria de jogos cooperativos, onde busca-se determinar como uma constelação de individuos racionais escolhem com quem e como se cooperar (Nobel Prize Description).

Formalizando, uma instância *I* de um SM consiste de um número *n* de homens e mulheres. Cada pessoa tem uma lista de preferência *L* que estritamente ordena todos os membro do sexo oposto. Se um homem *m* prefere *w1* à *w2* então L(*m*,*w1*) > L(*m*,*w2*), onde função L retorna a posição da mulher na lista de um homem, ou vice-versa. Uma notação similar é usada para a preferência das mulheres. Um *matching M* é um conjunto desconectado de pares homem-mulher de *I*. Se, em um *matching M*, um homem *m* e uma mulher *w* são combinados juntos então M(*m*) = *w* e M(*w*) = *m*. Um homem *m* e uma mulher *w* formam um par de bloqueio, ou simplesmente (*m,w*) bloqueia M, se as seguintes condições são satisfeitas: (i) M(*m*) != *w* ; (ii) L(*m, w*) > L(*m,* M(*m*)) e (iii) L(*w, m*) > L(*w,* M(*w*)). Um *matching M* é instável se existe um par bloqueante em para M, e estável caso o contrário (Iwama).

A formalização acima é valida apenas para combinações perfeitas, onde os dois conjuntos de homens e mulheres tem o mesmo tamanho e as listas de preferências dos indivíduos são completas, ou seja contém todas as pessoas do sexo oposto. Gale e Shapley propuseram o conhecido algoritmo de Gale-Shapley, que roda em $O(n^2)$ e sempre encontra um *matching* estável (Gale e Shapley, 1962). Isto é uma prova construtiva que neste caso qualquer instância do problema admite pelo menos um *matching* estável.

4.3.2 EXTENSÕES

Relembrando que o problema de SM original, descrito anteriormente, requer que as listas de preferência contenham todos os membros do sexo oposto em uma ordem estrita. No entanto isto inconveniente em sistemas de grande escala. Logo podemos considerar duas relaxações do problema, listas incompletas e empates na lista.

4.3.2.1 LISTAS DE PREFERÊNCIA INCOMPLETAS

Nesta variação a lista de preferência de cada pessoa pode estar incompleta, em outras palavras algumas pessoas podem excluir algumas pessoas com as quais não querem ser casadas. Vamos chamar este problema de SMI. Se a lista *L* de uma pessoa *p* contém a pessoa *q,* então *q* é aceitável para *p*. Um *matching* M é um conjunto desconectado de pares (*m, w*) tal que *w* é aceitável para *m*. Já que a lista é incompleta o *matching* não é mais perfeito, ou seja, pode haver pessoas que não sejam casadas com alguém. Logo, temos que extender a definição de par de bloqueio. Para um *matching* M um par (*m, w*) é um par de bloqueio se as seguintes condições são satisfeitas: (i) M(*m*) != *w* mas *m* e *w* são aceitáveis um para o outro; (ii) L(*m, w*) > L(*m,* M(*m*)) ou *m* é solteiro em M; e (iii) L(*w, m*) > L(*w,* M(*w*)) ou *w* é solteiro em M (Iwama).

Uma propriedade importante do SMI é que podemos particionar o conjunto de homens(mulheres) em dois conjuntos; um conjunto em que todos os homens(mulheres) são casados em todos os *matchings* estáveis, e o outro em que todos são solteiros em todos os *matchings* estáveis. Isto imediatamente implica que todos os *matchings* possíveis para uma dada instância são do mesmo tamanho. Além disso o algoritmos de Galey-Shapley pode ser facilmente adaptado para encontrar um *matching* estável neste caso (Gale e Sotomayor, 1985)..

4.3.2.2 LISTAS DE PREFERÊNCIA COM EMPATES

A outra variação é permitir empates na lista de preferência, onde uma pessoa poder ter a mesma preferência por outras duas pessoas diferentes, ou seja, a função L vai retornar o mesmo valor para as duas pessoas. Esse problema será chamado de SMT(SM with Ties).

No SMT existem três noções de estabilidade: super-estabilidade, forte-estabilidade e fraca estabilidade. Basicamente elas diferem na definição de par de bloqueio. Para cada um das estabilidade um par de bloqueio é definido da seguinte maneira(Iwama):

Super-estabilidade - Um par (*m, w*) é um par de bloqueio se M(*m*) != *w*, L(*m, w*) >= L(*m,* M(*m*)) e L(*w, m*) >= L(*w,* M(*w*)).

Forte-estabilidade - Um par (*x, y*) é um par de bloqueio se M(*x*) != *y*, L(*x, y*) > L(*x,* M(*y*)) e L(*y, x*) >= L(*y,* M(*x*)). Onde x e y podem ser qualquer um dos sexos e x e y são sexos opostos.

Fraca-estabilidade - Um par (*x, y*) é um par de bloqueio se M(*x*) != *y*, L(*x, y*) > L(*x,* M(*y*)) e L(*y, x*) > L(*y,* M(*x*)).

Nota-se que um *matching* super-estável é fortemente-estável, e um *matching*

fortemente estável é fracamente-estável. Um *matching* fracamente estável sempre existe e pode ser encontrado em tempo polinomial, conforme resultados anteriores mostrados. Em contraste, existem instâncias que não contém um *matching* super-estável ou fortemente-estável. No entanto, existe um algoritmo que decide se existe um *matching* super-estável(fortemente-estáve respectivamente) e encontra um caso exista, que roda em tempo de O(n^2) (Irving, 1992).

4.3.2.3 LISTA INCOMPLETAS COM EMPATES

Esta variação permite tanto incompletude das lista quanto empates nas listas de preferência. Esta extensão será chamada de SMTI (SM with Ties and Incomplete Lists). Definição dos pares de bloqueio, e logo a estabilidade, podem ser naturalmente obtidos combinando os dois casos descritos anteriormente. Logo temos novamente os três níveis de estabilidade.

Para as estabilidades super e forte, resultados similares ao SMT se mantém, existe um algoritmo para cada caso que decide se a instância permite um *matching* estável, e encontra um se existir. Os algoritmos rodam em tempo de O(*a*) para super-estabilidade, e O(*na*) para forte-estabilidade, onde *a* é o tamanho total de todas as listas de preferência (que é igual a $2n^2$ se todas as listas de preferência estão completas)(Manlove, 1999). Também, sob super e forte estabilidades, todos os *matchings* estáveis de uma única instância tem o mesmo tamanho, como no SMI.

Para a estabilidade fraca, um *matching* estável existe para qualquer instância e pode ser encontrado em O(*a*), mas agora, uma instância pode ter *matchings* de diferentes tamanhos, e o problema de achar o maior (chamado de MAX SMTI) é NP-Hard (Iwama e Manlove, 1999). Também sabe-se que não existe algoritmo que em tempo polinomial aproxime em 21/19 o tamanho ótimo, a menos que P=NP (Halldórsson, 2007). Tiveram vários avanços nos algoritmos aproximados e o melhor atualmente consegue uma razão aproximação de 1.875 (Ikawama e Myiazaki, 2007).

4.3.2.4 MATCHINGS ESTÁVEIS ÓTIMOS

(Aqui talvez fosse bom aprofundar nas formas que optimizam a qualidade dos casamentos, e.g. todos conseguem o melhor par possível. O solução em sua forma mais simples é ótimo apenas para um dos sexos. No entanto é preciso analisar os primeiros resultados da implementação para saber se isso pode ser útil. Alguma referências: D. Gusfield, “Three fast algorithms for four problems in stable marriage,” SIAM J. Comput., Vol. 16, Issue 1, pp. 111–128, 1987, R. W. Irving, P. Leather and D. Gusfield, “An efficient algorithm for the “optimal” stable marriage,” Journal of the ACM, Vol. 34, pp. 532–543, 1987.)

4.3.3 OUTROS MODELOS

Aqui serão descritos alguns problemas derivados do SM.

Man-Exchange Stable Marriage. Para o SM clássico, foi uma nova definição de estabilidade, a estabilidade de troca-entre-homens foi definida. Esta estabilidade requer, em adição a estabilidade original, a propriedade que nenhum par de homens preferiria trocar os seus parceiros entre si. Irving provou que o problema de perguntar a existência dessa propriedade é NP-Completo(Irving, 2004).

One-Sided Preference Lists. Existems problemas de casamento onde apenas uma das partes(digamos, homens) tem uma lista de preferência sobre o outro. Aqui são introduzidos dois deles.

Um *matching de rank-máximo*, ou *rank-guloso*, é um *matching* que casa o máximo número possível de homens com as suas primeiras escolhas de parceiras, e sujeito a esta condição, o máximo número de homens com as suas segundas escolhas, e assim em diante. O problema de achar um rank-guloso foi estudo por Irving, que propos algortimos em tempo polinomial(Irving, 2003 e Irving,2004).

Many-to-many Stable Marriage. (Neste caso são considerados casamentos polygâmicos com uma cota. O problema é interessante mas não parece ser útil para o problema a ser resolvido neste trabalho. Pode ser expandido futuramente. Algumas referências: M. Baiou and M. Balinski, “Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry),”Discrete Applied Mathematics, Vol. 101, pp. 1–12, 2000., V. Bansal, A. Agrawal and V. S. Malhotra, “Stable marriages with multiple partners: efficient search for an optimal solution,” Proc. ICALP 2003, LNCS 2719, pp. 527–542, 2003.)

4.4 BIG DATA

"Big data is what happened when the cost of storing information became less than the cost of *making the decision* to throw it away." -George Dyson

Em uma econômia dinâmica e global, organizações começaram a se basear fortemente em percepções vindas de seus clientes, processos internos e negócios, a fim de descobrir novas oportunidades para crescimento. No processo de descobrir e determinar essas percepções, grandes conjuntos de dados são gerados que devem então ser gerenciados, análisados e manipulados de forma automática. Todos os dias são criados 2.5 quintilhões de dados, tanto que cerca de 90% dos dados no mundo hoje foram criados nos últimos dois anos (Falta referência segura). Esses dados vem de todos os lugares: sensores usados para obter informações do clima, posts em sites de midia social, imagens e vídeos digitais, registros de transações de compras, sinais GPS de ceulares, entre outros. Big data é o chavão utilizado para descrever esse volume enorme de dados, tanto estruturado quanto desestruturado, que é tão grande, e é gerado tão rápido, que é difícil processá-los com técnicas tradicionais de banco de dados e software.

Doug Laney define Big data em termos das três dimensões que apresentam desafios para serem gerenciados. As dimensões são as seguintes:

Velocidade: refere-se ao quão rápido dados são criados. E essa velocidade está aumentando considerávelmente. Olhando o infográfico abaixo é possível observar alguns exemplos da velocidade do dados. A cada minuto, 48 horas de vídeo são subidas para o Youtube, usuários do Twitter enviam 100.000 tweets e usuários do Intagram compartilham 3.600 fotos. Velocidade também está se tornando um aspecto chave no que diz respeito ao gerenciamento. Visitante do LinkedIn, por exemplo, não estão preparados para esperar mais do que pouco segundos para ver a lista possíveis amigos. Terabytes de dados precisam ser processados rapidamente.

Variedade: pois o resultado destas atividades não está relacionado a tipos específicos de dados. Estes vem de diversas fontes e em diversos formatos, podendo ser tanto estruturados. semi-estruturados ou desestruturados. Apesar de difícil de processar, essa variedade de dados pode fornecer grandes percepções sobre fatos não observáveis em um escopo restrito. No entanto isto requer novas de acessar e armazenar esses dados. Bancos de dados tradicionais não são mais adequados e por isso muitos frameworks e ferramentas estão surgindo no mercado. Exemplos incluem Hadoop, Cassandra e Spark.

Volume: é uma consequência natural da velocidade e da variedade. A agregação dos dados, em geral considerando todo um histórico, é necessário para que se possa fazer análise dos dados. Consequentemente é necessário um acúmulo dos dados, e dada a velocidade com que eles são gerados, altos volumes de armazenamento são fácilmente atingidos. Frequentemente é esperado que se possa definir um limite nominal, por exemplo 5 Terabyte ou 1 Petabyte, do qual pode-se começar a pensar em Big data. No entanto com o constante crescimento da geração de dados é díficil especificar e nominar este limite, se tal fosse feito, as definições mudariam a cada mês.

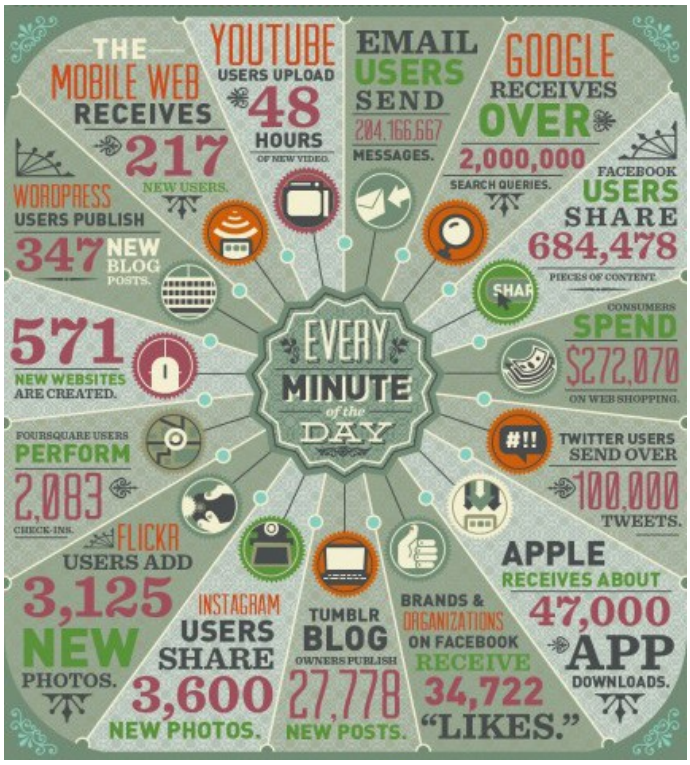

Ilustração 1: Exemplo de usos de aplicações de Big Data

4.5 MAP REDUCE

Nos ultimos cinco anos, com o crescimento da computação distribuída, autores e muitos outros programadores implementaram centenas de processos computacionais específicos para processar grandes quantidades de dados brutos, como registro de eventos e logs de requisição de dados. Com o intuito de derivar diversos tipos de dados, como índices invertidos, representações em grafos de documentos web, resumos de páginas pesquisadas, conjuntos de pesquisas mais frequentes e etc. Muitas das computações citadas são conceitualmente bem diretas. No entanto, os dados de entrada costumam ser grandes e os processos necessitam ser distribuídas através de centenas ou milhares de máquinas a fim de que estes terminem em um período de tempo aceitavél. Ao realizar essa distribuiçao, acabam surgindo problemas de como paralelizar os processos, distribuir os dados e gerenciar falhas , o que torna necessário uma grande quantidade de código complexo, obscurecendo a simplicidade das computações (Dean, 2008).

Em reação a esta complexidade adversa, pesquisadores do Google inventaram o paradigma MapReduce. Este modelo de programação visa o processamento e geração de grandes conjuntos de dados para certos tipos de computação que podem ser reduzidos a simples problemas de agregação. Neste modelo, o usuário especifíca uma função *map* que processa um par (chave, valor) para gerar um conjunto intermediário de pares (chave, valor), e uma função *reduce* que "funde" todos os valores intermediários associados com a mesma chave intermediária. Muitos dos problemas dos mundo real, inclusive os citados anteriormente, podem ser expressados neste modelo (Dean, 2008).

Programas escritos em MapReduce são automaticamente paralelizados e

executados em grandes clusteres de computadores. Todo o sistema de execução cuida dos detalhes de particionar os dados de entrada, agendar a execução dos programas através de um conjunto de máquinas, manejando falhas de máquinas e gerenciando a forma de comunicação entre as máquinas. Isto permite programadores sem muita experiência com sistemas paralelos e distríbuidos facilmente utilizarem os recursos de um grande sistema distribuído. Tudo isso combinada a um alto desempenho de execução.

Para entender o modelo considere o problema de contar palavras em um largo conjunto de documentos. O código abaixo exemplifica a solução deste problema em MapReduce:

```
map(String key, String value):
   // key: document name
   // value: document contents
   for each word w in value:
      EmitIntermediate(w, "1");

reduce(String key, Iterator values):
   // key: a word
   // values: a list of counts
   int result = 0;
   for each v in values:
      result += ParseInt(v);
   Emit(AsString(result));
```

A função map emite cada palavra mais um contador assicado as ocorrência(apenas 1 neste exemplo). A função reduce soma junto todos os contadores emitidos para uma palavra em particular.

4.6 EXPERIMENTO CEGO

Um experimento cego é um teste, ou experimento, no qual informações sobre o teste que podem levar a parcialidade dos resultados é ocultado do aplicador do teste, do participante, ou dos dois, até que o teste seja finalizado. Este tipo de experimento visa contornar o fato de que a parcialidade pode ser tanto intencional quanto inconsciente. Por exemplo, ao perguntar consumidores para compararem o gosto de diferentes marcas de um produto, as identidade dos produto devem ser ocultadas. Do contrário, consumidores geralmente irão tender a perferir a marca com a qual eles estão familiares. De forma similar, ao avaliar a efetividade de uma droga, tanto pacientes e doutores que administram a droga não devem as dosagens aplicadas em cada caso, para prevenir qualquer efeito placebo, parcialidade do observador, ou mesmo fraude (Friedman, 2010).

4.7 FUNDAMENTAÇÃO ESTATÍSTICA

4.7.1 SIGNIFICÂNCIA ESTATÍSTICA

A significância estatística de um resultado é uma medida estimada do grau em que este resultado é "verdadeiro" (no sentido de que seja realmente o que ocorre na população, ou seja no sentido de "representatividade da população"). Mais tecnicamente, o valor do *nível-p* representa um índice decrescente da confiabilidade de um resultado. Quanto mais alto o , menos se pode acreditar que a relação observada entre as variáveis na amostra é um indicador confiável da relação entre as respectivas variáveis na população. Especificamente, o *nível-p* representa a probabilidade de erro envolvida em aceitar o resultado observado como válido, isto é, como "representativo da população". Por exemplo, um *nível-p* de 0,05 (1/20) indica que há 5% de probabilidade de que a relação entre as variáveis, encontrada na amostra, seja um "acaso feliz". Em outras palavras, assumindo que não haja relação entre aquelas variáveis na população, e o experimento de interesse seja repetido várias vezes, poderia-se esperar que em aproximadamente 20 realizações do experimento haveria apenas uma em que a relação entre as variáveis em questão seria igual ou mais forte do que a que foi observada naquela amostra anterior. Em muitas áreas de pesquisa, o *nível-p* de 0,05 é costumeiramente tratado como um "limite aceitável" de erro (Hazewinkel, 2001).

4.7.2 INTERVALO DE CONFIANÇA

Um intervalo de confiança (IC) é um intervalo estimado de um parâmetro de interesse de uma população. Em vez de estimar o parâmetro por um único valor, é dado um intervalo de estimativas prováveis. O quanto estas estimativas são prováveis será determinado pelo coeficiente de confiança (1-α), para $\alpha \in (0, 1)$.

Intervalos de confiança são usados para indicar a confiabilidade de uma estimativa. Por exemplo, um IC pode ser usado para descrever o quanto os resultados de uma pesquisa são confiáveis. Sendo todas as estimativas iguais, uma pesquisa que resulte num IC pequeno é mais confiável do que uma que resulte num IC maior.

Se U e V são estatísticas (isto é, funções da amostra) cuja distribuição de probabilidade dependa do parâmetro θ, e

$$P(U < \theta < V|\theta) = 1 - \alpha*$$

então o intervalo aleatório (U,V) é um intervalo de confiança "100(1-α)% para θ". Portanto, podemos interpretar o intervalo de confiança como um intervalo que contém os valores "plausíveis" que o parâmetro pode assumir. Assim, a amplitude do intervalo está associada a incerteza que temos a respeito do parâmetro.

Considere X1,X2,...,Xn uma amostra aleatória retirada de uma população com distribuição f_θ que depende do parâmetro θ. Por exemplo, tomamos X1,X2,...,Xn uma amostra aleatória com distribuição normal com média μ desconhecida e desvio padrão conhecido σ=1. Para propormos um intervalo de confiança para o parâmetro θ, vamos introduzir o conceito de quantidade pivotal. Uma função Q da amostra (X1,X2,...,Xn) e do

parâmetro θ cuja distribuição de probabilidade não depende do parâmetro θ é denominada quantidade pivotal. Desta forma, dado o nível de confiança (1-α), tomamos

$$1 - \alpha = P[q_1 \leq Q(X_1, X_2, \ldots, X_n; \theta) \leq q_2]$$*

Se a quantidade pivotal Q for inversível, podemos resolver a inequação acima em relação a θ e obter um intervalo de confiança (Hazewinkel, 2001).

5 DESENVOLVIMENTO

5.1 RESOLUÇÃO DE ENTIDADES E SISTEMAS DE RECOMENDAÇÃO

Conforme mencionado anteriormente, sistemas de recomendação(SR) são de grande valia para lojas on-line (e-commerce). Dado seu valor e sua aplicabilidade, entende-se o por que é desejavel aprimorar sistemas de recomendação para que estes tenham:

Grande cobertura: Espera-se que um SR seja capaz de gerar produtos para o máximo de produtos possível e que o máximo de produtos possível esteja incluso em alguma recomendação.

Alta qualidade das recomendações: É desejável que as recomendações representem o comportamento esperado da melhor maneira possível. Uma recomendação de items similares para um devido item, deve conter no topo da recomendação os items que realmente são similares no mundo real e a ordem da lista também deve ser coerente com o mundo real

Diversidade nas recomendações: Uma recomendação deve recomendar de forma fornecer boas opções para o usuário e items extremamente semelhantes sobrecarregam uma recomendação e não adicionam opções. Por exemplo, exibir os diferentes tamanhos de uma mesma camisa em uma recomendação de alternativos, não agregam informação à recomendação.

Sistemas de Recomendação baseados em filtragem colaborativa costumam atender as necessidades acima descritas melhor do que os baseados em conteúdo. Eles conseguem capitar melhor as tendências e comportamentos, por se basearem nas interações dos usuários. Por isso que esta abordagem é a mais adotada para a construção de SR's e, logo , a que será adotada como o foco deste trabalho.

Também delimitaremos o foco deste trabalho a sistemas de recomendação com filtragem colaborativa baseados no modelo item-item. Este é o modelo mais usado na prática, em vez do modelo item-usuário, devido às lojas on-line terem uma grande quantidade de usuários, muitos deles anônimos, e os comportamentos de usuários serem mais dinâmicos do que de items. O modelo usuário-item é baseado numa matriz de usuário para usuário, que é extremamente grande e trabalhosa de computar.

5.1.1 LIMITAÇÕES E PROBLEMAS

Apesar da filtragem colaborativa render melhores resultados, ela ainda apresenta algumas limitações e problemas que afetam negativamente a cobertura, qualidade e diversidade das recomendações. O principal problema é o do *cold-start*, que afeta diretamente a cobertura e a qualidade das recomendações. Basicamente, como as recomendações são geradas através das interações entre usuários e produtos, com poucas interações não há informação suficiente para poder gerar recomendaçoes para certos produtos. Mesmo que a tendência seja de, a longo prazo, acumular essas informações, dependendo do caso, isto pode levar um tempo considerável.

No entanto, a tendência de que as recomendações melhorarem ao longo do tempo só é garantida em uma base fixa de items. Considerando o mundo real, a base de items de um e-commerce é extremamente dinâmica. Produtos são removidos e adicionados frequentemente por diversos motivos, como disponibilidade e promoções, e muitas vezes um produto já existente é recadastrado na base com um preço diferente ou em uma categoria diferente. O grande problema é que cada produto adicionado no catálogo inerentemente inicia em estado de *cold-start.* Além disso, todas as interações do produto anterior se tornaram inúteis. Assim produto recadastrado não tem recomendações e ainda pode acabar tendo suas interações divididas com o cadastro anterior caso este ainda permaneça ativo.

Outro caso comum em e-commerces que afeta diretamente a geração de recomendações são as variações de um mesmo produto. Por exemplo, um certo modelo de camiseta e suas variações de tamanho. Apesar de todos os tamanhos de referirem ao mesmo produto, cada um deles terá um único identificador. Isto faz com que as interações com o produtos fiquem dividas entre as suas variantes e geralmente não de forma uniforme. No caso, camisetas M são mais visitadas e compradas do que camisetas GG. Devido a SR's naturalmente não tratarem esses casos, ficam limitados a processar cada variação individualmente não levando em consideração que suas recomendações estão estritamente ligadas. Isto afeta principalmente a qualidade da recomendações e a diversidade. Estes produtos teram uma inércia maior para sair do estado de cold-start e caso saiam levarão mais tempo para que as recomendações atingam uma boa qualidade. Além do fato de poderem aparecer juntos nas listas de recomendação de outros produtos.

Como descrito, estas limitações e problemas tem impacto direto sobre as recomendações geradas e é de extremo interesse atacá-las para que o SR gere melhores resultados em um ambiente real de e-commerce.

5.1.2 UTILIZANDO RESOLUÇÃO DE ENTIDADES

É um tanto intuitivo que os problemas citados anterior podem ser resolvidos caso o sistema de recomendação tenha as informações sobre as duplicações e variações dos produtos do e-commerce. Para isso é necessário um processo de resolução de entidades. Com as duplicações e variações identificadas, é possível aprimorar o sistema de recomendação para que durante o processo de gerar as recomendações ele utilize essas informações e faça as devidas correções. No entanto, o problema de resolver entidades ainda é novo na área da computação e é considerávelmente difícil de se resolver e avaliar os resultados.

Logo, o intuito deste trabalho é propor uma forma inicial de como resolver a duplicações e variações de produtos, e avaliar os resultados da proposta de tanto quantitativamente quanto qualitativamente. Em seguida, propor uma solução de como utilizar essas informações da resolução no processo de geração de recomendações e avaliar os resultados quantitativamente e qualitativamente.

5.2 AMBIENTE E ESTRUTURA PARA O DESENVOLVIMENTO

Todos os processos e experimentos descritos neste trabalho foram desenvolvidos no ambiente de computação da Chaordic Systems. Devido ao foco deste trabalho em sistemas de recomendação, seria necessário grandes amostras de dados e uma estrutura computacional capaz de processá-los em tempo hábil. Sendo a Chaordic a maior empresa brasileira na área de sistemas de recomendação, esta oferece o ambiente perfeito para se obter os dados necessários e testar as soluções propostas.

Como base de informações foram utilizados os dados de eventos de compras de produtos de quatro lojas diferentes. Cada uma foi escolhida de forma a termos casos extremos e médios para experimentar. Para rodar os programas em MapReduce, utilizamos o framework da Chaordic Systems para execução de programas no EMR. O Elastic MapReduce, é o serviço de fornecimento de máquinas “on demand” que permite uma fácil maneira de requisitar clusters para executar programas em MapReduce

5.3 RESOLVENDO ENTIDADES

Para melhor definir o escopo do trabalho, serão definidos alguns níveis de resolução. As definições a seguir visam setar limites para melhor definir o trabalho a ser feito e possibilitar mensuração de avanços. Além de explicitar avanços que estejam fora do escopo de tempo e conhecimento deste trabalho. Os níveis são quatro:

Resolução exata: Neste nível a identificação deve ser feita de forma que não há sombra de dúvidas que as devidas entidades são as mesmas. Esta pode ser definida como uma função de igualdade, ou um conjunto delas, a qual deve retornar verdadeiro ou falso e somente retorna verdadeiro se os objetos realmente forem iguais. Esta forma pode ter uma abrangência pequena, no entanto não deve retornar Falso-Positivos.

Resolução direta: Aqui dois objetos são resolvidos comparando-os diretamente, ou seja, a função de similaridade foi chamada tendo os dois objetos como parâmetro e sua resolução não foi derivada de outras resoluções.

Resolução aproximada: Esta forma de resolução é baseada em uma função, ou conjunto delas, que retorna uma escala de similaridade entre dois objetos, por exemplo, utiliza um função $f : O \times P \rightarrow R$, onde O e P são conjuntos de objetos e R o conjunto dos reais. No entanto esta forma de resolução, pode retornar Falso-Positivos, logo é necessário uma forma de avaliar a qualidade das resoluções.

Resolução indireta: Neste nível usa-se informações sobre relações entre os objetos para inferir resoluções entre dois objetos, sem que estes necessitem serem comparados diretamente. Esta forma pode tirar vantagens de propriedades como a transitividade, para diminuir o escopo de objetos a ser resolvido.

5.3.1 PROCESSO PROPOSTO

Como esta é uma abordagem inicial ao problema, será proposto uma solução simples de resolução exata e direta. Esta solução utiliza o nome dos produtos para identificar variações e duplicados dentro de uma mesma loja. Dois produtos serão considerados iguais se a lista de palavras dos seus nomes normalizados forem iguais. Podemos intuitivamente dizer que dois produtos com nomes iguais, se ignorado variações na escrita, provavelmente são o mesmo produto.

Quanto a implementação do processo, é possível ver que este problema pode facilmente ser reduzido para o de agrupamento de chaves e valores. Os nomes dos produtos são as chaves, e os seus respectivos códigos identificadores, os valores. Logo, implementaremos o processo através do modelo de MapReduce. Isso permite executar o processo em um ambiente de computação distribuída, tornando-o um processo rápido de computar e escalável.

Abaixo está o pseudo-código do programa em MapReduce. O programa toma como entrada catálogos de produtos e gera como sáida o código do produto e uma lista de códigos dos produtos considerados iguais.

```
normalize(name):
	redecode_to_utf8(name)
	strip_html_and_convert_entities(s)
	normalize_case(s)
	normalize_diacritics(s)
	normalize_to_plain_ascii(s)
	normalize_to_alphanum_and_spaces(s)
	normalize_prepositions(s)
	normalize_whitespace(s)

map_products_to_name(data):
	product_id, name = data
	normalize(name)
	yield (name), (product_id)

reduce_products_grouping(name, grouped_product_ids):
	duplicated_products = set(grouped_product_ids)
	if length(all_products) > 1:
	for product in duplicated_products:
		product_set = set([product])
		duplicateds_list = list(duplicated_products.difference(product_set))
		output((product, duplicateds_list))
```

input -> map_products_to_name -> reduce_products_grouping -> output

5.3.2 RESULTADOS

Com a saída do processo anterior foi realizada uma análise quantitativa para saber o alcance que este teve e ter uma idéia da possível magnitude que processos mais elaborados podem ter. A análise foi realizado em cima de quatro bases de dados diferentes, cada uma com certas características de tamanho e diversidade. Cada uma das lojas abaixo foi escolhida visando representar os casos médios e extremos da forma mais fiél possível. Foram obtidos os seguintes resultados.

Base	Quantidade produtos	Quantidade duplicados	Quantidade grupos	% Duplicados	% Grupos	Média de produtos por grupo
S	2770716	886146	299510	31.98%	10.81%	2.95865246569397
N	281137	64605	21060	22.98%	7.49%	3.06766381766382
H	15510	13338	2941	86.00%	18.96%	4.53519211152669
M	120689	60548	29664	50.17%	24.58%	2.04112729234088

Tabela 1: Resultados absolutos e percentuais da resolução

Conforme a tabela acima, com a simples comparação exata de nomes foi possível resolver uma quantidade considerável de produtos. Em média pelo menos cerca de 50% dos produtos contém pelo menos um produto com o qual eles compartilham o nome, e logo, tem um forte nível de igualdade. A base H apresentou números bastante surpreendentes, no entanto isso não necessariamente é um bom indicativo. A alta taxa de duplicados e o baixo número de grupos pode indicar um agrupamento muito agressivo.

Em geral, esses números indicam que a resolução terá grandes impactos no processo de geração de recomendações. No entanto ainda é necessário uma análise qualitativa para averiguar a qualidade e precisão dos agrupamentos.

5.4 ACOPLANDO O PROCESSO AO SISTEMA DE RECOMENDAÇÃO

Tendo as informações dos produtos considerados iguais, é necessário agora acopla-la ao sistema de recomendação de forma utiliza-lá para aprimorar as recomendações. Aqui usaremos um sistema de recomendação de filtragem colaborativa baseado no modelo item-item implementado em MapReduce, descrito abaixo em pseudo-código. Este SR utiliza a co-ocorrência dos pares de produtos e a frequência individual dos produtos para calcular o nível de similaridade e relaciona items os agrupando por usuário.

A proposta seria utilizar essa informação para agregar os produtos iguais em um meta-produto mapeando os eventos de cada produto para eventos do meta-produto. Dessa maneira estaremos agregandos os eventos de produtos iguais em um único produto, contornando os problemas de divisão de eventos e de relacionamentos entre produtos iguais. Com isso espera-se um aumento na cobertura ao mesmo tempo que evitamos problemas de produtos iguais sendo recomendados um para o outro. Quanto à remapear os meta-produtos para os respectivos produtos, cada produto será emitido e terá os mesmos resultados. No caso, quando um meta-produto for a referência, este será separado nos respectivos produtos que o compõe e cada um destes produtos tera as mesmas recomendações. Quando um meta-produto estiver sendo recomendado, então este será dividido nos seus respectivos produtos e todos farão parte da lista. Dessa maneira, será possível utilizar diferentes filtros e políticas para decidir em tempo real quais destes produtos exibir.

O SR em MapReduce foi baseado nas implementações descritas em (Elsayed, 2008) e principalmente em (Yang, 2013). Abaixo está descrito em pseudo-código os dois principais passos que sofreram modificação para se acoplar a resolução e a posição de tais passos no *pipeline* de execução do programa.

```
reduce_products_to_user_and_frequency_resolution (product_tuple, values):
       product, event_type = product_tuple
       users_list = list()
       meta_prod_id = product
       # materialize and binarize the iterator
       for input_type, data in values:
              if input_type == 'resolutioninfo':
                     # Agregates all the duplicates, if any.
                     meta_prod_id = list(data)
                     meta_prod_id.append(product)
                     meta_prod_id.sort()
              else:
                     # This case the value is a user_id
                     users_list.append(data)
                     yield (meta_prod_id, event_type),
                            (users_list)


def yieldProductSimilarityTuples(key, sim_p1_p2):
        p1, p2 = key

        # We trown the Recs for every product in the metaProduct list. Hence
        # every p in metaProduct is gonna have the same recommendation. However
        # if p2 is a metaproduct, then every p in p2 is gonna be in p1's list.
        if isinstance(p1, basestring):
            p1 = [p1]
        if isinstance(p2, basestring):
            p2 = [p2]

        for p_1 in p1:
            for p_2 in p2:
                yield((p_1), (p_2,sim_p1_p2))


map_calculate_similarity(product_pair_tuple, frequency_data):
       p1, p2 = product_pair_tuple
       f1, f2, amf = frequency_data
       similarity_value_p1_p2 = calculate_similarity(f2, f1, amf)
       similarity_value_p2_p1 = calculate_similarity(f1, f2, amf)
       for key, value in yieldProductSimilarityTuples((p1, p2),
                                                      (similarity_value_p1_p2)):
              yield key, value
       for key, value in yieldProductSimilarityTuples((p2, p1),
                                                      (similarity_value_p2_p1)):
              yield key, value



input->map_to_product_event_type
->reduce_products_to_user_and_frequency_resolution
->reduce_products_to_users_and_frequency
->reduce_users_and_frequency_to_product_pairs->reduce_product_pairs_to_amf
->map_calculate_similarity->emit_top_n->output
```

5.4.1 RESULTADOS

Após a implementação do processo acima descrito, foi calculado os efeito que este teve sobre a cobertura de recomendaçãoes sobre o conjunto de produtos. Ou seja, foi calculado a quantidade de produtos com recomendação, antes e depois do processo utilizando resolução de entidades. Abaixo estão os resultados para a base de produtos de cada loja, onde é mostrado também a cobertura de recomendações com no mínimo n produtos na recomendação, com n indo 1 até 8. Nas legendas, C/ER se refere ao processo de geração de recomendações com resolução de entidades, e S/ER se refere ao processo sem o uso de resolução de entidades.

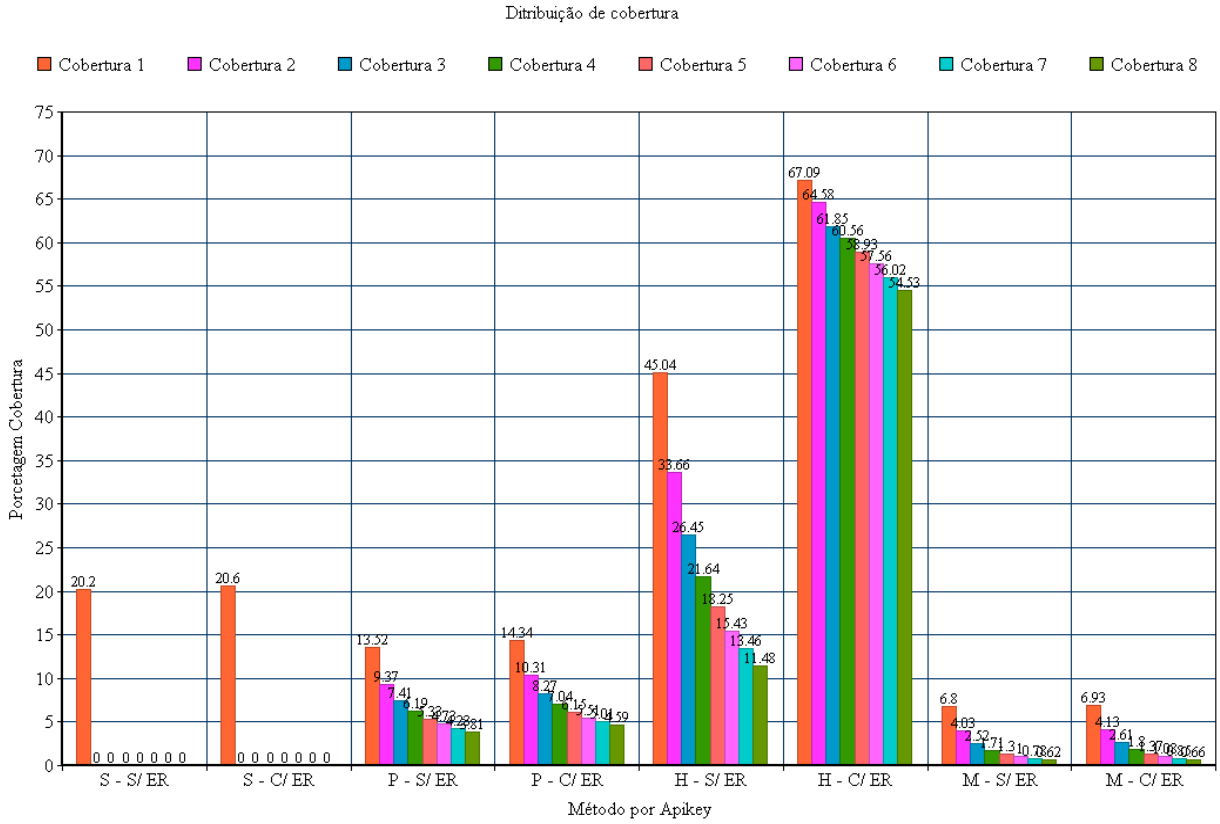

Ilustração 2: Gráfico com distribuição do tamanho das listas de recomendações

Através do gráfico é possível ver que a resolução não teve grande influência na cobertura de recomendações, com incrementos na faixa dos 5% a 10%. Somente um caso, da base H, houve um aumento grande, com cerca de 50% de incremento. Neste caso também é possível observar a atenuação do decaimento da cobertura com o aumento da extensão mínima da lista. Essa atenuação também ocorre nas outras bases, no entanto, de forma mais sútil. Isso mostra que a resolução também ajudou a estender a quantidade de produtos nas recomendações.

No caso da base S, não foi possível fazer está análise por extensão das listas devido a imensa quantidade de produtos que esta possui, passando da ordem dos dois milhões. Isto fazia com que memória das máquinas, em que os foram rodados os programas pra obtenção dos resultados, estourasse. No entanto isso não interferiu na

análise dos resultados.

No entanto bons números na análise quantitativa, neste caso, não implicam em bons resultados qualitativos, e vale o mesmo para o oposto. Os números extravagantes obtidos na base H podem na verdade podem implicar em um excesso de produtos agrupados, o que pode não ter boas implicações na qualidade dos resultados. Do mesmo modo, os números não tão significativos podem não espressar as melhorias na qualidade das recomendações. Por isso, na próxima parte do trabalho, faremos uma análise qualitativa dos resultados.

5.5 AVALIAÇÃO QUALITATIVA

De forma quantitativa foi possível atingir bons números na resolução, mas foram menos significativos no aprimoramento das recomendações. Os dados quantitativos foram relativamente fáceis de desenvolver e processar, e comprovaram a viabilidade dos métodos. No entanto é necessário dar mais um passo adiante na análise e verificar também a eficácia do método. Em outras palavras, é preciso avaliar a qualidade dos resultados obtidos. No caso, temos que avaliar os seguintes pontos em cada um dos processos:

Resolução: Se cada um dos produtos agrupados realmente são iguais, verificando a quantidade de casos Positivos e Falso-Positivos. Assim teremos uma medida de precisão da resolução.

Recomendações: Comparar cada uma das recomendações geradas com resolução com as respectivas recomendações geradas sem a resolução. Com isso teremos um nível de qualidade comparativa entre os tipos de recomendação, nos informando o quão melhor ficaram as recomendações com a resolução.

Além dos números quanto a qualidade da solução, esta avaliação também pode fornecer informações sobre o casos extremos, tanto de boa qualidade quanto de má qualidade, permitindo que se depure os motivos de cada caso acontecer. Isso pode ajudar na identificação de falhas no processo e a formular idéias para aprimorá-lo.

Entretanto, é intuitivo que não há como fazer este tipo de análise diretamente em cima dos dados, de forma manual. A quantidade de informações geradas é extremamente grande e difícil de ser diretamente avaliada e interpretada, sem contar problemas como a parcialidade na avaliação. Logo é necessário uma maneira viável, imparcial e *user-friendly* de avaliar os resultados, que seja também acessível a mais pessoas. A solução para tais problemas seria desenvolver uma ferramenta web de teste cego.

Com a ferramenta web de teste cego, atacaremos os seguintes problemas:

Grandeza dos resultados: Com a ferramenta podemos usar técnicas de probabilidade e estatística para diminuir a quantidade de dados a serem avaliados. Utilizando técnicas de amostragem, nível de significância e intervalo de confiança, podemos selecionar um conjunto representativo dos resultados, tornando a avaliação viável.

Parcialidade: O teste cego permite esconder informações, para o avalidor e o aplicador, que possam influenciar os resultados de forma tendenciosa. No caso de

comparação entre duas recomendações, ao visuáliza-las o avaliador não saberá qual se refere a cada método.

Diferentes avaliadores: A ferramenta sendo web, poderemos disponibilizá-la para várias pessoas poderem acessá-la. Cada indíviduo tem uma opnião diferente sobre qualidade e mesmo com amostragem a quantidade de resultados a serem avaliados ainda é grande para uma única pessoas. Com a ferramenta, as avaliações serão mais heterogeneas e poderemos avaliar as amostras mais rapidamente.

Armazenamento e análise: Podemos utilizar um banco de dados para salvar as informações obtidas através da ferramenta, permitindo uma persistência do dados e facilitando a extração destes. Além de que muitas das análise podem ser realizadas simplesmente com uma pesquisa SQL.

5.5.1 FERRAMENTA DE TESTE CEGO

Dado os problemas e necessidades para a realização da análise qualitativa, foi desenvolvido uma primeira versão de uma ferramenta de teste cego. Apesar do próprio desenvolvimento desta ferramenta não estar incluso no escopo deste trabalho, esta é de extrema importância para a validação e, logo, a conclusão deste. O desenvolvimento de uma ferramenta desta complexidade, poderia por si só ser tema de um trabalho, por isso a ferramenta aqui desenvolvida é de certa forma um protótipo, com apenas o necessário para se obter e armazenar as informações para análise. Logo, aspectos como design, usabilidade, performance e qualidade do código foram dados baixa importância.

Para o desenvolvimento da ferramenta, foi utilizado como base uma ferramenta usada para depuração de recomendações, usada pelo time de cientistas da Chaordic. Esta ferramenta já tinha pronta algumas funções para requisição, validação e exibição de recomendações em uma página web. No entanto ainda foi necessário desenvolver a base de dados da ferramenta, as API de requisação e armazenamento das informações e o layout da ferramenta. Devido a complexidade da ferramenta, não serão descritos aqui os detalhes do processo de desenvolvimento. Apenas os aspectos mais relevantes quanto as partes da ferramenta, e as principais caracteristicas e funcionalidades, serão descritos..

Abaixo alguns do pontos importantes e técnologias usadas de cada um dos módulos da ferramenta:

Base de dados: Foi utilizado o banco de dados relacional MySQL. No banco ficam armazenados os dados e relacionamentos dos experimentos cadastrados, produtos candidatos, algoritmos referencia das recomendações e resultados.

API's: Para requisitar e armazenar informações, tanto do banco de dados quanto da plataforma de recomendações, foi desenvolvido funções em NodeJS. O controle de amostras restantes e escolha aleatória fica neste módulo também.

Layout e Interface: Para o layout da página foi utilizado HTM, estilos CSS e a biblioteca bootstrap. Para o desenvolvimento das interações da página, foi utilizado a biblioteca JQuery.

Algumas das funcionalidades da ferramenta:

Randomização das recomendações: As recomendações exibidas trocam de lugar aleatoriamente para evitar que o avaliador seja tendencioso.

Seleção mutuamente exclusiva: O avaliador consegue selecionar apenas uma das opções. Isto facilita a análise dos dados permitindo a separação clara entre pontuações positivas, negativas e indesições.

Comparação entre N opções: É possível comparar várias recomendações entre si, tendo uma análise de qualidade comparativa, ou avaliar apenas uma recomendação, obtendo assim uma análise de qualidade da própria recomendação. No segundo caso, o usuário ve apenas uma recomendação e marca se é boa ou não.

Amostras em caixa com e sem reposição: Todas as amostras são retornadas pela API como se tivessem sido tiradas aleatóriamente de uma caixa sem reposição. Isso visa priorizar a cobertura de amostras diferentes avaliadas. Após todas as amostras terem sido avaliadas, a API passa a funcionar como uma caixa aleatória com reposição, visando a diversidade das avaliações.

Suporte a diferentes testes e avaliadores: Pode-se cadastrar vários testes diferentes na ferramenta, que podem ou não compartilhar algoritmos e amostras. Também pode-se autenticar os usuários que avaliaram cada resultado.

5.5.2 RESULTADOS DO TESTE CEGO

Para o teste cego foram selecionadasduas das bases de dados analisadas na análise quantitativa, a base da loja P e da loja H. Foram selecionadas estas devido à base H ser um exemplo de caso com grande impacto e a base P um exemplo de caso médio, com pouca influência na cobertura. Em seguida foi especificado o nível de confiança e o intervalo desejado. Devido ao teste não precisar de alta precisão, por ter fins qualitativos e o conceito de qualidade ser um tanto subjetivo, optou-se por um nível de confiança de 95% com 5% de intervalo de erro. Com isso, utilizando fórmulas baseadas em probabilidade e estatística, foi calculado um tamanho de amostra mínimo de cerca de 350 produtos para cada base de dados, totalizando 700 produtos para serem avaliados.

Devido a grande a grande quantidade de produtos foi utlizado um algoritmo baseado em *reservoir sampling* para a escolha aleatória das amostras. Tal algoritmo se encontra no Apêndice.

O primeiro teste realizado foi referente a qualidade da resolução, no qual o avaliador visualiza um produto e os produtos que foram agrupados com ele e entre si. O avaliador caso julgue o agrupamento consistente, marca a caixa de seleçao e clica em *Agreed*, assim dando um pontuação 1 para tal produto. Caso contrário, o avaliador não marca nada e clica em *Agreed*, assim dando uma pontuação 0 para tal produto. O avaliador também pode verificar informações adicionais de cada produto para auxiliar na escolha. Após a avaliação, realizada por nove pessoas e com um total de 710 avaliações, foram obtidos os seguintes resultados:

Teste	Pontuação	Contagem
product_clustering_by_name	0	90
product_clustering_by_name	1	620

Tabela 2: Resultados gerais da resolução

Teste	Base	Pontuação	Contagem
product_clustering_by_name	H	0	75
product_clustering_by_name	H	1	281
product_clustering_by_name	P	0	15
product_clustering_by_name	P	1	339

Tabela 3: Resultados por base da resolução

Os resultados indicam que no caso geral a resolução foi considerada boa em 87% das vezes, com uma margem de erro de + - 5%, e ruim ou não avaliável em 23% das vezes. Nos casos específicos, para a base H, os resultados positivos caem para 78%, já para a base P os positivos subiram para 97%. Os dois casos considerando, obviamente, os 5% de margem de erro.

No segundo teste cego, foi realizado uma comparação entre as recomendações de complementariedade geradas com e sem o agrupamento de produtos. Os avaliadores tinham que comparar as duas recomendações e escolher a melhor, levando em conta aspectos como o tamanho da lista, a diversidade de produtos e o potencial para melhoramentos. Escolhido a melhor recomendação, sua caixa era marcada, atribuindo a pontuação 1 para o algoritmo referente a recomendação marcada, e 0 para o outro algoritmo, e vice e versa. Caso estivesse em dúvida, o avaliador não marcaria nenhuma das duas caixas, atribuindo a pontuação 0 para os dois algoritmos. Neste teste, os avaliadores foram orientados a avaliar de forma comparativa as recomendações, não as suas respectivas qualidades. No caso, se as duas recomendações fossem ruins, a menos ruim ganharia a comparação e deveria ser escolhida.

O tamanho da amostragem foi o mesmo do teste anterior e as amostras foram similarmente obtidas por *reservoir sampling,* com a restrição da recomendação ter sido afetada pela resolução. Os seguintes resultados foram obtidos:

Algoritmo	Pontuação	Contagem
complementariedade_com_resolução_de_entidades	0	183
complementariedade_com_resolução_de_entidades	1	181
complementariedade	0	341
complementariedade	1	23

Tabela 4: Resultados gerais das recomendações

Conforme a tabela acima, é possível observar que em 56% da avaliações não houve diferença entre as duas recomendações ou a diferença não era perceptível. Nos casos em que houve alguma escolha de preferência, em 88% dos casos as recomendações com resolução foram avaliadas como melhores. Vale mencionar que as porcentagens acima tem uma confiança de 95% e 5% de margem de erro.

5.5.3 ANALISE DO RESULTADOS

Os resultados obtidos nos testes foram bastante positivos quanto ao uso da resolução de entidades. Tanto a resolução quanto as recomendações geradas tiveram um bom percentual de aceitação nos testes. De acordo com a Tabela 3, apesar da base H ter tido um percentual de negativos um pouco alto para uma resolução exata, cujo motivo é explicado a seguir, isto não impactou as recomendações negativamente de forma proporcional, tendo apenas uma pequena porcentagem com avaliação negativa. Já na base P, pode-se ver que uma boa porcentagem é representada por não-mudou ou indecisos, o que implica que a resolução teve poucos impactos, o que era esperado dado o sutil aumento na cobertura. No entanto, todos os casos em que houve uma decisão, as recomendações com resolução foram escolhidas quase 90% das vezes. Isto indica que os poucos resultados implicaram em boa qualidade das resoluções, conforme Tabela 4.

Um dos problemas que pôde ser identificado, que ocorreu especialmente com os produtos da base H, foi o de produtos com o mesmo nome que na verdade não deveriam ter sido agrupados. Como exemplo, houveram muitos produtos com o nome “Vestido” mas que alguns eram Feminino Adulto e outros Infantis. No caso, apesar se compartilharem o mesmo nome, não é benéfico para as recomendações eles serem agrupados já que cada um interage com tipos diferentes de produtos. Isto esta associado ao fato do nome do produto, neste caso, fornecer muita pouca informação para identificá-lo. Dado ao perfil dos produtos da loja H, com muito produtos com nomes poucos específicos, esse problema aconteceu com certa frequência, o que ocasionou nos números exorbitantes na análise quantitativa.

Um ponto da resolução que não possível avaliar é a revocação (recall). Foi possível estimar a quantidade de Falsos-Positivos através do teste cego, no entanto não foi possível estimar quanto do conjunto dos positivos foi identificado como positivo, basicamente pelo fato não se saber previamente o tamanho deste conjunto. Isto é necessário principalmente para avaliar a abrangência do método de resolução e ser possível trabalhar para aprimorar o método visando expandir-la. A estimação da revocação para para resolução de entidades ainda é nova e, por enquanto, depende basicamente de trabalho humano para a estimação.

6 CONCLUSÃO

Inicialmente, quando o tema deste trabalho foi proposto, a ideia era resolver entidades entre duas lojas diferentes a fim de aprimorar o alcance e a qualidade das recomendações de lojas com poucos eventos e com o catálogo muito dinâmico. Para tal, primeiramente seriam agrupados produtos similares dentro de uma mesma loja e então utilizar o problema do casamento estável para identificar o melhor casamento de grupos entre as diferentes lojas. No entanto, conforme o trabalho era desenvolvido, mostrou-se ser de grande dificuldade agrupar os produtos dentro de uma mesma loja e utilizar essa informação de forma correta. Avaliar os resultados, de forma quantitativa e qualitativa, também adicionou dificuldade ao trabalho. Com isso foi necessário vencer uma grande inércia, tendo que descobrir soluções e desenvolver ferramentas para o trabalho ter a base necessária para poder começar a abordar a resolução entre diferentes lojas. Com isso o trabalhou acabou se limitando a uma abordagem inicial para utilizar a resolução de entidades para produtos de uma mesma loja.

Os resultados obtidos na resolução foram satisfatórios. No entanto, parece claro que é necessário utilizar técnicas mais avançadas para se agrupar os produtos. Utilizar os nomes dos produtos não se mostrou tão preciso e abrangente quanto previsto. Uma simples mudança, como na ordem de algum termo, já foge ao alcance do método. Métodos mais avançados que utilizem tanto a entropia das informações quanto o nível de similaridade entre entidades, seria necessário para ampliar a precisão e a capacidade de revocação da resolução. É importante considerar também a importância desta necessitar ter alta precisão. Agregações incorretas podem gerar maus resultados e acabar tendo uma grande influa negativa no perfil de interações da loja, podendo até gerar recomendações que sejam ofensivas para os usuários.

Quanto a geração das recomendações, em geral os resultados foram discretos, no entanto todos foram positivos. A agregação dos produtos realmente recuperou informações perdidas e ampliou a interação entre produtos diferentes. Mesmo nos casos em que a resolução gerou uma quantidade considerável de falsos positivos, as recomendações ficaram mais completas e diversas, com poucos casos onde as recomendações foram negativamente afetadas. No entanto, esses resultados não foram proporcionais à computação necessária para gerá-los. Utilizar o conceito de meta-produto pode não ser a melhor maneira de utilizar as informações de resolução para aprimorar as recomendações. Muitas vezes dois produtos são extremamente parecidos, mas entretanto interagem com perfis de usuários diferentes, e agregá-los pode fazer com que essas identidades sejam perdidas. Esse tipo de problema se torna ainda mais evidente, e mais grave, se expandirmos o uso da resolução para entre diferentes lojas. É necessário pesquisar uma maneira de utilizar a resolução de maneira que ela gere mais impactos positivos e seja capaz de preservar comportamentos particulares que são essencias para uma boa recomendação.

Um dos principais marcos deste trabalho foi o uso do teste cego para complementar a avaliação dos resultados. Este permitiu uma análise imparcial e colaborativa dos dados, ajudando assim na depuração e na identificação de problemas. O uso de amostragem e níveis e intervalos de confiança deram significância estatística aos resultados, agregando confiabilidade sem a necessidade de verificações exaustivas. Além disso, o método tem potencial para ser usado não somente para avalição, mas também para treinamento de processos inteligentes.

O trabalho realizado ajudou a pavimentar o caminho para o desenvolvimento e a avaliação de utilizar resolução de entidades pra a geração de recomendações em e-commerces, e foi de grande valia para mostrar a viabilidade e o potencial de se utilizar tais métodos. Apesar de não ter sido possível atingir todos os objetivos planejados, uma

forte base foi estruturada para que a pesquisa aplicação continue, para que tais objetivos sejam alcançados e novos obejtivos sejam estabelecidos.

6.1 TRABALHOS FUTUROS

Como mencionado anteriormente, este trabalho começou com a ideia de resolver produtos entre diferentes lojas. O próximo passo lógico seria perseguir este objetivo. Através de formas de resolução aproximada, pode ser possível identificar grupos de produtos muito similares dentro de cada uma das lojas. Com esse grupos formados, é possível casar os grupos entre as lojas utiizando algoritmos para formação de casamento estável. Com este casamento definido, será possível recomendar produtos de diferentes lojas para um dado usuário, visando não apenas personalizar ao máximo a compra, mas também permitindo ao usuário otimizar seus gastos.

Para esta resolução aproximada, existem diferentes técnicas que podem ser usadas. Pode-se usar LSH, que são hash tables onde valores muito paracidos são mapeados para uma mesma chave, ou pHash, onde valores próximos vão ter hash com alta semelhança. É possível também utilizar medidas de similaridade, como a de Levenshtein, combinadas com técnicas de Aprendizado de Máquina para aprender os limiares adequados. Isso é possível dado a grande quantidade de dados disponível em e-commerces e a possibilidade de utilizar feedbacks de usuários, explicítos ou implícitos, para supervisionar o aprendizado.

7 REFERÊNCIAS

Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35

Schwartz, B.: The Paradox of Choice. ECCO, New York (2004).

P. Resnick and H. R. Varian. Recommender systems. Special issue of Communications of the ACM, pages 56–58, March 1997.

Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM 35(12), 61–70 (1992).

Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems An Introduction. Cambridge University Press (2010).

Hill, W., L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In Proceedings of CHI'95.

Resnick, P., N. Iakovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In Proceedings of the 1994 Computer Supported Cooperative Work Conference, 1994.

Adomavicius, G. Towards the Next Generation of Recommender Systems: A survey of the State-of-art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, Volume 17 Issue 6, June 2005. Página 734-749.

Deshpande, M. and G. Karypis. Item-Based Top-N Recommendation Algorithms. ACM Transactions on Information Systems, 22(1):143-177, 2004.

Segaran,T. Programming Collective inteligence, 2007.

Schafer, J.B., Konstan, J.A., Riedl, J.: E-commerce recommendation applications. Data Mining and Knowledge Discovery 5(1/2), 115–153 (2001).

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, David M. Pennock (2002). Methods and Metrics for Cold-Start Recommendations. Proceedings of the 25th Annual International ACMSIGIR Conference on Research and Development in *Information Retrieval (SIGIR 2002)*. New York City, New York: ACM. pp. 253–260.

D. Gale and L. S. Shapley. College admissions and the stability of marriage, Amer. Math. Monthly, Vol.69, pp. 9–15, 1962.

National Resident Matching Program, http://www.nrmp.org/

Nobel Prize, 2012. http://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2012/popular-economicsciences2012.pdf

Iwama, K. and Miyazki, S. A survey on the Stable Marriage Problem. International Conference on Informatics Education and Research for Knowledge-Circulating Society

D. Gale and M. Sotomayor. Some remarks on the stable matching problem. Discrete Applied Mathematics, Vol.11, pp. 223–232, 1985.

Irving, R. W. Stable marriage and indifference. Discrete Applied Mathematics, Vol.48, pp. 261–272, 1994.

D. F. Manlove, “Stable marriage with ties and unacceptable partners,” University of Glasgow, computing Science Department Research Report, TR-1999-29, 1999

K. Iwama, D. F. Manlove, S. Miyazaki, and Y. Morita. Stable marriage with incomplete lists and ties. Proc. ICALP 99, LNCS 1644, pp. 443–452, 1999.

M. M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa, “Improved approximation results of the stable marriage problem,” ACM Transactions on Algorithms, Vol. 3, Issue 3, Article No. 30, 2007.

K. Iwama, S. Miyazaki, N. Yamauchi, “A 1.875-approximation algorithm for the stable marriage problem,” Proc. SODA 2007, pp. 288–297, 2007.

R. W. Irving, “Man-exchange stable marriage,” University of Glasgow, Computing Science Department Research Report, TR-2004-177, 2004.

R. W. Irving, “Greedy matchings,” University of Glasgow, Computing Science Department Research Report, TR-2003-136, 2003.

R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch, “Rank-maximal matchings,” Proc. SODA 2004, pp. 68–75, 2004.

Hernandez, M. A. & Stolfo, S. J. (1998). Real-world Data is Dirty: Data Cleansing and the Merge/Purge Problem. In Journal of Data Mining and Knowledge Discovery.

Benjelloun, O., Garcia-Molina, H., Jonas, J., Su, Q. &Widom, J. (2005). Swoosh: A Generic Approach to Entity Resolution. Technical Report, Stanford University.

Bhattacharya, I. & Getoor, L. (2006). A Latent Dirichlet Allocation Model for Entity Resolution. 6th SIAM Conference on Data Mining, Bethesda, MD.

Bhattacharya I. & Getoor L. (2004). Deduplication and Group Detection Using Links. KDD Workshop on Link Analysis and Group Detection, Aug. 2004, Seattle.

Bilenko, M. & Raymond J. Mooney, R. J. (2003). Adaptive Duplicate Detection Using Learnable String Similarity Measures. In the Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge.

Muller, H. & Freytag, J.-C (2003). Problems, Methods and Challenges in Comprehensive Data Cleansing. Technical Report (Humboldt-Universitt zu Berlin, Institut fr Informatik Publication No. HUB-IB-164).

Jeffrey Dean and Sanjay Ghemawat (2008). MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113.

Friedman, L.M., Furberg, C.D., DeMets, D.L. (2010). Fundamentals of Clinical Trials. New York: Springer, pp. 119-132.

Hazewinkel, M., ed. (2001), "Confidence estimation", Encyclopedia of Mathematics, Springer, ISBN 978-1-55608-010-4

Yang, B., Myung, J., Lee, S., and Lee, D. (2013). A MapReduce-based filtering algorithm for vector similarity join. In *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication* (ICUIMC '13). ACM, New York, NY, USA, , Article 71 , 5 pages.

Elsayed, T., Lin, J., Oard, D. (2008). Pairwise document similarity in large collections with MapReduce. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers* (HLT-Short '08). Association for Computational Linguistics, Stroudsburg, PA, USA, 265-268.