LUCAS FERNANDES BRUNIALTI

Biclusterização aplicada em Sistemas de Recomendação baseados em Conteúdo Textual

São Paulo

2014

LUCAS FERNANDES BRUNIALTI

Biclusterização aplicada em Sistemas de Recomendação baseados em Conteúdo Textual

Texto de Exame de Qualificação apresentado à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo como parte dos requisitos para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Orientador: Profa. Dra. Sarajane Marques Peres

Co-Orientador: Prof. Dr. Valdinei Freire da Silva

São Paulo

2014

Autorização para Reprodução

Ficha catalográfica

Folha de Aprovação

Texto de Exame de Qualificação de Mestrado sob o título *"Biclusterização aplicada em Sistemas de Recomendação baseados em Conteúdo Textual"*, apresentado por Lucas Fernandes Brunialti e aprovado em ___ de _____ de _____, em São Paulo, Estado de São Paulo, pela comissão examinadora constituída pelos doutores:

Prof. Dr. _____
Presidente
Instituição: _____

Prof. Dr. _____
Instituição: _____

Prof. Dr. _____
Instituição: _____

Resumo

BRUNIALTI, Lucas Fernandes. **Biclusterização aplicada em Sistemas de Recomendação baseados em Conteúdo Textual**. 2015. NúmeroDePáginas f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, Ano2.

Biclusterização representa uma estratégia de análise de dados com potencial para encontrar grupos de objetos similares considerando a correlação parcial existente entre os atributos descritivos dos mesmos. Esse potencial pode ser particularmente útil para Sistemas de Recomendação baseados em conteúdo, nos quais se faz necessária a sugestão de itens úteis, porém diversificados, para um usuário. Esta necessidade configura-se como um problema de *serendipidade*, uma das propriedades de Sistemas de Recomendação. O objetivo deste trabalho é analisar a aderência dos resultados provenientes de algoritmos de biclusterização, aplicados aos itens a serem recomendados, à meta de otimização da *serendipidade*. Para alcançar tal objetivo, este trabalho propõe um estudo da aplicação de algoritmos clássicos de biclusterização à conteúdo textual referente ao escopo de um Sistema de Recomendação de notícias. A hipótese defendida é que devido à análise de correlações parciais dos atributos descritivos dos dados, seria possível encontrar notícias com similaridades parciais entre si devido às particularidades presentes nas mesmas. Desta forma, uma mesma notícia que pertence a mais de um contexto poderia ser recomendada a usuários que estariam, à princípio, interessandos em assuntos diferentes. Como forma de avaliação dos resultados obtidos, é proposta uma análise comparativa entre os resultados da recomendação obtida via biclusterização e os resultados de recomendação obtida via filtro colaborativo, onde o problema da *serendipidade* é reconhecidamente bem resolvido.

Palavras-chave: Biclusterização. Sistemas de Recomendação. Recomendação de Notícias. Mineração de Textos.

Abstract

BRUNIALTI, Lucas Fernandes. **Work title**. 2015. NumberOfPages p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, Year2.

Write here the English version of your "Resumo "...

Keywords: Keyword1. Keyword2. Keyword3. Etc.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Dada a quantidade e dinamicidade de informação presente na web, as pessoas, usuárias da web, se vêem diante de uma dificuldade de achar o que querem e quando querem. Quando usuários da web querem comprar um produto ou ler sobre uma notícia em específica, não conseguem localizar o que querem, ou até mesmo, não sabem nem o que querem. Ricci, Rokach e Shapira (2011) argumenta que muitos usuários não tem conhecimento suficiente para saber o que querem ou não, consumir da web. Com o objetivo de solucionar esse problema, surgem os Sistemas de Recomendação (SRs), que com base nas preferências do usuário, buscam predizer ou sugerir itens (produtos, artigos, músicas e etc), com potencial de serem de grande valia para os usuários.

No domínio de notícias, o mesmo problema acontece. Para isso, portais de notícias da web necessitam de oferecer conteúdo personalizado. No entanto, aprender as preferências dos usuários e ao mesmo tempo oferecer conteúdo útil, não é uma tarefa trivial, principalmente no contexto de notícias, em que as preferências dos usuários mudam com muita frequência (LI et al., 2011).

Um SR de notícias simples e hipotético poderia ter a seguinte estratégia para recomendação: dado o histórico de notícias visitadas pelo usuário, as notícias recomendadas seriam as mais semelhantes ao seu histórico. Esse sistema seria especializado em oferecer notícias que o usuário já esta acostumado à ler, então, há uma grande probabilidade de não existir nada de novo para o usuário nas recomendações oferecidas. Por exemplo, se dois usuários estão navegando nesse sistema, um com padrão de leitura de notícias sobre esporte e outro de notícias sobre música, o sistema hipotético recomendaria para o primeiro usuário apenas notícias sobre esporte, e para o segundo, apenas notícias sobre música. O problema esta em o SR não apresentar nenhuma novidade nas recomendações, nenhuma recomendação que, provavelmente, será útil para o usuário, pois os mesmos já sabem buscar as notícias que costumam ler. Sendo assim, o SR precisa ser capaz de apresentar recomendações com *serendipidade*, ou seja, que sejam diversas o bastante, e recomendações úteis para os usuários. Uma melhoria para o sistema hipotético (Figura 1) seria o SR perceber de alguma forma que as notícias sobre esporte tem intersecção com

as notícias sobre música, e recomendar algumas notícias sobre música para o usuário com preferências de notícias de esporte, como exemplo, é sabido que o o evento de beisebol - superbowl - tem abertura com grandes artigos da música, notícias sobre esse fato poderiam ser de grande interesse para o usuário com preferências de notícias de esporte, e vice-versa.

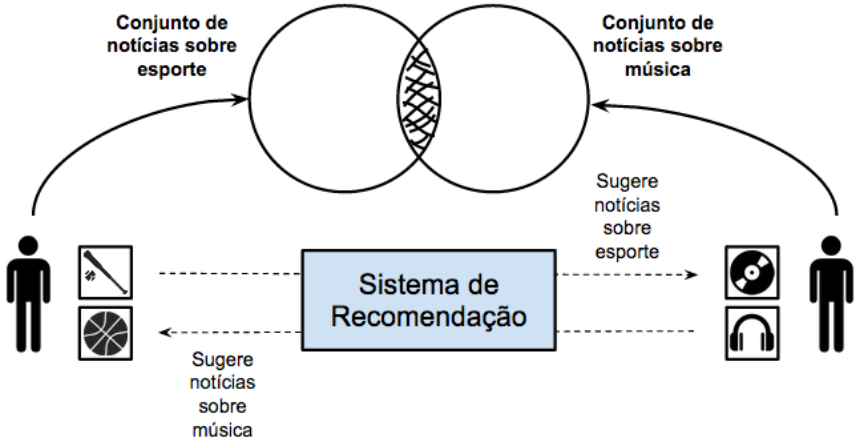


Figura 1 – *Ilustração do caso de dois usuários em um portal de notícias com um SR.*

1.1 Apresentação do Problema de Recomendação

Seja um conjunto de usuários $U = \{u_1, \ldots, u_n\}$, notícias $\mathcal{N} = \{N_1, N_2, \ldots, N_m\}$ que podem ser representadas, por exemplo, como um conjunto de termos $N = \{t_1, t_2, \ldots\}$, e o histórico $h_{u_i,N_j} \in \mathcal{H}$ indicando se o usuário u_i leu a notícia N_j. O problema de recomendação de notícias, com base no contexto, é a definição de uma função de similaridade entre notícias $s : N \to N \times N$ para aprender uma função $l : \mathcal{H}_u, s \to L_u$, onde $\mathcal{H}_u$ é o subconjunto que representa o histórico de notícias que u leu, e L_u a lista de recomendações de notícias direcionadas à u.

A idéia é que a lista de recomendações L_u apresentem *serendipidade* o bastante para serem úteis para o usuário u.

1.2 Apresentação do contexto de Biclusterização

Seja um conjunto de dados definido por um conjunto de objetos $\{x_1, \ldots, x_n\} \in X$ e atributos descritivos desses objetos $\{y_1, \ldots, y_m\} \in Y$. A biclusterização é uma técnica para análise e mineração de dados que é capaz de agrupar objetos (x) similares, conside-

rando condicionalmente, os atributos descritivos (y) desses objetos, formando grupos de objetos e atributos, simultâneamente ou não, chamados de biclusters.

Os biclusters podem apresentar de diferentes formas (Seção 2.2.1), alguns algoritmos impõem restrições à esses grupos, como por exemplo, o par de objeto atributo (x_i, y_j) pode pertencer apenas a um bicluster (KLUGER et al., 2003), enquanto outros permitem sobreposição de biclusters (CHENG; CHURCH, 2000).

1.3 Hipótese

A falta de *serendipidade* em recomendações é um problema para SRs baseados em conteúdo, então, como a Biclusterização é capaz de expandir o espaço de busca, levando em consideração subconjuntos de atributos para a determinação de grupos, ou biclusters, com sobreposição. Então, a hipótese desse projeto é que a aplicação de algoritmos de Biclusterização em SRs baseados em conteúdo pode contribuir para melhorar a função de similaridade entre itens s, otimizando a *serendipidade* e, portanto, melhorando a qualidade das recomendações, fazendo uma contribuição para a área de SRs.

1.4 Objetivos

O objetivo geral desse trabalho é amenizar o problema de oferecer recomendações com *serendipidade* em Sistemas de Recomendação que se baseiam no conteúdo de notícias. O objetivo será atingido através do uso de técnicas de mineração de texto para estruturação do conteúdo textual presentes nas notícias. Então, usando as técnicas e algoritmos de Biclusterização, que possibilita encontrar grupos de notícias com sobreposição de diferentes assuntos. Assim, foram estabelecidos os seguintes objetivos específicos:

- Levantamento do referencial teórico de Sistemas de Recomendação.
  - Levantamento do estado da arte de Sistemas de Recomendação baseados em conteúdo textual.
- Levantamento do referencial teórico de Biclusterização.
- Construção de um corpus de notícias e um conjunto de dados de recomendação.
- Estudo das técnicas de Mineração de Texto.

- Estudo das técnicas de biclusterização.
- Implementação das técnicas de biclusterização aplicadas à texto
- Análise dos resultados obtidos pelas estratégias propostas.

1.5 Metodologia

Para conhecer a área de Sistemas de Recomendação foi realizado um levantamento do referencial teórico através da análise exploratória de livros e artigos do tipo revisão bibliográfica (ADOMAVICIUS; TUZHILIN, 2005; LOPS; GEMMIS; SEMERARO, 2011; RICCI; ROKACH; SHAPIRA, 2011; SHANI; GUNAWARDANA, 2011; JANNACH et al., 2011; BURKE, 2002), o que permitiu, em seguida, o aprofundamento na área de SRs. Para o aprofundamento na área de SRs, foi realizada uma Revisão Sistemática de SRs baseados em conteúdo textual. A Revisão Sistemática (RS) é um tipo de revisão bibliográfica documentada, na qual cada passo da pesquisa é registrado seguindo-se rigorosos critérios, permitindo facilmente a auditoria e reprodução da pesquisa. Segundo Kitchenham (2004), a RS é um meio de avaliar, identificar e interpretar todas as pesquisas relevantes disponíveis em uma determinada área com base em questões de pesquisa. Também foi realizado um levantamento do referencial teórico, através da análise exploratória de livros e artigos do tipo revisão bibliográfica, na área de Mineração de Dados (BERRY; KOGAN, 2010; FELDMAN; SANGER, 2006; MINER et al., 2012; HOTHO; NÜRNBERGER; PAAß, 2005; WEISS; INDURKHYA; ZHANG, 2010), para que fosse possível realizar a análise e estruturação do conteúdo não-estruturado, presentes nas notícias.

Sendo assim, com a extração das notícias do portal iG[1] através de um *web crawler* utilizando a linguagem python[2] e do banco de dados postgreSQL[3] para guardar as notícias, foi possível construir o corpus iG (Subseção 4.1.1) de notícias. As notícias foram capturadas a partir de uma página de início, fornecida para o *web crawler*, que era selecionada a fim de equalizar a distribuição de notícias por ano e por assunto. Este trabalho também conta com uma base de cliques em notícias (Subseção 4.1.2), doada pelo portal de notícias iG.

Para aplicação dos algoritmos de biclusterização, foi realizado um levantamento do referencial teórico, através da análise exploratória de livros, artigos do tipo revisão

[1] http://ig.com.br/
[2] https://www.python.org/
[3] http://www.postgresql.org/

bibliográfica e artigos que se referem à criação dos algoritmos (CHENG; CHURCH, 2000; TANAY; SHARAN; SHAMIR, 2005; MADEIRA; OLIVEIRA, 2004; SANTAMARíA; MIGUEL; THERóN, 2007; KLUGER et al., 2003; PRELIĆ et al., 2006).

Para gerar conhecimento do corpus iG, serão aplicados algoritmos de Biclusterização que estão sendo implementados[4] nos textos das notícias processados e representados por diversas estratégias (TF-IDF, TF-IDF normalizado e n-gramas). Assim, será possível avaliar o resultado dos algoritmos utilizando a base de dados de cliques iG.

1.6 Organização do documento

Este documento é composto por 4 capítulos incluindo esta introdução. O Capítulo 2 conceitos fundamentais necessários para a compreensão deste projeto, incluindo as áreas de Sistemas de Recomendação, Biclusterização e Mineração de Texto; o capítulo 3 apresenta uma revisão bibliográfica de Aprendizado de Máquina aplicado à SRs baseados em Conteúdo; por fim, o capítulo 4 apresenta a proposta desse projeto de mestrado, assim como o cronograma para a sua conclusão.

[4]https://github.com/lucasbrunialti/biclustering-experiments

Capítulo 2

Conceitos Fundamentais

Este capítulo introduz os conceitos fundamentais para o entendimento dessa dissertação, fazendo um apanhado dos conceitos na área de Sistemas de Recomendação (seção 2.1), Biclusterização (seção 2.2) e Mineração de Texto (seção 2.3).

2.1 Sistemas de Recomendação

A grande maioria das pessoas que usam a Internet muito provavelmente já interagiram com algum Sistema de Recomendação (SR), por isso, o seu conceito é intuitivo. Porém, por ser uma área relativamente nova (LOPS; GEMMIS; SEMERARO, 2011), muitos autores não usam uma definicão que reflete a realidade dos SRs, isso acontece por ser uma área relativamente nova que teve um crescimento muito grande nos últimos anos (tese-chap2).

Resnick e Varian (1997), quem criou o termo Sistemas de Recomendação (tese-chap2) (NEUMANN, 2007), argumentam que Sistemas de Recomendação servem para nos ajudar em processos de tomada de decisão do nosso dia-a-dia, como quais itens comprar, quais músicas ouvir, ou quais notícias ler. Além disso, Resnick e Varian (1997) provê uma taxonomia para definição de Sistemas de Recomendação:

- Conteúdo recomendado: os itens que são recomendados pelo Sistema de Recomendação, ex: produtos, músicas, notícias e/ou etc.
- Entrada dos usuários: as interações que os usuários realizam com os itens são a entrada para um SR, estas podem ser implícitas (ex: o usuário x leu a notícia y) ou explícitas (ex: o usuário x classificou o filme y como 5 estrelas).
- Target of recommendation: os itens recomendados podem ser diretamente para um usuário (personalizado), direcionados para um grupo de usuários ou todos os usuários (não-personalizado).
- Técnicas para recomendação (agregações): qual as estratégias e os algoritmos que os SRs usam para criar recomendações.

- Uso das recomendações: trata-se de como mostrar as recomendações para os usuários, ex: filtrando recomendações negativas, ordenando pelo fator numérico, etc.

Porém, definições mais recentes (BURKE, 2002, 2007), descrevem SRs como qualquer sistema que produz recomendações personalizadas ou tem o efeito de guiar um usuário de modo personalizado, mostrando itens que possam ser interessantes para este usuário, dentro de uma grande quantidade de opções. Isso faz com que SRs que provêm recomendações não personalizadas deixem de adequar com a definição de SR. É provável que isso se deve ao fato que das estratégias usadas atualmente, que têm o foco de produzir recomendações personalizadas.

Formalmente, Burke (2002, 2007) define SRs como um conjunto de itens I que podem ser recomendados e um conjunto de usuários U que as preferências são conhecidas, um usuário u pra o qual as recomendações são geradas, e algum item i que queremos predizer a preferência para u. Adomavicius e Tuzhilin (2005) extende a definição com uma função de utilidade f que mede o quão útil é o item i para o usuário u: $f : I \times U \rightarrow R$, em que $R = \{r_{u_1,i_1}, r_{u_1,i_2}, \ldots, r_{u_1,i_m}, \ldots, r_{u_n,i_1}, \ldots, r_{u_n,i_m}\}$ é um conjunto ordenado com valores faltantes, sendo r_{u_i,i_j} um inteiro ou real que representa a interação do usuário u_i no item i_j. No entanto, existem tipos de SRs que não estimam f completamente, podendo otimizar funções auxiliares para gerar as recomendações à um usuário u (LOPS; GEMMIS; SEMERARO, 2011).

Em síntese, um Sistema de Recomendação tem a função de auxiliar os usuários de uma aplicação à interagir com itens, provendo sugestões de quais itens interagir, baseando-se no histórico de interações desses usuários com esses itens.

2.1.1 Tipos de Sistemas de Recomendação

Para estimar f e chegar no conjunto ordenado R existem diversas estratégias, daí surgem os tipos de SRs. Os tipos de SRs diferem quanto ao domínio, informações usadas para recomendação, algoritmos (FELDMAN; SANGER, 2006), e principalmente nas propriedades em que cada tipo se destaca.

Burke (2002, 2007) provê uma taxonomia já considerada clássica Lops, Gemmis e Semeraro (2011), que categoriza os SRs em cinco diferentes tipos:

- *Filtragem colaborativo*, o primeiro tipo de SR que foi implementado (RESNICK; VARIAN, 1997), tem como idéia básica encontrar outros usuários $u_{1,\ldots,n}$ em U, sendo

$n < |U|$ e não necessariamente em ordem, com preferências semelhantes à u, e então recomendar itens que $u_{1,...,n}$ interagiram e que u ainda não interagiu, estabelecendo alguma métrica para estimar f. Medidas geralmente usadas incluem *Correlação de Pearson* e *Similaridade dos Cossenos*, também são usados técnicas para redução de dimensionalidade, como *Decomposição de Valores Singulares* e *Fatorização de Matriz* (JANNACH et al., 2011).

- *Baseado em conteúdo*, é um dos tipos de SR que otimiza funções auxiliares à f. Descreve os itens por características possibilitando o uso de medidas de similaridade entre itens. Então, com as interações dos usuários de U sob itens em I, é construído um perfil de interesses para cada usuário. As recomendações são feitas a partir da combinação do perfil de interesses de um usuário u com os itens em I que u ainda não interagiu. Neste caso são usadas técnicas de Recuperação de Informação (JANNACH et al., 2011) para representar os itens e calcular similaridades entre itens, assim como técnicas de Aprendizado de Máquina Supervisionado e não-supervisionado (JANNACH et al., 2011; BURKE, 2002).

- *Baseado em Conhecimento*, tem o intuito de sugerir itens, de forma personalizada, baseando-se nas necessidades ou regras estabelecidas por um usuário u e nas características dos itens em I. São estabelecidas medidas de similaridade para estimar o quanto as necessidades do usuário match as recomendações (JANNACH et al., 2011; LOPS; GEMMIS; SEMERARO, 2011; BURKE, 2007).

- *Híbrido*, é capaz de combinar as vantagens de cada tipo de SR descrito para suprir as limitações associadas à cada tipo. A dificuldade esta em como combinar as diferentes técnicas de cada algoritmo (JANNACH et al., 2011; BURKE, 2007). Burke (2007) identificou 7 tipos de SRs Híbridos em uma revisão da literatura: Pesagem, atribui um peso para cada algoritmo; Switching, seleciona um dos algoritmos (ou tipos); Mixed, recomendações são mostradas em conjunto; Combinação de características, diferentes fontes são combinadas em apenas um algoritmo; Feature Augumentation, uma técnica é usada para computar características que servem de entrada para outra técnica; Cascade, é atribuído um grau de prioridade para cada algoritmo; Meta-level, uma técnica gera um modelo, que é usado como entrada para outras técnicas.

2.1.1.1 Vantagens e desvantagens

Cada um dos tipos de SRs descritos possuem algumas limitações independentes se comparados com os outros tipos ou não. Em Jannach et al. (2011), Adomavicius e Tuzhilin (2005), Burke (2002), Lops, Gemmis e Semeraro (2011) são nomeadas os problemas no desenvolvimento de SRs de *novo usuário* (*user cold-start*), *novo item* (*item cold-start*), *esparsidade*, *sobre-especialização* e *análise de conteúdo limitada* referentes aos SRs.

Os problemas de *novo usuário* e *novo item* são similares, basicamente, enquanto o primeiro se trata da dificuldade de gerar recomendações para novos usuários, o segundo se trata de gerar recomendações para novos itens. O problema de *novo usuário* esta presente como uma desvantagem nos SRs de filtragem colaborativa e baseado em conteúdo, pois o SR não conhece as preferências dos novos usuários, tendo dificuldade de construir um modelo que tem como base essas preferências. Já o problema de *novo item* é considerado uma vantagem para os SRs baseados em conteúdo, enquanto uma desvantagem para os SRs baseados em filtragem colaborativa. Como a filtragem colaborativa se baseia apenas nas interações de usuários em itens, um item novo, que não teve ou teve poucas interações, não será recomendado, diferentemente do SR baseado em conteúdo, que leva em consideração a representação do item para construir recomendações.

Contrariamente, o problema de *sobre-especialização* é uma vantagem para os SRs de filtragem colaborativa e uma desvantagem para os SRs baseados em conteúdo. A *sobre-especialização* diz respeito ao problema de sugerir apenas itens previsíveis para o usuário, por exemplo, se o usuário viu notícias apenas de esporte, ele já espera receber sugestões de notícias de esporte, porém este usuário muito provavelmente pode gostar de ler notícias de outras categorias. A capacidade do SR de sugerir notícias imprevisíveis é chamado de *serendipidade* (JANNACH et al., 2011; LOPS; GEMMIS; SEMERARO, 2011). SRs baseados em conteúdo sofrem desse problema pois combina o perfil de preferências de um usuário com os itens em I, restrigindo o espaço de busca para realizar a sugestão de itens. Enquanto isso, os SRs baseados em filtragem colaborativa são capazes de oferecer sugestões úteis e serendipitas, pois são capazes de ampliar o espaço de busca através da estratégia de sugerir itens que usuários semelhantes à u interagiram e que u ainda não interagiu.

Os SRs baseados em filtragem colaborativa são os únicos que sofrem do problema de *esparsidade*, que é o fato de usuários interagirem com apenas um pequeno subconjunto do conjunto de itens, tornando a matriz de interações ou preferências ($U \times I$) esparsa. Isso faz com que aumente a necessidade de ter uma grande quantidade de usuários, pois

este tipo de SR necessita de intersecções nas interações de itens por usuários, para que seja possível encontrar usuários similares a um dado usuário.

Assim como os SRs de filtragem colaborativa, os SRs baseados em conteúdo sofrem de um problema único, que é a *análise de conteúdo limitada*. Este problema se refere à representação dos itens, que tem de ser suficiente para discriminá-los (LOPS; GEMMIS; SEMERARO, 2011). Em SRs baseados em conteúdo é comum ter que limitar a representação de itens, tanto pelo número de características quanto pela modelagem. Por exemplo, no contexto de notícias na web, se a representação for um vetor com o número de ocorrências de cada palavra, perdemos a relação entre as palavras, e também todo o conteúdo que não é texto, como imagens, vídeos, etc.

2.1.2 Avaliação da Recomendação

Diferentes técnicas foram mostradas e uma variedade de problemas relacionados à essas técnicas, mas como avaliar se uma estratégia adotada no desenvolvimento realmente é efetiva? SRs podem ser avaliados através de *experimentos online*, que podem descobrir a real influência do SR no comportamento do usuário, e *experimentos offline*, que estimam o erro de predição e simulam o comportamento do usuário no SR usando um conjunto de dados (SHANI; GUNAWARDANA, 2011).

Para experimentos online podem ser estabelecidas variáveis através da captura implícita ou explícita do comportamento, como satisfação do usuário e taxa de cliques (*click-through rate* - CTR). Uma das maneiras para realizar esse tipo de experimento é por meio de testes A/B (JANNACH et al., 2011), em que cada usuário, ao interagir com o SR, recebe um tratamento diferente aleatóriamente, dentro dos possíveis tratamentos estabelecidos pelo experimento. Assim, é possível dizer, por exemplo, a influência de um novo componente no SR no comportamento dos usuários.

Tradicionalmente, SRs são avaliados através de experimentos offline (JANNACH et al., 2011). Por serem simples, esses tipos de experimentos podem ser usados para para selecionar algoritmos, no entanto, Shani e Gunawardana (2011) argumentam que não é possível medir diretamente a influência das recomendações no comportamento dos usuários. Para simular o comportamento do usuário em um SR usando um conjunto de dados, são estabelecidos dois subconjunto das interações de u aleatoriamente ($\{r_{u,i_4}, r_{u,i_3}, \dots\}$), um para treinamento dos modelos conj treino_u e outro para teste conj teste_u. Assim é possível adotar estratégias semelhantemente com às adotadas em Aprendizado de Máquina, como *matriz de confusão*, *precisão*, *revocação*, *f1-score*, *cross-validation* e etc.

A técnica de *matriz de confusão* é possível ser usada em *experimentos online* e *offline*, da seguinte maneira: se o usuário gostar do item sugerido à ele, é considerada uma predição correta (verdadeiro positivo); se o usuário não solicita preferência pela sugestão ou se não existe informações da preferência do usuário para esta sugestão, será considerada uma predição errada (falso positivo); contrariamente, se o SR não fazer essas sugestões, é considerada uma emissão correta (verdadeiro negativo); por fim, se o SR não sugerir itens que o usuário tem preferência, é consuderada uma predição errada (falso negativo).

Uma das métricas mais comuns usadas para *experimentos offline* é o *erro absoluto médio* (*Mean Absolute Error* - MAE) e *raiz do erro quadrático médio* (*Root Mean Squared Error* - RMSE) (JANNACH et al., 2011), o qual foi usado como métrica para a competição Netflix Prize[1], que teve grande repercursão na academia e na indústria. A defição formal das métricas MAE e RMSE são descritas nas equações 2.1 e 2.2, respectivamente.

$$MAE = \sum_{u \in U} \frac{\sum_{i \in \text{conj teste}_u} |f(u,i) - r_{u,i}|}{|\text{conj teste}_u|} \tag{2.1}$$

$$RMSE = \sum_{u \in U} \sqrt{\frac{\sum_{i \in \text{conj teste}_u} (f(u,i) - r_{u,i}^2}{|\text{conj teste}_u|}} \tag{2.2}$$

A métrica MAE computa o erro médio entre as predições feitas pelo SR ($f(u,i)$) e os valores reais das preferências dos usuários ($r_{u,i}$) para todos os usuários em U, enquanto o RMSE amplifica erros grandes, pois eleva o mesmo ao quadrado. Essas métricas são usadas para valores reais, ou seja, $r_{u,i} \in [0,1]$, por exemplo.

Para valores binários de $r_{u,i}$ ou quando deseja-se prever o número de recomendações relevantes para um usuário u, são usadas as métricas *precisão* e *revocação* (JANNACH et al., 2011).

Para medir a *serendipidade* das recomendações, Shani e Gunawardana (2011) propõem uma estratégia: estabelecer uma medida de distância entre itens e rotular os itens com menor distância entre si como ausentes de *serendipidade*, assim, algoritmos que evitem esses itens, serão considerados superiores.

[1] http://www.netflixprize.com/

2.2 Biclusterização

Técnicas e algoritmos de biclusterização são usadas, principalmente, no contexto de expressão genética. No entanto, algoritmos de biclusterização se fazem útil quando se deseja encontrar *modelos locais*. Ou seja, enquanto algoritmos de clusterização têm o intuito de encontrar *modelos globais*, que geram grupos de dados levando em consideração todas as características, algoritmos de biclusterização geram grupos de dados em que as características tem alta correlação (MADEIRA; OLIVEIRA, 2004).

Para a descrição do problema formal de biclusterização usa-se a seguinte definição (MADEIRA; OLIVEIRA, 2004): uma matriz A, $n \times m$, um conjunto de linhas $X = \{x_1, \ldots, x_n\}$ e um conjunto de colunas $Y = \{y_1, \ldots, y_m\}$, em que a_{ij}, geralmente um número real, e representa a relação entre a linha x_i e a coluna y_j. O problema de biclusterização é encontrar biclusters, que são submatrizes de A, denotados por A_{IJ}, em que $I \subseteq X$ e $J \subseteq Y$. Assim, o bicluster A_{IJ} é um grupo dos exemplos em I, perante as características com alta correlação J.

2.2.1 Tipos de biclusters

Como a definição de bicluster não inclui uma prévia estrutura da matriz A e dos biclusters A_{IJ}, diversos algoritmos propostos na literatura diferem quanto ao tipo de bicluster que são capazes de encontrar. Uma taxonomia dos tipos de biclusters é proposta por Madeira e Oliveira (2004):

- *Biclusters com valores constantes*, se trata de biclusters em que todos os valores de A_{IJ} são constantes: $a_{ij} = \mu, \forall i, j \in I, J$, onde μ é um valor constante dentro de A_{IJ}. Porém, em conjuntos de dados reais, esses biclusters estão presentes com algum tipo de ruído $\mu + \eta_{ij}$, onde η_{ij} é o ruído associado com os valures de μ e a_{ij} (MADEIRA; OLIVEIRA, 2004).

- *Biclusters com valores constantes nas linhas ou colunas*, se trata de biclusters com valores constantes nas linhas: $a_{ij} = \mu + \alpha_i, \forall i, j \in I, J$ ou $a_{ij} = \mu \cdot \alpha_i, \forall i, j \in I, J$, onde α_i é um fator aditivo ou multiplicativo para cada linha; ou ainda biclusters com valores constantes nas colunas: $a_{ij} = \mu + \beta_j, \forall i, j \in I, J$ ou $a_{ij} = \mu \cdot \beta_j, \forall i, j \in I, J$, onde β_j é um fator aditivo ou multiplicativo para cada coluna (MADEIRA; OLIVEIRA, 2004).

- *Biclusters com valores coerentes*, em que são considerados valores próximos entre

si (coerentes) para definição de um bicluster: $a_{ij} = \mu + \alpha_i + \beta_j, \forall i, j \in I, J$, ou $a_{ij} = \mu' \cdot \alpha'_i \cdot \beta'_j, \forall i, j \in I, J$, sendo que se $\mu = \log \mu' \implies \alpha_i = \alpha'_i, \beta_j = \beta'_j$ (MADEIRA; OLIVEIRA, 2004).

- *Biclusters com evoluções coerentes*, têm seus valores com evoluções coerentes, por exemplo, um bicluster com $a_{i4} \leq a_{i3} \leq a_{i2} \leq a_{i1}$ tem valores com evolução coerente na coluna (MADEIRA; OLIVEIRA, 2004). Seus valores podem ser gerados por uma função geradora de valores com evolução coerente $a_{ij} = g(a_{ij}), \forall i, j \in I, J$, sendo $g(\cdot)$ não linear e não constante, para que o tipo de bicluster não seja classificado nos casos anteriores.

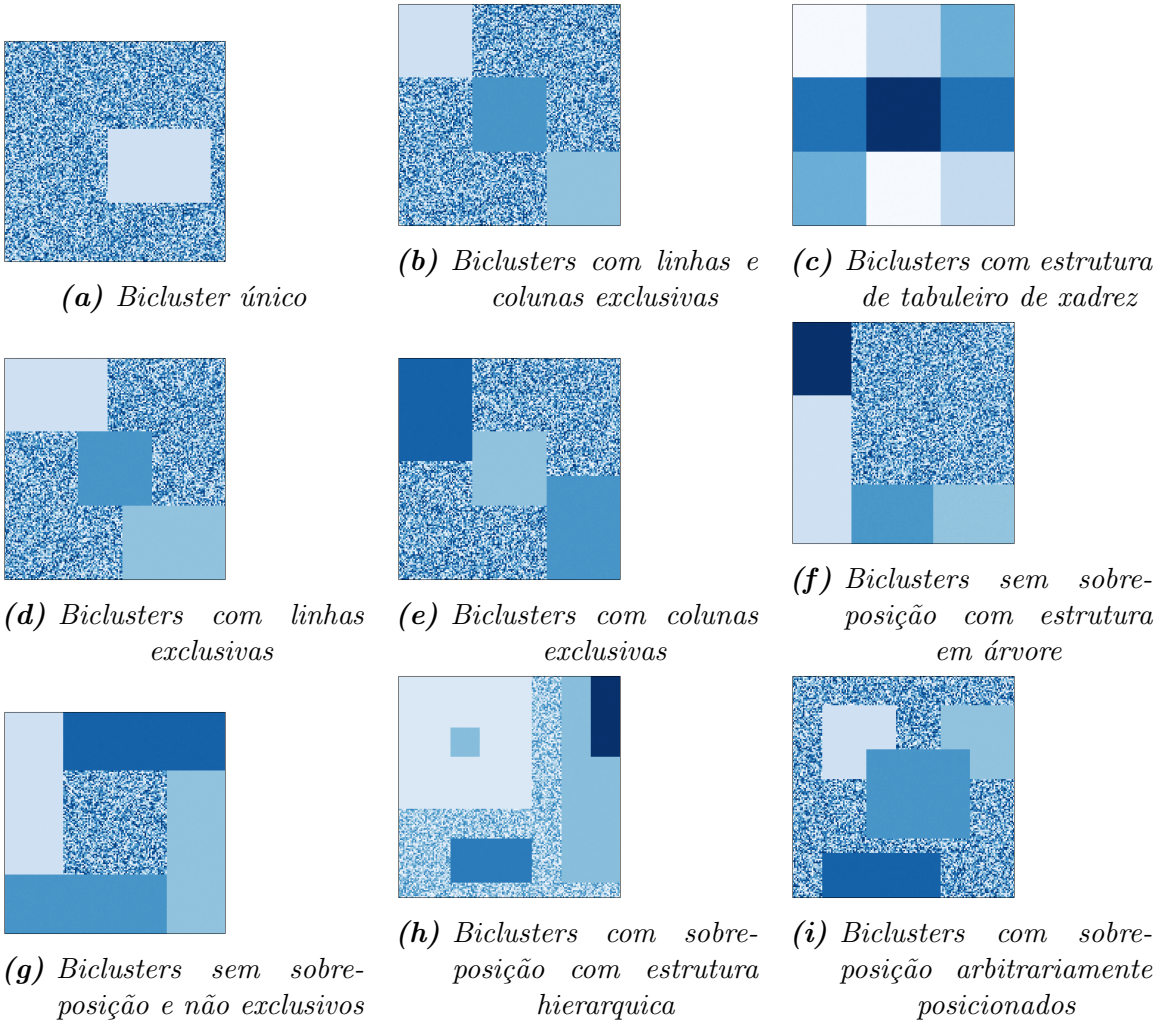
Os biclusters também se diferem quanto a sua estrutura, cada algoritmo faz uma suposição da estrutura de biclusters que é capaz de encontrar. A Figura 2 sumariza as diferentes estruturas de biclusters, com as linhas e colunas ordenadas, para permitir a visualização dos biclusters por meio do mapa de calor dos valores de A.

2.2.2 Algoritmos para biclusterização

Diversos algoritmos para encontrar biclusters, de diferentes tipos e estruturas, foram propostos na literatura (TANAY; SHARAN; SHAMIR, 2005; MADEIRA; OLIVEIRA, 2004).

Um dos mais comuns e simples algoritmos de biclusterização, que encontra biclusters com valores coerentes e estrutura com sobreposição e arbitrariamente posicionados, é o *Coupled Two-way Clustering* (CTWC) (GETZ; LEVINE; DOMANY, 2000), pois encontra biclusters através da clusterização de características e exemplos (linhas e colunas), separadamente. O algoritmo de clusterização usado por (GETZ; LEVINE; DOMANY, 2000) foi o *Superparamagnetic Clustering* (SPC), pois é capaz de determinar o número de clusters automaticamente e com uma estratégia de clusterização hierárquica *top-down* é capaz de gerar clusters estáveis (GETZ; LEVINE; DOMANY, 2000). O SPC tem como entrada uma matriz de similaridade e um parâmetro temperatura, que controla o quão estáveis serão os clusters que o algoritmo irá gerar. Assim, o CTWC encontra clusters estáveis de linhas e colunas através do SPC, e iterativamente, performa o SPC novamente nos clusters de linhas e colunas encontrados, mantendo na memória um par do subconjunto de linhas e do subconjunto de colunas (biclusters), assim como os clusters estáveis de linhas e colunas, separadamente.

O algoritmo Cheng e Church (CHENG; CHURCH, 2000), é capaz de encontrar o



(a) *Bicluster único*

(b) *Biclusters com linhas e colunas exclusivas*

(c) *Biclusters com estrutura de tabuleiro de xadrez*

(d) *Biclusters com linhas exclusivas*

(e) *Biclusters com colunas exclusivas*

(f) *Biclusters sem sobreposição com estrutura em árvore*

(g) *Biclusters sem sobreposição e não exclusivos*

(h) *Biclusters com sobreposição com estrutura hierarquica*

(i) *Biclusters com sobreposição arbitrariamente posicionados*

Figura 2 – *Diferentes estruturas de biclusters, quadrados com cores sólidas representam biclusters (Adaptado de (MADEIRA; OLIVEIRA, 2004))*

mesmo tipo de bicluster que o algoritmo CTWC, de forma gulosa: biclusters com valores coerentes e estrutura com sobreposição e arbitrariamente posicionados. Este algoritmo esta sendo objeto de estudo desse projeto de mestrado para aplicação em dados textuais. Cheng e Church (2000) introduz o conceito de bicluster no domínio de expressões genéticas. Além disso, para encontrar biclusters, ou δ-biclusters, na matriz A, os autores definem o *Resíduo Quadrático Médio* (RQM):

$$\begin{aligned} H_{IJ} &= \frac{1}{|I||J|} \sum_{i,j \in I,J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \\ H_{iJ} &= \frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \\ H_{Ij} &= \frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \end{aligned} \tag{2.3}$$

em que

$$a_{iJ} = \frac{1}{|J|}\sum_{j\in J} a_{ij}, \quad a_{Ij} = \frac{1}{|I|}\sum_{i\in I} a_{ij}, \quad a_{IJ} = \frac{1}{|I||J|}\sum_{i,j\in I,J} a_{ij} \tag{2.4}$$

onde H_{IJ} é o RQM de uma submatriz A_{IJ}, H_{iJ} o RQM da linha i, H_{Ij} o RQM da coluna j, a_{iJ} a média dos valores da linha i, a_{Ij} a média dos valores da coluna j e a_{IJ} a média dos valores da submatriz A_{IJ}, definida pelos subconjuntos I e J.

Então, um bicluster um perfeito A_{IJ} teria o RQM $H_{IJ} = 0$, pois $a_{ij} = a_{ij}, \forall i, j \in I, J$, fazendo $a_{iJ} = a_{Ij} = a_{IJ}$. No entanto, se apenas minizar o RQM, um bicluster com apenas um elemento seria perfeito, o que pode não refletir a realidade. Além disso, em conjunto de dados reais existe ruído, podendo esconder o bicluster perfeito.

Para encontrar biclusters, ou δ-biclusters, Cheng e Church (2000) fazem uma estratégia gulosa que retira linhas e colunas, visando a minimização do RQM, respeitando um parâmetro δ, que é calibrado pelo usuário. Então, um bicluster é encontrado quando o RQM de uma submatriz A_{IJ} é $H_{IJ} \leq \delta$, para algum $\delta \geq 0$. As etapas de remoções de elementos da matriz são apresentadas nos algoritmos 1 e 2.

Input: A, I, J, δ
Output: I, J onde $H_{IJ} \leq \delta$
while $H_{IJ} > \delta$ **do**
    encontre a linha $l_{max} = \arg\max_{i\in I} H_{iJ}$;
    encontre a coluna $c_{max} = \arg\max_{j\in J} H_{Ij}$;
    if $l_{max} > c_{max}$ **then**
        remova l_{max} do subconjunto I;
    else
        remova c_{max} do subconjunto J;

Algoritmo 1: Remove uma linha ou coluna a cada iteração.

Input: A, I, J, δ, α
Output: I, J onde $H_{IJ} \leq \delta$
while $H_{IJ} > \delta$ **do**
    remova $i \in I$ onde $H_{iJ} > \alpha \cdot H_{IJ}$;
    remova $j \in J$ onde $H_{Ij} > \alpha \cdot H_{IJ}$;
    encontre a coluna $c_{max} = \arg\max_{j\in J} H_{Ij}$;
    if *não houve nenhuma remoção* **then**
        chame o algoritmo 1

Algoritmo 2: Remove múltiplas linhas e colunas de A a cada iteração.

O algoritmo 2 é usado para acelerar o processo de busca de um δ-bicluster, convergindo mais rapidamente para uma solução quanto maior for o parâmetro α, em que $\alpha \geq 0$. Ainda, para amenização do problema de encontrar δ-biclusters perfeitos com apenas um elemento, ou poucos elemento, é utilizado o algoritmo **??**, que adiciona nós sem aumentar o RQM do bicluster.

Input: A, I, J
Output: I', J' onde $H_{I'J'} \leq H_{IJ}$
1 **while** $H_{IJ} > \delta$ **do**
2 compute a_{iJ} e a_{Ij} para todo i, j e H_{IJ};
3 remova $j \notin J$ onde $\frac{1}{|I|}\sum_{j \in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H_{IJ}$;
4 recompute a_{iJ} e H_{IJ};
5 remova $i \notin I$ onde $\frac{1}{|J|}\sum_{i \in I}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H_{IJ}$;
6 **if** *não houve nenhuma adição* **then**
7 $I' \longleftarrow I$;
8 $J' \longleftarrow J$;

Algoritmo 3: Adiciona linhas e colunas de A_{IJ} a cada iteração.

Por fim, o algoritmo **??** é a consolidação dos algoritmos **??**, 2 e 1 e a iteração para encontrar k δ-biclusters, um a um, sendo k fornecido pelo usuário.

Input: A, k, δ, α
Output: k δ-biclusters
1 $A' \leftarrow A$;
2 **for** 1 **to** k **do**
3 $B \leftarrow$ algoritmo 2 com A', δ, α;
4 $C \leftarrow$ algoritmo 1 com B, δ;
5 $D \leftarrow$ algoritmo **??** com A, C;
6 reporte D como uma solução;
7 adicione ruído em A' para a submatriz D;

Algoritmo 4: Algoritmo Cheng & Church, encontra k δ-biclusters.

Além dos algoritmos apresentados, existem outros algoritmos que são capazes de encontrar outros tipos de biclusters (seção 2.2.1), além de serem recentes (YANG; LESKOVEC, 2013; HOCHREITER et al., 2010; CABANES; BENNANI; FRESNEAU, 2012), mostrando que ainda há interesse na área de pesquisa de biclusterização.

2.2.3 Avaliação de biclusterização

Para determinar parâmetros, descobrir a qualidade e/ou estabilidade dos biclusters encontrados por algoritmos, é necessário estabelecer métricas de avaliação. Existem

duas maneiras de avaliar biclusters (HOCHREITER et al., 2010): *interna*, usa os dados dos resultados dos algoritmos, juntamente com métricas de qualidade e/ou estabilidade, para avaliar as soluções geradas; *externa*, utiliza os dados reais das soluções de biclusters de um conjunto de dados, usando estratégias para comparação, obtendo assim, maior confiança nas soluções.

A avaliação interna pode não ser tão precisa quanto a avaliação externa, porém é útil para descobrir parâmetros ótimos. Apesar de PreliĆ et al. (2006) sugerirem não usar avaliações internas, por não estar claro como extender noções de separação e homogeinidade, no entanto, Santamaría, Miguel e Therón (2007) descreveu métricas de consistência para verificando se um bicluster é consistente com a sua definição, seja aditiva, multiplicativa e/ou constante, fazendo uma comparação dos elementos do bicluster:

$$\begin{aligned} C_l(A_{IJ}) &= \frac{1}{|I|} \sum_{i=1}^{|I|-1} \sum_{j=i+1}^{|I|} \sqrt{\sum_{k=1}^{|J|} (a_{ik} - a_{jk})^2} \\ C_c(A_{IJ}) &= \frac{1}{|J|} \sum_{i=1}^{|J|-1} \sum_{j=i+1}^{|J|} \sqrt{\sum_{k=1}^{|I|} (a_{ki} - a_{kj})^2} \end{aligned} \tag{2.5}$$

em que $C_l(A_{IJ})$ é o índice de consistência das linhas do bicluster A_{IJ} e $C_c(A_{IJ})$ é o índice de consistência das colunas do bicluster A_{IJ}. Ainda, a consistência do bicluster inteiro C pode ser definida pela média:

$$C(A_{IJ}) = \frac{|I| \cdot C_l + |J| \cdot C_c}{|I| + |J|} \tag{2.6}$$

Uma das métricas externas, que são usadas para comparar biclusters encontrados com biclusters reais em um conjunto de dados, é a métrica *concensus score* (HOCHREITER et al., 2010). Através da maximização das similaridades entre biclusters encontrados e reais, usando o *índice de Jaccard* como medida de similaridade e o algoritmo Húngaro para solucionar o problema de maximização. Tendo como saída um *score* $\in [0, 1]$, em que 0 significa que os biclusters comparados são totalmente diferentes, e 1 o inverso.

2.3 Mineração de Texto

Técnicas de Mineração de Texto são muito usadas para SRs baseados em conteúdo textual (LOPS; GEMMIS; SEMERARO, 2011), principalmente quando o contexto do SR se trata de informações não-estruturadas. Mineração de Texto lida com análise de texto, suportando a sua natureza não-estruturada, imprecisa, incerta e difusa, para extração de

informação e conhecimento (HOTHO; NÜRNBERGER; PAAß, 2005). Além disso, a área de Mineração de Texto utiliza de técnicas das áreas de Recuperação de Informação, Processamento de Linguagem Natural (PLN), conectando essas técnicas com algoritmos e métodos de Descoberta de Conhecimento em Banco de Dados, Mineração de Dados, Aprendizado de Máquina e Estatística (HOTHO; NÜRNBERGER; PAAß, 2005).

Feldman e Sanger (2006), apresentam uma arquitetura geral para aplicações de Mineração de Textos composta por quatro etapas: tarefas de pré-processamento, que preparam os dados para a central de operações de mineração; central de operações de mineração, incluem algoritmos para a descoberta de padrões, tendências e conhecimentos através de técnicas e algoritmos; componentes de apresentação, incluem interfaces para o usuário apresentando visualizações dos conhecimentos gerados na etapa anterior; e técnicas de refinamento, também descrito como uma fase de pós-processamento, que inclui métodos para filtrar informações redundantes.

2.3.1 Tarefas de pré-processamento

As tarefas de pré-processamento incluem rotinas, processos e métodos para a estruturação dos textos presentes nos documentos. A estruturação se faz necessária para a extração de informações e descoberta de conhecimento por meio de técnicas e algoritmos (HOTHO; NÜRNBERGER; PAAß, 2005).

2.3.1.1 Representação textual

Para a estruturação dos textos é necessário a definição da representação textual dos documentos. O vetor de palavras, ou *Vector Space Model* (SALTON; WONG; YANG, 1975), é a representação clássica usada para representar documentos textuais (referencia). Cada dimensão desse vetor esta associado à palavra, sendo que todas as dimensões representam todas as palavras do conjunto de documentos. Formalmente, temos um conjunto de documentos $D = \{d_1, d_2, \ldots, d_n\}$, em que d_i representa um documento e n o número total de documentos, e um conjunto de palavras $P = \{p_1, p_2, \ldots, p_m\}$, em que p_j representa uma palavra e m o número de palavras presentes em todos os documentos. Representando a frequência de uma palavra, o número de vezes que p_j aparece em um documento d_i, por $fp(p_j, d_i)$, o vetor de palavras pode ser construído e representado da seguinte forma: $\vec{vp}_{d_i} = (fp(p_1, d_i), fp(p_2, d_i), \ldots, fp(p_m, d_i))$. Salton, Wong e Yang (1975) argumentam que a representação textual de documentos em vetor de palavras é suficiente para separar documentos. Ao invés de frequência de palavra, também é comumente

usado, a representação binária (referencia), ou seja, p_j aparecendo em d_i corresponde à entrada 1 na dimensão j em $\vec{vp}_{d_i}$. Há também outros métodos para representação textual, como *n-gramas* e *ontologias*.

Ainda sobre o vetor de palavras, Salton, Wong e Yang (1975) mostra com experimentos em diversos conjuntos de dados, que o uso da normalização nos vetores usando a técnica de Frequência de Palavras-Frequência de Documentos Inversa (*Term Frequency-Inversed Document Frequency* - TFIDF) é capaz de melhorar a separação de documentos:

$$\begin{aligned} \textit{fp-fdi}(p_j, d_i) &= fp(p_j, d_i) \cdot fdi(p_j) \\ \textit{fp-fdi}(p_j, d_i) &= fp(p_j, d_i) \cdot \left(log_2 \frac{n}{fd(p_j) + 1} \right) \end{aligned} \tag{2.7}$$

em que $fdi(p_j)$ representa a Frequência de Documentos Inversa da palavra p_j e $fd(p_j)$ a frequência de documentos que contém p_j. Essa normalização faz com que a frequência das palavras que aparecem em muitos documentos seja reduzida, e a frequência das palavras que aparecem em alguns raros documentos seja aumentada, com um fator de $\log_2$.

2.3.1.2 Tokenização

Para realizar a estruturação de textos e representar os textos dos documentos em vetores de palavras, o primeiro processo a ser realizado é a *tokenização*, que cria um dicionário de palavras para cada documento através da quebra dos textos desses documentos. A quebra do texto pode ser feita através de caracteres delimitadores de termos, como espaços em branco, pontuações e etc. No entanto, existem casos que esses caracteres podem não ser delimitadores de termos, como por exemplo os termos *Prof.* e *Sr.*. Este problema é chamado de determinação de fim de sentença, e pode ser resolvido por métodos estáticos (*hard-coded*), baseados em regras e métodos de Aprendizado de Máquina (WEISS; INDURKHYA; ZHANG, 2010).

2.3.1.3 Filtragem

Métodos de filtragem têm a função de retirar palavras do conjunto P que não contribuem para distinguir ou identificar documentos, como exemplo, conjunções (*e*, *pois*, *que*), artigos (*um*, *o*, *a*), preposições (*de*, *para*) e etc. A técnica de retirar determinadas palavras de P a partir de uma lista, é chamada de *stopwords*. Também são usadas outras técnicas, como a eliminação de palavras com a frequência muito alta ou muito baixa.

2.3.1.4 Stemming

A fim de reduzir a ambiguidade de palavras, o método de *stemming* é capaz de juntar em uma única forma palavras relacionadas (MINER et al., 2012), como exemplo o verbo *fazer*, que pode se apresentar em diversas formas, como *fazendo*, *fez*, etc. Esse processo pode ser capaz de aumentar a capacidade da representação em distinguir ou identificar documentos, além de reduzir a dimensionalidade, reduzindo também a esparsidade.

2.3.1.5 Redução de Dimensionalidade

A representação em vetor de palavras pode resultar em vetores esparsos num espaço de alta dimensão, que pode fazer com que algoritmos sofram do problema de *Maldição de Dimensionalidade* (footnote haykin). Para amenização desse problema, são usados métodos de *redução de dimensionalidade*. A técnica mais comum de *redução de dimensionalidade* é chamada *Análise dos Componentes Principais* (*Principal Component Analysis - PCA*) (MURPHY, 2012). Esta técnica tem o objetivo de encontrar uma representação compacta através da descoberta de k vetores n-dimensionais ortogonais aos dados ($\vec{v}$), em que $k \leq m$. Os vetores são encontrados a partir da minimização da projeção dos dados em $\vec{v}$. Depois de encontrados os vetores $\vec{v}$, é feita a projeção dos dados nesses vetores, resultando em uma representação num espaço mais compacto (HAN; KAMBER; PEI, 2011). É possível aplicar o algoritmo *PCA*, no vetor de palavras, diminuindo a dimensionalidade e esparsidade, superando o problema de *Maldição de Dimensionalidade*.

Capítulo 3

Sistemas de Recomendação baseados em Conteúdo e Aprendizado de Máquina

Os CBRSs têm fortes relações com a área de Recuperação de Informação (ADOMAVICIUS; TUZHILIN, 2005; JANNACH et al., 2011) e Aprendizado de Máquina (ADOMAVICIUS; TUZHILIN, 2005; LOPS; GEMMIS; SEMERARO, 2011) para representação de itens e perfis de usuários, e aprendizado do perfil do usuário. Basicamente, este tipo de sistema analisa o conteúdo de diversos itens, extraindo atributos para representação, com esses mesmos atributos (ou às vezes até mais (CAPELLE et al., 2012)) representa-se o perfil do usuário. Sabendo os interesses dos usuários, através do perfil construído, o sistema seleciona itens que o usuário ainda não consumiu e que sejam relacionados com os seus interesses.

3.1 Arquitetura de Sistemas de Recomendação baseados em Conteúdo

Em (LOPS; GEMMIS; SEMERARO, 2011) é proposta uma arquitetura para o desenvolvimento de CBRSs, a qual separa o processo de recomendação em três fases (Figura 3). O analisador de conteúdo tem como entrada os itens não estruturados, assim, através de técnicas de representação e pré-processamento, são obtidos os itens representados. Uma das representações mais comuns (ADOMAVICIUS; TUZHILIN, 2005; LOPS; GEMMIS; SEMERARO, 2011; JANNACH et al., 2011) para representação textual é o Modelo do Espaço Vetorial (SALTON; WONG; YANG, 1975), aonde cada item ou documento $d_i \in D$, sendo D um conjunto de documentos de tamanho n, é representado por um vetor de termos $d_i = \{t_1, t_2, \ldots, t_m\}$, em que m é o número de termos. Assim, calcula-se o $TFIDF_{t_k}^{d_i}$ *(Term-Frequency Index-Frequency)* do documento d_i e o termo t_k (SALTON; WONG; YANG, 1975):

Em que $f_{t_k}^{d_i}$ é a frequência do termo t_k no documento d_i e $f_{d_i}^{t_k}$ é a número de

documentos d_i que possuem t_k. Fazendo com que o peso de termos que aparecem em muitos documentos seja reduzido. Há também outros métodos de representação, como *n-grams* e semântica (LOPS; GEMMIS; SEMERARO, 2011; JANNACH et al., 2011).

Com a representação dos itens estruturada realizada, ocorre a representação dos perfis, que geralmente é baseado na representação dos itens, ou seja, o perfil do usuário u_j é dado por $\{< d_1, r_1 >, \ldots, < d_j, r_J >\}$, sendo r_j o quão o usuário gostou do item, seja pela manifestação explícita, por exemplo em que o usuário avaliou o item, ou pela manifestação implícita, tendo como exemplo quando o usuário lê uma notícia, fica muito tempo na página, etc. Finalmente, é possível aprender os perfis dos usuários com Aprendizado de Máquina (ADOMAVICIUS; TUZHILIN, 2005; LOPS; GEMMIS; SEMERARO, 2011; JANNACH et al., 2011), por exemplo, para prever se o usuário gosta ou não de um determinado item (classificação).

A terceira fase é o Componente de Filtragem, que basicamente pega a saída do classificador, seleciona os itens mais relevantes para os usuários, e apresenta uma lista de recomendações. Geralmente, essa lista é ordenada e apresentada os top N itens mais relevantes (ver Seção **??**).

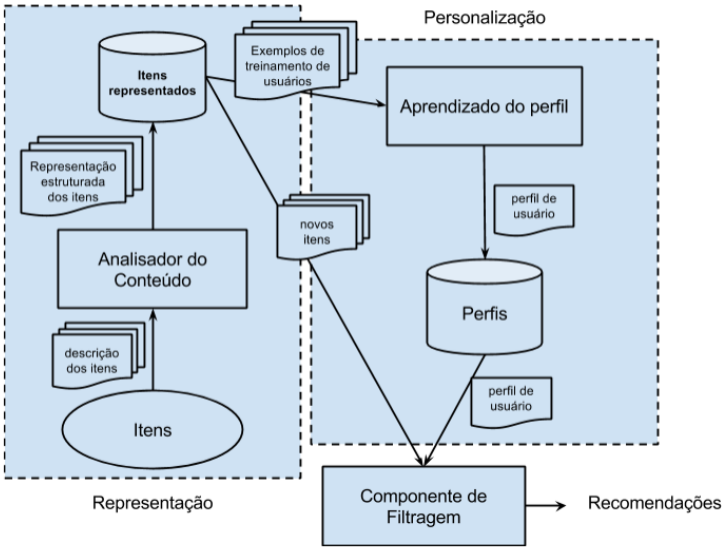


Figura 3 *– Arquitetura de um CBRS (Adaptado de (LOPS; GEMMIS; SEMERARO, 2011)).*

Essa arquitetura apresentada foi adaptada de (LOPS; GEMMIS; SEMERARO, 2011), para maior compreendimento e melhor extração de informações nos trabalhos revisados (ver Seção **??**).

Capítulo 4

Proposta

A proposta desse projeto de mestrado envolve a aplicação de algoritmos de biclusterização para o problema de recomendação baseada em conteúdo textual, a qual, segundo a revisão sistemática que esta sendo realizada, foi muito pouco explorada: apenas um entre 1 943 estudos primários relevantes nas bases de dados (não publicado) se refere à aplicação de biclusterização em conteúdo textual.

Com maior especificidade, a proposta desse projeto (Figura 4) é definida pela utilização de técnicas de mineração de texto para o processamento das notícias presentes no corpus iG (Subseção 4.1.1), após a estruturação do texto, serão realizadas diversas representações (TF-IDF, TF-IDF normalizado e n-gramas) para a criação de biclusters com algoritmos de biclusterização, realizando experimentos para verificar a consistência dos biclusters criados pelos diferentes algoritmos aplicados. Assim, será definida uma estratégia para gerar uma lista de recomendações e comparar com a base de cliques iG (Subseção 4.1.2), utilizando métricas de SRs para gerar experimentos e avaliar as recomendações geradas. Além disso, será feita uma comparação com um SR do tipo filtro colaborativo, para comparar a *serendipidade* entre o método proposto e o filtro colaborativo.

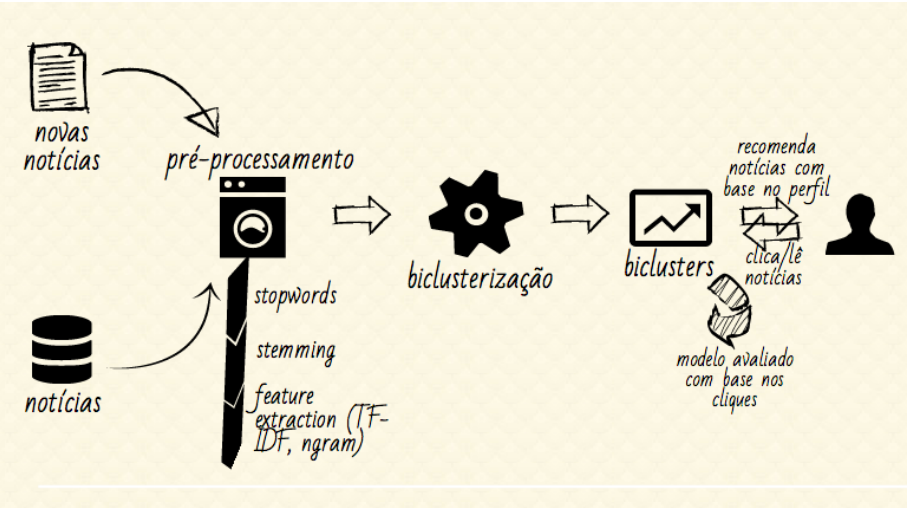


Figura 4 – *Ilustração da proposta desse mestrado.*

4.1 Bases de dados iG

4.1.1 Corpus iG

O corpus iG, construído neste projeto, é composto por um conjunto de notícias do portal iG[1] $\mathcal{N} = \{n_1, n_2, \ldots, n_m\}$, em que cada notícia n_i é representada pela tupla $(\textit{permalink}, \textit{título}, \textit{subtítulo}, \textit{corpo}, c_i)$, em que *permalink* é o endereço eletrônico fixo da notícia, e, c_i um elemento do conjunto de canais $\mathcal{C} = \{\textit{gente}, \textit{ultimosegundo}, \textit{delas}, \textit{economia}, \textit{esporte}, \textit{saude}, \textit{igay}, \textit{deles}, \textit{tecnologia}, \textit{igirl}, \textit{jovem}, \textit{arena}, \textit{luxo}\}$, onde cada canal representa um assunto ou tópico de notícias.

O número total de notícias do corpus é $m = 4\ 593$, com mais de 250 caracteres no corpo, no perído de 02 de Janeiro de 2012 até 11 de Outubro de 2014. As notícias estão bem distribuídas por ano: 1 551 notícias em 2012, 1 933 notícias em 2013 e 1 109 em 2014. Como cada notícia esta associada com um canal, foi coletada a distribuição de notícias por canal (Tabela **??**).

Tabela 1 *– Distribuição de notícias por canal (ci) do corpus iG.*

canal (*ci*)	número de notícias
gente	196
ultimosegundo	555
delas	252
economia	907
esporte	342
saude	88
igay	210
deles	141
tecnologia	359
igirl	527
jovem	524
arena	421
luxo	71

Analisando a distribuição de notícias por ano foi possível verificar que os links escolhidos como partida para o *web crawler* realizar a extração de notícias foram efetivos para deixar a distribuição perto de uniforme, com média e desvio padrão de aproximadamente 1531 ± 337 notícias. Contrariamente, a distribuição de notícias por canal não ficou perto do uniforme, com média e desvio padrão de aproximadamente 353 ± 225 notícias.

[1] http://www.ig.com.br/

4.1.2 Base de cliques iG

A base de dados de cliques iG, doada para a realização desta pesquisa, é composta por um conjunto usuários anônimos $U = \{u_1, \ldots, u_l\}$ que foram capturados através do controle de cookies dos navegadores do portal, assim, cada usuário $u \in U$ interage com o conjunto de notícias $\mathcal{N}$ através de cliques, representados por $q^t_{u,n}$, um clique em uma notícia n que foi dado por u em um dado mometo do tempo t. Assim, se considerar cada clique $q^t_{u,n}$ como uma preferência do usuário u por n, é possível construir a matriz de preferências $U \times I$ (Seção 2.1), no contexto, $I = \mathcal{N}$.

Originalmente a base de cliques iG é composta de 487 487 395 cliques, com $m = xxx$ notícias, coletadas, aproximadamente, do período de abril de 2013 à novembro de 2014. Essa base de dados tem tamanho total, sem compressão, de $1TB$, o que dificulta a sua mineração. No entanto, pretende-se usar apenas as notícias que compõem o corpus iG.

4.1.3 Pré-processamento do corpus iG

O corpus iG, para representar as notícicas de maneira estruturada, foi pré-processado usando algumas técnicas de Mineração de Texto (Seção 2.3). Foi criado um *pipeline* para o pré-processamento das notícias que contou com as seguintes etapas:

1. Concatenação das características textuais (título, subtítulo e corpo da notícia) da notícia; normalização do texto, convertendo para minúsculo.

2. Filtragem de trechos do texto que correspondem com uma lista de expressões regulares definidas através da análise do corpus.

3. *Tokenização*, usando espaços, pontuações e expressões regulares como delimitadores.

4. Definição de uma lista de *stopwords*, a partir de uma lista já definida pela biblioteca nltk[2] (*Natural Language Processing Toolkit*) da linguagem python, foram adicionadas mais *stopwords* à essa lista (por exemplo: *leia mais*, *veja aqui* e etc.), através de uma análise empírica das notícias do corpus.

5. *Stemming*, para reduzir ambiguidade e a dimensionalidade das notícias, foi aplicado o algoritmo Removedor de Sufixos da Língua Portuguesa (RSLP) Stemmer (ALVARES; GARCIA; FERRAZ, 2005), que leva em consideração a teoria da língua

[2] http://www.nltk.org/

portuguesa para criar 8 etapas, em que, cada etapa é composta por um conjunto de regras e então aplicada uma regra por vez. Este algoritmo é capaz de tratar e remover formas plurais, femininas/masculinas, adverbiais, aumentativas e diminutivas, substantivas, verbais e acentos.

6. Representação estruturada dos termos em:

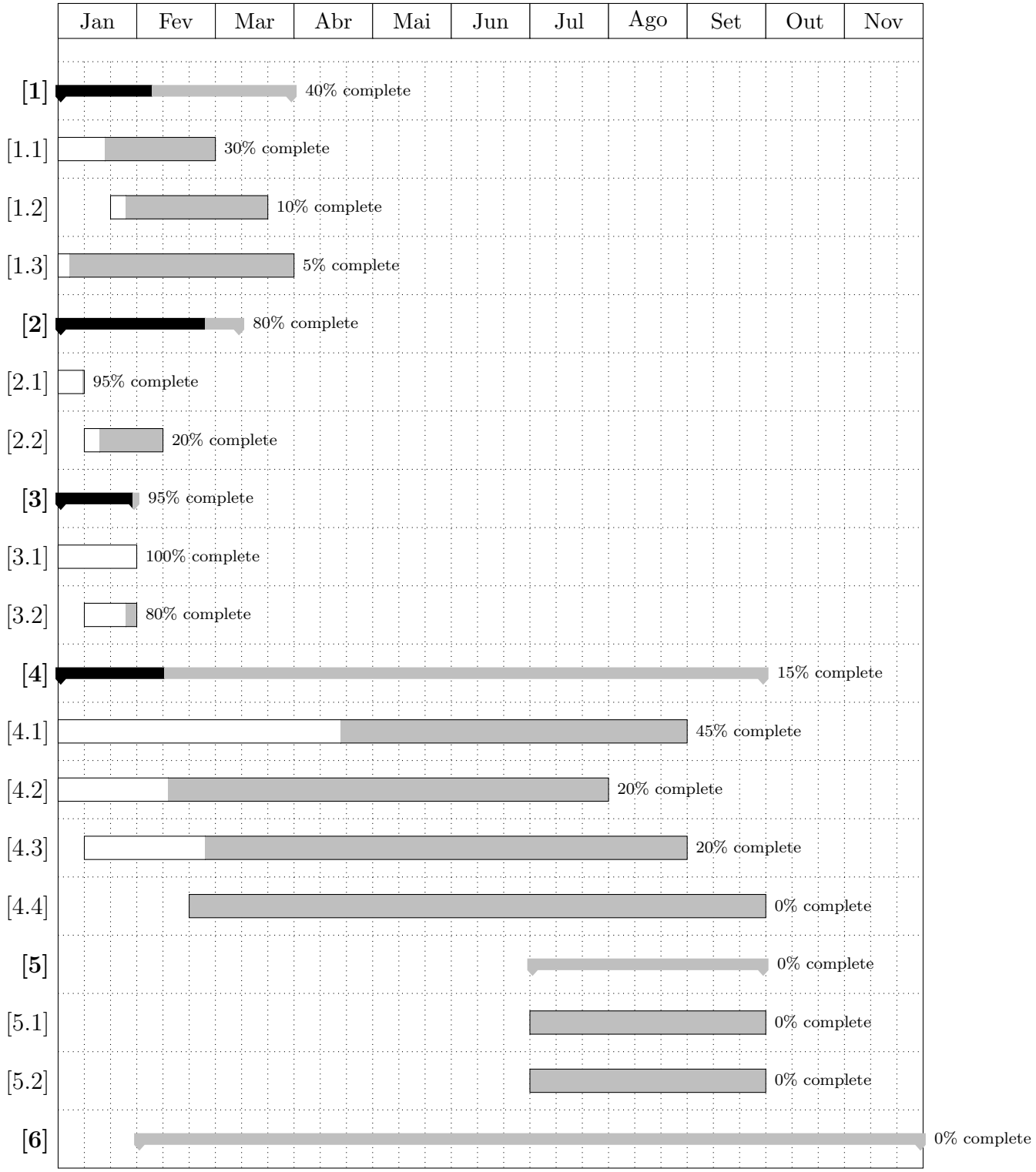
   6.1. TF-IDF, realiza a contagem dos termos e aplica a fórmula de TF-IDF.

   6.2. TF-IDF normalizado, realiza a contagem dos termos, aplica a fórmula de TF-IDF e faz a normalização para que todos os valores fiquem no intervalo de 0 à 1.

   6.3. *2-grams*, realiza a contagem dos termos concatenando 2 a 2.

   6.4. *3-grams*, realiza a contagem dos termos concatenando 3 a 3.

4.2 Próximos passos - cronograma



O cronograma apresentado se refere às atividades a serem desenvolvidas, o progresso indica o que já foi realizado, sendo que as subtarefas já realizadas podem já ter sido completas e não estão indicadas neste cronograma. As tarefas e subtarefas são descritas:

[1] Revisão Sistemática de Sistemas de Recomendação baseados em Conteúdo, compreende ao andamento da RS, restando analisar os trabalhos encontrados de 2005 à 2011. Como a RS está sendo feita por 3 pesquisadores, é demandado mais trabalho e, portanto, mais tempo, no entanto, a concentração de trabalhos nos anos mais recentes é maior, significando que a quantidade de trabalho para a realização dessa tarefa não é proporcinal aos anos de abrangência da revisão.

- [1.1] Aplicação dos critérios de inclusão e exclusão.
- [1.2] Extração dos resultados.
- [1.3] Elaboração de um artigo, com o intuito de publicação na revista *User Modeling and User-Adapted Interaction*[3] (UMUAI).

[2] Estudo de técnicas de Mineração de Texto, como foi necessário o uso de conceitos desta área para a construção do corpus iG, grande parte da literatura necessária já foi levantada e estudada.

- [2.1] Estudo de técnicas de pré-processamento de texto.
- [2.2] Estudo de técnicas de redução de dimensionalidade para texto, esta questão foi pouco estudada, e dada a sua importância, será desprendido mais estudos para a mesma.

[4] Construção de um corpus de notícias e estruturação do mesmo, como o corpus já foi construído, esta tarefa já esta 95% completa.

- [3.1] Implementação de técnicas de pré-processamento de texto.
- [3.2] Implementação de técnicas de representação de texto.

[4] Estudo e aplicação de técnicas de Biclusterização.

- [3.1] Estudo das técnicas de Biclusterização, incluindo algoritmos e formas de avaliação.
- [3.2] Implementação das técnicas e algoritmos de Biclusterização.
- [3.3] Validação das técnicas implementadas em conjuntos de dados sintéticos.
- [3.4] Aplicação dos algoritmos no corpus iG.

[5] Análise dos resultados obtidos pelas estratégias propostas e implementadas.

- [5.1] Avaliação através de medidas internas de biclusterização.

[3] http://www.umuai.org/

[5.2] Avaliação através de medidas externas de biclusterização, como a informação dos canais presentes no corpus iG e a base de dados de cliques iG.

[6] Escrita da dissertação.

Referências

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, v. 17, n. 6, p. 734–749, 2005.

ALVARES, R. V.; GARCIA, A. C. B.; FERRAZ, I. N. Stembr: A stemming algorithm for the brazilian portuguese language. In: BENTO, C.; CARDOSO, A.; DIAS, G. (Ed.). *EPIA*. [S.l.]: Springer, 2005. (Lecture Notes in Computer Science, v. 3808), p. 693–701.

BERRY, M. W.; KOGAN, J. (Ed.). *Text Mining: Applications and Theory*. Chichester, UK: Wiley, 2010.

BURKE, R. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, Kluwer Academic Publishers, Hingham, MA, USA, v. 12, n. 4, p. 331–370, 2002.

BURKE, R. Hybrid web recommender systems. In: *The Adaptive Web: Methods and Strategies of Web Personalization*. Berlin: [s.n.], 2007. cap. 12, p. 377–408.

CABANES, G.; BENNANI, Y.; FRESNEAU, D. Enriched topological learning for cluster detection and visualization. *Neural Networks*, v. 32, p. 186–195, 2012.

CAPELLE, M. et al. Semantics-based news recommendation. In: *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics - WIMS '12*. New York, New York, USA: ACM Press, 2012. p. 1.

CHENG, Y.; CHURCH, G. M. Biclustering of expression data. In: *Procedures of the 8th ISMB*. [S.l.]: AAAI Press, 2000. p. 93–103.

FELDMAN, R.; SANGER, J. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge, MA, USA: Cambridge University Press, 2006. Hardcover.

GETZ, G.; LEVINE, E.; DOMANY, E. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA*, v. 97, p. 12079–12084, 2000.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

HOCHREITER, S. et al. Fabia: factor analysis for bicluster acquisition. *Bioinformatics*, v. 26, n. 12, p. 1520–1527, 2010.

HOTHO, A.; NÜRNBERGER, A.; PAAß, G. A brief survey of text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, v. 20, n. 1, p. 19–62, maio 2005.

JANNACH, D. et al. *Recommender Systems An Introduction*. [S.l.]: Cambridge University Press, 2011.

KITCHENHAM, B. *Procedures for Performing Systematic Reviews*. [S.l.], 2004.

KLUGER, Y. et al. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research*, v. 13, n. 4, p. 703–716, 2003.

LI, L. et al. Personalized news recommendation: A review and an experimental investigation. *Journal of Computer Science and Technology*, v. 26, n. 5, p. 754–766, 2011.

LOPS, P.; GEMMIS, M. de; SEMERARO, G. Content-based recommender systems: State of the art and trends. In: RICCI, F. et al. (Ed.). *Recommender Systems Handbook*. [S.l.]: Springer, 2011. p. 73–105.

MADEIRA, S. C.; OLIVEIRA, A. L. Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 1, p. 24–45, January 2004.

MINER, G. et al. *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*. 1st. ed. [S.l.]: Academic Press, 2012.

MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. [S.l.]: The MIT Press, 2012.

NEUMANN, A. Motivating and supporting user interaction with recommender systems. In: KOVáCS, L.; FUHR, N.; MEGHINI, C. (Ed.). *Research and Advanced Technology for Digital Libraries*. [S.l.]: Springer Berlin Heidelberg, 2007, (Lecture Notes in Computer Science, v. 4675). p. 428–439.

PRELIĆ, A. et al. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, Oxford University Press, Oxford, UK, v. 22, n. 9, p. 1122–1129, maio 2006.

RESNICK, P.; VARIAN, H. R. Recommender systems. *Communications of the ACM*, ACM, New York, NY, USA, v. 40, p. 56–58, March 1997.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to recommender systems handbook. In: RICCI, F. et al. (Ed.). *Recommender Systems Handbook*. [S.l.]: Springer, 2011. p. 1–35.

SALTON, G.; WONG, A.; YANG, C. S. A vector space model for automatic indexing. *Communications of the ACM*, ACM, New York, NY, USA, v. 18, n. 11, p. 613–620, 1975.

SANTAMARíA, R.; MIGUEL, L.; THERóN, R. Methods to bicluster validation and comparison in microarray data. *Lecture Notes in Computer Science: Proceedings of IDEAL'07*, v. 4881, p. 780–789, 2007.

SHANI, G.; GUNAWARDANA, A. Evaluating recommendation systems. In: RICCI, F. et al. (Ed.). *Recommender Systems Handbook*. [S.l.]: Springer, 2011. p. 257–297.

TANAY, A.; SHARAN, R.; SHAMIR, R. Biclustering algorithms: A survey. In: *In Handbook of Computational Molecular Biology Edited by: Aluru S. Chapman & Hall/CRC Computer and Information Science Series*. [S.l.: s.n.], 2005.

WEISS, S. M.; INDURKHYA, N.; ZHANG, T. *Fundamentals of predictive text mining*. London; New York: Springer-Verlag, 2010.

YANG, J.; LESKOVEC, J. Overlapping community detection at scale: A nonnegative matrix factorization approach. In: *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2013. (WSDM '13), p. 587–596.