

LUCAS FERNANDES BRUNIALTI

**Resolução do problema de coagrupamento
em matrizes de dados esparsas usando
fatoração de matrizes**

São Paulo

2016

LUCAS FERNANDES BRUNIALTI

**Resolução do problema de coagrupamento em
matrizes de dados esparsas usando fatoração de
matrizes**

Versão original

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Versão corrigida contendo as alterações solicitadas pela comissão julgadora em xx de xxxxxxxxxxxxxxxxx de xxxx. A versão original encontra-se em acervo reservado na Biblioteca da EACH-USP e na Biblioteca Digital de Teses e Dissertações da USP (BDTD), de acordo com a Resolução CoPGr 6018, de 13 de outubro de 2011.

Orientador: Profa. Dra. Sarajane Marques Peres

Coorientador: Prof. Dr. Valdinei Freire Silva

São Paulo

2016

Ficha catalográfica

Errata

Elemento opcional para versão corrigida, depois de depositada.

Dissertação de autoria de Lucas Fernandes Brunialti, sob o título “**Resolução do problema de coagrupamento em matrizes de dados esparsas usando fatoração de matrizes**”, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Sistemas de Informação, aprovada em ___ de _____ de _____ pela comissão julgadora constituída pelos doutores:

Prof. Dr. _____

Presidente

Instituição: _____

Prof. Dr. _____

Instituição: _____

Prof. Dr. _____

Instituição: _____

Escreva aqui sua dedicatória, se desejar, ou remova esta página...

Agradecimentos

“Escreva aqui uma epígrafe, se desejar, ou remova esta página...”

(Autor da epígrafe)

Resumo

BRUNIALTI, Lucas Fernandes. **Resolução do problema de coagrupamento em matrizes de dados esparsas usando fatoração de matrizes.** 2016. 87 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2016.

Coagrupamento é uma estratégia para análise de dados capaz de encontrar grupos de objetos, então denominados cogrupos, que são similares entre si de acordo com um subconjunto dos seus atributos descritivos, e assim, um objeto pode pertencer a mais de um grupo se subconjuntos diferentes de atributos forem considerados. Essa característica pode ser particularmente útil para aplicações nas quais a similaridade parcial entre objetos faz sentido, conferindo ao resultado da análise de dados algumas características interessantes como serendipidade ou flexibilidade no modelo de agrupamento. Contextos de aplicação caracterizados por apresentar subjetividade, como mineração de textos, são candidatos a serem submetidos à estratégia de coagrupamento; a flexibilidade em associar textos de acordo com características parciais representa um tratamento mais adequado à tal subjetividade. Entretanto, análise de grupos considerando dados textuais representa um contexto no qual existe o problema de esparsidate de dados, que precisa ser adequadamente tratado para que os bons resultados sejam obtidos. Um método para implementação de coagrupamento capaz de lidar com esse tipo de dados é a fatoração de matrizes. Nesta dissertação de mestrado são propostas duas estratégias para coagrupamento baseadas em fatoração de matrizes, e capazes de encontrar cogrupos organizados com sobreposição em uma matriz esparsa de valores reais positivos. As estratégias são apresentadas em termos de sua derivação formal, tendo sido obtida a prova de convergência para uma delas. Ainda, resultados experimentais são fornecidos a partir de problemas baseados em conjuntos de dados sintéticos e em conjuntos de dados reais contextualizados na área de mineração de textos. Os resultados confirmam a hipótese de que as estratégias propostas são capazes de descobrir cogrupos com sobreposição, e que tal organização de cogrupos fornece informação detalhada, e portanto de valor diferenciado.

Palavras-chaves: Coagrupamento. Fatoração de Matrizes. Esparsidate. Análise de Agrupamento. Sistemas de Recomendação. Mineração de Texto.

Abstract

BRUNIALTI, Lucas Fernandes. **Matrix Factorization for coclustering in sparse data matrices.** 2016. 87 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, Defense Year.

Coagrupamento é uma estratégia para análise de dados capaz de encontrar grupos de objetos, então denominados cogrupos, que são similares entre si de acordo com um subconjunto dos seus atributos descritivos, e assim, um objeto pode pertencer a mais de um grupo se subconjuntos diferentes de atributos forem considerados. Essa característica pode ser particularmente útil para aplicações nas quais a similaridade parcial entre objetos faz sentido, conferindo ao resultado da análise de dados algumas características interessantes como serendipidade ou flexibilidade no modelo de agrupamento. Contextos de aplicação caracterizados por apresentar subjetividade, como mineração de textos, são candidatos a serem submetidos à estratégia de coagrupamento; a flexibilidade em associar textos de acordo com características parciais representa um tratamento mais adequado à tal subjetividade. Entretanto, análise de grupos considerando dados textuais representa um contexto no qual existe o problema de esparsidade de dados, que precisa ser adequadamente tratado para que os bons resultados sejam obtidos. Um método para implementação de coagrupamento capaz de lidar com esse tipo de dados é a fatoração de matrizes. Nesta dissertação de mestrado são propostas duas estratégias para coagrupamento baseadas em fatoração de matrizes, e capazes de encontrar cogrupos organizados com sobreposição em uma matriz esparsa de valores reais positivos. As estratégias são apresentadas em termos de sua derivação formal, tendo sido obtida a prova de convergência para uma delas. Ainda, resultados experimentais são fornecidos a partir de problemas baseados em conjuntos de dados sintéticos e em conjuntos de dados reais contextualizados na área de mineração de textos. Os resultados confirmam a hipótese de que as estratégias propostas são capazes de descobrir cogrupos com sobreposição, e que tal organização de cogrupos fornece informação detalhada, e portanto de valor diferenciado.

Keywords: Coclustering. Matrix Factorization. Sparsity. Clustering Analysis. Recommender Systems. Text Mining.

Lista de figuras

Figura 1 – Fatoração da matriz original de dados X em três outras matrizes: U , S e V (Adaptado de Yoo e Choi (2010)).	32
Figura 2 – A reconstrução da primeira linha x_1 de X , através da multiplicação da matriz indicadora de grupos de linhas U pela matriz dos protótipos de linhas (SV^T).	33
Figura 3 – Um exemplo sintético que compara o algoritmo para BVD contra o algoritmo para ONMTF, com pontos sendo dados, as linhas pontilhadas sendo os protótipos de linhas (SV^T), e os três conjuntos de pontos sendo os grupos. (a) O algoritmo para BVD encontra uma solução com dois protótipos em um mesmo grupo, deixando um dos grupos sem nenhum protótipo para representá-lo. Apesar de ser uma solução correta, ou seja, encontra um mínimo local, não é a desejada. (b) O algoritmo para ONMTF é capaz de encontrar a solução em que cada protótipo aproxima cada grupo de dados, através das restrições referentes à ortogonalidade, este é capaz de restringir as possíveis soluções para a fatoração $X \approx USV^T$. (YOO; CHOI, 2010)	36
Figura 4 – Uma Variedade Stiefel no espaço Euclidiano, que quando $b = 1$ essa superfície será uma esfera, e um dos possíveis vetores que está contido no conjunto dessa variedade (tangente à esfera).	38
Figura 5 – Dados sintéticos gerados a partir das diferentes estruturas de biclusters. (a) Um único bicluster. (b) Biclusters com linhas e colunas sem intersecção. (c) Biclusters com estrutura em xadrez. (d) Biclusters com linhas sem intersecção e colunas com intersecção. (e) Biclusters com linhas com intersecção e colunas sem intersecção.	45
Figura 6 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo k -means.	48
Figura 7 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo <i>fuzzy k-means</i>	49
Figura 8 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo <i>ONMTF</i>	50

Figura 9 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo <i>FNMTF</i>	51
Figura 10 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo <i>OvNMTF</i>	52
Figura 11 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo <i>BinOvNMTF</i>	53
Figura 12 – Exemplo de uma notícia do canal arena e sua disposição quanto aos coclusters de notícias e palavras.	61
Figura 13 – Visualização <i>word cloud</i> de palavras para cada cocluster de palavras gerados pelo algoritmo <i>ONMTF</i>	62
Figura 14 – Visualização <i>word cloud</i> de palavras para cada cocluster de palavras do cocluster de notícias “arena”, gerados pelo algoritmo <i>OvNMTF</i>	63
Figura 15 – Visualização <i>word cloud</i> de palavras para cada cocluster de palavras do cocluster de notícias “esporte”, gerados pelo algoritmo <i>OvNMTF</i>	63
Figura 16 – Visualização <i>word cloud</i> de palavras para cada cocluster de palavras do cocluster de notícias “jovem”, gerados pelo algoritmo <i>OvNMTF</i>	63
Figura 17 – Arquitetura de um CBRS	69
Figura 18 – Analisador de Contexto.	69
Figura 19 – Sumarização da proposta.	75

Lista de algoritmos

Algoritmo 1 – Algoritmo baseado em atualização multiplicativa para solução do BVD . . .	35
Algoritmo 2 – Algoritmo baseado em atualização multiplicativa para solução do ONMTF . . .	36
Algoritmo 3 – Algoritmo FNMTF	40
Algoritmo 4 – Algoritmo baseado em atualização multiplicativa para solução do OvNMTF . .	41
Algoritmo 5 – Algoritmo BinOvNMTF	43

Lista de tabelas

Tabela 1 – Estatísticas das bases de dados usadas nos experimentos.	55
Tabela 2 – Estatísticas das bases de dados usadas nos experimentos.	56
Tabela 3 – Matriz S para $ONMTF$ com $k = l = 3$.	62
Tabela 4 – Matriz S para $OvNMTF$ com $k = 3$ e $l = 6$.	63
Tabela 5 – Distribuição de notícias por canal (ci) do corpus iG.	77

Lista de abreviaturas e siglas

- Sigla/abreviatura 1 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 2 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 3 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 4 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 5 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 6 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 7 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 8 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 9 Definição da sigla ou da abreviatura por extenso
- Sigla/abreviatura 10 Definição da sigla ou da abreviatura por extenso

Lista de símbolos

Γ Letra grega Gama

Λ Lambda

ζ Letra grega minúscula zeta

\in Pertence

Sumário

1	Introdução	19
1.1	Definição do problema	21
1.1.1	Estruturas de coagrupamentos	22
1.1.2	Coagrupamento e fatorização de matrizes	22
1.2	Hipótese	23
1.3	Objetivos	23
1.4	Metodologia	24
1.5	Organização do documento	25
2	Conceitos Fundamentais	26
2.1	Tipos de biclusters	26
2.2	Algoritmos para Biclusterização	27
2.3	Avaliação de Biclusterização	29
3	Fatoração de matrizes	31
3.1	Non-negative Matrix Factorization	31
3.2	Fatoração de Matrizes Não-Negativas para Coclustering	31
3.2.1	Decomposição de Valores em Blocos para Coclustering .	32
3.2.2	Fatoração Ortogonal tripla de Matrizes Não-negativas .	34
3.3	Fatoração tripla rápida de Matrizes Não-negativas .	39
4	Algoritmos propostos	41
4.1	Fatoração Tripla de Matrizes Não-negativas Sobrepostas	41
4.2	Fatoração Binária Tripla de Matrizes Não-negativas Sobrepostas	42
5	Experimentos e Resultados	44
5.1	Experimentos com Base de Dados sintéticas	44
5.1.1	Experimentos de reconstrução	47

5.1.2	Resultados da reconstrução de dados sintéticos com <i>k-means</i>	47
5.1.3	Resultados da reconstrução de dados sintéticos com <i>fuzzy k-means</i>	48
5.1.4	Resultados da reconstrução de dados sintéticos com <i>ONMTF</i>	49
5.1.5	Resultados da reconstrução de dados sintéticos com <i>FNMTF</i>	50
5.1.6	Resultados da reconstrução de dados sintéticos com <i>OvNMTF</i>	51
5.1.7	Resultados da reconstrução de dados sintéticos com <i>BinOvNMTF</i>	52
5.1.8	Resultados quantitativos com dados sintéticos	53
5.2	Experimentos com base de dados reais	55
5.2.1	Descrição das bases de dados	55
5.2.2	Pré-processamento	57
5.2.3	Experimentos quantitativos	59
5.2.3.1	Setup dos algoritmos	59
5.2.3.2	Setup dos experimentos	59
5.2.3.3	Base de dados <i>NIPS</i>	59
5.2.3.4	Base de dados <i>20Newsgroup</i>	59
5.2.3.5	Base de dados <i>IG</i>	59
5.2.3.6	Base de dados <i>IG toy</i>	59
5.2.4	Experimentos qualitativos	59
5.2.4.1	Análise de dados utilizando <i>ONMTF</i>	60
5.2.4.2	Análise de dados utilizando <i>FNMTF</i>	63
5.2.4.3	Análise de dados utilizando <i>OvNMTF</i>	63
5.2.4.4	Análise de dados utilizando <i>BinOvNMTF</i>	63
6	Mineração de Texto	64
6.1	Tarefas de pré-processamento	64
6.1.1	Representação textual	64
6.1.2	Tokenização	65
6.1.3	Filtragem	66
6.1.4	Stemming	66
6.1.5	Redução de Dimensionalidade	66

7	Sistemas de Recomendação baseados em Conteúdo e Aprendizado de Máquina	68
7.1	Arquitetura de Sistemas de Recomendação baseados em Conteúdo	68
7.1.1	Analisador de Contexto	69
7.1.2	Aprendizado do Perfil	71
7.1.3	Componente de Filtragem	72
8	Proposta	74
8.1	Revisões Bibliográficas	75
8.2	Construção das Bases de Dados	77
8.2.1	Corpus iG	77
8.2.2	Pré-processamento do corpus iG	78
8.2.3	Base de cliques iG	78
8.3	Estratégias para Validação e Extensões	79
8.3.1	Estratégias para Validação	79
8.3.2	Possíveis Extensões	79
Referências¹		81
Apêndice A – Tabela Workshop		86

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

Segundo Jain, Murty e Flynn (1999), a análise de agrupamento pode ser vista como uma tarefa exploratória que tem o objetivo de organizar uma coleção de dados em um conjunto de grupos, segundo a similaridade ou dissimilaridade existente entre esses dados. Tradicionalmente, os métodos usados para análise de agrupamento são desenvolvidas para minimizar a similaridade intragrupo e maximizar a similaridade intergrupos; e precisam encontrar uma organização “natural” em grupos que acomode cada dado do conjunto sob análise em um grupo específico.

Estratégias de diferentes naturezas – particional, hierárquica, baseada em densidade, etc (XU; WUNSCH, 2005; HAN; KAMBER, 2006), podem ser usadas para alcançar o objetivo da análise de agrupamento, e cada uma delas possui características que as fazem mais ou menos suscetíveis para conjuntos de dados de diferentes naturezas. Ainda, sob o contexto clássico da tarefa de agrupamento, os métodos precisam lidar com a similaridade intrínseca entre os dados tomando como base a comparação de todas as suas características descritivas e, de alguma forma, precisam ser capazes de descobrir quais características de fato tornam dados em um grupo de dados mais similares entre si.

Ao longo do tempo, pesquisadores da área de análise de agrupamento vêm propondo flexibilizações na definição da tarefa de agrupamento de forma a adequá-la a contextos mais realísticos nos quais a organização natural dos dados em um conjunto de dados não pressupõe restrições como a pertinência de um dado a um único grupo ou a possibilidade de agrupar dados de acordo com similaridades em subconjuntos de atributos descritivos ([referência](#), [referência](#), [referência](#)). Essa forma de tratar a tarefa de agrupamento permite melhorias no processo de descoberta de agrupamentos sobre dois aspectos: facilita o trabalho do método que busca os grupos, pois flexibiliza a maneira como os atributos descritivos dos dados ou a pertinência do dado aos grupos influencia o processo de agrupamento; fornece um conjunto de informações diferenciado que permite que análises mais refinadas sejam realizadas quando da interpretação dos grupos apresentados como resultado.

Esse diferencial pode ser especialmente útil quando o contexto da aplicação da análise de agrupamento apresenta alguma subjetividade em termos de interpretação de resultados, um fato bastante comum em tarefas de mineração de textos, por exemplo. Considere o contexto de um sistema recomendação (SR) de notícias baseado em conteúdo (um conteúdo textual). Um SR de notícias simples e hipotético poderia apresentar a seguinte

estratégia para elaborar suas recomendações: dado um conjunto de notícias organizadas em grupos por um método de análise de agrupamento com base na similaridade de seus conteúdos; se um usuário visitar uma notícia, o SR verifica quais são as demais notícias pertencentes ao grupo daquela que foi lida pelo usuário e as recomenda para ele (Figura ??a). Embora essas recomendações pareçam ser ideais, e sob algum aspecto de análise elas são, é factível assumir que esse usuário está recebendo um serviço de recomendação prático, mas talvez menos útil e interessante do que poderia ser e com baixa serendipidade (um resultado de alta serendipidade é aquele que é diferente do que é esperado e comumente praticado, e é mais ou igualmente útil ao contexto).

Uma possibilidade de melhoria nesse sistema hipotético seria usar um algoritmo de análise de agrupamento que permitisse descobrir uma organização de grupo de notícias baseada em **similaridades parciais** ou baseada **em partes** (FRANCA, 2010; HO, 2008). Assim o sistema seria dotado da capacidade de perceber, por exemplo, que algumas notícias podem trazer conteúdo referente a diferentes contextos se forem analisadas apenas sob determinados aspectos. Nesse caso, os grupos formados durante a análise de agrupamento seriam capazes de refletir a diversidade de contextos abordados em uma notícia, fazendo-a pertencer a diferentes grupos, por diferentes motivos. A recomendação, nesse caso, seria potencialmente mais serendípita. Por exemplo, é sabido que eventos de beisebol – o *superbowl* – possuem uma abertura cultural na qual grandes artistas da música fazem apresentações; ou eventos de esportes radicais, como tirolesa, acontecem em eventos de música contemporâneos – *rock in rio*. Tais notícias deveriam aparecer em grupos caracterizados por notícias referentes a esporte, notícias referentes a música, ou notícias referentes a esporte e música (Figura ??(b, c, d)).

Na figura ??(b,c,d) é introduzido graficamente o conceito de coagrupamento. Nesse contexto, o problema de mineração de textos é modelado como o problema de encontrar uma organização dos textos em grupos que considerem similaridades parciais. Assim, um texto pode pertencer a um ou mais cogrupos, a depender dos atributos descritivos sendo considerados. A nomenclatura coagrupamento deriva da estratégia de análise de dados executada durante o processo de descoberta de grupos. Nesse caso, tanto os dados (linhas) quanto os seus atributos (descritivos) são mutuamente submetidos a uma análise de similaridade, e portanto, grupos de dados (linhas) são estabelecidos com respeito a grupos de atributos (colunas).

A associação da análise de coagrupamentos a mineração de textos é interessante por diferentes aspectos. A mineração de textos constitui-se como um problema no qual é preciso lidar com a necessidade de apresentação de resultados com boa interpretabilidade e com um espaço dos dados de alta-dimensionalidade. O primeiro problema é bem resolvido com a estratégia de coagrupamento pois os grupos de atributos que são gerados por ela podem revelar informação antes escondida nos dados (TJHI; CHEN, 2009), e que em um processo de agrupamento tradicional não poderiam ser, pelo menos diretamente, descobertas. Ainda segundo (TJHI; CHEN, 2009), análise de coagrupamento pode apresentar bom desempenho em espaços de alta-dimensionalidade porque seu processo de agrupar atributos (características) pode ser visto com uma redução de dimensionalidade dinâmica para o espaço dos dados.

A despeito da capacidade intrínseca do processo de coagrupamento em lidar de forma diferenciada com o problema de alta-dimensionalidade, ainda se faz necessário notar que no contexto de mineração de dados, ocorre também o problema de esparsidade na representação dos dados. Assim, para implementar a estratégia de coagrupamento com alguma eficiência, é necessário adotar métodos que tenham a capacidade de lidar com esparsidade.

Dentre os diferentes métodos existentes na literatura referentes à implementação de análise de coagrupamento (citar artigos dos vários algoritmos que seguem outras linhas), métodos que usam fatoração de matrizes não negativas (LEE; SEUNG, 2000; LEE; SEUNG, 1999) têm sido vistos como uma boa alternativa a ser aplicada no contexto de mineração de textos (XU; LIU; GONG, 2003; SHAHNAZ et al., 2006; YOO; CHOI, 2010).

1.1 Definição do problema

Eu estou entendendo que temos duas facetas do problema. Um deles é o mais óbvio que é dar um jeito de descobrir os grupos com sobreposição e mostrar que é útil para recomendação/textos etc. O outro é fazer a fatoração de matriz funcionar pra isso. Então acho que temos que dividir essa definição em duas partes: sobreposição nos cogerupos e fatoração funcionando nisso. Por isso dividi em duas partes, para ver se conseguimos mostrar isso.

1.1.1 Estruturas de coagrupamentos

Então aqui entra a parte de mostrar as estruturas de cogrupos possíveis e destacar aquela que fatoração já resolve e depois a que não resolve e a que queremos resolver. Também contextualizar no problema de recomendação ou análise de textos. Figuras precisam entrar aqui para mostra as estruturas.

1.1.2 Coagrupamento e fatorização de matrizes

A estratégia de coagrupamento pode ser apresentada como o processo de agrupamento simultâneo de linhas e colunas em uma matriz de dados, de forma que seja possível encontrar **cogrupos** nos quais um **grupo de objetos** (linhas) associado a um deles diz respeitos a objetos que são similares entre si considerando um **grupo de atributos** (colunas), também associado ao cogrupo.

Com maior formalidade, dada uma matriz $X(N, M)$ em que N é o número de linhas, M o número de colunas e $x(m, n)$ é, geralmente, um número real representando a relação entre a linha x_n e a coluna y_m , o problema de **coagrupamento** consiste em encontrar um conjunto \mathcal{C} de submatrizes $G(I, J)$, onde $I = \{i_1, \dots, i_r\}$ com $r \leq L$ e $J = \{j_1, \dots, j_s\}$ com $s \leq C$, que maximize a similaridade entre os elementos $g\{i, j\}$.

A **esparsidade** em uma matriz é caracterizada pela existência de poucos elementos diferentes de zero (0). Em termos gerais, a esparsidade de uma matriz pode ser medida como a proporção de elementos iguais a zero (0) que ela contém. Problemas de otimização que envolvem matrizes esparsas são caracterizados por apresentarem alta complexidade combinatorial para os quais algoritmos eficientes em matrizes não esparsas tem seu desempenho bastante prejudicado.

Fatorar uma matriz consiste em encontrar duas, ou mais, novas matrizes que ao serem multiplicadas, reconstroem a matriz original. Considere uma matriz $R(N, M)$ em que N é o número de linhas, M o número de colunas. A fatoração desta matriz em duas novas matrizes consiste em encontrar duas matrizes $U(N, K)$ e $D(M, K)$, tal que $R = U \times D^t = \hat{R}$. Se K é escolhido tal que seja menor do que N e M , então é dito que U e D são representações compactas de R . Se a matriz R , e as suas decomposições, são não negativas, tem-se o caso de fatorização de matriz não-negativa.

O problema de coagrupamento pode ser modelado de tal forma que a fatoração de matriz é capaz de fornecer uma aproximação da organização em cogerupos presente no conjunto de dados sob análise.

Considere que o conjunto de dados sob análise é representado pela matriz $X(N, M)$, a fatoração dessa matriz em duas (ou mais) novas matrizes $U(N, K)$ e $D(M, K)$ significa que K grupos de linhas foram descobertos, de acordo com K grupos de colunas.

Se três matrizes são geradas na fatoração, $U(N, L)$, $S(L, K)$ e $D(M, K)$, a interpretação pode incluir uma noção de pesos (matriz S) que relacionam grupos de linhas e grupos de colunas, e dimensões diferentes para as matrizes U e D podem ser admitidas de modo que o número de grupos de linhas pode ser diferente do número de grupo de colunas.

Imagino que agora tem que entrar uma apresentação rápida do algoritmo novo.

1.2 Hipótese

Fatoração de matrizes considerando a decomposição da matriz original em ... como descrever em algo nível aqui?? ... possibilita a descoberta de cogerupos com sobreposição (de colunas); a partir das novas matrizes é possível extrair informação detalhada sobre a relação dos grupos de linhas em relação ao grupo de colunas que pode agregar valor à solução de um problema real de recomendação.

1.3 Objetivos

O objetivo geral desse trabalho é propor novas estratégias, baseadas em fatoração de matrizes, que sejam capazes de descobrir cogerupos com sobreposição (de coluna) em uma matriz de valores reais positivos.

Assim, com o intuito de melhor definir o contexto de estudo deste trabalho, foram estabelecidos os seguintes objetivos específicos:

- apresentação de uma derivação formal, incluindo uma prova de convergência, para a estratégia de coagrupamentos usando fatoração de matrizes não-negativas capaz de lidar com matrizes binárias;

- apresentação de uma derivação formal para a estratégia de coagrupamento usando atoração de matrizes não negativas capaz de lidar com matrizes de valores reais positivos;
- apresentar experimentos que ilustram a eficácia das estratégias propostas considerando dados sintéticos e dados reais, mostrando inclusive o potencial de descoberta de informações de valor diferenciado para as aplicações em teste (recomendação baseada em conteúdo textual).

1.4 Metodologia

A análise exploratória da literatura especializada foi escolhida como estratégia para a aquisição de conhecimento sobre a área de coagrupamento e fatoração de Matrizes aplicada à coagrupamento.

E não estou conseguindo encontrar uma forma de descrever a parte referente à concepção das estratégias propostas e também não sei como definir as estratégias referente às derivações.

A fim de permitir a validação das estratégias propostas e, portanto, a verificação da hipótese, fez-se necessário a definição de: (a) um ambiente de teste controlado, representado por uma coleção de conjuntos de dados sintéticos, contendo cada um dos conjuntos situações diferentes referentes à estrutura de coagrupamento e variações em relação à esparsidade; (b) um contexto para realização de uma prova de conceito, no qual um conjunto de dados real foi construído.

Para a prova de conceito foi escolhido usar o conteúdo referente à notícias publicadas no portal iG¹. Trata-se de um portal de notícias brasileiro muito conhecido, com um volume de notícias bastante grande e com notícias categorizadas em canais, que representam os assuntos dessas notícias. Essas características conferem liberdade para a configuração de experimentos de diferentes naturezas, como experimentos considerando determinadas categorias de notícias, tipos de notícias ou datas de publicação das notícias.

A partir do conteúdo de notícias do portal iG foi construído um corpus de dados textuais, categorizados de acordo com as categorias já usadas no referido portal. Todo o conteúdo do corpus passou por rotinas de pré-processamento comuns na área de Mineração de Texto: *tokenização*, filtragem de *stopwords*, remoção de sufixos (*stemming*), representação

¹ <http://ig.com.br/>

da relação “termos × documentos” usando estratégias de frequência de termos, como TF-IDF e *n-grams*.

Os resultados da aplicação das estratégias de coagrupamento foram validados utilizando técnicas de avaliação interna, para a verificação da consistência dos biclusters encontrados (SANTAMARÍA; MIGUEL; THERÓN, 2007), e externas (HOCHREITER et al., 2010), avaliando o quanto os biclusters encontrados estão em consenso com as classes de notícias (HOCHREITER et al., 2010).

Então precisaremos dizer aqui como fizemos a avaliação qualitativa.

1.5 Organização do documento

Esta dissertação é composta por **XXX** capítulos incluindo esta introdução. Os demais capítulos estão divididos em duas partes: a primeira é dedicada a explorar a estratégia de coagrupamento implementada com algoritmos baseadas em fatoração de matrizes; a segunda é dedicada a explorar o contexto de sistemas de recomendação baseados em conteúdo textual a partir da aplicação das estratégias de coagrupamento estudadas.

No capítulo 2 são apresentados os principais conceitos referentes à área de coagrupamentos. [Especificar mais detalhes](#)

Estratégias de fatorização de matrizes aplicadas à coagrupamentos são discutidas no capítulo

A principal contribuição deste trabalho, as estratégias, é apresentada em detalhes no capítulo

2 Conceitos Fundamentais

Técnicas e algoritmos de Biclusterização são usadas, principalmente, no contexto de expressão genética. No entanto, algoritmos de Biclusterização se fazem úteis 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 ([FRANCA, 2010; MADEIRA; OLIVEIRA, 2004](#)).

Para a descrição do problema formal de Biclusterização usa-se a seguinte definição ([MADEIRA; OLIVEIRA, 2004](#)): seja uma matriz A , de dimensão $N \times M$, um conjunto de linhas $X = \{x_1, \dots, x_n, \dots, x_N\}$ e um conjunto de colunas $Y = \{y_1, \dots, y_m, \dots, y_M\}$, em que a_{nm} geralmente é um número real e representa a relação entre a linha x_n e a coluna y_m ; 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 objetos em I , perante as características com alta correlação J .

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 valores 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 diferem quanto as suas estruturas. Cada algoritmo usado para implementar Biclusterização faz uma suposição da estrutura de biclusters que é capaz de encontrar. A Figura ?? 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 , sendo os biclusters A_{IJ} representados por cores sólidas e o fundo da matriz ruído.

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 algoritmos de Biclusterização mais comum e simples, que encontra biclusters com valores coerentes, em estrutura com sobreposição e arbitrariamente posicionados, é o *Coupled Two-way Clustering* (CTWC) (GETZ; LEVINE; DOMANY, 2000). O algoritmo CTWC é capaz de encontrar biclusters através da clusterização de objetos e atributos (linhas e colunas), separadamente. O algoritmo de clusterização usado por Getz, Levine e Domany (2000) foi o *Superparamagnetic Clustering* (SPC), o qual 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 gerará. Assim, o CTWC encontra clusters estáveis de linhas e colunas através do SPC, e iterativamente executa o SPC 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.

Já o algoritmo de [Cheng e Church \(2000\)](#) é capaz de encontrar o mesmo tipo de bicluster que o algoritmo CTWC, porém usando uma estratégia gulosa: biclusters com valores coerentes e estrutura com sobreposição e arbitrariamente posicionados. Este algoritmo está sendo objeto de estudo desse projeto de mestrado para aplicação em dados textuais e por isso segue aqui descrito em mais detalhes. Nesse algoritmos, para encontrar biclusters, ou δ -biclusters, na matriz A , os autores definem o *Resíduo Quadrático Médio* (RQM):

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

em que

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}, \quad a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}, \quad a_{IJ} = \frac{1}{|I||J|} \sum_{i,j \in I,J} a_{ij} \quad (2)$$

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 perfeito A_{IJ} teria o RQM $H_{IJ} = 0$, pois $a_{ij} = a_{iJ} = a_{IJ} = a_{Ij}$, $\forall i, j \in I, J$, 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 \(2000\)](#) usam 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.

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 -bicusters perfeitos com apenas um elemento, ou poucos elementos, é utilizado o algoritmo 3, que adiciona nós sem aumentar o RQM do bicluster.

Por fim, o algoritmo 4 é a consolidação dos algoritmos 3, 2 e 1 e a iteração para encontrar k δ -bicusters, um a um, sendo k fornecido pelo usuário.

Além dos algoritmos apresentados, existem outros algoritmos que são capazes de encontrar outros tipos de biclusters (Seção 2.1), além de serem recentes ([FRANÇA; ZUBEN, 2010](#); [YANG; LESKOVEC, 2013](#); [HOCHREITER et al., 2010](#); [CABANES; BENNANI; FRESNEAU, 2012](#)), mostrando que ainda há interesse na área de pesquisa de Biclusterização.

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 homogeneidade, [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:

$$\begin{aligned} C_l(A_{IJ}) &= \frac{1}{|I|} \sum_{i=1}^{|I|-1} \sum_{j=i+1}^{|I|} \sqrt{\sum_{k=1}^{|J|} (a_{ik} - a_{jk})^2} \\ C_c(A_{IJ}) &= \frac{1}{|J|} \sum_{i=1}^{|J|-1} \sum_{j=i+1}^{|J|} \sqrt{\sum_{k=1}^{|I|} (a_{ki} - a_{kj})^2} \end{aligned} \quad (3)$$

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:

$$C(A_{IJ}) = \frac{|I| \cdot C_l + |J| \cdot C_c}{|I| + |J|} \quad (4)$$

Uma das métricas externas que são usadas para comparar biclusters encontrados com biclusters reais em um conjunto de dados, é a métrica *consensus score* (HOCHREITER et al., 2010). Essa métrica calcula a 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. A saída da avaliação é um *score* $\in [0, 1]$, em que 0 significa que os biclusters comparados são totalmente diferentes, e 1 o inverso.

3 Fatoração de matrizes

3.1 Non-negative Matrix Factorization

3.2 Fatoração de Matrizes Não-Negativas para Coclustering

Algoritmos de coclustering baseados em Fatoração de Matrizes Não-negativas (*Non-negative Matrix Factorization* - NMF) podem ser úteis em múltiplos contextos, aparecendo em diversos tipos de aplicações, como clusterização de genes e análise de microarray em bioinformática, filtragem colaborativa em sistemas de recomendação, e clusterização de documentos em mineração de textos. Isso acontece pois muitas das representações usadas nessas aplicações se apresentam em tuplas contendo um par de elementos, cada um pertencendo a um conjunto finito (LONG; ZHANG; YU, 2005). Por exemplo, na aplicação de clusterização de documentos em mineração de textos, usa-se, comumente, dois conjuntos: documentos e palavras, sendo cada observação representada pela contagem de uma determinada palavra em um determinado documento. Note ainda, que conjuntos de dados utilizados nos contextos citados, apresentam-se como uma matriz de dados positiva, expondo outra característica em que NMF podem ser úteis para esses tipos de dados. Além disso, essas técnicas, por reduzir a dimensionalidade original do conjunto de dados, são capazes de lidar com dados de alta dimensionalidade e/ou esparsos, que são geralmente presentes nessas aplicações.

Formalizando, algoritmos de Coclusterização baseados em Fatoração de Matrizes Não-negativas, têm como entrada uma matriz de dados X que representa uma aplicação em algum contexto, sendo $X \in \mathbb{R}_+^{n \times m}$, contendo números reais positivos com n linhas e m colunas. Esta matriz é formada por um conjunto de vetores de linhas $\mathcal{X} = \{x_1, \dots, x_n\}$ e um conjunto de vetores de colunas $\mathcal{Y} = \{y_1, \dots, y_m\}$, e cada observação representada por x_{ij} , que é justamente um valor da matriz X .

O objetivo é particionar \mathcal{X} em k grupos de linhas, denotados pelos subconjuntos ordenados $I_p \subseteq \mathcal{X}$, e l grupos de colunas, subconjuntos $J_q \subseteq \mathcal{Y}$, podendo assim, pela junção desses subconjuntos, formar cogrupos (I_p, J_q) . Note então, que os algoritmos tradicionais presentes na literatura são capazes de formar kl cogrupos.

As próximas subseções irão mostrar definições dos algoritmos de coclusterização baseados em NMF presentes na literatura, assim como estratégias usadas para selecionar os cogrupos mais relevantes dos kl possíveis.

3.2.1 Decomposição de Valores em Blocos para Coclustering

Um dos primeiros algoritmos propostos na literatura para a resolução do problema de *coclustering* baseado em NMF, é o Decomposição de Valores em Blocos (*Block Value Decomposition* - BVD). Esta decomposição recebe esse nome, justamente, por ter capacidade de encontrar estruturas em blocos na matriz de dados. Isso é possível pois o algoritmo é capaz de explorar a relação entre linhas e colunas, através da decomposição em três matrizes, U a matriz de coeficientes de linhas, S a matriz com estrutura em blocos, e V a matriz de coeficientes de colunas, sendo os coeficientes o grau com que cada vetor (linha ou coluna) de X pertence a um grupo, e a estrutura em blocos, uma representação compacta da matriz original. Sendo assim, multiplicando as três matrizes geradas pela fatoração, é possível reconstruir a matriz original X por partes.

O objetivo é encontrar grupos de forma simultânea, sendo k grupos de \mathcal{X} (linhas) e l grupos de \mathcal{Y} (colunas) ([LONG; ZHANG; YU, 2005](#)).

A aproximação $X \approx USV^T$ é capaz de gerar diversas informações interpretáveis através da análise do resultado da fatoração. Os valores da matriz U podem ser interpretados como o grau com que cada objeto x_i . $\forall i = \{1, \dots, n\}$ pertence à um dos k grupos de linhas, da mesma forma, os valores da matriz V podem ser interpretados como o grau com que cada objeto y_j . $\forall j = \{1, \dots, m\}$ pertence à um dos l grupos de colunas. A matriz S também permite interpretação quando são considerados os grupos formados por U e V , cada valor s_{pq} $\forall p = \{1, \dots, k\}, \forall q = \{1, \dots, l\}$ representa a intensidade em que o grupo de objetos J_p e o grupo de características J_q se relacionam, ou seja, fixando um grupo de objetos I_p , s_{pq} representa a intensidade que J_q colaborou para a sua formação, alternadamente, fixando um grupo de características J_q , s_{pq} representa a intensidade na qual o grupo de objetos I_p colaborou para a sua formação.

Um exemplo de fatoração em três matrizes e sua interpretação pode ser visto na Figura 1.

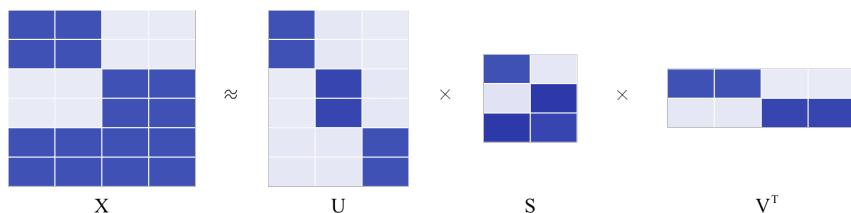


Figura 1 – Fatoração da matriz original de dados X em três outras matrizes: U , S e V (Adaptado de [Yoo e Choi \(2010\)](#)).

Note ainda, que é possível obter os protótipos responsáveis por cada parte da reconstrução da matriz original X . Os vetores das colunas de (US) são vetores base (protótipos de colunas), geradores de vetores de colunas de X . Assim como os vetores de linhas de (SV^T) , que são vetores base (protótipo de linhas), geradores de vetores e linhas de X . A Figura 2 mostra um exemplo de como os protótipos são responsáveis pela reconstrução de X .

The diagram illustrates the matrix factorization process. It starts with a matrix X (3x5 grid) which is approximately equal to the product of matrices U (3x3 grid), S (3x3 grid), and V^T (3x5 grid). This is shown as $X \approx U \times \begin{pmatrix} S \\ \times \\ V^T \end{pmatrix}$. Below this, it shows the decomposition $= U \times S \times V^T$. Finally, it shows the reconstruction $\approx U(SV^T)$ (3x5 grid).

Figura 2 – A reconstrução da primeira linha x_1 de X , através da multiplicação da matriz indicadora de grupos de linhas U pela matriz dos protótipos de linhas (SV^T) .

Apresentando o problema formalmente, temos:

Problema 1 (Problema de Decomposição de Valores em Blocos).

$$\begin{aligned} \mathcal{F}_1(U, S, V) &= \min_{U, S, V} \|X - USV^T\|_F^2 \\ &\text{suj. a} \quad U \geq 0, \\ &\quad S \geq 0, \\ &\quad V \geq 0 \end{aligned} \tag{5}$$

onde $U \in \mathbb{R}_+^{n \times k}$, $S \in \mathbb{R}_+^{k \times l}$, $V \in \mathbb{R}^{m \times l}$ e $\|\cdot\|_F$ denota a norma de Frobenius para matrizes.

Introduzindo a função lagrangeana, associada à \mathcal{F}_1 :

$$\mathcal{L}(U, S, V, \Lambda_1, \Lambda_2, \Lambda_3) = \|X - USV^T\|_F^2 - \text{tr}(\Lambda_1 U^T) - \text{tr}(\Lambda_2 S^T) - \text{tr}(\Lambda_3 V^T)$$

onde $\Lambda_1 \in \mathbb{R}^{n \times k}$, $\Lambda_2 \in \mathbb{R}^{k \times l}$ e $\Lambda_3 \in \mathbb{R}^{m \times l}$ são os multiplicadores de Lagrange.

Pela teoria de otimização não-linear com restrições, $\Theta = (U^*, S^*, V^*, \Lambda_1^*, \Lambda_2^*, \Lambda_3^*)$ será um mínimo local estacionário de \mathcal{F}_1 , se e somente se, respeitar as condições de regularidade de *Karush-Kuhn-Tucker* (KKT) ([BAZARAA; SHERALI; SHETTY, 2006](#)):

$$U^* \geq 0, \quad S^* \geq 0, \quad V^* \geq 0 \quad (6a)$$

$$\nabla_{\Theta} \mathcal{L} = 0 \quad (6b)$$

$$\Lambda_1^* \odot U^* = 0, \quad \Lambda_2^* \odot S^* = 0, \quad \Lambda_3^* \odot V^* = 0 \quad (6c)$$

onde \odot denota o produto de Hadamard.

É possível expandir as equações em [6b](#), calculando as derivadas:

$$\begin{aligned} \nabla_U \mathcal{L} &= USV^T VS^T - XV^T S^T - \Lambda_1 = 0 \\ \nabla_S \mathcal{L} &= U^T USV^T V - U^T XV - \Lambda_2 = 0 \\ \nabla_V \mathcal{L} &= S^T U^T USV^T - S^T U^T X - \Lambda_3 = 0 \end{aligned}$$

Aplicando o produto Hadamard dos dois lados de cada uma das equações e utilizando das condições em [6c](#):

$$\begin{aligned} \nabla_U \mathcal{L} &= U \odot USV^T VS^T - U \odot XV^T S^T = 0 \\ \nabla_S \mathcal{L} &= S \odot U^T USV^T V - S \odot U^T XV = 0 \\ \nabla_V \mathcal{L} &= V \odot S^T U^T USV^T - V \odot S^T U^T X = 0 \end{aligned}$$

Desta forma, é possível resolver para U de forma algébrica:

$$\begin{aligned} U \odot USV^T VS^T &= U \odot XV^T S^T \\ \implies U \odot \frac{USV^T VS^T}{USV^T VS^T} &= U \odot \frac{XV^T S^T}{USV^T VS^T} \\ \therefore U &= U \odot \frac{XV^T S^T}{USV^T VS^T} \end{aligned}$$

Da mesma maneira é possível resolver para S e V , assim como mostram as equações [9](#) e [8](#), descritas no algoritmo [1](#) ([LONG; ZHANG; YU, 2005](#)). Considere t o contador de iterações, e $U^{(t)}$, $S^{(t+1)}$ e $V^{(t+1)}$, as matrizes U , S e V , na iteração t , respectivamente.

3.2.2 Fatoração Ortogonal tripla de Matrizes Não-negativas

Baseado no Problema de decomposição em blocos, [Ding et al. \(2006\)](#) propõem o problema [2](#), e o chama de Fatoração Ortogonal Tripla de Matrizes Não-negativas

Algoritmo 1 Algoritmo baseado em atualização multiplicativa para solução do BVD

```

1: function BVD( $X$ ,  $maxIter$ )
2:   Inicialize:  $U^{(0)} \geq 0, V^{(0)} \geq 0, S^{(0)} \geq 0$  e  $t \leftarrow 0$ .
3:   while (não convergiu) ou ( $t \leq maxIter$ ) do
4:     
$$U^{(t+1)} \leftarrow U^{(t)} \odot \frac{XV^{(t)^T}S^{(t)^T}}{U^{(t)}S^{(t)}V^{(t)^T}V^{(t)}S^{(t)^T}} \quad (7)$$

5:     
$$V^{(t+1)} \leftarrow V^{(t)} \odot \frac{S^{(t)^T}U^{(t+1)^T}X}{S^{(t)^T}U^{(t+1)^T}U^{(t+1)}S^{(t)}V^{(t)^T}} \quad (8)$$

6:     
$$S^{(t+1)} \leftarrow S^{(t)} \odot \frac{U^{(t+1)^T}XV^{(t+1)}}{U^{(t+1)^T}U^{(t+1)}S^{(t)}V^{(t+1)^T}V^{(t+1)}} \quad (9)$$

7:      $t \leftarrow t + 1$ 
8:   end while
9:   return  $U^{(t+1)}, S^{(t+1)}, V^{(t+1)}$ 
10: end function

```

(*Orthogonal Non-negative Matrix Tri-factorization - ONMTF*). Colocando duas restrições para ortogonalidade nas matrizes indicadoras de grupos de linhas e colunas, restringe o problema da fatoração $X \approx USV^T$ para um número menor de possíveis soluções, buscando a unicidade, assim como mostra a figura 3.

Problema 2 (Problema de Fatoração Ortogonal tripla de Matrizes Não-negativas).

$$\begin{aligned} \mathcal{F}_2(U, S, V) &= \min_{U, S, V} \|X - USV^T\|_F^2 \\ &\text{suj. a } U \geq 0, S \geq 0, V \geq 0, \\ &\quad U^T U = I, \\ &\quad V^T V = I \end{aligned} \quad (10)$$

onde $U \in \mathbb{R}_+^{n \times k}$, $S \in \mathbb{R}_+^{k \times l}$, $V \in \mathbb{R}^{m \times l}$, $U \geq 0, S \geq 0, V \geq 0$ sendo todos os elementos de U, S e V , maior que 0, respectivamente, $U^T U = I$ e $V^T V = I$ as restrições de ortonormalidade para as matrizes indicadoras de grupos de linhas e colunas, respectivamente, e $\|\cdot\|_F$ denota a norma de Frobenius para matrizes.

Ding et al. (2006) propõem uma solução semelhante ao que foi apresentado na subseção 3.2.1, fazendo a derivação através da função lagrangeana e a introdução dos multiplicadores de lagrange, utilizando as condições de otimização não-linear de KKT, derivando as regras para atualização multiplicativa para U , S e V , apresentadas no algoritmo 2. Considere t o contador de iterações, e $U^{(t)}$, $S^{(t+1)}$ e $V^{(t+1)}$, as matrizes U , S e V , na iteração t , respectivamente.

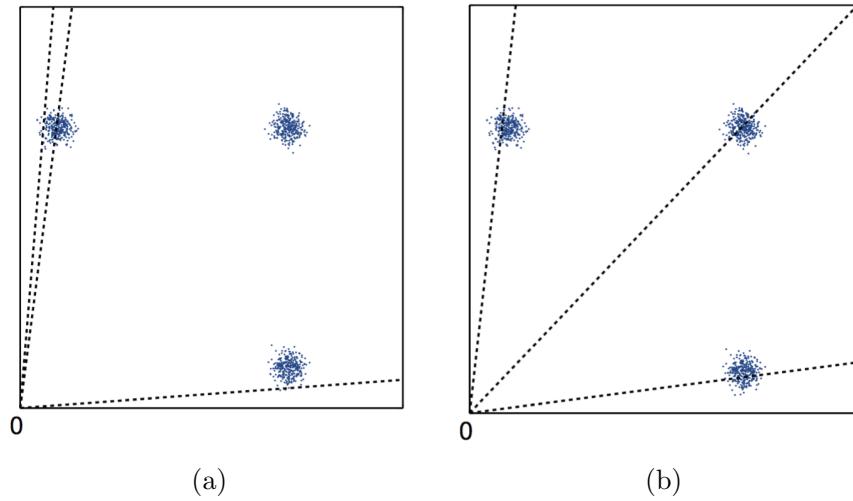


Figura 3 – Um exemplo sintético que compara o algoritmo para BVD contra o algoritmo para ONMTF, com pontos sendo dados, as linhas pontilhadas sendo os protótipos de linhas (SV^T), e os três conjuntos de pontos sendo os grupos. (a) O algoritmo para BVD encontra uma solução com dois protótipos em um mesmo grupo, deixando um dos grupos sem nenhum protótipo para representá-lo. Apesar de ser uma solução correta, ou seja, encontra um mínimo local, não é a desejada. (b) O algoritmo para ONMTF é capaz de encontrar a solução em que cada protótipo aproxima cada grupo de dados, através das restrições referentes à ortogonalidade, este é capaz de restringir as possíveis soluções para a fatoração $X \approx USV^T$. (YOO; CHOI, 2010)

Algoritmo 2 Algoritmo baseado em atualização multiplicativa para solução do ONMTF

```

1: function ONMTF( $X$ ,  $maxIter$ )
2:   Inicialize:  $U^{(0)} \geq 0$ ,  $V^{(0)} \geq 0$ ,  $S^{(0)} \geq 0$  e  $t \leftarrow 0$ .
3:   while (não convergiu) ou ( $t \leq maxIter$ ) do
4:     
$$U^{(t+1)} \leftarrow U^{(t)} \odot \frac{XV^{(t)}S^{(t)T}}{U^{(t)}U^{(t)T}XV^{(t)}S^{(t)T}} \quad (11)$$

5:     
$$V^{(t+1)} \leftarrow V^{(t)} \odot \frac{X^TU^{(t+1)}S}{V^{(t)}V^{(t)T}X^TU^{(t+1)}S^{(t)}} \quad (12)$$

6:     
$$S^{(t+1)} \leftarrow S^{(t)} \odot \frac{U^{(t+1)T}XV^{(t+1)}}{U^{(t+1)T}U^{(t+1)}S^{(t)}V^{(t+1)T}V^{(t+1)}} \quad (13)$$

7:      $t \leftarrow t + 1$ 
8:   end while
9:   return  $U^{(t+1)}$ ,  $S^{(t+1)}$ ,  $V^{(t+1)}$ 
10: end function

```

No artigo de [Yoo e Choi \(2010\)](#), é proposta uma abordagem mais simples para a derivação das regras de atualização multiplicativas, considere uma função de otimização qualquer \mathcal{J} e seu respectivo gradiente $\nabla \mathcal{J}$:

$$\nabla \mathcal{J} = [\nabla \mathcal{J}]^+ - [\nabla \mathcal{J}]^-$$

onde $[\nabla \mathcal{J}]^+$ é a parte positiva do gradiente, $[\nabla \mathcal{J}]^-$ a parte negativa do gradiente. Se $[\nabla \mathcal{J}]^+ \geq 0$ e $[\nabla \mathcal{J}]^- \geq 0$, então, é possível definir uma regra de atualização multiplicativa, para otimizar os parâmetros Θ da função \mathcal{J} :

$$\Theta \leftarrow \Theta \odot \left(\frac{[\nabla \mathcal{J}]^-}{[\nabla \mathcal{J}]^+} \right)^{\cdot \eta} \quad (14)$$

onde \odot representa o produto Hadamard, $(\cdot)^{\cdot \eta}$ representa a potência para cada elemento, e η uma taxa de aprendizado ($0 < \eta \leq 1$). Então, se Θ for inicializado com elementos positivos, é possível verificar que a regra de atualização multiplicativa da equação 14 mantém a não-negatividade de Θ .

Também, é utilizada uma abordagem diferente para a derivação de regras de atualização multiplicativas, visando um algoritmo para a solução do problema 2. Neste caso, o gradiente é calculado com base em uma superfície com restrições que preserva a ortogonalidade. Essa superfície com restrições é chamada de Variedade de Stiefel (*Stiefel Manifold*), neste caso é usada a Variedade de Stiefel no espaço euclidiano, denotada por $\mathcal{H}_{a,b}$, sendo essa variedade o conjunto de $a \times b$ matrizes ortonormais no espaço \mathbb{R}^a , formalmente:

$$\mathcal{H}_{a,b} = \{Y \in \mathbb{R}^{a \times b} : Y^T Y = I\}$$

Note que quando $b = 1$, a superfície se torna uma esfera. Para otimização, considerando que essa esfera sejam as restrições do problema, o ideal é propor métodos que permanecem na esfera, então, todos os vetores tangentes à essa esfera, podem ser direções possíveis para um algoritmo de otimização iterativo, como mostra a Figura 4.

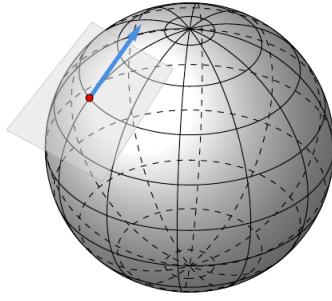


Figura 4 – Uma Variedade Stiefel no espaço Euclidiano, que quando $b = 1$ essa superfície será uma esfera, e um dos possíveis vetores que está contido no conjunto dessa variedade (tangente à esfera).

[Edelman, Arias e Smith \(1999\)](#) definem o gradiente em um ponto Y , com a restrição $Y^T Y = I$, de uma função \mathcal{J} definida em uma Variedade Stiefel no espaço euclidiano como:

$$\tilde{\nabla}_Y \mathcal{J} = \nabla_Y \mathcal{J} - Y(\nabla_Y \mathcal{J})^T Y \quad (15)$$

onde $\nabla_Y \mathcal{J}$ é o gradiente da função \mathcal{J} para todos os elementos da matriz Y .

Dada a equação 15, [Yoo e Choi \(2010\)](#) propõem os seguintes cálculos dos gradientes em Variedades Stiefel para \mathcal{F}_2 , com U no conjunto $\{U^T U = I\}$, e com V no conjunto $\{V^T V = I\}$:

$$\begin{aligned}\tilde{\nabla}_U \mathcal{F}_2 &= \nabla_U \mathcal{F}_2 - U(\nabla_U \mathcal{F}_2)^T U = USV^T X^T U - XVS^T \\ \tilde{\nabla}_V \mathcal{F}_2 &= \nabla_V \mathcal{F}_2 - V(\nabla_V \mathcal{F}_2)^T V = VS^T U^T XV - X^T US\end{aligned}$$

Restando apenas o cálculo do gradiente para S , que como não há restrições, será igual à atualização do algoritmo para solução do BVD:

$$\tilde{\nabla}_S \mathcal{F}_2 = \nabla_S \mathcal{F}_2 = U^T USV^T V - U^T X V$$

É possível, assim, usar a estratégia da equação 14 e os gradientes calculados, para propor uma solução para o problema de ONMTF, alternativas às atualizações

das equações 11, 12 e 13, através da atualização multiplicativa. Essas atualizações são apresentadas nas equações 16a, 16b e 16c.

$$U^{(t+1)} \leftarrow U^{(t)} \odot \frac{XV^{(t)}S^{(t)T}}{U^{(t)}S^{(t)}V^{(t)T}X^TU^{(t)}} \quad (16a)$$

$$V^{(t+1)} \leftarrow V^{(t)} \odot \frac{X^TU^{(t+1)}S^{(t)}}{V^{(t)}S^{(t)T}U^{(t+1)T}XV^{(t)}} \quad (16b)$$

$$S^{(t+1)} \leftarrow S^{(t)} \odot \frac{U^{(t+1)T}XV^{(t+1)}}{U^{(t+1)T}U^{(t+1)}S^{(t)}V^{(t+1)T}V^{(t+1)}} \quad (16c)$$

3.3 Fatoração tripla rápida de Matrizes Não-negativas

O problema de Fatoração tripla rápida de Matrizes Não-negativas (*Fast Non-negative Matrix Tri Factorization - FNMTF*) foi proposto por [Wang et al. \(2011\)](#), com os seguintes argumentos contra o uso prático dos problemas até agora propostos para encontrar cogrupos: os problemas de fatoração em três matrizes propostos até o momento, envolviam soluções algorítmicas iterativas, com multiplicações de matrizes intensas em cada etapa; e pelo fato dos algoritmos propostos realizarem coagrupamento relaxado, necessitando de pós-processamento para a extração dos grupos, o que implica em inúmeras soluções para o agrupamento.

Problema 3 (Fatoração tripla rápida de Matrizes Não-negativas).

$$\begin{aligned} \mathcal{F}_3(F, S, G) &= \min_{F, S, G} \|X - FSG^T\|_F^2 \\ &\quad F \in \Psi^{n \times k}, \\ &\quad G \in \Psi^{m \times l} \end{aligned} \quad (17)$$

onde $X \in \mathbb{R}^{n \times m}$, $S \in \mathbb{R}^{k \times l}$, $\Psi = \{0, 1\}$.

Assim como nos outros algoritmos, F é uma matriz indicadora dos grupos de linhas, G uma matriz indicadora dos grupos de colunas, e S contém os fatores que conectam um grupo de linhas aos grupos de colunas, e, um grupo de colunas aos grupos de linhas.

Como não há restrições em \mathcal{F}_3 , é possível encontrar uma regra de atualização para S , e portanto, minimização de \mathcal{F}_3 :

$$\begin{aligned} \nabla_S \mathcal{L} &= F^T X G - F^T F S G^T G = 0 \\ \implies F^T F S G^T G &= F^T X G \\ \implies (F^T F)^{-1} F^T F S G^T G (G^T G)^{-1} &= (F^T F)^{-1} F^T X G (G^T G)^{-1} \end{aligned}$$

$$\therefore S = (F^T F)^{-1} F^T X G (G^T G)^{-1} \quad (18)$$

Sendo assim, é possível resolver os subproblema para atualizar F e G . Primeiramente, fixando S e G e resolvendo o problema 3 para F de forma iterativa, verificando para cada linha de X , qual o protótipo mais o aproxima (semelhante ao algoritmo de agrupamento *K-means*), da mesma forma, é possível fixar S e F para alcançar uma solução iterativa para G , assim como mostra o algoritmo 3. Considere os índices $i = \{1, \dots, n\}$, $j = \{1, \dots, m\}$, $p = p' = \{1, \dots, k\}$, e $q = q' = \{1, \dots, l\}$, o contador de iterações t , e $U^{(t)}$, $S^{(t+1)}$ e $V^{(t+1)}$, as matrizes U , S e V , na iteração t , respectivamente.

Algoritmo 3 Algoritmo FNMTF

```

1: function FNMTF( $X, maxIter$ )
2:   Inicialize:  $F^{(0)} \geq 0, S^{(0)} \geq 0, G^{(0)} \geq 0$  e  $t \leftarrow 0$ .
3:   while (não convergiu) ou ( $t \leq maxIter$ ) do
4:
  4:    $S^{(t+1)} \leftarrow (F^{(t)^T} F^{(t)})^{-1} F^{(t)^T} X G^{(t)} (G^{(t)^T} G^{(t)})^{-1}$            (19)
5:
  5:    $\tilde{g}_{pj} \leftarrow (S^{(t+1)} G^{(t)^T})_{pj}, \forall p, j$ 
6:
  6:    $(F^{(t+1)})_{ip} \leftarrow \begin{cases} 1 & p = \arg \min_{p' \in \{1, \dots, k\}} \|x_{i \cdot} - \tilde{g}_{p \cdot}\|^2 \\ 0 & \text{caso contrário} \end{cases} \quad \forall i, p$            (20)
7:
  7:    $\tilde{f}_{iq} \leftarrow (F^{(t+1)} S^{(t+1)})_{iq}, \forall i, q$ 
8:
  8:    $(G^{(t+1)})_{jq} \leftarrow \begin{cases} 1 & q = \arg \min_{q' \in \{1, \dots, l\}} \|x_{\cdot j} - \tilde{f}_{\cdot q'}\|^2 \\ 0 & \text{caso contrário} \end{cases} \quad \forall j, q$            (21)
9:    $t \leftarrow t + 1$ 
10:  end while
11:  return  $F^{(t+1)}, S^{(t+1)}, G^{(t+1)}$ 
12: end function

```

4 Algoritmos propostos

4.1 Fatoração Tripla de Matrizes Não-negativas Sobrepostas

Problema 4 (Problema de Fatoração Tripla de Matrizes Não-negativas Sobrepostas).

$$\begin{aligned} \mathcal{F}_4(U, S, V_{(1)}, \dots, V_{(k)}) = & \min_{U, S, V_{(1)}, \dots, V_{(k)}} \left\| X - U \sum_{p=1}^k I_{(p)} S V_{(p)}^T \right\|_F^2 \\ & \text{suj. } a \quad U \geq 0, \\ & \quad S \geq 0, \\ & \quad V_{(p)} \geq 0, \quad \forall p \end{aligned} \quad (22)$$

sendo os índices $p = \{1, \dots, k\}$, o conjunto de matrizes $\mathcal{V} = \{V_{(1)}, \dots, V_{(k)}\}$, em que cada $V_{(p)} \in \mathbb{R}_+^{m \times l}$, uma matriz seletora $I_{(p)}$ que contém 1 na posição $I_{p,p}$ da sua diagonal e 0 no resto, $U \in \mathbb{R}_+^{n \times k}$, $S \in \mathbb{R}_+^{k \times l}$, e $\|\cdot\|_F$ a norma de Frobenius para matrizes.

Algoritmo 4 Algoritmo baseado em atualização multiplicativa para solução do OvNMTF

```

1: function OvNMTF( $X, maxIter$ )
2:   Inicialize:  $U^{(0)} \geq 0, S^{(0)} \geq 0, V_{(0)}^{(0)} \geq 0, \dots, V_{(k)}^{(0)} \geq 0$  e  $t \leftarrow 0$ .
3:   while (não convergiu) ou ( $t \leq maxIter$ ) do
4:
     $U^{(t+1)} \leftarrow U^{(t)} \odot \sum_{p=1}^k \frac{X V_{(p)}^{(t)T} S^{(t)T} I_{(p)}}{\sum_{p'=1}^k U^{(t)} I_{(p)} S^{(t)} V_{(p)}^{(t)T} V_{(p')}^{(t)T} S^{(t)T} I_{(p')}} \quad (23)$ 
5:   for  $p \leftarrow 1, k$  do
6:
     $V_{(p)}^{(t+1)} \leftarrow V_{(p)}^{(t)} \odot \frac{S^{(t)T} I_{(p)} U^{(t+1)T} X}{S^{(t)T} I_{(p)} U^{(t+1)T} U^{(t+1)} I_{(p)} S^{(t)} V_{(p)}^{(t)}} \quad (24)$ 
7:   end for
8:
     $S^{(t+1)} \leftarrow S^{(t)} \odot \sum_{p=1}^l \frac{I_{(p)} U^{(t+1)T} X V_{(p)}^{(t+1)T}}{\sum_{p'=1}^l I_p U^{(t+1)T} U^{(t+1)} I_{(p')} S^{(t)} V_{(p')}^{(t+1)T} V_{(p)}^{(t+1)T}} \quad (25)$ 
9:    $t \leftarrow t + 1$ 
10:  end while
11:  return  $U^{(t+1)}, S^{(t+1)}, V_{(1)}^{(t+1)}, \dots, V_{(k)}^{(t+1)}$ 
12: end function

```

4.2 Fatoração Binária Tripla de Matrizes Não-negativas Sobrepostas

Problema 5 (Problema de Fatoração Binária Tripla de Matrizes Não-negativas Sobrepostas).

$$\begin{aligned} \mathcal{F}_5(F, S, G_{(1)}, \dots, G_{(k)}) = & \min_{F, S, G_{(1)}, \dots, G_{(k)}} \left\| X - F \sum_{p=1}^k I_{(p)} S G_{(p)}^T \right\|_F^2 \\ & \text{suj. a} \quad F \in \Psi^{n \times k}, \\ & \quad S \in \Psi^{k \times l}, \\ & \quad G_{(p)} \in \Psi^{m \times l}, \quad \forall p \end{aligned} \tag{26}$$

sendo $\Psi = 0, 1$, os índices $p = \{1, \dots, k\}$, o conjunto de matrizes $\mathcal{G} = \{G_{(1)}, \dots, G_{(k)}\}$, uma matriz seletora $I_{(p)}$ que contém 1 na posição $I_{p,p}$ da sua diagonal e 0 no resto, $S \in \mathbb{R}_+^{k \times l}$, $X \in \mathbb{R}_+^{n \times m}$, e $\|\cdot\|_F$ a norma de Frobenius para matrizes.

Algoritmo 5 Algoritmo BinOvNMTF

1: **function** BINOVNMTF($X, maxIter$)
 2: **Inicialize:** $F^{(0)} \geq 0, S^{(0)} \geq 0, G_{(0)}^{(0)} \geq 0, \dots, G_{(k)}^{(0)} \geq 0$ e $t \leftarrow 0$.
 3: **while** (não convergiu) ou ($t \leq maxIter$) **do**
 4:

$$(S^{(t+1)})_{pq} \leftarrow \frac{1}{|\{\mathbf{x}_{i \cdot} \in \mathcal{C}_p\}| |\{\mathbf{x}_{\cdot j} \in \mathcal{W}_q\}|} \sum_{i=1}^n \sum_{j=1}^m \sum_{\mathbf{x}_{i \cdot} \in \mathcal{C}_p} \sum_{\mathbf{x}_{\cdot j} \in \mathcal{W}_q} x_{ij} \quad (27)$$

 5: **for** $p \leftarrow 1, k$ **do**
 6:

$$\tilde{F} \leftarrow \begin{bmatrix} & & & \vdots \\ & \mathbf{f}_{i \cdot}^{(t)}, \forall \mathbf{x}_{i \cdot} \in \mathcal{C}_p & & \end{bmatrix} S^{(t+1)}$$

 7:

$$\tilde{X} \leftarrow \begin{bmatrix} & & & \vdots \\ & \mathbf{x}_{i \cdot}, \forall \mathbf{x}_{i \cdot} \in \mathcal{C}_p & & \end{bmatrix}$$

 8:

$$(G_{(p)}^{(t+1)})_{jq} \leftarrow \begin{cases} 1 & q = \arg \min_{q' \in \{1, \dots, l\}} \|\tilde{\mathbf{x}}_{\cdot j} - \tilde{\mathbf{f}}_{\cdot q'}\|^2 \\ 0 & \text{caso contrário} \end{cases} \quad \forall j, q \quad (28)$$

 9: **end for**
 10:

$$\tilde{\mathbf{g}}_{p \cdot} \leftarrow (\mathbf{s}_{p \cdot}^{(t+1)} G_{(p)}^{(t)^T})_{pj}, \forall p$$

 11:

$$(F^{(t+1)})_{ip} \leftarrow \begin{cases} 1 & p = \arg \min_{p' \in \{1, \dots, k\}} \|\mathbf{x}_{i \cdot} - \tilde{\mathbf{g}}_{p' \cdot}\|^2 \\ 0 & \text{caso contrário} \end{cases} \quad \forall i, p \quad (29)$$

 12: $t \leftarrow t + 1$
 13: **end while**
 14: **return** $F^{(t+1)}, S^{(t+1)}, G_{(1)}^{(t+1)}, \dots, G_{(k)}^{(t+1)}$
 15: **end function**

5 Experimentos e Resultados

Para fim de validação dos algoritmos foram feitos experimentos utilizando bases de dados sintéticas, base de dados toy e base de dados reais. Estes experimentos foram realizados à fim de comprovar a utilidade de métodos de Cocluterização baseados em Fatoração de Matrizes, tanto os algoritmos desenvolvidos quanto os já presentes na literatura, para coclusterização de documentos e mineração destes documentos. Os algoritmos usados nos experimentos compõem algoritmos de clustering tradicionais: *k-means* e *fuzzy k-means*, e algoritmos de coclustering baseados em fatoração de matrizes da literatura: *ONMTF*, *FNMTF*; e os algoritmos propostos no capítulo 4: *OvNMTF* e *BinOvNMTF*.

5.1 Experimentos com Base de Dados sintéticas

As bases de dados sintéticas foram criadas com inspiração nas diferentes estruturas de biclusters, propostas por [Madeira e Oliveira \(2004\)](#), descritas com maior detalhes no capítulo 2. Dentre as nove possíveis estruturas de biclusters, foram escolhidas as estruturas de bicluster com intersecção de linhas ou colunas em conjunto com as estruturas mais simples, que fazem sentido com o objetivo principal deste trabalho, que é, em linhas gerais, encontrar soluções algorítmicas baseadas em fatoração de matrizes que resolvam problemas de cocluster com intersecção de linhas e/ou colunas.

A forma de interpretar as matrizes da figura 5 é considerar cada elemento dessa matriz com uma cor relacionada com o seu valor ou intensidade, organizando as linhas e colunas, de tal forma que os valores similares formem retângulos, que são biclusters.

Todas as matrizes de dados da figura 5 foram geradas sinteticamente, primeiramente criando uma matriz com 150 linhas e 150 colunas, e preenchendo-a com valores ponto flutuante de 0.0 à 1.0, gerados a partir de uma distribuição uniforme, denotado pela função $unif(0.0, 1.0)$, que gera valores aleatórios uniformes de 0.0 à 1.0. Este processo evita que ocorram divisões por 0 nos algoritmos.

Então, foram gerados os bicluster, considerando um conjunto de valores centrais $\mathcal{C} = \{20, 40, 60, 80, 100, 120, 140, 160, 180\}$, e escolhendo um valor central $c \in \mathcal{C}$ de forma aleatória, adicionado valores de 0.0 à 10.0, gerados a partir da função $unif(0.0, 10.0)$, que gera valores aleatórios uniformes de 0.0 à 10.0, para então formar um bicluster. Assim,

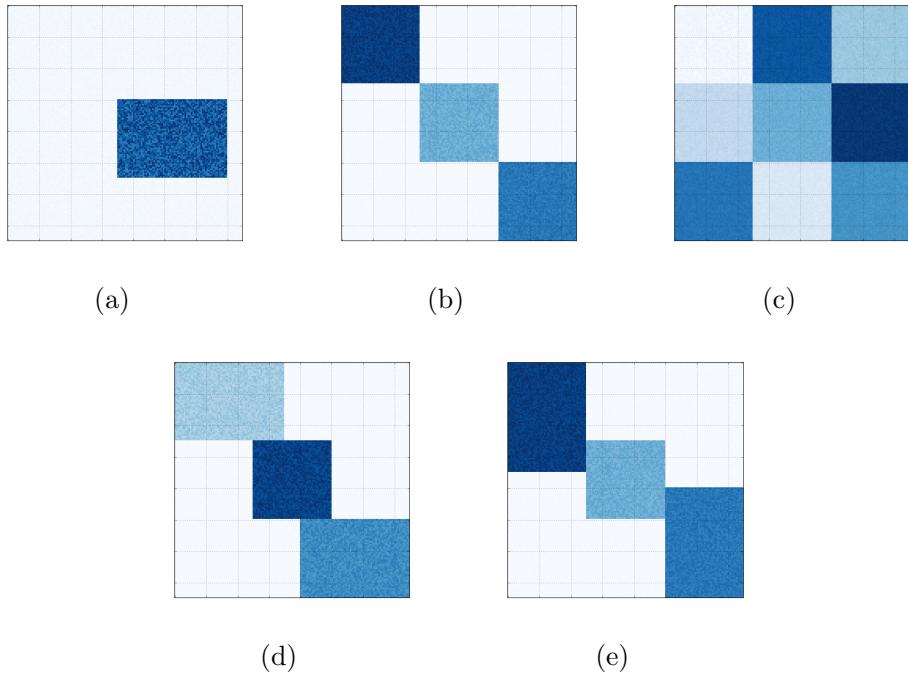


Figura 5 – Dados sintéticos gerados a partir das diferentes estruturas de biclusters. (a) Um único bicluster. (b) Biclusters com linhas e colunas sem intersecção. (c) Biclusters com estrutura em xadrez. (d) Biclusters com linhas sem intersecção e colunas com intersecção. (e) Biclusters com linhas com intersecção e colunas sem intersecção.

considerando um bicluster denotado por B , uma submatriz dos dados originais, este pode ser gerado pela equação 30.

$$b_{ij} = c + \text{unif}(0, 10) \quad (30)$$

sendo i e j os índices das linhas e colunas de B , respectivamente, b_{ij} um elemento de B , e $\text{unif}(0, 10)$ uma função geradora de números de 0 à 10, que respeita a distribuição uniforme. Considerando a matriz de dados sintéticos como X , cada elemento dessa matriz recebe o bicluster criado: $x_{ij} = b_{ij}, \forall i, j$.

Uma forma de analisar o resultado dos algoritmos é analisá-los quanto a sua capacidade de reconstrução, ou seja, após a execução de um algoritmo de fatoração de matrizes, coletar as matrizes resultantes da fatoração e combiná-las, da mesma forma que o seu problema foi proposto, de forma a reconstruir a matriz que foi fatorada.

Os seguintes biclusters e coclusters podem ser encontrados nas matrizes da figura 5, considerando que cada matriz têm dimensão 150×150 , então, os índices das linhas estão entre 0 e 149 e os índices das colunas também de 0 à 149:

- **Figura 5a** há um bicluster, formado pelas linhas de 60 à 110 e pelas colunas de 70 à 140; sendo então, dois coclusters de linhas, o primeiro formado pelas linhas de 60 à 110 e o segundo pelas demais linhas, e dois cogrupos de colunas, o primeiro formado pelas colunas de 70 à 140 e o segundo pelas demais colunas.
- **Figura 5b** há três biclusters sem intersecção, o primeiro formado pelas linhas de 0 à 49 e pelas colunas de 0 à 49, o segundo formado pelas linhas de 50 à 99 e pelas colunas 50 à 99, o terceiro formado pelas linhas de 100 à 149 e pelas colunas de 100 à 149; sendo então, três coclusters de linhas sem intersecção e três coclusters de colunas sem intersecção, o primeiro formado pelas linhas de 0 à 49, o segundo formado pelas linhas de 50 à 99, o terceiro formado pelas linhas de 100 à 149, o quarto formado pelas colunas de 0 à 49, o quinto formado pelas colunas de 50 à 99, e o sexto formado pelas colunas de 100 à 149, respectivamente. Também existe outra forma de observar os coclusters presentes neste caso, se for considerado os coclusters de colunas para cada cocluster de linhas, seriam os mesmos três coclusters de linhas e dois coclusters de colunas para cada cocluster de linha, ou seja, para o primeiro cocluster de linhas (linhas 0 à 49), os dois coclusters de colunas são os coclusters formados pelas colunas de 0 à 49 e pelas demais colunas (50 à 149), seguindo a mesma lógica para os demais coclusters. Note que desta maneira, é possível que coclusters de colunas tenham intersecção.
- **Figura 5c** há nove biclusters com intersecção em estrutura de xadrez, formados através da divisão da matriz de dados sintética em partes iguais; sendo assim, serão 3 coclusters de linhas sem intersecção e 3 coclusters de colunas sem intersecção, o primeiro formado pelas linhas de 0 à 49, o segundo formado pelas linhas de 50 à 99, o terceiro formado pelas linhas de 100 à 149, o quarto formado pelas colunas de 0 à 49, o quinto formado pelas colunas de 50 à 99, e o sexto formado pelas colunas de 100 à 149. É possível fazer a mesma observação de coclusters realizada no item **Figura 5b**, totalizando nove coclusters, três coclusters de linhas e três coclusters de colunas para cada cocluster de linha.
- **Figura 5d** há três biclusters com intersecção nas colunas, o primeiro formado pelas linhas de 0 à 49 e pelas colunas de 0 à 69, o segundo formado pelas linhas de 50 à 99 e pelas colunas 50 à 99, e o terceiro formado pelas linhas de 100 à 149 e pelas colunas de 79 à 149; sendo então, 3 coclusters de linhas sem intersecção e 3 coclusters de colunas com intersecção. Considerando os coclusters de colunas para cada cocluster

de linhas, serão nove coclusters, dois coclusters de colunas para cada um dos três coclusters de linhas.

- **Figura 5e** Este caso é exatamente igual ao item **Figura 5d** se for observado a transposição da matriz de dados sintética.

Para cada um dos matrizes (subimages da figura 5) foi executado todos os seguintes algoritmos: *k-means*, *fuzzy k-means*, *ONMTF*, *FNMTF*, *OvNMTF* e *BinOvNMTF*. Foram criados experimentos para analisar a capacidade de reconstrução dos algoritmos e a capacidade de particionamento, tanto das linhas quanto das colunas.

5.1.1 Experimentos de reconstrução

Foram realizados experimentos avaliando a capacidade de reconstrução dos algoritmos para cada uma das bases de dados sintéticas criadas. Esse tipo de experimento se torna importante a partir do momento que o objetivo é avaliar como os algoritmos se comportam com estruturas com intersecção de cocluster nas linhas ou colunas.

5.1.2 Resultados da reconstrução de dados sintéticos com *k-means*

Para esse experimento foi usado a implementação do algoritmo *k-means* da biblioteca *scikit-learn* ([PEDREGOSA et al., 2011](#)) da linguagem Python¹.

Como critério de parada do algoritmo, foi utilizada a regra do número máximo de iterações, com o valor de 300 iterações, ou a diferença do erro de minimização assumir valor menor ou igual à $1e - 4$. O parâmetro do número de clusters foi configurado diferente para cada conjunto de dados sintéticos criado: $k = 2$ para (a) e $k = 3$ para (b), (c), (d) e (e).

A partir da execução de 10 rodadas do algoritmo, medindo a taxa de erro que o algoritmo minimiza, foi coletado o melhor modelo resultante para avaliar a sua capacidade.

As reconstruções, presentes nas subfiguras da Figura 6, tiveram suas linhas coloridas através do centróide que as representavam, ou seja, se a linha 10 pertencer ao centróide 1, então os valores da linha 10 serão os valores do centróide.

¹ <https://www.python.org/>

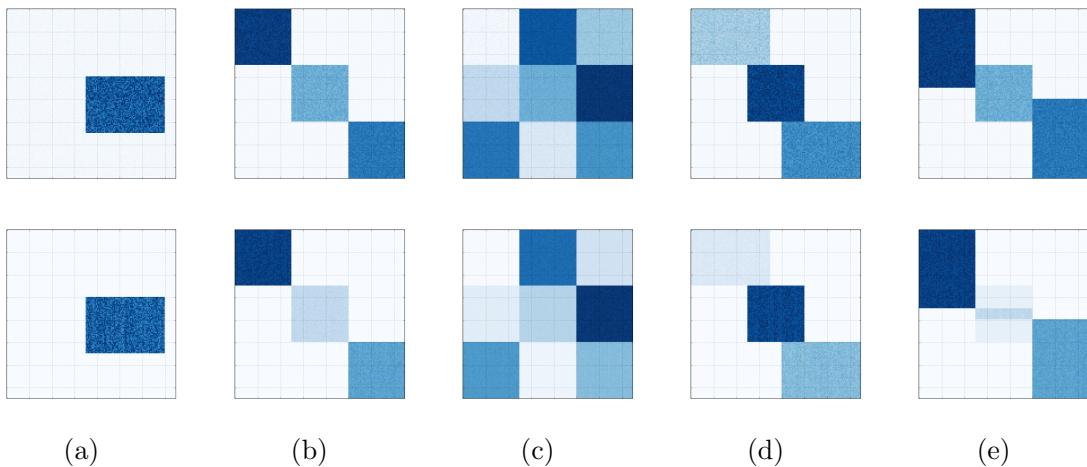


Figura 6 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo *k-means*.

Note que o modelo resultante da execução é capaz de realizar a reconstrução perfeita dos casos (a) à (d). No entanto, o modelo não tem informações sobre a estrutura de coclusters de colunas presentes nos dados. O caso (e) é o único que o algoritmo não é capaz de realizar a reconstrução, isso ocorre pois o algoritmo não é capaz de encontrar intersecção de clusters.

5.1.3 Resultados da reconstrução de dados sintéticos com *fuzzy k-means*

Para esse experimento foi usado a implementação do algoritmo *fuzzy k-means* da biblioteca *scikit-fuzzy*² da linguagem Python.

Como critério de parada do algoritmo, assim como o experimento anterior, foi utilizada a regra do número máximo de iterações, com o valor de 300 iterações, ou a diferença do erro de minimização assumir valor menor ou igual à $1e - 4$. O parâmetro do número de clusters foi configurado da seguinte maneira: $k = 2$ para (a) e $k = 3$ para (b), (c), (d) e (e). O parâmetro de exponenciação foi configurado com o valor 2.

A partir da execução de 10 rodadas do algoritmo, medindo a taxa de erro que o algoritmo minimiza, foi coletado o melhor modelo resultante para avaliar a sua capacidade.

Da mesma forma que o experimento anterior, as reconstruções presentes nas subfiguras da Figura 7, tiveram suas linhas coloridas através do centróide que as representavam.

² <http://pythonhosted.org/scikit-fuzzy/>

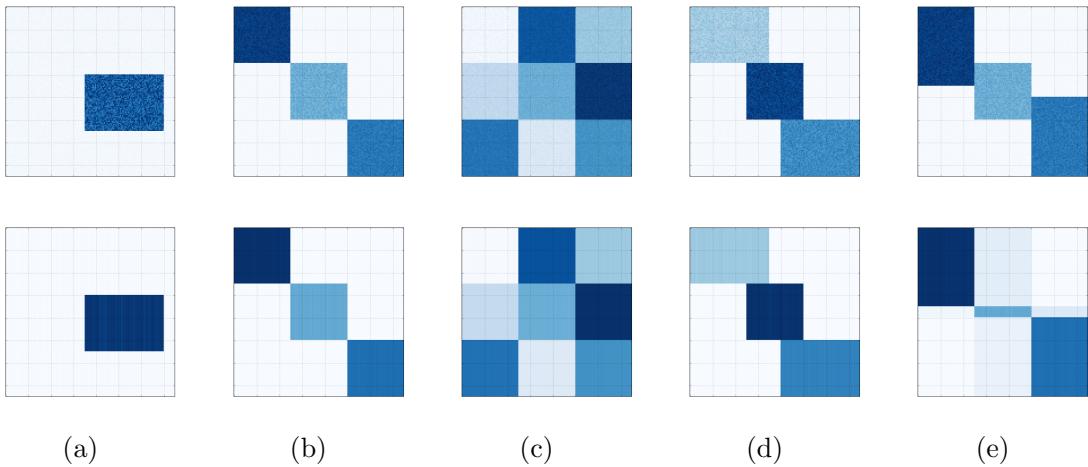


Figura 7 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo *fuzzy k-means*.

Semelhante ao experimento de reconstrução com o *k-means*, este também é capaz de realizar a reconstrução perfeita dos casos (a) à (d), porém, apesar do algoritmo lidar com intersecção de clusters, este não foi capaz de fazer a reconstrução do caso (e).

5.1.4 Resultados da reconstrução de dados sintéticos com *ONMTF*

Para esse experimento, foi feita a implementação do algoritmo *ONMTF* usando a linguagem Python.

Como critério de parada do algoritmo, foi utilizada 1000 como número máximo de iterações, ou a diferença do erro de minimização assumir valor menor ou igual à $1e - 4$. Os parâmetros do número de coclusters de linhas e colunas foi configurado da seguinte maneira: $k = l = 2$ para (a) e $k = l = 3$ para (b), (c), (d) e (e).

Para o funcionamento do algoritmo *ONMTF* é necessária a normalização dos dados, então, todas as matrizes de dados sintéticas foram normalizadas para que a norma das linhas seja igual à 1.

A partir da execução de 10 rodadas do algoritmo, medindo a taxa de erro que o algoritmo minimiza, foi coletado o melhor modelo resultante para avaliar a sua capacidade.

As reconstruções presentes nas subfiguras da Figura 8, foram construídas a partir da multiplicação das matrizes fatoradas, ou seja, USV^T .

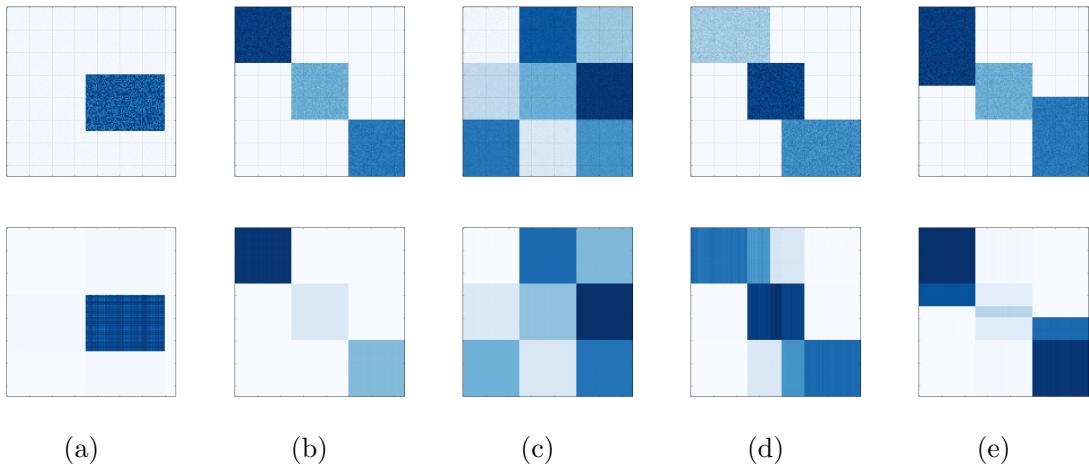


Figura 8 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo *ONMTF*.

Analizando as reconstruções realizadas, é possível perceber que a reconstrução é realizada com êxito nos casos (a), (b) e (c). O algoritmo falhou em reconstruir as intersecções de linhas e colunas presentes nos casos (d) e (e), respectivamente.

No entanto, é possível ver sombras nas regiões das matrizes reconstruídas com intersecção de biclusters, isso significa que quando observa-se a matriz V (que contém as relações de pertinência entre coclusters e colunas (características)) para o caso (d), é possível ver valores associados ao segundo cocluster, porém menores que as outras associações com os clusters. Isso é possível pois o algoritmo não força a associação única e exclusiva entre uma linha e um cocluster ou uma coluna e um cocluster.

5.1.5 Resultados da reconstrução de dados sintéticos com *FNMTF*

Para esse experimento, foi feita a implementação do algoritmo *FNMTF* usando a linguagem Python.

Como critério de parada do algoritmo, foi utilizada 300 como número máximo de iterações, ou a diferença do erro de minimização assumir valor menor ou igual à $1e - 4$. Os parâmetros do número de coclusters de linhas e colunas foi configurado da seguinte maneira: $k = l = 2$ para (a) e $k = l = 3$ para (b), (c), (d) e (e).

Para o algoritmo *FNMTF* não é necessária a normalização dos dados.

A partir da execução de 10 rodadas do algoritmo, medindo a taxa de erro que o algoritmo minimiza, foi coletado o melhor modelo resultante para avaliar a sua capacidade.

As reconstruções presentes nas subfiguras da Figura 9, foram construídas a partir da multiplicação das matrizes fatoradas, ou seja, FSG^T .

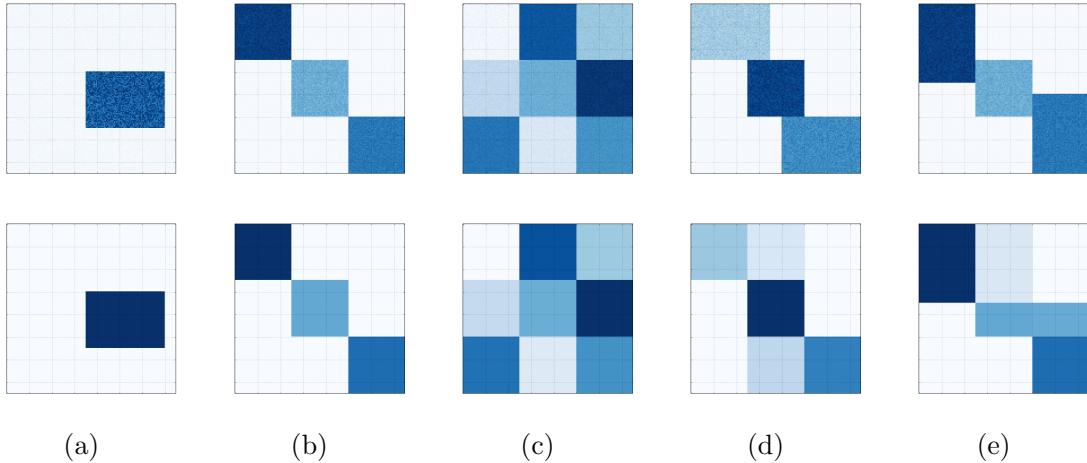


Figura 9 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo *FNMTF*.

Este caso é semelhante ao anterior, o algoritmo conseguiu realizar a reconstrução com êxito dos casos (a), (b) e (c), falhando na reconstrução da intersecção dos casos (d) e (e). No entanto, como não há informação de pertinência, ou seja, o algoritmo impõem que cada linha ou coluna pode pertencer apenas a um cocluster, a informação de intersecção é corrompida.

5.1.6 Resultados da reconstrução de dados sintéticos com *OvNMTF*

Para esse experimento, foi feita a implementação do algoritmo *OvNMTF* usando a linguagem Python.

Como critério de parada do algoritmo, foi utilizada 1000 como número máximo de iterações, ou a diferença do erro de minimização assumir valor menor ou igual à $1e - 4$. Os parâmetros do número de coclusters de linhas e colunas foi configurado da seguinte maneira: $k = l = 2$ para (a); e $k = 3$ e $l = 2$ para (b), (c), (d) e (e).

Para o funcionamento do algoritmo *OvNMTF* é necessária a normalização dos dados, então, todas as matrizes de dados sintéticas foram normalizadas para que a norma das linhas seja igual à 1.

A partir da execução de 10 rodadas do algoritmo, medindo a taxa de erro que o algoritmo minimiza, foi coletado o melhor modelo resultante para avaliar a sua capacidade.

As reconstruções presentes nas subfiguras da Figura 10, foram construídas a partir da multiplicação das matrizes fatoradas, ou seja, $U \sum_{p=1}^k I_{(p)} S V_{(p)}^T$.

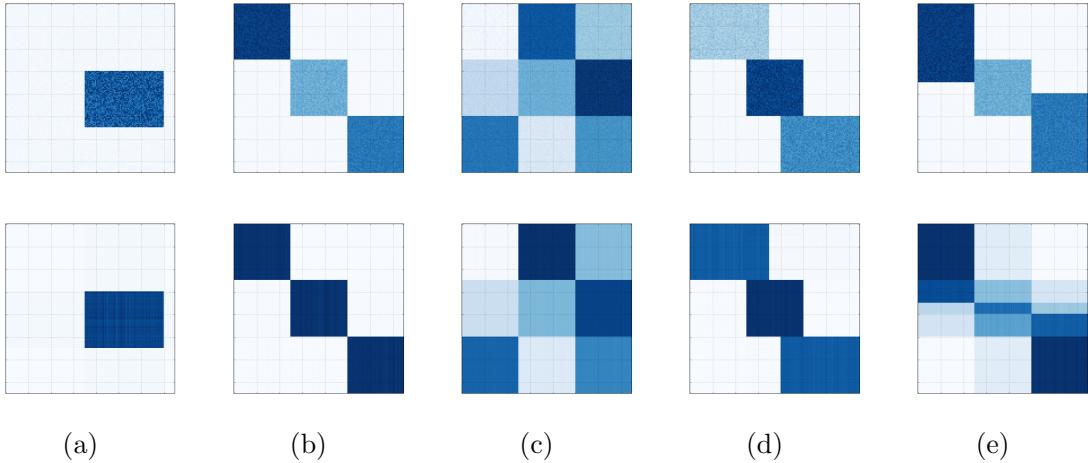


Figura 10 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo *OvNMTF*.

O algoritmo conseguiu realizar a reconstrução com êxito dos casos (a), (b), (c) e (d), com valores suavemente diferentes (isso pode ser notado pela cor das matrizes reconstruídas) devido à normalização necessária. Note que o algoritmo é capaz de lidar com coclusters com intersecção de colunas, então é capaz de resolver o caso (d). Como o algoritmo não é capaz de lidar com coclusters com intersecção nas linhas, este não é capaz de realizar a reconstrução do caso (e), porém, assim como o experimento com *ONMTF*, o *OvNMTF* têm a informação de pertinência de cada linha ou coluna com cada cocluster. Por isso, as intersecções são preservadas, observando as áreas das intersecções de linhas com sombra, na matriz de dados sintéticos.

5.1.7 Resultados da reconstrução de dados sintéticos com *BinOvNMTF*

Para esse experimento, foi feita a implementação do algoritmo *BinOvNMTF* usando a linguagem Python.

Como critério de parada do algoritmo, foi utilizada 1000 como número máximo de iterações, ou a diferença do erro de minimização assumir valor menor ou igual à $1e - 4$. Os parâmetros do número de coclusters de linhas e colunas foi configurado da seguinte maneira: $k = l = 2$ para (a); e $k = 3$ e $l = 2$ para (b), (c), (d) e (e).

A partir da execução de 10 rodadas do algoritmo, medindo a taxa de erro que o algoritmo minimiza, foi coletado o melhor modelo resultante para avaliar a sua capacidade.

As reconstruções presentes nas subfiguras da Figura 11, foram construídas a partir da multiplicação das matrizes fatoradas, ou seja, $F \sum_{p=1}^k I_{(p)} S G_{(p)}^T$.

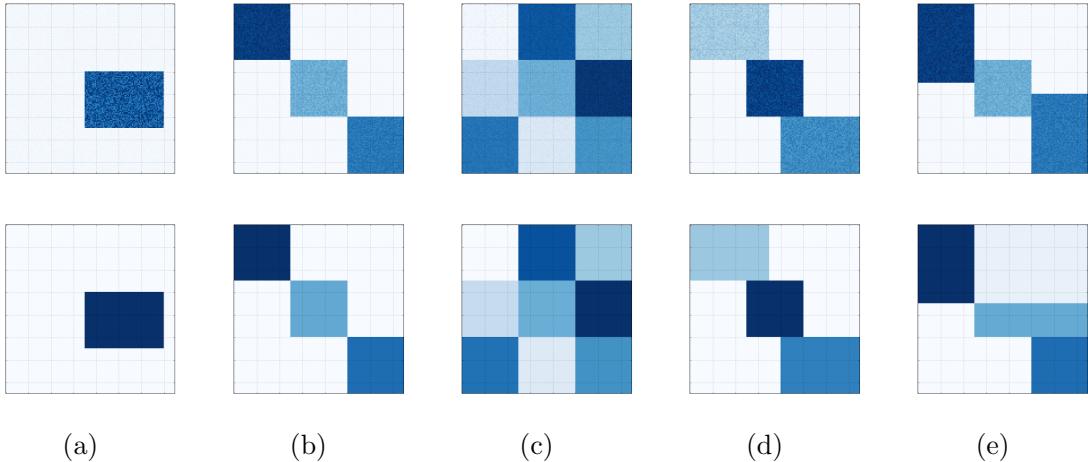


Figura 11 – As primeiras cinco matrizes são as matrizes originais, as demais são suas respectivas reconstruções, realizadas pelo *BinOvNMTF*.

Semelhante ao experimento anterior, o algoritmo conseguiu realizar a reconstrução com êxito dos casos (a), (b), (c) e (d), com valores suavemente diferentes (isso pode ser notado pela cor das matrizes reconstruídas) devido à normalização necessária. Note que o algoritmo é capaz de lidar com coclusters com intersecção de colunas, então é capaz de resolver o caso (d). Como o algoritmo não é capaz de lidar com coclusters com intersecção nas linhas, este não é capaz de realizar a reconstrução do caso (e), porém, assim como o experimento com *ONMTF*, o *OvNMTF* têm a informação de pertinência de cada linha ou coluna com cada cocluster. Por isso, as intersecções são preservadas, observando as áreas das intersecções de linhas com sombra, na matriz de dados sintéticos.

5.1.8 Resultados quantitativos com dados sintéticos

Os mesmos modelos que foram gerados nos experimentos de reconstrução, foram usados para a construção dos resultados desta subseção. Como os as bases de dados foram construídos de forma sintética, foi possível criá-los a partir de rótulos, ou seja, a qual cocluster uma determinada linha ou coluna pertence, permitindo a realização de avaliação externa.

A métrica usada para validação dos cocluster foi o *Rand Index*, este índice gera valores entre 0 e 1, com 0 indicando que o particionamento de dados gerado pelo modelo não concorda em nenhum par de pontos com o particionamento real e 1 indicando que o particionamento gerado pelo modelo é exatamente igual ao particionamento real.

Os resultados são independentes da organização das linhas das matrizes, ou seja, com ou sem embaralhamento é obtido os mesmos resultados para as métricas. O embaralhamento foi realizado de forma aleatória gerando novos índices das linhas a partir de uma distribuição uniforme.

É possível correlacionar os resultados obtidos nas reconstruções com os resultados obtidos através da validação com o *Rand Index*.

A avaliação externa foi feita para o particionamento de linhas e para o particionamento de colunas, observando os valores destacados na tabela ?? da coluna *Rand Index* (linhas), é possível ver que nenhum dos algoritmos foi capaz de particionar perfeitamente (com valor 1.0) as linhas para a base de dados (e), que contém intersecção, assim como nos experimentos com reconstrução. Claramente, isso ocorre pois nenhum dos algoritmos é capaz de encontrar coclusters com intersecção de linhas.

Na coluna *Rand index* (colunas) da tabela ??, também é possível observar que não há resultados para os algoritmos *k-means* e *fuzzy k-means*, pois os mesmos não resolvem o problema de particionamento de colunas. Os valores destacados desta mesma coluna mostra que os algoritmos criados neste trabalho (*OvNMTF* e *BinOvNMTF*) são capazes de particionar coclusters com intersecção de linhas, base de dados sintéticas do caso (d). Os algoritmos *ONMTF* e *FNMTF* não são capazes de lidar com intersecção nas colunas, por isso, o valor 0.78 de *Rand Index*.

Todos os outros casos com valores não destacados na tabela ?? obtiveram particionamento perfeito, ou seja, *Rand Index* igual à 1.0, assim como os experimentos das reconstruções.

Tabela 1 – Estatísticas das bases de dados usadas nos experimentos.

Algoritmo	Base de dados	<i>Rand Index</i> (linhas)	<i>Rand index</i> (colunas)
<i>k-means</i>	(a)	1.0	-
<i>k-means</i>	(b)	1.0	-
<i>k-means</i>	(c)	1.0	-
<i>k-means</i>	(d)	1.0	-
<i>k-means</i>	(e)	0.78	-
<i>fuzzy k-means</i>	(a)	1.0	-
<i>fuzzy k-means</i>	(b)	1.0	-
<i>fuzzy k-means</i>	(c)	1.0	-
<i>fuzzy k-means</i>	(d)	1.0	-
<i>fuzzy k-means</i>	(e)	0.78	-
<i>ONMTF</i>	(a)	1.0	1.0
<i>ONMTF</i>	(b)	1.0	1.0
<i>ONMTF</i>	(c)	1.0	1.0
<i>ONMTF</i>	(d)	1.0	0.78
<i>ONMTF</i>	(e)	0.78	1.0
<i>FNMTF</i>	(a)	1.0	1.0
<i>FNMTF</i>	(b)	1.0	1.0
<i>FNMTF</i>	(c)	1.0	1.0
<i>FNMTF</i>	(d)	1.0	0.78
<i>FNMTF</i>	(e)	0.53	1.0
<i>OvNMTF</i>	(a)	1.0	1.0
<i>OvNMTF</i>	(b)	1.0	1.0
<i>OvNMTF</i>	(c)	1.0	1.0
<i>OvNMTF</i>	(d)	1.0	1.0
<i>OvNMTF</i>	(e)	0.78	1.0
<i>BinOvNMTF</i>	(a)	1.0	1.0
<i>BinOvNMTF</i>	(b)	1.0	1.0
<i>BinOvNMTF</i>	(c)	1.0	1.0
<i>BinOvNMTF</i>	(d)	1.0	1.0
<i>BinOvNMTF</i>	(e)	0.53	1.0

5.2 Experimentos com base de dados reais

5.2.1 Descrição das bases de dados

Nos experimentos deste trabalho foram usadas quatro base de dados de textos, sendo elas no domínio de notícias e artigos. Em todas as bases foram usados os rótulos verdadeiros para avaliar a qualidade dos clusters, usadas apenas após a fase de criação dos modelos, ou seja, não serviram de entrada para os algoritmos.

1. **20Newsgroups**³ Esta coleção de documentos se tornou popular para experimentos de algoritmos de aprendizado de máquina, nas tarefas de classificação de textos e clusterização de textos. Os documentos compreendem posts de usuários anônimos do *Usenet newsgroup* (um repositório para grupos de discussões de um sistema distribuído de comunicação chamado *Usenet*). Cada post está organizado de forma particionada em 20 diferentes grupos, como computação, ciências, política e etc.,

³ <http://qwone.com/~jason/20Newsgroups/>

com cada post correspondendo a um tópico, distribuídos uniformemente entre esses grupos.

2. ***NIPS14-17***⁴ Esta base de dados contém uma coleção de trabalhos acadêmicos publicados no congresso *NIPS* (*Neural Information Processing Systems*) no período de 2003 à 2003, dos volumes 14 à 17. A construção da base de dados *NIPS14-17* teve sua construção iniciada por *Sam Roweis*⁵, que tratou os dados que *Yann LeCun* processou usando um OCR ([GLOBERSON et al., 2007](#)). Originalmente, a base de dados tem todos artigos científicos publicados no NIPS dos volumes 0 ao 17, porém apenas os artigos dos volumes 14 à 17 possuem rótulos. Os documentos estão particionados em tópicos que compreendem as áreas técnicas, como teoria de aprendizado, neurociência, algoritmos e arquiteturas e etc, distribuídos de forma desbalanceada.
3. ***IG*** Este conjunto de dados foi criado neste trabalho, e consiste em um corpus de notícias extraídas do portal iG⁶. Cada documento contém o endereço eletrônico, título, subtítulo, corpo e canal da notícia, em que cada canal representa um assunto da notícia, que é usado como rótulo. O número total de notícias do corpus é 4593, sendo essas notícias formadas por mais de 250 caracteres no seu corpo, publicadas no período de 2 de janeiro de 2012 à 11 de outubro de 2014 classificadas em 13 canais, que representam os assuntos dessas notícias.
4. ***IG toy (ou reduzido)*** Este conjunto de dados é um subconjunto do conjunto de dados *IG*. Composto por apenas três canais (rótulos) de forma balanceada, foi criado para possibilitar experimentos com maior controle.

A tabela ?? sumariza as estatísticas para cada conjunto de dados, contendo o número de palavras totais em todos os documentos, o número de documentos

Tabela 2 – Estatísticas das bases de dados usadas nos experimentos.

Conjuntos de dados	# Palavras	# Documentos	# Clusters reais
<i>20 Newsgroup</i>	12.998	18.221	20
<i>NIPS14-17</i>	17.583	420	9
<i>IG</i>	19.563	4.593	13
<i>IG toy</i>	6.764	3.00	3

⁴ <http://robotics.stanford.edu/~gal/data.html>

⁵ <http://www.cs.nyu.edu/~roweis/data.html>

⁶ <http://ig.com.br/>

A fim de estruturar a informação nas bases de dados para construção dos modelos, foi necessária uma fase de pré-processamento, descrita na subseção 5.2.2.

5.2.2 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).

Para realizar a estruturação de textos e representar os textos dos documentos em vetores de termos, o primeiro processo a ser realizado é a *tokenização*, que cria um dicionário de termos 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). Nos experimentos é utilizado como *tokenizador*, uma expressão regular que separa caractéres não contíguos.

Após isso, é realizada uma etapa de filtragem, que têm a função de retirar termos 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 determinados termos a partir de uma lista, é chamada de filtro de *stopwords*. Também são usadas outros filtros, como a exclusão de pontuações, elementos da web (como *www* e *links*), numerais e tokens monetários, e eliminação de termos com a frequência nos documentos menor que 2. Para a base de dados *IG* é usada uma lista de *stopwords* diferente das demais, por esta se apresentar no idioma português.

Para definição da representação textual dos documentos das bases de dados foram utilizadas múltiplas formas. Uma das representações utilizadas nos experimentos, é o modelo do espaço vetorial, ou *Vector Space Model* (SALTON; WONG; YANG, 1975). Esta é a representação clássica usada para representar documentos textuais (SEBASTIANI, 2002; LOPS; GEMMIS; SEMERARO, 2011). Cada dimensão desse vetor está associada a um termo, sendo que todas as dimensões representam todos os termos do conjunto de documentos.

Formalmente, há um conjunto de documentos $\{d_1, \dots, d_n\}$, em que $d_i, \forall i \in \{1, \dots, n\}$ representa um documento e n o número total de documentos, e um conjunto de termos $\{t_1, \dots, t_m\}$, em que $t_j, \forall j \in \{1, \dots, m\}$ representa um termo e m o número de termos em todos os documentos. Representando a frequência de um termo pelo número de vezes que t_j aparece em um documento d_i , denotado por $tf(t_j, d_i)$, o vetor de termos pode ser construído e representado da seguinte forma: $\mathbf{v}_{d_i} = [tf(t_1, d_i), \dots, tf(t_m, d_i)]$. Salton, Wong e Yang (1975) argumentam que a representação textual de documentos em vetor de termos é suficiente para separar documentos.

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

Essa normalização pode ser calculada como descrito na equação 31, e tem o efeito de fazer com que a frequência dos termos que aparecem em muitos documentos seja reduzida, e a frequência dos termos que aparecem em alguns raros documentos seja aumentada, com um fator de \log_2 .

$$\begin{aligned} tfidf(t_j, d_i) &= tf(t_j, d_i) \cdot IDF(t_j) \\ tfidf(t_j, d_i) &= tf(t_j, d_i) \cdot \left(\log_2 \frac{n}{df(t_j)+1} \right) \end{aligned} \quad (31)$$

em que $idf(t_j)$ representa a frequência de documentos inversa do termo t_j , e $df(t_j)$ a frequência de documentos que contém t_j .

Nos experimentos presentes neste trabalho, também é usada normalização norma- L_2 para que todos os vetores de termos $\mathbf{v}_{d_i}, \forall i$ tenham comprimento iguais, ou seja, $\|\mathbf{v}_{d_i}\| = 1$.

Sendo assim, são realizados experimentos com as seguintes formas: frequência de termos (tf), frequência de termos normalizada ($tf-norm$), frequência de termos-frequência de documentos inversa ($tfidf$), e frequência de termos-frequência de documentos inversa normalizada ($tfidf - norm$).

5.2.3 Experimentos quantitativos

5.2.3.1 Setup dos algoritmos

5.2.3.2 Setup dos experimentos

5.2.3.3 Base de dados ***NIPS***

5.2.3.4 Base de dados ***20Newsgroup***

5.2.3.5 Base de dados ***IG***

5.2.3.6 Base de dados ***IG toy***

5.2.4 Experimentos qualitativos

Estes experimentos mostram a aplicabilidade dos algoritmos no domínio de mineração de textos, com ênfase nas informações que são possíveis de extrair dos modelos gerados por algoritmos de coclustering baseados em fatoração de matrizes: *ONMTF*, *FNMTF*, *OvNMTF* e *BinOvNMTF*.

Essa análise qualitativa se faz importante, para mostrar que os algoritmos são capazes de encontrar tópicos (grupos de palavras), e possivelmente, explicar os grupos de notícias formados, ou até mesmo, realizar rotulação dos grupos de notícias formados.

Foi necessária a construção de um estudo de caso para avaliar os algoritmos de Coclustering baseados em Fatoração de Matrizes. O estudo de caso é composto por uma análise dos coclusters de palavras correlacionando-os com os coclusters de notícias.

Para isso, foi usado o conjunto de dados ***IG toy***, que foi construído, justamente, para fazer esse tipo de análise.

O portal IG⁷ é um dos maiores portais de notícias brasileiro ([SITES..., 2016](#)) que é composto por sites importantes como o noticiário Último Segundo, o iG Gente, o iG Esportes, a TV iG, o iG Economia, o Delas e etc. Cada um desses sites é caracterizado como um canal que contém notícias de um determinado assunto macro.

O conjunto de dados ***IG toy*** é composto por três desses canais: esportes, que contém notícias, principalmente, dos esportes mais populares no Brasil, como o futebol e o UFC; jovem, que contém notícias mais interessantes para o público jovem em geral,

⁷ <http://www.ig.com.br/>

como notícias sobre filmes, esportes e músicas voltados para o público jovem; arena, que compõem notícias de games, novidades de todos os tipos de games, consoles e coberturas de eventos de games. Sendo 100 notícias para cada canal para compor o conjunto de dados, totalizando 300 notícias.

Esses canais foram escolhidos para formar o conjunto de dados ***IG toy*** por serem de assuntos similares, notícias do canal jovem podem ter semelhanças com notícias do canal esportes, como notícias sobre surfe, por exemplo, notícias do canal arena podem ser similares com notícias do canal de esportes, por existirem games que simulam esportes, ou até mesmo, possuírem similaridades com notícias do canal jovem, pois games é um assunto ligado ao público jovem na sua maioria. Essa escolha torna possível verificar como os algoritmos tratam essa intersecção entre palavras nas notícias.

As subseções a seguir irão mostrar como cada um dos algoritmos de coclusterização baseados em fatoração de matrizes: são úteis para análise de tópicos e palavras no conjunto de dados ***IG toy***.

5.2.4.1 Análise de dados utilizando *ONMTF*

O algoritmo *ONMTF* é capaz de encontrar coclusters de notícias e coclusters de palavras, além disso, cada cocluster de notícias é relacionado com os coclusters de palavras por um fator, assim como cada cocluster de palavras está relacionado com os coclusters de notícias pelo mesmo fator. Estes fatores são extraídos da matriz S .

Isso significa que o algoritmo permite que palavras ou notícias estejam presentes em múltiplos coclusters, com um fator de pertinência para cada cocluster, ou seja, é possível realizar *soft clustering*. Uma abordagem para realizar *hard coclustering* é atribuir uma notícia ou palavra ao cocluster com maior pertinência, a qual foi utilizada nos experimentos quantitativos (Subseção 5.2.3).

Note também que cada notícia está presente em cada um dos coclusters com um fator associado, assim como cada palavra está presente em cada um dos coclusters com um fator associado.

Para as análises construídas com o *ONMTF* foi utilizado o modelo que obteve a melhor taxa segundo a métrica *Rand Index* no experimento descrito na seção 5.2.3.6, que foi o modelo com $k = 3$ e $l = 3$ com a representação TF-IDF normalizado.

A figura 12 exemplifica as informações que um modelo gerado pelo algoritmo *ONMTF* provê. A notícia “Avaliação do FIFA 15 por um jogador fanático” foi usada como exemplo. O resultado da coclusterização de notícias foi que dos três coclusters de notícias, a notícia pertence ao cocluster (rotulado manualmente) esportes com um fator equivalente à 40%, também pertence ao cocluster rotulado como arena com um fator equivalente à 60%, e não pertence ao cocluster rotulado como jovem. Cada cocluster de notícias é formado pela combinação dos coclusters de palavras com os fatores que indicam a relação entre coclusters de notícias e coclusters de colunas, denotado pelas linhas que os conecta. Cada cocluster de palavras é ilustrado pelas palavras mais relevantes que o compõem, ou seja, as palavras que contém os maiores fatores (representado em porcentagem) para aquele determinado cocluster.

Ainda sobre a Figura 12, note que as palavras do corpo da notícia foram coloridas de acordo com as cores dos coclusters de palavras as quais pertencem. É possível perceber que existem mais palavras que caracterizam a notícia como sendo sobre o assunto games, o que vai de acordo com a coclusterização realizada.

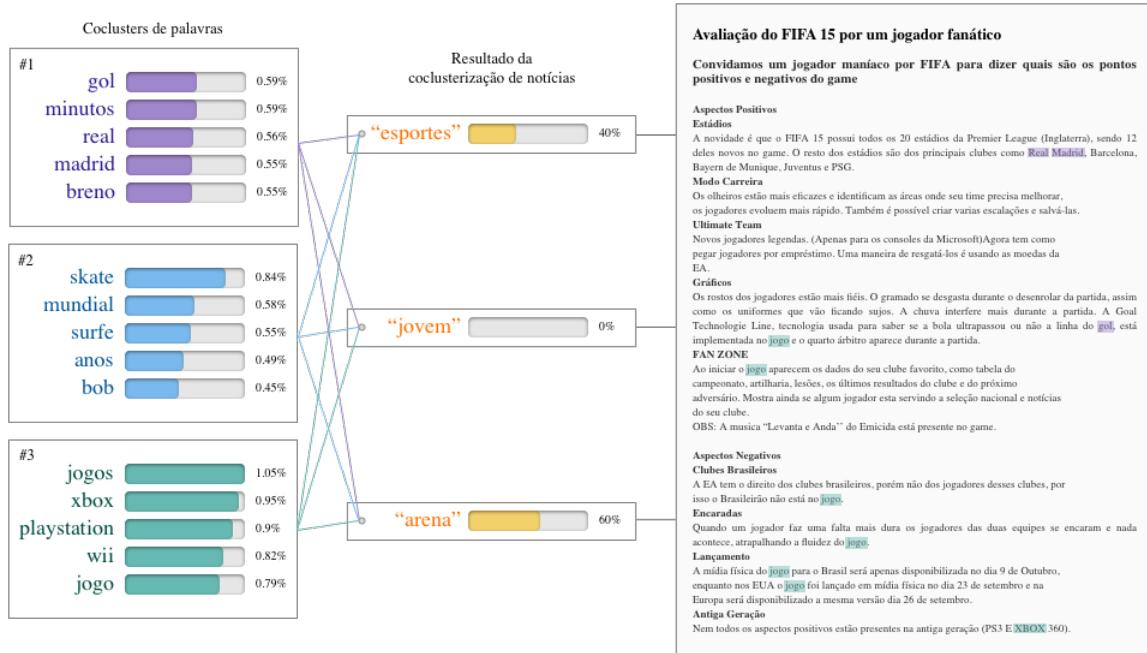


Figura 12 – Exemplo de uma notícia do canal arena e sua disposição quanto aos coclusters de notícias e palavras.

Os fatores que conectam os coclusters de notícias com os coclusters de palavras podem ser observados na matriz S . A matriz S que foi usada nesta análise pode ser observada na Tabela ???. Note que os valores estão normalizados para que a soma dos

elementos de cada linha tenha resultado da soma igual a 1. A normalização foi feita para tornar claro que cada cocluster de palavras foi usado pelo algoritmo para caracterizar um grupo de notícias, os valores mostram que o cocluster de notícias “arena” está associado com um maior fator ao cocluster de palavras #3, o cocluster de notícias “esporte” está associado com um maior fator ao cocluster de palavras #1, e o cocluster de notícias “jovem” está associado com um maior fator ao cocluster de palavras #2.

Tabela 3 – Matriz S para $ONMTF$ com $k = l = 3$.

-	Cocluster de palavras #1	Cocluster de palavras #2	Cocluster de palavras #3
Cocluster de notícias “arena”	0.0068	0.0188	0.9744
Cocluster de notícias “esporte”	0.9604	0.0348	0.0048
Cocluster de notícias “jovem”	0.0444	0.9324	0.0232

Os grupos de palavras foram summarizados através da visualização em *word cloud*, em que o tamanho das palavras é definido pelo seu fator de pertinência ao cocluster correspondente. A Figura 13 mostra essa visualização contendo as 100 palavras com maior fator para cada cogrupo.



Figura 13 – Visualização *word cloud* de palavras para cada cocluster de palavras gerados pelo algoritmo QNMF.

A Figura 13 mostra claramente que cada cocluster de palavras ficou responsável por caracterizar cada um dos três coclusters de notícias. O cocluster de palavras #1 contém palavras sobre esportes, principalmente sobre futebol, como gol, minutos, nomes de seleções, times de futebol e campeonatos. Note que nesse cocluster aparece a palavra jogo, de tamanho pequeno, localizada entre as letras i e n da palavra minutos, que também aparece no cocluster de palavras #3, porém, de tamanho claramente maior. No cocluster de palavras #2 aparecem palavras de esportes, principalmente sobre surfe e skate, como nomes de atletas desses esportes e campeonatos. O cocluster de palavras #3, como esperado, contém palavras ligadas ao assunto games, como jogo, nomes de consoles (xbox, playstation e wii), nomes de grandes empresas do ramo e nomes de games.

5.2.4.2 Análise de dados utilizando *FNMTF*

5.2.4.3 Análise de dados utilizando *OvNMTF*

O algoritmo *OvNMTF* é capaz de encontrar coclusters de notícias e coclusters de palavras, semelhante aos outros.

Tabela 4 – Matriz S para *OvNMTF* com $k = 3$ e $l = 6$.

-	#1	#2	#3	#4	#5	#6
Cocluster de notícias “arena”	0.1465	0.2516	0.0535	0.3294	0.1499	0.0691
Cocluster de notícias “esporte”	0.0949	0.0943	0.2239	0.1287	0.2878	0.1703
Cocluster de notícias “jovem”	0.0381	0.2545	0.3324	0.2220	0.0932	0.0597

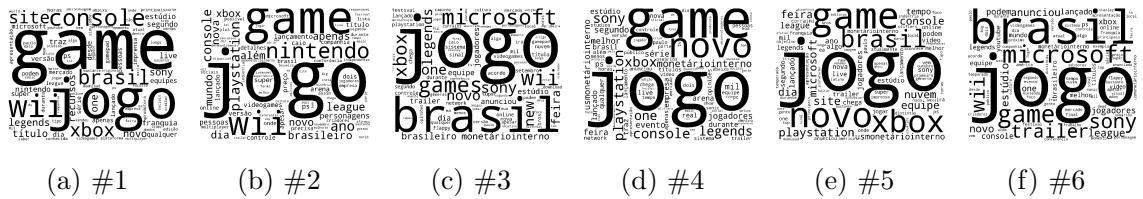


Figura 14 – Visualização *word cloud* de palavras para cada cocluster de palavras do cocluster de notícias “arena”, gerados pelo algoritmo *OvNMTF*.



Figura 15 – Visualização *word cloud* de palavras para cada cocluster de palavras do cocluster de notícias “esporte”, gerados pelo algoritmo *OvNMTF*.



Figura 16 – Visualização *word cloud* de palavras para cada cocluster de palavras do cocluster de notícias “jovem”, gerados pelo algoritmo *OvNMTF*.

5.2.4.4 Análise de dados utilizando *BinOvNMTF*

6 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 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 e 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*, que incluem algoritmos para a descoberta de padrões, tendências e conhecimentos por meio de técnicas e algoritmos; *componentes de apresentação*, que incluem interfaces para o usuário, apresentando visualizações dos conhecimentos gerados na etapa anterior; e *técnicas de refinamento*, também descritas como uma fase de pós-processamento, que incluem métodos para filtrar informações redundantes.

6.1 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).

6.1.1 Representação textual

Para a estruturação dos textos é necessário a definição da representação textual dos documentos. O vetor de termos, ou *Vector Space Model* (SALTON; WONG; YANG, 1975), é a representação clássica usada para representar documentos textuais (SEBASTIANI, 2002; LOPS; GEMMIS; SEMERARO, 2011). Cada dimensão desse vetor está associada a

um termo, sendo que todas as dimensões representam todos os termos do conjunto de documentos. Formalmente, há um conjunto de documentos $D = \{d_1, d_2, \dots, d_n\}$, em que d_i representa um documento e n o número total de documentos, e um conjunto de termos $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$, em que t_j representa um termo e m o número de termos presentes em todos os documentos. Representando a frequência de um termo pelo número de vezes que t_j aparece em um documento d_i , denotado por $ft(t_j, d_i)$, o vetor de termos pode ser construído e representado da seguinte forma: $\vec{vt}_{d_i} = (TF(t_1, d_i), TF(t_2, d_i), \dots, TF(t_m, d_i))$. Salton, Wong e Yang (1975) argumentam que a representação textual de documentos em vetor de termos é suficiente para separar documentos. Ao invés de frequência de termos, também é usado, a representação binária (SEBASTIANI, 2002), ou seja, t_j aparecendo em d_i corresponde à entrada 1 na dimensão j em \vec{vt}_{d_i} . Há também outros métodos para representação textual, como *n-gramas* e *ontologias* (LOPS; GEMMIS; SEMERARO, 2011).

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

$$\begin{aligned} TF-IDF(t_j, d_i) &= TF(t_j, d_i) \cdot IDF(t_j) \\ TF-IDF(t_j, d_i) &= TF(t_j, d_i) \cdot \left(\log_2 \frac{n}{DF(t_j) + 1} \right) \end{aligned} \quad (32)$$

em que $IDF(t_j)$ representa a frequência de documentos inversa do termo t_j , e $DF(t_j)$ a frequência de documentos que contém t_j . Essa normalização faz com que a frequência dos termos que aparecem em muitos documentos seja reduzida, e a frequência dos termos que aparecem em alguns raros documentos seja aumentada, com um fator de \log_2 .

6.1.2 Tokenização

Para realizar a estruturação de textos e representar os textos dos documentos em vetores de termos, o primeiro processo a ser realizado é a *tokenização*, que cria um dicionário de termos 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](#)).

6.1.3 Filtragem

Métodos de filtragem têm a função de retirar termos do conjunto \mathcal{T} 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 determinados termos de \mathcal{T} a partir de uma lista, é chamada de *stopwords*. Também são usadas outras técnicas, como a eliminação de termos com a frequência muito alta ou muito baixa.

6.1.4 Stemming

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

6.1.5 Redução de Dimensionalidade

A representação em vetor de termos pode resultar em vetores esparsos num espaço de alta dimensão, que pode fazer com que algoritmos sofram do problema de *Maldição de Dimensionalidade*, que diz respeito à perda de densidade em espaços de alta dimensão, isto significa que medidas de distância se tornam incapazes de detectar padrões em um conjunto de dados ([HAYKIN, 2008](#)). 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 termos, diminuindo a dimensionalidade e esparsidade, superando o problema de *Maldição de Dimensionalidade*.

7 Sistemas de Recomendação baseados em Conteúdo e Aprendizado de Máquina

Os Sistemas de Recomendação baseados em Conteúdo (SRsbC) 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, extraíndo 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.

7.1 Arquitetura de Sistemas de Recomendação baseados em Conteúdo

Lops, Gemmis e Semeraro (2011) propõem uma arquitetura para o desenvolvimento de SRsbC, a qual separa o processo de recomendação em três fases (Figura 17). O analisador de conteúdo tem como entrada os itens não estruturados, assim, através de técnicas de Mineração de Dados, os itens são representados de forma estruturada. Uma representação comum, no contexto de conteúdo textual, é o *Vector Space Model* (ADOMAVICIUS; TUZHILIN, 2005; LOPS; GEMMIS; SEMERARO, 2011; JANNACH et al., 2011) (Seção 6.1). Onde representa-se o vetor de termos do documento d_i por $\vec{v}_{t_{d_i}} = (ft(t_1, d_i), ft(t_2, d_i), \dots, ft(t_m, d_i))$, em que t_i é um termo, para então usar a representação de TF-IDF.

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 é dado por $\{(d_1, r_{u,d_1}), \dots, (d_j, r_{u,d_n})\}$, sendo r_{u,d_n} o quanto o usuário u gostou do documento d_n , seja pela manifestação explícita, por exemplo em que o usuário avaliou o documento, 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 utilizando de técnicas e algoritmos de 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 a fase Componente de Filtragem, que basicamente recebe a saída do classificador, seleciona os itens mais relevantes para os usuários, e apresenta uma lista de recomendações. Geralmente, essa lista é ordenada por um *score*, apresentando os itens mais relevantes (*top N*).

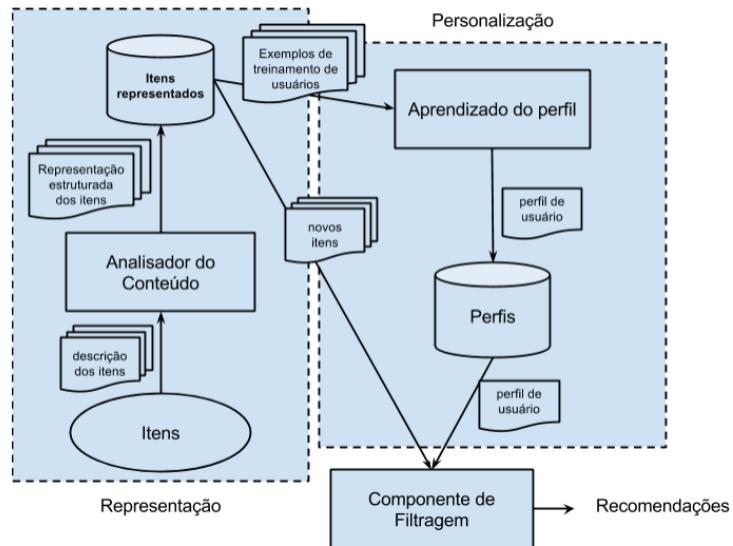


Figura 17 – Arquitetura de um CBRS

Essa arquitetura apresentada foi adaptada de Lops, Gemmis e Semeraro (2011), para maior compreendimento e maior adequação com esse trabalho. As subseções 7.1.1, 7.1.2 e 7.1.3 fazem uma revisão da literatura das técnicas de Aprendizado de Máquina aplicadas aos módulos de um SRsbC.

7.1.1 Analisador de Contexto

O Analisador de Contexto tem como função representar o item de uma maneira estruturada, a Figura 18 explica como é construída essa etapa. A entrada são os itens, que são pré-processados, para então aplicar técnicas de Aprendizado de Máquina (Redução de Dimensionalidade ou Tarefas de Aprendizado de Máquina). Esta etapa compreende o foco deste trabalho.

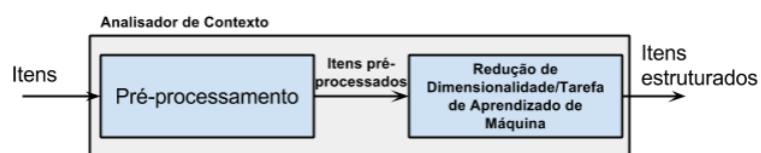


Figura 18 – Analisador de Contexto.

Na etapa de pré-processamento, é comum o uso da representação TF-IDF, sendo que existem estudos que propõem outros métodos de representação (CAPELLE et al., 2012; MOERLAND et al., 2013): SF-IDF (*Semantics Frequency-Inverse Document Frequency*) e SF-IDF+, que obtiveram melhores resultados nos testes apresentados, além disso, Cleger-Tamayo, Fernández-Luna e Huete (2012) considerou o uso do TF-IDF, mas não resultou em melhores performances. No entanto, em um SR no contexto de artigos científicos (BEEL et al., 2013), foram realizados diversos testes considerando diversos parâmetros e configurações diferentes de representação, resultados mostraram que a configuração com TF-IDF performou melhor que outras. Outras técnicas de Mineração de Dados também são utilizadas, como *stopwords*, lematização e descarte de termos com frequência abaixo de um limiar.

Na etapa de Redução de Dimensionalidade/Tarefa de Aprendizado de Máquina faz-se o uso extensivo dos algoritmos: LSA (*Latent Semantic Analysis*) (TARAGHI et al., 2013; DOMINGUES et al., 2012; SPAETH; DESMARAIS, 2013), LSI (*Latent Semantic Indexing*) (SAAYA et al., 2013), LDA (*Latent Dirichlet Allocation*) (TANTANASIRIWONG, 2012; QU; LIU, 2012; WANG et al., 2012; VAZ; Martins de Matos; MARTINS, 2012), que pode ser visto como uma extensão de Biclusterização (SKILLICORN, 2012), e Variáveis Latentes (CLEGER-TAMAYO; FERNÁNDEZ-LUNA; HUETE, 2012), que podem ser vistos por resolver uma tarefa de clusterização, que é agrupar termos em grupos que são chamados de tópicos (WANG et al., 2012). O desafio é escolher o número de tópicos, por isso, grande parte dos trabalhos que fazem o uso dessas técnicas, realizam testes variando o número de tópicos. Esses testes geralmente mostram que esses algoritmos ajudam na performance da recomendação (CLEGER-TAMAYO; FERNÁNDEZ-LUNA; HUETE, 2012; TANTANASIRIWONG, 2012; SAAYA et al., 2013; SPAETH; DESMARAIS, 2013; VAZ; Martins de Matos; MARTINS, 2012).

Na representação estruturada dos itens, alguns estudos propõem representações no contexto de notícias e artigos científicos (BIELIKOVA; KOMPAN; ZELENIK, 2012; LOPS et al., 2013): são diferenciadas as relevâncias de cada um dos atributos textuais (exemplo título, conteúdo, categoria e etc.), por meio de pesagem. Estudos mostraram que com os pesos apropriados é possível melhorar a qualidade da recomendação. Atributos que não são textuais também são usados no contexto de notícias, como no SR de notícias para celular em (YEUNG; YANG; NDZI, 2012), que o tempo da notícia modifica o vetor que representa o item, fazendo a multiplicação por um fator α . o SR de livros (VAZ; Martins de

Matos; MARTINS, 2012) que tem como objetivo representar o estilo de escrita de cada autor, faz o uso de atributos como: o tamanho do documento, n-gramas e *vocabulary richness*.

7.1.2 Aprendizado do Perfil

Nesta etapa é onde o perfil do usuário é representado e aprendido pelo SR. Na maioria das vezes o perfil é representado por um vetor de documentos de tamanho k , que o usuário visitou ou apresentou um feedback positivo $d_{prefs} = \{d_{pref_1}, d_{pref_2}, \dots, d_{pref_k}\}$. Há outras formas de representação, como em Yeung, Yang e Ndzi (2012), que além da anterior, incorpora informações demográficas, tratando o problema de *user cold-start* em SRsbC. Vaz, Martins de Matos e Martins (2012) propõem uma representação para o perfil do usuário usando um método da área de Recuperação de Informação: algoritmo de *Rocchio*, onde cada documento d_{pref_i} é classificado pelo usuário em positivo ou negativo (como exemplo, gostou ou não gostou), assim o algoritmo faz uma mistura dos objetos positivos e negativos, com um peso diferente para cada tipo de objeto, obtendo um vetor. Então, esse vetor é comparado com vetores de itens (usando similaridade dos cossenos), para obter itens semelhantes. Variando os parâmetros, os autores chegaram na conclusão que, incorporando objetos negativos na representação do perfil para treinamento do algoritmo, piora a qualidade das recomendações.

O aprendizado do perfil do usuário é como um problema de classificação, onde o vetor de objetos é dado por $X = \{d_{pref_i}, y_i^{+-}\}$, sendo que y_i^{+-} representa o rótulo, ou seja, se d_{pref_i} é um item que o usuário gostou(+) ou não(−). Então, é treinado um classificador que irá classificar itens que o usuário ainda não consumiu, para saber se é um item que o usuário irá consumir/gostar. Diversos algoritmos são usados para resolver esse tipo de problema: Redes Bayesianas (YEUNG; YANG; NDZI, 2012; CLEGER-TAMAYO; FERNÁNDEZ-LUNA; HUETE, 2012), Naïve Bayes (LEE et al., 2012; SEMERARO et al., 2012), SVM (*Support Vector Machine*) (TANTANASIRIWONG, 2012; LEE et al., 2012).

Existem trabalhos que tratam o aprendizado do perfil com técnicas de Aprendizado Semi-Supervisionado, Lee et al. (2012) faz uso de comitê de máquinas para construir um modelo de Aprendizado de Máquina que classifica apenas classes positivas, visando classificar se o usuário de um e-commerce irá gostar ou não de um produto. Primeiramente, classifica objetos sem rótulo, para depois entrarem no modelo final que agrupa todos os objetos (SVM ou Naïve Bayes). Foi verificado que o SR proposto trata o problema de

poucos dados para a identificação do perfil do usuário, pois não necessita apenas de dados rotulados.

É possível tratar o problema de aprendizado do perfil como um problema de clusterização (DAVOODI; KIANMEHR; AFSHARCHI, 2012; BIELIKOVA; KOMPAN; ZELENIK, 2012). Davoodi, Kianmehr e Afsharchi (2012) apresentam um SR de especialistas, que representa o perfil dos usuários com semântica e constrói uma *Rede Social*, para então, usar o algoritmo de clusterização (*k-means*) para encontrar perfis de usuário. Além desse, (BIELIKOVA; KOMPAN; ZELENIK, 2012) apresenta um SR de notícias que faz o uso de clusterização hierárquica, tendo como medida de similaridade a similaridade dos cossenos e índice de jaccard. Com uma abordagem *bottom-up* de clusterização e uma estrutura de árvore binária, é realizado a clusterização das notícias: as folhas representam as notícias, e os nós pais, clusters que representam temas das notícias. O usuário desse sistema é representado por caminhos nesta árvore construída, podendo ser recomendados diversas notícias dentro de diversos tópicos, que podem surpreender o usuário, amenizando o problema de serendipidade em SRsbC.

7.1.3 Componente de Filtragem

Essa é a etapa mais simples, por ser na maioria das vezes, apenas uma filtragem das recomendações já calculadas na etapa de Aprendizagem do Perfil, essa estratégia é apresentada em Cleger-Tamayo, Fernández-Luna e Huete (2012), Qu e Liu (2012), Wang et al. (2012), Davoodi, Kianmehr e Afsharchi (2012), Mannens et al. (2011), Semeraro et al. (2012). Outra estratégia simples é a determinação de um limiar (CAPELLE et al., 2012; LOPS et al., 2013), ou seja, os valores da lista de recomendação gerada são filtradas pelo limiar estabelecido. Nos estudos apresentados, foram encontrados muitos SRs Híbridos, que faziam uma outra abordagem para a filtragem, usando uma combinação dos métodos de Filtro Colaborativo e baseados em conteúdo (LOPS et al., 2013; QU; LIU, 2012; DOMINGUES et al., 2012; SPAETH; DESMARAIS, 2013; VAZ; Martins de Matos; MARTINS, 2012).

O trabalho desenvolvido por Bielikova, Kompan e Zelenik (2012) foi o único que apresentou uma estratégia diferente para o Componente de Filtragem, com a árvore binária montada, todas as notícias dos menores para os maiores grupos, que não foram lidas pelo usuário, foram separadas para a recomendação. Então, é construída uma matriz com as recomendações, sendo as linhas ordenadas pelos grupos menores para os grupos maiores, e

as colunas ordenadas pelas notícias mais recentes. Assim, cada coluna é transformada em um vetor, concatenando-os e formando uma lista que é apresentada para o usuário.

8 Proposta

A proposta desse projeto de mestrado envolve a aplicação de algoritmos de Biclusterização para o problema de recomendação baseado em conteúdo textual, com a hipótese de que as recomendações geradas, possam amenizar o problema da serendipidade, visto que técnicas de Biclusterização são capazes de encontrar clusters através da análise de subconjuntos de atributos.

Formalmente, os algoritmos de Biclusterização irão atingir o módulo de representação dos itens em um SRsbC, mais especificamente, na função de similaridade entre itens $s : I \rightarrow I \times I$, que pode ser capaz de encontrar itens similares, que poderiam não ser similares caso s fosse gerado por um algoritmo de clusterização, por exemplo. Então, espera-se que o fator de mudança na função de similaridade entre itens s , acrescente um fator de serendipidade que influencie na aproximação da função l_u , que representa o perfil do usuário u , e portanto, na lista de recomendações L_u , direcionadas à u , dado que $l_u : \mathcal{H}_u, s \rightarrow L_u$, sendo \mathcal{H}_u o subconjunto que representa o histórico de itens que u acessou.

Então, para a validação dessa hipótese, se faz necessária a definição de um contexto para servir de prova de conceito. Este contexto é referente à notícias presentes no portal iG¹, um portal de notícias brasileiro muito conhecido, com um volume de notícias bastante grande e com alguma estrutura de classificação de notícias já existente. Essas características conferem liberdade para a configuração de experimentos de diferentes naturezas, como experimentos considerando determinadas classes de notícias, tipos de notícias ou datas de publicação das notícias.

Como se tratam de notícias, é necessário fazer o uso de técnicas de Mineração de Texto para representar o conteúdo textual de notícias de maneira estruturada, possibilitando a construção de um corpus de notícias, que será usado para a validação da hipótese. Para então, realizar a implementação de algoritmos de Biclusterização, que formarão biclusters para recomendar aos usuários, com base nos seus respectivos históricos \mathcal{H}_u .

Também, o portal iG possui informações históricas e anônimas referentes ao registro de navegação de usuários. Esse registro permitirá a construção de um conjunto de dados de preferências, que poderá ser usado para realização de *testes offline* para validar a prova de conceito, e portanto, a hipótese.

¹ <http://ig.com.br/>

Os passos da proposta são summarizados na Figura 19, onde se tem a fase de aquisição de notícias que comporá o corpus de notícias, para entrada na fase de pré-processamento, que utilizará de técnicas de Mineração de Texto para a estruturação do conteúdo textual, realizando diversas representações (TF-IDF, TF-IDF normalizado e *n-grams*). Isso possibilitará a criação de biclusters através de algoritmos de Biclusterização, para então, dado o histórico de itens que o usuário navegou, analisar os biclusters que contém esses itens, e construir a lista de recomendação com base em outras notícias presentes nesses biclusters. Assim, será possível avaliar a qualidade dos biclusters gerados, em relação às notícias recomendadas e às informações históricas. Verificando também, como a estratégia se adequa à serendipidade, através da comparação das recomendações geradas pela estratégia por recomendações geradas através da técnica de filtro colaborativo.

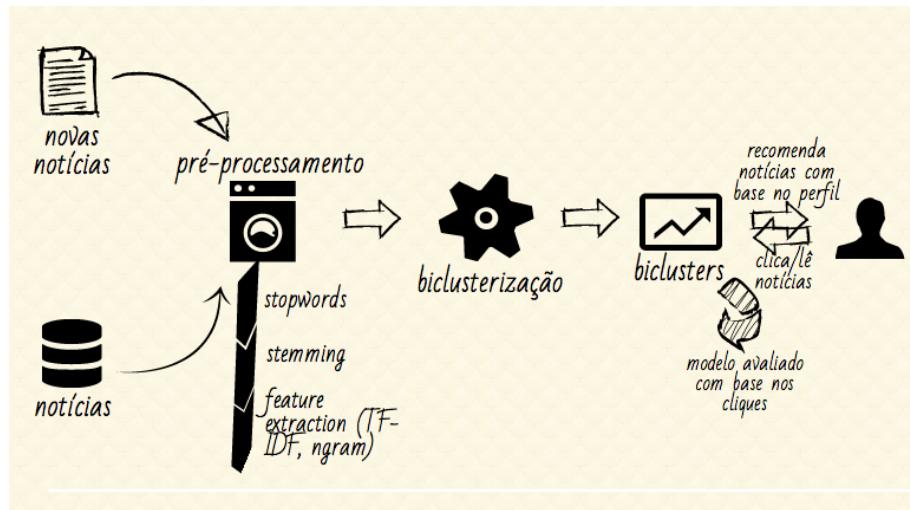


Figura 19 – Sumarização da proposta.

Com o objetivo de especificar ainda mais as etapas da proposta e trazer informações sobre o desenvolvimento das mesmas, as seções Revisões Bibliográficas (Seção 8.1), Construção das Bases de Dados (Seção 8.2), Estratégias para Validação e Extensões (Seção 8.3) e Cronograma - Próximos Passos (Seção ??) são apresentadas.

8.1 Revisões Bibliográficas

Se fez necessária, primeiramente, de uma análise exploratória para obter noções básicas da área de SRs, que foi realizada através da leitura de livros e artigos do tipo revisão bibliográfica, o que permitiu, em seguida, o aprofundamento na área de SRs, por meio da RS em Aprendizado de Máquina e SRsbC Textual.

A RS comprehende estudos entre 2003 à 2014 que fazem referência ao estudo do estado da arte ou estado da prática sobre SRsbC textuais e apresentem alguma solução com o uso de técnicas de Aprendizado de Máquina. Os estudos foram selecionados através da busca sistemática nas seguintes bases de dados: Scopus², ISI web of Science (WoS)³, IEEE Xplore⁴, ACM Digital Library⁵ e SpringerLink⁶. Para cada busca foi escrito um protocolo (veja um exemplo no Apêndice ??). Para realizar a filtragem dos estudos encontrados foi utilizada uma abordagem *Tollgate*: três pesquisadores aplicam os critérios de inclusão de maneira independente, e então, o consenso na decisão final é decidido pela maioria. Sendo assim, a fim de extrair os dados dos estudos, é realizada a leitura dos estudos incluídos e seus dados sumarizados em uma tabela (veja um exemplo no Apêndice ??).

Com um total de 304 estudos após a identificação nas bases de dados e remoção de duplicatas, foi feita a leitura e extração de dados de 18 estudos, que compreendem os estudos publicados entre os anos de 2012 à 2014. A finalização da RS se dará após aplicação da filtragem e extração de dados para os demais anos.

Para aplicação dos algoritmos de Biclusterização, foi realizado um levantamento do referencial teórico, através da análise exploratória de livros, artigos do tipo revisão bibliográfica e artigos que se referem à criação dos algoritmos (FRANCA, 2010; CHENG; CHURCH, 2000; TANAY; SHARAN; SHAMIR, 2005; MADEIRA; OLIVEIRA, 2004; SANTAMARÍA; MIGUEL; THERÓN, 2007; KLUGER et al., 2003; PRELIĆ et al., 2006). Isso possibilitou a geração de conjuntos de dados sintéticos (Seção 2.1), e implementação do algoritmo de Cheng e Church (2000) nesses conjuntos, disponibilizado em <https://github.com/lucasbruni/alti/biclustering-experiments>.

Além disso, também foi necessária a realização de uma análise exploratória das áreas de Mineração de Texto, para que fosse possível a estruturação dos conteúdos textuais presentes nas notícias (Seção 8.2).

² <http://www.scopus.com/>

³ <http://apps.webofknowledge.com/>

⁴ <http://ieeexplore.ieee.org>

⁵ <http://dl.acm.org>

⁶ <http://link.springer.com/>

8.2 Construção das Bases de Dados

8.2.1 Corpus iG

A extração das notícias do portal iG foi realizada através da implementação de um *web crawler* utilizando a linguagem python⁷. 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 categoria (canal).

O corpus iG é composto por um conjunto de notícias do portal iG⁸ $\mathcal{N} = \{I_1, \dots, I_m, \dots, I_M\}$, em que cada notícia I_m é representada pela tupla (*permalink*, *título*, *subtítulo*, *corpo*, c_i), em que *permalink* é o endereço eletrônico fixo da notícia, e, c_i um elemento do conjunto de canais $\mathcal{C} = \{gente, ultimosegundo, delas, economia, esporte, saude, igay, deles, tecnologia, igirl, jovem, arena, luxo\}$, onde cada canal representa um assunto ou categoria de notícias.

O número total de notícias do corpus é $M = 4\,593$, com mais de 250 caracteres no *corpo*, no período de 02 de Janeiro de 2012 à 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 está associada com um canal, foi coletada a distribuição de notícias por canal (Tabela 1).

Tabela 5 – Distribuição de notícias por canal (ci) do corpus iG.

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

⁷ <https://www.python.org/>

⁸ <http://www.ig.com.br/>

Analizando 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, uma hipótese é que isso se deve à idade e popularidade do canal, por exemplo, os canais *luxo* e *deles* são muito mais recentes e menos populares que o *ultimosegundo*.

8.2.2 Pré-processamento do corpus iG

Com o corpus iG criado, o intuito da fase de pré-processamento é representar as notícias de maneira estruturada, para isso foram utilizadas técnicas de Mineração de Texto (Seção 6). Foi criado um *pipeline* para o pré-processamento das notícias que contou com as seguintes etapas:

8.2.3 Base de cliques iG

A base de dados de cliques iG, doada para a realização deste trabalho, é composta por um conjunto usuários anônimos $U = \{u_1, \dots, u_n, \dots, u_N\}$ 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 h_{u_n, I_m}^t , um clique em uma notícia I_m que foi dado por u_n em um dado momento do tempo t . Assim, se considerar cada clique h_{u_n, I_m}^t como uma preferência do usuário u_n por n , é possível construir a matriz de preferências $U \times I$ (Seção ??), no contexto, $\mathcal{I} = \mathcal{N}$.

A base de cliques iG é composta de 487 487 395 cliques, com notícias coletadas, aproximadamente, do período de abril de 2013 à novembro de 2014. Essa base de dados tem tamanho total, sem compressão, de 100GB, o que dificulta a sua mineração. No entanto, pretende-se usar apenas as notícias que compõem o corpus iG.

8.3 Estratégias para Validação e Extensões

8.3.1 Estratégias para Validação

A primeira forma de avaliação do trabalho, consistirá na análise dos biclusters criados através de medidas de avaliação internas, onde usa-se os próprios biclusters, juntamente com métricas de qualidade e/ou estabilidade, para avaliar as soluções geradas. A métrica estudada até o momento para avaliação interna é a métrica de consistência ([SANTAMARÍA; MIGUEL; THERÓN, 2007](#)), que verifica se os biclusters encontrados correspondem com a definição de bicluster.

Também serão realizados *testes offline*, utilizando a base de cliques iG, para verificar se o modelo proposto, é capaz de oferecer recomendações consistentes, comparando as recomendações geradas para um usuário u com o histórico deste mesmo usuário, utilizando as métricas de precisão e revocação, variando o número de recomendações oferecidas. Segundo [Jannach et al. \(2011\)](#), essa estratégia é a que mais se adequa quando a recomendação é em forma de lista que é apresentado à um usuário. Este método também pode ser interpretado como uma avaliação externa dos biclusters.

Ainda utilizando a base de cliques iG, será realizada outra avaliação com *testes offline*, comparando as recomendações obtidas com recomendações geradas por um filtro colaborativo, que será, a princípio, implementado neste trabalho, fazendo uma comparação da recomendação gerada com a estratégia proposta e da recomendação gerada com o filtro colaborativo. Serão utilizadas as mesmas métricas presentes na avaliação anterior.

8.3.2 Possíveis Extensões

Sobre a geração da lista de recomendações direcionadas à um usuário (L_{u_n}), uma forma mais simples, já descrita neste trabalho, é verificar os itens do histórico do usuário, buscando os biclusters aos quais esses itens pertencem, e dentro desses biclusters procurar por notícias para a construção da recomendação. Uma possível abordagem para a geração de L_{u_n} é observar o histórico de itens que o usuário u_n interagiu, como um modelo escondido de Markov, em que deseja-se descobrir a probabilidade do próximo item que o usuário vai consumir: $P(h_{u_n, I_{m+1}} | h_{u_n, I_1}, \dots, h_{u_n, I_m})$, este modelo é chamado, especificamente, de *Variable Length Markov Chains* ([MURPHY, 2012](#)). Ainda, é possível combinar os resultados

dos biclusters com o modelo escondido de Markov, fazendo com que este calcule a probabilidade do próximo bicluster em que o usuário irá navegar.

Outra possível extensão, diz respeito ao uso de *Ensemble* de Clusterização ([STREHL; GHOSH, 2003](#)), como forma de substituir os algoritmos de Biclusterização, visto que estes podem considerar subconjuntos de atributos para a formação de clusters. Assim, é possível fazer comparações com uma técnica semelhante à originalmente proposta: Biclusterização.

Referências⁹

- 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. Citado na página 68.
- BAZARAA, M.; SHERALI, H.; SHETTY, C. *Nonlinear Programming: Theory and Algorithms*. [S.l.]: Wiley, 2006. ISBN 9780471787761. Citado na página 34.
- BEEL, J. et al. Introducing docear's research paper recommender system. In: *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries - JCDL '13*. New York, New York, USA: ACM Press, 2013. p. 459. Citado na página 70.
- BIELIKOVA, M.; KOMPAN, M.; ZELENIK, D. Effective hierarchical vector-based news representation for personalized recommendation. *Computer Science and Information Systems*, COMSIS CONSORTIUM, v. 9, n. 1, p. 303–322, jan. 2012. Citado 2 vezes nas páginas 70 e 72.
- CABANES, G.; BENNANI, Y.; FRESNEAU, D. Enriched topological learning for cluster detection and visualization. *Neural Networks*, v. 32, p. 186–195, 2012. Citado na página 29.
- 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. Citado 3 vezes nas páginas 68, 70 e 72.
- CHENG, Y.; CHURCH, G. M. Biclustering of expression data. In: *Procedures of the 8th ISMB*. [S.l.]: AAAI Press, 2000. p. 93–103. Citado 2 vezes nas páginas 28 e 76.
- CLEGER-TAMAYO, S.; FERNÁNDEZ-LUNA, J.; HUETE, J. Top-n news recommendations in digital newspapers. *Knowledge-Based Systems*, v. 27, p. 180–189, 2012. Citado 3 vezes nas páginas 70, 71 e 72.
- DAVOODI, E.; KIANMEHR, K.; AFSHARCHI, M. A semantic social network-based expert recommender system. *Applied Intelligence*, v. 39, n. 1, p. 1–13, out. 2012. Citado na página 72.
- DING, C. et al. Orthogonal nonnegative matrix tri-factorizations for clustering. In: *In SIGKDD*. [S.l.]: Press, 2006. p. 126–135. Citado 2 vezes nas páginas 34 e 35.
- DOMINGUES, M. A. et al. Combining usage and content in an online recommendation system for music in the long tail. *International Journal of Multimedia Information Retrieval*, v. 2, n. 1, p. 3–13, nov. 2012. Citado 2 vezes nas páginas 70 e 72.
- EDELMAN, A.; ARIAS, T. A.; SMITH, S. T. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, v. 20, n. 2, p. 303–353, abr. 1999. ISSN 0895-4798. Disponível em: <<http://dx.doi.org/10.1137/S0895479895290954>>. Citado na página 38.
- FELDMAN, R.; SANGER, J. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge, MA, USA: Cambridge University Press, 2006. Hardcover. Citado na página 64.

⁹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

FRANCA, F. de. *Biclusterização na Análise de Dados Incertos*. Tese (Doutorado) — Universidade Estadual de Campinas, Campinas, SP, BR, 11 2010. Citado 3 vezes nas páginas 20, 26 e 76.

FRANÇA, F. de; ZUBEN, F. V. Finding a high coverage set of 5-biclusters with swarm intelligence. In: *Evolutionary Computation (CEC), 2010 IEEE Congress on*. [S.l.: s.n.], 2010. p. 1–8. Citado na página 29.

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. Citado na página 27.

GLOBERSON, A. et al. Euclidean embedding of co-occurrence data. *The Journal of Machine Learning Research*, MIT Press Cambridge, MA, USA, v. 8, p. 2265–2295, 2007. Citado na página 56.

HAN, J.; KAMBER, M. *Data mining: Concepts and Techniques*. 2. ed. [S.l.]: Morgan Kaufmann San Francisco, Calif, USA, 2006. Citado na página 19.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. Citado na página 67.

HAYKIN, S. *Neural Networks and Learning Machines (3rd Edition)*. 3. ed. [S.l.]: Prentice Hall, 2008. Hardcover. Citado na página 66.

HO, N.-D. *Nonnegative Matriz Factorization Algorithms and Applications*. Tese (Doutorado) — Université Catholique de Louvain, Louvain-la-Neuve, Belgique, 6 2008. Citado na página 20.

HOCHREITER, S. et al. Fabia: factor analysis for bicluster acquisition. *Bioinformatics*, v. 26, n. 12, p. 1520–1527, 2010. Citado 3 vezes nas páginas 25, 29 e 30.

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. Citado 2 vezes nas páginas 57 e 64.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: A review. *ACM Computing Surveys*, ACM, v. 31, p. 264 – 323, September 1999. Citado na página 19.

JANNACH, D. et al. *Recommender Systems An Introduction*. [S.l.]: Cambridge University Press, 2011. Citado 2 vezes nas páginas 68 e 79.

KLUGER, Y. et al. Spectral biclustering of microarray data: Co-clustering genes and conditions. *Genome Research*, v. 13, n. 4, p. 703–716, 2003. Citado na página 76.

LEE, D. D.; SEUNG, H. S. Learning the parts of objects by nonnegative matrix factorization. *Nature*, v. 401, p. 788–791, 1999. Citado na página 21.

LEE, D. D.; SEUNG, H. S. Algorithms for non-negative matrix factorization. In: *NIPS*. [s.n.], 2000. p. 556–562. Disponível em: <citeseer.ist.psu.edu/lee01algorithms.html>. Citado na página 21.

LEE, Y.-H. et al. A cost-sensitive technique for positive-example learning supporting content-based product recommendations in b-to-c e-commerce. *DECISION SUPPORT SYSTEMS*, ELSEVIER SCIENCE BV, v. 53, n. 1, p. 245–256, abr. 2012. Citado na página 71.

LONG, B.; ZHANG, Z. M.; YU, P. S. *Co-clustering by block value decomposition*. [S.l.]: ACM Press, 2005. 635–640 p. Citado 3 vezes nas páginas 31, 32 e 34.

LOPS, P. et al. Content-based and collaborative techniques for tag recommendation: An empirical evaluation. *Journal of Intelligent Information Systems*, v. 40, n. 1, p. 41–61, 2013. Citado 2 vezes nas páginas 70 e 72.

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. Citado 5 vezes nas páginas 57, 64, 65, 68 e 69.

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. Citado 4 vezes nas páginas 26, 27, 44 e 76.

MANNENS, E. et al. Automatic news recommendations via aggregated profiling. *Multimedia Tools and Applications*, v. 63, n. 2, p. 407–425, jul. 2011. Citado na página 72.

MINER, G. et al. *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*. 1st. ed. [S.l.]: Academic Press, 2012. Citado na página 66.

MOERLAND, M. et al. Semantics-based news recommendation with sf-idf. In: *ACM International Conference Proceeding Series*. Madrid: [s.n.], 2013. Citado na página 70.

MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. [S.l.]: The MIT Press, 2012. Citado 2 vezes nas páginas 66 e 79.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 47.

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. Citado 2 vezes nas páginas 29 e 76.

QU, Z.; LIU, Y. User participation prediction in online forums. In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012. (EACL '12), p. 367–376. Citado 2 vezes nas páginas 70 e 72.

SAAYA, Z. et al. The curated web: A recommendation challenge. In: *Proceedings of the 7th ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2013. (RecSys '13), p. 101–104. Citado na página 70.

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. Citado 4 vezes nas páginas 57, 58, 64 e 65.

- 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. Citado 4 vezes nas páginas 25, 29, 76 e 79.
- SEBASTIANI, F. Machine learning in automated text categorization. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 34, n. 1, p. 1–47, 2002. Citado 3 vezes nas páginas 57, 64 e 65.
- SEMERARO, G. et al. A folksonomy-based recommender system for personalized access to digital artworks. *Journal on Computing and Cultural Heritage*, v. 5, n. 3, p. 1–22, out. 2012. Citado 2 vezes nas páginas 71 e 72.
- SHAHNAZ, F. et al. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, v. 42, n. 2, p. 373 – 386, 2006. Citado na página 21.
- SITES mais acessados do Brasil. 2016. Disponível em: <<http://www.alexa.com/topsites/countries;1/BR>>. Citado na página 59.
- SKILLICORN, D. B. *Understanding High-Dimensional Spaces*. [S.l.]: Springer, 2012. I-IX, 1-108 p. Citado na página 70.
- SPAETH, A.; DESMARAIS, M. Combining collaborative filtering and text similarity for expert profile recommendations in social websites. In: CARBERRY, S. et al. (Ed.). *User Modeling, Adaptation, and Personalization*. [S.l.]: Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 7899). p. 178–189. Citado 2 vezes nas páginas 70 e 72.
- STREHL, A.; GHOSH, J. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, JMLR.org, v. 3, p. 583–617, March 2003. Citado na página 80.
- TANAY, A.; SHARAN, R.; SHAMIR, R. Biclustering algorithms: A survey. In: *Handbook of Computational Molecular Biology Edited by: Aluru S. Chapman & Hall/CRC Computer and Information Science Series*. [S.l.: s.n.], 2005. Citado 2 vezes nas páginas 27 e 76.
- TANTANASIRIWONG, S. A comparison of clustering algorithms in article recommendation system. In: *Proceedings of SPIE - The International Society for Optical Engineering*. Singapore: [s.n.], 2012. v. 8349. Citado 2 vezes nas páginas 70 e 71.
- TARAGHI, B. et al. Web analytics of user path tracing and a novel algorithm for generating recommendations in open journal systems. *Online Information Review*, v. 37, p. 672–691, 2013. Citado na página 70.
- TJHI, W.-C.; CHEN, L. Dual fuzzy-possibilistic coclustering for categorization of documents. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 17, p. 533 – 543, June 2009. Citado na página 21.
- VAZ, P. C.; Martins de Matos, D.; MARTINS, B. Stylometric relevance-feedback towards a hybrid book recommendation algorithm. In: *Proceedings of the fifth ACM workshop on Research advances in large digital book repositories and complementary media - BooksOnline '12*. New York, New York, USA: ACM Press, 2012. p. 13. Citado 3 vezes nas páginas 70, 71 e 72.

- WANG, H. et al. Fast nonnegative matrix tri-factorization for large-scale data co-clustering. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*. AAAI Press, 2011. (IJCAI'11), p. 1553–1558. ISBN 978-1-57735-514-4. Disponível em: <<http://dx.doi.org/10.5591/978-1-57735-516-8-IJCAI11-261>>. Citado na página 39.
- WANG, J. et al. Recommending flickr groups with social topic model. *Information Retrieval*, v. 15, n. 3-4, p. 278–295, abr. 2012. Citado 2 vezes nas páginas 70 e 72.
- WEISS, S. M.; INDURKHYA, N.; ZHANG, T. *Fundamentals of predictive text mining*. London; New York: Springer-Verlag, 2010. Citado 2 vezes nas páginas 57 e 66.
- XU, R.; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, IEEE, v. 16, p. 645 – 678, May 2005. Citado na página 19.
- XU, W.; LIU, X.; GONG, Y. Document clustering based on non-negative matrix factorization. In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2003. (SIGIR '03), p. 267–273. ISBN 1-58113-646-3. Disponível em: <<http://doi.acm.org/10.1145/860435.860485>>. Citado na página 21.
- 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. Citado na página 29.
- YEUNG, K. F.; YANG, Y.; NDZI, D. A proactive personalised mobile recommendation system using analytic hierarchy process and bayesian network. *Journal of Internet Services and Applications*, v. 3, n. 2, p. 195–214, jul. 2012. Citado 2 vezes nas páginas 70 e 71.
- YOO, J.; CHOI, S. Orthogonal nonnegative matrix tri-factorizations for co-clustering: multiplicative updates on stiefel manifolds. *Information Processing and Management*, v. 46, p. 559–570, 2010. Citado 6 vezes nas páginas 10, 21, 32, 36, 37 e 38.

Apêndice A – Tabela Workshop

Tabela Workshop

Contextualização / motivação	Algoritmos de coclusterização tem o objetivo de encontrar grupos em uma matriz de dados. Tais grupos são chamados coclusters por se caracterizarem como um subconjunto de linhas e colunas, e podem apresentar sob diferentes tipos de organização. Especialmente, organizações nas quais diferentes coclusters possuem linhas e/ou colunas em comum (organização com sobreposição) podem ser úteis para diversas aplicações, como mineração de textos, filtro colaborativo, recuperação de informação, etc.
Problema de pesquisa	Encontrar coclusters organizados com sobreposição em uma matriz de dados esparsa contendo valores reais positivos que representam dados de um domínio de aplicação.
Objetivo geral	Propor estratégias algorítmicas baseadas em Fatoração de Matrizes (FM) que sejam capazes de encontrar coclusters organizados com sobreposição em uma matriz esparsa de valores reais positivos.
Trabalhos relacionados	Biclustering algorithms for Biological Data Analysis: A Survey - Survey em algoritmos de coclusterização/biclusterização. Discute diferentes organizações de coclusters e algoritmos para cada uma delas. Fast Nonnegative Matrix Tri-Factorization for Large-Scale Data Co-clustering - Apresenta algoritmos de FM para coclusterização em dados binários. Não trata coclusters organizados com sobreposição. Orthogonal nonnegative matrix tri-factorization for co-clustering: Multiplicative updates on Stiefel manifolds - Apresenta uma survey com algoritmos baseados em FM juntamente com um novo algoritmo que encontra coclusters ortogonais. Estes algoritmos não encontram coclusters organizados com sobreposição.

Justificativa e relevância	Problemas reais se manifestam em domínios que podem apresentar coclusters organizados com sobreposição sobre matrizes de dados esparsas. Um exemplo é a mineração de textos aplicada à agrupamento de notícias. A representação vetorial de textos (as notícias), conhecidamente, gera matrizes esparsas. Notícias podem ser agrupadas em diferentes grupos a depender de quais de suas características são analisadas. Tais fatores caracterizam o problema sob estudo neste projeto.
Proposta para Solução	Estão sendo desenvolvidos algoritmos utilizando técnicas de FM com o intuito de resolver o problema de encontrar coclusters organizados com sobreposição em domínio nos quais os dados se apresentam como uma matriz esparsa.
Dados	Estão sendo usados: dados sintéticos; dados de notícias da base de dados Reuters; dados de trabalhos acadêmicos da Rochester University; dados de notícias de um grande portal brasileiro; e um conjunto de dados (movie lens) para teste também na área de filtro colaborativo.
Forma de validação	Provas de correção e convergência dos algoritmos desenvolvidos; comparação e experimentação com os algoritmos presentes nos trabalhos correlatos. Avaliação via índices de validação de clustering e coclusterização.
Limitações	FM supõe que os dados e características são separáveis linearmente. Esta restrição não é garantida em problemas reais, mas não impede que soluções baseadas em FM gerem resultados úteis.
Resultados esperados	Contribuições científicas: desenvolvimento de algoritmos inéditos baseados em FM para coclusterização organizados com sobreposição em matrizes esparsas positivas.