Paula Vitoria Martins Larocca, 769705
Departamento de Computação
UFSCar
paula.larocca@estudante.ufscar.br

Rafael Naoki Arakaki Uyeta, 800207
Departamento de Computação
UFSCar
rafael.uyeta@estudante.ufscar.br

Vinícius Gonçalves Perillo, 800219
Departamento de Computação
UFSCar
vinicius.perillo@estudante.ufscar.br

Resumo - O presente trabalho tem como objetivo aplicar e discutir os resultados de algoritmos para o agrupamento de uma base de dados.

I. OBJETIVO

O objetivo do estudo é utilizar métodos de agrupamento em uma base de dados e comparar os resultados obtidos a partir das classes pré-definidas no dataset.

II. BASE DE DADOS

A base de dados utilizada é referente à produtos *online*, a qual apresenta dois atributos: o nome do produto e uma descrição textual do mesmo. Tais produtos podem receber uma das 4 classificações: livro, brinquedo, maquiagem e game.

III. PROCESSAMENTO DA BASE

O tipo de informação contida na base é textual, dessa forma é preciso fazer um tratamento para que essa informação possa ser interpretada pelos algoritmos. Este tipo de estudo é um campo da área de processamento de dados chamado de Processamento de Linguagem Natural(PLN).

III.I LIMPEZA

A princípio, iniciou-se com a limpeza da base. A categoria “descrição” possuía muitos nulos (28,53%), entretanto, remover o atributo geraria uma perda gigantesca de informação. Assim, escolheu-se remover os itens sem descrição para preservar o máximo de informação e homogeneidade da estrutura dos dados. Na sequência, as colunas com o nome do produto e descrição foram fundidas.

III.II PLN

A seguir, iniciou-se o PLN. Em primeiro lugar foi feita a *tokenização,* quando os textos são separados em *tokens*, no caso, palavras, e remoção de todas as *stopwords* como artigos, pronomes, conjunções e preposições. Após a limpeza obtivemos a seguinte nuvem de palavras:

Imagem 1: Word cloud sem limpeza

Nela pode-se ver que algumas palavras se referem ao contexto de vendas online e não sobre os produtos como: produto, frete, mercado, compra, pagamento, entrega, envio, anúncio, cliente, vendedor, prazo. Tais palavras não auxiliam no reconhecimento dos produtos, assim foram consideradas stopwords e removidas, gerando uma segunda nuvem de palavras.

Imagem 2: WordCloud com limpeza de palavras

Agora é possível ver que mais palavras são mais significativas, assim temos as frases “limpas” e “mais limpas”.

Também foi feita a *lemmatização* das tokens, ou a diminuição da variedade das palavras e transformação delas em seu *lemma*, (versão mais simples da palavra, sem variação de gênero, quantidade e conjugação de verbos).

III.III EXTRAÇÃO DE FEATURES

Com o texto pré-processado, foi preciso transformá-lo em informação para os modelos. Para isso, foram escolhidos dois métodos de processamento:

- **TF-IDF (Term Frequency–Inverse Document Frequency):** que não leva em consideração o significado das palavras, ele consiste em uma contagem de palavras no texto normalizada pela quantidade total de palavras,
- **Word-Embeddings:** que leva em consideração o significado das palavras, ele consiste em posicionar as palavras em um espaço vetorial em que as dimensões são escalas de significado que as palavras podem ser inseridas, por exemplo gênero, sentimento, e até coisas mais abstratas como realeza. Utilizamos um modelo pré treinado da biblioteca SpaCy chamado *pt_core_news_lg*. Além disso, como cada palavra possui sua localização no espaço, e e a componente de texto é uma sentença, foi feita a média das posições das palavras para definir a posição daquela tupla.

IV. MÉTODO HIERÁRQUICO AGLOMERATIVO

IV.I DEFINIÇÃO

Os métodos hierárquicos são técnicas simples em que os dados são particionadas de forma hierárquica. Isso significa que é possível obter mais informações com essas partições, como por exemplo o grau de semelhança entre dois objetos, além de saber se determinado objeto é um *outlier* ou não.

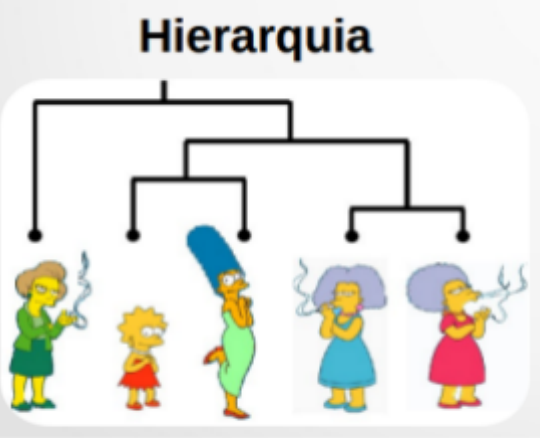


Imagem 3: Dendograma

A figura 1 apresenta um dendograma, o diagrama gerado pela hierarquização e que apresenta o relacionamento entre os objetos, assim como o seu nível de semelhança (quanto menor o número de nós entre dois objetos, mais semelhantes eles são). A partir dessa imagem, é possível entender que esse método de agrupamento não necessita de uma definição prévia de quantidade de *clusters*, visto que dependendo do número de grupos desejados, basta realizar um corte em um nível que gere a quantidade desejada.

No caso do algoritmo aglomerativo, a ideia é que cada objeto da base de dados comece como um grupo próprio e elementar, ou seja, de um único elemento, e conforme forem sendo feitas as análises, esses grupos se fundem um ao outro, em diferentes níveis, até formar um único grupo. Isso pode ser visto na imagem 1 quando analisada de baixo para cima, em que inicialmente há 5 grupos unitários (as 5 personagens femininas) que foram agrupados logicamente até formar um único grupo com todos os elementos.

IV.II MOTIVOS DA ESCOLHA

Esse algoritmo foi escolhido para que, com os diferentes níveis de hierarquia, pudesse analisar como as aglomerações podem se diferenciar dos rótulos

V. APLICAÇÃO DO MÉTODO ESCOLHIDO NAS BASES

Relembrando, para aplicar o método hierárquico foram criadas 4 bases através do processamento de linguagem:

- ***l_tfidf_features****:* features geradas por TF-IDF e extraídas das tokens “limpas”
- ***ml_tfidf_features****:* features geradas por TF-IDF e extraídas das tokens “mais limpas”
- ***l_we_features****:* features geradas por Word-Embedding e extraídas das tokens “limpas”
- ***ml_we_features****:* features geradas por Word-Embedding e extraídas das tokens “ mais limpas”

Em todas elas utilizamos a validação das silhuetas, o índice de Davies-Bouldin, a métrica Adjusted Rand Index (ARI) e a métrica da homogeneidade para encontrar o melhor número de *clusters* para a aplicação.

No caso, essas duas últimas foram escolhidas porque elas analisam os *clusters* juntamente com as labels definidas na base. O ARI avalia a qualidade dos agrupamentos de forma a permitir que o usuário saiba o quão bem os agrupamentos preditos se alinham com os agrupamentos reais, sendo que quanto mais próximo de 1 melhor é essa distribuição. A métrica de homogeneidade reflete a qualidade do agrupamento, avaliando a homogeneidade dele em relação aos rótulos verdadeiros, sendo que quanto mais próximo de 1 melhor é a distribuição.

V.I BASE: *L_WE_FEATURES* COM LINKAGE WARD

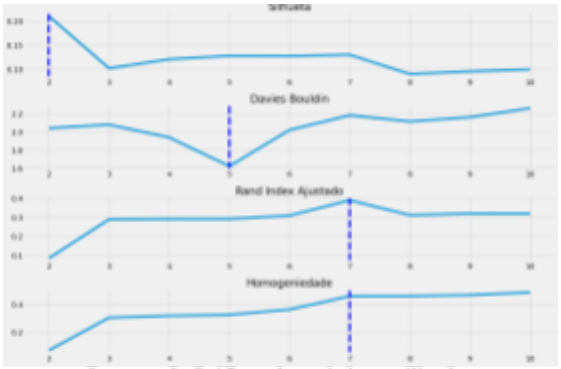


Imagem 5: Gráficos das métricas utilizadas (índice x número de clusters)

A imagem acima mostra os valores obtidos pela métrica, de forma que a linha pontilhada indica o número de *clusters* adequado, de acordo com cada métrica. Com isso, verificamos a distribuição dos dados com 2, 5 e 7 *clusters* (as duas últimas métricas acusaram a mesma quantidade de grupos como o ideal).

l_we_silhueta	categoria	
1	brinquedo	394
	maquiagem	390
	game	174
	livro	79
2	livro	759
	game	448
	maquiagem	398
	brinquedo	274

Name: categoria, dtype: int64

Imagem 6: 2 clusters

l_we_davies	categoria	
1	maquiagem	385
	brinquedo	305
	game	163
	livro	75
2	brinquedo	86
	maquiagem	3
	livro	2
3	game	11
	brinquedo	3
	livro	2
	maquiagem	2
4	livro	708
	game	19
	brinquedo	10
5	game	429
	maquiagem	398
	brinquedo	264
	livro	51

Name: categoria, dtype: int64

Imagem 7: 5 clusters

l_we_best_fit	categoria	
1	maquiagem	269
	brinquedo	109
	livro	8
	game	4
2	brinquedo	196
	game	159
	maquiagem	116
	livro	67
3	brinquedo	86
	maquiagem	3
	livro	2
4	game	11
	brinquedo	3
	livro	2
	maquiagem	2
5	livro	708
	game	19
	brinquedo	10
6	maquiagem	326
	brinquedo	48
	livro	3
	game	2
7	game	427
	brinquedo	216
	maquiagem	72
	livro	48

Name: categoria, dtype: int64

Imagem 8: 7 clusters

Através dessas imagens, é possível entender como os objetos estão distribuídos dentro dos *clusters*. Com 2 clusters (imagem 6) o agrupamento conseguiu manter eficientemente os livros em um único *cluster*. Com 5 (imagem 7) e 7 (imagem 8) grupos já é possível perceber uma melhora na distribuição, visto que apresenta um grupo dominado pelos livros, um pelos brinquedos.

V.II *ML_WE_FEATURES* COM LINKAGE WARD

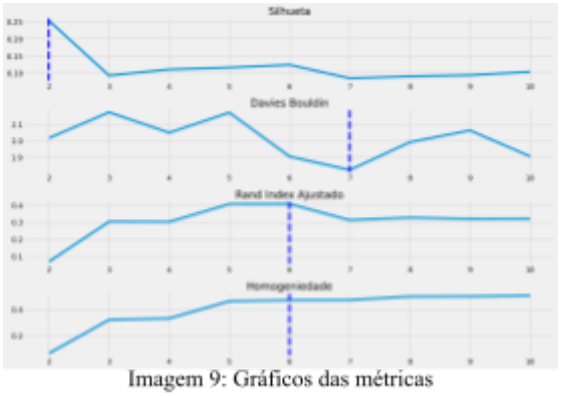


Imagem 9: Gráficos das métricas utilizadas (índice x número de clusters)

A imagem acima mostra os valores obtidos pelas métricas, quando utilizada a base *ml_we_features*. Nela, verifica-se que a distribuição dos dados com 2, 7 e 6 *clusters* (as duas últimas métricas acusaram a mesma quantidade de grupos como o ideal) devem apresentar o melhor desempenho.

ml_we_silhueta	categoria	
1	brinquedo	349
	maquiagem	292
	game	112
	livro	46
2	livro	792
	game	510
	maquiagem	496
	brinquedo	319

Name: categoria, dtype: int64

Imagem 10: 2 clusters

ml_we_davies	categoria	
1	maquiagem	286
	brinquedo	259
	game	100
	livro	41
2	brinquedo	86
	maquiagem	3
	livro	2
3	game	12
	brinquedo	4
	livro	3
	maquiagem	3
4	livro	476
	game	11
	brinquedo	10
5	livro	245
	brinquedo	1
	game	1
6	maquiagem	391
	brinquedo	5
	livro	3
	game	1
7	game	497
	brinquedo	303
	maquiagem	105
	livro	68

Name: categoria, dtype: int64

Imagem 11: 7 clusters

ml_we_best_fit	categoria	
1	maquiagem	286
	brinquedo	259
	game	100
	livro	41
2	brinquedo	86
	maquiagem	3
	livro	2
3	game	12
	brinquedo	4
	livro	3
	maquiagem	3
4	livro	721
	game	12
	brinquedo	11
5	maquiagem	391
	brinquedo	5
	livro	3
	game	1
6	game	497
	brinquedo	303
	maquiagem	105
	livro	68

Name: categoria, dtype: int64

Imagem 12: 6 clusters

A partir da imagem 10, é possível verificar que o desempenho dessa base foi muito semelhante ao desempenho da base anterior. Manteve grande parte dos livros em um único cluster, além de conseguir separar um pouco melhor os games. Enquanto isso, novamente a divisão com 7 clusters (imagem 11) e 6 clusters (imagem 12) apresentam um desempenho mais satisfatório, conseguindo manter uma maior homogeneidade dentro dos clusters, já que apresentam clusters dominados por brinquedos e livros, por exemplo.

V.III *L_TFIDF_FEATURES* COM WARD

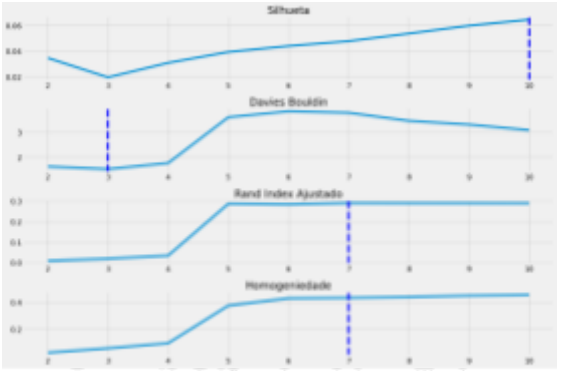


Imagem 13: Gráficos das métricas utilizadas (índice x número de clusters)

Diferentemente das bases gerada pela técnica de Word-Embedding, essa base já apresenta que a melhor opção pela silhueta são 10 clusters ao invés de 2, contudo o valor das duas últimas continua sendo o mesmo. Assim, foi estudada a distribuição dos elementos quando utilizados 10, 3

e 7 clusters.

l_tfidf_silhueta	categoria	
1	brinquedo	89
	maquiagem	10
	livro	2
	game	1
2	game	92
	brinquedo	6
3	game	88
4	livro	713
	brinquedo	8
	game	6
	maquiagem	1
5	maquiagem	220
6	game	47
	maquiagem	24
	brinquedo	20
	livro	3
7	brinquedo	24
8	maquiagem	36
9	brinquedo	20
10	brinquedo	501
	maquiagem	497
	game	388
	livro	120

Name: categoria, dtype: int64

Imagem 14: 10 clusters

l_tfidf_davies	categoria	
1	brinquedo	89
	maquiagem	10
	livro	2
	game	1
2	game	92
	brinquedo	6
3	livro	836
	maquiagem	778
	brinquedo	573
	game	529

Imagem 15: 3 clusters

l_tfidf_best_fit	categoria	
1	brinquedo	89
	maquiagem	10
	livro	2
	game	1
2	game	92
	brinquedo	6
3	game	88
4	livro	713
	brinquedo	8
	game	6
	maquiagem	1
5	maquiagem	220
6	game	47
	maquiagem	24
	brinquedo	20
	livro	3
7	brinquedo	545
	maquiagem	533
	game	388
	livro	120

Name: categoria, dtype: int64

Imagem 16: 7 clusters

Com essa base, é possível perceber que alguns grupos conseguiram ser ainda mais homogêneos, apresentando objetos de uma única classe, ou então duas. Contudo, fora os livros, esses acertos foram muito poucos, fato que pode ser observado ao analisar o último da imagem 14, da imagem 15 e da imagem 16, os quais apresentam um grande número de elementos e de classes diferentes também

V.IV ML_TFIDF_FEATURES COM WARD

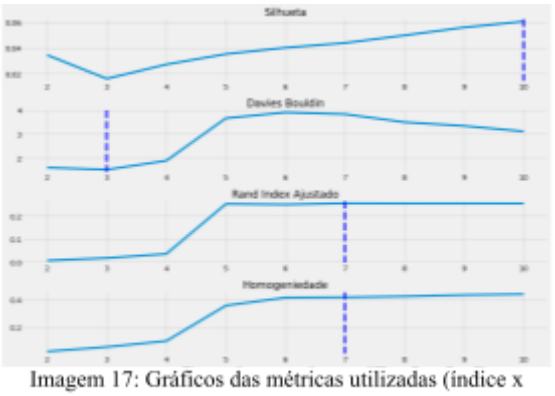


Imagem 17: Gráficos das métricas utilizadas (índice x número de clusters)

Ao visualizar a imagem 17, é possível perceber que a métricas acusaram os mesmos valores acusados na base *l_tfidf_features*

ml_tfidf_silhueta	categoria	
1	brinquedo	89
	maquiagem	10
	livro	2
	game	1
2	game	92
	brinquedo	6
3	game	101
4	livro	672
	game	5
	brinquedo	3
	maquiagem	1
5	maquiagem	227
6	game	40
	maquiagem	24
	brinquedo	22
	livro	3
7	brinquedo	24
8	maquiagem	36
9	brinquedo	20
10	brinquedo	504
	maquiagem	490
	game	383
	livro	161

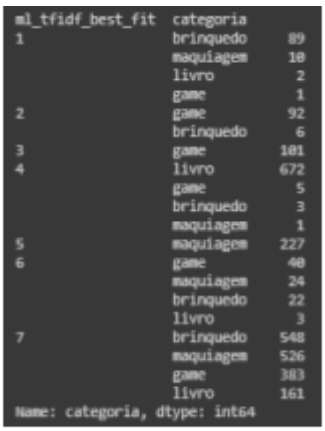
Name: categoria, dtype: int64

Imagem 18: 10 clusters

ml_tfidf_davies	categoria	
1	brinquedo	89
	maquiagem	10
	livro	2
	game	1
2	game	92
	brinquedo	6
3	livro	836
	maquiagem	778
	brinquedo	573
	game	529

Name: categoria, dtype: int64

Imagem 19: 3 clusters



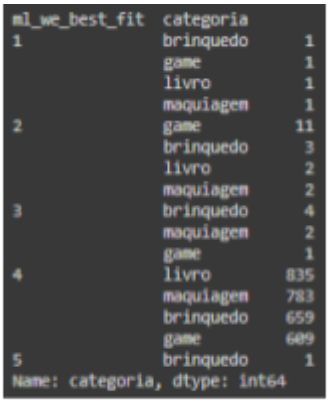
ml_tfidf_best_fit	categoria	
1	brinquedo	89
	maquiagem	10
	livro	2
	game	1
2	game	92
	brinquedo	6
3	game	101
4	livro	672
	game	5
	brinquedo	3
	maquiagem	1
5	maquiagem	227
6	game	40
	maquiagem	24
	brinquedo	22
	livro	3
7	brinquedo	548
	maquiagem	526
	game	383
	livro	161

Name: categoria, dtype: int64

Imagem 20: 7 clusters

Ao estudar essas imagens é possível perceber uma leve melhora em relação à base *l_tfidf_features*, apesar da imagem 19 apresentar a mesma distribuição da imagem 15. Nas imagens 18 e 19 é possível perceber que os clusters que eram predominados por uma classe específica aumentaram sua homogeneidade, como aumentando a quantidade de objetos da classe dominante. Contudo, ainda é notável o último cluster, o qual apresenta muitos elementos e de classes diferentes, ou seja, uma heterogeneidade elevada.

V.V OUTROS MÉTODOS (SINGLE, COMPLETE E AVERAGE)



ml_we_best_fit	categoria	
1	brinquedo	1
	game	1
	livro	1
	maquiagem	1
2	game	11
	brinquedo	3
	livro	2
	maquiagem	2
3	brinquedo	4
	maquiagem	2
	game	1
4	livro	835
	maquiagem	783
	brinquedo	659
	game	609
5	brinquedo	1

Name: categoria, dtype: int64

Imagem 21: Exemplo com linkage "Single"

Além da aplicação do linkage "Ward", também foram testados os métodos de linkage "Single", "Complete" e "Average", contudo todos apresentaram um resultado extremamente incoerente. Para exemplificar, a imagem 21 apresenta um cluster com quase todos os elementos da base, um cluster com um elemento elemento de cada classe e possui até mesmo um cluster com um único elemento. Esse padrão se repetiu com os outros dois métodos de *linkage*, fazendo com que o grupo optou por não inseri-los no estudo, visto que não teriam nada a agregar.

A hipótese gerada pelos estudantes, foi de que o PLN gerou uma base que favorece o uso do método "*Ward*", e que por isso ele apresenta um resultado tão significativo em relação aos demais métodos.

VI. CONCLUSÃO

Com base no estudo realizado, foi possível concluir que o melhor método de Linkage para a base gerada pelo PLN (tanto pelo método *Word Embendding*, quanto pelo método TF-IDF) foi o "*Ward*", enquanto os demais apresentaram um desempenho muito desfavorável.

O grupo também percebeu que nas bases geradas pelo *TF-IDF* havia objetos que foram muito bem separados, enquanto alguns clusters continham uma enorme quantidade de elementos e com classes diferentes. Isso fez com que o grupo tentasse entender o porquê desse comportamento, gerando a hipótese de que muito provavelmente ao utilizar essa base, os algoritmo conseguiram separar bem os objetos que continham grande presença do nome de suas classes na descrição (conter a palavra "brinquedo" na descrição de um objeto da classe brinquedo, por exemplo), já que o método é uma contagem de palavras normalizada enquanto os objetos que não continham essa peculiaridade não obtiveram sucesso em sua separação, gerando *clusters* muito grandes e completamente misturados.

Pode-se determinar que o algoritmo de hierarquização conseguiu separar com melhor eficiência as bases geradas pela técnica de *Word Embendding*. Isso era esperado, já que ela coloca as palavras em um espaço vetorial, o que favorece os algoritmos de aglomeração, uma vez que sua base de funcionamento é o cálculo de distância entre os pontos, além disso a vetorização é feita com um modelo de aprendizado não supervisionado que correlaciona as palavras. Embora tenha atuado melhor ainda possuía clusters bem heterogêneos em relação às classes e o grupo pode supor algumas hipóteses: o modelo pré-treinado foi feito com notícias podendo ter vetorizado mal algumas palavras; podemos ter dimensões que não são relevantes para o processamento e um estudo e atenção a limpeza dessas dimensões com certeza contribuiria para isso; o método para a vetorização da sentença pode não ter sido o mais adequado.

Além disso, pode-se notar que as *features* extraídas das *tokens* "mais limpas" geraram *clusters* mais homogêneos, comprovando, mesmo que de forma muito sucinta, nossa hipótese de que

remover as palavras que envolvem o contexto de vendas online ajudariam nisso. Isso pode ser explicado pois a presença dessas palavras causam sobreposição das tuplas dificultando os algoritmos de aglomeração.

Por fim, com mais tempo para o desenvolvimento do trabalho uma análise mais profunda sobre os *clusters* poderia ser feita a fim de entender se foram palavras que poderiam ter causado a sobreposição de classes em algum *deles,* se foi uma falta de detalhamento na descrição, ou se até mesmo uma imaturidade na aplicação dos métodos, tanto de PLN quanto de Aglomeração, por nossa parte.

VII. Referência

[1] Link do DataSet
[2] Modelo pré-treinado de português
[3] Link do Jupyter Notebook