Agrupamento de manchetes - Cluster

Ciro Javier Diaz Penedo*
Lucas Leonardo Silveira Costa*

Abstract

Neste trabalho tratamos do problema de agrupamento em headlines (manchetes) do jornal ABC (Australian Broadcasting Corporation) utilizando técnicas de aprendizado de máquina sem supervisão. Apresentamos e discutimos os resultados sobre os clusters encontrados.

1. Introdução

Atualmente o processo de detecção de padrões é frequentemente necessário em empresas, órgãos governamentais, bancos, etc. Tais processos são úteis para realizar políticas de decisões, por exemplo, um banco pode detectar padrões em seus clientes e definir taxas de serviços, políticos podem avaliar as características de seus candidatos e dos não-candidatos e assim investir em campanhas para conquistar mais eleitores, entre outras.

Esse processo para detectar padrões pode ser realizado por meio de aprendizado de máquina sem supervisão (*unsupervised learning*) utilizando agrupamento (*clustering*), isto é, não se conhece as características dos grupos existentes e nem a quantidade de elementos em cada grupo.

Uma aplicação útil usando agrupamento é a detecção de padrões em textos. O agrupamento em textos pode ser útil na pesquisa forense atuando na detecção de mensagens de criminosos [2] e na detecção de spam na caixa de *e-mail*.

Nesse trabalho utilizamos o conceito de *N-grams* [1] para a extração de atributos em textos e realizamos a detecção de padrões em *headlines* (manchetes) do jornal ABC (*Australian Broadcasting Corporation* - http://www.abc.net.au/), depois utilizamos o método *K-means* para agrupar as manchetes. Para aplicar este método é necessário transformar cada texto em um vetor numérico de atributos (*features*) que representam características relevantes deles para relacionar um texto (vetor) com outro.

A metodologia escolhida para encontrar a solução do problema é descrita a seguir:

*Do Instituto de Matemática, Estatística e Computação Científica da Universidade de Campinas (Unicamp). **Contato**: ra153868@ime.unicamp.br e ra153866@ime.unicamp.br

i. Simplificar a linguagem natural das *headlines*;
ii. Construir um dicionário de *N-grams*;
iii. Construir os vetores de atributos para cada *headline*;
iv. Aplicar o algoritmo de agrupamento (*k-means* no caso);
v. Discutir os resultados dos experimentos.

2. Simplificar a linguagem natural

A linguagem natural tem muitas regras para melhorar o entendimento entre as pessoas. Como o nosso objetivo é relacionar textos, precisamos simplifica-los ficando com apenas o essencial. Por isso todas as palavras vão ser consideradas em minúsculas, tiramos todos os símbolos que não estejam no alfabeto e mantemos apenas o radical [3] dos substantivos, adjetivos, verbos e advérbios. Assim o texto: *"João gosta de assistir filmes. Maria gosta de filmes também."* torna-se *"joão gost assist film maria gost film também"*

3. Construir um dicionário de *N-grams*

Um *N-gram* [1] é um conjunto ordenado de N palavras e um dicionário destes seria uma sequência de tuplas formadas por uma chave e uma descrição, em nosso a chave (*key*) é um *N-gram* e a descrição é a quantidade de vezes (*frequência*) que este aparece no conjunto de dados. Para descrever a construção do dicionário suponhamos que temos estas duas frases no conjunto de dados:

(1) João gosta de assistir filmes. Maria gosta de filmes também."

(2) João também gosta de assistir a jogos de futebol !"

Após simplificar os dados via Seção 2 o nosso dicionário de *1-grams* seria:

"joão":2, "gost":3, " assist":2, "film":2, "maria":1, "também":1, "jog":1, "futebol": 1 .

Para *2-grams* teríamos:

("joão","gost"):1, ("gost","assist"):2, ("assist",film"):1
...

4. Construir os vetores de atributos.

Nos experimentos computacionais usamos os *N-grams* do dicionário que possuem uma frequência maior ou igual a 10, mas para o exemplo anterior usaremos todas.

Posição	1ª	2ª	3ª	4ª
Palavra	joão	gost	assist	film
Posição	5ª	6ª	7ª	8ª
Palavra	maria	também	jog	futebol

Tabela 1. Tabelas com as palavras do dicionário

Esses oito *1-grams* da Tabela 1 representam os nossos atributos utilizados para a conversão dos textos em vetores, sendo assim, percorremos cada manchete e criamos um vetor binário de 8 entradas que indica se o texto possui o *1-gram* do dicionário ou não. Para o exemplo apresentamos o resultado na Tabela 2.

Palavra	1ª	2ª	3ª	4ª	5ª	6ª	7ª	8ª
Frase (1)	1	1	1	1	1	1	0	0
Frase (2)	1	1	1	0	0	1	1	1

Tabela 2. Amostras para realizar o cluster

Poderíamos também usar *2-gram* e nesse caso utilizaríamos o dicionário de *2-grams*. Também podemos usar *1-gram* + *2-gram* juntando ambos dicionários. Claramente que *N-grams* de 3, 4 ou mais palavras podem ser usados (e seria desejável), neste trabalho por problemas de potência computacional decidimos ficar com *1-grams* e *2-grams*.

5. Agrupamento (*K-means*)

O algoritmo *K-means* pode ser descrito como um problema de otimização (1).

$$\text{min: } J(c^1,\ldots,c^m,\mu_1,\ldots,\mu_k) = \sum_{i=1}^{m} dist(x^{(i)},\mu_{c^i}) \quad (1)$$

em que a função J é chamada de inércia [4], $x^{(i)}$ são os dados, $c^i = 1,\ldots,k$ são os índices que indicam qual o grupo que $x^{(i)}$ pertence, μ_k é o centróide do grupo k e a *dist* é alguma medida ,por exemplo: $||\cdot||_1$, $||\cdot||_2$, *cosine distance*.

O algoritmo se chama *K-means* pois a palavra *means* significa médias e indica a maneira como μ_k foi calculado, isto é, para obter os centróides usamos a equação 2

$$\mu_{kj} = \frac{1}{t}\sum_{i=1}^{t} x_j^{(c^i)}, \quad j = 1,\ldots,M, \qquad (2)$$

em que M é a dimensão de $x^{(i)}$, e desse modo, cada entrada j de μ_{kj} é a média das entradas $j's$ dos vetores $x^{(i)}$ pertencente o grupo k.

O algoritmo *K-means* é descrito a seguir:

Defina os k centróides iniciais e repita $i = 1,\ldots,$it. max.:

1. Atribua os índices c^i em $x^{(i)}$ em que c^i = arg(min $dist(x^{(i)},\mu_{c^i})$);
2. Atualize os centróides μ_k calculando a média dos $x^{(i)}$ pertencentes ao cluster k.

Uma variação do algoritmo *K-means* é o algoritmo *k-medoids* e nesse caso, μ_k é o elemento $x^{(h)}$ do grupo k o qual possui o menor valor da somatória entre as distancias de $x^{(h)}$ com os elementos do grupo k. Outra variação é o *K-median* que utiliza μ_k como sendo a mediana do grupo k e o *Mini Batch K-means* é uma variação que utiliza parcelas (*Batch*) dos danos para realizar os clusters, sendo bem útil quando se tem muitos dados e pouca potência computacional.

Para a escolha da quantidade de *cluster* (k) utiliza-se a *elbow rule* (regra do cotovelo). Podemos ver na equação 1 que se consideramos um único *cluster* o valor do minimo que a função alcançaria seria o maior valor possível, e em contrapartida se considerarmos cada ponto sendo um cluster o valor atingido será zero, pois o centróide do *cluster* k seria o próprio ponto.

Desse modo, a *elbow rule* consiste relacionar a quantidade de *clusters* k com o valor da inércia $J(c^1,\ldots,c^m,\mu_1,\ldots,\mu_k)$ em 1, e escolher o k tal que a variação de J de k para $k+1$ seja pequena.

Observação: Outras regras usadas para a escolha da quantidade de *cluster* são o *Silhoutte* e *Calinski Harabasz* [4] mas, não vamos usar elas neste trabalho.

6. Experimentos e Discussões

Os experimentos computacionais foram realizados em *python* e o conjunto de dados era composto por 1 milhão e uma manchetes publicadas entre 19/02/2003 até 31/12/2017 pela ABC. Após simplificar a linguagem natural (2) e construir os dicionários de *1-grams* e *2-grams* (3), construímos nosso vetor de atributos (4) usando os *N-grams* com frequências maiores ou iguais a 10. Isso seria em torno de 25000 features para cada dicionário. Aplicamos o algoritmo *k-means* variando o parâmetro k entre 2 e 21 para tentar aplicar a *elbow rule* (5). Os experimentos foram separados da seguinte maneira: No primeiro usamos as features obtidas via *1-grams* e no segundo juntamos os features obtidas via *1-grams* + *2-grams*. Após selecionar o melhor k (Experimentos 1 e 2) fizemos um terceiro experimento onde dividimos o conjunto de *headlines* por anos (15 subconjuntos no total), e aplicamos k-means em cada um deles.

6.1. Features baseadas em 1-grams

Na Figura 1 podemos analisar o gráfico da função inércia (equação 1) para o caso 1-*gram*. Nela observamos que de fato o valor da inércia vai diminuindo, porém, não é possível detectar se até $k = 21$ o valor irá se estabilizar.

Na Tabela 3 apresentamos a quantidade de elementos em cada *cluster* para $k = 20$ e 21 do caso 1-*gram*. Podemos observar que o *cluster* 1 possui perto de 74% dos dados, o

resto dos dados se distribuem nos outros *clusters*. Esse fato foi observado também em valores de k menores que 20.

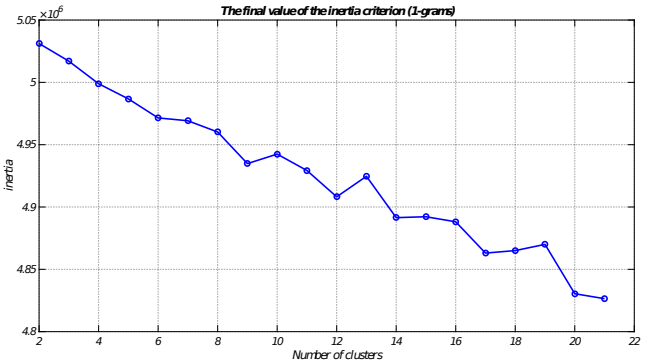

Figura 1. Valores da função inércia 1 em relação a quantidade de cluster considerada ($k = 2, \ldots, 21$) considerando 1-*gram*.

Cluster	1	2	3	4	5	6	7	8	9	10	11
k = 20	763984	28373	24319	23418	20632	14089	14027	13973	12508	10865	10624
k = 21	745966	37114	29213	27747	20618	18298	14718	11348	11282	10312	9542
Cluster	12	13	14	15	16	17	18	19	20	21	-
k = 20	10458	9149	9139	9092	8772	5702	5598	4068	1211	–	–
k = 21	8554	7836	7588	7109	6666	6193	6150	6097	5957	1693	–

Tabela 3. Quantidade de elementos em cada cluster, considerando $k = 20$ e 21 e com atributos obtidos com 1-*gram*.

6.2. Features com 1-grams + 2-grams

Na Figura 2 apresentamos o gráfico da função inércia considerando os atributos extraídos com 1-*gram* + 2-*gram*. Como na Figura 1 observamos que de fato o valor da inércia vai diminuindo, porém, não é possível detectar se até $k = 21$ o valor irá se estabilizar.

Na Tabela 4 apresentamos a quantidade de elementos em cada clusters $k = 18$ e 19 para o caso este caso e vemos que perto de 78% dos dados estão localizados no primeiro *cluster* como no Experimento descrito em 6.1.

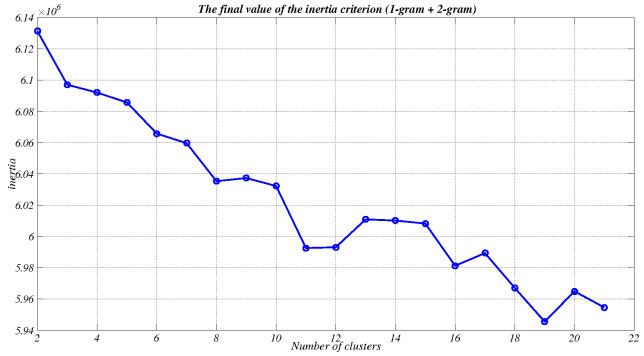

Figura 2. Valores da função inércia 1 em relação a quantidade de cluster considerada ($k = 2, \ldots, 21$) considerando 1-*gram* + 2-*gram*.

Cluster	1	2	3	4	5	6	7	8	9	10
k = 18	821361	32030	29574	19787	14207	11697	11501	10351	9611	8754
k = 19	783575	29295	25300	21402	18469	18032	15779	14158	14002	11201
Cluster	11	12	13	14	15	16	17	18	19	–
k = 18	6387	6247	6212	4533	3425	3342	602	380	–	–
k = 19	8790	8235	7265	7174	5924	4547	3444	2785	624	–

Tabela 4. Quantidade de elementos em cada cluster, considerando $k = 20$ e 21 e com atributos obtidos com 1-*gram* + 2-*grams*.

6.3. Nuvem de palavras com 1-gram + 2-grams

A Tabela 5 apresenta os principais temas dos cluster criados pelo método *K-means* considerando $k = 19$ para o caso 1-*gram* + 2-*gram*. As Figuras 3 e 4 mostram as nuvens de palavras usando as *headlines* dos clusters referentes a "*police*" e "*dies*".

Cluster	1	2	3	4	5
Tema	*	us	say	*	prices
Cluster	6	7	8	9	10
Tema	hospital	mayor	Hewitt	New York	New Zeland
Cluster	11	12	13	14	15
Tema	hit-run	missing	face	court	police
Cluster	16	17	18	19	-
Tema	*	trial	council	dies	-

Tabela 5. Principais palavras dos clusters considerando 19 clusters e com 1-*gram* e 2-*gram*. * palavras variadas.

Figura 3. Nuvem de palavras usando as *headlines* do cluster 15, considerando $k = 19$ clusters e com 1-*gram* + 2-*gram*.

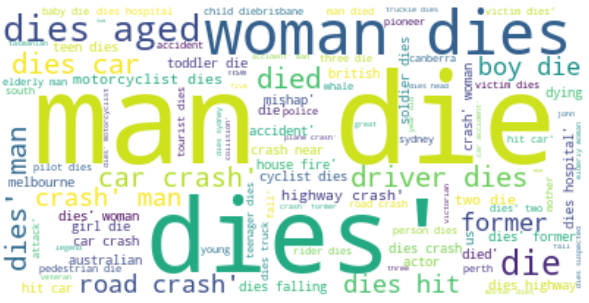

Figura 4. Nuvem de palavras usando as *headlines* do cluster 19, considerando $k = 19$ clusters e com 1-*gram* + 2-*gram*.

6.4. Clusters para cada ano

Usando $k = 19$ dividimos o conjunto de *headlines* por anos (15 subconjuntos no total), e aplicamos *k-means* em

cada um deles. A Figura 5 apresenta os valores da inércia para cada ano (2013-2017), nela podemos ver que a variação para cada ano é bem pequena. Na Figura 6 podemos observar a quantidade de elementos em cada *cluster* em cada ano, para todos os anos a cardinalidade do *cluster* com maior quantidade elementos foi representado na legenda.

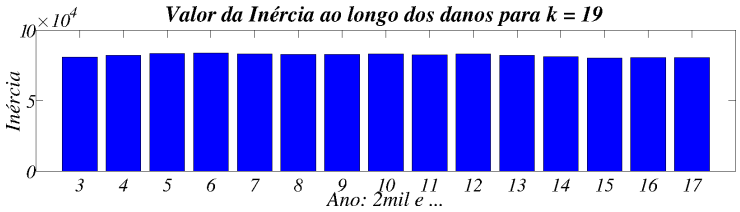

Figura 5. Valores da função inércia 1 em relação aos anos e considerando $k = 19$ clusters e 1-*gram* + 2-*gram*.

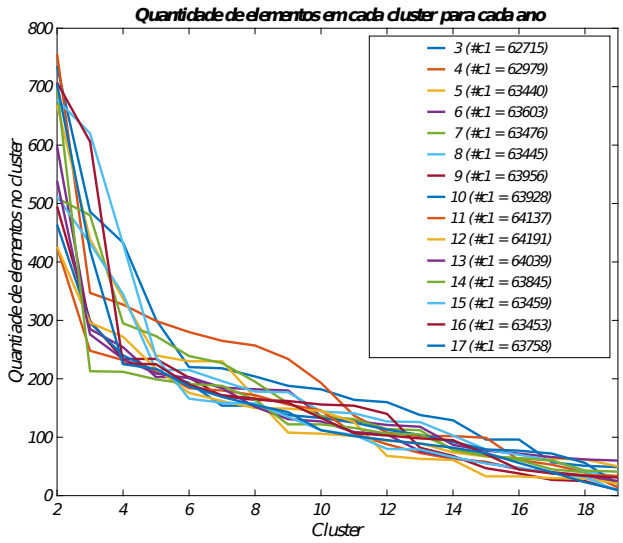

Figura 6. Cardinalidade de cada *cluster* para cada anos e considerando $k = 19$ clusters e 1-*gram* + 2-*gram*.

7. Conclusões e Trabalhos Futuros

Os resultados indicam que a utilização de *k-means* separa os dados devagar tendo sempre um cluster maior que vai diminuindo de tamanho na medida que aumentamos k. Achamos que isto acontece devido ao fato de que as *headlines* são cadeias de texto muito pequenas, entre 8 e 10 palavras, e temas que deveriam ser agrupados não conseguem se conectar, por exemplo, se o tema for esportes poderia ter *headlines* falando de tênis e futebol usando palavras diferentes.

Podemos observar consistência nos resultados. Por um lado a Figura 5 mostra que os valores da função inércia são similares para todos os anos e por outro, a Figura 6 indica que a quantidade de elementos de cada clusters ano a ano se é similar o qual seria esperado se o agrupamento estivesse relacionando os mesmos temas.

O uso de *3-grams* e *4-grams* pode melhorar o valor da inércia, estes atributos foram calculados porém os códigos demoravam para realizar o agrupamento e por isso ficamos apenas no caso $N = 1, 2$.

Além disso as notícias costumam ser muito localizadas no tempo, assim as mais conectadas seriam as referentes a um mesmo acontecimento (por exemplo "*brazil world cup*") que formariam um cluster pequeno de algumas dezenas ou centenas de *headlines*. Isto explicaria o alto valor da função inércia. Valores menores da inércia seriam alcançados só quando o valor de k é muito grande. Provavelmente um resumo da notícia (chamadas) permitiria que aparecessem nos dados mais palavras associadas ao tema que os relacionam e teríamos um melhor agrupamento.

Referências

[1] https://en.wikipedia.org/wiki/N-gram, Acessado em 09/04/2018. 1

[2] A. Rocha et al., "Authorship Attribution for Social Media Forensics,"in IEEE Transactions on Information Forensics and Security, vol. 12, no. 1, pp. 5-33, Jan. 2017. 1

[3] https://pt.wikipedia.org/wiki/Radical_(linguística), Acessado em 09/04/2018. 1

[4] http://scikit-learn.org/stable/modules/clustering.html, Acessado em 09/04/2018. 2