Clusterização de emails

André Almeida
RA: 164047
fda.andre@gmail.com

Igor Torrente
RA: 169820
igortorrente@hotmail.com

I. Resumo

Nesse trabalho, estudamos técnicas de aprendizado de máquina e reconhecimento de padrões para obter um modelo de *clusterização* automática de emails em assuntos a partir do texto do email.

II. Introdução

A. Classificador automático de emails

Classificar emails automaticamente é um problema explorado por empresas provedoras de serviço de email. Uma classificação importante para manter a qualidade do serviço é a classificação binária *spam* (mensagens indesejadas com propaganda ou com golpes cibernéticos) / não *spam* [1], de forma a deixar a caixa de entrada dos usuários livres de conteúdo indesejado. Uma outra funcionalidade que a classificação automatizada pode trazer é separar os assuntos dos usuários entre atualizações, redes sociais, fóruns, etc., como faz o serviço de email do Google, o Gmail [2].

B. Clustering

Algoritmos de *clusterização* dividem dados em grupos (clusters) baseados apenas nas informações providenciadas pelos dados. O objetivo é conseguir agrupar em um mesmo grupo dados similares/relacionados e dados que sejam diferentes estarem em grupos diferentes. Quanto mais parecidos os dados de um grupo forem, e quanto maior a diferença entre grupos distintos, melhor será a clusterização desses dados. Em alguns algoritmos, o número de clusters é definido pelo usuário (e.g. *K-means*) e em outros o algoritmo tenta encontrar o número ideal de clusters (e.g. *DBSCAN*) [3]. Nesse estudo, exploramos o primeiro tipo, especificamente o *K-means*. Esse tipo de técnica é considerada de aprendizagem não-supervisionada. Ao contrário das técnicas de aprendizado supervisionadas, os dados não possuem previamente nenhum tipo de classe ou rótulo.

1) K-means: Esse algoritmo de clusterização é o mais antigo e mais usado. A ideia dele é encontrar K centroides (onde K é um parâmetro definido pelo usuário), onde um centroide é a média de um grupo de pontos. As primeiras localizações dos centroides podem ser aleatórias. Cada ponto é então atribuído ao centroide mais próximo, e um grupo de pontos de um mesmo centroide formam um cluster. O centroide é atualizado baseado na média dos pontos do seu cluster. O algoritmo continua assim até que não ocorra mais atualizações nos pontos dos clusters [3]. Uma simplificação do algoritmo pode ser definida como:

Algorithm 1 K-Means

1: Selecione K pontos como os centroides iniciais;
2: **repeat**
3: Forme os K clusters atribuido cada ponto ao seu centroide mais próximo;
4: Reposicione os centroides de cada cluester;
5: **until** Centroides não mudem.

C. Base de dados

A base de dados consiste de em um conjunto de e-mails do anos 90 que pertencem a grupos de noticias, cada e-mail possui um newsgroup a que ele pertence, um assunto, datas, o corpo do e-mail e outros metadados relacionados. [4]

D. Tecnologias empregadas

Utilizamos a biblioteca *sklearn* [5] do *Python* para as chamadas de pre-processamento de dados e clusterização. O *Jupyter* foi escolhido como ambiente de desenvolvimento.

E. Infraestrutura computacional

Os recurso computacionais foram:

- Computador com um processador Intel Core i7-3537U (2.00GHz × 4) e com 7,7 GiB disponíveis de memória RAM com Arch Linux;
- Máquina virtual com 4 núcleos virtuais do processor IBM Power8, com 16GB de RAM e Ubuntu 17.04;
- Computador com core i5 4440, 8 GB de Ram e Windows 10.

III. Tratamento de dados

Para conseguir extrair as informações necessárias dos textos, não necessitamos do texto integral do email. O cabeçalho contendo os metadados do email provavelmente não diz muito sobre o conteúdo do corpo do texto. A base de dados proposta para o experimento (*data.csv*) contêm cerca 2.209 palavras, do total de 2 milhões de palavras únicas encontradas nos arquivos. Essa base de dados é do tipo *bag-of-words*, com a frequência de cada termo dividido pela quantidade de emails em que ela aparece.
Também foram exploradas outras maneiras de reduzir o conteúdo dos arquivos. Utilizamos redução de dimensionalidade com PCA no `data.csv` e criamos novos conjuntos de dados usando *stemming* e *stopwords* nos emails brutos.

A. *Bag-of-words*

O *bag-of-words* é um modelo de representação textual comumente usada para classificação de texto. Ele reserva uma posição do vetor a cada palavra utilizada no texto ou conjunto de textos, existem três formas comuns de se utilizar a *bag-of-words* [10]

1) Uma que se utiliza a posição reservada para contar a quantidade de vezes que a palavra foi usada no texto ou conjunto de textos;
2) Cada posição do vetor guarda a frequência que o termo aparece divida pelo total de palavras do texto, esse modelo é chamado de bag of words TF (*Term frequecy*);
3) Por fim cada posição guarda a relevância estatística do termo em relação a sua frequência no conjunto de textos. A relevância é calculada como o TF dividido pela quantidade de textos que aquela palavra aparece. Esse mode é chamado de bag of words TF-IDF (*requency-inverse document frequency*).

B. *Stemming*

A técnica de *stemming* reduz a palavra para sua raíz morfológica, similar à recuperar o radical de uma palavra. Um exemplo seria aplicar *stemming* em ”correr”, ”correndo” e ”corria”, onde as três palavras se tornariam ”corr”. Isso é útil tanto para maximizar o significado das palavras (conjugações diferentes do mesmo verbo contem um significado muito próximo) quanto reduzir o número de dados. Os *stemmers* utilizados foram: *Snowball* [6], *Wordnet* [7], *Porter* [8] e *Lancaster* [9], todos importados da biblioteca de processamento de linguagem natural do Python, NLTK [14]. Os textos que passaram por stemming também tiveram todas suas palavras convertidas para a forma minúscula.

C. *Stopwords*

Stopwords são palavras muito comuns e sem muita relevância para o significado do texto, como ”nós”, ”eu”, ”aquilo”, ”entre”, ”porque”, etc. Removendo elas, diminuímos uma boa parte do volume de dados, já que as palavras mais comuns costumam ser *stopwords*.

D. *Exemplos de tratamentos*

Aqui comparamos as 5 primeiras palavras mais populares entre alguns desses pré processamentos:

Com *stopwords*: the, to, of, a, and, is.
Sem *stopwords*: would, one, writes, dont, like.
Após *Lancaster*: us, would, on, writes, artic.

IV. Soluções propostas

Nessa seção e nas próximas, as métricas para validar a eficiência do modelo são:

- ***Score***: O *score* do `sklearn` para o K-Means é calculado por:

$$(-1) \sum_{p \epsilon Cluster} (centroid - p)^2$$

  Ele é usado para medir a distancia dos do centroide ao seu cluster identificando o cotovelo (*elbow*) no gráfico de número de clusters pelo score [11].
- ***Calinsk***: O índice de Calinsk é dada pela razão entre as dispersões médias dos cluster pela dispersão do próprio cluster [12].
- ***Silhouette***: Para cada objeto *i* obtêm-se o valor s(*i*)

$$s(i) = \frac{bi-ai}{max(ai,bi)}$$

  onde: a_i é a dissimilaridade média do objeto *i* em relação a todos os outros objetos do seu cluster. b_i é a dissimilaridade média do objeto *i* em relação a todos os outros objetos do cluster vizinho mais próximo. Silhouette é um coeficiente que varia de -1 a 1 que indica a consistencia do cluster avaliado. Com 1 indicando que objeto sendo bem ajustado com o cluster a que ele pertence e -1 muito pouco ajustado [13].

Utilizamos a função `sklearn.cluster.KMeans`, variando o número de clusters (*k*). Foram feitos testes de 2 à 200 clusters.

A partir disso, foram testados os seguintes modelos com o data.csv:

A. *K-Means sem PCA*

Pelo gráfico *k* por *Silhouette*, o melhor número de cluster é 80, já que a partir desse ponto o gráfico começa a se manter mais constante. O *score* foi de -14452.390996, o *Calinks* 67.2349438967 e o *Silhouette* 0.0546147507142.

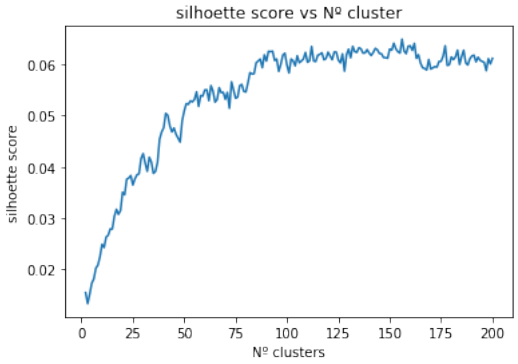

Figura 1: *Gráfico de k por Silhouette*

Clusters	Score	Calinski	Silhoette
2	-18038.0333456	312.302300127	0.0154703270706
20	-16451.1140207	119.058555969	0.0350597130577
40	-15527.5867898	91.7148367353	0.047632142909
60	-14899.7376406	77.3031750537	0.0550020513535
80	-14452.390996	67.2349438967	0.0546147507142
100	-14138.9536653	59.2252786152	0.0598284846652
120	-13838.2058328	53.9083198115	0.0623821272021
140	-13617.2681771	49.1624586366	0.0622077184482
160	-13467.8649372	44.7902655495	0.0636214443841
180	-13306.8956938	41.5605783281	0.0614516224447
200	-13183.9730324	38.6180843356	0.0611689653962

Tabela 1: *Amostragem dos resultados do K-Means sem PCA*

1) Análise dos medoids: Medoide é um objeto que representa o cluster, é escolhido o objeto mais próximo do centroide como o medoide daquele cluster. Aqui vemos alguns medoids para avaliar a qualidade da clusterização.

2) Com 80 clusters: Selecionamos alguns medoids do modelo com $k = 80$ para analise, e pegamos o assunto principal do texto:

- Cluster 1:
  - Email 8518: venda de um pré-amplificador;
  - Email 8545: doutrina de Deus.
- Cluster 2:
  - Email 300: Baseball e sobre o melhor jogador da liga;
  - Email 14728: Empresa agrícola privada ”DINA”.
- Cluster 3:
  - Email 442: Compra de um sintetizador;
  - Email 1581: Natureza.
- Cluster 4:
  - Email 15899: Grampo do governo e criptografia;
  - Email 18219: Entrevista para ESPN em que Stanky diz que é polones, e não judeu.
- Cluster 5:
  - Email 17142: Um lance injusto em uma partida de hockey no gelo;
  - Email 2009: Problemas com MacBook.

3) Com 47 clusters:

- Cluster 1:
  - Email 11724: Lista de perguntas e respostas para uma ferramenta do Windows;
  - Email 18438: Coleções de objetos, como armas e vasos.
- Cluster 2:
  - Email 9778: Dúvida com o uso do compilador GCC;
  - Email 17347: Astronomia.
- Cluster 3:
  - Email 285: ”Charlanismo” em alguns tratamentos médicos alternativos;
  - Email 1502: Foguetes.

4) Conclusão da análise dos Medoids: Pela análise superficial dos medoids, os clusters contem textos muito pouco relacionados.

B. K-Means com PCA

Foram testados diversos valores para a redução de variância de dados (0.85, 0.90, 0.93, 0.95 e 0.97). Apesar do pré-processamento e da redução de dimensionalidade, não foram observadas vantagens, tanto no custo computacional quanto no resultado do modelo. Dentre estes, o modelo que teve melhor desempenho foi o modelo com redução de 0.85. Possivelmente isso é explicado por que ao, reduzimos a dimensionalidade, nós também diminuimos a distância euclidiana dos membros dos cluster, o que resulta em uma Silhouette mais consistente com o restante e um score mais próximo a 1. Nesse modelo, com PCA de 0.85, o número de cluster mais adequado foi 100. Seu *score* foi de -11398.3292007, *Calinski* de 73.4076255749 e *Silhoette* de 0.0704448614472.

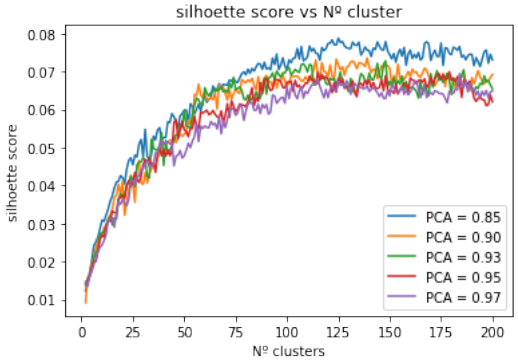

Figura 2: *Gráfico de k por Silhouette com PCA*

Para uma visualização gráfica aproximada da clusterização, reduzimos os dados para duas dimensões e obtivemos o seguinte gráfico de Voronoi $k = 80$:

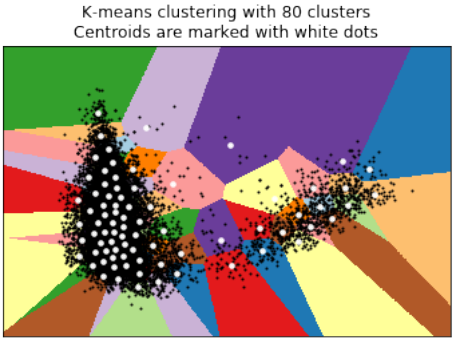

Figura 3: *Gráfico de Voronoi para a clusterização com 2 dimensões*

Apesar da grande redução de dimensionalidade, é possível observar uma tendência para o posicionamento dos dados em uma região específica, onde os cluster podem ter tido dificuldade em lidar com as intersecções das classes.

C. K-Means nas bases geradas

Utilizando conjunto de dados criados usados as *stopwords* e os *steamers* definidos na seção III, os emails originais foram pré-processados e criadas novas *bag-of-words*. Os modelos tinham alto custo computacional para serem gerados e processados e, para testes, rodamos com dois *k*s: 47 e 80. Esses números foram escolhidos após os resultados das seções anteriores. Todos as palavras foram convertidas para minúscula e as pontuações removidas. As palavras eram inseridas no *bag-of-words* se elas aparecessem em mais de 2 textos e em menos de 800, para obtermos dados mais representativos. Os resultados com diferentes stemmers foram muito próximos, sugerindo que todos tem desempenho muito similar. Apesar da baixa amostragem de apenas dois valores de *k*, os resultados tiveram métricas muito próximas e piores do que as encontradas nos testes anteriores, fazendo com que

estes testes fossem descontinuados.

Clusters	Stemmer	Score
47	WordNet	-18594.0481
47	Lancaster	-18567.6497
47	Snowball	-18562.8424
47	Porter	-18571.6947
80	WordNet	-18372.5459
80	Lancaster	-18336.7877
80	Snowball	-18332.9902
80	Porter	-18332.8160

Tabela 2: *Resultados obtidos com as bases geradas*

Contudo, uma vantagem deste modelo é conseguir verificar as palavras mais recorrentes de cada cluster, o que pode ser usado para verificar como o modelo está clusterizando os dados. Analisando, por exemplo, alguns clusters do modelo com WordNet e 47 clusters na tabela 3.

O cluster 4 se trata de assuntos ligados a esportes, o 31 também, porém com outros termos (incluindo baseball) enquanto o 46 parece se restringir ao baseball. Um email falando de baseball poderia permanecer na intersecção do 31 e 46, enquanto ele deveria de fato pertencer a ambos. Esse posicionamento nas margens dos clusters podem piorar o score do modelo.

Os clusters 1, 10 e 26 compartilham conceitos muito parecidos, caindo no mesmo problema de intersecção de assuntos. Embora os clusters consigam definir bem do que trata os assuntos dos seus componentes, o modelo aparenta falhar com assuntos que são muitos próximos, como, neste exemplo, cristianismo e religião.

Cluster	Palavras mais populares
4	hockey win fan goal playoff cup espn
31	player baseball hockey nhl league players
46	pitcher hit win ball pitching hitter
1	church bible faith christ catholic jesus
10	atheist religion atheism belief faith exist
26	jesus christ georgia covington heaven

Tabela 3: *Palavras mais populares de alguns clusters do modelo com Wordnet e 47 cluster.*

V. Conclusão

Os resultados dos modelos de clusterização não obtiveram *scores* razoáveis, levando a crer que o problema não seja ideal para o aprendizado não supervisionado. Conforme pode ser visto em [4], a divisão original era composta de 20 grupos, porém com intersecção entre os grupos, onde um mesmo email pode pertencer a mais de um dos grupos. Os emails que ficam nessas intersecções acabam por gerar um *score* ruim para o modelo, quando na verdade o ideal seria que o email pertence-se a mais de um dos clusters. A intersecção de clusters não é suportada pelo K-Means.

Acreditamos que a quantidade mais adequada de cluster é por volta de 90 para o algoritmo do K-Means, devido a sua aparente estabilização. A performance dos modelos em geral foi muito similar e nenhum deles é útil na prática.

VI. Estudos futuros

Para conseguir resultados mais promissores, estudos futuros podem abordar outros pré-processamentos de textos e modelos supervisionados de classificação, já que a base de dados disponibiliza rótulos e classes sobres seus dados.

References

[1] Guzella, T. S., & Caminhas, W. M. (2009). A review of machine learning approaches to spam filtering. Expert Systems with Applications, 36(7), 10206-10222.

[2] https://gmail.googleblog.com/2011/03/new-in-gmail-labs-smart-labels.html

[3] Tan, P. N. (2006). Introduction to data mining. Pearson Education India.

[4] http://qwone.com/~jason/20Newsgroups/

[5] Scikit-learn: Machine Learning in Python, acessado em 31 de Agosto de 2017, em http://scikit-learn.org/stable/index.html

[6] http://snowballstem.org/

[7] http://wordnet.princeton.edu/

[8] https://tartarus.org/martin/PorterStemmer/

[9] http://www.scientificpsychic.com/paice/paice.html

[10] https://ongspxm.github.io/blog/2014/12/bag-of-words-natural-language-processing/

[11] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[12] http://scikit-learn.org/stable/modules/clustering.html#calinski-harabaz-index

[13] www.facom.ufu.br/~backes/pgc204/Aula09-Agrupamentos.pdf

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