Aprendizagem Automática -Trabalho Laboratorial

Classificação de Críticas de Cinema do IMDb

ISEL 2020/2021

Docente: Gonçalo Marques

Aluno: Duarte Domingues 45140

Introdução

- Este trabalho prático baseia-se na resolução de um conjunto de tarefas, associadas a uma base de dados com textos de críticas de cinema de IMBD. A IMDb (de Internet Movie Database) é um conjunto de críticas de cinema recolhidas por Andrew Maas, um investigador da universidade de Stanford. Utilizou-se a biblioteca sklearn , uma biblioteca que contém muitas ferramentas uteis para *machine learning*.
- Para este trabalho prático os dados contêm 40 000 documentos de texto, classificados com uma pontuação numa escala de um a dez, em que as críticas neutras, com pontuação de 5 e 6, foram excluídas.
- Nas áreas de processamento natural de linguagem e recolha de informação a base de dados é chamada corpus.
- Para se poder realizar técnicas de aprendizagem automática como classificação e clustering em dados de texto, é necessário representar cada documento por um vetor numérico, para isto foi utilizado o modelo ***Bag of Words*** e o método **tf-idf**.
- Com base nos documentos de texto, três tarefas principais foram realizadas, respetivamente:

1. Determinar se a critica é positiva ou negativa (**processo de classificação binária**).
2. Prever a pontuação da critica, num valor entre 1-4 e 7-10 (**processo de classificação multiclasse**).
3. Descobrir através de palavras comuns entre diferentes textos, grupos de documentos que abordam áreas ou temas semelhantes (**processo de *clustering***).

Pre-Processamento de texto

- Apôs importar os dados é necessário realizar pre-processamento do texto, onde se divide cada documento em palavras ou "tokens". Os processos de limpeza de texto foram os seguintes:

1. eliminar carateres especiais, espaços não desejados e descartar palavras não desejadas. Decidiu-se não eliminar os números, e o carater " ' " devido a estes serem benéficos para os resultados obtidos.
2. Aplicação de ***Stemming***, técnica de transformar uma palavra na sua raiz, esta raiz pode não ser uma palavra real, mas apenas uma forma canónica da palavra original. Este processo é útil para estandardizar o vocabulário e reduzir o seu tamanho, melhorando o processo de classificação. Existem diversos tipos de *Stemming*, foram testados três algoritmos de *stemming*: ***SnowballStemmer***, ***PorterStemmer*** e ***LancasterStemmer***.
3. Remoção de ***Stop Words*** , baseia-se na remoção de palavras duma dada língua que ocorrem frequentemente, de modo a reduzir o tamanho do vocabulário. Visto que algumas das palavras da lista de *stop words* podem ser uteis no processo de classificação, e como se usou n-gramas, que vai ser abordado mais a frente, o uso de *stop words* seria negativo, logo, não foi usado no processo de classificação. Entretanto, no processo de *clustering* as *stop words* foram removidas, pois estas não adicionam algum tipo de beneficio no processo de agrupamento de dados.

- Exemplo de uma critica apôs pre-processamento dos documentos de texto:

```
the movi was sub par but this televis pilot deliv a great springboard into what has becom a sci fi fan ideal program
the actor deliv and the special effect for a televis seri are spectacular have an intellig interest script doesn't hu
rt either stargat sg1 is current one of my favorit program
```

Modelo Bag of Words e representação tf-idf

- O modelo Bag of Words, é um modelo de representação de texto que descreve a ocorrência de palavras dentro de um documento de texto, tendo os documentos de texto divididos por tokens como explicado no slide anterior. Involve duas componentes:

1. **Vocabulário** – vocabulário de todos os tokens no corpus.
2. **Codificação** – Consiste em contar o número de ocorrências de tokens por documento, representado cada documento por um vetor de d dimensões, um por cada token no vocabulário, com valores proporcionais ao numero de ocorrências dessa palavra no documento. O resultado final é uma matriz documento – termo (N * D) em que N é o número de documentos e D é a dimensão do vocabulário.

- Foi usado também utilizado o método TF-IDF, um método que procura refletir a importância de uma palavra no corpus, onde um termo com peso maior é considerado mais relevante num corpus, alterando forma da matriz documento – termo. A ideia é dar um peso superior a palavras que aparecem frequentemente num documento especifico, mas não em muitos documentos do corpus. Este método divide-se em duas etapas:

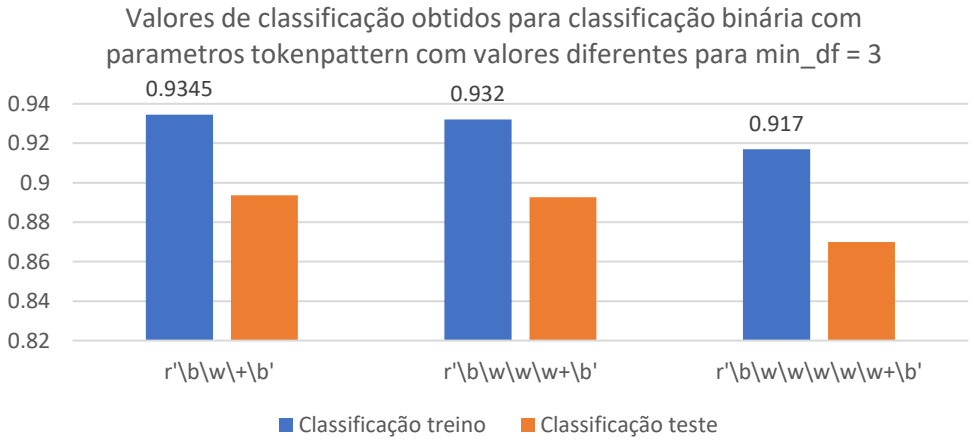
1. **Term Frequency** — Número de ocorrências normalizado de p em t.
2. **Inverse Document Frequency** – Mede a importância de um termo num conjunto de documentos.

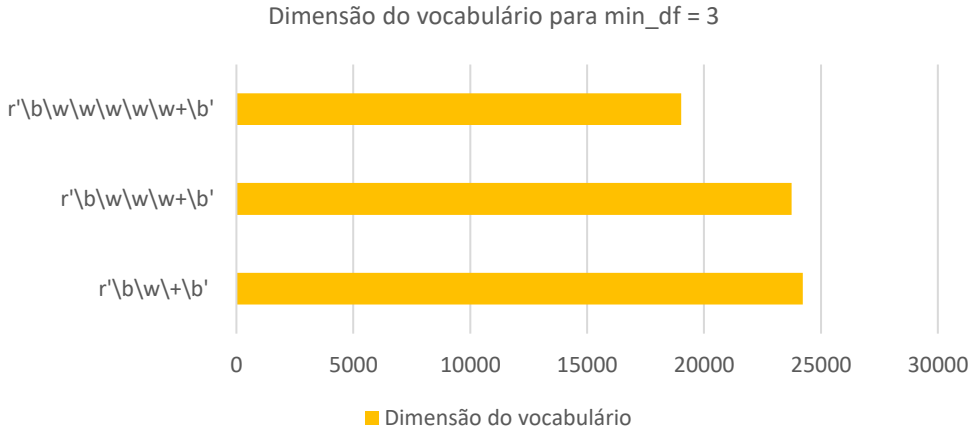
$$\text{tf-idf}(p,t) = \text{tf} \times \left(\log\left(\frac{N+1}{N_p+1}\right)+1\right)$$

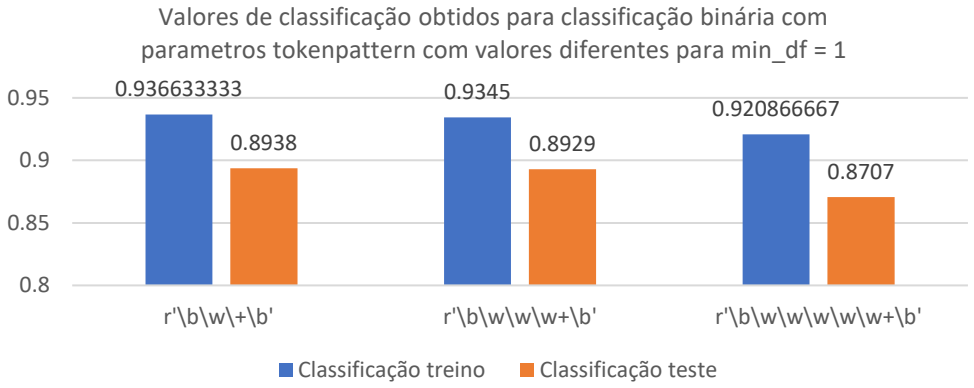
- O sklearn tem funções que permitem realizar a transformação tf-idf, com parâmetros que oferecem possibilidades uteis no processo de conversão de documentos de texto para a matriz de tf-idf features, tendo sido analisado os parâmetros opcionais para o desempenho dos modelos projetados.

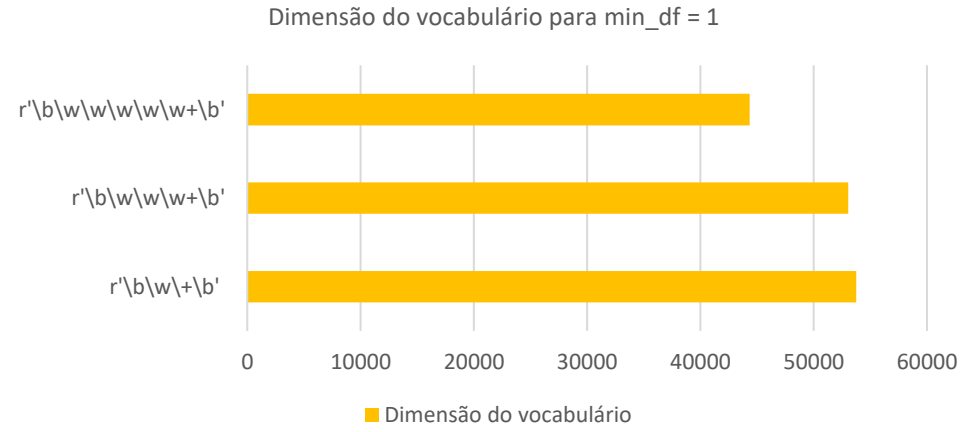
Análise dos parâmetros na função TfidfVectorizer

- A função **TfidfVectorizer** tem dois parâmetros, **mindf** e **token_pattern,** que influenciam significativamente o tamanho do vocabulário obtido e os resultados alcançados na classificação. O parâmetro mindf só extrai palavras que aparecem num número N de documentos e o tokken_pattern permite especificar o número de sequência de carateres mínimos permitidos.

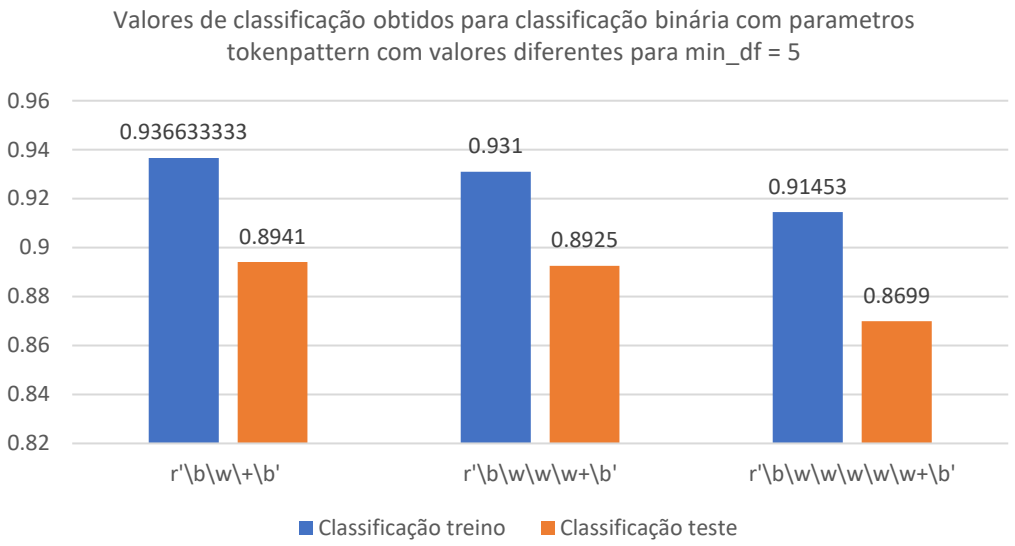


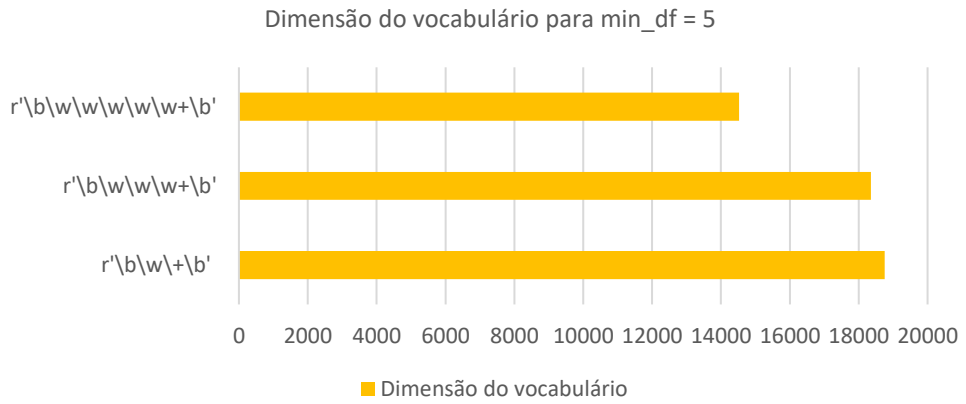






Análise dos parâmetros na função TfidfVectorizer





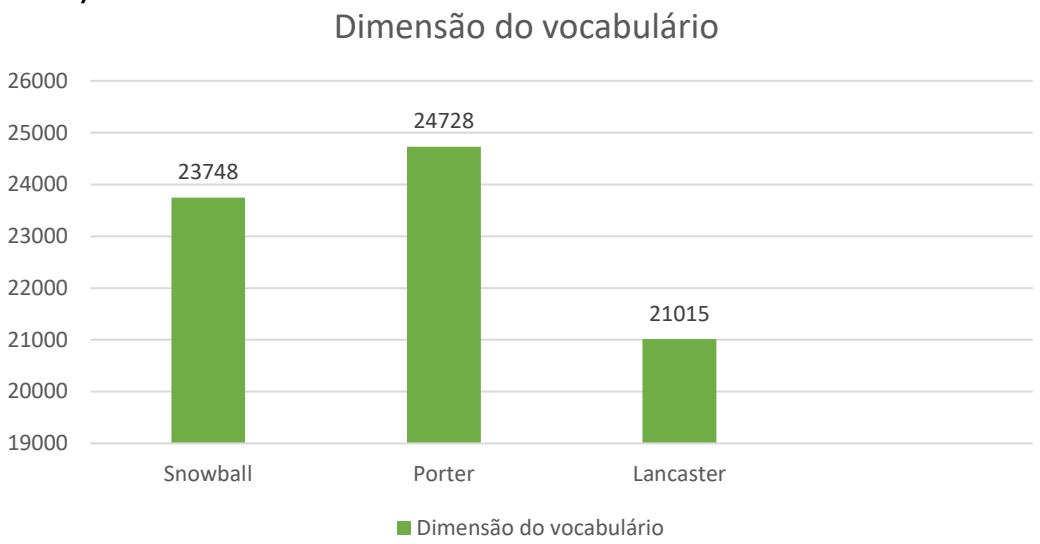
- Através dos seguintes gráficos pode-se observar que, quanto maior for a sequência de carateres no processo de tokenização, melhor serão os resultados obtidos. Porém quanto maior for a sequência de carateres, maior será o vocabulário, apesar de não haver grandes alterações entre token_pattern=r'\b\w\w\w+\b' e token_pattern=r'\b\w\+\b' devido a existirem poucas palavras no vocabulário com menos que três carateres.

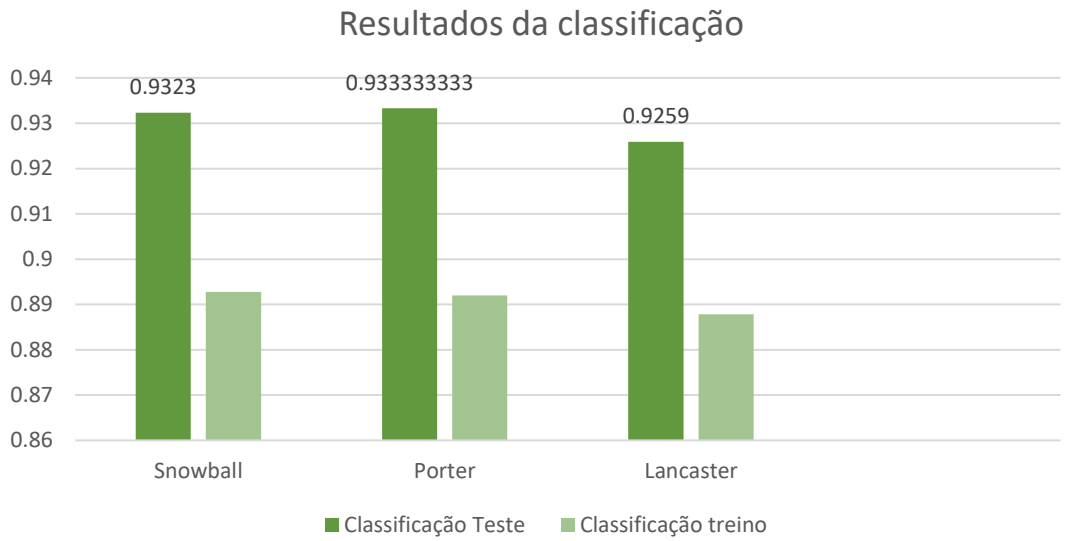
- Entendeu-se também que, o tamanho do vocabulário é altamente influenciado pelo parâmetro min_df, quanto menor for o valor de min_df, maior é o tamanho do vocabulário. Apesar do parâmetro min_df alterar significantemente o tamanho do vocabulário notou-se que este parâmetro não foi muito decisivo nos valores obtidos para a classificação, porém notou-se que os valores obtidos, foram ligeiramente superiores consoante o parâmetro min_df fosse inferior.

- É de notar que a qualidade dos resultados obtidos é proporcional à dimensão do vocabulário.

Processo de escolha do melhor algoritmo de *Stemming*

- De forma a escolher o melhor algoritmo de ***stemming***, foram realizados testes de modo a analisar como diferentes tipos de *stemming* iriam afetar os resultados obtidos para a classificação binária, e o tamanho do vocabulário. Foram usados os parâmetros((min_df=3,token_pattern=r'\b\w\w\w+\b')) no tfidf, e (penalty='l2',max_iter=1000,solver='saga',C=1.1,tol=1e-3) no classificador binário.

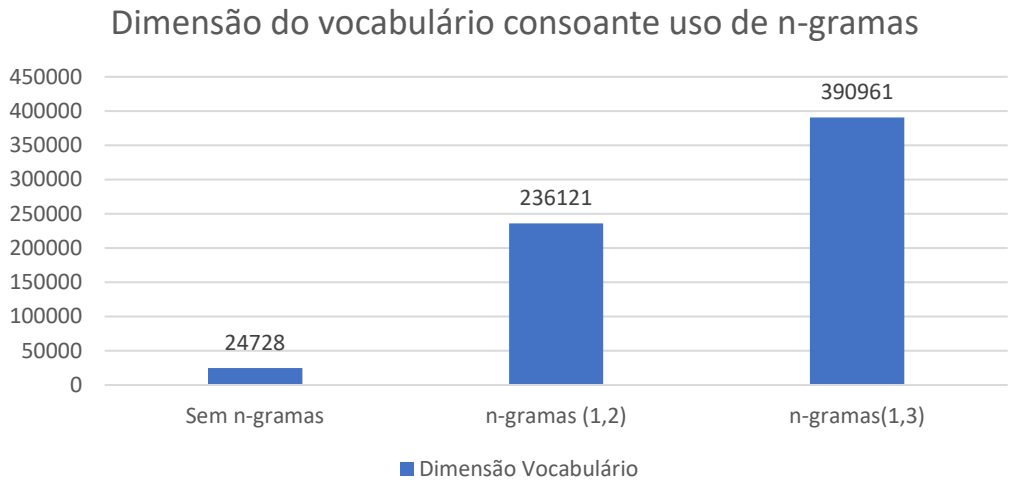


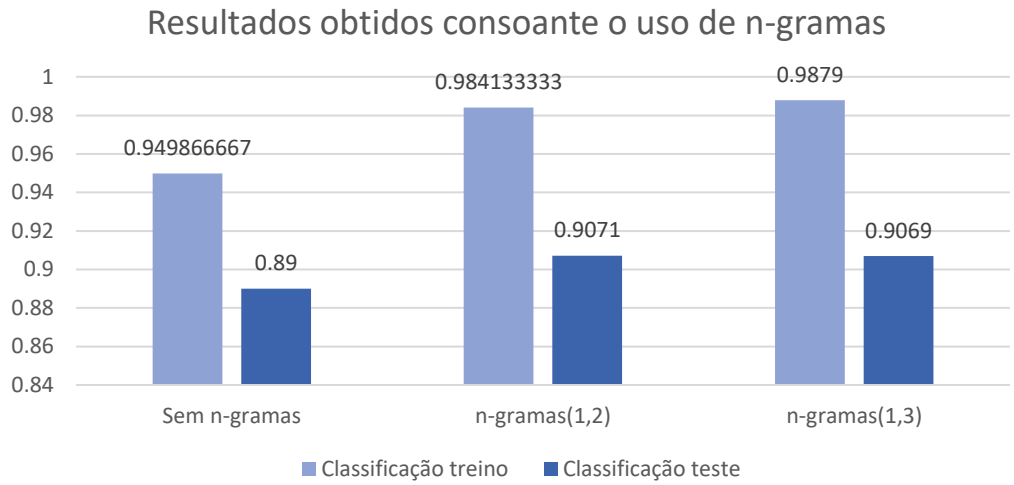


- Como se pode observar a partir dos resultados, o algoritmo de *Snowball* e *Porter* traduzem-se em vocabulários de tamanho semelhantes com resultados semelhantes. Entretanto o algoritmo de *Lancaster* realiza um *stemming* mais intenso reduzindo de forma mais significativa o tamanho do vocabulário, mas obtendo resultados ligeiramente mais fracos.

Análise da utilização de n-gramas

- N gramas é um processo que permite incluir como tokens não só as palavras isoladas, mas também a sequência de duas ou mais palavras que habitualmente apareçam nos documentos, conjunto de duas palavras são denominados bi-gramas, conjunto de três palavras tri-gramas, e em geral, n-gramas. Com este método irão aparecer sequencias de palavras. Ao utilizar n-gramas é possível capturar parte do contexto de uma frase no modelo ***bag of words***.





- Como se pode observar nos gráficos de cima, o uso de n-gramas aumenta exponencialmente a dimensão do vocabulário, o que vai aumentar consideravelmente o tempo de processamento do processo de classificação.
- Apesar do aumento do vocabulário, a utilização de bi-gramas ou tri-gramas melhora substancialmente os resultados obtidos na tarefa de classificação, logo o uso de n-gramas é benéfico para o desempenho dos modelos projetados. Como o uso de tri-gramas quase que duplica a dimensão do vocabulário e não mostra ter uma vantagem relevante em relação ao uso de bi-gramas averiguou-se que bi-gramas é o mais apropriado neste contexto.

Discriminante logístico comparação com regressão linear

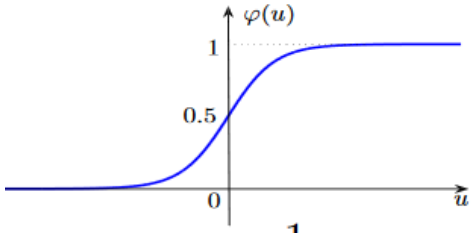
- Neste trabalho prático na tarefa de classificação, quer seja binária ou multi-classe, estamos a lidar com dados categorizados, para o problema binário os dados estão classificados entre 0 e 1, e para multi-classe numa escala de 0 a 10, excluído 5 e 6.
- Para o processo de classificação de dados categorizados o uso de descriminantes logísticos usa-se em vez de discriminantes lineares. Descriminantes lineares funcionam melhor para prever valores contínuos, elaborando regressão, em que o valor predito pode ser um número qualquer, não um valor dentro de um número de categorias especificas, por estes motivos usou-se discriminantes logísticos no processo de classificação neste trabalho prático.
- Regressão linear tem o objetivo de mapear uma ou mais variáveis independentes (Xn) para uma variável dependente (y). A ideia é traçar uma reta para o y estimado segundo a seguinte fórmula:

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d = \underbrace{\begin{bmatrix} w_0 & w_1 & \cdots & w_d \end{bmatrix}}_{\mathbf{w}^T} \underbrace{\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}}_{\mathbf{x}} = \mathbf{w}^T\mathbf{x} = \mathbf{x}^T\mathbf{w}$$

- Método de funcionamento da regressão linear baseia-se em tentar encontrar os pesos (w0 , w1, wn…) que levam a encontrar a melhor linha para o valor estimado de y consoante as variáveis independentes. Melhor linha é determinada em termo do custo menor.
- O valor ótimo de w é estimado a partir do método dos mínimos quadrados. Técnica que minimiza a função de custo do erro quadrático medio. Este método permite estimar analiticamente a solução minimizando o erro quadrático médio entre a predição do modelo e o valor desejado.
- Para determinar a qualidade dos resultados do modelo é usado o coeficiente $\boldsymbol{R^2}$, que indica a percentagem da variância na variável dependente que é predita pelas variáveis independentes.

Discriminante logístico

- A regressão logística semelhantemente à regressão linear usa a combinação dos pesos das variáveis independentes, mas agora passa estes valores por uma função sigmoide, convertendo o valor estimado de y para uma gama entre [0 a 1], a partir de um ***threshold***, neste caso 0.5.



$$\varphi(u) = \frac{1}{1 + e^{-u}}$$

$$\boldsymbol{u} = \boldsymbol{W}^T\boldsymbol{x}$$

- Em relação ao caso do uso de descriminante logístico em classificação multi classe podem ser usados os algoritmos de treino ***one-vs-rest*** ou *cross-entropy.*
- A estratégia *one-vs-rest* divide uma classificação multi-classe em uma classificação binária por classe.
- Para realizar a classificação com descriminantes logísticos, foi usada a classe sklearn.logisticRegression. Esta classe tem um conjunto de parâmetros que devem ser otimizados de forma a ter os melhores resultados possíveis.
- Para poder fazer a classificação binária foi necessário converter as classificações de 0 a 10 para classificações binárias.

Escolher os melhores parâmetros para os discriminantes logísticos

- De modo a descobrir os melhores parâmetros para o discriminante logístico recorreu-se ao uso do *RandomizedSearchCV*. *RandomizedSearchCV* percorre de forma aleatória um conjunto de *hyper* parâmetros e calcula o score, medida de acurácia dos métodos de predição, retornando o melhor conjunto de parâmetros consoante o score.
- Com este método é possível controlar o número de diferentes combinações de parâmetros a ser testados, sendo o tempo de processamento bastante inferior do que o *GridSearchCV* , uma *class* do sklearn com funcionamento semelhante.
- Inicialmente é necessário especificar um dicionário com os parâmetros que queremos que sejam testados, vão ser testados parâmetros da classe ***TfidfVectorizer*** e da classe ***LogisticRegression*** .
- Parâmetros a ser testados :

1. **tfidf__min_df (*TfidfVectorizer)*** – controla as palavras que são extraídas, consoante o número de vezes que aparecem em documentos.
2. **tfidf__ngram_range (*TfidfVectorizer)*** – controla o tipo de n-gramas realizado.
3. **C (*logisticRegression*)**: Inverso do peso dado ao termo de regularização. Valores pequenos correspondem a uma regularização “forte”.
4. **Penalty (*LogisticRegression*)**: Modo de regulação, l1 igual a lasso, l2 igual a ridge.

- De seguida quando se chama a função é necessário ajustar os parâmetros do *RandomizedSearchCV*.

```
grid_search=RandomizedSearchCV(pipeline, parameters, cv = 3, n_jobs=-1, verbose=3,pre_dispatch=8,n_iter = 4 ).fit(X,y)
```

- No pipeline é passado o modelo que queremos que seja testado, nos *parameters* é passado o dicionário de parâmetros que queremos que seja testado, cv=3 especifica o número de splits necessárias para ocorrer validação cruzada, a validação cruzada é uma técnica para avaliar a capacidade de generalização de um modelo, a partir de um conjunto de dados. n-iter controla o número de combinações aleatórias que vão ser testadas e n-jobs está relacionado com o desempenho do CPU.

Valores ótimos

- No processo de classificação os dados foram divididos em dados de treino e de teste, de forma aleatória, com 30000 amostras para treino e 10000 para teste, usando o método **train_test_split** do sklearn.
- Correndo o método de procura referido anteriormente, a combinação de melhores parâmetros para classificação binária foi a seguinte:

```
Pipeline(steps=[('tfidf',
                 TfidfVectorizer(min_df=2, ngram_range=(1, 2),
                                 token_pattern='\\b\\w+\\b')),
                ('logReg',
                 LogisticRegression(C=3.4, max_iter=1000, solver='saga',
                                    tol=0.001))])
```

- Resultados do *score* obtidos com estes parâmetros para a classificação binária foram:

```
Treino classificacao binaria 0.9859
Teste classificacao binaria 0.911
```

- Para a classificação multi-classe a estratégia que deu melhores resultados foi a *one-vs-rest,* definida no parâmetro **multi_class = "ovr"**, tendo sido utilizada uma regularização forte.
- Para a classificação multi-classe a melhor combinação de parâmetros foi a seguinte:

```
dl=LogisticRegression(penalty='l1',solver='saga',max_iter=1000,C=2.1,tol=1e-3,multi_class='ovr')
```

- Melhores resultados foram os seguintes:

```
Treino classificacao Multiclasse 0.5976666666666667
Teste classificacao Multiclasse 0.4787
```

Classificadores naïve bayes e ridge

- Além dos classificadores mencionados aleatoriamente, foi testado também um classificador naive bayes para classificação binária, BernoulliNB . O classificador mencionado é adequado para trabalhar com dados discretos, logo aplicam-se bem no contexto do trabalho prático.
- Para estes classificador foram testados os melhores parâmetros de alpha e do fit_prior, os melhores parâmetros obtidos **fit_prior** = false, para não aprender prioridades de classes a priori, e **alpha** = 1.28, parâmetro de alisamento.
- Melhores resultados obtidos para o classificador BernoulliNB foram :

```
Treino classificacao binaria bernoli 0.9883
Teste classificacao binaria bernoli 0.8895
```

- Foi também testado um classificador Ridge para classificação multi-classe, é baseado no método de regressão **Ridge**, converte os dados na gama =[1,1] e classifica com a abordagem *one-versus-all*. Neste classificador foi testado também o parâmetro alpha.
- Melhores parâmetros para o classificador Ridge:

```
RidgeClassifier(alpha=1.8).fit(X, y)
```

- Melhores resultados obtidos para o classificador Ridge foram :

```
Treino classificacao ridge 0.9656666666666667
Teste classificacao ridge 0.4785
```

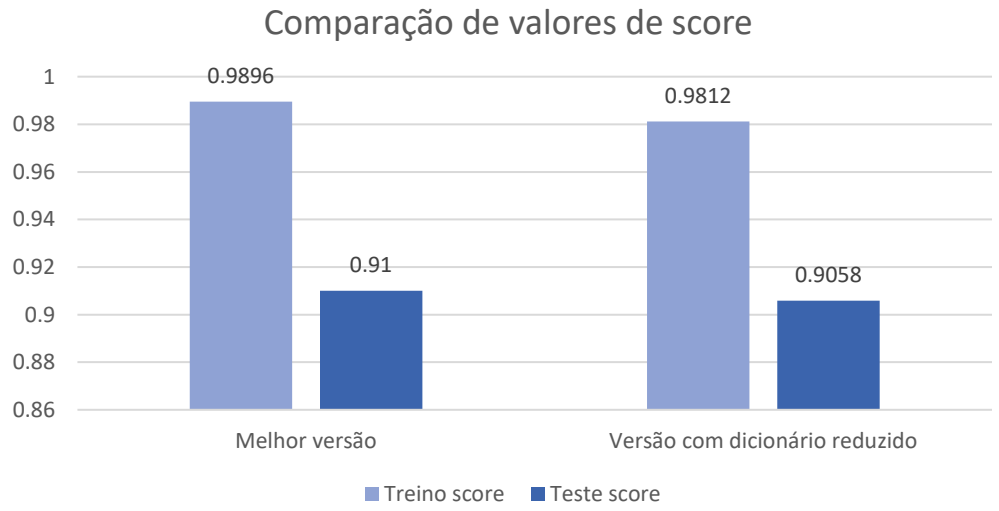
- É de notar que no classificador *Ridge* há uma discrepância muito elevada entre o valor de score de treino e de teste, o que é sinal de sobre aprendizagem.

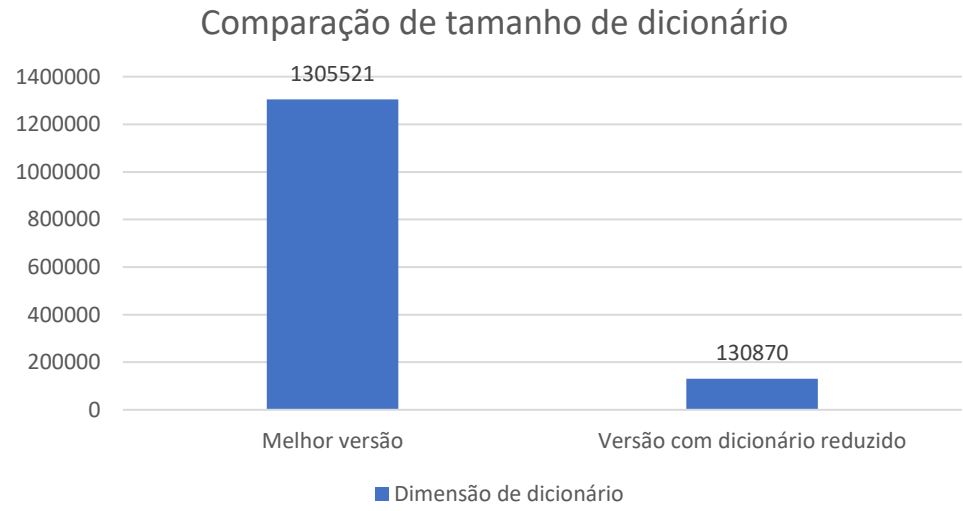
Dimensão minima de vocabulário

- Nesta etapa tentou-se averiguar qual a dimensão mínima do vocabulário para qual o desempenho dos classificadores binários seja ainda próximo dos melhores resultados obtidos.
- De forma a reduzir a dimensão do vocabulário, foi escolhido o algoritmo de *stemming Lancaster*, como já comprovado anteriormente é o que reduz ao máximo o vocabulário e continua a ter resultados aceitáveis.
- Utilizou-se um classificador de regressão logística com os parâmetros definidos anteriormente.
- Para os parâmetros "*min_df*" e "*token_pattern*" na função tfidfVectorizer foram usados os parâmetros:

  TfidfVectorizer(min_df=5,token_pattern=r'\b\w\w\w+\b', gram_range=(1,2))

- Comparações entre os resultados obtidos entre os melhores resultados obtidos, e os resultados obtidos de modo a minimizar o tamanho do vocabulário:





Clustering

- A ideia principal do *clustering* é separar um conjunto de dados em um número definido K de clusters, que são grupos que abordam temas semelhantes, neste caso em dados não rotulados, logo é uma aprendizagem não supervisionada.
- O tipo de *clustering* utilizado foi o **K-means,** neste algoritmo o espaço vetorial é dividido por conjuntos de clusters consoante o centroide do cluster. No espaço vetorial os centroides são colocados de forma aleatória e movem-se para o centro dos pontos mais próximos de eles mesmos. A localização dos centroides é atualizada iterativamente, sendo as posições dos centroides alterada a cada iteração, quanto maior for o número de iterações melhor serão os resultados obtidos.
- No processo de pre-processamento de texto para o *clustering* removeram-se **as *stop words***, devido a estas não trazerem qualquer beneficio para separar os dados por grupos.
- Foram removidas um grupo de palavras, que apareciam regularmente nas palavras mais relevantes no cluster, porém, não adicionavam nenhum beneficio para separar os dados por grupos, respetivamente:

```
['film', 'movi', 'like', 'make', 'watch', 'realli', 'just', 'act', 'seen', 'plot', 'time', 'scene', 'end', 'look', 'a
ctor', 'don', 'think']
```

- A melhor combinação para os parâmetros de *tfidfVectorizer* para o processo de *clustering* foi a seguinte:

```
TfidfVectorizer(min_df=5,token_pattern=r'\b\w\w\w+\b',stop_words="english")
```

- Esta combinação de parâmetros permite obter um vocabulário de dimensão equivalente a **21002**, uma dimensão relativamente curta, mantendo bons resultados. Se o valor de min_df fosse menor teriam sido tomadas em conta algumas palavras pouco relevantes para o processo de clustering, logo min_df=5 pareceu ser o valor ideal, palavras com menos comprimento que três também não adicionavam grande beneficio para os resultados obtidos e iria aumentar a dimensão do dicionário significativamente.

Análise em componentes principais

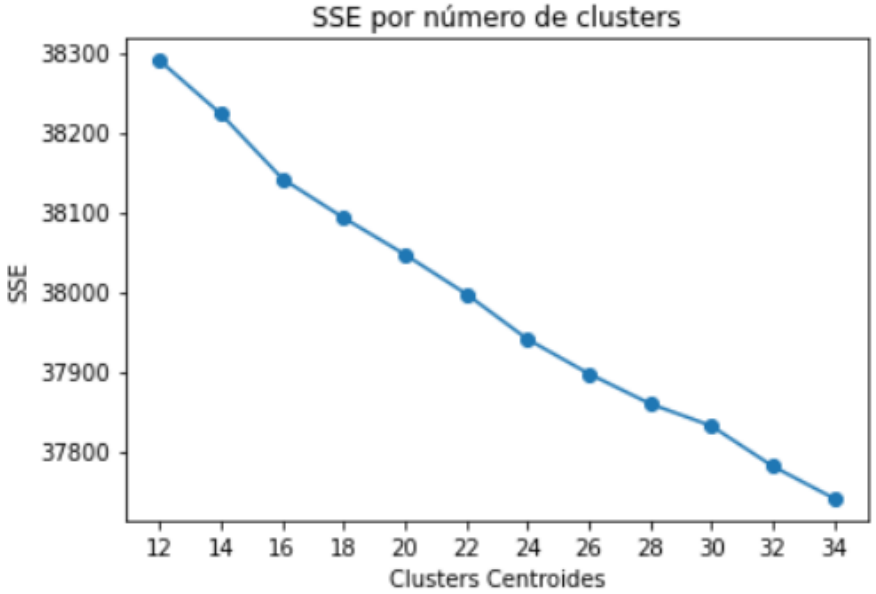
- Como o algoritmo k-means em bases de dados de tamanho elevado, como neste caso, não tem o melhor desempenho, optou-se por realizar Análise em componentes principais para reduzir a dimensionalidade dos dados.
- Utilizou-se a função **TruncatedSVD** de modo a poder lidar com matrizes esparsas.
- No processo de escolha do número de componentes escolheu-se um número cujo a soma da percentagem da variância explicada por cada fosse igual a cerca de 76 porcento, neste caso 3800, de forma a preservar grande parte da variância original.
- Desta forma a dimensionalidade dos dados é reduzida de 21002 para 3800, uma redução da dimensionalidade dos dados para 18 porcento do tamanho original.
- Para estas medições foi usada a função **svd.explained_variance_ratio_.sum()**.

```
svd = TruncatedSVD(n_components=3800)
```

```
soma da percentagem da variância  por componente  0.7577029161496971
```

Escolher o número de clusters inicial

- Para escolher o número inicial de k-means usou-se o **método "elbow"**, este método baseia-se em correr o k-means para um gama de valores de K (número de clusters) e para cada valor de K calcular a soma dos erros quadrados. De seguida faz-se um "plot" da soma dos erros quadrados , parte do gráfico que parecer um "cotovelo" será o melhor valor para k.
- Alem do método "elbow", utilizou-se o **método Silhouette**, este métodos serve para analisar a distância de separação entre os clusters resultantes. O silhouete score é calculado a partir da média da distância de pontos dentro de um cluster e a média da distância dos pontos até ao cluster vizinho mais próximo.
- Pela análise dos gráfico e dos valores de *Silhouete score*, escolheu-se 30 para o número inicial de clusters.
- Como se pode observar a partir dos dados, com maior número de clusters a soma dos erros quadrados diminui e o score Silhoute normalmente diminui.

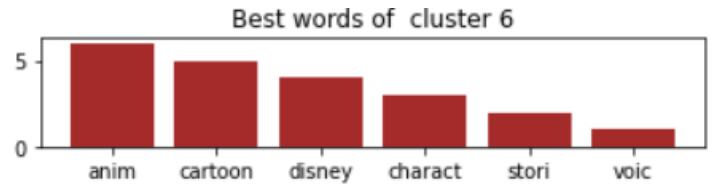


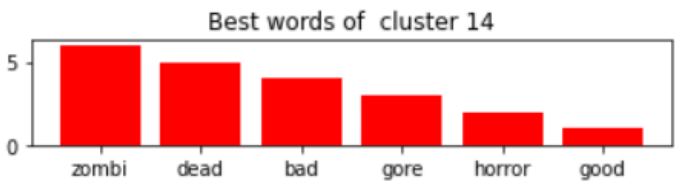

```
12: 0.0023617951447074 83
14: 0.0028008031777785837
16: 0.0032758390199061757
18: 0.00345167515476532
20: 0.0036283588356641034
22: 0.0038304414041944552
24: 0.003975960894574218
26: 0.00424091856874147
28: 0.004366857856273622
30: 0.00458448462804836 1
32: 0.004898221146186713
34: 0.004815504071512248
```

Silhouette score por número de clusters

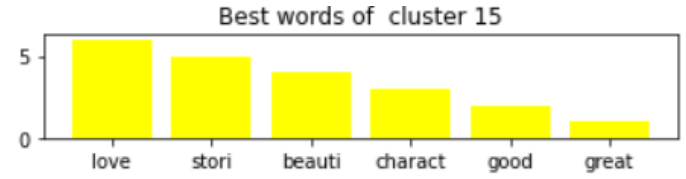
Resultados obtidos

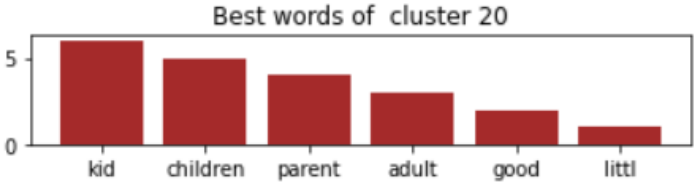
- Foram analisadas as seis palavras mais relevantes por cluster de forma a identificar palavras relevantes a grupos especificos por cluster, de modo a medir o sucesso do processo de clustering.

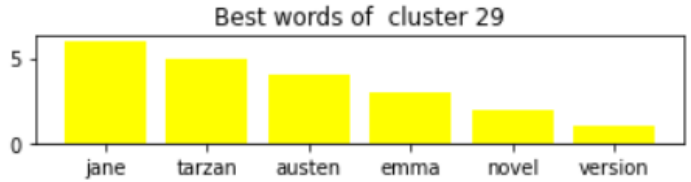




Resultados obtidos







- Como se pode observar, a maior parte dos clusters relacionam-se com um tema especifico, como por exemplo no cluster 27 video-jogos e no cluster 20 conteudo infantil.
- A partir do clustering foi possivel separar as criticas em grupos especificos.

Conclusões

- A elaboração desta primeira fase do trabalho serviu para colocar em prática o conhecimento adquirido na disciplina de Agentes Autónomos numa situação real, como por exemplo, regressão linear, análise em componentes principais e métodos de agrupamento de texto.
- Neste trabalho prático aprendeu-se a lidar com textos de criticas realizando diferentes processos de classificação e um processo de *clustering*.
- Constata-se que neste trabalho prático aprendeu-se a otimizar os parâmetros para diferentes classes da biblioteca sklearn referentes a processos de aprendizagem automática, recorrendo a uma análise critica de resultados e ao uso de classes como o *RandomizedSearchCV* que permite medir combinações ideias de diferentes parâmetros.
- Esta disciplina ajudou a desenvolver uma visão critica em relação aos resultados obtidos para diferentes tipos de problemas.

Bibliografia

- Slides disponibilizados no moodle ISEL 2020/2021 na unidade carricular Agentes Autónomos pelo docente Gonçalo Marques
- **https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html**
- **https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568**
- **https://towardsdatascience.com/logistic-regression-model-tuning-with-scikit-learn-part-1-425142e01af5**
- **https://scikit-learn.org/0.16/modules/generated/sklearn.grid_search.RandomizedSearchCV.html**
- **https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python/**