Projeto de NLP:

Análise de Sentimento de comentários de filmes

Alessandro Silva Angeruzzi

23/12/2021

Sumário

1. Introdução

Este projeto consiste na implementação de um modelo de predição baseado em Análise de Sentimento, que possa avaliar comentários em inglês de filmes e classificá-los como negativos ou positivos.

Os fontes do projeto se encontram em: https://github.com/angeruzzi/NLP_AnaliseSentimento_ComentariosIMDB

2. Sobre os dados

Para treinamento e testes do modelo foi utilizado um dataset disponibilizado pelo pesquisador Andrew Maas (https://ai.stanford.edu/~amaas/) no link http://ai.stanford.edu/~amaas/data/sentiment/ .

Estes dados foram utilizados originalmente no artigo "Learning Word Vectors for Sentiment Analysis" (https://ai.stanford.edu/~amaas/papers/wvSent_acl2011.pdf).

Foi disponibilizado 50 mil avaliações de filmes coletadas do site IMDB (https://www.imdb.com/) sendo no máximo 30 avaliações do mesmo filme, visto que as críticas para o mesmo filme tendem a ter classificações próximas. Esse número de avaliações é composto por 25 mil positivas e 25 mil negativas, separadas igualmente também já em dados para treinamento e para testes.

A classificação fornecida de cada comentário foi baseada nas notas de 1 a 10 que acompanhavam os mesmos, sendo notas menores ou iguais a 4 consideradas como negativas e notas maiores ou iguais a 7 positivas. Os comentários com notas 5 e 6 não foram considerados para se evitar avaliações neutras.

Na pasta disponibilizada há dois diretórios, [train] e [test], e em cada um contém outros dois diretórios, [pos] e [neg], que contêm os arquivos sendo um por comentário.

O título de cada arquivo segue a seguinte convenção: [[id] _ [rating] .txt] , onde [id] é um id único e [rating] é a avaliação com estrelas do comentário em uma escala de 1 a 10.

Além dos arquivos de comentários, foram incluidos arquivos de Bag of words (BoW) no formato LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/), que foram utilizados nos seus experimentos, e um arquivo com a classificação esperada de algumas palavras já calculadas por Potts,2011.

3. Conceitos

Segue a definição de alguns conceitos e técnicas que serviram de base para o desenvolvimento desse projeto.

3.1. NLP – Natural Language Processing

NLP, ou como conhecido na sigla em português PLN (Processamento de Linguagem Natural), é uma subárea da IA (Inteligência Artifical) que tem como objetivo o "entendimento" da linguagem do ser humano para uso computacional.

Esta área remete a 1950 quando Alan Turing publicou o artigo *"Computing Machinery and Intelligence"* onde propôs o Teste de Turing (https://en.wikipedia.org/wiki/Turing_test) para definição da inteligência de máquinas.

Até 1990 a *Symbolic NLP* era a linha de pesquisa de dominância na NLP, que se baseia na criação de regras para entendimento das linguagens e de forte cooperação com a área Linguística.

Com o avanço computacional se abriu espaço para a *Statistical NLP* com a introdução de representações estatísticas e algoritmos de machine learning para o processamento de linguagem.

Atualmente a linha mais promissora é a *Neural NLP,* com o uso de *Deep Neural Networks* (Redes Neurais Profundas), que tem conseguido resultados promissores, porém de treinamento difícil e caro.

3.2. Análise de sentimentos

A Análise de Sentimento é uma de várias aplicações da NLP, que consiste na avaliação de trechos de texto, comentários ou opiniões acerca de algo e a identificação do sentimento (geralmente binário, positivo ou negativo) ali inserido pelo autor.

3.3. Modelo Bag of Words

O modelo Bag of words (bow) é um tipo de representação de texto para uso em problemas de NLP; basicamente consiste em uma matriz em que as colunas são palavras de um vocabulário definido, e cada linha representa uma frase onde as células indicam a presença ou não da palavra da coluna na frase, podendo ser preenchida de forma binária, por contagem ou outra representação como a tf-idf.

Por exemplo, tomemos 2 frases:

1) Eu não gostei do trailer e nem do filme

2) Eu gostei desse filme

Representação das frases com Bag of words utilizando contagem simples:

	Eu	**não**	**gostei**	**desse**	**do**	**trailer**	**e**	**nem**	**filme**
Bow1	1	1	1	0	2	1	1	1	1
Bow2	1	0	1	1	0	0	0	0	1

3.4. TF-IDF

O Tf-Idf (*term frequency–inverse document frequency*) é uma medida estatística que indica a importância de uma palavra de um texto em relação a uma coleção de textos, podendo ser utilizada com o modelo de bag of words e apresentar vantagens em relação a uma contagem simples da palavra por exemplo.

O valor tf–idf de uma palavra aumenta proporcionalmente à medida que aumenta o número de ocorrências dela em um texto, porém esse valor é equilibrado pela frequência da palavra no conjunto de textos de treinamento, com isso se distingue se a palavra é importante no texto avaliado ou simplesmente é uma palavra mais comum na língua que está sendo utilizada.

3.5. Técnicas de limpeza dos dados

Para a melhor avaliação de um texto é importante deixá-lo mais enxuto, removendo caracteres ou palavras que podem atrapalhar ou mesmo que não agregam muita importância ao conteúdo dele, logo é importante remover itens como Caracteres especiais, de pontuação ou estranhos a língua; pode-se avaliar também a remoção de algumas classes de palavras que não tragam impacto na avaliação, como artigos, preposições e pronomes.

3.6. Análise pela Morfologia

Existem diversos tratamentos que podem ser aplicados no texto do ponto de vista da morfologia, neste projeto utilizei 2: Stemming e Lemmatization.

Podemos ter palavras que representam o mesmo sentido, porém são diferentes devido a flexão (gênero, número e grau) ou mesmo conjugação no caso de verbos, ter uma representação única dessas palavras pode trazer grandes benefícios para o modelo de avaliação, e é isso que as duas técnicas propõem.

A Stemming consiste em reduzir as palavras para o seu radical, já a Lemmatization para o lema da palavra.

As bibliotecas utilizadas para a aplicação foram “Porter” para Stemmer e a “Wordnet” para Lemmatization, ambas disponíveis no pacote NLTK do python.

3.7. Modelos de Machine Learnig

Os modelos de machine learning são algoritmos definidos para uso em uma classe de problema específica (exemplo Regressão ou Classificação) mas de uso genérico quanto aos dados de entrada. Após treinado com determinados dados o modelo passa a ser específico para um tipo de problema.

Os algoritmos utilizados nos testes foram 3 inicialmente, sendo os mais comuns encontrados na literatura na aplicação de problemas de NLP, são eles: Logistic Regression, SVC e Multinomial Naive Bayes; incluí mais 2, que são menos frequentes neste contexto, mas também utilizados, o Decision Tree e o Random Forest, justamente com a intenção de ter uma comparação com modelos não lineares.

Após os testes iniciais, dados os resultados do modelo SVC, inclui mais 2 modelos de conceito próximo a ele, o Linear SVC e o SGD Classifier.

Foi utilizada a biblioteca Scikit Learn que possui a implementação de todos os modelos citados.

3.8. Tipos de Scores de modelos de classificação

Existem métricas para avaliação dos resultados dos modelos, que são utilizadas principalmente de forma comparativa entre modelos e não absoluta.

Acompanhei 5 métricas de avaliação de modelos de classificação, todas baseadas na matriz de confusão dos resultados. A Matriz de confusão é uma tabela que confronta os resultados Preditos com os Reais (esperados) da seguinte forma:

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Com os valores da Matriz de confusão calcula-se as métricas

- **Acurácia** (Acuracy): Quantidade classificada como Positivos e Negativos corretamente
  (TP + TN) / (TP + TN + FP + TN)

- **Sensibilidade** (Recall): Taxa de acertos de Positivos em relação a todos verdadeiramente positivos
  TP / (TP + FN)

- **Precisão** (Precision): Taxa de acertos de Positivos em relação a todos preditos como positivo
  TP / (TP + FP)

- **Especificidade:** Taxa de acertos de Negativos em relação a todos verdadeiramente negativos
  TN / (FP + TN)

- **F1-Score:** média harmônica entre Precisão e Sensibilidade
  (2* TP) / (2* TP + FP + FN)

3.9. Tuning de Parâmetros

O tuning de parâmetros consiste na seleção de valores de parâmetros de um algoritmo de machine learning com o intuito de se obter um melhor resultado.

Existem algumas técnicas de tuning e para o projeto utilizei o método Randomized Search CV disponível no pacote do Scikit Learn. Este método realiza uma busca randômica entre os valores possíveis de cada parâmetro, pode não ser tão preciso quanto outros métodos que percorrem todo o range de valores definindos, porém possui uma melhor performance principalmente quando temos um grande volume de dados para treino.

4. Projeto

4.1. Estrutura de Diretórios

Os diretórios do projeto estão organizados da seguinte forma:

- **Raiz**
  - **corpus** (Arquivos para treino e teste do modelo)
    - **test** (Arquivos para testes)
      - **neg** (comentários negativos)
      - **pos** (comentários positivos)
    - **train** (Arquivos para treino)
      - **neg**
      - **pos**
  - **docs** (Documentação e Logs)
  - **models** (Modelo e Vocabulários gerados)
  - **source** (Fontes do projeto)
  - **topredict** (Arquivos para predição)

4.2. Módulos

Segue a lista de arquivos de fonte disponíveis na pasta source e a descrição de suas funções:

- **main**: Execução do sistema;
- **interface**: Funções para geração e controle do menu;
- **pipeline**: Sequências de execuções das ações disponíveis para o usuário;
- **loader**: Funções de carregamentos dos arquivos de textos;
- **cleaner**: Função de limpeza e tratamento dos dados;
- **vectorizer**: Funções de vetorização dos textos em BOW e geração dos vocabulários;
- **modeler**: Funções para treinamento, carregamento e teste dos modelos e predição de novos comentários;
- **selector**: Funções de apoio aos testes e análises de modelos;
- **analyzer1**: Sequência de treinos e testes dos modelos para análise;
- **analyzer2**: tuning do modelo de melhor score.

4.3. Instalação

As informações para download e configuração deste projeto para execução se encontram no arquivo README.md do repositório do projeto.

4.4. Execução

Para execução do projeto deve-se:

- A partir da pasta raiz acessar a pasta dos fontes: cd source
- executar: python main.py

Será aberto um menu oferecendo 4 opções, segue prints com simulação delas:

1 – Fazer predição com o modelo já treinado:

Os arquivos para predição devem estar salvos na pasta .\topredict

```
----------------------------------------
        DIGITE A OPÇÃO DESEJADA
----------------------------------------
1 - Fazer predição com o modelo já treinado
2 - Treinar novamente o modelo
3 - Treinar o modelo e executar os testes
4 - Sair
----------------------------------------
Sua Opção:1

Aguarde...
Carregando arquivos para predição
Tratando Dados
Vetorizando dados
Executando predições

Resultados da Predição :
 1 : Comentários Positivos
-1 : Comentários Negativos
       name                                               text  prediction
0   01_8.txt  I went to an advance screening of this movie t...           1
1   02_2.txt  This was a decent movie for the first half. To...          -1
2   03_4.txt  David Bryce's comments nearby are exceptionall...           1
3   04_1.txt  This movie cannot be serious because it has a ...          -1
4   05_9.txt  The many other comments about the film say it ...           1
5   06_3.txt  Rita Hayworth plays a Brooklyn nightclub dance...          -1
6   07_8.txt  I was looking forward to The Guardian, but whe...          -1
7  08_10.txt  This movie is so misunderstood it is not even ...           1
8   09_7.txt  I just recently watched this 1954 movie starri...           1
9   10_1.txt  I found this movie really hard to sit through,...          -1
```

2 – Treinar novamente o modelo:

Realiza novamente o treinamento do modelo e salvar para uso posterior.

```
----------------------------------------
        DIGITE A OPÇÃO DESEJADA
----------------------------------------
1 - Fazer predição com o modelo já treinado
2 - Treinar novamente o modelo
3 - Treinar o modelo e executar os testes
4 - Sair
----------------------------------------
Sua Opção:2

Aguarde...
Carregando arquivos de Corpus
Tratando Dados de treino
Vetorizando dados e gerando vocabulário
Treinado o modelo
Modelo treinado e salvo
```

3 – Treinar o modelo e executar os testes:

Realiza novamente o treinamento assim como a opção 2 mas também executa os testes exibindo os scores médios obtidos.

```
----------------------------------------
        DIGITE A OPÇÃO DESEJADA
----------------------------------------
1 - Fazer predição com o modelo já treinado
2 - Treinar novamente o modelo
3 - Treinar o modelo e executar os testes
4 - Sair
----------------------------------------
Sua Opção:3

Aguarde...
Carregando arquivos de Corpus
Tratando Dados de treino
Vetorizando dados e gerando vocabulário
Treinado o modelo
Modelo treinado e salvo
Carregando arquivos de teste
Tratando Dados de teste
Vetorizando dados de teste
Executando testes

Score médio dos Testes Realizados:
   Acurácia  Sensibilidade (recall)  Precisão  Especificidade (bac)  F1-score
0    0.8894                 0.88808  0.890431                0.8894  0.889254
```

4 – Sair:

Encerra o programa.

```
----------------------------------------
        DIGITE A OPÇÃO DESEJADA
----------------------------------------
1 - Fazer predição com o modelo já treinado
2 - Treinar novamente o modelo
3 - Treinar o modelo e executar os testes
4 - Sair
----------------------------------------
Sua Opção:4
Encerrando
```

5. Análises e Testes Realizados

Para testes dos modelos criei combinações diferentes de tratamento dos dados variando:

- Tratamento Morfológico: Stemmer e Lematização
- Tipos de Tokens: Unigrams e Unigrams + Bigrams
- Contagem dos tokens: Contagem Simples e TF-IDF

Que resultou em 8 combinações diferentes de treinamento dos dados que foram utilizados nos 7 modelos citados.

Caso queira rodar as avaliações, a partir do diretório source pode executar: **python analyzer1.py**

Segue os resultados do processamento:

5.1. Logs das Combinações Testadas

Combinação 1: Stemming / Unigrams / Contagem Simples

Nro de Tokens Resultantes: **51.229**

Modelos	Acurácia	Sensibil. / Recall	Precisão	Especificid. / BAC	F1-score
Logistic Regression	0.85392	0.84440	0.860789	0.85392	0.852516
SVC	0.86948	0.88448	0.858718	0.86948	0.871409
Linear SVC	0.82852	0.81416	0.838234	0.82852	0.826022
SGD Classifier	0.83716	0.87800	0.811700	0.83716	0.843549
Multinomial Naive Bayes	0.81768	0.76232	0.857233	0.81768	0.806995
Decision Tree Classifier	0.72176	0.72096	0.722115	0.72176	0.721537
Random Forest Classifier	0.84944	0.84528	0.852372	0.84944	0.848811

Combinação 2: Stemming / Unigrams + Bigrams / Contagem Simples

Nro de Tokens Resultantes: **1.577.483**

Modelos	Acurácia	Sensibil. / Recall	Precisão	Especificid. / BAC	F1-score
Logistic Regression	0.87412	0.87232	0.875472	0.87412	0.873893
SVC	0.87064	0.87616	0.866593	0.87064	0.871350
Linear SVC	0.86864	0.86104	0.874330	0.86864	0.867634
SGD Classifier	0.86268	0.84280	0.877697	0.86268	0.859895
Multinomial Naive Bayes	0.82276	0.79520	0.841588	0.82276	0.817737
Decision Tree Classifier	0.74068	0.74104	0.740507	0.74068	0.740773
Random Forest Classifier	0.84224	0.87944	0.818541	0.84224	0.847898

Combinação 3: Stemming / Unigrams / TF-IDF

Nro de Tokens Resultantes: **51.229**

Modelos	Acurácia	Sensibil. / Recall	Precisão	Especificid. / BAC	F1-score
Logistic Regression	0.88000	0.88024	0.879818	0.88000	0.880029
SVC	0.87872	0.87256	0.883444	0.87872	0.877968
Linear SVC	0.86460	0.85272	0.873474	0.86460	0.862972
SGD Classifier	0.87780	0.87504	0.879897	0.87780	0.877462
Multinomial Naive Bayes	0.82196	0.77488	0.855427	0.82196	0.813164
Decision Tree Classifier	0.70992	0.70576	0.711681	0.70992	0.708708
Random Forest Classifier	0.84688	0.83568	0.854828	0.84688	0.845146

Combinação 4: Stemming / Unigrams + Bigrams / TF-IDF

Nro de Tokens Resultantes: **1.577.483**

Modelos	Acurácia	Sensibil. / Recall	Precisão	Especificid. / BAC	F1-score
Logistic Regression	0.87860	0.88480	0.873963	0.87860	0.879348
SVC	0.88544	0.89048	0.881594	0.88544	0.886014
Linear SVC	0.88904	0.88736	0.890352	0.88904	0.888853
SGD Classifier	0.88552	0.89736	0.876602	0.88552	0.886860
Multinomial Naive Bayes	0.85812	0.82968	0.879718	0.85812	0.853967
Decision Tree Classifier	0.71176	0.70432	0.714959	0.71176	0.709599
Random Forest Classifier	0.85044	0.83680	0.860268	0.85044	0.848372

Combinação 5: Lematização / Unigrams / Contagem Simples

Nro de Tokens Resultantes: **67.089**

Modelos	Acurácia	Sensibil. / Recall	Precisão	Especificid. / BAC	F1-score
Logistic Regression	0.85980	0.85008	0.866933	0.85980	0.858424
SVC	0.87016	0.88560	0.859072	0.87016	0.872134
Linear SVC	0.83732	0.82728	0.844232	0.83732	0.835670
SGD Classifier	0.84640	0.83496	0.854511	0.84640	0.844622
Multinomial Naive Bayes	0.82196	0.76608	0.862470	0.82196	0.811422
Decision Tree Classifier	0.72104	0.71352	0.724415	0.72104	0.718926
Random Forest Classifier	0.85184	0.84608	0.855940	0.85184	0.850982

Combinação 6: Lematização / Unigrams + Bigrams / Contagem Simples

Nro de Tokens Resultantes: **1.737.147**

Modelos	Acurácia	Sensibil. / Recall	Precisão	Especificid. / BAC	F1-score
Logistic Regression	0.87592	0.87568	0.876101	0.87592	0.875890
SVC	0.87096	0.87760	0.866098	0.87096	0.871811
Linear SVC	0.87140	0.86448	0.876612	0.87140	0.870504
SGD Classifier	0.86432	0.84968	0.875309	0.86432	0.862304
Multinomial Naive Bayes	0.82676	0.79712	0.847351	0.82676	0.821468
Decision Tree Classifier	0.73480	0.72384	0.740062	0.73480	0.731861
Random Forest Classifier	0.84752	0.87136	0.831704	0.84752	0.851070

Combinação 7: Lematização / Unigrams / TF-IDF

Nro de Tokens Resultantes: **67.089**

Modelos	Acurácia	Sensibil. / Recall	Precisão	Especificid. / BAC	F1-score
Logistic Regression	0.87964	0.87960	0.879670	0.87964	0.879635
SVC	0.88004	0.87456	0.884251	0.88004	0.879379
Linear SVC	0.86620	0.85264	0.876408	0.86620	0.864361
SGD Classifier	0.87872	0.87512	0.881467	0.87872	0.878282
Multinomial Naive Bayes	0.83116	0.78448	0.865261	0.83116	0.822893
Decision Tree Classifier	0.71468	0.71224	0.715733	0.71468	0.713982
Random Forest Classifier	0.84956	0.83768	0.858068	0.84956	0.847751

Combinação 8: Lematização / Unigrams + Bigrams / TF-IDF

Nro de Tokens Resultantes: **1.737.147**

Modelos	Acurácia	Sensibil. / Recall	Precisão	Especificid. / BAC	F1-score
Logistic Regression	0.87884	0.88456	0.874555	0.87884	0.879529
SVC	0.88516	0.89240	0.879662	0.88516	0.885985
Linear SVC	0.88988	0.88920	0.890411	0.88988	0.889805
SGD Classifier	0.88552	0.89472	0.878555	0.88552	0.886564
Multinomial Naive Bayes	0.86128	0.83256	0.883297	0.86128	0.857178
Decision Tree Classifier	0.70028	0.70144	0.699816	0.70028	0.700627
Random Forest Classifier	0.84856	0.82512	0.865704	0.84856	0.844925

5.2. Tuning do Modelo

A tunagem de parâmetros foi aplicada apenas ao modelo que obteve a melhor pontuação no F1-Score, o Linear SVC na combinação 8, variando apenas dois parâmetros:

- C: Este parâmetro define um balanço entre enviesamento (bias) e variância.
- Penalty: Parâmetro que define o tipo de penalização de valores (L1 ou L2) para se evitar overfiting.

Caso queira rodar o tuning, a partir do diretório source pode executar: **python analyzer2.py**

Os valores obtidos utilizando o Randomized Search CV foram:

- C: 1.5671297731744318
- Penalty: L2

6. Conclusão e Considerações

Analisando os dados gerados nos testes das combinações tendo como base de comparação o indicador F1-Score, ao comparar as combinações equivalentes diferenciando o tratamento morfológico (Stemming e Lematização) não foi possível identificar uma melhor estratégia.

Porém comparando os tipos de tokens Unigrams x Unigrams + Bigrams , percebe-se uma melhoria dos scores ao se incluir os Bigrams, mas também um aumento extremamente significativo da quantidade de tokens.

O uso do TF-IDF também se mostrou superior a Contagem Simples.

Quanto aos modelos, os melhores resultados foram se alternando entre o Logistic Regression, o SVC e o Linear SVC; o pior modelo foi o Decision Tree em todos os cenários.

Por fim, o melhor resultado obtido foi:

- Modelo: **Linear SVC**
- Parâmetros do Modelo**: C=1.5671297731744318, penalty = 'l2'**
- Tratamento Morfologico: **Lematização**
- Tipos de Tokens: **Unigrams + Bigrams**
- Contagem dos tokens: **TF-IDF**
- Vocabulário: **1.737.147** tokens
- F1-Score Médio nos testes: **0.889805**

A configuração encontrada teve um score sensivelmente melhor que os outros, porém traz uma preocupação quanto ao tamanho do vocabulário definido podendo causar problemas de performance; uma possibilidade de melhoria é a implementação de uma seleção de features a fim de diminuir o tamanho do vocabulário desde que não haja perda de eficácia.