Lista 13 - IA

Edson Pimenta de Almeida

Junho de 2025

Nota Importante

O notebook Jupyter com todo o código-fonte, experimentações e análises realizadas neste projeto está disponível publicamente e pode ser acessado através do link abaixo:

» Acessar Notebook no Google Colab «

Sumário

Parte I

Questão 01: Classificação de Comentários Tóxicos

1 Introdução

Este relatório detalha o desenvolvimento de um sistema de classificação de texto multirrótulo. O objetivo foi analisar e categorizar comentários da plataforma Jigsaw em seis diferentes classes de toxicidade: `toxic`, `severe_toxic`, `obscene`, `threat`, `insult` e `identity_hate`. Devido à natureza do problema, onde um único comentário pode pertencer a múltiplas categorias simultaneamente, foram empregadas técnicas específicas para classificação multirrótulo.

2 Configuração do Ambiente

Inicialmente, foram importadas todas as bibliotecas necessárias para a análise, pré-processamento, modelagem e avaliação. Isso incluiu `pandas` para manipulação de dados, `nltk` e `re` para limpeza de texto, `scikit-learn` para vetorização e métricas, e `scikit-multilearn` para a construção do modelo de classificação. O ambiente de desenvolvimento foi baseado em Python.

```
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords

# Ferramentas do Scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report, f1_score
from skmultilearn.problem_transform import ClassifierChain

# Ferramentas do LightGBM
from lightgbm import LGBMClassifier

# Downloads de recursos do NLTK
nltk.download('stopwords')
```

Listing 1: Importação das bibliotecas

3 Carregamento e Exploração dos Dados

Os dados de treinamento foram carregados a partir do arquivo `train.csv`. Em seguida, foi realizada uma exploração inicial para entender a estrutura do dataset, o formato dos comentários e a distribuição das classes (rótulos).

```
# Carregando os dados de treinamento
df_train = pd.read_csv('train.csv')
```

```
3
4 # Identificando as colunas de rotulos
5 label_cols = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', '
    identity_hate']
6
7 # Verificando a distribuicao das classes
8 print("\nDistribui  o das Classes:")
9 for col in label_cols:
10     print(f"{col}: {df_train[col].sum()} coment rios")
```

Listing 2: Carregamento e exploração inicial

4 Pré-processamento do Texto

Antes de treinar o modelo, os dados de texto da coluna `comment_text` foram submetidos a um processo de limpeza e normalização, etapa fundamental para a extração de características relevantes. Foi definida uma função para realizar as seguintes transformações:

- **Conversão para Minúsculas:** Padronização de todo o texto.
- **Remoção de Caracteres Especiais:** Exclusão de pontuações, números e outros símbolos não alfabéticos.
- **Remoção de Stopwords:** Exclusão de palavras comuns da língua inglesa que não agregam valor semântico.

Essa função de limpeza foi aplicada a cada comentário no dataset.

```
1 stop_words = set(stopwords.words('english'))
2
3 def clean_text(text):
4     text = text.lower()
5     text = re.sub(r'[^a-z\s]', '', text)
6     text = ' '.join([word for word in text.split() if word not in
    stop_words])
7     return text
8
9 df_train['clean_comment_text'] = df_train['comment_text'].apply(
    clean_text)
```

Listing 3: Função de limpeza e sua aplicação

5 Vetorização de Texto com TF-IDF

Modelos de machine learning não conseguem interpretar texto bruto. Por isso, os comentários limpos foram convertidos em vetores numéricos através da técnica **TF-IDF (Term Frequency-Inverse Document Frequency)**. Este método atribui um peso a cada palavra com base em sua frequência no documento e sua raridade no corpus total. O `TfidfVectorizer` do Scikit-learn foi utilizado para essa transformação, com um limite de 5000 características para otimizar o treinamento.

```
1 X = df_train['clean_comment_text']
2 y = df_train[label_cols]
3
```

```
vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = vectorizer.fit_transform(X)
```

Listing 4: Vetorização TF-IDF

6 Análise do Desbalanceamento e Estratégia de Modelagem

6.1 Análise Visual do Desbalanceamento de Classes

Uma etapa crucial da análise exploratória foi a verificação da distribuição das classes. Um gráfico de barras foi gerado para visualizar o número de ocorrências de cada rótulo, revelando um severo desbalanceamento. Categorias como `toxic` e `obscene` são classes majoritárias, enquanto `threat` e `severe_toxic` são classes minoritárias. Este desbalanceamento foi identificado como a causa principal do baixo desempenho do modelo inicial na detecção dessas classes raras.

6.2 Investigação de Estratégias Multirrótulo

A biblioteca `scikit-multilearn` foi investigada por ser especializada em classificação multirrótulo. Dentre os métodos de transformação de problema, a abordagem **Classifier Chains (Cadeias de Classificadores)** foi selecionada. Este método treina um classificador para cada rótulo de forma encadeada, onde a previsão de um classificador serve como característica para o próximo. Isso permite que o modelo capture dependências entre os rótulos.

7 Treinamento e Seleção do Modelo Final

O processo de modelagem foi iterativo. Com os dados pré-processados, foi realizada a divisão em conjuntos de treino e teste (80/20). Diferentes abordagens de balanceamento e algoritmos foram testados. A estratégia final, que apresentou o melhor equilíbrio de desempenho, consistiu em utilizar o classificador `LGBMClassifier` (LightGBM) combinado com a técnica de ponderação de classes, através do parâmetro `class_weight`.

```
# Divisao dos dados
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y,
    test_size=0.2, random_state=42)

# Definicao do classificador base LightGBM com balanceamento
base_classifier = LGBMClassifier(
    objective='binary',
    n_estimators=150,
    learning_rate=0.1,
    class_weight={0: 1, 1: 2}
)

# Criacao da cadeia de classificadores
classifier_chain = ClassifierChain(base_classifier)

# Treinamento do modelo
classifier_chain.fit(X_train, y_train)
```

```
17
18 # Realizacao das previsoes
19 predictions = classifier_chain.predict(X_test)
```

Listing 5: Treinamento do Modelo Final Selecionado

8 Resultados e Conclusão (Questão 01)

8.1 Resultados da Validação Local

O modelo final (‘LightGBM‘ com ‘class weight=0:1, 1:2‘) alcançou um desempenho superior às outras tentativas na validação local. As métricas-chave foram:

- **F1-Score (Micro):** 0.7185
- **F1-Score (Weighted):** 0.72

O relatório de classificação detalhado é apresentado na Tabela 1.

Tabela 1: Relatório de Classificação do Modelo Final.

Classe	precision	recall	f1-score	support
toxic	0.84	0.68	0.75	3056
severe_toxic	0.42	0.32	0.36	321
obscene	0.83	0.78	0.80	1715
threat	0.37	0.34	0.35	74
insult	0.66	0.72	0.69	1614
identity_hate	0.56	0.33	0.42	294

8.2 Validação Final (Kaggle)

Para a validação final, o modelo foi submetido à competição do Kaggle, obtendo os seguintes resultados no conjunto de dados de teste oficial:

- **Public Score (AUC ROC):** 0.78522
- **Private Score (AUC ROC):** 0.77802

A proximidade entre os scores público e privado demonstrou a **robustez e a boa capacidade de generalização** do modelo, confirmando que não houve superajuste (*overfitting*) e que o processo de validação local foi eficaz.

Parte II

Questão 02: Regras de Inferência

A seguir, são apresentadas as resoluções para os exercícios selecionados da seção "14.7 Regras de Inferência".

1) "Se é feriado, os bancos estão fechados"

Resposta: a) Se os bancos não estão fechados, não é feriado.
Justificativa: A resposta é a contrapositiva da sentença original (Se P $\rightarrow$ Q, então $\sim$Q $\rightarrow$ $\sim$P), que é logicamente equivalente.

4) "Se não chover, Cláudia vai à praia..."

Resposta: e) Fábia foi ao clube.
Justificativa: A premissa "choveu"ativa a segunda regra ("Se chover, Fábia vai ao clube") por *Modus Ponens*, garantindo a conclusão.

5) "Pedro é estudioso e trabalhador, ou Pedro é bonito..."

Resposta: a) Pedro é estudioso e trabalhador.
Justificativa: Como a segunda parte da disjunção ("Pedro é bonito") é falsa, a primeira parte deve ser verdadeira, por *Silogismo Disjuntivo*.

7) "Se Rubens estudar, então passará no concurso."

Resposta: a) Se Rubens não passar no concurso, então não terá estudado.
Justificativa: A afirmação é a contrapositiva da original, mantendo a equivalência lógica.

8) "Se Felipe toca violão, ele canta..."

Resposta: b) Se Felipe toca violão, então ele não toca piano.
Justificativa: Das premissas V $\rightarrow$ C e P $\rightarrow$ $\sim$C (ou C $\rightarrow$ $\sim$P), conclui-se por silogismo hipotético que V $\rightarrow$ $\sim$P.

9) "A proposição p $\rightarrow$ $\sim$q é equivalente a:"

Resposta: e) $\sim$p $\vee$ $\sim$q.
Justificativa: Uma condicional A $\rightarrow$ B é sempre equivalente a $\sim$A $\vee$ B.

13) "Se Carlos é administrador, então é pobre."

Resposta: c) Se Carlos não é pobre, então não é administrador.
Justificativa: Trata-se da contrapositiva da proposição original.

16) "Meu salário cobrirá as despesas somente se eu economizar."

Resposta: e) Se eu não economizar, meu salário não cobrirá as despesas.
Justificativa: A sentença "A somente se B"é A $\rightarrow$ B. A resposta é a contrapositiva ($\sim$B $\rightarrow$ $\sim$A).

17) "Ou Lógica é fácil ou Artur não gosta de Lógica..."

Resposta: b) Lógica é fácil e geografia é difícil.
Justificativa: Se Artur gosta de Lógica, então "Lógica é fácil"deve ser verdade. A partir disso, na segunda premissa, conclui-se que "Geografia é difícil".

19) "Se Iara não fala italiano, então Ana fala alemão..."

Resposta: a) Iara não fala italiano e Débora não fala dinamarquês.
Justificativa: Seguindo a cadeia de implicações a partir dos fatos dados ($\sim$F_fra e $\sim$C_chi), chega-se às conclusões $\sim$D_din e $\sim$I_ita.

20) "Se Carina é amiga de Carol..."

Resposta: b) Carina não é amiga de Carol ou não é cunhada de Carmem.
Justificativa: Das duas primeiras premissas, conclui-se que "Carina não é amiga de Carol". Como esta parte da disjunção é verdadeira, a disjunção inteira é verdadeira.

21) "se x = 7, então y = 9"

Resposta: c) Se y $\neq$ 9, então x $\neq$ 7.
Justificativa: A resposta é a contrapositiva da afirmação original.

23) "Se Júlio estiver certo, então Vítor estará enganado..."

Resposta: b) André não participou da corrida.
Justificativa: A cadeia de *Modus Ponens* a partir de "Júlio estava certo"leva à conclusão de que o carro não ficou pronto, o que, por sua vez, implica que André não participou da corrida.

24) "Se Carla é solteira, então Maria é estudante"

Resposta: c) Se Maria não é estudante, então Carla não é solteira.
Justificativa: A resposta é a contrapositiva da proposição original.

26) "Se x + y = 2, então x = 0. Ora, x não é zero."

Resposta: d) x + y $\neq$ 2.
Justificativa: Aplicação da regra de inferência *Modus Tollens*.

27) "O rei ir à caça é condição necessária para o duque..."

Resposta: c) O rei não foi à caça e o conde não encontrou a princesa.
Justificativa: A partir do fato de que o barão não sorriu, uma cadeia de *Modus Tollens* é acionada, provando ambas as partes da conjunção como verdadeiras.

28) "Se você se esforçar, então irá vencer."

Resposta: a) Seu esforço é condição suficiente para vencer.
Justificativa: Em uma condicional "Se P, então Q", P é, por definição, a condição suficiente para Q.

31) "Se Francisco desviou dinheiro..."

Resposta: e) Alguém não desviou dinheiro da campanha assistencial.
Justificativa: Tentar concluir sobre o delito de Francisco seria a falácia da negação do antecedente. A única conclusão logicamente válida é a generalização existencial da premissa "Francisco não desviou dinheiro".

32) "Se Rodrigo mentiu, então ele é culpado."

Resposta: a) Se Rodrigo não é culpado, então ele não mentiu.
Justificativa: A resposta é a contrapositiva da afirmação original.

Parte III

Questão 03: Lógica Clássica vs. Lógica Fuzzy

Nesta questão, foi explorada a diferença entre a lógica booleana tradicional e a lógica fuzzy para classificar a altura de personagens.

8.3 a) Aplicação da Lógica Clássica

Foi aplicada a regra "Se a altura for maior ou igual a 170 cm, então o personagem é alto (valor 1); caso contrário, não é alto (valor 0)". Os resultados são mostrados na Tabela 2.

Tabela 2: Classificação de altura usando lógica clássica.

Personagem	**Altura (cm)**	**Alto (lógica clássica)**
Ana	148	0
Bruno	165	0
Carla	172	1
Diego	180	1
Elisa	191	1

8.4 b) Análise Crítica da Lógica Clássica

A lógica clássica impõe uma fronteira rígida e irrealista. Não é sensato afirmar que uma pessoa com 169 cm não é "nada alta"(valor 0) e outra com 170 cm é "totalmente alta"(valor 1). Essa transição abrupta não representa bem a natureza gradual de conceitos humanos como "alto".

8.5 c) Proposta de Função de Pertinência Fuzzy

Para superar a limitação da lógica clássica, foi proposta uma função de pertinência fuzzy linear para a categoria "alto":

- Altura < 160 cm $\rightarrow$ Grau de pertinência $= 0$
- 160 cm $\leq$ Altura ≤ 190 cm $\rightarrow$ O grau aumenta linearmente de 0 a 1.
- Altura > 190 cm $\rightarrow$ Grau de pertinência $= 1$

A fórmula para o intervalo linear é:

$$\text{Grau de "alto} = \frac{\text{Altura} - 160}{30}$$

8.6 d) Aplicação da Lógica Fuzzy

Com base na função de pertinência, foram calculados os graus de "alto"para cada personagem, conforme a Tabela 3. Esta abordagem oferece uma representação muito mais nuançada e realista da característica "altura".

Tabela 3: Grau de "alto"usando lógica fuzzy.

Personagem	**Altura (cm)**	**Grau de "alto"(fuzzy)**
Ana	148	0.000
Bruno	165	0.167
Carla	172	0.400
Diego	180	0.667
Elisa	191	1.000