

---

# Análise de sentimento de textos de avaliação

— aplicação para avaliação real time —  
para plataforma de atendimento

Alunos: Heinrych e Luan Webá

---

# Dataset



FRED DIAS · UPDATED 2 YEARS AGO



19

New Notebook

Download (920 MB)



## Brazilian Portuguese Sentiment Analysis Datasets



Data Card

Code (1)

Discussion (0)

### About Dataset

#### Context

Despite the recent advances in NLP, the Brazilian Portuguese language still has limited linguistic resources. Due to the scarcity of text classification datasets in Brazilian Portuguese and works comparing the performance of machine learning models in benchmarks with pre-defined partitions, we selected five sentiment analysis datasets and proposed two target features (polarity and rating) and partitions for each label.

We hope to encourage other Brazilian researchers to evaluate their predictive models in the same partitions to facilitate replicability and comparison between results.

Usability

4.71

License

Unknown

Expected update frequency

Not specified

Tags

Text

NLP

Brazil

utlc\_apps.csv (285.13 MB)



Detail Compact Column

8 of 8 columns

# original_index	review_text	review_text_proc...	review_text_toke...	# polarity	
	review_text	review_text_proc...	review_text_toke...	# polarity	
1.84m	2.88m				
2483729	eu curtindo muito	eu curtindo muito	['eu', 'curtindo', 'muito']	1.0	5
2865805	Aplicativo absurdamente caro e o valor para se tornar prêmio é ainda muito mais caro. Joguei dinheir...	aplicativo absurdamente caro e o valor para se tornar premio e ainda muito mais caro. joguei dinheir...	['aplicativo', 'absurdamente', 'caro', 'valor', 'para', 'se', 'tornar', 'premio', 'ainda', 'muito', 'caro', 'joguei', 'dinheir...']	0.0	1
2734161	Não é mais tão simples com o novo layout	nao e mais tao simples com o novo layout	['nao', 'mais', 'tao', 'simples', 'com', 'novo', 'layout']	1.0	4
2066362	Estava com um problema para acessar, porém consegui resolver pelo fone fácil. Ótimo app	estava com um problema para acessar, porem consegui resolver pelo fone facil. otimo app	['estava', 'com', 'um', 'problema', 'para', 'acessar', 'porem', 'consegui', 'resolver', 'pelo', 'fon...']	1.0	5
2521806	Eu gostei muito do	eu gostei muito do	['eu', 'gostei',	1.0	5

Aplicativo absurdamente caro e o valor para se tornar prêmio é ainda muito mais caro. Joguei dinheir...	aplicativo absurdamente caro e o valor para se tornar premio e ainda muito mais caro. joguei dinheir...	['aplicativo', 'absurdamente', 'caro', 'valor', 'para', 'se', 'tornar', 'premio', 'ainda', 'muito', 'caro', 'joguei', 'dinheir...']	0.0	1.0
---	---	---	-----	-----

# Pré-processamento

<b><i>Pré Processamento</i></b>
<b><i>Remoção de #, @, URL</i></b>
<b><i>Remoção de stop words</i></b>
<b><i>Conversão de palavras para caixa baixa</i></b>
<b><i>Tokenização</i></b>
<b><i>Marcação de fala POS</i></b>
<b><i>Stemming</i></b>
<b><i>Normalização das palavras</i></b>

## ✓ sample dos dados

```
num_samples_per_class = 50000 # reshape dos dados, grande dms

df_zero = data[data.polarity == 0.0].sample(num_samples_per_class)
df_one = data[data.polarity == 1.0].sample(num_samples_per_class)

data = pd.concat([df_zero, df_one])

print(data.groupby(['polarity']).size())
```

[4]

```
... polarity
0.0      50000
1.0      50000
dtype: int64
```

```
PRE_TRAINED_MODEL_NAME = 'neuralmind/bert-base-portuguese-cased'  
  
tokenizer = BertTokenizer.from_pretrained(PRE_TRAINED_MODEL_NAME)
```

```
token_lens = []  
  
for txt in data.frase:  
    tokens = tokenizer.encode_plus(txt,  
                                   max_length=512,  
                                   truncation=True,  
                                   padding='max_length')  
    token_lens.append(len(tokens["input_ids"]))
```

```

class GPReviewDataset(Dataset):

    def __init__(self, reviews, targets, tokenizer, max_len):
        self.reviews = reviews
        self.targets = targets
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.reviews)

    def __getitem__(self, item):
        review = str(self.reviews[item])
        target = self.targets[item]

        encoding = self.tokenizer.encode_plus(
            review,
            add_special_tokens=True,
            max_length=self.max_len,
            truncation=True, # Adicione esta linha se necessário
            return_token_type_ids=False,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='pt',
        )

        return {
            'frase': review,
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'targets': torch.tensor(target, dtype=torch.long)
        }

```

```
from sklearn.model_selection import train_test_split

data_train, data_temp = train_test_split(data,
                                         test_size=0.2,
                                         stratify=data['polarity'],
                                         random_state=RANDOM_SEED)

data_val, data_test = train_test_split(data_temp,
                                         test_size=0.5,
                                         stratify=data_temp['polarity'],
                                         random_state=RANDOM_SEED)
```

```
import multiprocessing

num_cores = multiprocessing.cpu_count()
num_workers = min(num_cores, 0)
print(f'Number of available CPU cores: {num_workers}')

def create_data_loader(data, tokenizer, max_len, batch_size):
    ds = GPReviewDataset(
        reviews=data.frase.to_numpy(),
        targets=data.polarity.to_numpy(),
        tokenizer=tokenizer,
        max_len=max_len
    )

    return DataLoader(
        ds,
        batch_size=batch_size,
        num_workers=num_workers
    )
```



# Gerando as bases para a realização do treinamento

```
BATCH_SIZE = 16
MAX_LEN = 160
train_data_loader = create_data_loader(data_train, tokenizer, MAX_LEN, BATCH_SIZE)
val_data_loader = create_data_loader(data_val, tokenizer, MAX_LEN, BATCH_SIZE)
test_data_loader = create_data_loader(data_test, tokenizer, MAX_LEN, BATCH_SIZE)

data_valid2 = pd.read_csv('../lexicos/base_validacao.csv')
data_valid2 = data_valid2.dropna(subset=['polarity', 'frase'])
valid_data_loader2 = create_data_loader(data_valid2, tokenizer, MAX_LEN, BATCH_SIZE)

data_valid = pd.read_csv('../lexicos/base_validacao2.csv')
data_valid = data_valid.dropna(subset=['polarity', 'frase'])
valid_data_loader = create_data_loader(data_valid, tokenizer, MAX_LEN, BATCH_SIZE)
```

```
dataf = next(iter(train_data_loader))  
dataf.keys()
```

```
dict_keys(['frase', 'input_ids', 'attention_mask', 'targets'])
```

```
print(dataf['input_ids'].shape)  
print(dataf['attention_mask'].shape)  
print(dataf['targets'].shape)
```

```
torch.Size([16, 160])  
torch.Size([16, 160])  
torch.Size([16])
```

## Mascaramento

O sensor de **X1** veio **X2** baterias!

BERT

X1 = umidade

X2 = sem

```
import torch.nn as nn
class SentimentClassifier(nn.Module):

    def __init__(self, n_classes):
        super(SentimentClassifier, self).__init__()
        self.bert = BertModel.from_pretrained(PRE_TRAINED_MODEL_NAME, return_dict=False)
        self.drop = nn.Dropout(p=0.3)
        self.out = nn.Linear(self.bert.config.hidden_size, n_classes)

    def forward(self, input_ids, attention_mask):
        _, pooled_output = self.bert(
            input_ids=input_ids,
            attention_mask=attention_mask
        )
        output = self.drop(pooled_output)
        return self.out(output)
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
model = SentimentClassifier(2)
model = model.to(device)
```

# Treinamento do modelo

# Definição do modelo

```
EPOCHS = 10

optimizer = AdamW(model.parameters(), lr=3e-5, correct_bias=False)
total_steps = len(train_data_loader) * EPOCHS

scheduler = get_linear_schedule_with_warmup(
    optimizer,
    num_warmup_steps=0,
    num_training_steps=total_steps
)

loss_fn = nn.CrossEntropyLoss().to(device)
```

```

def train_epoch(
    model,
    data_loader,
    loss_fn,
    optimizer,
    device,
    scheduler,
    n_examples
):
    model = model.train()

    losses = []
    correct_predictions = 0

    # Aqui estamos usando tqdm para envolver o data_loader e mostrar uma barra de progresso
    for d in tqdm(data_loader, desc='Training'):
        input_ids = d["input_ids"].to(device)
        attention_mask = d["attention_mask"].to(device)
        targets = d["targets"].to(device)

        outputs = model(
            input_ids=input_ids,
            attention_mask=attention_mask
        )

        _, preds = torch.max(outputs, dim=1)
        loss = loss_fn(outputs, targets)

        correct_predictions += torch.sum(preds == targets)
        losses.append(loss.item())

        loss.backward()
        nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
        optimizer.step()
        scheduler.step()
        optimizer.zero_grad()

    return correct_predictions.double() / n_examples, np.mean(losses)

```

```

def eval_model(model, data_loader, loss_fn, device, n_examples):
    model = model.eval()

    losses = []
    correct_predictions = 0

    with torch.no_grad():
        for d in data_loader:
            input_ids = d["input_ids"].to(device)
            attention_mask = d["attention_mask"].to(device)
            targets = d["targets"].to(device)

            outputs = model(
                input_ids=input_ids,
                attention_mask=attention_mask
            )
            _, preds = torch.max(outputs, dim=1)

            loss = loss_fn(outputs, targets)

            correct_predictions += torch.sum(preds == targets)
            losses.append(loss.item())

    return correct_predictions.double() / n_examples, np.mean(losses)

```

```

%%time
from collections import defaultdict
history = defaultdict(list)
best_accuracy = 0

for epoch in range(EPOCHS):

    print(f'Epoch {epoch + 1}/{EPOCHS}')
    print('-' * 10)

    train_acc, train_loss = train_epoch(
        model,
        train_data_loader,
        loss_fn,
        optimizer,
        device,
        scheduler,
        len(data_train)
    )

    print(f'Train loss {train_loss} accuracy {train_acc}')

    val_acc, val_loss = eval_model(
        model,
        val_data_loader,
        loss_fn,
        device,
        len(data_val)
    )

```



Epoch 1/10

-----

Não foi possível renderizar o conteúdo para "application/vnd.jupyter.widget-view+json"  
{"model\_id":"503250d916fd4d2793241942cc9712cc","version\_major":2,"version\_minor":0}

Train loss 0.30852522151991724 accuracy 0.8846375000000001

Val loss 0.2662830392867327 accuracy 0.903

Valid loss 0.84 accuracy 0.5752398371696472

Valid 2 loss 0.7567567567567568 accuracy 0.6257485662187848

Test loss 0.26732184083759786 accuracy 0.8977

Epoch 2/10

-----

Não foi possível renderizar o conteúdo para "application/vnd.jupyter.widget-view+json"  
{"model\_id":"83a7afd0a1f347cba593a21fe4ef976e","version\_major":2,"version\_minor":0}

Train loss 0.2597990558188409 accuracy 0.912425

Val loss 0.2882350666642189 accuracy 0.9054000000000001

Valid loss 0.84 accuracy 0.48558540642261505

Valid 2 loss 0.7207207207207207 accuracy 0.7323708363941738

Test loss 0.3059738239437342 accuracy 0.8977

Epoch 3/10

# Ajuste fino do modelo

```

def get_predictions(model, data_loader):
    model = model.eval()

    review_texts = []
    predictions = []
    prediction_probs = []
    real_values = []

    with torch.no_grad():
        for d in data_loader:

            texts = d["frase"]
            input_ids = d["input_ids"].to(device)
            attention_mask = d["attention_mask"].to(device)
            targets = d["targets"].to(device)

            outputs = model(
                input_ids=input_ids,
                attention_mask=attention_mask
            )
            _, preds = torch.max(outputs, dim=1)

            probs = F.softmax(outputs, dim=1)

            review_texts.extend(texts)
            predictions.extend(preds)
            prediction_probs.extend(probs)
            real_values.extend(targets)

    predictions = torch.stack(predictions).cpu()
    prediction_probs = torch.stack(prediction_probs).cpu()
    real_values = torch.stack(real_values).cpu()
    return review_texts, predictions, prediction_probs, real_values

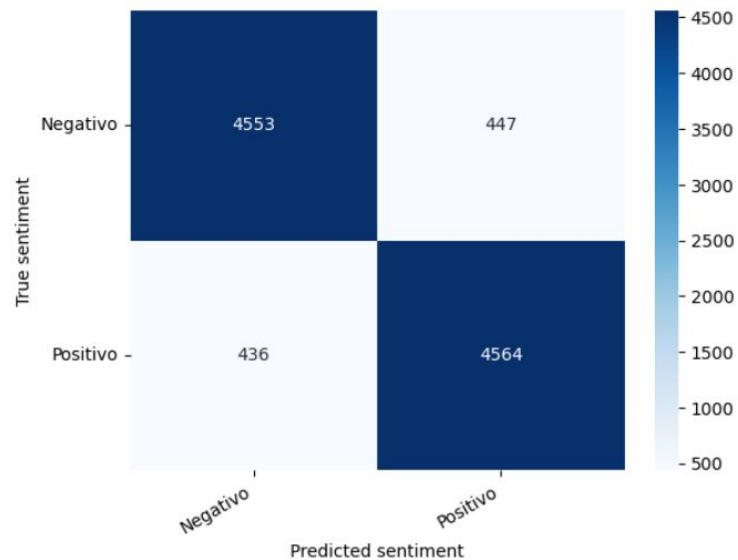
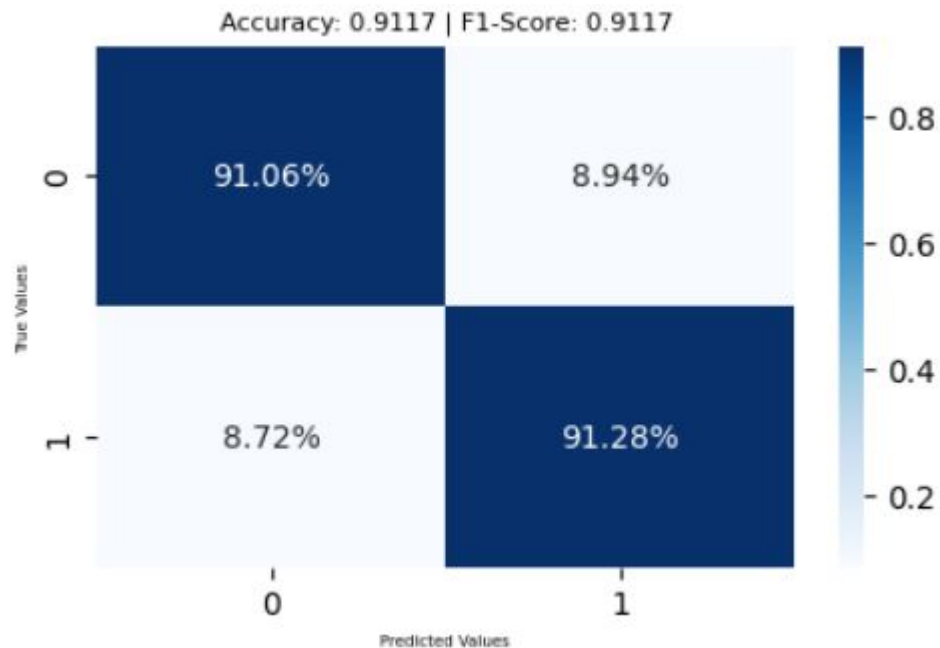
```

```

y_review_texts, y_pred, y_pred_probs, y_test = get_predictions(
    model,
    test_data_loader
)

```

```
y_review_texts, y_pred, y_pred_probs, y_test = get_predictions(  
    model,  
    test_data_loader  
)
```



# Implantação do modelo

**Aplicação do .model a base de mensagens e tela de contatos**

```
# Certifique-se de que o modelo está inicializado
model = SentimentClassifier(len(class_names))
model = model.to(device)

# Carregando o modelo
model.load_state_dict(torch.load(r'D:\Users\heindr\Desktop\TCC-Adriano\classsificação_bert\best_model_state.bin'))
```

```
import psycopg2
conn = psycopg2.connect(
    host="localhost",
    port=5432,
    user="postgres",
    password="aluno",
    database="projeto_inova"
)

cursor = conn.cursor()
```

```
import psycopg2

conn = psycopg2.connect(
    host="localhost",
    port=5432,
    user="postgres",
    password="aluno",
    database="projeto_inova"
)
cur = conn.cursor()

cur.execute("""
    CREATE TABLE IF NOT EXISTS messages (
        id SERIAL PRIMARY KEY,
        contact_id INT NOT NULL,
        body TEXT,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    );
""")

conn.commit()

cur.close()
conn.close()
```

```
import psycopg2
conn = psycopg2.connect(
    host="localhost",
    port=5432,
    user="postgres",
    password="aluno",
    database="projeto_inova"
)

cursor = conn.cursor()
```

```
import psycopg2

conn = psycopg2.connect(
    host="localhost",
    port=5432,
    user="postgres",
    password="aluno",
    database="projeto_inova"
)
cur = conn.cursor()

cur.execute("""
    CREATE TABLE IF NOT EXISTS contacts (
        contact_id SERIAL PRIMARY KEY,
        phone VARCHAR(15),
        status INT DEFAULT 1
    );
""")

conn.commit()

cur.close()
conn.close()
```



# Limpendo o Status da contacts

```
import pandas as pd
import psycopg2

conn = psycopg2.connect(
    host="localhost",
    port=5432,
    user="postgres",
    password="aluno",
    database="projeto_inova"
)

query = "SELECT * FROM contacts"
df = pd.read_sql(query, conn)

conn.close()

print(df)
```

	contact_id	phone	status
0	34507907	62998223865	1
1	32870421	6299385566	1
2	20671831	6299385567	1

```
import psycopg2

BATCH_SIZE = 16
MAX_LEN = 160

conn = psycopg2.connect(
    host="localhost",
    port=5432,
    user="postgres",
    password="aluno",
    database="projeto_inova"
)
cur = conn.cursor()

cur.execute("SELECT DISTINCT contact_id FROM contacts;")
contact_ids = [row[0] for row in cur.fetchall()]
```

```

for contact_id in contact_ids:
    i += 1
    query = """
        SELECT body FROM messages
        WHERE contact_id = %s
        ORDER BY created_at DESC
        LIMIT 10;
    """

    cur.execute(query, (contact_id,))
    df = pd.read_sql(query, conn, params=(contact_id,))
    df.rename(columns={'body': 'review_text'}, inplace=True)
    df = codigos.preprocess.preprocess_text_stop(df)
    messages = [row[0] for row in cur.fetchall()]

    valid_data_loader = create_data_loader2(df, tokenizer, MAX_LEN, BATCH_SIZE)

    y_review_texts_val, y_pred_val, y_pred_probs_val, y_val = get_predictions(model, valid_data_loader)

    df['y_pred'] = y_pred_val
    pd.set_option('display.max_colwidth', None)
    print(df[['frase', 'y_pred']])
    match i:
        case 1:
            a = df
        case 2:
            b = df
        case 3:
            c = df

    if any(pred == 0 for pred in y_pred_val):
        cur.execute("UPDATE contacts SET status = 0 WHERE contact_id = %s;", (contact_id,))
    else:
        cur.execute("UPDATE contacts SET status = 1 WHERE contact_id = %s;", (contact_id,))

# Confirme as alterações
conn.commit()

```

```

import pandas as pd
import psycopg2

# Conectando ao banco de dados
conn = psycopg2.connect(
    host="localhost",
    port=5432,
    user="postgres",
    password="aluno",
    database="projeto_inova"
)
cur = conn.cursor()

# Lendo os dados da tabela 'messages' usando Pandas
cur.execute("UPDATE contacts SET status = 1;")
conn.commit()
# Feche a conexão
conn.close()

# Agora 'df' contém os dados da tabela 'messages'

```

	contact_id	phone	status
0	34507907	62998223865	1
1	32870421	6299385566	1
2	20671831	6299385567	1

contact\_id  
0

	contact_id	phone	status
0	32870421	6299385566	0
1	20671831	6299385567	0
2	35924643	62998223865	1

	review_text	y_pred
0	Este contato foi editado em 07/02/2023	1
1	Certo! Vou preparar o seu carrinho de compras.	1
3	*Vitor diz:* Não responda minha mensagem.	0
4	*Vitor diz:* Tem dinheiro para comprar minhas cadeiras	0
5	*Vitor diz:* Oi	1
6	Contato redirecionado para Vitor pelo sistema em 07/02/2023 às 14:55	0
7		1
8	Olá, tudo bem Bem-vindo ao nosso atendimento pelo WhatsApp. Para melhor direcioná-lo, digite o número do departamento que deseja falar. 1 - Comercial 2 - Suporte 3 - Financeiro	1

contact\_id  
1

	contact_id	phone	status
0	32870421	6299385566	0
1	20671831	6299385567	0
2	35924643	62998223865	1

		review_text	y_pred
0		*Juscelino diz:* Grato pelo resposta, qualquer situação, pode nos acionar. Tenha um excelente dia!!!	0
2		Hoje pela manhã está funcionando perfeitamente, obg	1
4		*Juscelino diz:* Bom dia, Leandro. Tudo bem Tivemos um retorno positivo do nosso time técnico em relação a estabilidade da plataforma para hoje. Pode me contar como está a usabilidade hoje para você, por gentileza	1
6		*Luciana Rodrigues diz:* Olá Leandro, boa tarde! Me chamo Luciana faço parte do Time de Sucesso do Cliente na Poli! Primeiramente gostaria de pedir desculpas em nome da nossa empresa e reforçar com você que já identificamos a causa da instabilidade que afetou o dia de hoje e o nosso time técnico já está operando para retomar a normalização dos serviços. Lamentamos muito o ocorrido e reforçamos que estamos totalmente dedicados em solucionar por completo essa questão. Em breve estaremos operando dentro da normalidade. Qualquer dúvida, estamos aqui para poder te auxiliar!	0
8		NaN	0

contact\_id  
2

	contact_id	phone	status
0	32870421	6299385566	0
1	20671831	6299385567	0
2	35924643	62998223865	1

	review_text	y_pred
0	{"body":"Bom dia Nando , tudo bem com você Estou entrando em contato para confirmar nossa reunião prevista para o dia de hoje. Posso confirmar", "header":{"type":"none","text":"","mediaUrl":"","footer":"","buttons":[],"category":1,"language":46,"name":"polichat_quick_message_359030"}	1
1	Este contato foi editado em 25/05/2023	1
2	*Kennedy Moreira diz:* Maravilha Nando, te encaminhei o invite	1
3	ok	1
4	*Kennedy Moreira diz:* Vou te enviar o invite da nossa reunião	1
5	*Kennedy Moreira diz:* Boa tarde Nando	1
6	Contato redirecionado para Kennedy Moreira por Renato Fonseca em 22/05/2023 às 15:31	1
7	*Renato Fonseca diz:* Um momento	1
8	Preciso falar com o Kenedy	1
9	Souza e Melo Consultoria Previdenciária	1





EM ATENDIMENTO TODOS NÃO LIDOS



Deise Silva



Jerome Bueno

Posso realizar o pagamento por...

João Paulo

Há 30 minutos



Olá, gostaria de saber o valor do KG do bolo de chocolate com creme de avelã

10:25



Deise Silva

Deise Silva diz: Olá, gostaria de ...

Maria Clara

Há 30 minutos



Bruno Adorno

Bom dia, gostaria de saber quant...

Joyzi

Há 30 minutos



Viviane Ribeiro

Adorei o atendimento, foi nota 10...

Geovana

Há 30 minutos



Maria Joana

Jhonatan diz: A senhora necessita de m...

Jhonatan

Há 30 minutos



1

10:27

Olá! Bem-vindo ao atendimento au  
selecione a opção abaixo que atende  
necessidades:

- 1- Falar com atendente
- 2- Horário de Funcionamento
- 3- Entregas

Contato redirecionado para Maria Clara

Tr



# LOONEY TUNES



*"That's all Folks!"*