



Classificação de Reviews Olist:

Comparação de Modelos Redes Neurais

Grupo: Andre Hugo, Edgard Henrique, Gustavo Henrique



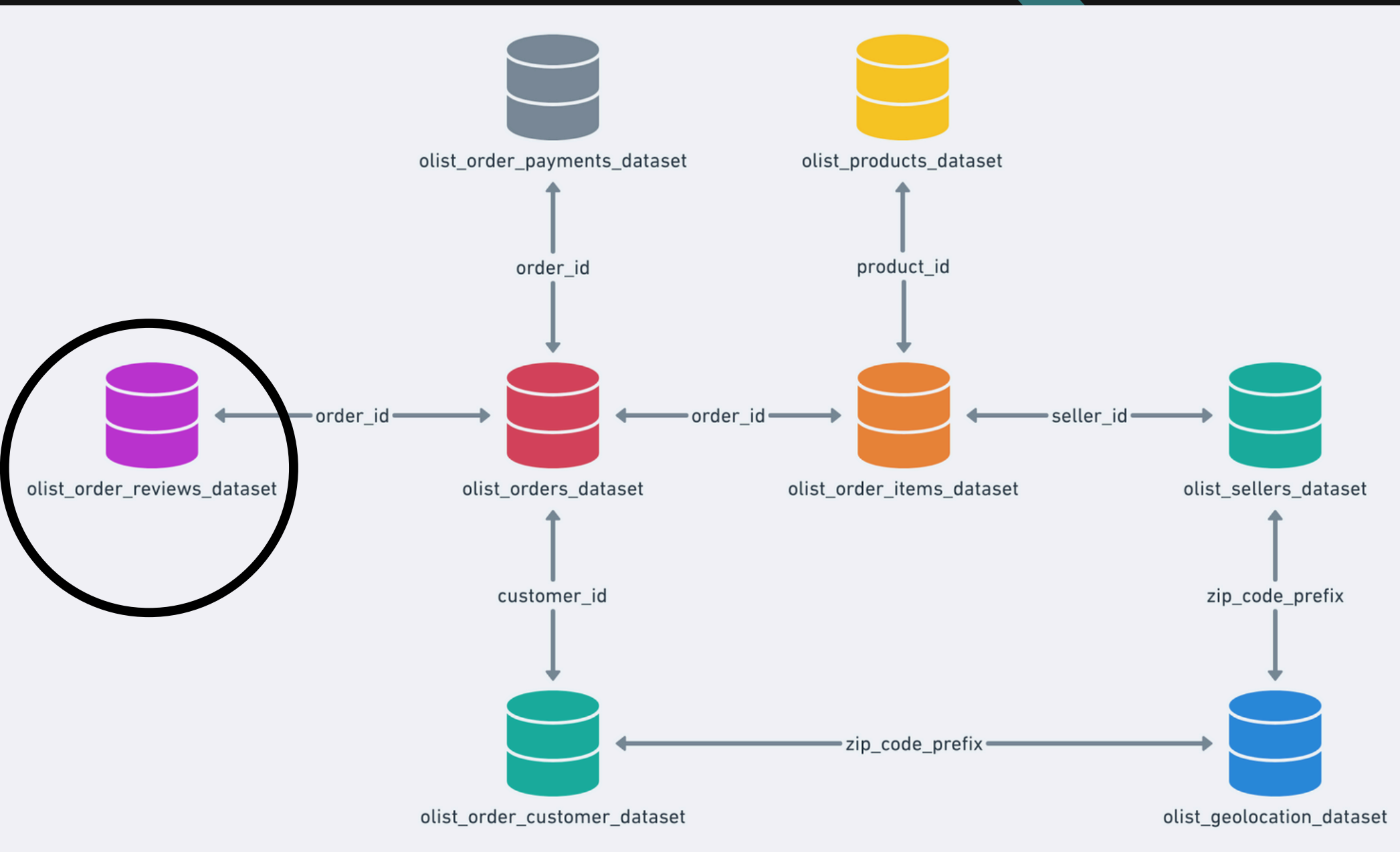
Dataset

Dataset: Brazilian E-Commerce Public
Dataset by Olist

- O conjunto de dados têm informações de 100 mil pedidos de 2016 a 2018 feitos em vários marketplaces no Brasil;
- Múltiplas dimensões: desde o status do pedido, preço, pagamento e entrega até a localização do cliente, atributos do produto e, finalmente, comentários escritos pelos clientes;
- Ideias de Análise: NLP, Clustering, Previsão de vendas, Feature Engineering, etc.

Dataset: Brazilian E-Commerce Public Dataset by Olist

- Banco: olist_order_reviews_dataset;
- Contém reviews de pedidos feitos nos mais diversos marketplaces do Brasil;
- As duas principais informações retiradas do dataset foram a avaliação do pedido e o comentário deixado para esta avaliação.





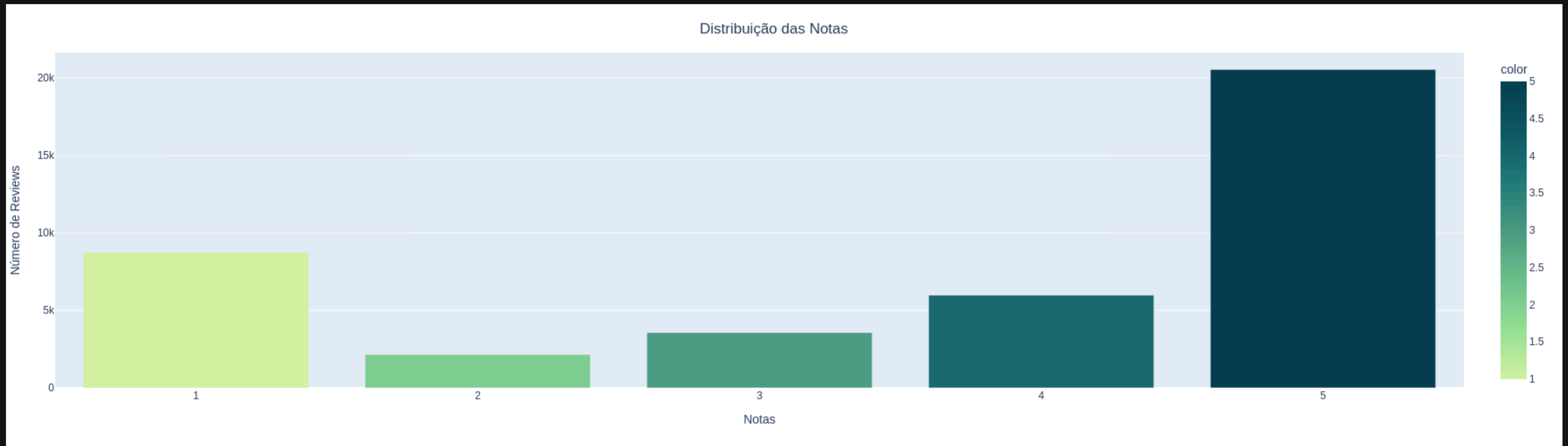
Dataset

Dataset: Brazilian E-Commerce Public Dataset by Olist

- O Dataset será utilizado para se fazer classificação de Reviews (Positivas ou Negativas);
- A abordagem principal será a de processamento de linguagem natural;
- Notas de avaliações de 1 a 5 estrelas serão interpretadas como 0 (reviews negativas) e 1 (reviews positivas).

Vizualizações

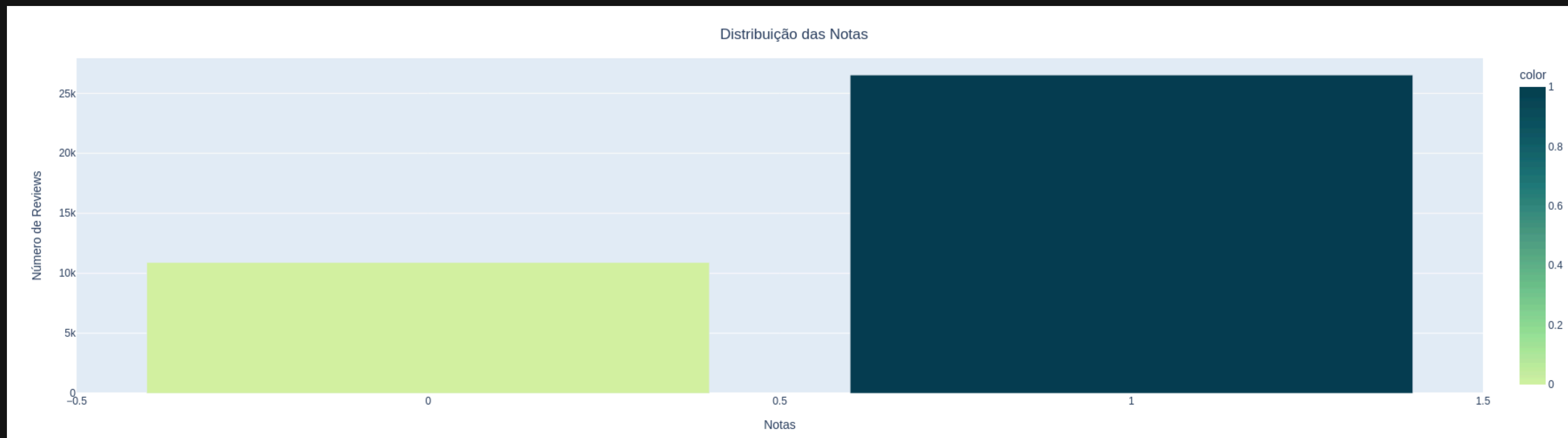
Distribuição de notas de avaliação



Vizualizações

Distribuição de avaliações negativas e positivas

Avaliações: 1 e 2 -> Negativas;
Avaliações: 4 e 5 -> Positivas.



Vizualizações

Principais palavras positivas e negativas

Avaliações: 1 e 2 -> Negativas;

Avaliações: 4 e 5 -> Positivas.



Reviews Negativas





Pré Processamento

Importancia no pré processamento dos dados

- Melhor Desempenho do Modelo
- Facilita o Ajuste ao Domínio Específico do problema;
- Aumento da Confiabilidade dos Resultados;



Pré Processamento

Pipeline de Pré Processamento de dados

- Remoção de avaliações nulas: comentários e notas;
- Remoção de Ruído: Regex para retirar ruídos do texto
- Redução de dimensionalidade: Remoção de Stopwords;
- Normalização de Dados: Remoção de caracteres repetidos, texto em lowercase, etc;
- Captação de Padrões Relevantes e suas variantes: Stemming ou lematização
- Transformação de Textos em Representações Numéricas: Tokenização e word embeddings;



Abordagem 1

- Embedding Próprio
- Modelo de Deep Neural Networks (DNN)

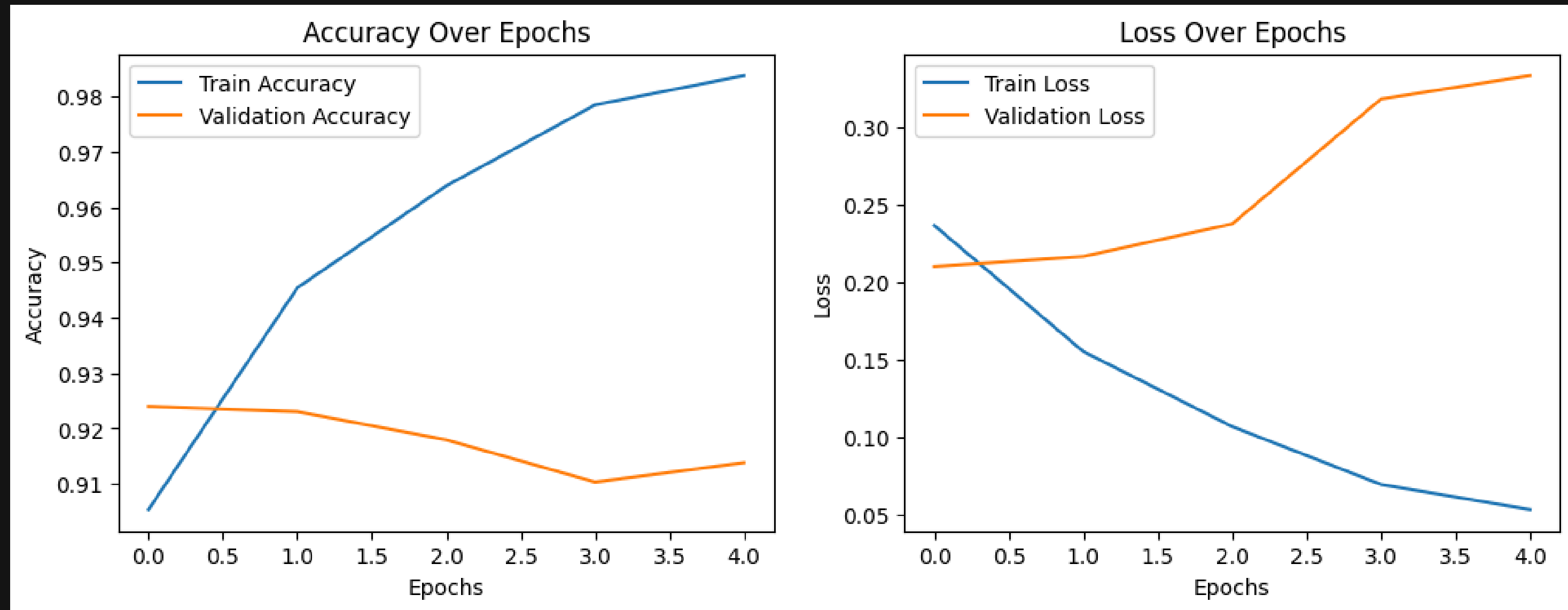
Arquitetura DNN

- Embedding:
 - Tamanho do Vocabulário: 5000;
 - Dimensionalidade do Embedding: 100;
- Flatten;
- Camadas Densas:
 - 64 neurônios, ativação ReLU;
 - 32 neurônios, ativação ReLU;
 - 16 neurônios, ativação ReLU;
 - 1 neurônios, ativação Sigmoid (Camada de Saída);

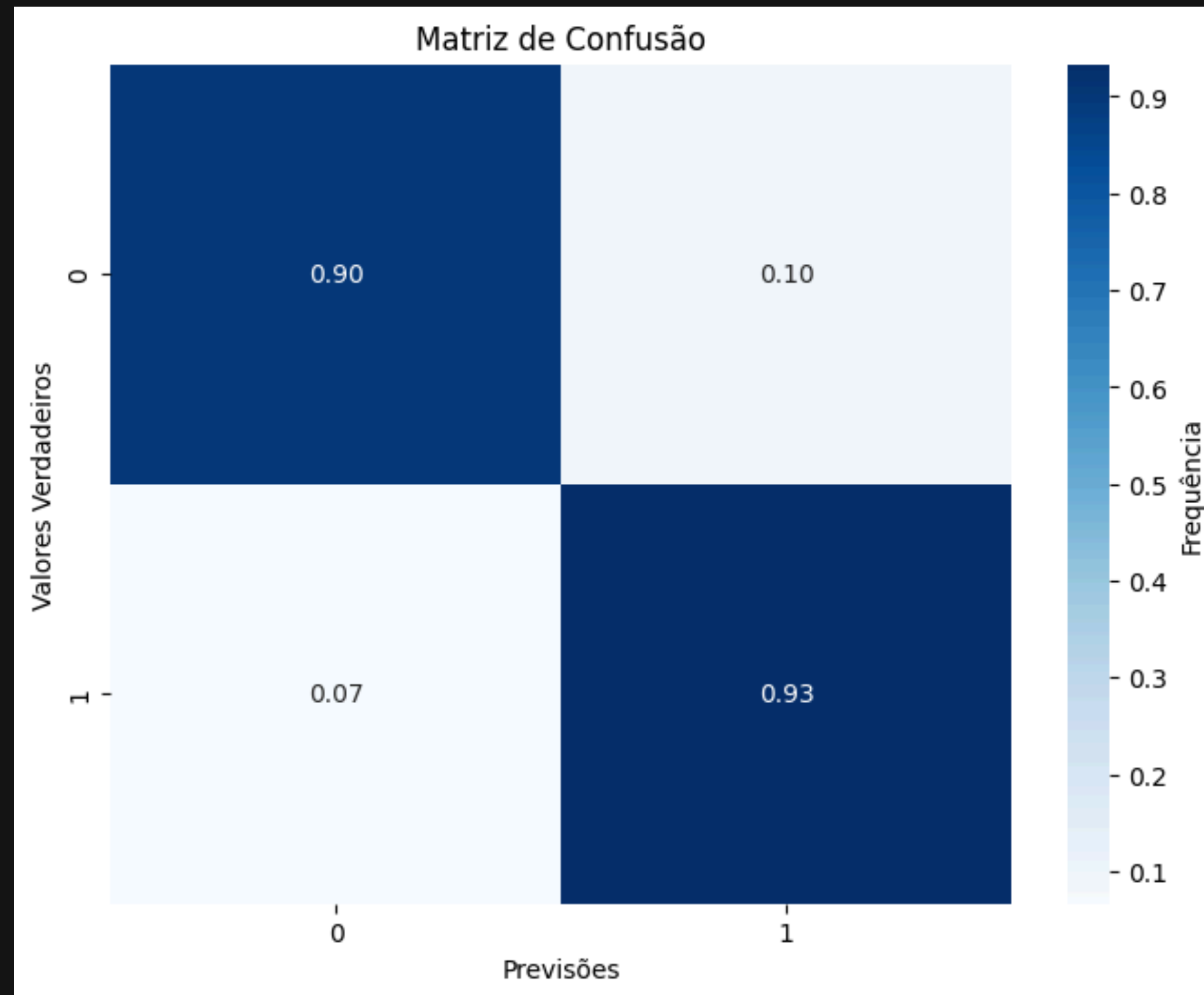
Treinamento

- 8 épocas
- Batch Size: 32;
- Otimizador: Adam (learning rate = 0.001);
- Função de Perda: Binary Crossentropy;
- Early Stopping:
 - Monitoramento de val_loss (perda na validação);
 - Paciencia: 4 épocas sem melhora;
- Recuperação de Pesos:
 - Melhor modelo recuperado a partir da época em que o menor val_loss foi observado.

Resultados



Resultados





Abordagem 2

- Embedding Próprio
- Modelo de Convolutional Neural Networks (CNN)

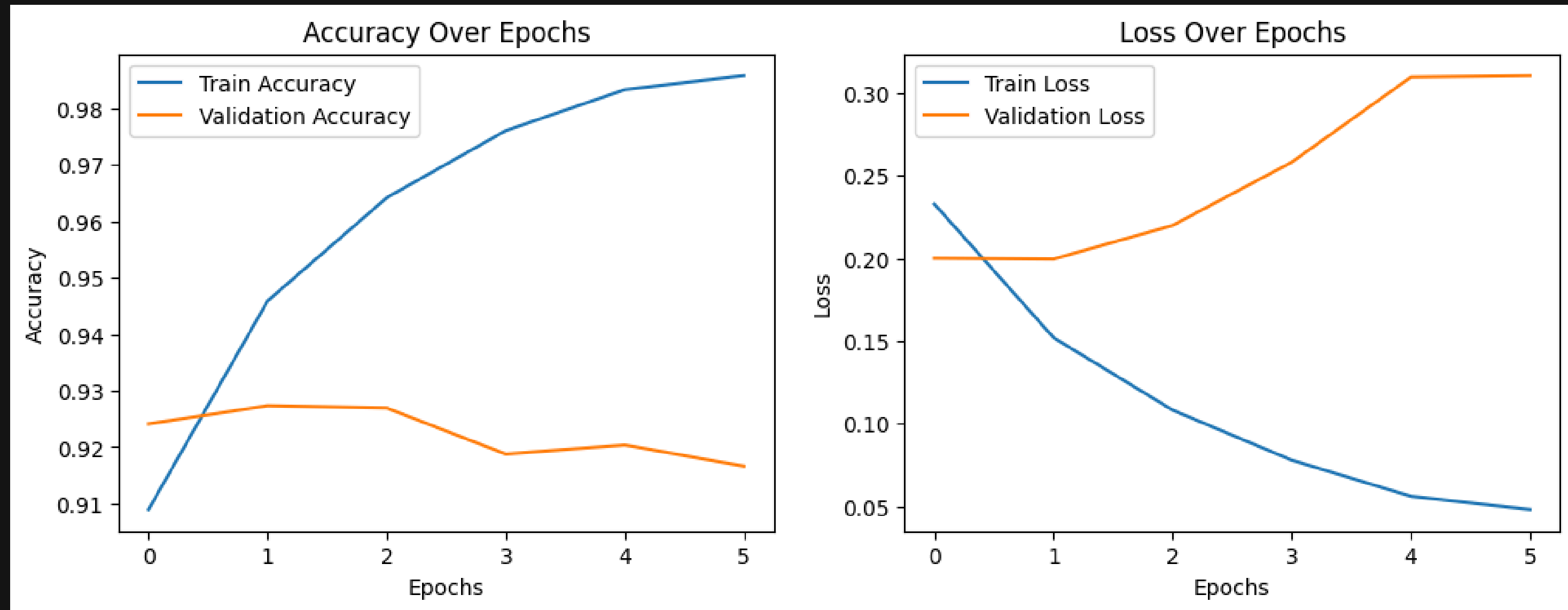
Arquitetura CNN

- Embedding:
 - Tamanho do Vocabulário: 5000;
 - Dimensionalidade do Embedding: 100;
- 2 Camadas Conv1D + MaxPooling:
 - Conv1D: 128 e 64 filtros, kernel size 5 e 3, ativação ReLU.
 - MaxPooling1D: Pooling de tamanho 2.
- Flatten;
- Camadas Densas:
 - 64 neurônios, ativação ReLU;
 - 32 neurônios, ativação ReLU;
 - 1 neurônios, ativação Sigmoid (Camada de Saída);

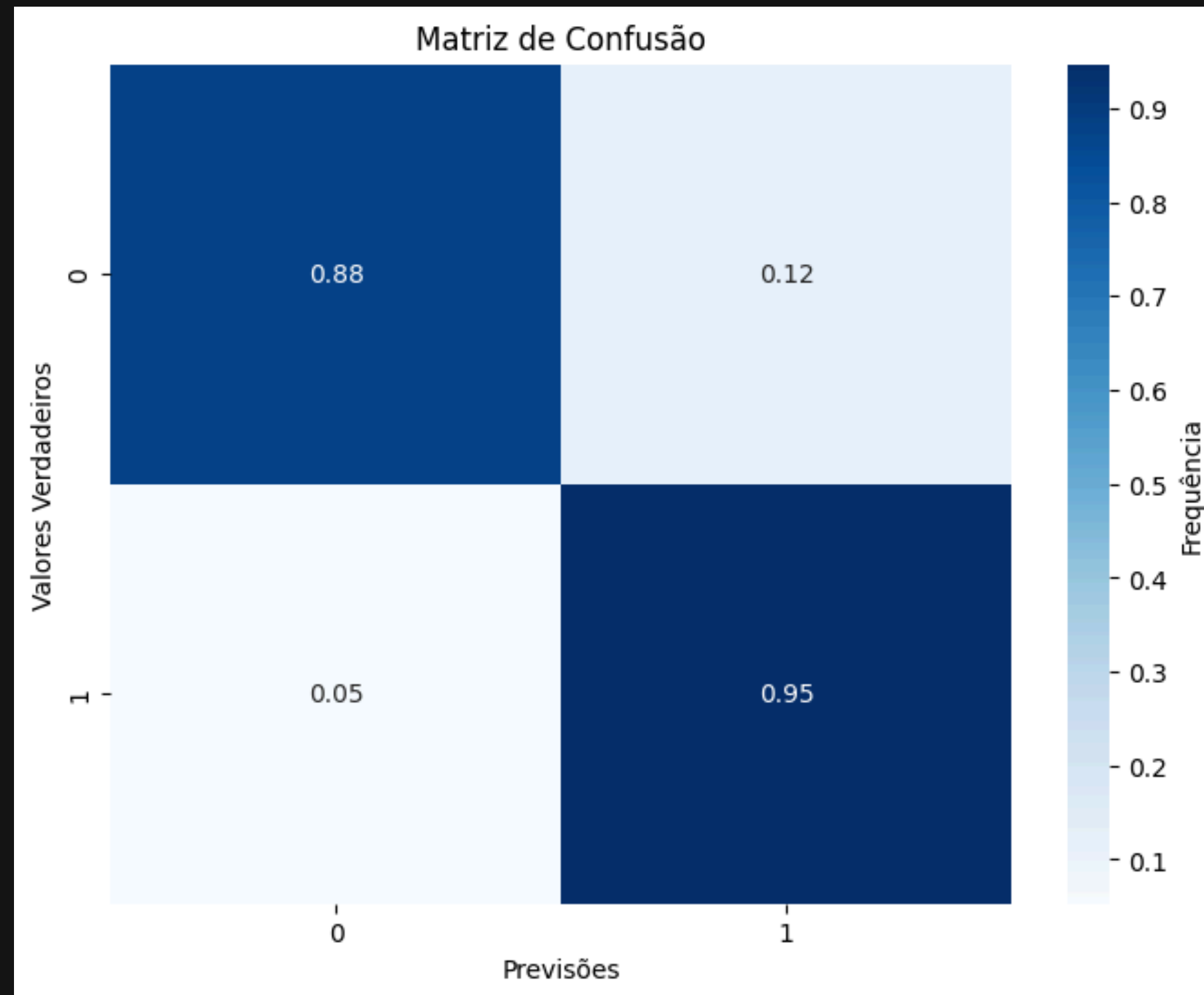
Treinamento

- 8 épocas
- Batch Size: 32;
- Otimizador: Adam (learning rate = 0.001);
- Função de Perda: Binary Crossentropy;
- Early Stopping:
 - Monitoramento de val_loss (perda na validação);
 - Paciencia: 4 épocas sem melhora;
- Recuperação de Pesos:
 - Melhor modelo recuperado a partir da época em que o menor val_loss foi observado.

Resultados



Resultados



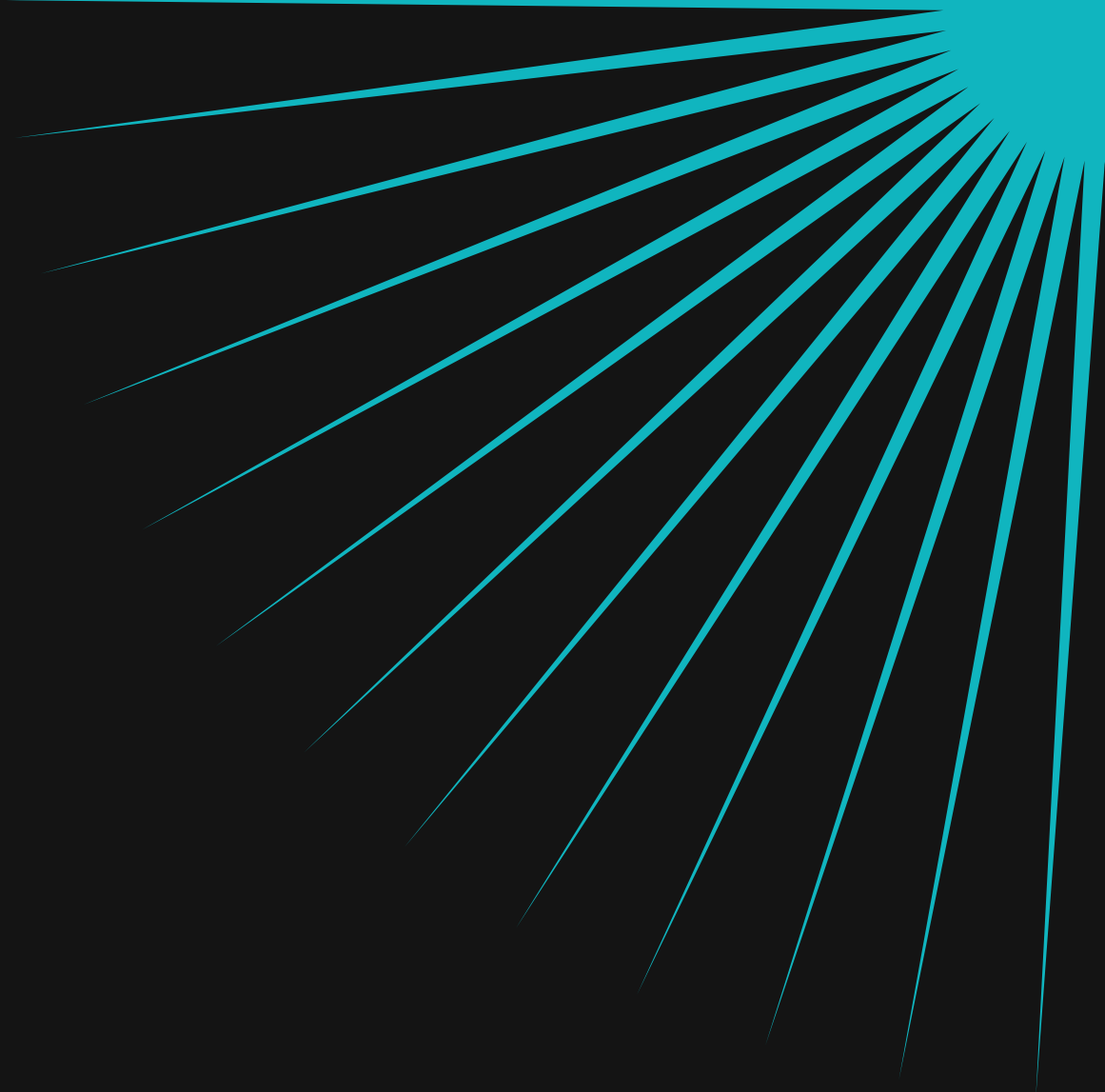


Abordagem 3

- Embedding Pré treinado em Português
- Modelos Híbridos LSTM e GRU com camada convolucional

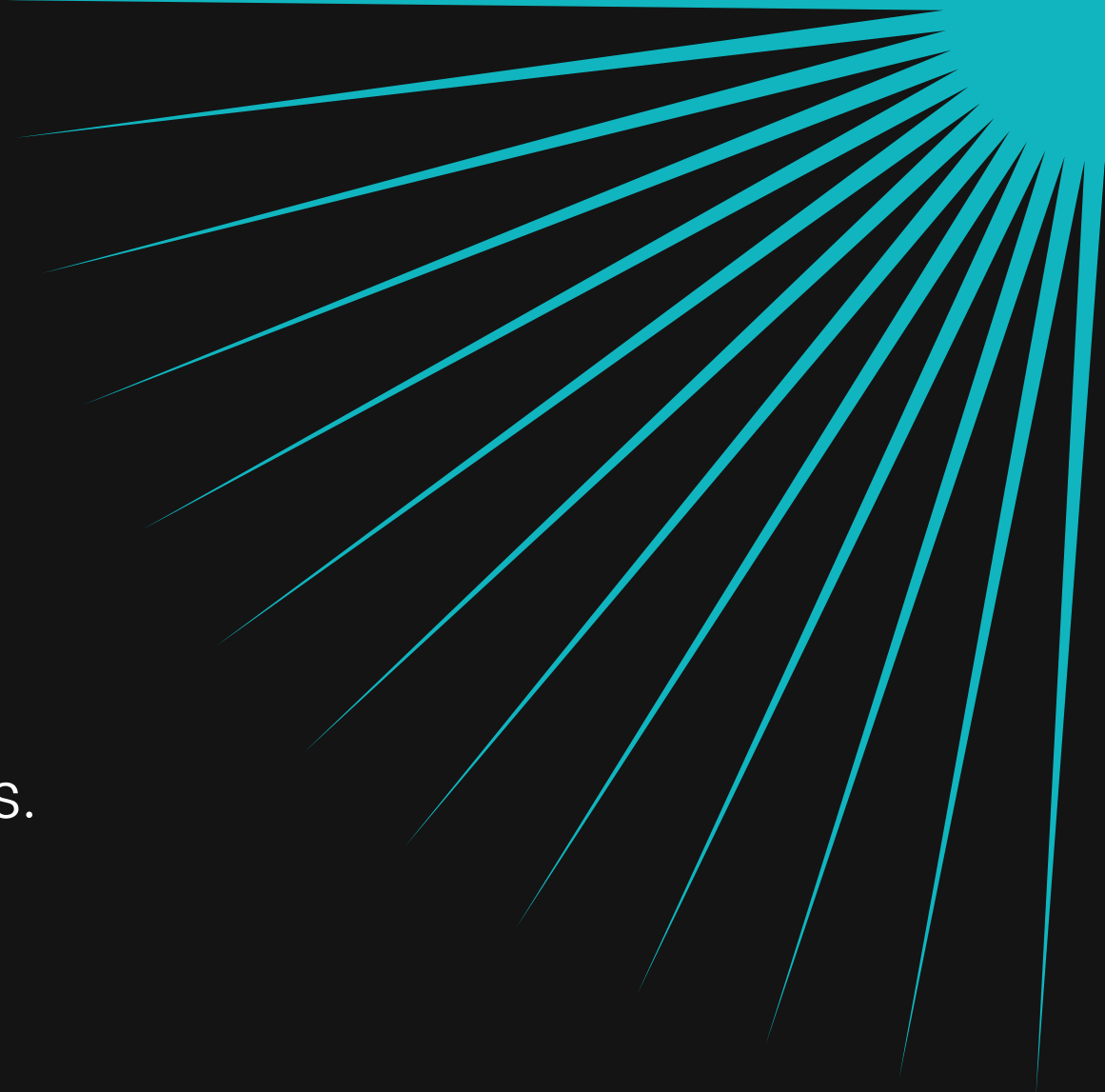
Embedding Word2Vec CBOW de 100 Dimensões

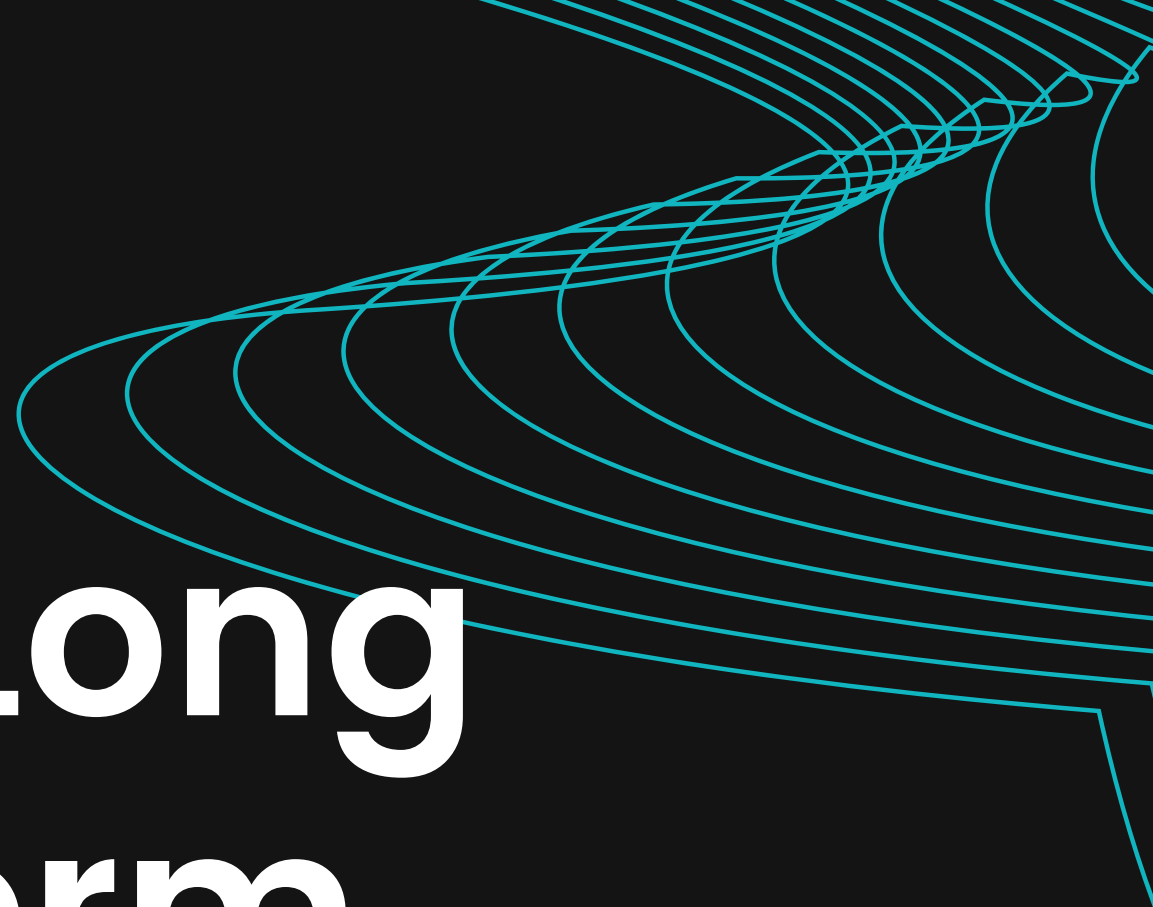
- Origem: Repositório NILC (Núcleo Interinstitucional de Linguística Computacional) USP São Carlos
- Cada palavra = Vetor de 100 números
- Esses números representam a posição da palavra em um espaço matemático que reflete seu significado semântico.
 - Palavras semelhantes estarão próximas no espaço vetorial (ex: "bom" e "ótimo").
- Por que 100 Dimensões?
 - Cada dimensão captura um “atributo oculto” da palavra, como:
 - Emoção
 - Função
 - Intenção
- 100 dimensões é o equilíbrio entre armazenar detalhes suficientes sobre as palavras sem tornar o modelo muito complexo.



Tokenização e Preparação dos Dados

- Tokenizer (Keras): Converte as palavras do texto em sequências numéricas.
- Funções de Tokenização:
 - Sequências Numéricas: As frases são convertidas em sequências de números que representam as palavras.
- Padronização de Sequências:
 - Padding: As sequências são preenchidas até o comprimento máximo (`max_length = 120`).
 - Truncating (post): Remove palavras extras do final de sequências longas.
 - Padding (post): Adiciona zeros ao final de sequências curtas.
- Vocab Size: 929.607 palavras únicas, representadas no modelo CBOW de 100 dimensões.





LSTM – Long Short Term Memory

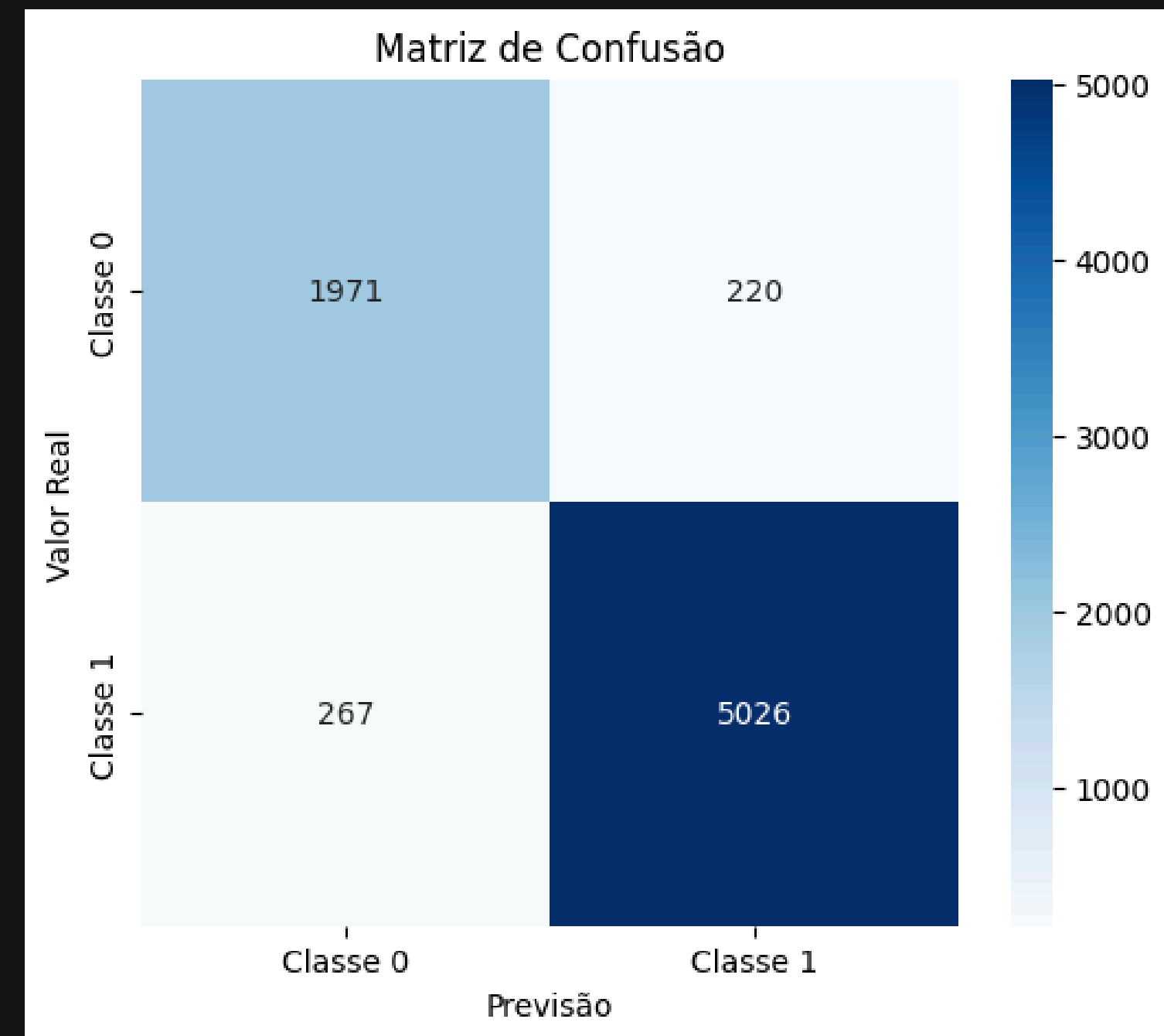
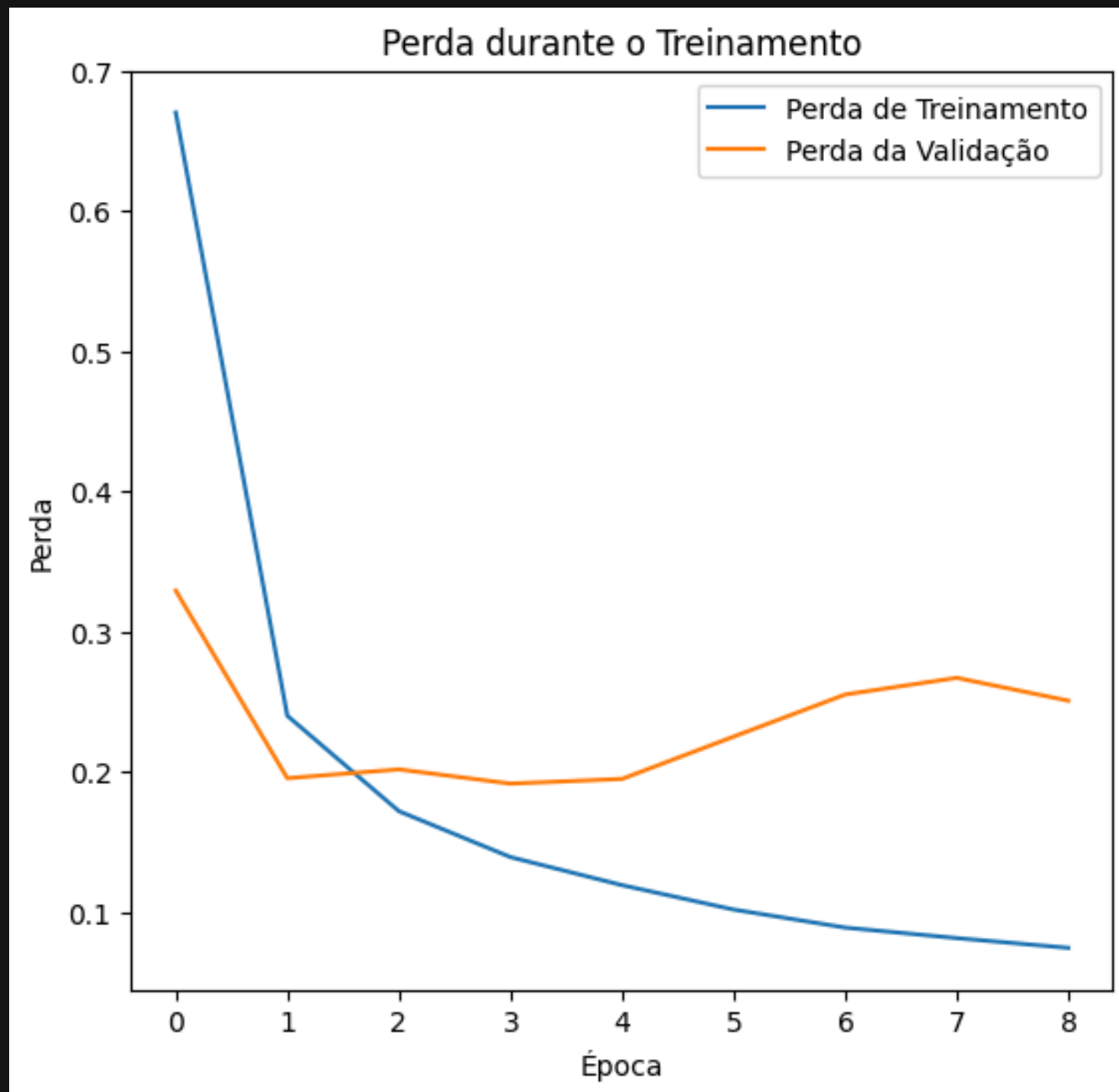
- Arquitetura:
- Embedding → Conv → LSTM bidirecional → Dense.
- Acurácia: 93%

Arquitetura LSTM

- Camada Conv1D + MaxPooling:
 - Conv1D: 32 filtros, kernel size 5, ativação ReLU.
 - MaxPooling1D: Pooling de tamanho 2.
- LSTM Bidirecional:
 - 2 camadas LSTM (64 unidades):
 - 1ª camada retorna sequências, 2ª retorna estados finais.
 - Dropout e Recurrent Dropout: 30% para evitar overfitting.
- Regularização:
 - Dropout (50%) e Batch Normalization antes da camada densa para melhorar generalização.
- Camada Densa:
 - 64 neurônios, ativação ReLU, regularização L2 ($\lambda=0.01$).

Treinamento

- 20 épocas
- Batch Size: 128
- Otimizador: Adam (learning rate = 0.001)
- Função de Perda: Binary Crossentropy
- Early Stopping:
 - Monitoramento de val_loss (perda na validação).
 - Paciencia: 5 épocas sem melhora.
 - Após 4 épocas, o menor val_loss foi alcançado.
- Recuperação de Pesos:
 - Melhor modelo recuperado a partir da época 4, onde o menor val_loss foi observado.



- Acurácia: 93% no teste
- Tempo de execução: 20 min



GRU – Gated Recurrent Unit

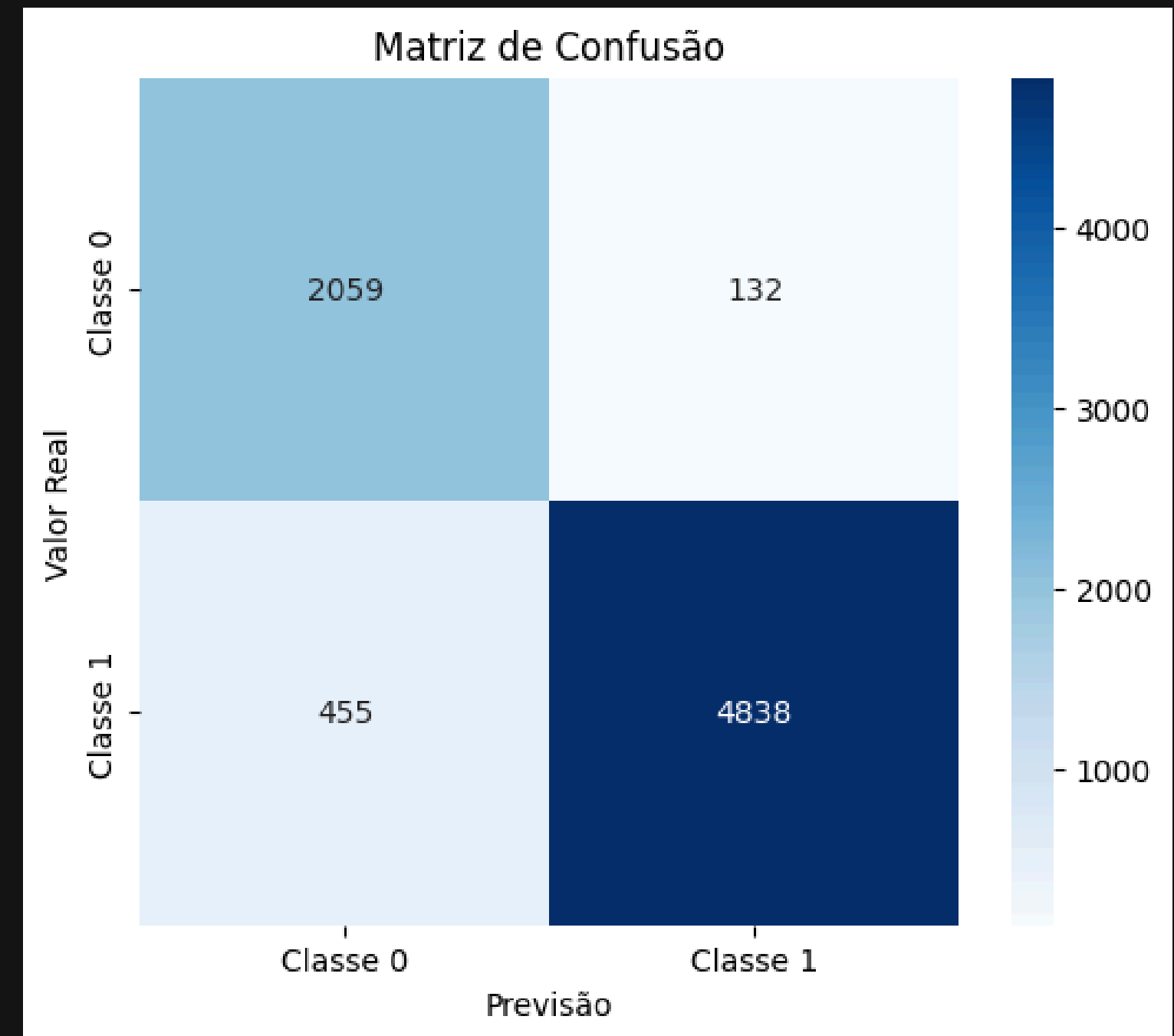
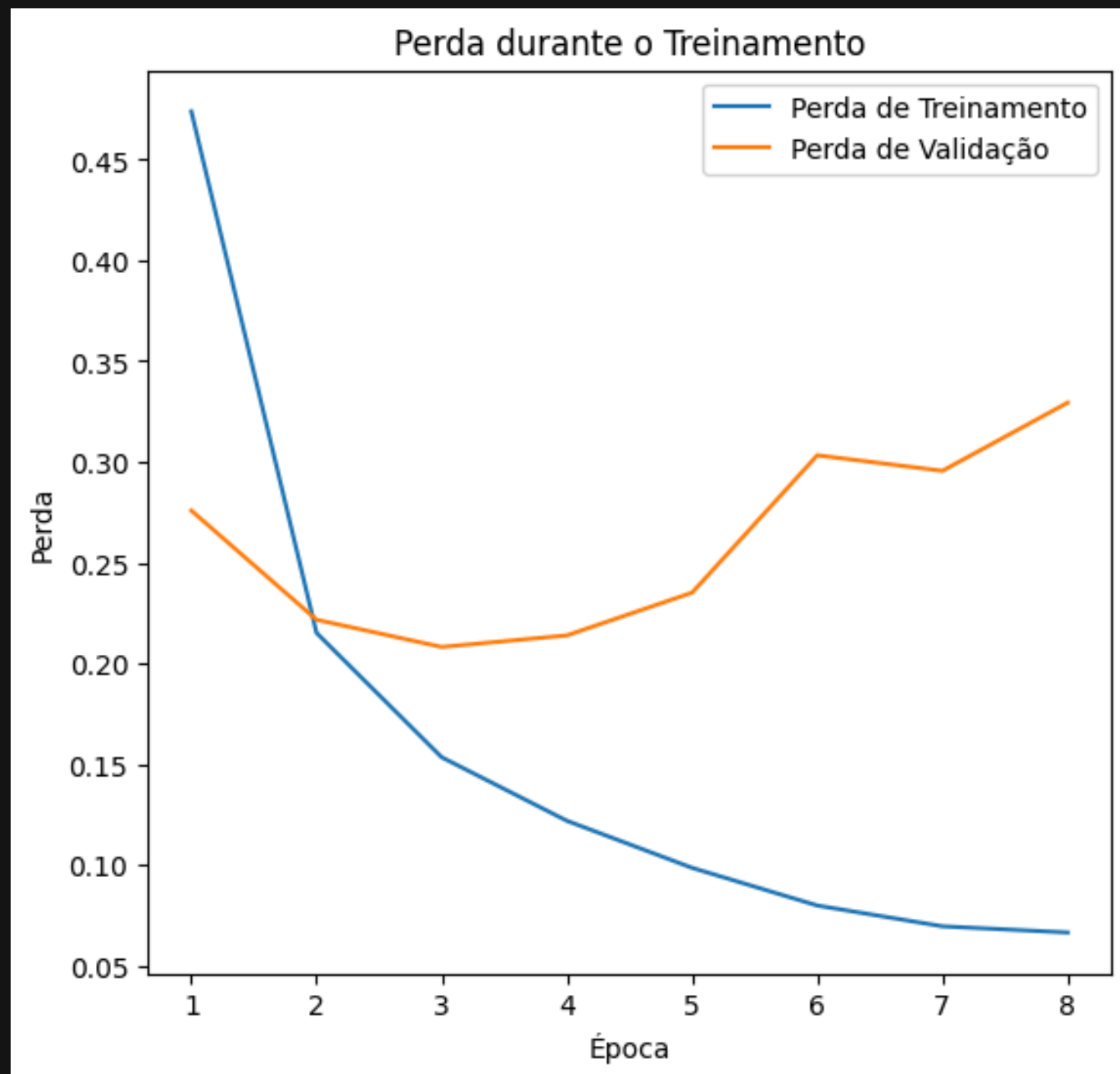
- Arquitetura:
- Embedding → Conv → GRU bidirecional → Dense.
- Acurácia: 92% .

Arquitetura GRU

- Camada Conv1D + MaxPooling:
 - Conv1D: 32 filtros, kernel size 5, ativação ReLU.
 - MaxPooling1D: Pooling de tamanho 2 para redução de dimensionalidade.
- GRU Bidirecional:
 - 2 camadas GRU (64 unidades):
 - 1ª camada retorna sequências, 2ª retorna estados finais.
 - Dropout de 20%.
- Regularização:
 - Dropout (30%) e Batch Normalization aplicados antes da camada densa.
- Camada Densa:
 - 64 neurônios, ativação ReLU, regularização L2 ($\lambda=0.005$).

Treinamento

- 20 épocas
- Batch Size: 128
- Otimizador: Adam (learning rate = 0.001)
- Função de Perda: Binary Crossentropy
- Early Stopping:
 - Monitoramento de val_loss (perda na validação).
 - Paciencia: 5 épocas sem melhora.
 - Após 3 épocas, o menor val_loss foi alcançado.
- Recuperação de Pesos:
 - Melhor modelo recuperado a partir da época 3, onde o menor val_loss foi observado.



- Acurácia: 92% no teste
- Tempo de execução: 1 min

BERT

(Bidirectional Encoder
Representations from
Transformers)

é um modelo de
aprendizado profundo para
**NLP (Processamento de
Linguagem Natural).**

BERT

Modelo Bidirecional:

- Lê o contexto em ambas as direções (esquerda e direita) para entender o significado completo.
- Exemplo: A palavra "banco" pode se referir a uma instituição financeira ou a um assento, dependendo do contexto.

Pré-treinamento e Fine-tuning:

- Pré-treinamento: O modelo é treinado em grandes volumes de texto para aprender a estrutura da linguagem.
- Fine-tuning: Ajustado para tarefas específicas, como classificação de sentimentos.

O Mascaramento no BERT

- **Processo:**
 - Uma fração das palavras de entrada é substituída por um token especial [MASK].
 - O modelo é treinado para prever essas palavras usando o contexto das palavras ao redor.
- **Vantagem:**
 - O MLM permite que o modelo entenda as relações semânticas complexas, pois ele processa todas as palavras da frase simultaneamente, ao contrário de abordagens unidirecionais.
- **Exemplo:**
 - Frase: "Attention [MASK] well."
 - O modelo tenta prever a palavra "works" com base no contexto das outras palavras

Por que o mascaramento é importante?

- **Ele força o modelo a entender o contexto completo da frase, aprendendo a prever palavras com base no que veio antes e depois.**
- **Isso é diferente de outros modelos que leem apenas da esquerda para a direita ou da direita para a esquerda. O BERT lê bidirecionalmente e, com o mascaramento, consegue captar relações mais profundas entre palavras.**

Impacto na Análise de Sentimentos:

- **Com o mascaramento, o BERT aprende a lidar melhor com frases complexas e nuances, como:**
 - **"O produto não é ruim."**
 - **O modelo consegue entender que "não é ruim" representa algo positivo, mesmo com a palavra "ruim" presente.**
- **o BERT transforma cada palavra em um vetor de números (embedding), onde palavras com significados semelhantes ficam próximas no espaço vetorial. Ele então usa essa representação numérica para fazer previsões.**

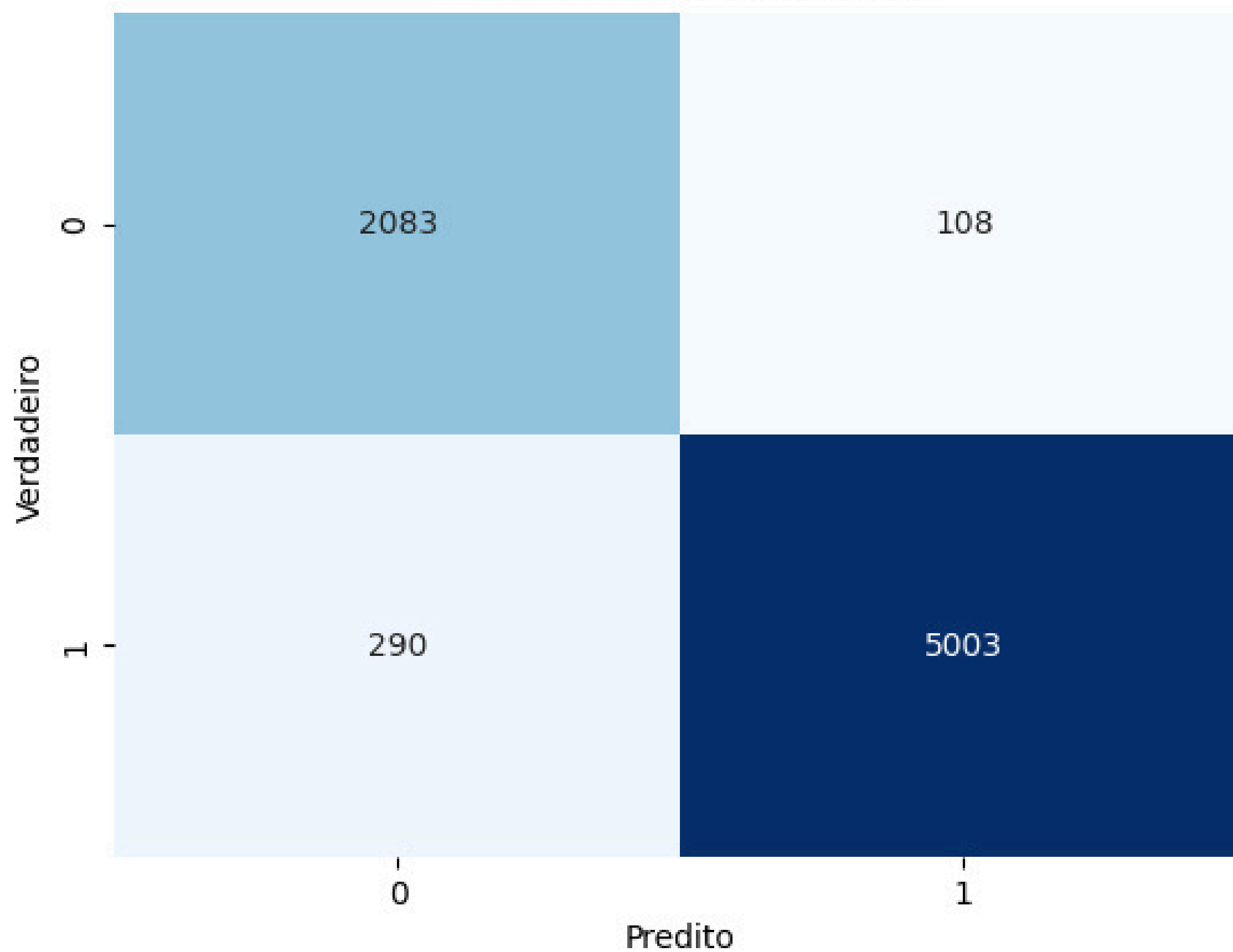
Como o BERT Melhora a Classificação de Sentimentos

- **Vantagens:**
 - Captura relações semânticas complexas entre palavras.
 - Entende nuances de contexto em frases complexas.
- **Exemplo:**
 - "O forno é pequeno, mas eficiente."
 - O BERT pode entender que, apesar da palavra "pequeno", o sentimento geral é positivo.
- **Benefícios:**
 - Entendimento amplo e profundo do idioma, permitindo que ele seja adaptado para uma variedade de tarefas específicas com poucos dados adicionais e com relativamente poucas camadas extras no topo.
 - Ideal para dados complexos, mas requer maior poder computacional.

Arquitetura e Ajustes Realizados:

- **Camadas:**
 - **BERT Base para português:** Modelo base com camadas Transformer (12 layers, 110 milhões de parâmetros).
 - **Camada densa (TFBertForSequenceClassification)**
 - Terá 2 neurônios, um para cada classe. Esses neurônios recebem o vetor de 768 dimensões e, com base nos pesos aprendíveis e no bias, produzem duas saídas (logits), cada uma representando a pontuação para uma das classes
- **Batch e Dataset:**
 - **Batch Size: 16**
- **Otimizador:**
 - **Adam com $\text{learning_rate} = 5e-5$.**
- **Função de Perda:**
 - **Sparse Categorical Crossentropy**
- **Early Stopping**
 - **Número de Épocas: 3**

Matriz de Confusão - Teste



- Acurácia: 95% no teste
- Tempo de execução: 45 min

Comparação de Desempenho

Acurácia Tempo de Execução

Modelo

DNN 92.4% < 1 min.

CNN 92.7% 7 min.

CNN LSTM 93% 20 min.

CNN GRU 92% 1 min.

BERT 95% 45 min.

- Abordagem 1:
 - DNN: ACC 92,4%, Tempo de exec. < 1 min.
- Abordagem 2:
 - CNN: ACC 92,7%, Tempo de exec. 7 min.
- Abordagem 3:
 - LSTM: ACC 93%, tempo de exec. 20 min
 - GRU: ACC 92%, temp de exec. 1 min
- BERT: 95%, tempo de exec. 45 min



Perguntas?

Obrigado!

