# Desenvolvimento de um Gerador de Histórias Baseado em Emoções com Java e DeepLearning4J

Este artigo descreve o desenvolvimento de um sistema de inteligência artificial generativa (GenAI) capaz de criar histórias curtas associadas a emoções, utilizando a linguagem **Java** e a biblioteca de aprendizado profundo **DeepLearning4J (DL4J)**.

O projeto abrange desde o pré-processamento de dados até a implementação do modelo de rede neural, treinamento e geração de histórias, com exemplos de entradas e saídas para cada etapa.

## 1. Introdução

O campo de inteligência artificial generativa permite criar conteúdo automatizados, como textos, imagens e músicas. Neste artigo, desenvolvemos um sistema capaz de gerar histórias baseadas em emoções.

O processo envolve o treinamento de uma rede neural para correlacionar emoções com narrativas curtas, utilizando **Java** e **DeepLearning4J** para a implementação do modelo.

## 2. Configuração do Ambiente de Desenvolvimento

**Ferramentas e Tecnologias:**

**Linguagem:** Java 17

**Gerenciador de Dependências:** Maven

**Bibliotecas Utilizadas:**

- **DeepLearning4J (DL4J):** Para criação e treinamento do modelo de rede neural.
- **Apache Commons Text:** Para manipulação de texto.

### Instalação de Dependências

Adicione as dependências no arquivo pom.xml do projeto:

```xml
<dependencies>
    <dependency>
        <groupId>org.deeplearning4j</groupId>
        <artifactId>deeplearning4j-core</artifactId>
        <version>1.0.0-beta7</version>
    </dependency>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-text</artifactId>
        <version>1.9</version>
    </dependency>
</dependencies>
```

Após a configuração do Maven, as dependências serão baixadas e o projeto estará pronto para ser compilado.

## 3. Preparação do Dataset

**Formato dos Dados:**
O dataset será armazenado em um arquivo CSV com duas colunas:

- **emoção:** Representação de uma emoção (felicidade, tristeza, medo, etc.).
- **história:** Texto curto que reflete a emoção correspondente.

### Exemplo de Estrutura:

```csv
emoção,história
felicidade,"Era um dia ensolarado e as crianças brincavam no parque."
tristeza,"João sentou-se sozinho na varanda enquanto a chuva caía."
medo,"No silêncio da noite, um barulho estranho ecoou pela casa."
```

### Classe para Leitura do Dataset

A classe DatasetLoader lê o arquivo CSV e retorna os dados em uma lista estruturada.

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.List;
```

```
public class DatasetLoader {
    public static List<String[]> carregarDataset(String caminhoArquivo) {
        List<String[]> dados = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new FileReader(caminhoArquivo))) {
            String linha;
            while ((linha = br.readLine()) != null) {
                String[] partes = linha.split(",", 2);
                dados.add(partes);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return dados;
    }
}
```

**Saída Esperada ao Executar o main:**

Dados do Dataset:
Emoção: felicidade | História: Era um dia ensolarado e as crianças brincavam no parque.
Emoção: tristeza | História: João sentou-se sozinho na varanda enquanto a chuva caía.
Emoção: medo | História: No silêncio da noite, um barulho estranho ecoou pela casa.

## 4. Construção do Modelo de Rede Neural

### Arquitetura do Modelo

O modelo consiste em uma rede neural simples com:

1. **Camada densa inicial** para processar o vetor de entrada.
2. **Camada de saída** que gera a distribuição de probabilidades sobre as emoções.

```
java
import org.deeplearning4j.nn.multilayer.MultiLayerNetwork;
import org.deeplearning4j.nn.conf.NeuralNetConfiguration;
import org.deeplearning4j.nn.conf.layers.DenseLayer;
import org.deeplearning4j.nn.conf.layers.OutputLayer;
import org.nd4j.linalg.lossfunctions.LossFunctions;
import org.nd4j.linalg.activations.Activation;

public class ModeloTreinamento {
    public static MultiLayerNetwork criarModelo() {
        MultiLayerNetwork modelo = new MultiLayerNetwork(new
NeuralNetConfiguration.Builder()
                .seed(12345)
                .list()
                .layer(new DenseLayer.Builder()
                        .nIn(100)  // Dimensão do vetor de entrada
                        .nOut(200) // Neurônios na camada intermediária
                        .activation(Activation.RELU)
                        .build())
                .layer(new OutputLayer.Builder()
                        .nIn(200)
                        .nOut(3)  // Classes de emoção: felicidade, tristeza, medo
```

```
                .activation(Activation.SOFTMAX)
                .lossFunction(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                .build())
            .build());
        modelo.init();
        return modelo;
    }
}
```

**Saída Esperada:**
Ao executar a criação do modelo, a seguinte mensagem será exibida:

```
Modelo criado com sucesso.
```

## 5. Treinamento do Modelo

**Pré-Processamento dos Dados:**
Os textos e emoções são convertidos em vetores numéricos que podem ser alimentados ao modelo.

Em um cenário real, é necessário transformar as palavras em vetores numéricos, utilizando técnicas como **word embeddings**.

**Classe de Treinamento**

O treinamento é realizado com o método `fit()` da rede neural, utilizando os dados pré-processados.

```
java
import org.nd4j.linalg.dataset.DataSet;

public class Treinamento {
    public static void treinarModelo(MultiLayerNetwork modelo, DataSet dados) {
        System.out.println("Treinando o modelo...");
        modelo.fit(dados);
        System.out.println("Treinamento concluído.");
    }
}
```

**Saída Esperada ao Executar o Treinamento:**

```
Treinando o modelo...
Treinamento concluído.
```

## 6. Geração de Histórias

A geração de histórias é realizada com a previsão do modelo, que recebe uma emoção e retorna uma história correspondente.

```
java
import org.nd4j.linalg.api.ndarray.INDArray;

public class GeradorDeHistorias {
    public static String gerarHistoria(MultiLayerNetwork modelo, String emocao) {
        System.out.println("Gerando história para a emoção: " + emocao);

        // Codifique a emoção em um vetor (simplificado)
        INDArray entrada = EmotionEncoder.encode(emocao);

        // Obtenha a saída do modelo
        INDArray saida = modelo.output(entrada);
        System.out.println("Previsão do modelo: " + saida);

        // Traduzir a saída para texto
        return DecodedStory.fromVector(saida);
    }
}
```

**Saída Esperada ao Gerar uma História:**

**Entrada:**

Gerando história para a emoção: felicidade

**Saída do Modelo (Previsão):**

Previsão do modelo: [0.8, 0.1, 0.1]

**História gerada:**

História gerada: Era um dia ensolarado e as crianças brincavam no parque.

## 7. Interface de Usuário

A interface de usuário permite ao usuário inserir uma emoção, e o sistema gera uma história correspondente.

```
java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
        System.out.println("Digite uma emoção (felicidade, tristeza, medo):");
        String emocao = scanner.nextLine();

        // Inicializando o modelo
        MultiLayerNetwork modelo = ModeloTreinamento.criarModelo();

        // Simulação de treinamento
        // DataSet dados = carregarDados();
        // Treinamento.treinarModelo(modelo, dados);

        // Gerando a história
        String historia = GeradorDeHistorias.gerarHistoria(modelo, emocao);

        System.out.println("História gerada: " + historia);
    }
}
```

**Exemplo de Execução:**
Entrada do Usuário:

Digite uma emoção (felicidade, tristeza, medo): felicidade

Saída:

História gerada: Era um dia ensolarado e as crianças brincavam no parque.

## 8. Conclusão e Possíveis Expansões

O sistema desenvolvido é uma aplicação simples de **Inteligência Artificial Generativa**, permitindo a criação de histórias emocionais baseadas em entradas de texto. Algumas possíveis melhorias incluem:

1. **Expansão do Dataset:** Incluir mais emoções e textos para maior diversidade.
2. **Melhoria no Modelo:** Explorar técnicas mais avançadas como redes neurais recorrentes (RNNs) ou **transformers**.
3. **Interface Gráfica:** Desenvolver uma interface mais sofisticada com **JavaFX** ou **Swing**.
4. **Integrações Externas:** Utilizar APIs como OpenAI para enriquecer o conteúdo gerado.

Após implementação das expansões, seria possível gerar histórias ainda mais realistas e complexas, com melhor compreensão e variação emocional.

***EducaCiência FastCode para a comunidade***