



# Aprendizagem Profunda - Módulo 1

## Grupo 2

Hugo Ramos  
João Vale  
Leonardo Barroso  
Luís Borges  
Tomás Oliveira



# Índice

- Construção do dataset
- Metodologia para avaliar resultados dos modelos
- Modelos com implementação própria
- Modelos com implementação em Tensorflow
- Análise dos resultados obtidos
- Trabalho futuro



# Construção do dataset

## Dataset 1: Construir dataset manualmente

- Entradas humanas: Web Scraping de papers científicos do site ARXIV.org com Beautiful Soup e extração do seu texto e processamento de dados
- Entradas de IA: Gerar entradas através de várias LLMs (ChatGPT, Claude, etc)

### Problemas:

- Dificuldade em gerar grandes quantidades de prompts de IA sem repetições e com contextos diferentes
- Demasiada dificuldade em aprender padrões
- Dados difíceis de filtrar e limpar



# Construção do dataset

## Dataset 2: Optimizar um dataset existente

- Dataset inicial com 2 milhões de entradas do Kaggle
- Filtrar apenas pelo conteúdo científico (campo lexical) e dividir as entradas de 80 a 120 palavras
- Balancear as entradas de AI e Humano

### Problemas:

- Filtragem de dados requer keywords inseridas manualmente
- Entradas muito similares entre si
- Demasiada dificuldade em aprender padrões



# Construção do dataset

## Dataset 3: Utilizar dataset existente

- Dataset do hugging face
- Com apenas 4000 entradas
- Fazer limpeza do dataset (remover erros de formatação, tabs, etc)

### Vantagens:

- Modelos demoram pouco tempo para executar as previsões
- Bons resultados nas previsões

# Metodologia para avaliar resultados dos modelos

1. Accuracy de teste e validação do modelo
2. Performance do modelo com as previsões do dataset fornecidos pelo professor (80 entradas até ao momento)
3. Performance com testes manuais
  - Gerar frases em várias LLMs com cerca de 100 palavras
  - Gerar entradas humanas no Quora e Wikipedia com cerca de 100 palavras

Que fatores ter em consideração para performance em 2 e 3?

- A precisão em prever os dados (accuracy final percentual)
- O grau de confiança (intervalo 0 a 1) com que prevê determinada entrada

Accuracy: 0.7375

Correctly Classified Samples:

	Label	Prediction	Source	Label_actual
0	Human	0.123358	Quora	Human
1	Human	0.133760	Sitepost	Human
2	AI	0.632273	GPT	AI

# Modelos com implementação própria

## Preparação dos dados



### **SimpleTokenizer**

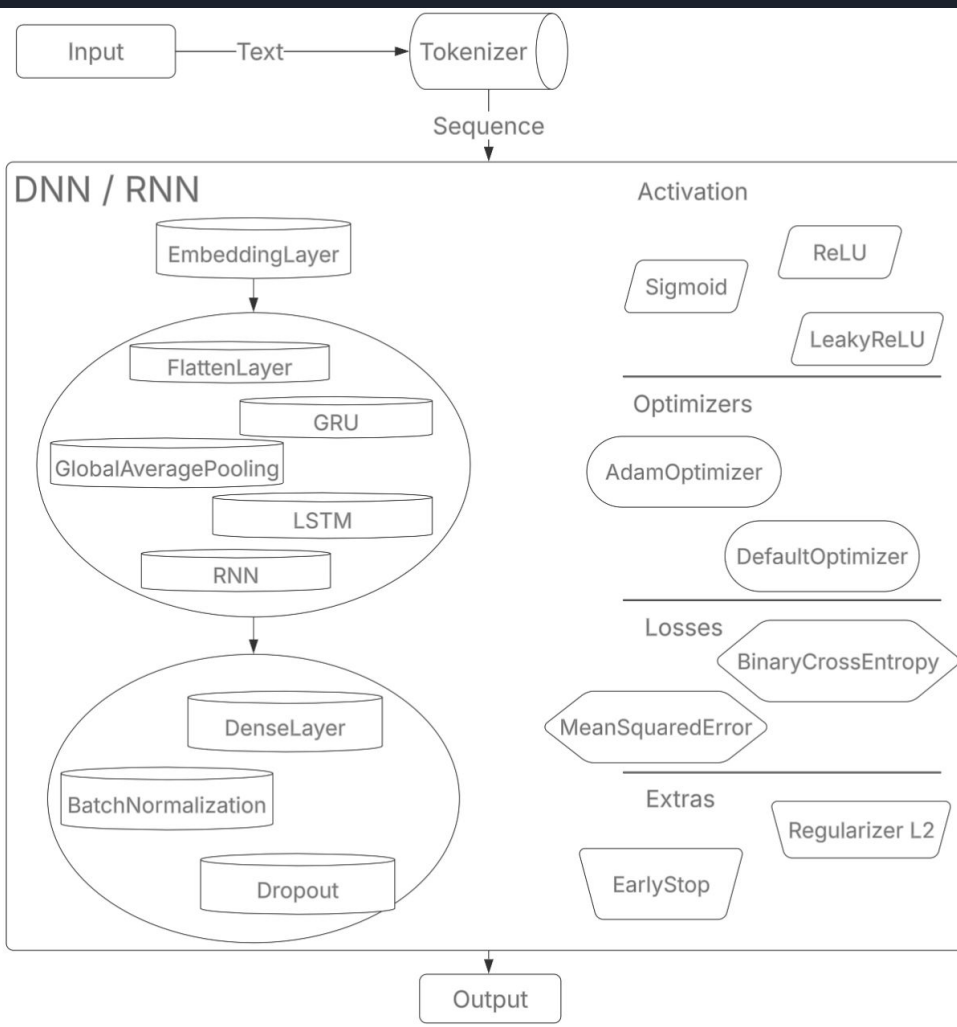
- Separação por espaços
- Converte todo o texto para minúsculas
- Mapeamento palavra-índice
- Padding com 0

### **AdvancedTokenizer**

- Gera unigrams
- Remoção de stopwords
- Filtro por frequência mínima
- Padding manual

### **RobustTokenizer**

- Regex para tokenização







# Modelos com implementação própria

## DNN (Deep Neural Network)

### **Embedding Layer**

- Converte tokens numéricos em representações vetoriais

### **GRU Layer**

- Identifica padrões sequenciais

### **GlobalAveragePooling**

- Reduz a dimensionalidade das matrizes

### **LeakyReLU**

- Evita neurónios “mortos”

### **Adam Optimizer**

- Learning Rate adaptativo




# Modelos com implementação própria

## DNN (Deep Neural Network) - Resultados

- Estrutura do melhor modelo (DNN):
  - Camada Embedding
  - 1 Camada GRU
  - 1 Camada GlobalAveragePooling
  - 1 Camada Dense
- Compilação e treino do modelo:
  - Learning Rate = 0.0001 Adam
  - Epochs = 25
  - EarlyStop = 10
  - Batch\_Size = 32

Modelo	Accuracy Teste	Accuracy Validação	Accuracy Validação (Dataset professor)
DNN Simples	0.67	0.6815	0.4750



# Modelos com implementação própria RNN (Recurrent Neural Network)

Mais eficientes para processamento sequencial - capturam melhor dependências

Foram criadas duas layers: RNN e LSTM

## **RNN:**

- Usa a técnica de Xavier/Glorot para inicialização
- Usa a tangente hiperbólica para calcular o output na forward\_propagation
- Calcula erros e atualiza pesos na backward\_propagation
- Usa bptt\_trunc para limitar o número de timesteps usados

## **LSTM:**

- Usa um cell state para armazenar informação importante a longo prazo
- Implementa 3 gates: forget, input e output, cada uma com os seus pesos
- forget gate: decide que informação remover do cell state
- input gate: decide que nova informação guardar no cell state
- output gate: decide que informação do cell state usar como output



# Modelos com implementação própria

## RNN (Recurrent Neural Network) - Resultados

- Estrutura do melhor modelo (RNN):
  - Camada Embedding
  - 1 Camada RNN
  - 1 Camada Dropout
  - 1 Camada Dense
- Compilação e treino do modelo:
  - Learning Rate = 0.0001
  - Epochs = 5
  - Batch\_Size = 32

Modelo	Accuracy Teste	Accuracy Validação	Accuracy Validação (Dataset professor)
RNN Simples	0.6831	0.6705	0.5750
RNN LSTM	0.6573	0.7028	0.5375



# Modelos com implementação em Tensorflow

## Preparação dos dados

- **Tokenizer (Keras)**
  - Converte o texto em sequências de inteiros com base na frequência de palavras
  - Utilização de padding para que todas as sequências tenham o mesmo comprimento
  - Necessário para usar camadas de Embedding
- **TF-IDF (TfidfVectorizer sklearn)**
  - Converte o texto em vetores com base na frequência das palavras pela sua importância
  - Não guarda a ordem das palavras
- **Embeddings**
  - Keras Embedding Layer
  - BERT (SentenceTransformer)



# Modelos com implementação em Tensorflow - DNN (Deep Neural Network)

- Estrutura do modelo:
  - Camada Embedding
  - 1 Camada AveragePooling1D - Downscaling
  - 4 Camadas Dropout
  - BatchNormalization para estabilizar o treino
  - 5 Camadas Dense
- Compilação e treino do modelo:
  - Otimizador Adam
  - EarlyStopping
  - ReduceLROnPlateau
  - 20 epochs


Accuracy Teste	Accuracy Validação	Accuracy Validação (Dataset professor)
0.9618	0.9676	0.65

# Modelos com implementação em Tensorflow

## RNN (Recurrent Neural Network)

- Estrutura do melhor modelo (LSTM):
  - Camada Embedding
  - 2 Camadas LSTM
  - 3 Camadas Dropout
  - BatchNormalization para estabilizar o treino
  - 2 Camadas Dense
- Compilação e treino do modelo:
  - Otimizador Adam
  - EarlyStopping
  - ReduceLROnPlateau
  - 15 epochs

Modelo	Accuracy Teste	Accuracy Validação	Accuracy Validação (Dataset professor)
RNN Simples	0.9630	0.9753	0.6250
RNN LSTM	0.9433	0.9522	0.7375
RNN LSTM (BERT)	0.8187	0.8336	0.5875
RNN GRU	0.8841	0.8891	0.7125



# Modelos com implementação em Tensorflow - CNN (Convolutional Neural Network)

- Estrutura do modelo:
  - Camada Embedding
  - 2 Camadas Conv1D
  - 2 Camadas MaxPooling1D
  - 2 Camadas Dropout
  - Camada Flatten para transformar a saída 2D em 1D para a camada Dense
  - BatchNormalization para estabilizar o treino
  - 3 Camadas Dense
- Compilação e treino do modelo:
  - Otimizador AdamW
  - EarlyStopping
  - ReduceLROnPlateau
  - 50 epochs

Accuracy Teste	Accuracy Validação	Accuracy Validação (Dataset professor)
0.9766	0.9661	0.6125



# Trabalho Futuro

- Melhorar os modelos manuais
- Utilizar LLMs para prever os resultados (se aplicável)
- Melhorar accuracy nos modelos com o Tensorflow
  - Modelo com dificuldades em prever, principalmente, as frases geradas pelo Claude

Accuracy: 0.7250

Misclassified Samples:

	Label	Prediction	Source	Label_actual
5	Human	0.089927	Claude	AI
6	Human	0.295848	GPT	AI
7	AI	0.587474	Wikipedia	Human
9	Human	0.274408	Claude	AI
12	Human	0.228653	Claude	AI
13	Human	0.345124	Claude	AI
20	AI	0.636160	Wikipedia	Human
22	Human	0.240277	Claude	AI
23	AI	0.662945	Quora	Human
28	Human	0.447925	Claude	AI
32	AI	0.575411	Paper	Human
42	Human	0.359312	Claude	AI
46	AI	0.835760	Quora	Human
48	AI	0.793478	Paper	Human