Relatório - Projeto Final

Processamento de Linguagem Natural

FilosofIA

Integrantes do trabalho:

- Guilherme Iram
- Tales Nobre
- Raphael Nascimento

Professor:

- Yuri Malheiros

1) Apresentação do problema

A filosofia é a matéria primordial da humanidade. Tomados por seu espírito, muitos pensadores em suas respectivas épocas escreveram muito a respeito de sua própria investigação filosófica. Entretanto, devido à barreira natural da morte física comum a todos os humanos, eles só puderam escrever a respeito daquilo que viveram no passado, sem poder dizer nada a respeito do futuro em diante a partir de sua morte. Pensando nisso, decidimos criar um modelo de Processamento de Linguagem Natural que fosse capaz, com base nos livros escritos de um número seleto de autores, ser capaz de avaliar se uma dada frase de input no modelo, retorna qual seria o autor mais próximo de ter escrito e/ou afirmado aquela dada frase.

2) Objetivos

O trabalho tem por objetivo, através de técnicas de Processamento de Linguagem Natural, criar um modelo de rede neural artificial, capaz de identificar qual filósofo, baseado em suas obras escritas, combinaria melhor com uma determinada sentença do usuário.

3) Dados utilizados e pré-processamento dos dados

Os dados utilizados foram livros em inglês já disponíveis em domínio público de diversos autores da filosofia, sociologia, psicologia, literatura etc. A fonte principal desses livros foi o Project Gutenberg (disponível em https://www.gutenberg.org/) , que é um site com mais de 70.000 ebooks disponíveis gratuitamente. Uma das opções de download era justamente o texto puro com código UTF-8 e formato .txt, o que caiu como uma luva para o uso do modelo de Processamento de Linguagem Natural.

O principal pré-processamento realizado foi excluir todo o texto que envolvia os direitos e detalhes a respeito do Project Gutenberg que todo livro disponibilizado por eles tinha em sua composição. Além disso, adicionamos ao dataset 2 colunas além do texto de cada livro em si, que são o título e o autor da obra. Isso foi automatizado via baixar os arquivos e escrevendo seus respectivos títulos como “nome do autor-nome da obra.txt”.

Após isso, o próximo passo foi fazer um tratamento inicial tentando remover todo tipo de ruído dos textos dos filósofos devido a sua origem do Project Gutenberg, e além disso, transformamos cada um dos seus livros em uma série de sentenças/ períodos com label associado ao autor desse mesmo livro. A partir desse novo dataset com as diversas frases que compunham as obras dos autores, finalizamos a etapa de processamento dos dados.

4) Rede Neural

Os dados de entrada foram submetidos a uma vetorização utilizando a estratégia do TF-IDF. Devido ao tamanho das matrizes geradas a partir do TF-IDF, que poderiam causar problemas de memória, optamos por utilizar uma amostra do conjunto de dados de treinamento original.

Os parâmetros foram definidos de forma empírica. Os parâmetros da rede neural foram selecionados de forma empírica. Escolhemos ter 50 neurônios nas camadas de entrada, camada escondida e camada de saída. Após experimentar diferentes otimizadores, constatamos que o Adam, com uma taxa de aprendizado (learning rate) de 1e-4, apresentou os melhores resultados.

Devido à natureza multiclasse do problema em questão, utilizamos a função de perda cross entropy. Essa função é apropriada para problemas em que há múltiplas classes e busca minimizar a divergência entre as probabilidades previstas pela rede e as classes reais dos dados.

A imagem abaixo ilustra o código implementado pelo nosso grupo:

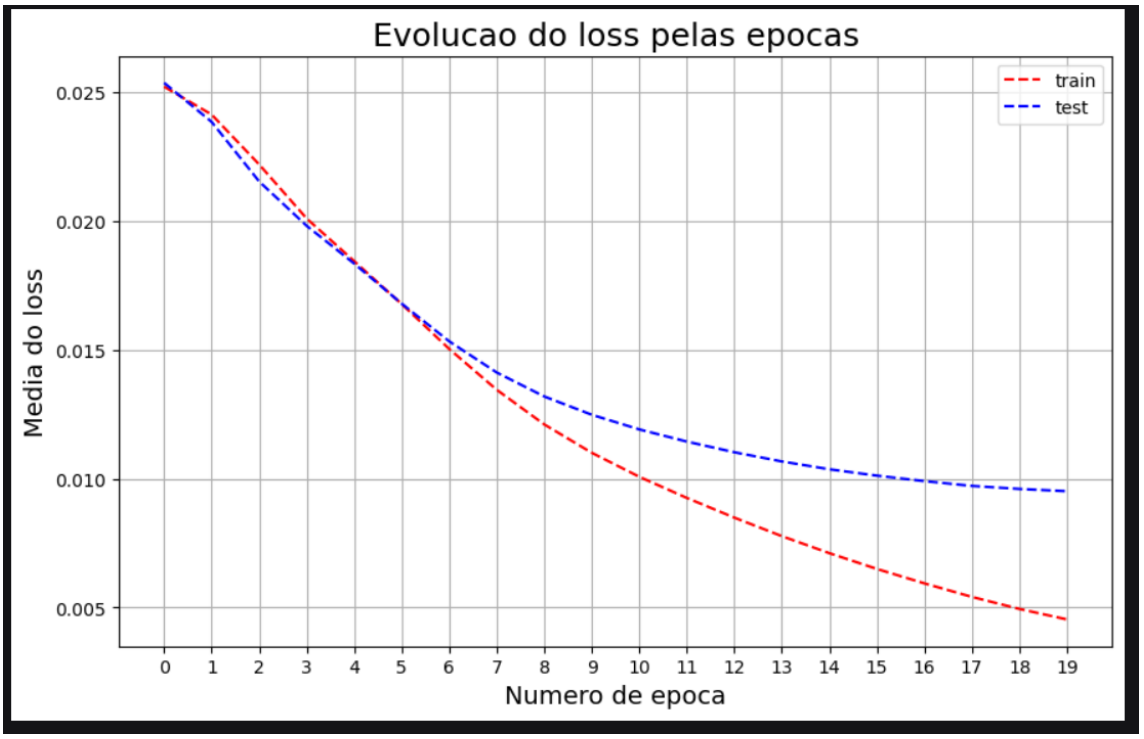
```
device = 'mps' if torch.backends.mps.is_available() else 'cpu'
print('Using {} device'.format(device))

class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()

        self.linear_relu_stack = nn.Sequential(
            nn.Linear(len_entrada, 50),
            nn.ReLU(),
            nn.Linear(50, 50),
            nn.ReLU(),
            nn.Linear(50, len_saida)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

model = NeuralNetwork().to(device)
print(model)

loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)
```

Vale ressaltar que o propósito desta rede neural é avaliar qual filósofo tem maior probabilidade de se relacionar com uma determinada sentença de entrada em formato de texto. Nossa classificação foi focada em cinco filósofos renomados: Aristóteles, Nietzsche, Platão, Schopenhauer e St. Tomás de Aquino.

5) Resultados

Temos uma imagem que mostra um gráfico da evolução da perda do modelo em função das épocas. Nesse gráfico, observamos que a perda do modelo diminui à medida que o treinamento avança, indicando que o modelo está aprendendo e se ajustando aos dados.

No entanto, a partir da época 9, começamos a observar um fenômeno conhecido como overfitting, que é quando o modelo se ajusta muito bem aos dados de treinamento, mas começa a ter um desempenho inferior em dados não vistos anteriormente, como os dados de teste. Isso pode ser observado no gráfico, onde a perda no conjunto de treinamento continua a diminuir, mas a perda no conjunto de teste fica, praticamente, estagnado.



Por fim, temos uma imagem que ilustra a interface intuitiva e amigável da nossa aplicação, permitindo que os usuários interajam facilmente com o sistema e obtenham resultados precisos de classificação de texto com base nos filósofos analisados.