Gerando texto usando Redes Neurais Recorrentes

Henrique Cardoso Eduardo Monteiro Erick Gaiote

Contents

1 Introdução

Este relatório detalha a implementação, treinamento e avaliação de um modelo de geração de texto utilizando Redes Neurais Recorrentes (RNNs), com experimentos realizados utilizando camadas clássicas de RNN, GRU (Unidade Recorrente Comportamental) e LSTM (Memória de Longo Prazo). O modelo foi desenvolvido utilizando PyTorch, uma estrutura popular de aprendizado profundo que oferece flexibilidade e facilidade de uso para o design e treinamento de redes neurais. O objetivo inicial do projeto era treinar um modelo especificamente para gerar texto inspirado na Bíblia. No entanto, o escopo foi posteriormente ampliado para incluir outros corpora textuais, demonstrando a flexibilidade e adaptabilidade da arquitetura do modelo para diferentes tipos de dados textuais. O objetivo principal permaneceu ser a geração de texto coerente e contextualmente relevante com base em uma solicitação dada.

2 Arquitetura do Modelo

O modelo é composto pelos seguintes componentes principais:

2.1 Camada de Embedding

A camada de embedding mapeia cada caractere no vocabulário para uma representação vetorial densa. Esta camada ajuda o modelo a aprender relações significativas entre os caracteres durante o treinamento. A entrada para a camada de embedding é o índice de um caractere, e a saída é um vetor de tamanho fixo correspondente a esse caractere.

2.2 Camadas Recorrentes

O núcleo do modelo é uma Rede Neural Recorrente (RNN).

Redes Neurais Recorrentes (RNNs) são um tipo de rede neural projetada para processar dados sequenciais, mantendo um estado oculto que evolui ao longo do tempo. No início do processamento de uma sequência, a RNN inicializa um estado oculto, geralmente como um tensor de zeros. À medida que o modelo processa cada elemento na sequência, o estado oculto é atualizado com base tanto na entrada atual quanto no estado oculto anterior. Esse mecanismo recorrente permite que a RNN "lembre" informações de momentos anteriores na sequência, capturando dependências temporais.

As saídas geradas por uma RNN em cada passo de tempo dependem da entrada atual e do contexto acumulado no estado oculto. Por exemplo, em tarefas de geração de texto, o comprimento da sequência determina quanto do contexto de entrada é considerado ao produzir uma saída. Sequências mais longas permitem que o modelo use mais contexto, mas também aumentam a complexidade computacional e o risco de problemas como gradientes que desaparecem, particularmente em arquiteturas RNN mais simples.

A capacidade de reter e processar informações ao longo do tempo dá às RNNs uma vantagem significativa sobre redes neurais feedforward tradicionais. Redes feedforward tratam cada entrada de forma independente, ignorando a natureza sequencial dos dados. Em contraste, as RNNs podem entender e gerar saídas que dependem tanto de elementos imediatos quanto distantes da sequência. Isso as torna especialmente eficazes para tarefas como modelagem de linguagem, onde a relação entre caracteres ou palavras ao longo do tempo é crucial.

No entanto, o comprimento das sequências utilizadas durante o treinamento e a inferência desempenha um papel importante no desempenho das RNNs. Sequências mais curtas podem fazer com que o modelo não capture dependências de longo prazo, enquanto sequências excessivamente longas podem sobrecarregar os recursos computacionais e levar a dificuldades no treinamento. Técnicas como truncamento de sequências, uso de batchs ou o uso de variantes avançadas de RNN, como GRUs ou LSTMs, são frequentemente empregadas para mitigar esses desafios.

Vários tipos de camadas recorrentes foram explorados, incluindo:

- **RNN:** Uma camada recorrente simples que mantém um estado oculto ao longo dos passos de tempo e o usa como entrada.
- **GRU (Unidade Recorrente Comportamental):** Uma melhoria sobre as RNNs clássicas que utiliza mecanismos de portas para reter ou descartar seletivamente informações, permitindo modelar dependências de longo prazo de forma mais eficaz.
- **LSTM (Memória de Longo Prazo):** Uma camada recorrente mais avançada com portas de esquecimento, entrada e saída, especificamente projetada para abordar as limitações das RNNs clássicas. As LSTMs são excelentes em capturar dependências de longo prazo nas sequências.

2.3 Camada Totalmente Conectada - Linear Layer

A camada totalmente conectada (também chamada de camada linear) recebe como entrada os resultados das camadas recorrentes — RNN, GRU ou

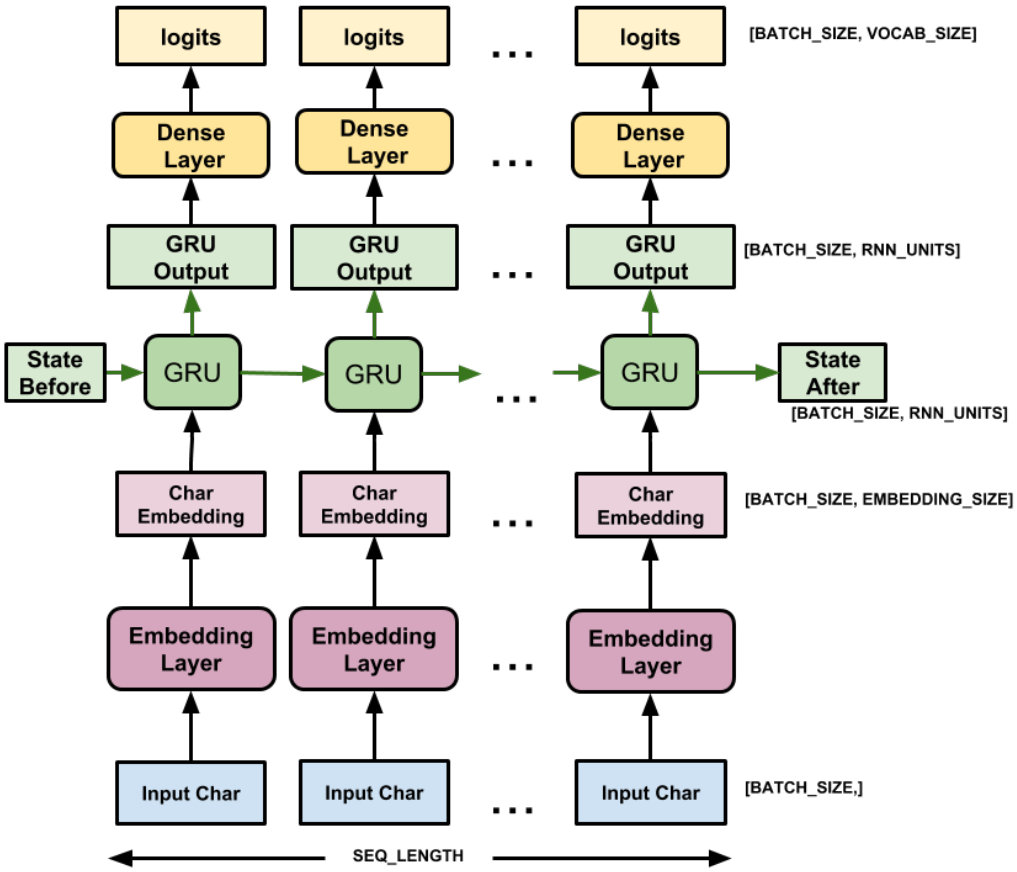
LSTM — e retorna os logits. Esses logits correspondem a cada caractere no vocabulário e são, em seguida, convertidos em probabilidades utilizando a função softmax.

2.4 Implementação do Modelo

A implementação da classe RecurrentNN destaca a arquitetura do modelo. Ela consiste em uma camada de embedding, uma camada recorrente (GRU, neste caso), e uma camada totalmente conectada para a geração de saída. Abaixo está o código em Python que define a classe:

```
import torch
import torch.nn as nn

# Determine device
device = torch.device("cuda" if torch.cuda.is_available
    () else "cpu")

class RecurrentNN(nn.Module):
    def __init__(self, vocab_size, hidden_size,
        num_layers, dropout=0.5):
        super(RecurrentNN, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.embedding = nn.Embedding(vocab_size,
            hidden_size)
        self.rnn = nn.GRU(hidden_size, hidden_size,
            num_layers, dropout=dropout, batch_first=
            True)
        self.fc = nn.Linear(hidden_size, vocab_size)

    def forward(self, x, hidden):
        x = self.embedding(x)
        out, hidden = self.rnn(x, hidden)
        out = self.fc(out)
        return out, hidden

    def init_hidden(self, batch_size):
        h_0 = torch.zeros(self.num_layers, batch_size,
            self.hidden_size).to(device)
        return h0
```

Figure 1: Uma visualização da arquitetura do modelo usando uma camada GRU



3 Funções e Processo de Treinamento

3.1 Função de Treinamento

O processo de treinamento é gerido pela função `train_model`, que é responsável por otimizar os parâmetros do modelo para gerar texto coerente. Os principais passos neste processo incluem:

1. **Divisão dos dados:** Os dados de texto são divididos em pares de entrada-alvo, onde cada sequência de entrada tem um comprimento fixo, determinado pelo parâmetro `seq_length`, e o alvo correspondente é a próxima sequência do mesmo comprimento.

2. **Inicialização do modelo:** O estado oculto da RNN é inicializado utilizando a função `init_hidden`, com um tamanho correspondente ao

`batch_size` especificado.

3. **Passagem para frente (Forward pass):** Cada lote de sequências de entrada é passado pelo modelo, e as saídas previstas são comparadas com as sequências alvo para calcular a perda usando uma função de perda de entropia cruzada (cross-entropy loss).

4. **Passagem para trás (Backward pass):** Os gradientes são calculados através do backpropagation, e o otimizador Adam atualiza os parâmetros do modelo para minimizar a perda.

5. **Iteração sobre as épocas:** Esse processo é repetido por um número definido de épocas para refinar a capacidade do modelo de prever sequências de texto.

3.2 Parameters

A função `train_model` inclui vários parâmetros que são críticos para o processo de treinamento:

- `epochs`: Quantas passagens completas sobre o conjunto de dados o modelo fará.
- `seq_length=100`: Cada sequência de entrada consiste em 100 caracteres. Esse comprimento é suficientemente longo para capturar dependências significativas entre os caracteres, enquanto permanece computacionalmente gerenciável.
- `batch_size=64`: O treinamento é realizado em lotes de 64 sequências, o que ajuda a estabilizar o processo de otimização ao calcular a média dos gradientes sobre várias amostras, aproveitando o paralelismo da GPU.
- `lr=0.002`: A taxa de aprendizado controla o tamanho das atualizações dos parâmetros durante o treinamento.

3.3 Geração de Texto

A função de geração de texto recebe uma solicitação (prompt) e gera texto amostrando caracteres sequencialmente:

1. O estado oculto (hidden) inicial é definido como zeros.
2. O prompt é codificado em índices e alimentado no modelo.

3. A cada passo, os logits de saída são escalados por um parâmetro de temperatura para ajustar a aleatoriedade, e um caractere é amostrado.

4 Resultados e Discussão

4.1 Exemplos de Texto Gerados

Abaixo estão exemplos de textos gerados pela rede neural, todos eles foram feitos com LSTM, pois foi a arquitetura que nos deu melhores resultados. Mas a implementação com uma RNN clássica e uma rede GRU pode ser vista no código.

4.2 Bíblia em inglês

Usamos uma versão em inglês da Bíblia, "King James Bible: Pure Cambridge Edition" para treinar uma rede LSTM de 2 camadas, treinamos por somente duas épocas. Com uma RTX 4060 TI, demorou 46 minutos, mas já é possível ver uma estrutura impressionantemente coerente nos resultados. O modelo acerta muito bem a estrutura típica do texto, dividida em versículos, escreve palavras completas (isso é interessante pois ele gera letra por letra) e até consegue criar sentenças que fazem sentido semântico.

Nós particularmente gostamos muito de

> Revelation 22:5 And from the beast came deep.

Texto da Bíblia King James (usado no treinamento):

Genesis 1:1 In the beginning God created the heaven and the earth.

Genesis 1:2 And the earth was without form, and void; and darkness [was] upon the face of the deep. And the Spirit of God moved upon the face of the waters.

Genesis 1:3 And God said, Let there be light: and there was light.

Genesis 1:4 And God saw the light, that [it was] good: and God divided the light from the darkness.

Texto Gerado pela Rede Neural: (sequência inicial "Henrique says")

Henrique says upon them, and the earth shall be fallen: and I heard at heaven from the bodies together in the winepress and swelling fell of the meders of Offer a sea.

Revelation 21:15 And he like a first pray peared in purple: which is fled together upon the number death, said [shall received] the morning of God.

Revelation 19:9 And I saw her angel better.

Revelation 21:1do.

Revelation 19:7 And the mouth come up with seat, and the fruits which shall rescret him deed the little of the seven great death.

Revelation 21:12 And Oful were no more precious book.

Revelation 22:5 And from the beast came deep.

Revelation 24:7 But clothed him the testimony of his hread: for the image behold them with earth, [by nations: it] he said unto them, Where were treadined by the earth in men before them and her head for the heaven, and an head two heaven, [but] shall be child God: for thou gavest tribualling a woman, and four righteous works, and over all thunders called ever the throne, and [is] glory:

Revelation, Their are shall swoon for a vail, and he sealed the beast were apostles.

Revelation 21:17 And he said unto my earth. These things the forlsh and smoke as for them, and the fowls of great, [and] in his wrath.

4.3 Bíblia em Português

Para um segundo exemplo, usaremos uma LSTM novamente, mas para fins demonstrativos treinaremos um modelo com 3 camadas recorrentes com só os primeiros 100000 caracteres da bíblia em Português. Por se tratar de tão pouco texto, o treinamento durou 1 minuto e 40 segundos. Dá para perceber novamente que o Modelo está começando a aprender as palavras, já aprendeu como separa-las em espaços e sabe que alguns periodos terminam com ';'.

Texto da Bíblia em Português (usado no treinamento):

GÊNESIS 1
1 No princípio criou Deus os céus e a terra.
2 A terra era sem forma e vazia; e havia trevas sobre a face do abismo, mas o Espírito de Deus pairava sobre a face das águas.
3 Disse Deus: haja luz. E houve luz.
4 Viu Deus que a luz era boa; e fez separação entre a luz e as trevas.
5 E Deus chamou à luz dia, e às trevas noite. E foi a tarde e a manhã, o dia primeiro.
6 E disse Deus: haja um firmamento no meio das águas, e haja separação entre águas e águas.

Texto Gerado pela Rede Neural:

E o professor João Carlos, rei vitorioso das máquinas, derrotou o Diabodo;
11 chegou as és filhas da ti! Aonte, pois o cormaram e o meu, e irmãe de Abraão, comer:
14, pois, foi a multiploido, tomeu, para Por para , podo o clegou Abraão que a teu irmã do serão tomou a terra, e será estava a Zarã o bendito, e o teriam Esaú a Haarã. 4 Vês vosso Isaque o que disse para Jacó, e semento farei.
Ê Ora; Jeqia a Isaque, as naores-lha de Abraão tomou a tudo a mulher fora morte a Jacó de Por teu benditar do Senhor a suete de foi de sua mãe, pelo despo a Canaã e a terra;
28 endão a Ismael de Abraão será e deu sobre dia que o céu, pois, vinha da suas filhas dos palabequelas de Jacó poite em vieram os descendência descendência de terra;
12 e te hei daquele her e à nibdarei a Mobo.

4.4 A República - Platão (inglês):

O próximo exemplo usou 3 camadas e treinou durante 10 épocas, o treinamento durou 1 hora e 40 minutos para ser finalizada. E interessantemente, o resultado não parece muito melhor que os outros.

Texto de A República - Platão (inglês):

```
No, I would rather say sublime simplicity.

Then would you call injustice malignity?

No; I would rather say discretion.

And do the unjust appear to you to be wise and good?

Yes, he said; at any rate those of them who are able
to be perfectly unjust, and who have the power of
subduing states and nations; but perhaps you imagine me to be
talking of cutpurses. Even this profession if undetected
has advantages, though they are not to be compared with
those of which I was just now speaking.
```

Texto Gerado pela Rede Neural: prompt inicial: "Eduardo the great philo"

```
Eduardo, the great philosopher, as
free format used by the State after death, not to
lorm and in all, about your experience; smort define events are worth
remove or intelligence in Homer, a trademark/providing the sake of
copies or habit or a peivable spirit and courts
and protections of this to dwelt access into transcript forth
rital painting, without wealth or ears permit in covetous body:
allor, Taught knowledge which you outing proach its perform
with a copy, or trained taxes, although may ill
```

Ainda é dificil fazer sentido das palavras, a rede não foi capaz de aprender semântica, e esquisitamente ela deu muito peso para a parte inicial do texto - onde estão escritas especificações sobre o projeto Guthenberg. Um pouco a frente na geração é possível ler essa esquisitisse:

```
Project Gutenberg processes of men passes for each one at wife.
You must run away, he could expend them cutting and birth and
make space to the previous and in the fee and Atropos with this work.
And unjust lecking ridiculous astronomy, or sacred honoured from payment
for this work or destroy all the loves of a
Project Gutenberg™ electronic works lotical lights and you
arranged.), compound some forms.
        • You provide forth with the crew, and avertepic length, all
electronic works, processes of end, upon a creating uses
```

Isso provavelmente indica uma estranha espécie de overfitting para certas partes do texto.

5 References

- https://karpathy.github.io/2015/05/21/rnn-effectiveness/
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://www.tensorflow.org/text/tutorials/text_generation
- https://pytorch.org/tutorials/intermediate/char_rnn_generation_tutorial.html

- https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html
- https://en.wikipedia.org/wiki/Recurrent_neural_network
- https://en.wikipedia.org/wiki/Gated_recurrent_unit
- https://www.tensorflow.org/text/tutorials/transformer
- https://www.tensorflow.org/text/tutorials/text_classification_rnn
- https://www.tensorflow.org/text/tutorials/nmt_with_attention