118 RNN USING PyTorch & QUESTION ANSWERING SYSTEM USING PyTorch

Tags: Lecture Notes Machine Learning

A Recurrent Neural Network (RNN) is a type of neural network designed to recognize patterns in sequences.
Unlike traditional feedforward networks, which process inputs independently, RNNs maintain a memory of previous inputs by using loops in their architecture.

This makes them well-suited for tasks where context and order matter, such as time series forecasting, speech recognition and text generation.

How RNNs Work:

- **Sequential Processing**
  RNNs process data one element at a time.
  - **Hidden States (Hidden Vectors):**
    At each step, an RNN uses its previous hidden state (a form of memory) to process the current input and produce an output.
- **Information Persistence:**
  The hidden state is then passed to the next step, allowing it to "remember bits" of information that helps the network maintain context from previous inputs.
- **Contextual Understanding:**
  By retaining this information, RNNs can understand the relationships between different elements in a sequence, which is vital for tasks like understanding a sentence.

Key Features & Advantages:

- **Handles Time-Series Data:**
  Their ability to remember past information makes them excellent for data that unfolds over time, such as financial market data or sensor readings.
- **Natural Language Processing (NLP):**
  RNNs excel at understanding the nuances of language, allowing them to power applications like sentiment analysis, machine translation, and text generation.

- *Contextual Predictions:*
  They can predict the next word in a sentence, not just based on the current word, but also considering the words that came before it.

Common Application:

- **Speech Recognition:** Converting spoken language into text.
- **Text Generation:** Creating new text, like stories or code.
- **Machine Translation:** Translating text from one language to another.
- **Time Series Forecasting:** Predicting future trends in data like stock prices or weather patterns.

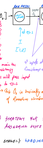
---

1) Run data_loader.ipynb DONE:

Ok.

2) Run main.ipynb DONE:

Completed. This is now combined and nicely provided the tutorial data used by each method.



Also this nicely can switch to English language. And even that tokenizer specific so we need to insert this work lab section.

The song of Hiawatha by Longfellow and not Pinocchio and when using a by using Pinocchio

Generally speaking the tokenizer is a tokenizer specific and so we need to insert this work lab section

3) Optimize Hiawatha text:

Can make functions to do all the work for us

Ok, this is basically a set of functions

4) Question and Answer Model Architecture:

DONE:1) Verify the output

DONE:2) Convert Bidir Output Dimensions into Dropdowns:

It can be individually fixed as need be instead the using lab setting

we find we need to make a variability so that we can do all the unique model options in the default

with the unique variable we will make it dictionary[model name]= variable model with value

that will provide a mechanism make to work work of the updates by referring to the corresponding value of the model from the dictionary[]

that will speed this numerical values to optimize model for long runtime if not perfect or safe

DONE:3) Optimize the RNN Architecture

DONE:4) Refactor the model so output

DONE:5) Performance Optimization

5) RNN Architecture for the model:

Generally speaking the tokenizer is a tokenizer specific and so we need to insert this work lab section

Hidden layers are transfer the read, the activation state, the vocabulary & dim-size

Input layers are data, vocabulary & dim-size

Output layers are vocabulary & dim-size

RNN NAME=QUESTION ANSWERING SYSTEM USING PYTORCH

September 24, 2025

```
Import Libraries
!pip install gdown
import os
!if [ -d data_set ]; rm -rf data_set; fi
!git clone https://github.com/Ryan-Rhys/The-Song-of-Hiawatha.git
!mv "The-Song-of-Hiawatha" data_set
!ls -l data_set
!head data_set/01\ Introduction.txt
!head data_set/02\ Indian\ Boyhood.txt
!head data_set/03\ Footprints\ in\ the\ Snow.txt
!head data_set/04\ The\ Hunting\ of\ the\ Deer.txt
!head data_set/05\ Picture\ Writing.txt
!head data_set/06\ Chibiabos.txt
!head data_set/07\ The\ Feast\ of\ Honor.txt
!head data_set/08\ The\ Death\ of\ Kwasind.txt
!head data_set/09\ Hiawatha's\ Childhood.txt
!head data_set/10\ Hiawatha's\ Friends.txt
Import Libraries
!pip install -q -U
!pip install -q -U "huggingface_hub[cli,hf_xet]"
!hf auth login --add-to-git-credential --force
import os, sys, glob, re, unicodedata, collections, pathlib, random, math, time, gc
from pathlib import Path
import numpy as np
import torch, torch.nn as nn, torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader, random_split
device = "cuda" if torch.cuda.is_available() else "cpu"
print("Device:", device)
if device == "cuda":
    torch.backend.cudnn.benchmark = True
torch.manual_segment_length = 512
torch.manual_stride = 256
torch.manual_min_tokens = 1024
torch.manual_max_chunks = 2000
torch.manual_cache = True
torch.manual_cache_dir = "./tokenized_cache"
torch.manual_shuffle = True
torch.manual_seed = 42
torch.manual_num_workers = 2
torch.manual_pin_memory = True
torch.manual_persistent_workers = True
torch.manual_prefetch_factor = 2
torch.manual_drop_last = True
torch.manual_verbose = True
torch.manual_log_every = 50
torch.manual_max_steps = None
torch.manual_max_epochs = None
torch.manual_max_time = None
torch.manual_max_memory = None
torch.manual_max_gpu_memory = None
torch.manual_max_cpu_memory = None
torch.manual_max_disk_memory = None
torch.manual_max_network_memory = None
torch.manual_max_latency = None
torch.manual_max_throughput = None
torch.manual_max_error_rate = None
torch.manual_max_retry_count = None
torch.manual_max_retry_delay = None
torch.manual_max_timeout = None
torch.manual_max_connections = None
torch.manual_max_requests = None
torch.manual_max_bandwidth
```