Pyrap
Neural Network Poetry
Luke Wood

Project Description

This script trains a neural network to write poetry or raps based on various files that have been placed in a folder. The network trains using random numbers as the input and poems or songs in the folder as the outputs. The final result can be quite humorous, but also displays the incredible power of neural networks.

Implementation

The first step in this program is setup. I first began by taking various song lyrics and scraping out punctuation along with the verse headers. I then created and ran the following script to train the neural network and save it to a file.

pyrap.py

```
from pybrain.datasets import SupervisedDataSet
from pybrain.utilities import percentError
from pybrain.tools.shortcuts import buildNetwork
from pybrain.supervised.trainers import BackpropTrainer
from random import randint
from pybrain.structure import FeedForwardNetwork
from pybrain.tools.customxml import NetworkWriter
#END IMPORTS

#CLASS DECLARATIONS BEGIN
#END CLASS DECLARATIONS
#FUNCTION DECLARATIONS BEGIN
def wordinlist(word_list,tword):
	for word in word_list:
		if word == tword:
			return True
	return False
def converttoint(word_list,words):
	asint = list()
	for word in words:
		try:
			asint.append(word_list.index(word))
```

```
                except Exception:
                        asint.append(-1)
        return asint
def inttowords(word_list,nums):
        aswords = list()
        for i in nums:
                if i < -.5:
                        aswords.append("")
                else:
                        try:
                                aswords.append(word_list[int(i)])
                        except Exception:
                                aswords.append("")
        return aswords

#FUNCTION DECLARATIONS END

#BEGIN VARIABLE DECLARATIONS
word_list = list()
insize=2
print("How long do you estimate the longest rap/poem is?")
outsize = int(input())
nwords = list()
ds = SupervisedDataSet(insize,outsize)
print("Enter the size for the three layers (High numbers will take a very long time, 
reccomended to keep below 30):")
size = int(input())
net = buildNetwork(insize,size,size,size,outsize)
trainer = BackpropTrainer(net,ds)
#END VARIABLE DECLARATIONS


#fetching list of all possible words
with open('progfiles/wordlist.txt') as f:
        content = f.readlines()
        for line in content:
                for subline in line.split(':'):
                        for word in subline.split():
                                word_list.append(word.strip())
#adding any nonexisting words from the raps to nwords
with open('progfiles/raps.txt') as f:
        content = f.readlines()
        for line in content:
                slines = line.split(":")
                rappername = slines[0]
                for subline in slines:
                        words = set(word.strip() for word in subline.split())
                        for word in words:
                                if not wordinlist(word_list,word.lower()):
                                        word_list.append(word.lower())
                                        nwords.append(word.lower())
#adding them in to the wordlist file
with open('progfiles/wordlist.txt', 'a') as f:
        for word in nwords:
                f.write(word + " ")

#opening up raps
with open('progfiles/raps.txt') as f:
        contents = f.readlines()
        for line in contents:
```

```
                        linesplit = line.split(":")
                        if(len(linesplit) ==2):
                                inp = list();
                                inp.append(randint(0,20))
                                inp.append(randint(0,20))
                                rap = linesplit[1].split()
                                rap = converttoint(word_list,rap)
                                for i in range(0,outsize-len(rap)):
                                        rap.append(-1)
                                if len(rap)>outsize:
                                        rap = rap[:-(len(rap)-outsize)]
                                if len(rap) == outsize:
                                        if len(inp) == insize:
                                                ds.addSample(inp,rap)
print("Training, this may take awhile.")
trainer.trainUntilConvergence()
print("Congratulations, your network has been trained")
print("Please enter where you want to save your file")
fname = input()
NetworkWriter.writeToFile(net,fname + ".xml")

n1 = randint(0,20)
n2 = randint(0,20)

output = net.activate((n1,n2))
output = inttowords(word_list,output)
toprint = ""
for word in output:
        if not word == "":
                toprint+=word+" "
print(toprint)
```

After this script finishes running, it is very simple to run the program over and over producing different songs or poems based on those which the network was trained on. Below is the source code for the script that runs the network.

```
from random import randint
import random
from datetime import datetime
import pyraptools
from pybrain.tools.customxml import NetworkReader
print("Enter the name of the rapper xml file")
rappername = input()
net = NetworkReader.readFrom(rappername+".xml")
word_list = list()
rapper_list = list()
random.seed(datetime.now())
#fetching list of all possible words
with open('progfiles/wordlist.txt') as f:
        content = f.readlines()
        for line in content:
                for subline in line.split(':'):
                        for word in subline.split():
                                word_list.append(word.strip())
```

```
#fetching list of rappers
with open('progfiles/rapperlist.txt') as f:
	content = f.readlines()
	for line in content:
		for word in line.split(","):
			if not word =="\n":
				if not word =="":
					rapper_list.append(word.strip())

n1 = randint(0,20)
n2 = randint(0,20)
print("Enter file to save the rap to.")
fname = input()
output = net.activate((n1,n2))
output = pyraptools.inttowords(word_list,output)
toprint = ""
for word in output:
	if not word == "":
		toprint+=word+" "
with open(fname, "w+") as f:
	f.write(toprint)
```

Results and Discussion

While the results of this program are primarily humorous, they also display some of the incredible potential of neural networks. Neural networks can be applied to almost any dataset that can be quantified in order to obtain useful information.

Concluding Remarks

Upon finishing this project, I realized just how computing intensive neural networks can be. One of my larger tests of this project took around 30 hours of computing time before the network was fully trained. Due to this, I am now creating a neural network library in C++ that I hope will yield much faster results.