Recurrent Neural Networks(RNN) Project: TV Script Generation

Faruk Geles

December 2020

Abstract

This project of the Udacity Deep Learning Nanodegree is about using Recurrent Neural Networks, to generate our own Seinfeld TV scripts. For this purpose we will use Seinfeld dataset of scripts from 9 seasons. The Neural Network will generate new fake TV scripts using the patterns which it recognizes i.e. learns from the training data.

1 The Goal of the Project

In this project we build a machine learning model, which will be tained with a TV script(Seinfeld), in order to generated a fake TV script from the learned patterns in the training data. The algorithm has two main parts:

- Proprocessing the data, embedding into a vector shape
- Generating a machine learning model using RNN and using the model to generate fake TV scrips

1.1 Used Data Set

The data which we use for this project are already provided in the project subfolder "$./data/Seinfeld_Scripts.txt$" The file $Seinfeld_Scripts.txt$ contain following data

- In this dataset there are roughly 46367 unique words
- The number of lines is 109233
- The average number of words in each line: 5.544240293684143

The first 10 lines of this dataset look like:

jerry: do you know what this is all about? do you know, why were here? to be out, this is out...and out is one of the single most enjoyable experiences of life. people...did you ever hear people talking about we should go out? this is what theyre talking about...this whole thing, were all out now, no one is home. not one person here is home, were all out! there are people trying to find us, they dont know where we are. (on an imaginary phone) did you ring?, i cant find him. where did he go? he didnt tell me where he was going. he must have gone out. you wanna go out you get ready, you pick out the clothes, right? you take the shower, you get all ready, get the cash, get your friends, the car, the spot, the reservation...then youre standing around, what do you do? you go we gotta be getting back. once youre out, you wanna get back! you wanna go to sleep, you wanna get up, you wanna go out again tomorrow, right? where ever you are in life, its my feeling, youve gotta go.

jerry: (pointing at georges shirt) see, to me, that button is in the worst possible spot. the second button literally makes or breaks the shirt, look at it. its too high! its in no-mans-land. you look like you live with your mother.

george: are you through?

jerry: you do of course try on, when you buy?

george: yes, it was purple, i liked it, i dont actually recall considering the buttons.

2 The Approach

For the TV script generation problem we use a Recurrent Neural Network(RNN)

- Prepocessing data
- Passing the data to a Recurrent Neural Network model to generate fake TV script

2.1 Data Preprocessing

The first thing to do to any dataset is pre-processing. Implement the following pre-processing functions below:

- Generate Loopup Table: To create a word embedding, you first need to transform the words to ids. In this function, create two dictionaries:
  - Dictionary to go from the words to an id, we'll call $vocab_to_int$
  - Dictionary to go from the id to word, we'll call int_to_vocab

  Return these dictionaries in the following tuple $(vocab_to_int, int_to_vocab)$
- Tokenize Punctuation

2.2 Network Architecture

1- In the first trial I started with the parameters

- $sequence_length = 100$
- $batch_size = 64$
- $num_epochs = 5$
- $learning_rate = 0.0005$
- $embedding_dim = 100$
- $hidden_dim = 500$
- $n_layers = 2$

In the training the loss stays approximately around 3.9+0.05 and 3.9-0.05, approximately around 4.0

2- In the second training I decreased the learning rate to 0.0005. And also removing more noise we can look at the batch size too small values like 1,2,4,8,16 result in too small and to large like 512,1024, 2048 could be computationally taxing and could result in worse accuracy so, 32-64-128-256 are potentially good starting values).

We also decrease the sequence length to 25 since the average number of words in each line is 5.54

So the number of layers depends on the complexity of the problem. The more complex the problem the more learning capacity will be needed. The number and architecture of the hidden units is the main measure for a model's learning capacity. If we provide the model with two much capacity it might tend to overfit and just try to memorize the training set(If the training accuracy is much better than validation accuracy, it could help to decrease the number of hidden units or we use here also dropout to avoid overfitting). So the number of hidden units, the more the better but much larger than ideal value can lead to overfitting. I decrease the $embedding_dim$ to 100 and increase $hidden_dim$ to 500

- $sequence_length = 20$
- $batch_size = 64$
- $num_epochs = 10$
- $learning_rate = 0.0005$
- $embedding_dim = 100$
- $hidden_dim = 500$
- $n_layers = 2$

3 Results of RNN Model for TV script generation

```
jerry: is to sliding in a little generous administration.

george:(shouting) oh, i know, i was so worried about the show.

elaine: what?

jerry: i was just wondering.....

george:(interrupting) well, i- i don't think you should be going to be able to get the bathroom.

elaine:(smiling) oh, yeah. yeah.

jerry: i think i got the whole thing off for the rest of the building.

elaine:(quietly) yeah, yeah.

jerry:(quietly) yeah, you know, i don't think you can do anything about that.

kramer: well, i gotta take it.

jerry: you know what? you know, i don't know what you do with you.

kramer: yeah.

jerry: what about the car?

george: i was just trying to make a call.

jerry: well, what about your mother?

kramer: i got it in my office!

jerry: you can't believe it?

kramer: yeah.

elaine: you can't believe this.

jerry: you know, i don't know what to do with the guy who lives in the building.

george: well, it's not my name.(he leaves)

jerry:(to kramer) you know you gotta go get some coffee.

george: yeah, yeah, yeah. i gotta go.(hangs up the door)

kramer: oh. oh, hi.

jerry: hey
```

Figure 1: Some results for fake TV script generated by our model.

4 Ideas for Future Work

- 1) More optimization of the parameters
- 2) More layers

- 3) Increasing the training episodes