Machine Learning Engineering Capstone Proposal

Thomas Lisankie
March 2018

Domain Background

Poetry and lyrical music are quite old artforms at this point. They aim to tell some sort of story and also have the words and rhythm stick in the reader/listener's mind. One of the elements that poets and writers will often incorporate is rhyme. However, thinking of a word that rhymes with another can be quite a pain. Rhyme dictionaries tend to attempt to alleviate this problem but often return words that just have a similar spelling and don't actually sound similar (since spelling and pronunciation often don't match well in English).
I personally want to work in this area because I want to take the first steps towards a set of algorithms that could write music that flows and rhymes well.
Here is an academic paper that has investigated similar situations using deep learning.

Problem Statement

The problem I am addressing is training a model that will tell a user how well two words the user inputs rhyme based on their pronunciation. This is a classification problem.

Datasets and Inputs

The dataset I will be using is based off of the CMU Pronouncing Dictionary. This is a dictionary compiled by researchers at Carnegie Mellon University which lists the spelling of English words along with a sequence of phonemes (the smallest unit of sounds in languages that distinguish words from one another) that represent the pronunciation of each word. I will refer to these sequences as phonemic transcriptions
However, this is not the dataset I will be using for training. The dataset I will be using is a sampling of the data in this dictionary where each of the transcriptions is paired with each of the other transcriptions once and each of these pairs is listed with a measure of how well they rhyme.
I am crafting my dataset by reading in the set of phonemic transcriptions, shuffling them and then only taking the first 5,000 on this newly shuffled list (the reason I'm doing this will be explained in a moment). I then create a new dataframe which consists of entries of each of these transcriptions paired with all the other transcriptions at least once (the 0th entry is popped off after each pairing to protect against unnecessary duplicates). I then apply a modified version of an algorithm I created a couple of years ago called Prhymer which returns a scalar value between 0.0 and 1.0 describing how similar the two words sound to one another. The modified version however will return categories where each of the categories is found by converting the numeric value to a floored value. For example, if two phoneme sequences receive a result from the algorithm of 0.412, the categorization mechanism will floor this value to 0.4. This way,

phoneme sequence comparisons will have 10 discrete categories in increments of 0.1 (0.0, 0.1, …, 0.9, 1.0) rather than an infinite number of possible values. This makes it possible to use an error metric like categorical cross entropy to evaluate the new model against the target.
The reason I only take 5,000 of the transcriptions is because the original CMU Pronouncing Dictionary has over 133,000 entries. If we are creating pairs of each of the transcriptions with one another (including duplicates for the sake of argument), this means squaring the number of entries in order to get the total number of pairs. 133,000^2 is 17,689,000,000 which is far too many entries to manage for my computational resources to handle in a reasonable amount of time and probably far more than enough for a model to get an idea of what it means for two sequences of phonemes to rhyme with one another. 5,000^2 on the other hand is only 25,000,000 which is a pretty large amount of data and my computational resources can handle it in a reasonable amount of time. I will be using 80% of the data for training and 20% of the data for testing.

Solution Statement

One possible solution here is to use a convolutional neural network model that uses at least a couple of convolutional layers. Each of these convolutional layers will be one-dimensional since we are dealing with text data and not image data (which would require two-dimensional convolution layers).

Benchmark Model

The benchmark model I will be using will be a random forest classifier trained on the same data that I will be training the solution model on.

Evaluation Metrics

One evaluation metric that could be used is categorical cross entropy. The benchmark model will have a consistent error rate of 0 of course since it is producing the targets for the solution model to fit. Categorical cross entropy is defined with the following formula:

$$H(p, q) = -\sum_{x} p(\mathbf{x}) log(q(\mathbf{x}))$$

If my solution model is able to classify more phoneme sequence pairs correctly than the benchmark model, I will consider that success.

Project Design

I am first going to have to explore the dataset I am creating. Since the vast majority of words do not rhyme with one another, almost all the data will probably have a Prhymer value of 0.7 or less. Because of this, it may be necessary to augment the data a bit so that the model is able to learn what rhyming pairs of sequences look like and not just what badly rhyming pairs look like.

If I am unable to augment the data, I may want to remove some of the data instead in order to make it less lopsided.
I may also want to visualize the data in order to gain further insights and possibly see where outliers fall so I can remove them.
After this, I am going to want to explore at least a few different model architectures in order to see which perform the best on the training set. Some CNN architectures that I am currently considering are LeNet, GoogLeNet, and ResNet architectures. I may also want to use a validation technique when I am trying out all these different models. These models will likely be deep convolutional networks.
Out of the model architectures that perform the best, I will tune the hyperparameters to further optimize and see which works best. Specifically, I think that number of units per layer, number of layers, varieties of layers, and activation functions will have to be tuned. I will start with only a few layers and increase the number of layers as necessary in order to match my evaluation goal.