Names Generator

objection:

train a character-level language model using RNN network to generate names given their origin country and first letter

---

model description:

The model gets as input at each stage a language, a letter and a state (represented as a matrix) and concatenated it to a single matrix. Afterward, the model uses a linear layer on the concatenated input, once for getting the new state and once for getting the output.
Then, the state is updated to the newly received state. After that, the new state and the input are passed to another linear layer. A dropout is performed on the received output and finally a $softmax$ for receiving the next letter.

Training process:

First, the training set transformed to a single array containing arrays of size 2, each holding a language and a name from that language. Then, a random shuffle is performed on the array.
During the training, each sub-array (language and a name) is transformed to a tensor vector by one hot function so it could be passed to the model as input.

The process:

- For each epoch, a sample is taken from the training set. The number of stages is the same as the length of the word.
- The model is getting the first letter in the word and the given language in addition to the state, which initialized to a tensor zero vector
- After the model predicts the next letter, the loss is calculated with respect to the real letter.
- A backward pass is performed to minimize the loss
- The model gets the next letter and so on until all letters are passed

Hyperparameters:

- $Learning\ rate\ =\ 0.0005$
- $Dropout\ probability\ =\ 0.1$

Examples of final generated names:

- German, ‘H’ : Haner
- Arabic, ‘A’: Asara
- Russian, ‘B’: Bakov
- French, ‘L’: Lener
- Italian, ‘M’: Marov

The attached files:

- $train.py$ – train the model and save the final weights to pkl file
- $eval.py$ – load the final weights and test the model on the test-set
- $utils.py$ – utility functions
- $weights.zip$ – contains $weights.pkl$ with final weights