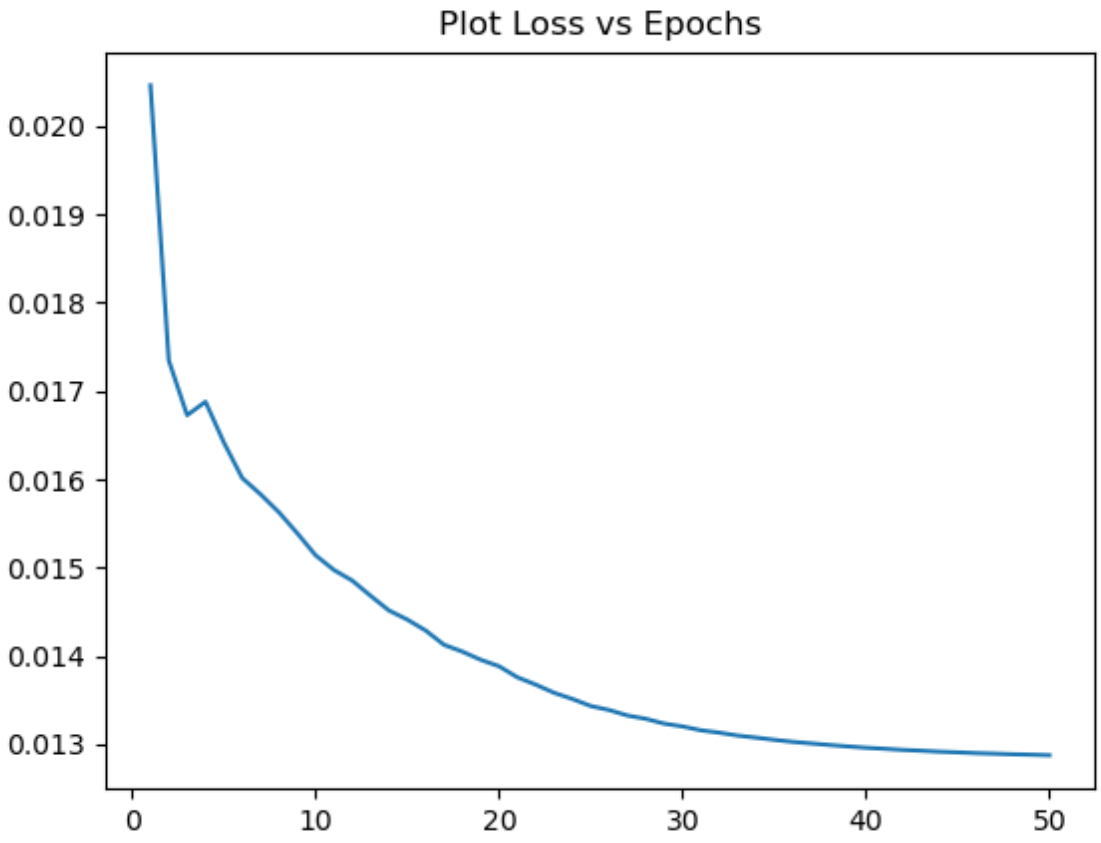
HMW7

1,2. As loss function I used the MSE loss that calculates the mean squared error between each element in the output and the true value of each letter of the specific name. The output of the model is a coded word, of the same length of the input.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

As is possible to see in the following plot the loss converges around the value of 0.0128, reaching an accuracy in the prediction of the next letter of about 73%. The training was done on 50 epochs.



This behavior is due to the fact the dataset is not very big and performances can't be improved anymore after this point, also because the model has its limit and we cannot reach 0 loss. In conclusion the behavior is the same of the one usually seen in the previous homework.

3. To change the choice of the letters and generate different names, I decided to take, for each step of the name generation, the 3 most probable next letters, and randomly (with respect to the specific probability) take one of them; as shown in the following code

```
prob = model(feed_mod).detach().numpy()[0][-1]
prob_letters = model(feed_mod).detach().numpy()[0][-1].argsort()[::-1]
letter_index = random.choices(prob_letters[0:3], weights=[prob[prob_letters[0]], prob[prob_letters[1]], prob[prob_letters[2]]])
letter = alpha_dict_rev[letter_index[0]]
```

As is possible to see I take the arg of the three most probable letters and I weight them with their probabilities when the function *random.choices* is called.
These are 20 names for with starting letter 'a' and letter 'x':

```
ariah
alisana
alisha
aleson
antoson
ana
alani
anna
alani
anna
annah
alana
alaree
arian
arianna
ariana
alie
alaree
anna
antoston
```

```
xistina
xayle
xinnelen
xalian
xaylan
xinter
xince
xinton
xinthanton
xalian
xinner
xonnery
xinton
xannelia
xintere
xoaish
xinceny
xilliana
xalene
xincelis
```