EEE 443/543 - Project #7

1. Introduction

In this project the aim was to design and train a character-level text generation LSTM. The goal is to generate person names.

2. Methodology

Initially, names.txt file containing 2000 names is downloaded and added to the working directory. The file is preprocessed such that all the letters are lower-case. The encoding process is done as described. 26 letters in English alphabet and a special character EON (End Of Name) is encoded using one-hot encoding.

The training sequence is generated as described. (x_i , y_i) i = 1,.., 2000, where each x_i is a length-11 sequence of 27 dimensional vectors to the LSTM, and the y_i is the corresponding desired output, which is again a length-11 sequence of 27 dimensional vectors. For names shorter than 11 letters EONs are appended to make each input equal length.

The loss function that is used is Cross Entropy Loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

Where:

- N: Number of samples.
- C: Number of classes.
- $y_{i,c}$: Ground truth (one-hot encoded) for sample (i) and class (c).
- $\hat{y}_{i,c}$: Predicted probability for sample (i) and class (c).

This loss measures how well the predicted probability distribution matches the true class, it gets smaller when the model is more confident about the correct answer.

The LSTM model also includes 128-size hidden layer followed with a size 128x27 fully connected layer in order to capture more complex features of the dataset.

2.1 Training Loss

The model trained for different number of epochs, and generated names as well as the loss graph are observed.

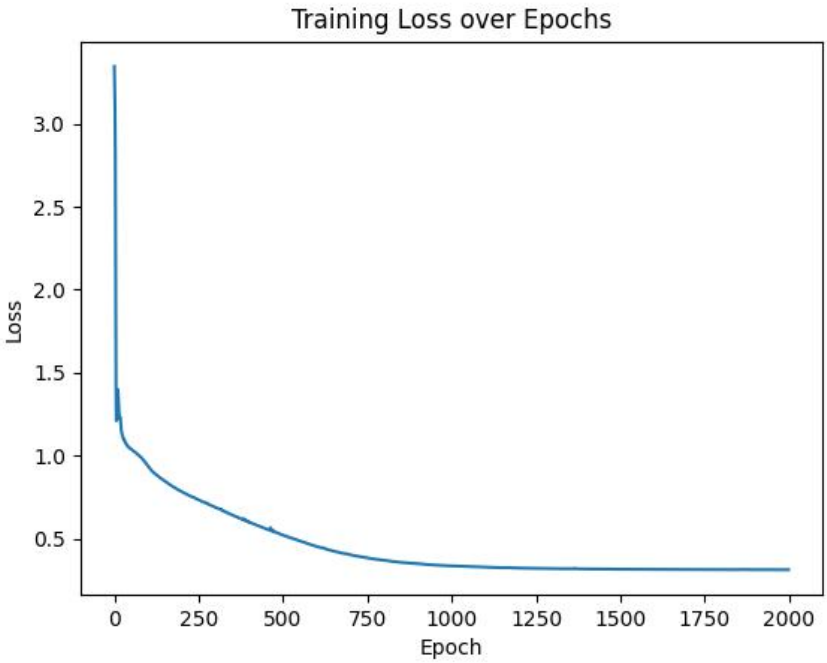

Fig.1 *Training Loss over Epochs*

It can be observed from ***Fig.1*** that the training loss decreases rapidly at first and gradually converges to approximately 0.42 after around 1300 epochs. The non-zero final loss suggests that the model has learned meaningful patterns but not a perfect mapping. It is likely due to inherent data noise or model capacity limits. Non-zero training loss prevents over-fitting. Thus, convergence to a non-zero value reflects a balance between fitting the training data and maintaining generalization.

In the context of a character-level text generation LSTM, a final training loss of approximately 0.42 is generally reasonable and expected because the model learns to predict the next character from a distribution over all possible characters, not a single deterministic output. Even a perfect model will have some loss if it spreads probabilities across multiple plausible next characters. Also at the character level, there are many contexts where multiple next characters are possible for instance, in English, after "th", both "e" and "a" might follow. This means some level of loss is inevitable. Therefore, converging to 0.42 means the LSTM has likely learned useful patterns.

2.2 Name Generation

In this part of the assignment the quality of the LSTM model is tested by generating names and assessing the quality.

The name generation process begins by feeding an initial letter into a trained LSTM model as a one-hot encoded vector. The LSTM processes this input and outputs a probability distribution over the next possible characters. Instead of always selecting the most likely next character, new hyper-parameter called temperature is used.

The output logits from the model are divided by a temperature value before applying the softmax function. This controls the resulting probability distribution:

Low temperature (<1): Sharpens the distribution, making high-probability choices even more likely and reducing randomness.

High temperature (>1): Flattens the distribution, increasing the chances of selecting less likely characters and producing more diverse and creative names.

After adjusting with temperature, we sample the next character from the modified probability distribution. This character is appended to the name, and the process repeats, feeding the updated sequence back into the model. The process terminates either when the "EON" token is generated or when a predefined maximum name length is reached.

To prevent names from ending too early, the model penalizes the probability of "EON" during early time steps.

2.3 Qualitative Testing

This part includes generated names that starts with "a" and "x" with different hyper-parameter values in order to test and increase the performance of the generative model. Here, it is important to note that, the distribution of the initials are not uniformly distributed, shown in ***Fig.2***. This results for a lesser represented initials such as "x" or "q" the algorithm may generate repetitive results or nonsense.

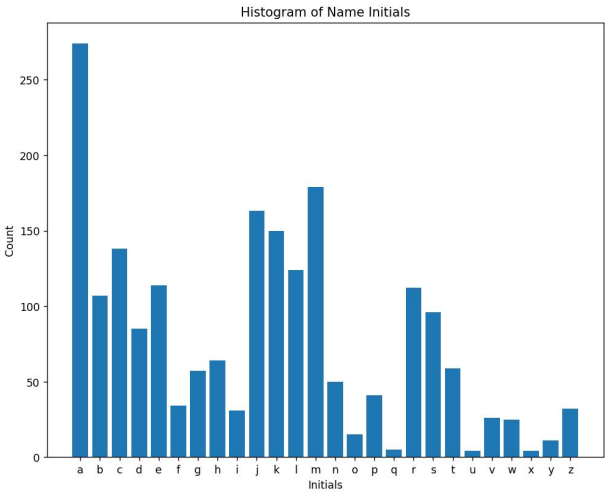

Fig.2 *Distribution of name initials*

Initially, with the distribution issue in mind, it was decided that in order to prevent over-fitting and increase variety, small number of epochs should be used.

Some generated names with their corresponding hyper-parameters:

Epochs = 100, Temperature = 1:

```
Enter initial letter (or 'quit' to exit): a
['alle', 'ane', 'are', 'asle', 'afe', 'asa', 'aora', 'ane', 'aon', 'aor', 'ane', 'aa', 'a', 'ai', 'asao', 'aia
', 'ahi', 'ar', 'ayr', 'aic']
Enter initial letter (or 'quit' to exit): x
['xtsn', 'xvre', 'xan', 'xuyy', 'xja', 'xien', 'xilea', 'xnei', 'xelie', 'xva', 'xiry', 'xnnan', 'xlat', 'xabo
', 'xfoe', 'xsot', 'xtiia', 'xprvay', 'xmae', 'xsnloyn']
```

Epochs = 300, Temperature = 1:

```
Enter initial letter (or 'quit' to exit): a
['ara', 'ari', 'aro', 'ay', 'adan', 'aphia', 'ale', 'an', 'adee', 'alia', 'aisa', 'ad', 'ax', 'ar', 'as', 'ahh
', 'avia', 'amor', 'ath', 'arla']
Enter initial letter (or 'quit' to exit): x
['xina', 'xaniea', 'x', 'xisa', 'xes', 'xona', 'xom', 'xon', 'xi', 'xandre', 'xon', 'xomia', 'xina', 'xovia',
'xon', 'xon', 'xtso', 'xmsum', 'xlento', 'xamdra']
```

However, the resulting names was not satisfactory with small number of epochs. The number of epochs is increased until reaching convergence.

Epochs = 1000, Temperature =1

```
Enter initial letter (or 'quit' to exit): a
['aro', 'alsr', 'amle', 'auss', 'adhi', 'anne', 'amon', 'alha', 'auan', 'ale', 'azle', 'ana', 'aro', 'abol', '
ahm', 'ana', 'ambel', 'aro', 'ale', 'anne']
Enter initial letter (or 'quit' to exit): x
['xzee', 'xan', 'xzee', 'xan', 'xan', 'xzee', 'xie', 'xzee', 'xan', 'xife', 'xav', 'xan', 'xave', 'xine', 'xie
', 'xie', 'xan', 'xzee', 'xan', 'xav']
```

Epochs = 1000, Temperature =0.8

```
Enter initial letter (or 'quit' to exit): a
['ahta', 'ana', 'ali', 'ana', 'aro', 'aro', 'amon', 'ann', 'ara', 'ayah', 'ara', 'ash', 'adha', 'al', 'ann',
ans', 'alia', 'ach', 'amin', 'avah']
Enter initial letter (or 'quit' to exit): x
['xav', 'xzee', 'xan', 'xzee', 'xav', 'xan', 'xan', 'xie', 'xae', 'xav', 'xav', 'xzee', 'xlee', 'xan', 'xan',
'xzee', 'xzee', 'xan', 'xzee', 'xave']
```

Epochs = 1000, Temperature =2.5

```
Enter initial letter (or 'quit' to exit): a
['arlde', 'adhta', 'amon', 'ash', 'adha', 'amra', 'avia', 'aore', 'ana', 'adby', 'azme
', 'adha', 'ara', 'azey', 'ana', 'aamhah', 'alla', 'abrl', 'abal', 'alme']
Enter initial letter (or 'quit' to exit): x
['xan', 'x', 'xave', 'xieg', 'xme', 'xne', 'xtey', 'xaw', 'xips', 'xan', 'xae', 'xape'
, 'xaes', 'xi', 'xvery', 'xmen', 'xie', 'xvee', 'xmen', 'xle']
```

Number of epochs increased more to 2000.

Epochs = 2000, Temperature =1

```
Enter initial letter (or 'quit' to exit): a
['aus', 'athn', 'ash', 'aus', 'alie', 'ali', 'ash', 'alie', 'aah', 'aph', 'amin', 'alie'
, 'alia', 'ana', 'ara', 'ara', 'avan', 'amin', 'adh', 'ana']
Enter initial letter (or 'quit' to exit): x
['xavy', 'xive', 'xze', 'xza', 'xa', 'xzin', 'xive', 'xa', 'xav', 'xza', 'xze', 'xan', '
xive', 'xiv', 'xa', 'xa', 'xze', 'xa', 'xza', 'xive']
```

Epochs = 2000, Temperature =1.5

```
Enter initial letter (or 'quit' to exit): a
['anha', 'ali', 'abla', 'alie', 'aza', 'abia', 'avan', 'ara', 'anh', 'aron', 'ach', 'a
is', 'an', 'aza', 'alia', 'aus', 'adhi', 'ahhn', 'avin', 'anh']
Enter initial letter (or 'quit' to exit): x
['xa', 'xiv', 'xze', 'xa', 'xive', 'xive', 'xive', 'xibe', 'xiv', 'xzo', 'xza', 'xive'
, 'xive', 'xin', 'xa', 'xab', 'xive', 'xiv', 'xle', 'xza']
```

Epochs = 2000, Temperature =10

```
Enter initial letter (or 'quit' to exit): a
['aymhhfkhr', 'apnnfmirwkf', 'aqgmr', 'aiof', 'auhbqi', 'azlzr', 'ahpriphhdor', 'a', 'ag
asina', 'adevhonmzob', 'ahe', 'ah', 'ach', 'aenjiyn', 'azhkiavbwh', 'amdrplhaz', 'al', '
alokjiamo', 'aserabop', 'aug']
Enter initial letter (or 'quit' to exit): x
['xaqaqthy', 'x', 'xnifa', 'xczpxygnpdb', 'xqeit', 'xi', 'xlstnurnwaj', 'xw', 'xdzirlpzd
sk', 'xxzlvh', 'xkfrbza', 'xsbpgaeb', 'xk', 'xekisplwfnn', 'xaxiiityhw', 'xpp', 'xh', 'x
bw', 'xko', 'xzapgier']
```

Here, up to now, it can be observed that with increased temperature the generated names are random and not realistic, whereas, for decreased temperature names that are generated were realistic but repetitive and short.

In order to get more realistic and longer names beam search algorithm was applied. Beam search is a decoding algorithm used in text generation that keeps the top k most probable sequences called the "beam width" at each step instead of just the single most probable one. This allows it to explore multiple paths in parallel, increasing the chances of finding a better overall sequence compared to greedy search.

Beam search with Temperature = 1.5 and beam width = 20:

```
Enter initial letter (or 'quit' to exit): a
['ariel', 'azalea', 'ariel', 'alexis', 'amari', 'angel', 'ariel', 'aubri', 'amari', 'amari', 'andrea', 'azariah',
 'ariel', 'angel', 'amira', 'azariah', 'ariel', 'azariah', 'amirah', 'amari']
Enter initial letter (or 'quit' to exit): x
['xzavier', 'ximena', 'xzavier', 'ximena', 'xzavier', 'xzavier', 'xzavier', 'xzavier', 'xzavier', 'xzavier', 'xza
vier', 'ximena', 'xzavier', 'ximena', 'xzavier', 'xzavier', 'xzavier', 'xzavier', 'xzavier', 'xzavier']
```

Beam search resulted in repetitive and deterministic names especially for the lesser represented letter “x”. Therefore, for this specific task, this approach seemed not applicable. Therefore, beam search is omitted.

Returned to initial name generation technique. However, names were still too short. In order to tackle this issue, penalization to probability of “EON” during early time steps is applied. This is done simply by manually subtracting 10 from the logit corresponding to the “EON” character. This significantly reduces the probability of selecting “EON”.

2.4 Final decision

Epochs = 2000, Temperature =2.5 with EON penalty

```
Enter initial letter (or 'quit' to exit): a
['ayuh', 'aythrr', 'arupi', 'amoe', 'aniwara', 'abquio', 'aytha', 'awuso', 'aceo', 'ayah', 'aponarnn', 'ama',
anba', 'aplor', 'ara', 'abiana', 'aly', 'axa', 'aminon', 'ama']
Enter initial letter (or 'quit' to exit): x
['xbel', 'ximele', 'xivenov', 'xzor', 'xinae', 'xanton', 'xavzor', 'xiviarooror', 'xanzbrr', 'xaiza', 'xineero'
, 'xjuspziro', 'xavhrqobrro', 'xrillepz', 'xiviner', 'xzier', 'xovarorro', 'xanzbrh', 'xuslaz', 'xzier']
```

Finally, the generated names were realistic, diverse and sufficiently long concluding the assignment with success. I wish I were named Xovarorro!

3. Conclusion

In summary, the project successfully developed, trained, and optimized its character-level LSTM-based person name generation model. Preprocessing the dataset, adjusting the values of hyperparameters like temperature and epochs, and trying techniques like beam search and EON token penalty helped the model find an equilibrium in realism and diversity in generated names. Although beam search added some constraints, leaving the fine-tuned approach in its original sampling mode worked better. The optimized model showed its capacity for producing coherent, diverse names, establishing the success of the training method and the model structure in sequential generation of names.