Report 2

- Description of the measurement strategies;
- Statistical Analysis (similar to our in-class work);
- Visualization of the data. You need to describe your number visually and have one separate page for the visual storytelling.

Brief background

The character-level language modeling system being developed aims to generate text by predicting the next character based on the preceding sequence of characters. By training on a any collection of names, the system learns the patterns and dependencies within the characters to generate plausible and contextually relevant text. The system's performance and effectiveness are evaluated using quantitative metrics such as cross entropy and negative log likelihood, which measure the dissimilarity between the predicted probability distribution and the true distribution of the target characters. Lower values of these metrics indicate better accuracy and alignment with the underlying patterns in the data. Additionally, qualitative evaluation methods can be employed, involving human assessment of the generated text's coherence, fluency, and relevance. By combining quantitative and qualitative evaluation, the system's ability to generate meaningful and coherent text can be comprehensively assessed, ensuring its effectiveness and utility in practical applications.

Measurement of the system

Cross entropy is a commonly used loss function and evaluation metric in machine learning, particularly in language modeling tasks. It quantifies the dissimilarity between the predicted probability distribution generated by the model and the true distribution of the target data.

In the context of character-level language modeling, cross entropy measures the average number of bits needed to encode the true characters using the predicted probability distribution. It provides an estimation of how well the model captures the underlying patterns and probability distribution of the target data.

The cross entropy loss is computed by comparing the predicted probability distribution, obtained through the model's output, with the true probability distribution of the target characters. This comparison involves taking the logarithm of the predicted probabilities and multiplying them by the corresponding true probabilities. The resulting values are summed across all characters and averaged over the dataset.

Negative log likelihood (NLL) is closely related to cross entropy and is often used as a synonymous term. In fact, cross entropy is equivalent to the average negative log likelihood. By taking the negative logarithm of the predicted probabilities, the cross entropy loss ensures that the model is penalized more for larger prediction errors and less for smaller errors.

In the evaluation of character-level language models, the negative log likelihood (or cross entropy) is used as the objective function during model training. The goal is to minimize the cross entropy loss by adjusting the model's parameters through techniques like gradient descent or more advanced methods such as ADAM.

Moreover, the negative log likelihood is also used as an evaluation metric during model testing and validation. It provides a quantitative measure of how well the model is able to predict the true characters, with lower values indicating better performance.

By utilizing negative log likelihood (cross entropy) as an evaluation metric, we can assess the accuracy and quality of the character-level language models in capturing the underlying patterns and probability distributions of the target data. It helps to determine the effectiveness of the models in generating meaningful and coherent text, as well as guide the optimization of the models through training iterations.

The figures below illustrate the loss function during the optimization process of the MLP model. Since the model is trained on mini-batches, the left picture shows a bumpy loss function curve. This variability in the loss function arises because the model's performance can vary between individual mini-batches. In some cases, the model may perform exceptionally well, while in others, it may perform poorly. To address this issue, the second figure demonstrates a smoother loss function curve achieved by taking the average over a certain interval. By averaging the loss function values over this interval, the random fluctuations caused by individual mini-batches are mitigated, resulting in a more stable representation of the overall performance of the model.

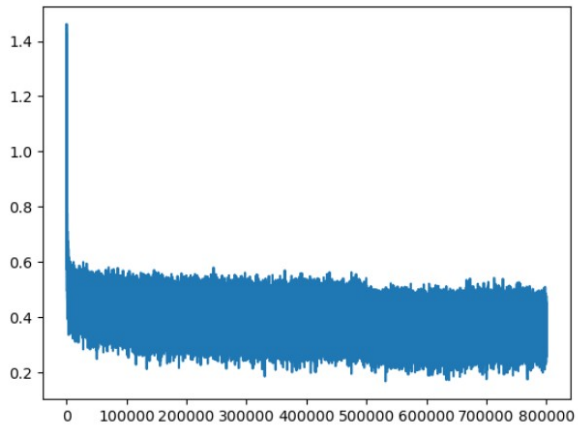


Figure 1: Loss function on mini-bathces.

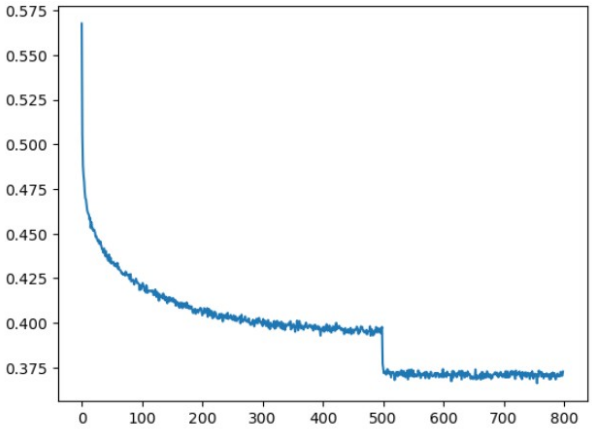


Figure 2: Average loss on each 1000 iteration.

The sharp decrease observed in the plot can be attributed to the learning rate decay. **Learning rate decay** is a technique commonly used in optimization algorithms to gradually reduce the learning rate over time. By applying learning rate decay, the initial learning rate is set higher to allow for faster convergence in the early stages of training. As the training progresses, the learning rate is gradually reduced, enabling the model to make smaller adjustments and fine-tune its parameters. The sharp decrease in the loss function indicates a significant adjustment in the model's parameters, which often occurs when the learning rate is reduced. This adjustment helps the model converge to a more optimal solution by taking smaller steps towards the minimum of the loss function.

In the depicted scenario, the learning rate is initially set to 0.1 for the first 500,000 iterations. This higher learning rate allows for faster progress in the initial stages of training when the model's parameters are far from optimal. After the initial 500,000 iterations, the learning rate is decayed to 0.01 for about 300,000 iterations. This reduction in the learning rate helps stabilize the training process and enables the model to converge to a better solution by making smaller adjustments to the parameters.

Statistical Analysis and Visualization

The statistics below are derived from a filtered dataset obtained from an initial dataset comprising 13 million records. The filtering process involved applying various criteria based on languages or domains to obtain a refined subset of data for analysis.

- Count: There are a total of 91,073 words in the dataset.
- Mean: The average length of the words is approximately 8.16. This value represents the central tendency of word lengths in the dataset.
- Standard Deviation (Std): The standard deviation of 2.62 indicates the variability or dispersion of word lengths around the mean. This number can be considered as relatively normal which indicates that there is some variation in the word lengths within the dataset, but it is not excessively high.
- Minimum (Min): The shortest word in the dataset has a length of 3 characters.
- 25th Percentile (Q1): 25% of the words in the dataset have a length of 6 characters or less. This is the lower quartile.
- Median (50th Percentile or Q2): The median word length is 8 characters. This means that 50% of the words in the dataset have a length of 8 characters or less.
- 75th Percentile (Q3): 75% of the words in the dataset have a length of 10 characters or less. This is the upper quartile.
- Maximum (Max): The longest word in the dataset has a length of 16 characters.

The distribution and boxplot of the word length are illustrated below. It is obvious that there are no outliers in the dataset.

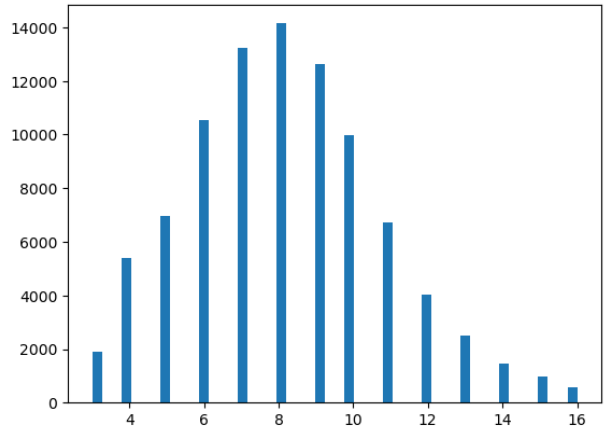


Figure 4: Distribution of the words length in the cleaned and filtered dataset.

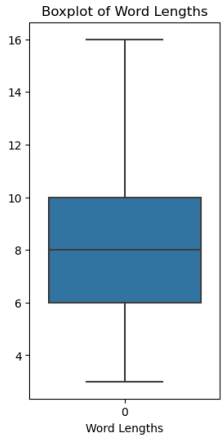


Figure 3: Boxplot of the words' length.

Another useful way to visualize the data is using count-based n-grams. In the below figure bigram counts are depicted. Note that as the context length increases the matrix grows exponentially and huge portion of it becomes sparse which is

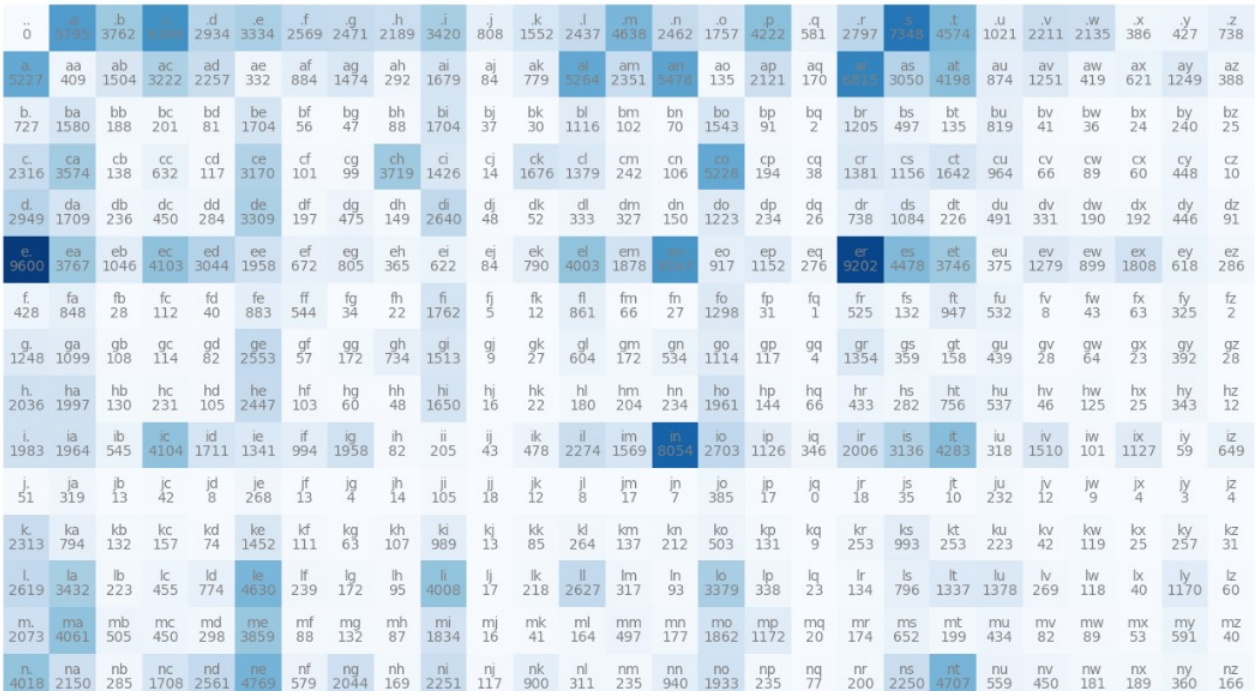


Figure 5: Snippet of bigram counts of the words.

Below are the 5 most and least likely bigrams that occurs in the dataset (Note that '.' is the end token):

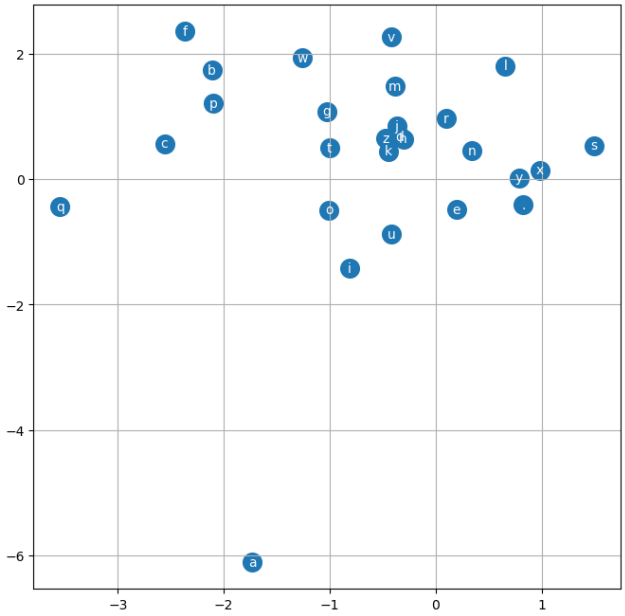
Most likely:

- Character 's' followed by '.' (12,608 times)
- Character 'e' followed by '.' (11,974 times)
- Character 'e' followed by character 'r' (11,442 times)
- Character 'i' followed by character 'n' (10,039 times)
- Character 't' followed by character 'e' (9,394 times)

Least likely:

- Character 'q' followed by character 'z' (4 times)
- Character 'f' followed by character 'z' (4 times)
- Character 'q' followed by character 'j' (3 times)
- Character 'f' followed by character 'q' (2 times)
- Character 'q' followed by character 'k' (2 times)

Another significant method to visualize and observe is using trained model features. In the MLP model there is feature vector that embeds all the character to the n-dimensional space. Typically I experimented the model using 10 and 20 dimensional space which gives best result. But in this case it is not possible to represent this vector. Even thought it is not optimal I trained the model using 2 dimensional embedding just for the visualization purposes.



We can observe that even with very small space model is able to learn and cluster the characters. It is obvious that letter ‘a’ and ‘q’ has special place in the space. Letters ‘i’, ‘z’, ‘k’, ‘d’, ‘h’ are clustered very closely. Letters ‘f’, ‘p’, ‘b’, ‘c’ are placed a bit closely and far from other group of letters. This clustering is dependent on the domain of the dataset. For instance, I tested the same training and visualization on the English person names and almost all the vowels were clustered closely in the space.

Sampling from the model

Samples from the models to evaluate the models quantitative and qualitative. Note the improvement over loss and from generated samples. Right now more complex architectures such as RNN and Wavenet are in progress which is expected to give better results.

	Untrained model	**Bigram count-based model**	**Trigram count-based model**	**Trigram neural-net based model**	**MLP Model**
Spec		27 by 27 count matrix	729 by 729 count matrix	729 by 27 weight matrix with only one linear layer	10 dimensional feature vector, 200 number of hidden neurons, 3 block size, 11,897 total parameters
Loss	3.784	2.725	2.496	2.496	2.363
Sample 1	xaxljywzqyuuarun faiupngcjom	paruis	tics	llitekra	rid
Sample 2	uonnwjekiwouhly dx	joa	nutelamic	merchr	forcend
Sample 3	ywxxeklvr	ftrtx	prel	trettravers	welluma
Sample 4	uzpzcqohmccy	ts	tovil	alspep	cloudson
Sample 5	gvrvlccdvixprjb	halloum	reelesto	agica	rantown