Assignment 2

T-725-MALV Natural Language Processing

Andrea Terenziani
andrea23@ru.is

1 Project description

1.1 Abstract

Large-scale models like GPT-3 and GPT-4 have made headlines; however, understanding the nuances of training and generating from smaller models can provide insights into the workings of generative language models. In this project, the goal was to create three small character-level GPT models using Karpathy's nanoGPT repository on a corpus of text. Evaluate the quality and creativity of the generated text under different parameter settings.

The chosen corpus was Dante Alighieri's Divine Comedy, one of the foundational poems of Italian literature. It was chosen for two main reasons:

1. It is a considerably large text, for a total of more than 500 thousand words;
2. It has a clear and intentional structure in its verses and rhymes, since Dante chose to write (almost) only verses of eleven syllables and with a distinct "ABA-BCB-CDC-DED-..." rhyming scheme.

1.2 Measured metrics

1.2.1 Syllable count

The consistent structure of the poem actually lead to one of the goals of the project, which was to compare the syllable count distributions in the original and generated verses, trying to improve the similarity between them as the model became more complex. For each sample text, such similarity was measured using:

- Jensen-Shannon divergence
- Cross-entropy

Each sentence, both original and generated, had to be split into syllables. For this, I've decided to use the `Pyphen` package, which provides methods to hyphenate text in multiple language, including Italian. It should be noted that this was chosen also for the sake of time, since it is not fully accurate for every word. However, it did quickly provide a somewhat robust and "independent" way to count the syllables in each verse.

The syllable count distribution for the original text, as measured with Pyphen, is given by figure 1.

1.2.2 Vocabulary overlap

The other analyzed metric was the "overlap" between the vocabulary of the original text and that of each generated sample. While this could have been measured with the Jaccard coefficient, such metric has the downside of not accounting for a large difference in size between the two sets, which is what would happen between the vocabulary of a 500k words text and that from a text of just a few hundred words. As an alternative, I've decided to simply measure which fraction of the generated vocabulary is present in the original counterpart.

In all cases, the vocabulary was generated using the `nltk` library: the input text is first converted into a list of tokens, with `word_tokenize`, and then such list is converted into a frequency distribution, with `FreqDist`. The keys of this distribution are then used as the text's vocabulary.

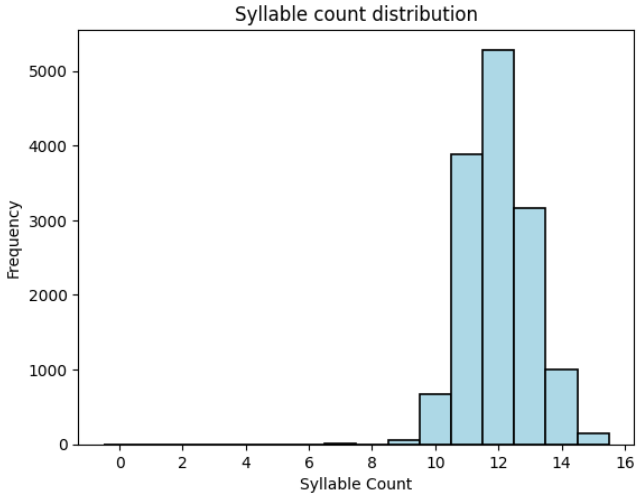

Figure 1: Distribution of the n° of syllables in the original text

`eval_iters`	`block_size`	`batch_size`	`lr_decay_iters`	`max_iters`
20	64	12	2000	2000
–	128	–	–	4000
–	–	–	–	–
–	256	–	–	–
–	512	–	–	6000

Table 1: Training hyperparameters

`n_layers`	`n_head`	`n_embd`	`dropout`
4	4	128	0.0
4	4	128	–
–	–	–	–
12	12	768	–
12	12	768	–

Table 2: Model structure hyperparameters

2 Model setup and training

2.1 Hyperparameters

The project consisted of 5 different nanoGPT models, increasing in complexity and length of training. The hyperparameters used for training and for structuring each model are listed in tables 1 and 2, respectively, where each "–" indicates the use of the nanoGPT default value for that parameter. Notably, the first model was the "CPU-friendly" version described in the nanoGPT documentation, used to have a lower bound of the possible result. On the other hand, the third model simply used the default values for all parameters.

The general design choice was to increase both the "intensity" of the training and the complexity of the model, especially the `block_size` parameter, which controls the length of the text sent to the model as input at each training iteration, and the dimensionality of the embedding used (as `n_embd`).

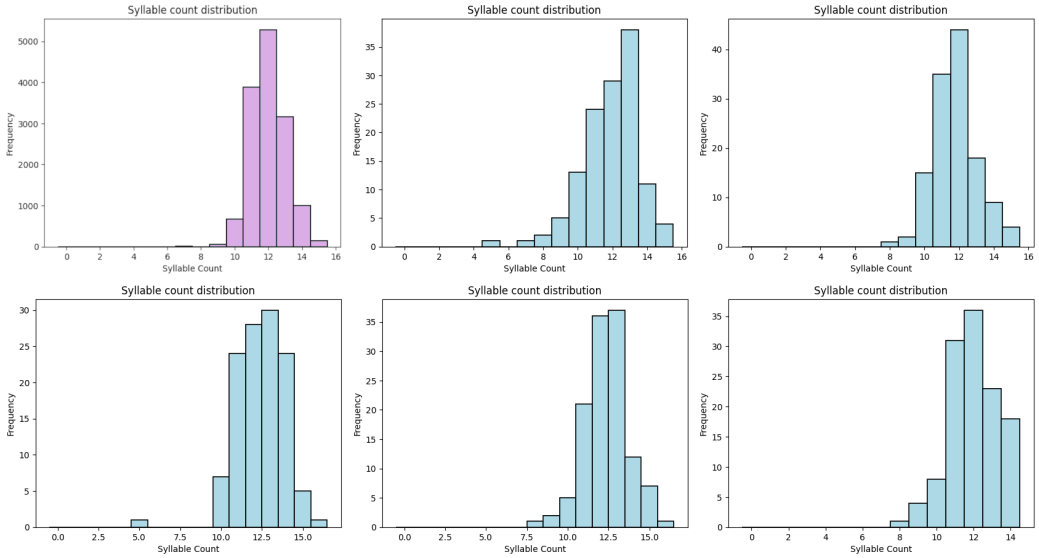

Figure 2: Syllable count across models (in blue) and in the original text (in pink). Each model's plot is shown in order from top-center to bottom-right

2.2 Results

2.2.1 Metrics

Table 3 contains the values of the metrics listed in section 1.2 and, additionally, the final loss values for the training and validation sets.

The two loss metrics remain consistent in the first two models but start to diverge notably in the third, with the last model having a training loss two orders of magnitude lower than the validation loss, a sign of overfitting.

As can be seen in figure 2 and in table 3 through the values for the Jensen-Shannon distance, the syllable count distribution remained lackluster at best, with the third model having the best result - however, all models managed to get close to a distribution centered between 12 and 13 syllables, though not as sharp as it should have been.

The third model also excelled in the vocabulary overlap metric, for which all models, except the first, obtained satisfactory values. Since a GPT is a kind of NLP model tailored for generating likely words given an initial text, instead of detecting poetic or lyrical structures, it would make sense that this metric be easier to satisfy.

2.2.2 Sample outputs

It should be noted that each model produced an output text much longer than shown here, which had to be cut for the sake of readability.

The first model already produces text that definitely has the "feeling" of reading a 14th century italian poem, including with some abbreviations of words typical of poetry. However, the lexicon is, aside from very short stopwords and few exceptions, completely chaotic and incoherent. The use of a very small block size of 64 meant that this model processed excessively short sequences from the original text.

> *prita e le pirtele di futti a spionta gugguida ch'ardi primentri non a l'accornonde e gia' intene, come con e' quanto de l'un cost'alti la monto 'l di trarde la soltra di sottra, e sora' guava in bor tante a bensi rarpende la prove chi conda grasto, che a di tra sove suo veder li parso, qui su' qualui e in coma volger tu sollo". E e che 'l partuo m'anda mani qualti pien lo stu cal besta e pien rave". Poli a ciad lui aletro granza in'onfignio, ben a me stre ond'alte forti*

Loss (train)	Loss (val)	Cross-entropy	Jensen-Shannon	Vocab. overlap
1.7751	1.7753	1.685	29.784%	37.546%
1.4479	1.4821	1.596	22.623%	72.627%
0.1481	2.2943	**1.755**	**30.575%**	**87.049%**
0.1028	2.7737	1.617	24.388%	80.687%
0.0499	3.505	1.608	24.226%	78.252%

Table 3: Performance across metrics

With the third model we already see a much improved result, which much more real words and sentences that remain somewhat coherent for much longer. Note, in the following sample, the word “micino”. This is a real italian word, meaning "little kitty", that was not present in the entirety of the original poem. Though it was almost surely a lucky coincidence, the model seems to have been able to generate a completely novel yet completely real word.

> *Vidi come gelato ancora e atteso, per lo spirito parole o fanno spazia la gente che tua giustizia a questo". Ond'io l'una e l'altra parola fiata la gente che si' tornar l'occhio di fuor li fiamma venian di quella governa improcesta". Sempre a te mano a* ***micino*** *a la cima ch'io dissi: "Maestro, chi hai ti sarien di cui quando tu chiamato? e chi e' chi son li altri tocchi? Ed ecco lo grazia li altri del petto: quando di rispondia che 'l ciel tuo diserratto si mori' e con lo 'ntelletto patrone.*

The final two models produce text that, to an italian reader, seems comparable to that of the third model - the following sample was produced by the fifth GPT. Once again, however coincidental it may be, we find a word that is both real and not present in the input text, in this case the verb “poetava” (meaning "to compose poetry") in the first line.

> *Poi* ***poetava*** *com'al duca intorna la cagion d'un gia' d'un cosi' l'ombra che facea di famma in la mo di cui la speranza cortesi e l'altro; la giustizia, come sarebbe valle le nel loco dopo li angeli aperti, e quel suo primo 'l mondo di Santa seguitar si per la mente permuta. Quivi la non passi di Cammina, non anche il giorno e poi e l'altra la fede n e' prima che suole, meno una madria.*

3 Conclusion

This project could be expanded upon firstly by performing a more structured exploration of the hyperparameters, both by trying to vary more of them and by trying more values for each.

Secondly, it could be interesting to find a way to quantify how well the model follows the rhyming scheme of a given poem or poems. A basic approach would be to look at the final syllables of each word, but this does not consistently apply to every poem or language. Furthermore, the counting of syllables is not a trivial, nor language independent, task in and of itself.