Exploratory Data Analysis (EDA) of Tiny Stories dataset

Ahmadreza Bagherzadeh

ahmadrezabaqerzadeh@gmail.com

ABSTRACT

This report presents my exploratory data analysis (EDA) of the Tiny Stories dataset, a collection of short, simple narratives designed for training and evaluating language models. I downloaded this dataset from Ronen Eldan's Hugging Face repository. and conducted a thorough examination of its key characteristics. My analysis focuses on story length distribution, word frequency patterns, token diversity, and common syntactic structures.

Through this EDA, I demonstrate how the dataset's intentionally simple vocabulary and controlled narrative style make it particularly suitable for few-shot learning experiments and lightweight language model pretraining. I highlight the dataset's unique advantages for educational NLP applications and low-resource model development scenarios. The findings provide valuable insights for researchers considering this dataset for their natural language processing projects.

EXPLORATORY DATA ANALYSIS FINDINGS

The dataset contains stories that are short with very simple and organized structures. Below are two examples of these stories:

story1:

One day, a little girl named Lily found a needle in her room. She knew it was difficult to play with it because it was sharp. Lily wanted to share the needle with her mom, so she could sew a button on her shirt.
Lily went to her mom and said, "Mom, I found this needle. Can you share it with me and sew my shirt?" Her mom smiled and said, "Yes, Lily, we can share the needle and fix your shirt."
Together, they shared the needle and sewed the button on Lily's shirt. It was not difficult for them because they were sharing and helping each other. After they finished, Lily thanked her mom for sharing the needle and fixing her shirt. They both felt happy because they had shared and worked together.

story2:

The dataset has two sections: train and validation. The train section contains 2.119719 million stories, and the validation section contains 21,990 short stories. My analysis focuses on the train dataset, and by analyzing the training data, I aim to gain a comprehensive view of the entire dataset.

With an initial analysis, we can see that each story consists of an average of 897 character-level tokens and 207 word-level tokens.

If we consider each character as a token in sentences, the train dataset consists of 1.9 billion tokens, all of which include 174 unique tokens. The space token, with 364,233,176 repetitions, constitutes 19.4% of the entire data. This means if we want to continue a sentence with one token, there is a 19.4% probability that the next token will be a space. Below is a table of the top 10 most frequent tokens/characters:

token/character	frequency
space	364,233,176
e	180,782,399
a	127,963,378
t	118,625,067
o	99,438,049
h	93,158,669
n	88,723,326
i	81,541,616
d	81,365,133
s	78,717,020

Among all tokens, 7 tokens appear only once, and 167 tokens appear more than once. Below is a table of these tokens:

token/character	frequency
		1
Ñ	1
f	1
î	1
®	1
µ	1
Ò	1

The average token repetition count is 1.093154 million times, with 27 tokens exceeding this average and 147 tokens falling below it. Below is a table of the 10 least frequent tokens:

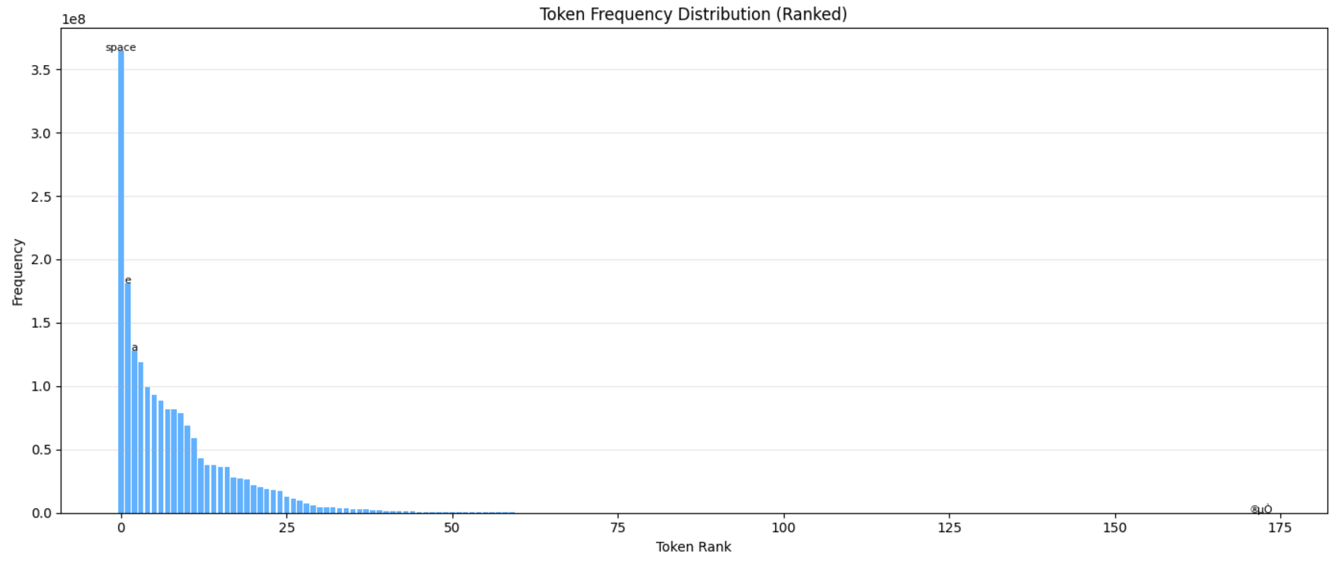
token/character	frequency
š	1
@	1
É	1
		1
Ñ	1
f	1
î	1
®	1
µ	1
Ò	1

Among the least frequent tokens:

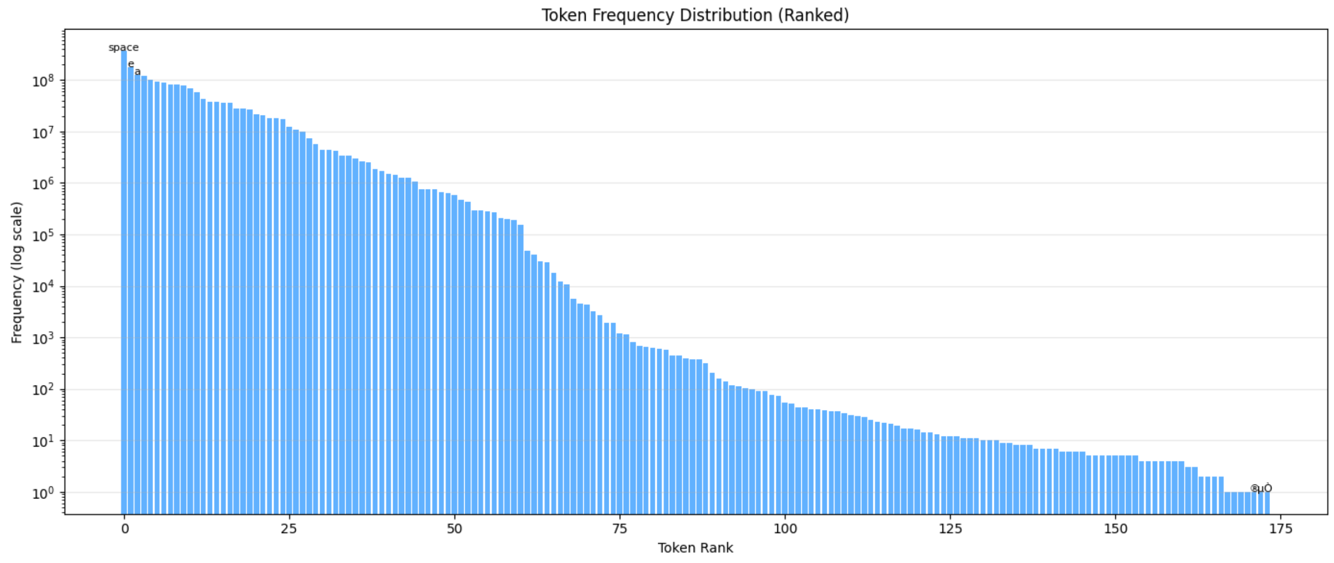
- **7 tokens appear once**
- **4 tokens appear twice**
- **2 tokens appear three times**

- **161 tokens appear more than three times**

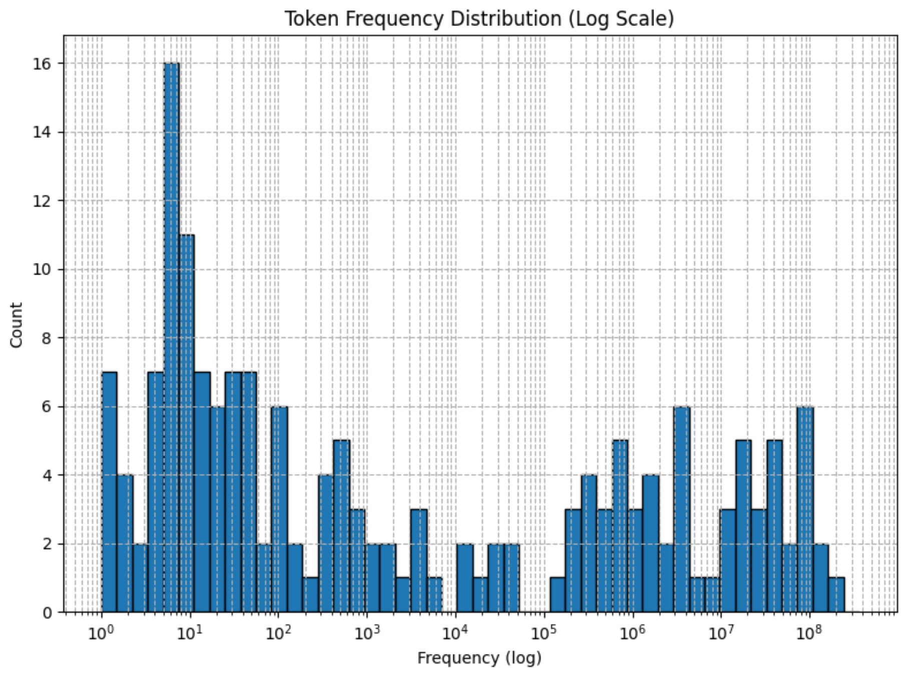
Let's look at a bar plot of the token repetition counts:



The plot highlights the top 3 (most frequent) and bottom 3 (least frequent) tokens. From the plot, we can see that approximately 25 of the most frequent tokens dominate the data. To be precise, these **25 tokens** constitute **95.1944%** of the data, and the **top 50 tokens** constitute **99.8122%**. Since the repetition counts vary significantly between characters, the plot appears somewhat unusual. Let's examine it on a logarithmic scale:



Now, by plotting the number of tokens against their frequencies, we can extract more insights:



This plot makes the earlier bar plot analysis more understandable, as we can see that very few tokens/characters with high frequencies control the data.

Now that we've examined the characters and their distribution in the train dataset, let's explore the words. If we analyze tokens at the word level, we see that the entire train dataset consists of **439.039906 million tokens**, including **63,577 unique tokens**.

At the word level, we don't have a dedicated space token because we know that a space automatically follows each token, and encoding/decoding can be done easily without needing an explicit space token. However, if we examine the impact of spaces on textual structure at the word level, the number of spaces relative to tokens is significant—spaces constitute 45.34% of word-level text.

But as mentioned earlier, since word-level language models don't use space tokens, they don't affect the model's output.

At the word level, the analysis differs slightly. For example:

19,834 words appear only once, constituting 0.004% of the data (an extremely small percentage that can practically be ignored).

These tokens, however, make up 31.19% of unique tokens.

Implication: In word-level language models, you could eliminate these 31.19% of vocabulary (vocab) and train the model with fewer parameters.

The average repetition count for these tokens is 6,905, meaning tokens are repeated 6,905 times on average. Since 31.19% of tokens appear only once, the average of 6,905 is quite high for all tokens. This indicates that the data at the word level is also controlled by a small number of tokens.

Here's where it gets interesting:

60,847 tokens appear less frequently than the average.

We started with 63,577 tokens, but only 2,730 tokens push the average higher.

This means **4.29% of unique tokens** heavily influence the average, while the remaining **95.71%** have minimal impact.

The word-level token with the highest repetition count is **«.»** (period), appearing **36.459483 million times**. Despite the high count, this isn't unusual because this token plays a critical role in sentences across all languages.

In percentage terms:

8.3044% of sentences in this dataset consist of the «.» token.

On average, there are 8 «.» tokens per 100 tokens, or roughly 1 period every 12 tokens.

Below is a table of the top 10 most frequent tokens:

token/character	frequency
.	36,459,483
the	20,239,799
and	18,112,895
,	17,359,980
to	12,648,384
a	11,475,893
was	9,436,658
he	7,977,807
she	7,710,551
it	7,104,640

The repetition counts are very high, and these **10 tokens constitute 33.8297% of the data**.

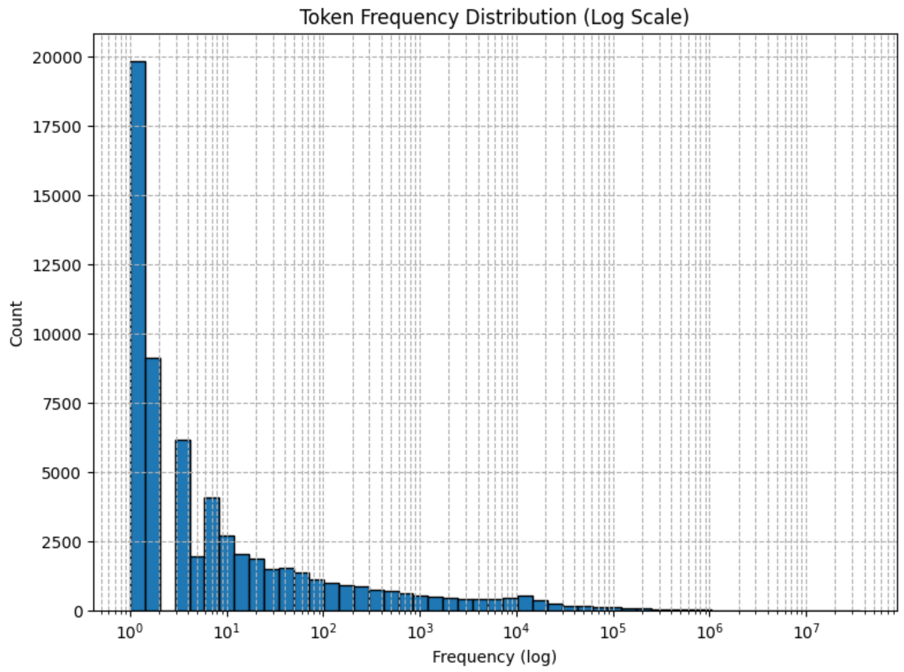
Let’s expand this to more tokens:

Top 20 tokens: 42.6826% of data

Top 50 tokens: 55.6748% of data

50.5558% of the data consists of just 35 tokens (half the data is built from 35 tokens, while the other half uses 63,542 tokens).

Let's visualize the data distribution by plotting the number of tokens against their frequencies:



From the plot, we can see that high-frequency tokens are few, but their impact on the dataset is enormous. If we were to plot a box plot of these tokens, the high-frequency data would appear as outliers—yet in reality, they have far more influence than tokens within the box.

The two methods discussed above (character-level and word-level tokenization) each have pros and cons, suggesting that a hybrid approach is optimal.

Word-Level Tokenization:

+ **Semantic Preservation**
  – Maintains meaningful word units (e.g., "cat" ≠ "c", "a", "t")
+ **Efficient for Common Words**
  – Frequent words get single tokens (good for short sequences)
+ **Pre-trained Model Compatibility**

  - Works well with traditional NLP models (Word2Vec, GloVe)

– **Vocabulary Bloat**

  - Large vocabularies (~50k–100k tokens) for diverse languages

– **Out-of-Vocabulary (OOV) Problem**

  - Fails on unseen words (e.g., “antiestablishmentarianism”)

– **Morphology Ignorance**

  - Treats “run” and “running” as unrelated tokens

Character-Level Tokenization:

+ **Tiny Vocabulary**

  - Only ~26 letters + symbols (e.g., English: <100 tokens)

+ **Handles Any Word**

  - No OOV issues (can construct any word from characters)

+ **Morphological Awareness**

  - Learns subword patterns (e.g., “-ing” suffix)

– **Longer Sequences**

  - “cat” $\longrightarrow$ 3 tokens $\longrightarrow$ higher computational cost

– **Weak Semantic Signals**

  - Harder to learn word meanings from characters alone

- – **Context Fragmentation**
  - Relationships between distant characters are hard to model

Hybrid Approach (BPE/WordPiece):

Modern systems (like BERT, GPT) use subword tokenization (BPE/WordPiece) to balance these tradeoffs:

- + **Pros of Both**
  - Medium vocab size (e.g., 30k tokens)
  - Handles OOV via subwords (“unhappiness” → “un”, “happiness”)

Solution: Byte Pair Encoding (BPE)

Combines the strengths of both methods.

I trained this algorithm 5 times on the train dataset with different vocab sizes:

20k, 10k, 5k, 1k, and 500 tokens.

Evaluation Metric:

Examine the last 20 tokens of each tokenizer:

If they’re semantically weak → Too character-like (model will struggle with meaning).

If they’re semantically strong → Too word-like (vocab becomes bloated).

Note: The last 20 tokens help determine a balance. If they’re moderately meaningful, they’re suitable for model training.

Below is the list of the last 20 tokens for tokenizers of different vocab sizes:

Tokenizer with different vocab size:

vocab size	20k	10k	5k	1k	500
	Ġglancing	ĠCarol	Ġsneezed	Ġde	Ġmu
	Ġlists	Ġsnuggling	Ġsport	able	ook
	Ġuncles	Ġplumber	Ġwizard	Ġimportant	ach
	ffled	Ġrestoring	ĠEach	Ġremember	Ġagain
	Ġdisliked	Ġwinners	Ġupstairs	Ġfish	Ġhome
	Ġamusing	Ġpounding	Ġthrilled	llow	Ġwhen
	ĠRy	ĠStanley	Ġprovide	Ġsound	Ġgood
	ĠRiya	elve	Ġbrain	Ġslide	hat
	Ġbaffled	Ġswoop	Ġawe	Ġus	ep
	ĠDoc	Ġfeathered	Ġdiscuss	maz	Ġwho
	ĠDovey	Ġorganised	Jake	Ġreplied	est
	sterious	Ġtaps	Ġpanic	Ġamaz	ried
	shade	Ġfashionable	Ġwarning	Ġac	Ġfound
	Ġshopped	Ġstinky	Ġtwisted	lease	Ġfl
	Ġsquirting	Ġgorgeous	Ġglowed	Ġwork	Ġthen
	ĠKiss	Ġrider	Ġuniverse	Ġwatch	Ġch
	clear	Ġdeserves	Ġfiref	Ġrain	Ġdec
	Ġwelled	ĠPlay	Ġtherm	Ġshowed	pped
	Ġcauses	Ġmacaroni	Ġvacation	Ġrabbit	Ġwal
	ĠOxy	ĠNell	Ġworse	Ġokay	as