Final project in the course **Deep learning, Methods and applications**

at Umeå University year 2025

Author: **Thomas Selin**

Source code: **github.com/Thomas-Selin/language-model**

Model files: **Available on request** (size 369 mb each)

Introduction

During this project I developed a language model using a decoder only transformer model.

The goal was to develop a model that could hopefully answer simple single questions regarding the topics that were in the datasets, mainly the dataset with context to the questions in the question-answer dataset.

The model was trained in two steps, first base-training, also known as pre-training, with a few general English text datasets. The second step was fine-tuning with a dataset containing question-answer pairs, also in English.

The code that was developed for training used as a starting point the code from Andrej Karpathy's nano-gpt lectures but most parts of that code was modified and many other features were implemented, for example a more capable tokenizer.

The model was evaluated, mainly looking at if it showed signs of having learned patterns in the English language.

Limitations

Limited hardware resources and limited time constrained me to use a small model architecture. The choice of training hyperparameters was also restricted because of limited hardware and time.

These restrictions lowered the expected results, and the result was expected to be nowhere close to what we come to expect from current large language models.

Method and materials

The datasets that were used for base-training was the public datasets Fineweb Edu (v1.4.0 2025-26), the Cosmopedia v2 subset of SmolLM-Corpus, TinyTexts, and the context data for the question-answer dataset.

The question-answer dataset that was used for finetuning with question-answer pairs was the SQuAD v1 dataset which contains 87 600 question-answer pairs with context.

The order I fed the dataset to the training process was to some extent shuffled, especially at the later stages of training. In the early part of base training I used only the Fineweb Edu dataset and at the later stages of base training I interleaved all the base training datasets.

The method that was used for creating the model was to develop a flexible training process with ability to adjust learning parameters during the training process and the ability to mix datasets during the training process. A lot of work went into creating an easy to use, memory efficient and stable data pipeline and training process.

The training was done on a NVidia A30 GPU with 24Gb video memory.

My optimizer was AdamW and my loss function was Cross-Entropy Loss.

My starting point training hyper-parameters were:

- Batch size: 64
- Block size: 512
- Max epochs: 100
- Eval interval: 5
- Learning rate: 0.0003
- Eval iters: 100
- Embedding dimension: 768
- Number of heads: 12
- Number of layers: 8
- Dropout: 0.1
- Early stopping patience: 40

The main adjustments during training was increasing batch size, lowering learning rate during fine tuning, lowering dropout during fine tuning. Also, I used max epochs = 8 during fine tuning with early stopping patience at 2 epochs.

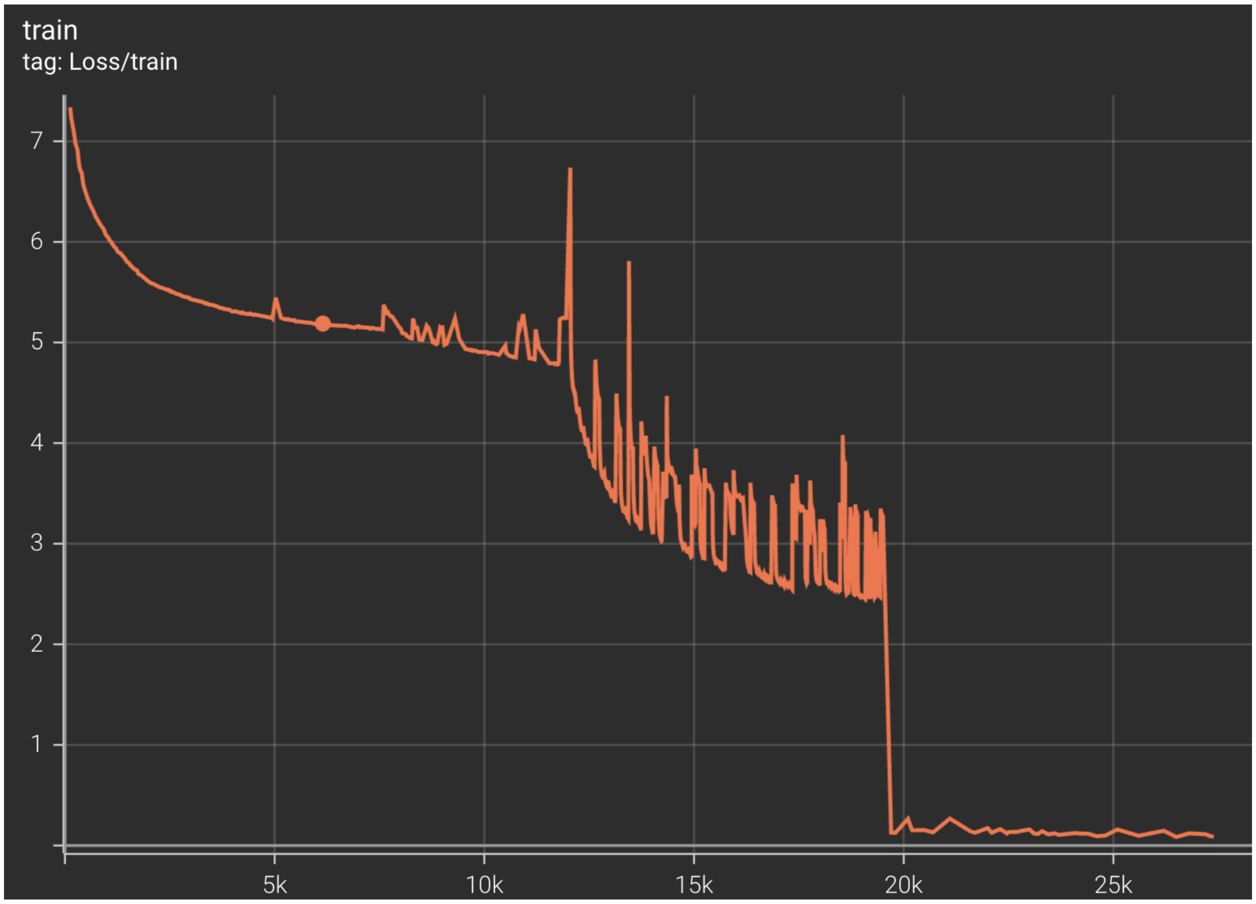
I evaluated the loss every 5th step.

A subword ByteLevel BPE (Byte Pair Encoding) tokenizer was developed. It was used for encoding and decoding text to/from token id:s. It was also used to create the vocabulary from sample text in the dataset. The size of the generated vocabulary was 12856.

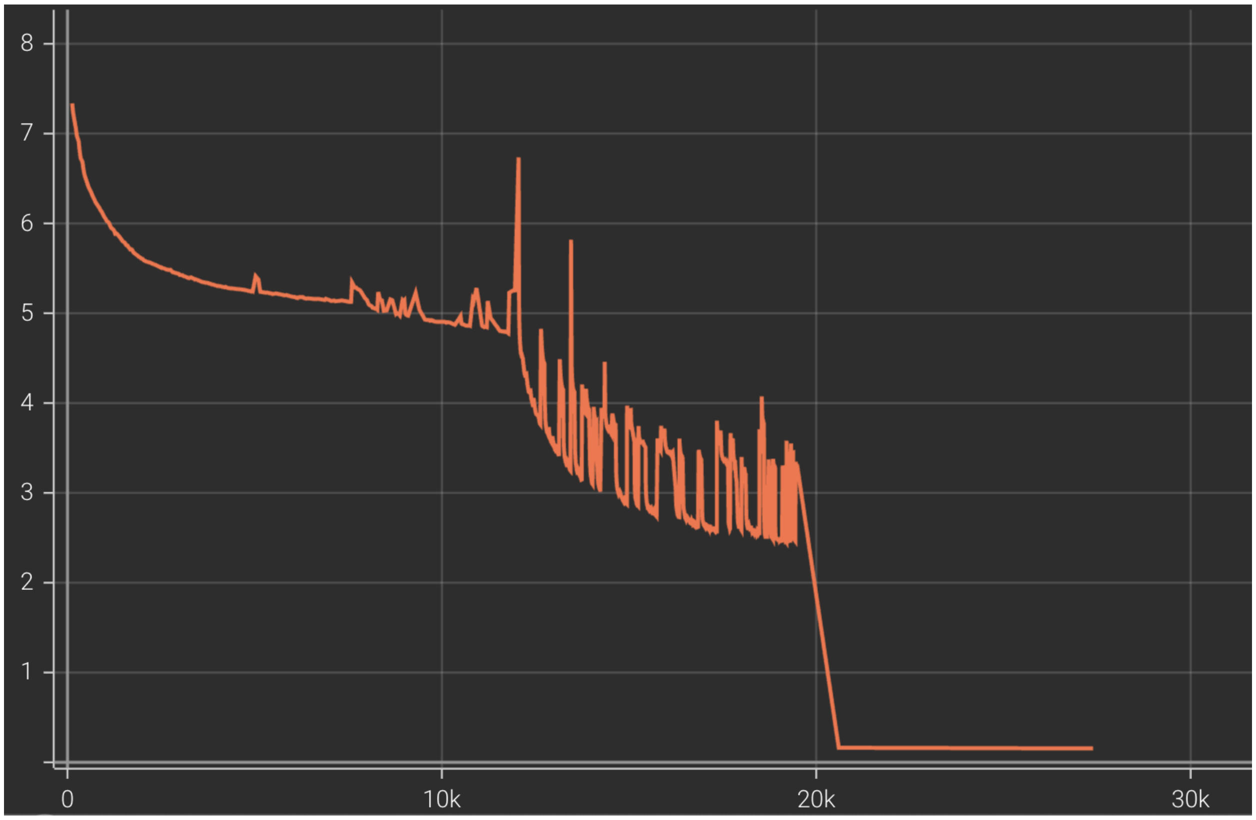
Results

The final model has 67 million parameters, and a file size of 369 megabytes in pytorch (.pt) format. The created model exhibits the capability to answer questions both on the topics that were in the context of the question-answer pairs and also to some extent other general questions. I used TensorBoard for tracking and visualizing the training process:

Training loss plot:



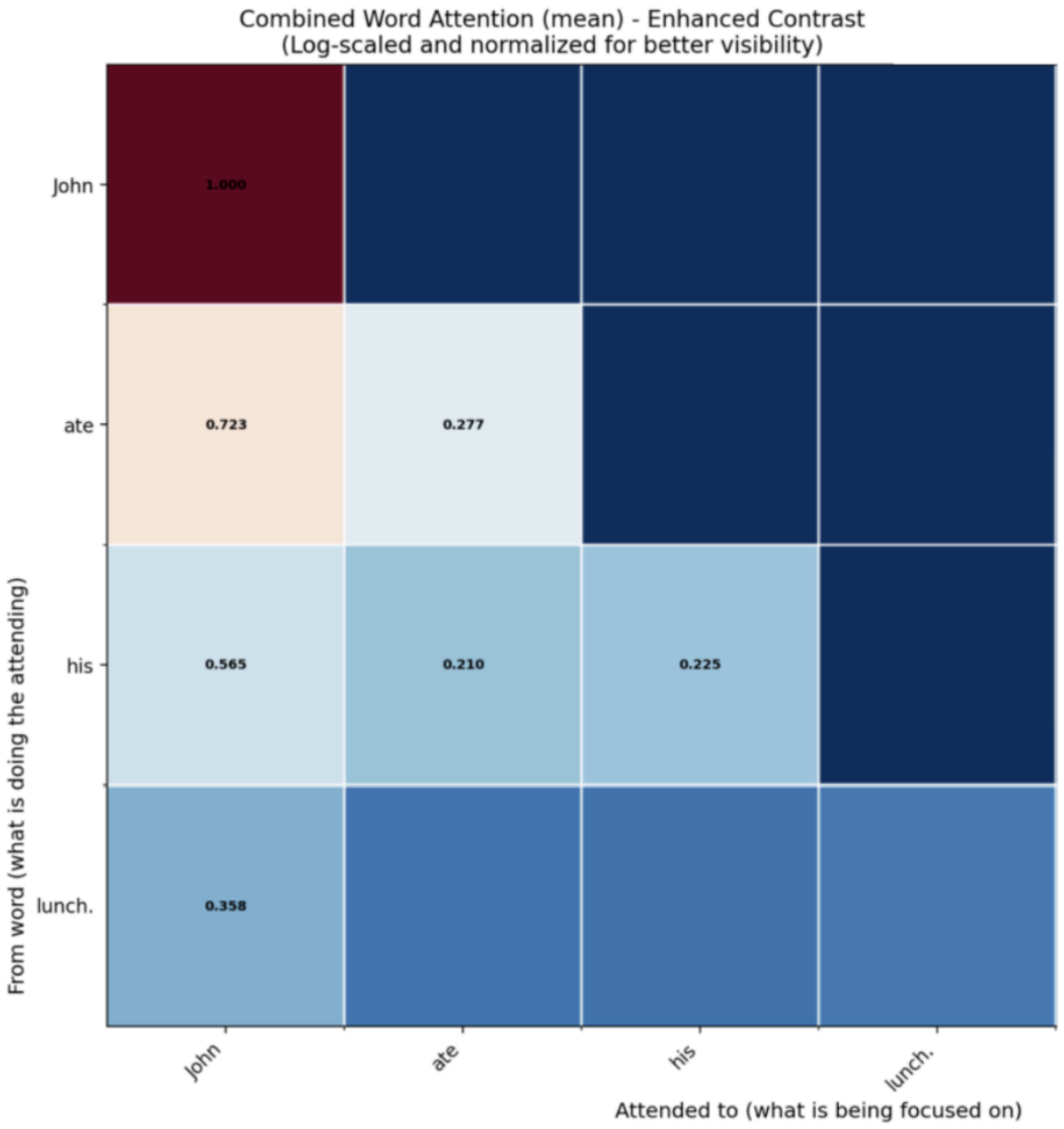
Validation loss plot:



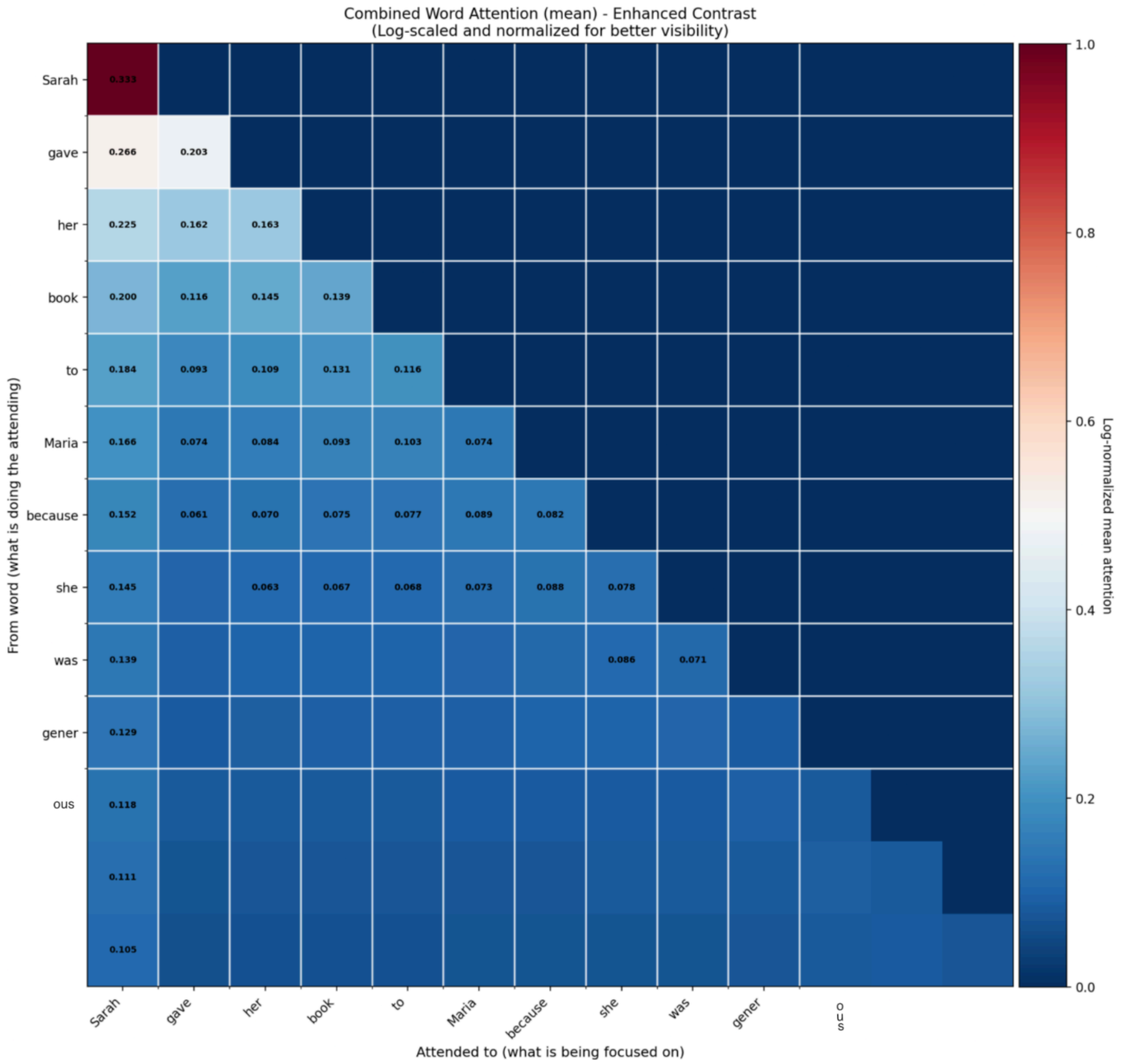
The loss plots didn’t show significant signs of overfitting. Initially I made the mistake of dimensioning the model for a larger vocabulary than was actually used. The second drop in the loss curve (corresponding to an increased rate of loss decline) was at the point when I resized the to the correct vocabulary size and increased batch_size to improve training speed and performance.

The third drop in loss decline rate was at the point when I started fine tuning.

Early stopping was sometimes (for certain files in the datasets) reached during base training but it was not reached during fine tuning.



In the above image you can see that the model pays relatively high attention (mean value of all attention heads) to the word “John” in relation to the words “his” which is correct as that word refers to “John” in this sentence.



In the above image you can see that the model pays relatively high attention (mean value of all attention heads) to the word “Sarah” in relation to the words “her” and “she” which is correct as both those words refer to “Sarah” in this sentence.

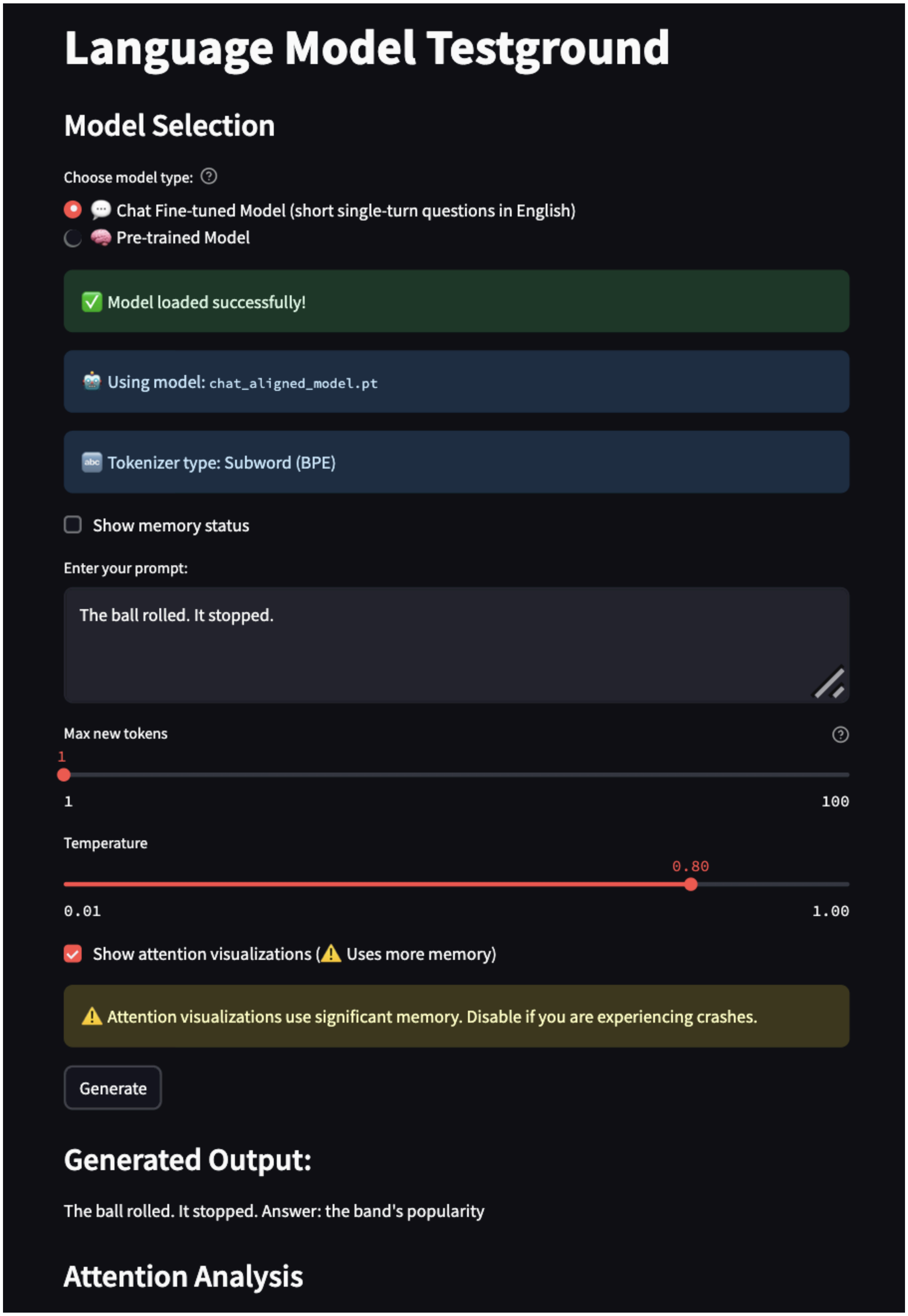
The model struggled with multi-sentence attention, for example "The ball rolled. It stopped." where a good result would be high attention for the word “it” to the word “ball”.

Typical output

Input: *"Which Emmy award was American Idol nominated for nine times?"*
Output: *"Answer: Golden Globe Award"*

Comment: This is factually incorrect but the answer format is good and the answer has to do with the subject. This was a common problem. I believe it would have been easier for the model to learn facts if it had more parameters. It would probably also have helped if I increased the number of training epochs for the question-answer data set.

I developed a web user interface to inference the model with adjustable settings:



Language Model Testground

Model Selection

Choose model type:

● 💬 Chat Fine-tuned Model (short single-turn questions in English)
○ 🧠 Pre-trained Model

✅ Model loaded successfully!

🤖 Using model: `chat_aligned_model.pt`

🔤 Tokenizer type: Subword (BPE)

☐ Show memory status

Enter your prompt:

The ball rolled. It stopped.

Max new tokens: 1 (range 1 – 100)

Temperature: 0.80 (range 0.01 – 1.00)

☑ Show attention visualizations (⚠️ Uses more memory)

⚠️ Attention visualizations use significant memory. Disable if you are experiencing crashes.

Generated Output:

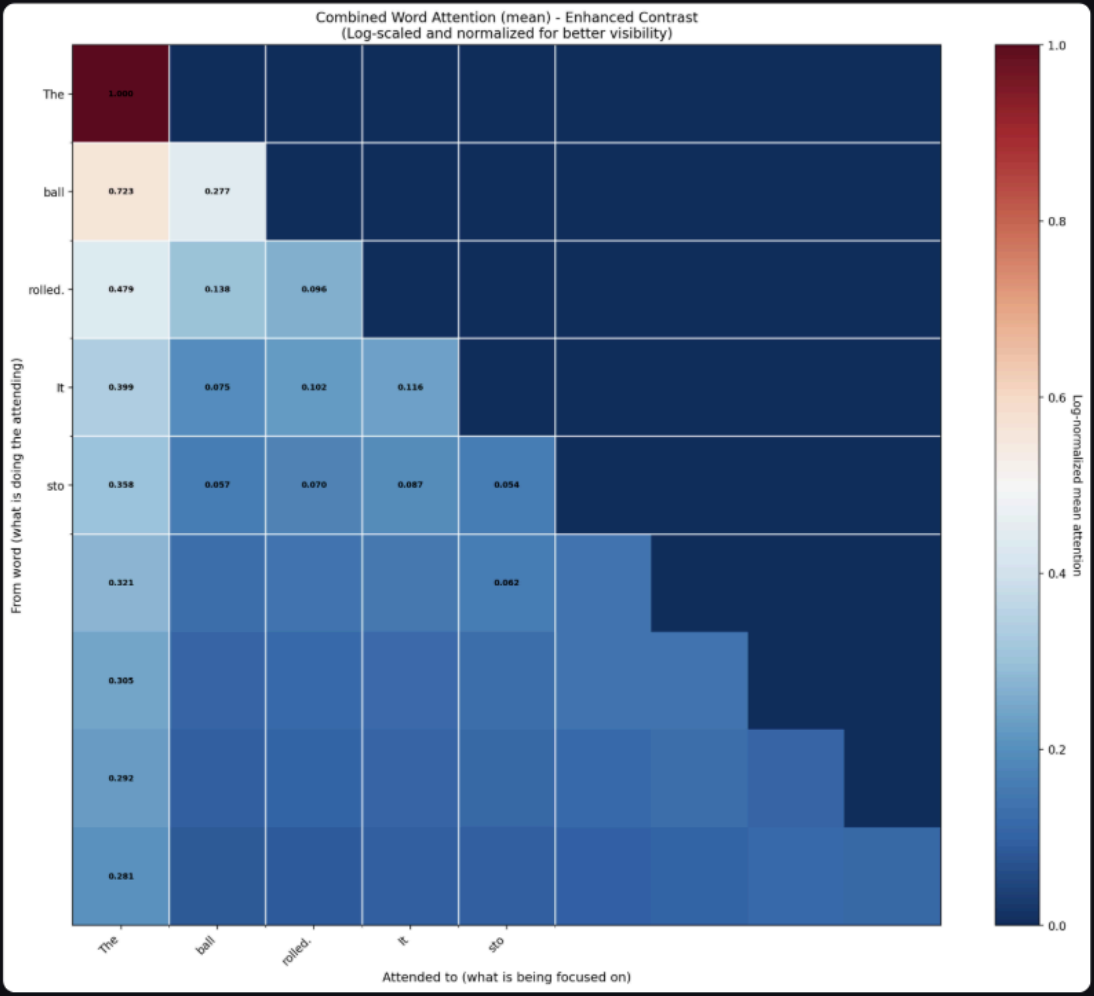
The ball rolled. It stopped. Answer: the band's popularity

Attention Analysis

Attention Analysis

Input Focus | Word Relationships

Shows which words refer to or relate to other words (e.g., 'it' → 'bear' in the sentence "The bear ate the cake because it was hungry")



Skipped layer-specific visualization to conserve memory

Memory after generation: 79.8% used

Clean Memory

Discussion

The, by today's standards, very small language model shows signs of having learned important patterns in the English language. It shows signs of having learned seemingly correct attention between words.

There are many choices that can be made when training this type of model and it would be good to explore more of these choices and the impact on the performance, for example different architecture or dataset.

Overall, it was a project that I learned a lot from and it demonstrates the ability of this small model to learn patterns in language, such as topic, to start with big letters, to end with punctuation and attention between words.