Computional Linguistic Task 1 - report

Jerzy Boksa

October 2025

1 Introduction

The goal of this assignment is to design, implement, and evaluate a small-scale language model using two architectures:

A Recurrent Neural Network (RNN) variant (e.g., LSTM, GRU) and a Transformer-based model must both be trained for the causal language modeling task (prediction of the next token).

All LSTM and Transformer architectures were implemented from scratch, following the guidance and tutorials. No pre-built implementations or libraries were directly used.

1.1 Dataset and tokenizer

The dataset used in this assignment is the `wolne_lektury_corpus` from Speakleash, which contains 6,619 polish books, poems, and texts. The dataset was preprocessed to remove unnecessary content, such as the author's name or publication date at the beginning of the poems. After preprocessing, all documents were parsed into a single tensor containing 64,369,661 tokens.

For tokenization, the pre-trained tokenizer `dkleczek/bert-base-polish-uncased-v1` was used, which has a vocabulary size of 60,000.

The token tensor was then split into two disjoint datasets: 90% of the tokens were allocated to the training dataset (`train_dataset`), and 10% to the evaluation dataset (`eval_dataset`).

During data loading, sequences of length `seq_len` = 128 tokens were created. This results in $(64369661 \cdot 0.9) :$ `seq_len` $= 452599$ sequences in the training dataset and 50288 in the evaluation dataset. For efficiency, a maximum of 196.000 sequences was used from the training dataset, and 1.960 sequences from the evaluation dataset.

The implementation of data preparation can be found partially at the beginning of `playground.ipynb`, while the data loading routines are implemented in `training.py`.

1.2 Hardware

Both training experiments were conducted on the RunPod platform using an NVIDIA A40 GPU with 48 GB of VRAM and 9 vCPUs. This configuration provided sufficient memory and computational power to efficiently train both the LSTM and Transformer models on the full dataset. Almost 100% of gpu utilization was used during training.

2 Models

As mentioned earlier, both models were implemented from scratch based on tutorials, papers, and blog posts. Additionally, the model parameters were carefully selected so that the total number of trainable parameters in both models is approximately similar.

2.1 Text Generation

It is important to note that the next token is not chosen as the most likely token (i.e., `argmax`), but rather sampled according to the predicted probability distribution. This introduces stochasticity into the generation process and allows the models to produce more diverse and natural sequences.

2.2 Lstm

The implementation of the LSTM model was based on [1] and [2]. Additionally, a dropout layer was introduced between LSTM layers to improve regularization. For each time step, the hidden states are updated sequentially across layers, and dropout is applied to the outputs of all but the last layer. Furthermore, Layer Normalization (LayerNorm) is applied before returning the final logits. The full implementation can be found in the file `lstm.py`.

2.2.1 Parameters

The LSTM model was configured with the following parameters:

- Number of layers: 8
- Embedding dimension: 552
- Hidden size: 552

This configuration results in a total of 85,819,824 trainable parameters.

2.3 Transformer

The Transformer implementation is a decoder-only architecture. It was developed based on the tutorials in [3] and [4]. One key difference from the original implementation is that the normalization layers are applied before the transformation, as recommended by Karpathy [3]. Implementation can be found in the file `transformer.py`.

2.3.1 Parameters

The Transformer model was configured with the following parameters:

- Number of attention heads per layer: 8
- Embedding dimension: 512
- Number of Transformer blocks (layers): 8

This configuration results in a multi-head attention mechanism with 8 heads per layer across 8 layers. The total number of trainable parameters for this Transformer architecture is approximately 86,773,344.

3 Training

Both models were trained using the same training loop, dataset, and hyperparameters to ensure a fair comparison. The loss function employed for training was the categorical cross-entropy, which is standard for next-token prediction tasks in language modeling.

The main training functions and utilities are implemented in `training.py`, while their usage can be found in `train_lstm.py` and `train_transformer.py`.

Note that the parameters used in these files may not reflect the actual training configuration, as the scripts were tested on smaller setups using a CPU. The real parameters are described in the following subsection.

3.1 Reducing LR on Plateu

During training, the *ReduceLROnPlateau* technique was employed to dynamically adjust the learning rate. This method monitors the evaluation loss and reduces the learning rate if no improvement is observed over a specified number of epochs (or training intervals). In our case, the learning rate was decreased by a factor of 0.70 if the evaluation loss did not improve for 2 consecutive epochs.

The main purpose of this approach is to prevent overfitting and help the model converge more smoothly, particularly in later stages of training. A minimum learning rate of 5×10^{-7} was enforced, ensuring that the learning rate never becomes too small to allow further training progress.

More formally, the parameters can be interpreted as follows:

- **factor=0.70**: the learning rate is multiplied by 0.70 when the condition is triggered.
- **patience=2**: the scheduler waits for 2 consecutive epochs without improvement before reducing the learning rate.
- **min_lr**=5×10^{-7}: the learning rate is not allowed to decrease below this threshold.

This strategy was particularly helpful during real-time training, allowing the learning rate to adjust dynamically as training progressed, rather than being reduced only at fixed epochs.

3.2 Parameters

Apart from the learning rate scheduling using *ReduceLROnPlateau*, all other training parameters were identical for both models. Both models were trained for approximately 10 hours with the following settings:

- Sequence length: 128 tokens
- Start Learning rate: 1×10^{-4}
- Batch size: 196

Given the size of the training dataset, these settings correspond to 1,000 training batches per epoch. For evaluation, 100 batches were processed per epoch.

3.3 Lstm

The LSTM model, after a training period of 10 hours, achieved a training loss of 4.7683 and an evaluation loss of 5.1676. Throughout the training process, the learning rate remained constant at its initial value and was not decreased. Within this time frame, the model completed 30 full epochs.

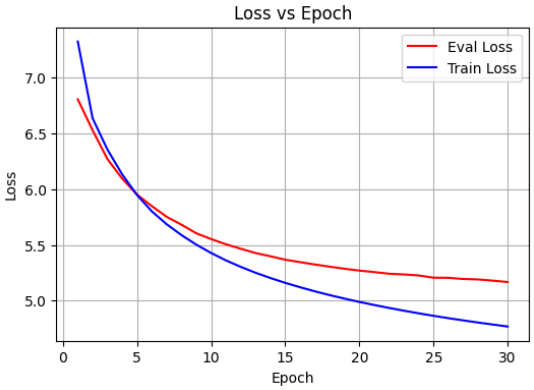


Figure 1: LSTM Learning rate history

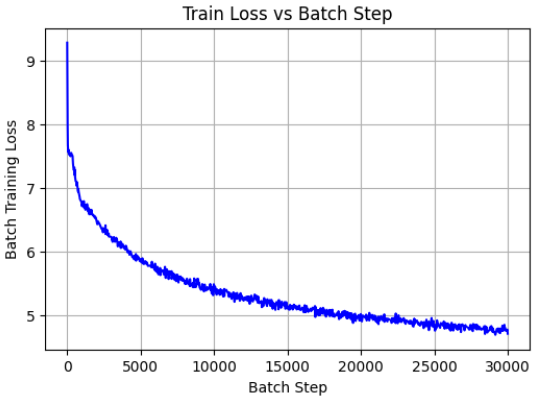


Figure 2: LSTM Batch Loss

3.4 Transformer

The Transformer model, after 10 hours of training, achieved an evaluation loss of 4.8825 and a training loss of 3.9965. The learning rate reached its minimum value of 3.4×10^{-5} at the 48th epoch. Over the course of training, the model completed a total of 50 full epochs on the hardware configuration described earlier.

As shown in the plots below, there is no overfitting and the evaluation loss has started to plateau. Moreover, the learning rate began to decrease after the 40th epoch, which corresponds to approximately 80% of the training process, indicating that the model achieved a very good result.

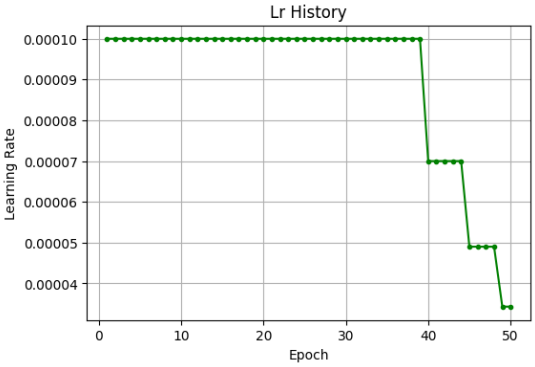


Figure 3: Transformer Learning rate history

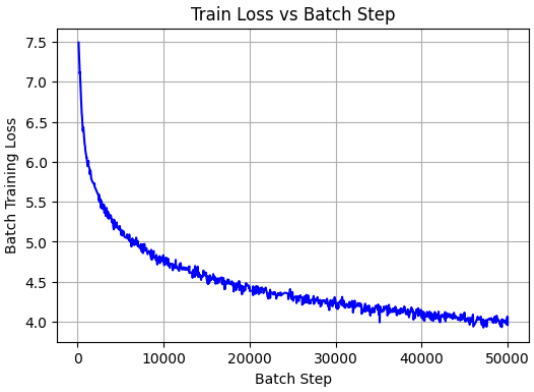


Figure 4: Transformer Batch Loss

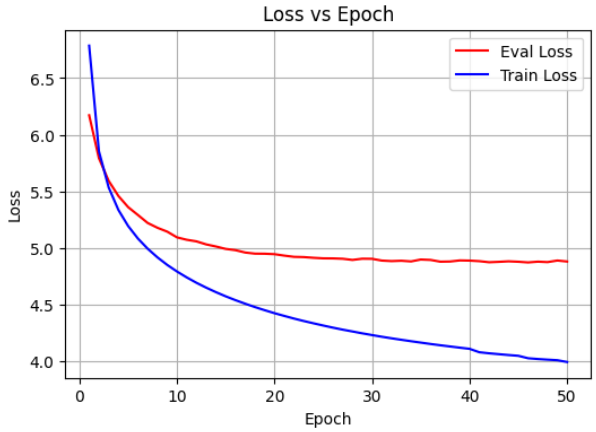


Figure 5: Loss per Epoch

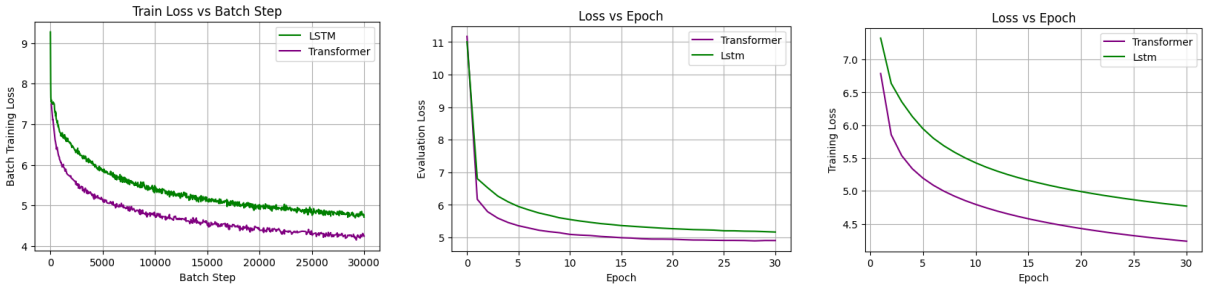
3.5 Comparission

Looking at the training metrics, the Transformer model clearly outperformed the LSTM. It achieved both lower training and evaluation losses, indicating better generalization and overall model quality.

Model	**Training Loss**	**Evaluation Loss**
LSTM	4.7683	5.1676
Transformer	3.9965	4.8825

Table 1: Comparison of training and evaluation losses for LSTM and Transformer models.

The plots below illustrate the training progress. As the LSTM model completed fewer steps and batches, the X-axis has been restricted to its maximum number of steps and batches for a fair comparison.



(a) Batch Loss (b) Evaluation Loss (c) Training Loss

Figure 6: Comparison of LSTM and Transformer performance across different loss metrics.

Based on the plots, the Transformer clearly outperformed the LSTM. It not only achieved lower training and evaluation losses, but also converged much faster. Expecteations are met.

4 Inference

The way models work can be analyzed through two types of evaluation. First, perplexity measures how surprised the model is when encountering a given token. Second, the model's overall performance can be assessed on specific text generation tasks, reflecting its general understanding and fluency.

Inference was tested locally on my machine - on cpu.

4.1 Perplexity

Perplexity is a measure of how surprised a model is when predicting the next token in a sequence. The evaluation procedure is as follows:

- Use the validation set previously employed during training (1000 samples; 1960 were used in training).
- For each token in the validation set, calculate its perplexity and accumulate the results.
- Compute the average perplexity across the repetitions.

Similarly, the perplexity on a subset of 1000 training samples can be calculated using the same procedure.

Lower perplexity values indicate better performance according to this metric. We expect the transformer model to perform better, as it achieved a lower evaluation loss during training. For reference, perplexity will also be computed for untrained (empty) models.

Model	**Eval Perplexity**	**Train Perplexity**
Empty LSTM	60056.167969	60066.878906
Trained LSTM	180.005676	82.685699
Empty Transformer	68069.562500	68285.226562
Trained Transformer	132.591293	33.860310

Table 2: Perplexity values for different models on training and validation sets.

Both models have been successfully trained, as indicated by their significantly lower perplexity compared to the untrained (empty) models. As expected, the Transformer achieved lower perplexity, which is consistent with the training results. Interestingly, the untrained LSTM performed slightly better than the untrained Transformer, which could be attributed to differences in weight initialization.

4.2 Text Generation

To evaluate how the models perform on the text generation task, 5 prompts—excerpts from https://wolnelektury.pl/ —will be selected. The task for the models is to complete each prompt with up to 128 tokens. We expect the models to produce sentences that are partially coherent and capture the stylistic characteristics of school books.

Prompt 1: ""

Transformer:
narcy moja mosci ksiazke miałem wydac za bryłke — rzecze don kichot — ze czas widze jasnosc słonca — nie przeniknałem wszystkiego. i juz swiadekem zdjety pokazał rzeczy, ktore za chwile zginęło...

LSTM:
mikołaj i nia miał rozkaz powyrazin w ratuszu. dzien o krolom, mieszkali z wilna w powiecie i zbierali w trzecim skarbcu ; aby wojsko nie mogły sie podsycic za pomoca sprzecznosci nadsyłania sobie zycia, bo lekiem sie to jest wesołe, a dla magistratow waznego panstwo i stad kasrylewce był znacznie promos. dowiedziano sie to raz po wielu jakich i z posi ramionami odjechalismyi. byli ludzie zołnierze, badani bigary i innirodzicy, a podług szlachty zebrali sie w udziale, polacy, obiecujac debu chłopi ; przeciez

Prompt 2: "Jesteś potwornie brzydkie — rzekły dzikie kaczki"

Transformer:
jestes potwornie brzydkie — rzekły dzikie kaczki owca w westerplatona. jagus i zak pomarli, nie przyjmujac przyjscia na swoja brzydka twarz jagnieta, wilgotna, zarazem ze słodycza. czynia pod drzwiami, wiedzma wsparła sie o stopy na biodrach wysokich, falistych wargami napiersnie rdzawych lisci. zasrebrzyła sie szary mrok na las. swist posrod burzy pełen ciepłych lisci, zadzwoniły przed siebie i ocierała okropny bol. sposrod trzcin drzew runał jak nieboszczyk załosny. wtedy ojciec widzi głodne swiatłosc

LSTM:
jestes potwornie brzydkie — rzekły dzikie kaczki nadal przypomniawszy, ze graja mikro [, ze herbata były człowiekiem, dziewczynki czy akt. jak inne rytmajace swieza droga przez promienie gesty liscie tygrysy, ktorych najmniejszy blaski study -log, bezkreczowy krasny, policjant stworzył, pluszkany a szklizm! obojetny zyw wsrod ruin, ktore zerwano na ziemie, rosły nagle na słonce, groznie posypane, wiec nie miały siły, kto sie stac nie chciało. pali sie spokojnie i siada na hasło, kiedy nadchodzija niby tru

Prompt 3: "Nazajutrz w kąciku pod murem, ujrzano zmarznięte ciało dziewczynki"

Transformer:
nazajutrz w kaciku pod murem, ujrzano zmarzniete ciało dziewczynki o, moj ojcze, za kiedyz sie kominy sledzic nie moge! piesn wolnosci jest jednym dowodem tajemnicy bytu wielce nam wyprobuja. nie chce, zebys studiował tore w zwierciadłach, w czesci powiesciopisarz, w nature uzebrane, lecz by w drodze nie niesc, czułosci, dotad niestety, w nizszej sferze jedyny kres miał nasiaka skromnosc uswiecona, lecz skoro nie potrzebuje ozywiac nature człowiecza, co w piersiach, fra

LSTM:
nazajutrz w kaciku pod murem, ujrzano zmarzniete ciało dziewczynki krzysztof kamil baczynski parket deszcz snuł sie orszak, wiatr jesienny dzwiek. lasem, uderzały mozstwo swiatła, przerywane puste ziarna, posia wierzch nasycone, złociste oraz ciezkie jak lod nocne i nieznane, coraz z wiekszego wrogich skrzyp, jakby miotał sie nad wszystkie swiatłem rozowe, zołte, przenikane w powietrzu lecz wesołe przez skalne miasto to istoty, wskros nieskonczonosc, podczas mrocznej pustki ludzkiej do domu ummerowego. zwiesił

Prompt 4: "nie ufając na ślepo nowinkom i wiedząc z doświadczenia, że nie zawsze gorączka rozwoju"

Transformer:
nie ufajac na slepo nowinkom i wiedzac z doswiadczenia, ze nie zawsze goraczka rozwoju siła była czyms realniejszy od forma tego, co było marzeniem akcjonariuszy, wytrwałe juz dawno zapomniane, a inne, mniej bolesne, głebokie, niezrozumiałe, a bardziej zagadkowe, bardzo pokorne a głebokie. zatajenie nie przenikało nas tak bardzo, ze nieskonczonosc oczow trwała mi fala za ton sie oddala. miałem tam tak senne widziec oczy bez przerwy nad soba, pod pobladła figura. zatrzymałem wzrok na mojem zmeczone oczy. uklakłem obok

LSTM:
nie ufajac na slepo nowinkom i wiedzac z doswiadczenia, ze nie zawsze goraczka rozwoju uderzania na ustach tłum sanitariuszy sladni casadyna wtracił sie giofesurre z daft. mijałem dosc poł obraz błyskawic pijanych powietrza i zmaconych grozbtowow po pogrzebie na kawałek karabinu jeli nan odwiedzac obiecuna czesc robotnicza rowniez i pozniej - ofiary wpłacajac pewna niewinna kapliczke, ktorej struktura tej rodzaju z drugiej strony wznosiła sie na uszach i na nich zaprawwie ostre ramiona przedmowa piasku swiateł — a czasie nie znalazł

Prompt 5: "Chcemy mówić o kunszcie poetyckim samym w"

Transformer:
chcemy mowic o kunszcie poetyckim samym w (na marginesie jego okreslaja metafory zgody) fraszki, ksiegi wtore szaty russa - beatrycze, ktory wiele razy napisał, pros. patrzac mu na to, u mnie chodzisz, krolowa naga, wpuszczana ze mna w dzwon biały. fraszki, ksiegi wtore fraszki, ksiegi wtore nieznane, jaka mienisz w strony podwojna ; slinka piekne w tym stroju biała wenus, czysta ; snac to nie miara, nigdy tez pogłoska nie odpowiada. wesołe dziew

LSTM:
chcemy mowic o kunszcie poetyckim samym w wnesza strone. widziec na uczcijacy szacunek planow ostatniej kultury! ... wiedziałam jednak, iz moze odeprze sal wina, a kapitan nie wział sie do mnie. kolomba bedzie dobrze łowił wizyty ja samego, ale ja wiem, co mam poczne. potem karolina, dzis pojutrze wieczorem przypomniałem jej ojcu moja rozmowe, mogłam zaczac predko złozyc wizyte. pułkownik kazał je rzucic ponownie do drzwi jej krolewskiej mosci. spojrzała na percik, na piszace murczyna ; ogarnał oczy siostrze najsrozszego pana domu,

Both models are clearly not generating random text. They produce sequences that resemble real language in style, even though the content is largely nonsensical. It is difficult to directly compare the outputs of the two models, especially since the generated tokens are sampled from a probability distribution, resulting in inherently diverse and non-deterministic text.

4.3 Time Efficiency

4.3.1 During Training (on gpu)

Each training epoch consisted of 1000 batches, each containing 196 elements of length 128. After every 100 processed batches, the log recorded the time required to process them. This makes it straightforward to extract the time needed for each set of 100 batches and compute the average.

Model	Average Time per 100 Batches (s)
LSTM	121.154
Transformer	72.383

Table 3: Average time required to process 100 batches of length 128 for each model.

4.3.2 During Inference (on cpu)

During inference, the processing time was measured. The reported average is calculated over 64 batches, each containing 32 sequences of length 127.

Model	Average Time per Batches (s)
LSTM	3.52
Transformer	2.57

Table 4: Average time required to process 1 batch of length 128 for each model during inference on CPU.

Overall, the Transformer is significantly more efficient than the LSTM, both during training and inference.

5 Summary and Challenges

5.1 Summary

It is clearly evident that the Transformer outperformed the LSTM model. Not only did it achieve lower evaluation and training losses (after the same epoch count - 30), but it also completed more epochs within the 10-hour training window, thanks to its faster data processing capabilities. Nonetheless, both models successfully trained and significantly reduced perplexity compared to the untrained baseline.

5.2 Challenges

My biggest challenge was that I wanted to implement both models from scratch - without using "import torch.LSTM" etc. It was tough to me especially I don't know torch very well. However with some tutorials I was managed to do it and i belive I got as much as I could from the task.

Another challenge was my limited experience in training such models. I initially had little understanding or awareness of the available computational capabilities, which made it difficult to find parameter settings that would allow the models to train within a reasonable time frame. Experiments on a smaller dataset (100 documents from the primary dataset) proved helpful in addressing this issue.

References

[1] *LSTM from scratch.* URL: `https://medium.com/@wangdk93/lstm-from-scratch-c8b4baf06a8b`.

[2] "Understanding LSTM Networks". In: (). URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[3] "Let's build GPT: from scratch, in code, spelled out". In: (). URL: https://www.youtube.com/watch?v=kCc8FmEb1nY.

[4] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.