# NLU Language Modeling project

*Davide Lobba (232089)*

University of Trento

davide.lobba@studenti.unitn.it

## 1. Introduction

The following report presents the results of the Natural Language Understanding course project at the University of Trento. The objective of this project was to create a Language Model trained on the Penn Treebank dataset [1]. The work was structured as follows:

- Implementation of a standard baseline LSTM [2]
- Improvements to the LSTM following the paper Merity et. al [3]
- Implementation of GPT2 small provided by Huggingface [4]. Zero-shot and finetuning analysis on Penn Treebank.

## 2. Task Formalisation

Language modeling is a fundamental task in Natural Language Processing (NLP) that aims to predict the likelihood of words in a sentence, as well as generate new words based on a given context. The goal of this task is to train a model that can produce new meaningful text according to the user's needs. It is essential to train the model on a massive corpus of text, such as books, articles or web pages. This is because, during the training phase, the language model learns the dependencies and probability distributions between words, which enables it to generate text that is coherent and meaningful.

In recent years, the task of language modeling has been revolutionised by Large Language Models (LLMs) such as BERT [5] or GPT-3 [6]. These models have reached superior performance rather than classical methods such as RNN, GRU or LSTM.

Language modeling can be subdivided into two categories: Causal language modeling and Masked language modeling. The first predicts the next token given a sequence of tokens, and the model can only attend to tokens on the left. The second predicts a masked token in a sentence, and the model will be able to look at the token list bidirectionally. In this project, I will focus only on Casual language modeling.

## 3. Data Description & Analysis

The dataset used for training and evaluating the models used in this project is a preprocessed version [7] of the Penn Treebank (PTB) [1]. This version of PTB contains only lower-cased words, numbers are replaced with N, rare words are replaced by unk and the vocabulary consists of the most frequent 10k words. This is one of the most popular datasets used by the NLP community, and it is used for both character-level and word-level language modeling. The dataset was created by the University of Pennsylvania by collecting articles from the Wall Street Journal.

The corpus of the PTB dataset is written in English and it consists of 24 sections. The first 18 sections are used for training, sections 19 to 21 for testing, and sections 22 to 24 for evaluating the model. The classic PTB dataset includes different types of annotations, such as Part-of-Speech, Syntactic, and Semantic parsing, instead preprocessed PTB includes only raw text. The dataset used for this project is the preprocessed version because it is needed only the raw text, indeed the text itself will serve as the label for the model.

The training set consists of 887521 words, 42068 sentences with a rounded mean length of 21 words per sentence, a rounded mean of 121 chars per sentence and a rounded mean of 6 chars per word.

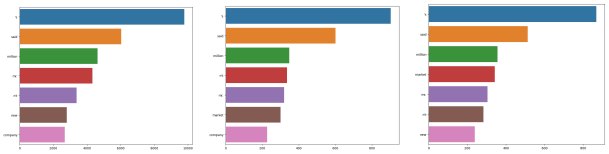
The testing set consists of 78669 words, 3761 sentences with a rounded mean length of 21 words per sentence, a rounded mean of 120 chars per sentence and a rounded mean of 6 chars per word.

The validation set consists of 70390 words, 3370 sentences with a rounded mean length of 21 words per sentence, a rounded mean of 119 chars per sentence and a rounded mean of 6 chars per word. For better visualisation, table 1 provides a summary of the dataset's statistics.

| | Train set | Test set | Validation set |
|---|---|---|---|
| Number of words | 887521 | 78669 | 70390 |
| Number of sentences | 42068 | 3761 | 3370 |
| Average words per sentence | 21.09 | 20.91 | 20.88 |
| Average chars per sentence | 121.27 | 119.63 | 118.62 |
| Average chars per word | 5.74 | 5.71 | 5.67 |

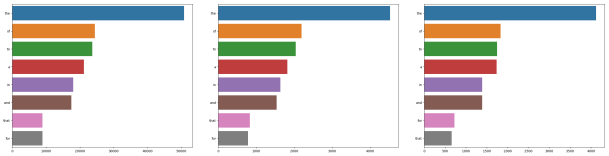
Table 1: *Table of corpus statistics analysis*

Regarding the label distribution, I conducted an analysis of the top seven non-stop words, and top seven stop-words, as shown in Figure 1, 2. It is important to note that for the purpose of this analysis, I decided to exclude the tokens "unk", "N", and "$" from the chart, in order to better highlight the most commonly used words in the dataset. However, it is worth mentioning that "unk" and "N" have the highest occurrences in the dataset. In the training set, the occurrences of "unk" are 45020, "N" are 32481, and "$" are 7541. In the test set, the occurrences of "unk" are 4794, "N" are 2523, and "$" are 564. In the validation set, the occurrences of "unk" are 3485, "N" are 2603, and "$" are 659. This demonstrates the high bias of the Penn Treebank dataset towards those tokens. This high bias is going to produce a bad language model, which will produce in turn highly biased text. This bias in the dataset towards certain tokens will have an impact on the performance of the model, as it will be demonstrated in the examples presented in section 5.5. In these examples, it will be shown that a significant number of tokens generated by a model trained on the PTB dataset will be "unk" and "N", indicating that the model struggles to generalise to rare or unseen words.



(a) *Train set* (b) *Test set* (c) *Validation set*

Figure 1: *Bar chart of top non stop words*



(a) *Train set* (b) *Test set* (c) *Validation set*

Figure 2: *Bar chart of top stop words*

# 4. Model

As mentioned in section 1, the goal of this project is to develop a language model trained on the Penn Treebank dataset. Specifically, all the models used for this model are unidirectional, so the text will be processed only from the left to the right. This means that the model utilizes the previous words in the input sequence to predict the next word, as opposed to masked language models which also take into account the context from right to left. The models developed in this project include:

- LSTM vanilla
- LSTM with improvements proposed by Merity et al. [3]
- GPT2 provided by Hugging Face[4]

## 4.1. LSTM vanilla

The LSTM vanilla is a Long Short-Term Memory (LSTM) model. LSTM is a type of Recurrent Neural Network (RNN) that is commonly used in NLP tasks, such as language modeling and text classification. The architecture consists of an embedding layer, an LSTM layer and an output layer which is a fully connected layer. The embedding layer converts each word into a vector representation, the LSTM layer processes the sentence, and the fully connected layer makes predictions for the class probability of each word.

One of the main challenges that I encountered in this project is the overfitting of the LSTM vanilla. Despite the train perplexity decreasing, the validation perplexity was found to be increasing, indicating that the model had stopped learning. To address this issue, I decided to implement some regularization techniques proposed by Merity et al. [3]. Those regularization techniques are known to be very effective in preventing the problem of overfitting in RNN or LSTM models.

## 4.2. LSTM improved

Regarding the LSTM with modifications, the improvements that I decided to implement are:

- Optimizer NT-ASGD: it is an improved version of the ASGD optimizer
- Weight dropout: a dropout applied only on the recurrent hidden-to-hidden weight matrices of the LSTM
- Variational dropout (or Locked dropout): a locked binary dropout mask applied to input and output of the LSTM
- Embedding dropout: a dropout applied on the embedding matrix at a word level
- Weight tying: sharing of the weights between embedding and classification layer

### *4.2.1. Optimizer NT-ASGD*

The Non-monotonically Triggered ASGD is an averaged stochastic gradient descent technique. It is effective in improving accuracy because it has a non-monotonic criterion that conservatively triggers the averaging when the validation metric fails to improve for multiple cycles.

**Algorithm 1** Non-monotonically Triggered ASGD (NTASGD)

**Inputs:** Initial point $w_0$, learning rate $\gamma$, logging interval $L$, non-monotone interval $n$.

Initialize $k \leftarrow 0$, $t \leftarrow 0$, $T \leftarrow 0$, $logs \leftarrow []$
**while** stopping criterion not met **do**
  Compute stochastic gradient $\hat{\nabla} f(w_k)$ and take SGD step $(1)$
  **if** mod$(k, L) = 0$ and $T = 0$ **then**
    Compute validation perplexity $v$.
    **if** $t > n$ and $v > \min_{l \in \{t-n, \ldots, t\}} logs[l]$ **then**
      Set $T \leftarrow k$
    **end if**
    Append $v$ to $logs$
    $t \leftarrow t + 1$
  **end if**
**end while**
**return** $\frac{\sum_{i=T}^{k} w_i}{k - T + 1}$

### *4.2.2. Weight dropout*

The weight dropout was proposed by Wan et al. [8]. It consists of applying dropout only on the recurrent hidden-to-hidden weight matrices $[U^i, U^f, U^o, U^c]$ of the LSTM in order to prevent overfitting on the recurrent weights. It is important to notice that the dropped weights remain dropped for the entirety of the forward and backward pass, so the same set of weights is reused over different time steps.

### *4.2.3. Variational dropout*

The variational dropout was proposed by Gal and Ghahramani [9]. It consists of applying a binary dropout mask only once upon the first call and then to repeatedly use that locked dropout mask for all repeated connections within the forward and backward pass.

### *4.2.4. Embedding dropout*

Embedding dropout is a variation of the Variational dropout proposed by Gal and Ghahramani [9], in which the dropout, instead of being applied to inputs and outputs of the LSTM, is applied on the embedding matrix at a word level. The embedding matrix with embedding dropout applied is used for a full forward and backward pass.

### *4.2.5. Weight tying*

Weight tying was proposed by Inan et al [10]. It is a huge improvement regarding the efficiency and accuracy of the model

because it prevents the model from having to learn a one-to-one correspondence between the input and output, resulting also in less trainable parameters.

### 4.3. Transformer GPT-2

GPT-2 (Generative Pre-training Transformer 2) [11] is a transformer developed by OpenAI. Nowadays, Large Language Models (LLMs) are the state of the art for the language modeling task. For this project, I decided to use the smaller version of GPT-2, which has 124M parameters, due to computational constraints.
The pipeline is the following:

- Loading of Penn Treebank raw text
- Tokenization of the text using the pretrained tokenizer of GPT-2
- Finetuning the model using a fixed-size stride of 1024
- Evaluation of the model using the perplexity evaluation metric

Regarding the Zero-shot test, the pipeline is exactly the same but without finetuning.

## 5. Evaluation

The performance of a language model is typically evaluated using metrics such as perplexity, cross-entropy, and bits-per-character (BPC). Perplexity evaluates how well a language model is able to predict the next word in a sequence of words, given the preceding words. I remark that a causal language model aims to predict a token given a set of previous tokens. For this reason, perplexity is the most suitable evaluation metric. In this project, the models presented in section 4 were evaluated using perplexity as metric, which is calculated using the formula:

$$\text{PP(W)} = exp^{\text{LCE(W)}}$$

In which the $L_{CE}$ is the cross-entropy loss computed as follow:

$$L_{CE}(word_i) = -\frac{1}{N}\sum_{i=1}^{N}[log(P(word_i|word_1, ..., word_{i-1})]$$

Perplexity is a measure of how much a language model is "perplexed" by seeing a certain word given a set of preceding words. A lower perplexity value indicates that the model is able to make more accurate predictions, and thus it is considered to be a more effective model.

### 5.1. Evaluating LSTM vanilla

For the LSTM vanilla the hyperparameters used are the following:

- Embedding size of 300;
- Hidden size of 300;
- Batch size for training 32, for testing 16;
- One hidden layer;
- SGD optimizer with a learning rate of 1.0 without weight decay or momentum;
- Scheduler ReduceLROnPlateau with a reducing factor of 0.5 and patience 2;
- 60 epochs.

The LSTM vanilla model achieved, on the validation set, a perplexity of 128.84, with an average of 129.54 over five runs. However, those results could be improved a lot as demonstrated in section 5.2.

### 5.2. Evaluating LSTM with improvements

To address the issue of overfitting with the LSTM vanilla model, I implemented several regularization techniques proposed by Merity et al. in their paper [3]. These techniques include the use of the optimizer NT-ASGD, weight dropout, variational dropout, embedding dropout, weight tying, and adjustments to the model's hyperparameters.

- Embedding size of 500;
- Hidden size of 500;
- Batch size for training 32, for testing 16;
- Three hidden layers;
- NT-ASGD, which is an improved version of ASGD and SGD, with a learning rate of 10.0, a weight decay of 1.2e-6 and a clip of 1.0;
- Scheduler ReduceLROnPlateau with a reducing factor of 0.5 and patience 2;
- Embedding dropout of 0.1;
- Weight dropout of 0.5;
- Input and output dropout of 0.4;
- 60 epochs.

The LSTM improved reached, on the validation set, a perplexity of 79.89, with an average perplexity of 80.12 over five runs. Moreover, on the test set the model reached a perplexity of 75.96.

From those results, it is worth noticing that regularization methods proposed by Merity et al [3], such as weight dropout, or variational dropout are very effective. Furthermore, have been conducted an ablation study for the analysis of how much each regularization operation affects the perplexity of the model.

### 5.3. Ablation study LSTM

In this section will be analysed how much each method proposed by Merity et al [3] affects the model. For better visualisation, Table 2 and Figure 3 summarise the results obtained with this ablation study.

| Model | PP Validation set | PP Test set |
|---|---|---|
| LSTM with all improvements | 79.90 | 75.96 |
| LSTM no NT-ASGD | 80.52 | 76.06 |
| LSTM no variational dropout | 95.30 | 91.75 |
| LSTM no weight dropout | 87.65 | 83.73 |
| LSTM no embedding dropout | 83.70 | 79.17 |
| LSTM no weights tied | 87.25 | 82.78 |

Table 2: *Ablation study of the LSTM with improvements*

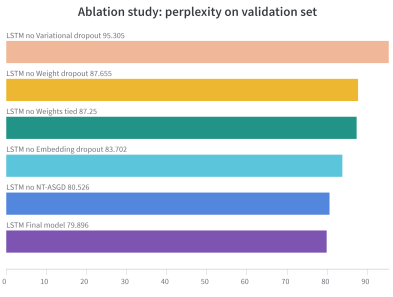



Figure 3: *Ablation study LSTM*

From Table 2 and Figure 3 it is possible to highlight how much is crucial the variational dropout in preventing overfitting and improving the performance of the LSTM. It is also worth noticing that weights dropout and weight tying significantly contribute to the improvement of the network while embedding dropout and the use of NT-ASGD do not affect heavily the performance.

The results obtained with LSTM improved are much better than the results obtained with LSTM vanilla, which indeed was expectable. Additionally, I decided to propose a further analysis presenting a transformer.

### 5.4. Evaluating Transformer GPT-2

GPT-2 [11] is a transformer pretrained on a vast corpus of English raw text in a self-supervised manner, unlike LSTM which is trained only on Penn Treebank. There are different versions of GPT-2 provided by Huggingface[4]; for computational issues, I decided to opt for the smaller one which has 124M parameters. The evaluation of GPT-2 was performed as follows:

- Zero-shot analysis
- Finetuning

The evaluation of GPT-2 was performed using a zero-shot analysis approach, where a sliding window with a stride of 1024 tokens was used to calculate the conditional likelihood of each token. As expected, the results obtained with GPT-2 were significantly better than those obtained with the LSTM architecture, indeed the perplexity reached is 66.02, which is consistent with the perplexity reported on GPT-2 paper (65.85 zero-shot perplexity using GPT-2 small) [11]

Furthermore, as mentioned before, has been performed also a finetuning on the Penn Treebank dataset in order to analyse how GPT-2 can adapt to this dataset. The results were astonishing, indeed with a finetuning of just 3 epochs the perplexity obtained was 25.65. The finetuning has been performed using as optimizer AdamW with a learning rate of 2e-05, weight decay of 0.01 and stride of 1024.
It is curious to mention that the smaller version of GPT-2 finetuned reaches a better perplexity than the larger version of GPT-2 (which is GPT-2 XL) with zero-shot, indeed the perplexity of GPT-2 XL zero-shot on PTB dataset is 35.76, as reported on the paper [11]

Table 3 and Figure 4 summarise all the results obtained with this project.

| Model | PP Validation set | PP Test set | N° params |
|---|---|---|---|
| LSTM Vanilla | 128.84 | 125.32 | 6.7M |
| LSTM Improved | 79.89 | 75.96 | 14M |
| GPT-2 Zero-shot | 66.02 | 57.68 | 124M |
| GPT-2 Finetuning | 25.65 | 23.19 | 124M |

Table 3: *Results of the project*

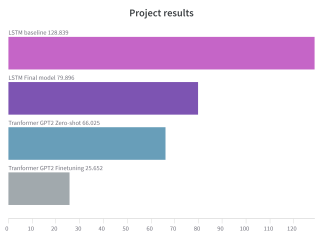



Figure 4: *Project results on validation set*

### 5.5. Evaluating inference prompt

Regarding the testing of the different models, I evaluated the performance of each model by generating text for a given sentence, with a maximum number of tokens generated of 15. As expected, GPT-2 zero-shot produced the highest-quality sentences. Below, you can find some examples of the generated text for each model.

Prompt: ”the new plan is”
LSTM vanilla: ”the new plan is expected to be completed by the end of the week”
LSTM improved: ”the new plan is subject to approval of the bill”
GPT-2 Zero-shot: ”the new plan is to continue to develop new technologies, which would further enhance safety, performance”
GPT-2 finetuned: ”the new plan is aimed at unk them to the N population by requiring unk”

Prompt: ”the journal”
LSTM vanilla: ”the journal is a N of the N and N of the N and N”
LSTM improved: ”the journal said it will N N N million shares N million shares outstanding N outstanding”
GPT-2 Zero-shot: ”the journal of the Russian Academy of Sciences, and then a researcher at MIT”
GPT-2 finetuned: ”the journal reported that N people died in N in north africa from the unk”

From those results, as mentioned in section 3, it can be highlighted again how much the Penn Treebank dataset is biased towards the tokens N and unk. This is confirmed by the sentences produced using GPT-2 finetuned, indeed also sentences produced by GPT-2 finetuned present a lot of N and unk tokens, which yield to an incomprehensible sentence. In contrast, GPT-2 zero-shot is able to produce more sophisticated and grammatically correct sentences, demonstrating its strong ability to generate coherent and meaningful sentences. This is likely due to the fact that GPT-2 zero-shot has been trained on a larger and more diverse corpus of text, allowing it to generalize better compared to models trained on the small and biased Penn Treebank dataset. Furthermore, the finetuning of GPT-2 on PTB dataset shows a remarkable improvement in the perplexity but still, the generated sentences are not as good as the ones generated by the zero-shot GPT-2.

## 6. Conclusion

In the end, the perplexity obtained on the validation set with LSTM vanilla is only 128.84 because this model is prone to overfitting, but with some regularization techniques proposed by Merity et al. [3] the perplexity improved and the model reached 79.90. Furthermore, the analysis performed on GPT-2 small with zero-shot led to a perplexity of 66.02, but it has been improved with a finetuning on PTB which led to a perplexity of 25.65. However, as confirmed in section 5.5, the best model for the generation of new text is GPT-2 zero-shot because it has a much vast vocabulary and it is able to generalise better. All in all, I am satisfied with the results obtained from this project.

# 7. References


[1] M. Marcus, "Building a Large Annotated Corpus of English: The Penn Treebank:." [Online]. Available: http://www.dtic.mil/docs/citations/ADA273556

[2] R. C. Staudemeyer and E. R. Morris, "Understanding lstm – a tutorial into long short-term memory recurrent neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1909.09586

[3] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," 2017. [Online]. Available: https://arxiv.org/abs/1708.02182

[4] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2019. [Online]. Available: https://arxiv.org/abs/1910.03771

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[7] D. Nunes, "Preprocessed penn tree bank," Jun. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3910021

[8] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. III–1058–III–1066.

[9] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," 2015. [Online]. Available: https://arxiv.org/abs/1512.05287

[10] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," 2016. [Online]. Available: https://arxiv.org/abs/1611.01462

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2018. [Online]. Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf