HLT project
Reservoir in Neural Language Models

Luca Moroni (-)

- 2022

1 Introduction

This years the field of Natural Language Process was revolutionized by the application of artificial intelligence techniques, especially the usage of the deep learning with attention based models ([9], [2]). The application of the neural language models have reached capacities near the human ones, and even more, in certain tasks. But at what cost those models manage to do so well? The fact is that they are getting bigger and bigger, increasing the number of parameters, in an exponential way.

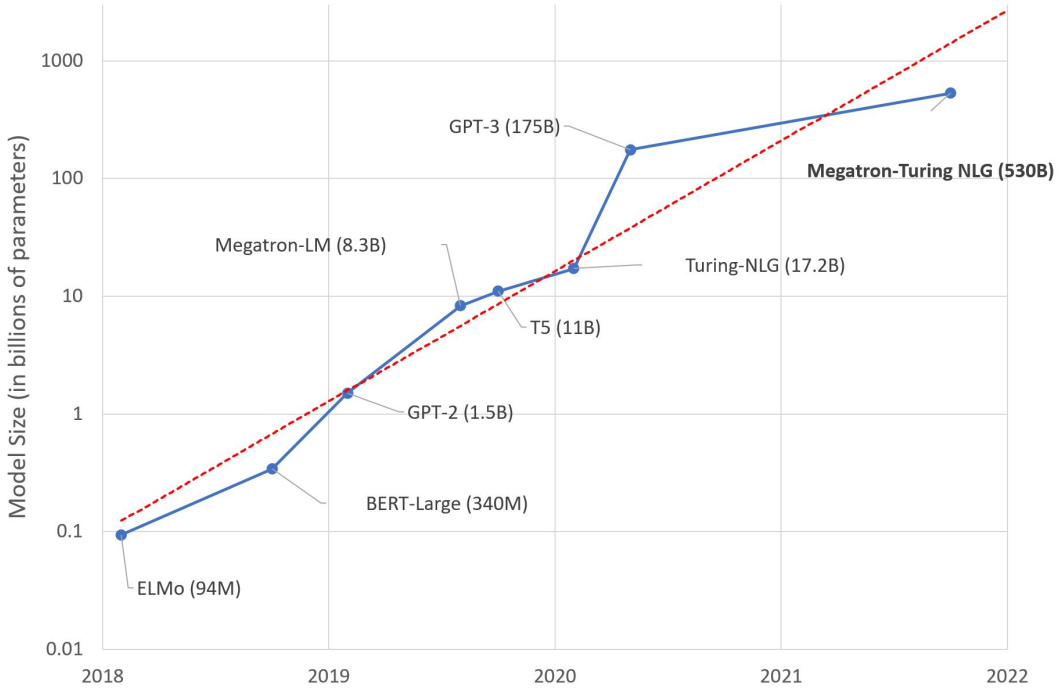

Figure 1: Growing of models' parameters during time

More parameters implies more parameters to learn, a modern neural network need a time proportional to the number of parameters to learn them, using backpropagation

techniques. Several studies have raised several problems relative to this exponential increment on the dimension of those models. First of all, if we don't change the course, the energy consumption will be more and more in near future, and this translates in a huge carbon footprint and a very destroyer environmental impact. Scientists are studying several way to measure this impact ([7], and others) and to mitigate this fact they are developing alternative metrics, taking into account the energy consumption and not only the accuracy, to evaluate the neural language models. Nowadays this alternative ways to evaluate a model are not much taken into account because the major players in NLP filed (the big tech companies) have other objectives in mind, their principal aim is to make better models than the competitors, so developing models than do some zero dot something more in term of accuracy with respect to other vendors' models on the market. To the other hand some scientists ([1]) have raised the fact that those bigger models are so powerful because they tend to learn by heart. Due to the fact that those models are so big, they need a huge computational power to be used and deployed, so another problem is the democratization of them, and the impossibility sometimes to the public researcher to have the same capacity and the necessary power technologies to do some kind of experiments as the private companies.
The aim of this project is to observe the effect of the Reservoir computing (randomized neural networks) in the field of the Natural Language Processing.
Reservoir methodologies, in the first meaning, implies the usage a high dimensional random projection followed by a linear readout layer, exploiting the cover theorem that state, A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.
The principal aim of the experiments, of this project, is to try techniques that mitigate the carbon footprint of the modern Neural Language Models.
In practice We have trained BERT-like models over MLM task and then tested them using finetuning tasks (mnli and sst2). For first we have trained a medium-BERT as a baseline for our experiments, then we have tried some other architectures based on reservoir methods. We have defined three different architectures:

- **RESBERT**: bert model in which some layers are randomly initialized and never trained.
- **UBERT**: bert model in which some layers are substituted with Euler state networks.
- **RFFNBERT**: bert models in which some layers are substituted with a randomly initialized (and never trained) feed forward neural network.

1.1 Euler State Networks

The Euler State Networks are a kind of Recurrent Neural Networks introduced in [3], They are developed following from the idea behind the Echo State Networks, so modelling the Recurrent Networks as dynamical systems, initialized in a random fashion constraining the parameters to impose some theoretical properties (as echo state property), the Euler State Networks makes use of forward Euler discretization and anti-symmetric recurrent matrices to design reservoir dynamics that are both stable and non-dissipative by construction. Experiments done in [3] have shown that those models can do well on tasks which require long term memorization skills. Moreover other experiments have shown that EuSN is capable of matching the level of accuracy of trainable Recurrent Neural Networks, while allowing up to 100-fold savings in computation time and energy consumption.

2 Prior Works

The works present in this project are based over several papers, first of all we can cite Reservoir Transformers ([8]), in which are present the experiments for ResBert and RFFN-Bert over MLM and downstram tasks, mnli and sst2. In our opinion those experiment can be extended with Ubert, and moreover in that paper the code is not present and that, for us, is a huge lackness. The definition of Ubert model was driven by the works done in [10] by Kiela et al. in which was shown how reservoir models can reach a very good results in the encoding for sentence classification, in particular the best results was given by the Echo State Network ([4]), so since the Euler State Network is a novelty version of the ESNs and moreover, in theory, have better properties to deal with sequences, our assumption is that those models can do well compounded in BERT-like models.

3 Data

Two kind of training phase was doing during the experiments.
For first we have pretrained the four models over a dataset composed by **wikipedia** (20220301.en) see `https://huggingface.co/datasets/wikipedia` and **bookcorpus** see `https://huggingface.co/datasets/bookcorpus`.
After the pretraining of the models we have done two finetuning tasks, from **GLUE** bechmarks, to evaluate the models , see `https://huggingface.co/datasets/glue`.
The first is **MNLI** benchmark, given a premise sentence and a hypotesis sentence, the task is to predict whether the premise entails the hypotesis. And **SST2** whose task is to predict the sentiment of a given sentence.

4 Architecture

In this section I will explain the architectures of the models. All the models described are a augumentation of the classical BERT architecture.
The implementation of the following models can be seen in this link `https://github.com/Andrew-Wyn/echo-state-transformers/tree/ilc-cnr-echo-state-transformers`

4.1 RESBERT

The RESBERT model have the same architecture of BERT model in which some layers are kept frozen during training and in which the projection matrices (K, Q, V) are initialized in an orthogonal way ([6]). In the implementation is possible to use a scaling factor to immerse the vector in a bigger space (scaling up the dimensionality) during the random projection, and before to pass to the next layer project them in the original dimension using a Gaussian random projection matrix.

4.2 UBERT

The UBERT model is a BERT model in which some layers are substituted with Euler State Networks whose parameters are not learned. Even in this case we can scaling up the dimensionality an project back.

4.3 RFFNBERT

The RFFNBERT model is a BERT model in which some layers are substituted with a Feed Forward Neural network whose parameters are not learned,

$$output = GaussianProj(FFN(LayerNorm(input))) + input.$$

Where the FFN immerge the input in a higher dimensional space in random way and the $GaussianProj$ reduce the dimensionality to the input one.

5 Experiments

The experiments are done using the Hugging Face libraries, in particular Hugging Face Transformers and Hugging Face Datasets.
The code of the experiments can be seen in the following link `https://github.com/Andrew-Wyn/ILC_reservoir_transformer_experiments`

5.1 Preprocessing

The preprocessing of the uninion of wikipedia and bookcorpus datasets follow the work done in [5], summing up the phases,

- Take 10% of the dataset as the **validation** part.
- **Split** the text parts in sentences.
- **Tokenization**.
- **Grouping** the texts (sentences) in chunks of 256 tokens, put [CLS] as first token.

5.2 Architectures

in Bert-like models have to be defined some architectural parameters, in our case, due to time limits we have choosen the following numbers. For all the models the number of layers is 8, the hidden size is 256, the number of head for each layer is 4, the intermediate size of the transformer layers (size of the feed forward part) is set to 1024. In RESBERT model the reservoir scaling factor is set to 1, following the works done in [8] and the reservoir layers (the ones that not learn) is two (3 and 6). In UBERT two layers is substituted with the Euler State Networks (3 and 6), in this case the reservoir scaling factor is set to 5, trying to exploit the cover theorem. Even in RFFNBERT two are the layers substituted with the feed forward network (3 and 6), and even in this case the reservoir scaling factor is set to 5.

5.3 Working Machines

The machines used to do the work are the following, to do the pretraining we have used the CINECA MARCONI 100, using a single node for every pretraining and the four GPU (NVIDIA Volta V100 GPUs, Nvlink 2.0, 16GB) on each node. Since the long time needed to schedule the processes in the MARCONI 100, to run the GLUE finetuning tasks we have used a 16 core computer in the CNR-ILC laboratory.

5.4 learning parameters

Due the time limits, we have tryed only one set of parameters during pretraining and finetuning.
For the pretraining we have used the following ones, we have follow the ones used by [5], the rest is the default ones,

- **per_device_train_batch_size** = 32.
- **per_device_eval_batch_size** = 8.
- **warmup_step** = 30000.
- **weight_decay** = 0.01.
- **num_train_epochs** = 3.

- **learning_rate** = 1.e-4.

For the finetuning we have choosen the following ones,

- **max_seq_length** = 256.
- **per_device_train_batch_size** = 32.
- **per_device_eval_batch_size** = 8.
- **num_train_epochs** = 4.
- **learning_rate** = 2e-5.

For the finetuning we have used the following script, `https://github.com/huggingface/transformers/blob/main/examples/pytorch/text-classification/run_glue.py`.

6 Results and Analysis

In this section we will expose the results of the experiments described above.
In the following table we can see the accuracy of the models after the pretraining phase.

Model	Train Loss	Eval Loss	Eval Accuracy
BERT	0.91	2.33	0.57
RESBERT	0.90	2.43	0.55
UBERT	0.24	3.8	0.35
RFFNBERT	0.79	2.45	0.55

Table 1

We can notice that, in the end the BERT can do better in evaluation, but the reservoir methods can do better in the train data, going in overfitting, especially for UBERT model, this is a fact that can be solved doing more runs choosing the proper learning parameters, and applying some kind of regularization techniques.
In the following plots we can see the results of the MNLI finetuning task, the first plots are referred to the BERT model without pretraining, only finetuning training.

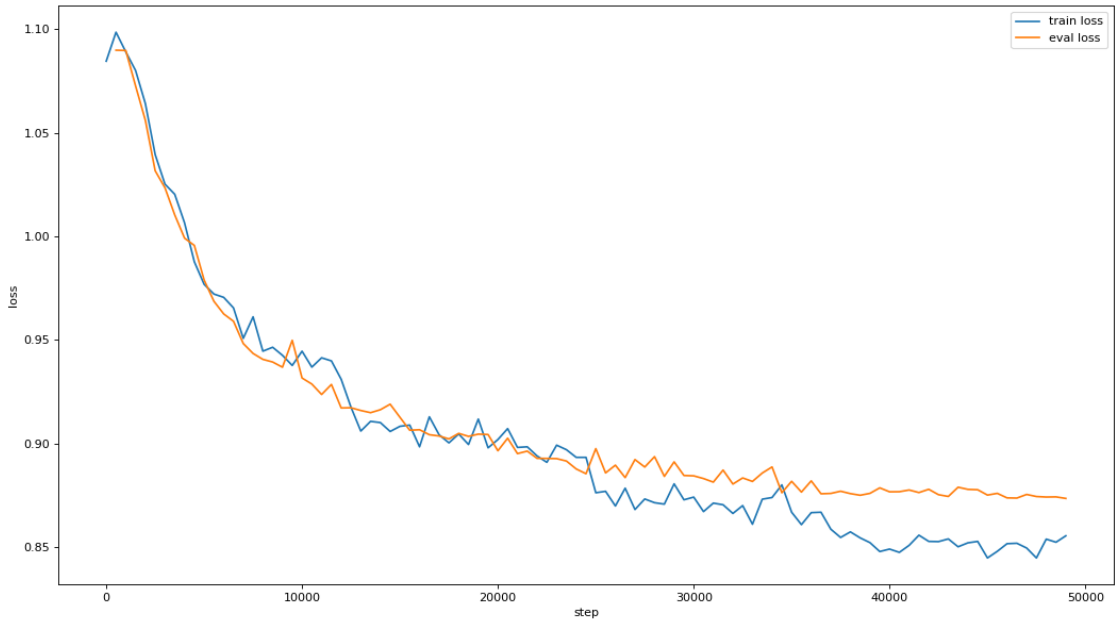

Figure 2: BERT not pretrained: Plot of mnli, eval loss and train loss.

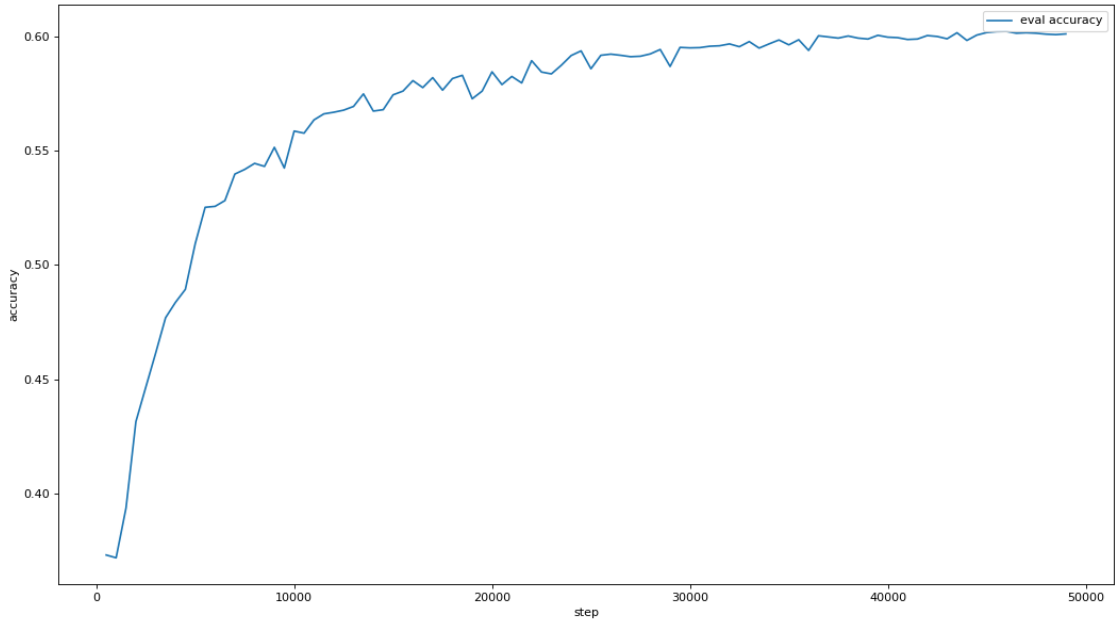

Figure 3: BERT not pretrained: Plot of mnli, eval accuracy.

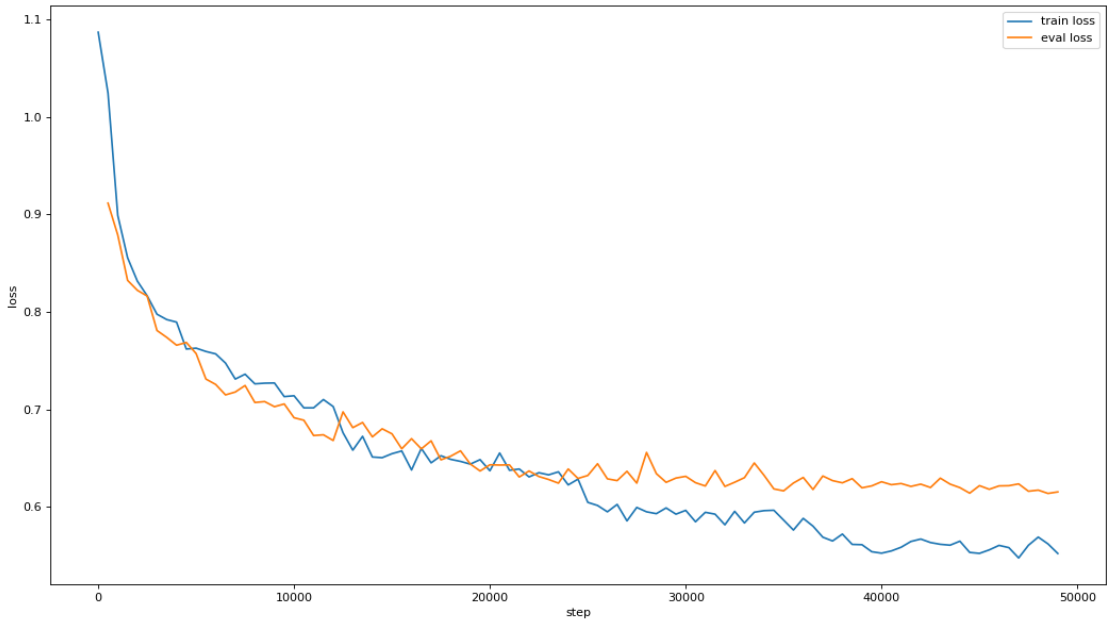

Figure 4: BERT: Plot of mnli, eval loss and train loss.

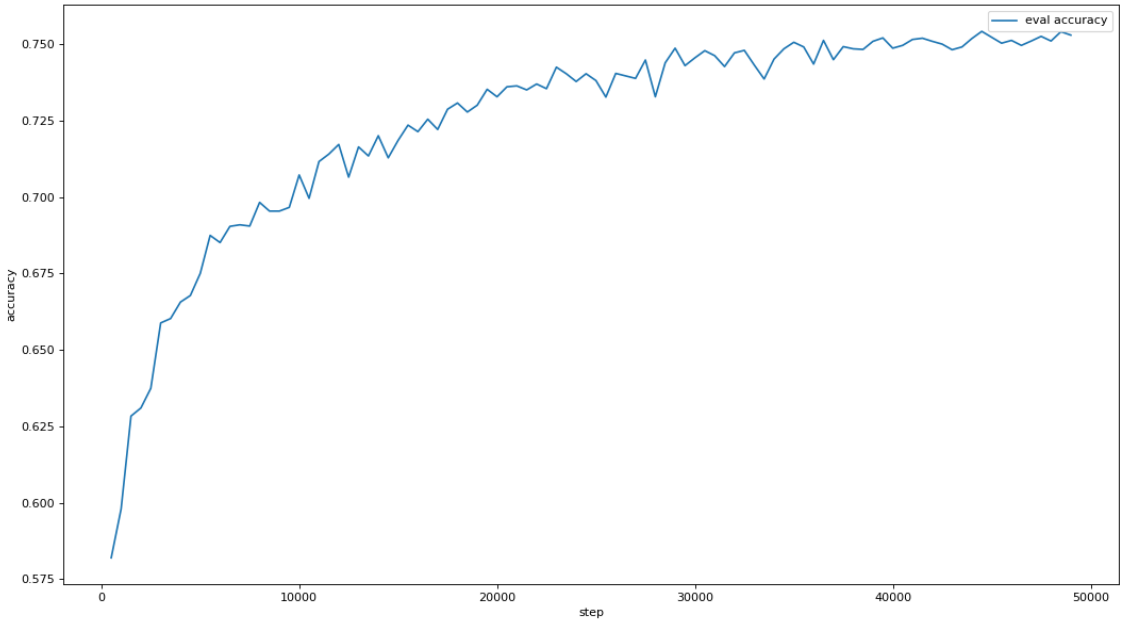

Figure 5: BERT: Plot of mnli, eval accuracy.

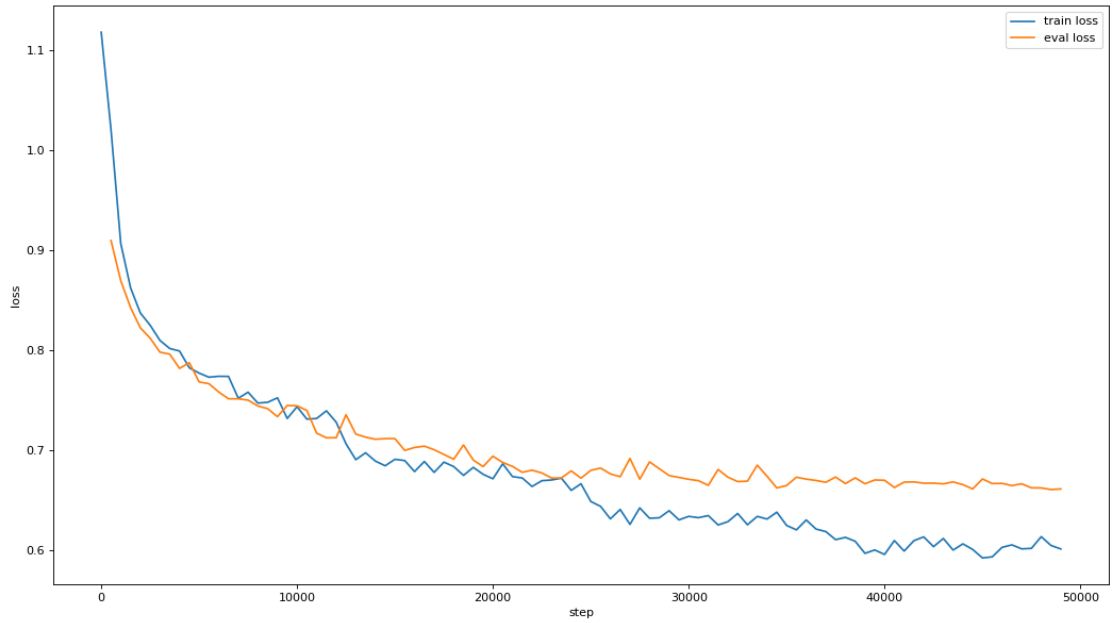

Figure 6: RESBERT: Plot of mnli, eval loss and train loss.

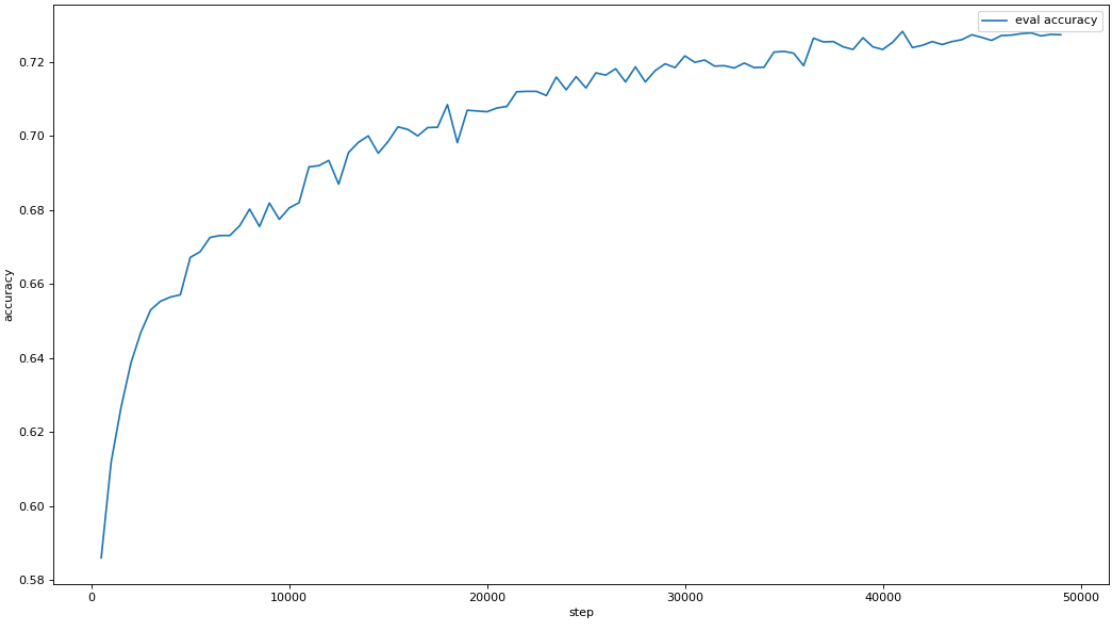

Figure 7: RESBERT: Plot of mnli, eval accuracy.

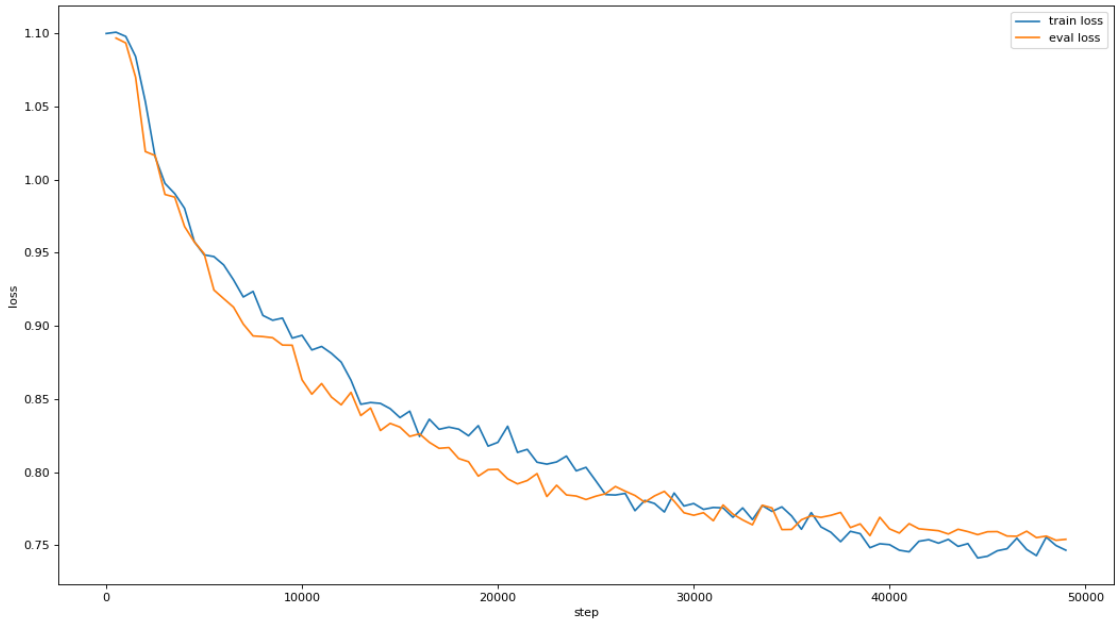

Figure 8: UBERT: Plot of mnli, eval loss and train loss.

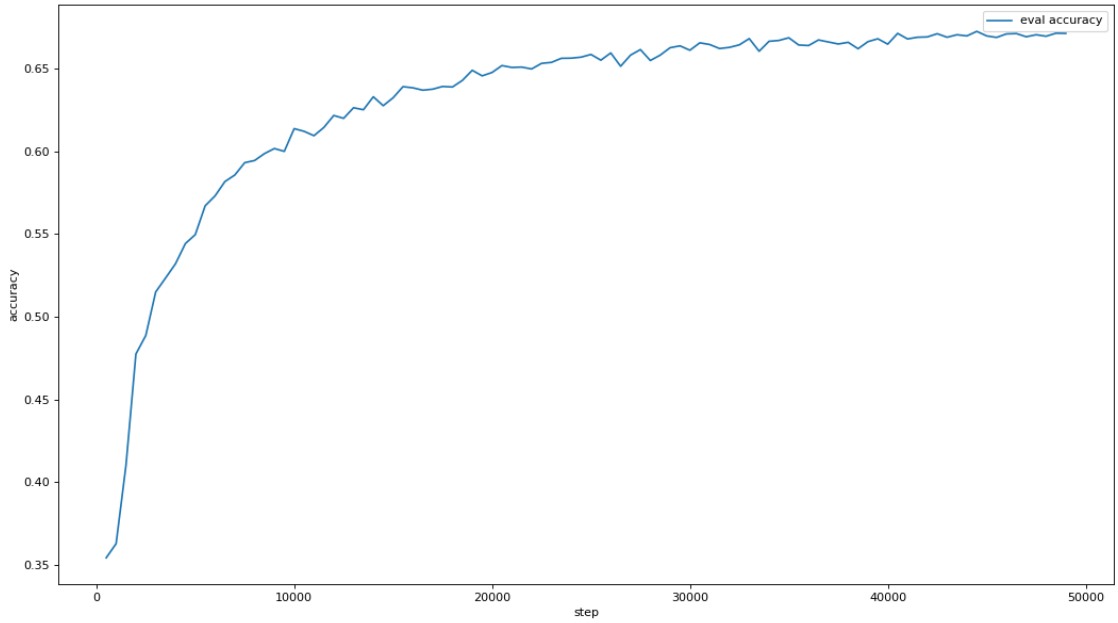

Figure 9: UBERT: Plot of mnli, eval accuracy.

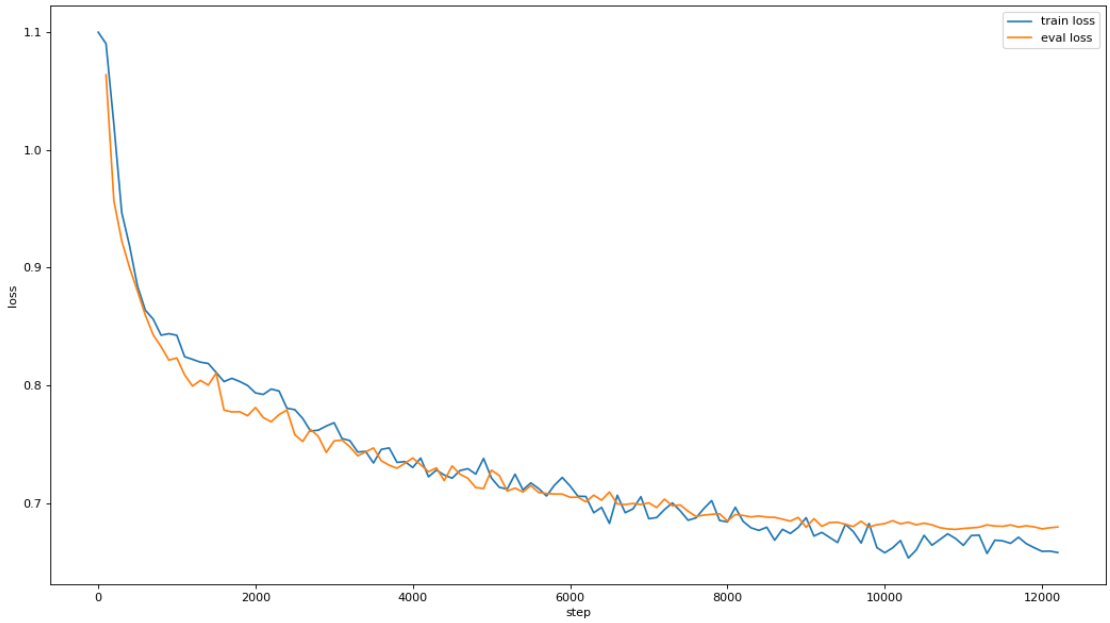

Figure 10: RFFNBERT: Plot of mnli, eval loss and train loss.

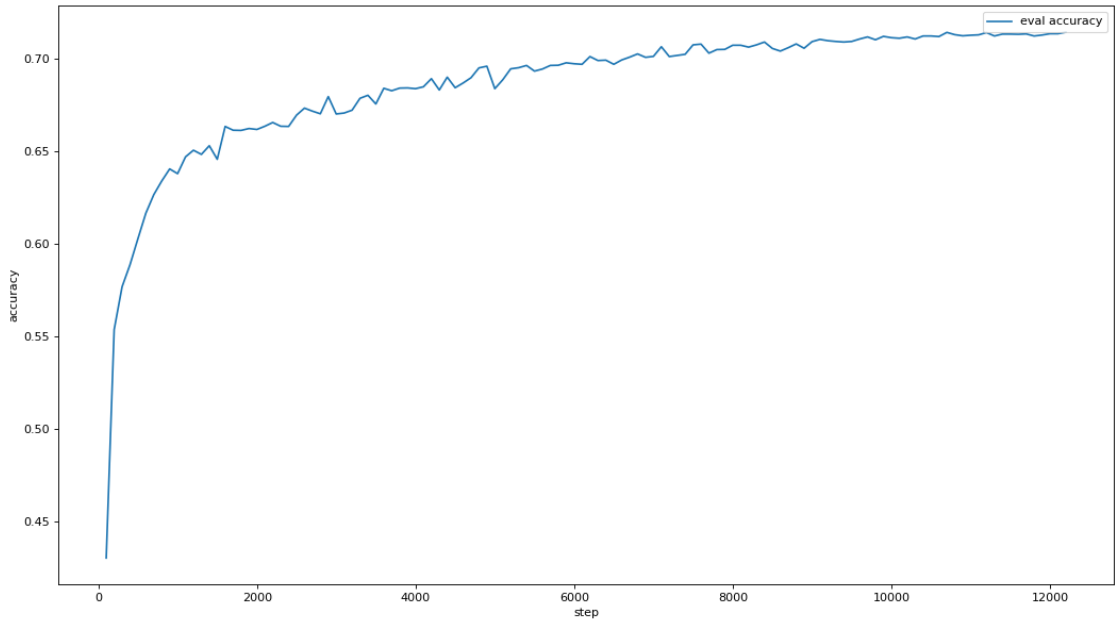

Figure 11: RFFNBERT: Plot of mnli, eval accuracy.

In the following plots we can see the results of the SST2 finetuning task, the first plots are referred to the BERT model without pretraining, only finetuning training.

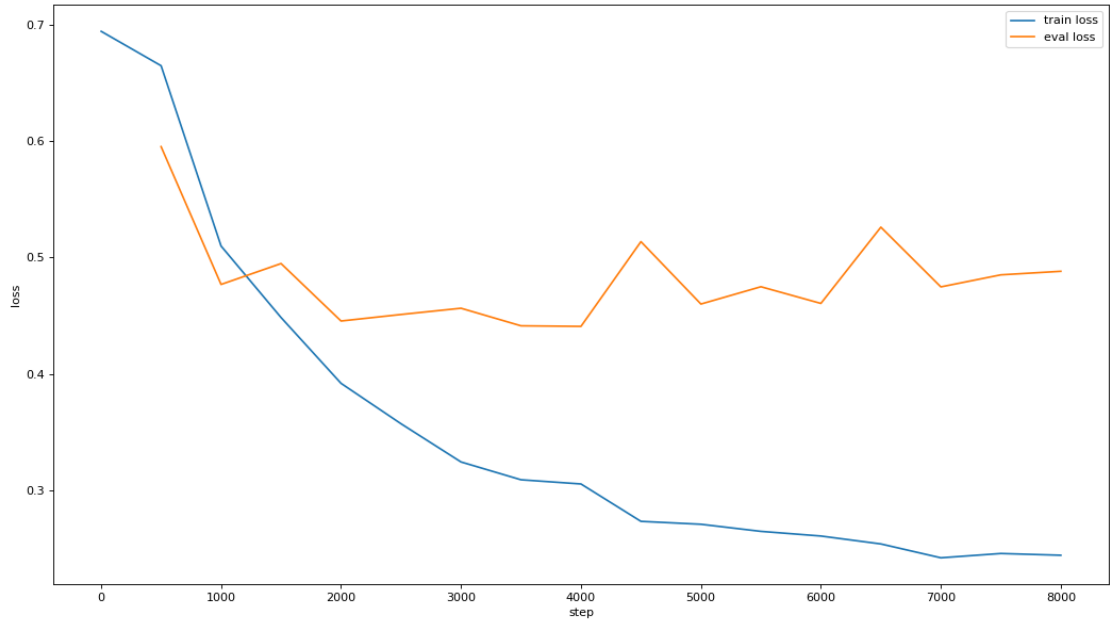

Figure 12: BERT: Plot of sst2, eval loss and train loss.

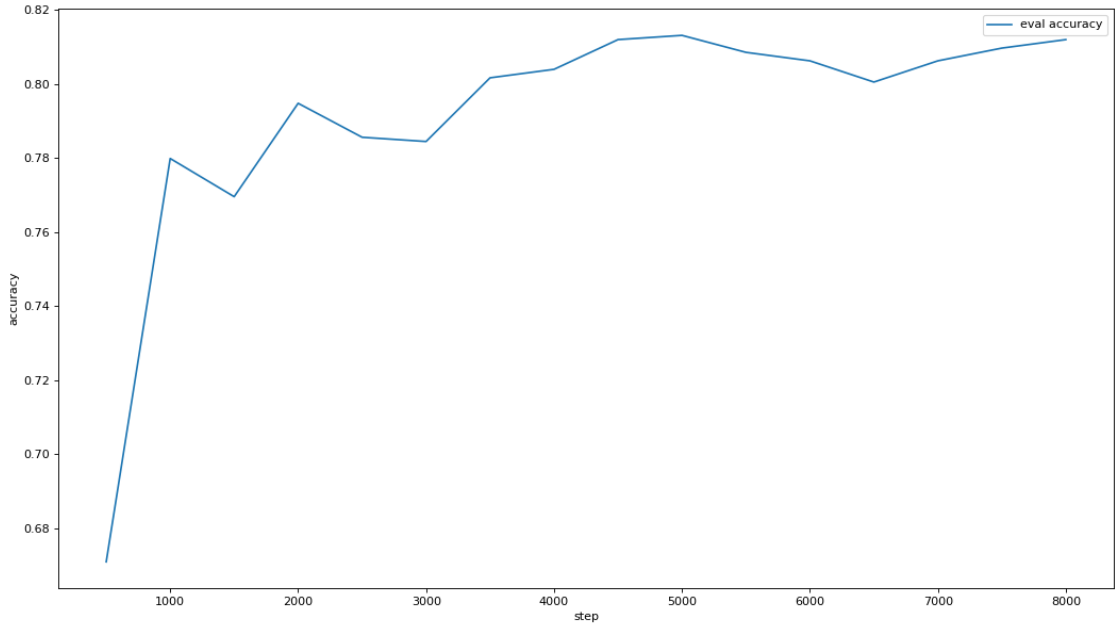

Figure 13: BERT: Plot of sst2, eval accuracy.

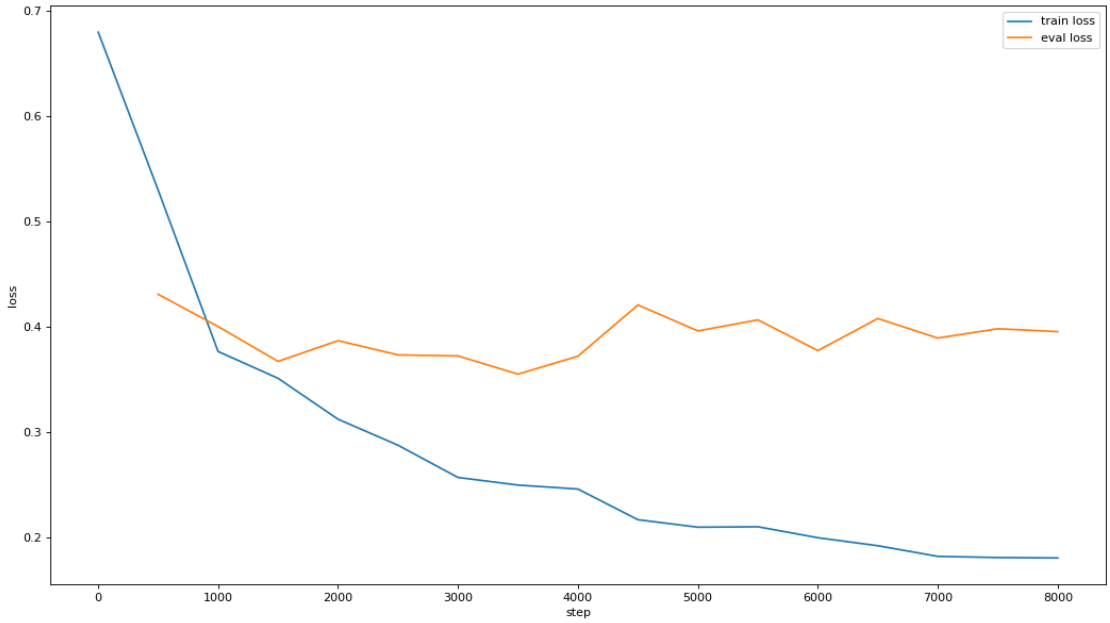

Figure 14: BERT: Plot of sst2, eval loss and train loss.

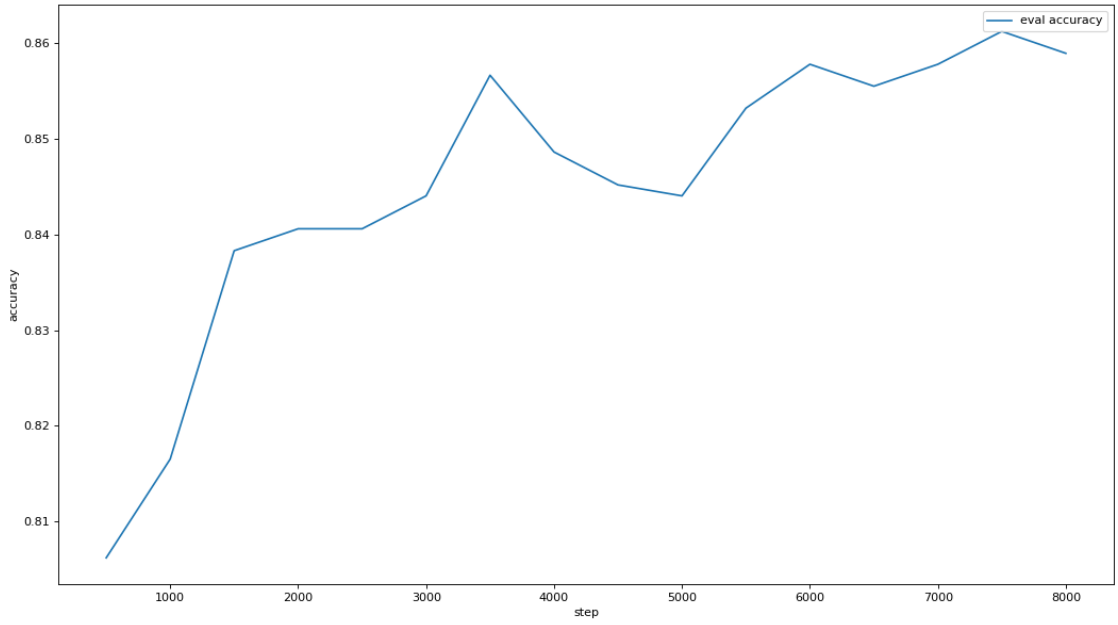

Figure 15: BERT: Plot of sst2, eval accuracy.

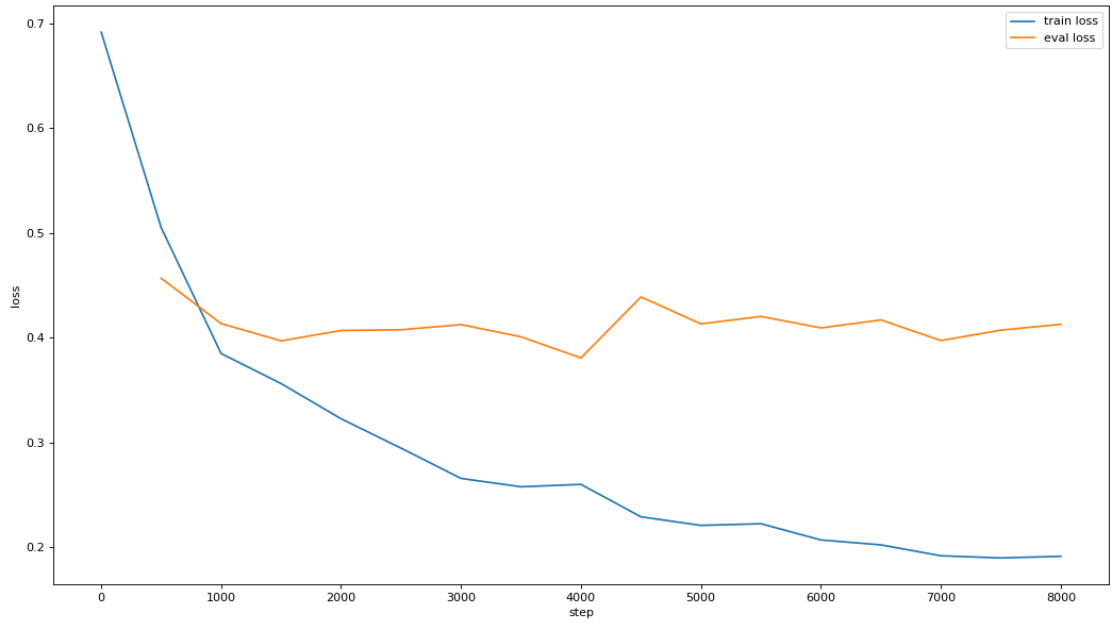

Figure 16: RESBERT: Plot of sst2, eval loss and train loss.

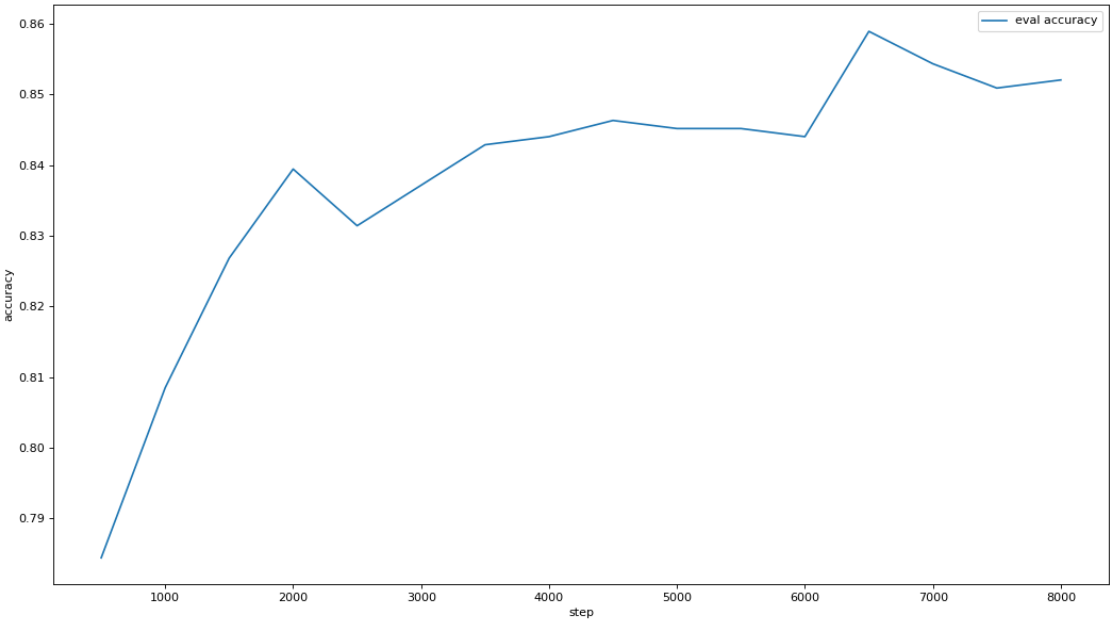

Figure 17: RESBERT: Plot of sst2, eval accuracy.

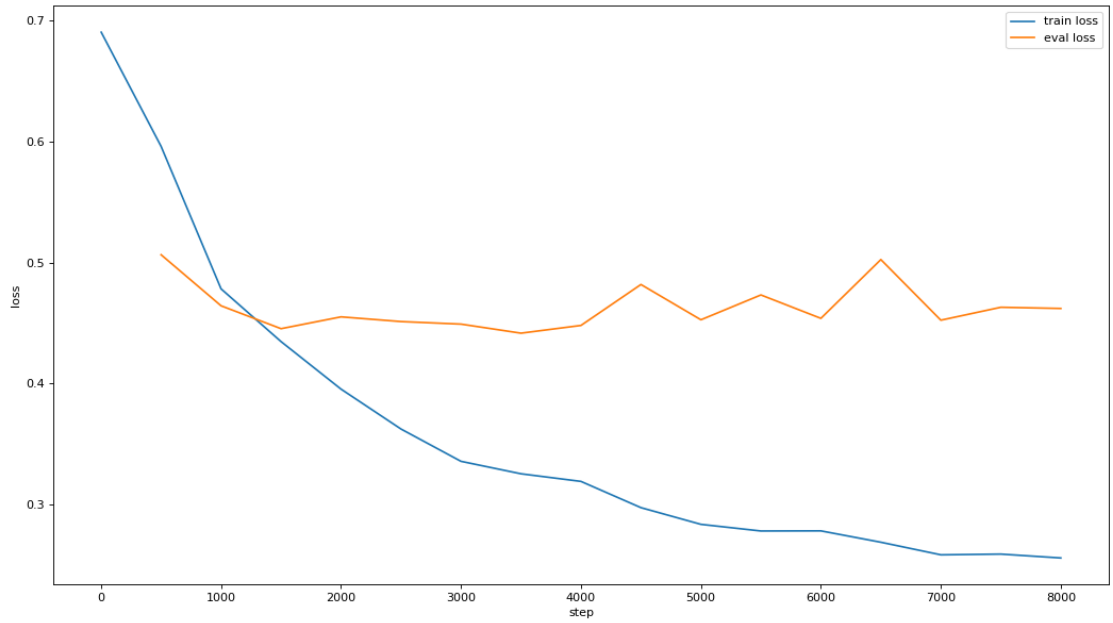

Figure 18: UBERT: Plot of sst2, eval loss and train loss.

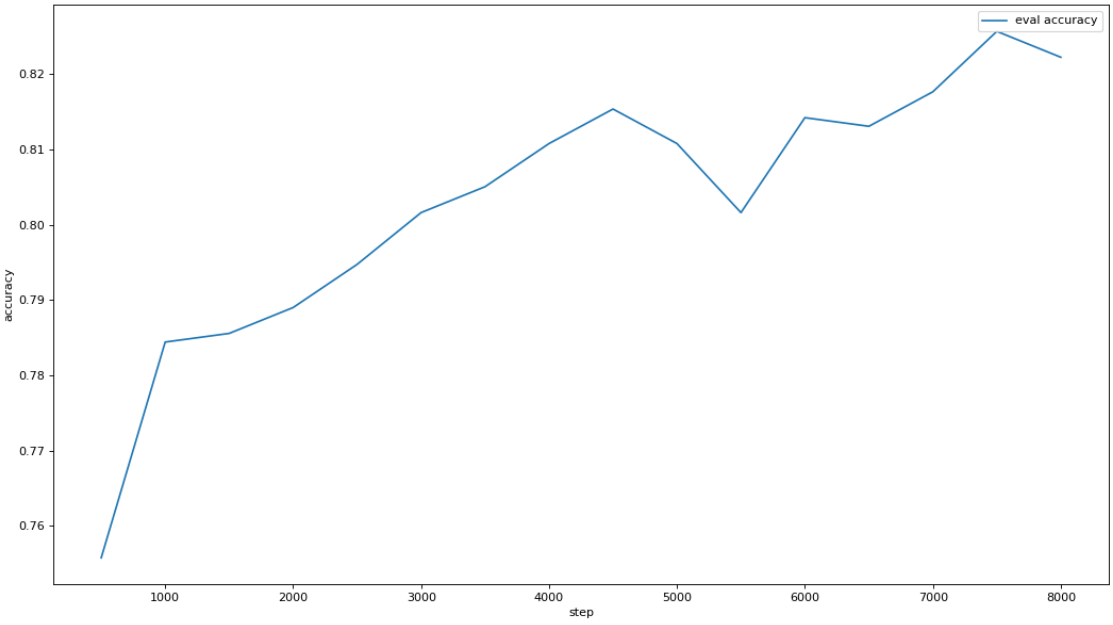

Figure 19: UBERT: Plot of sst2, eval accuracy.

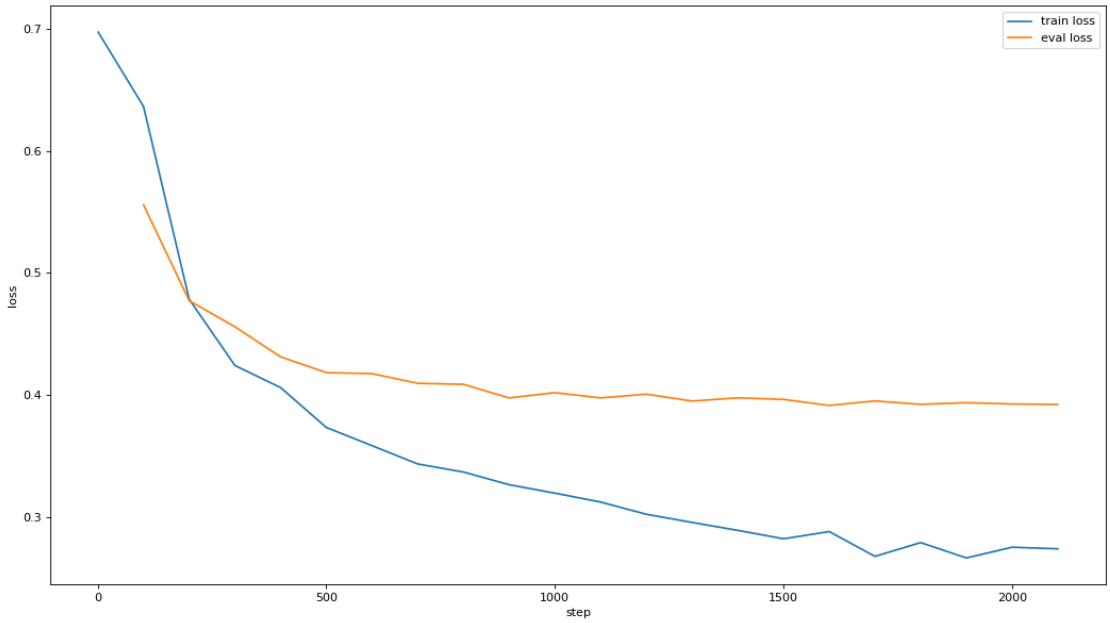

Figure 20: RFFNBERT: Plot of sst2, eval loss and train loss.

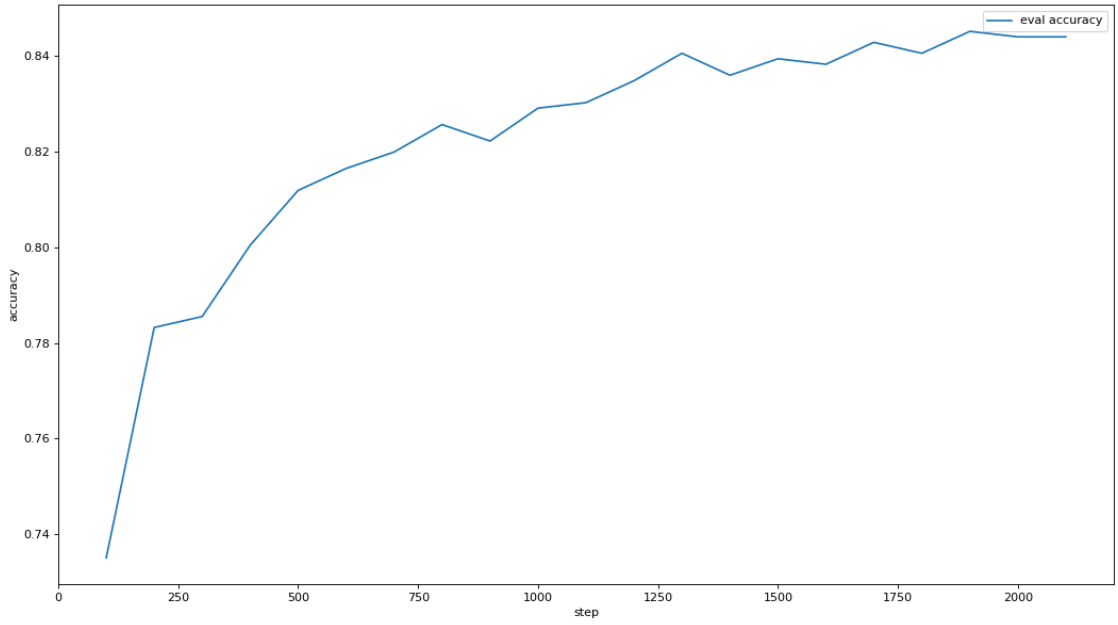

Figure 21: RFFNBERT: Plot of sst2, eval accuracy.

In the following table we can sum up the results.

Model	Train loss	Eval loss	Eval accuracy
BERT not pretrained	0.85	0.87	0.60
BERT	0.55	0.61	0.75
RESBERT	0.60	0.66	0.72
UBERT	0.74	0.75	0.67
RFFNBERT	0.65	0.67	0.71

Table 2: Mnli: train loss, eval loss and eval accuracy

Model	Train loss	Eval loss	Eval accuracy
BERT not pretrained	0.24	0.48	0.81
BERT	0.18	0.39	0.85
RESBERT	0.19	0.41	0.85
UBERT	0.25	0.46	0.82
RFFNBERT	0.27	0.39	0.84

Table 3: SST2: train loss, eval loss and eval accuracy

The full trained BERT can do sligthly better, but take apart the UBERT model, the difference is not so huge and we have a gaining in training times as we see in the following tables. Even in this case We could do some hyperparameters search

Model	Train Runtime (seconds)	Train samples per second
BERT	65407.6438	894.848
RESBERT	59007.2886	991.91
RFFNBERT	38499.5527	1520.275

Table 4: Training runetime and Train sample per second, during pretraining.

We can see that, RESBERT give a gain of 10% on train time needed, while RFFNBERT can give a gain of 42%. In the table is not presented the time of UBERT since the implementation of a recoursive neural network in pytorch need some optimization techniques that I didn't do well.
The next table sum up the number of parameters of each trained model,

Model	Number of Parameters
BERT	14360634
RESBERT (scaling factor: 1)	14360634
UBERT (scaling factor: 5)	17371194
RFFNBERT (scaling factor: 5)	14095418

Table 5: Number of paramaters for each model

7 Conclusions

In the end we can say that, the results shown that the reservoir models can do better over the train set in the pretraining phase, going in overfitting (especially for UBERT). The results on the finetuning tasks are proportional to the validation loss on the pretraining task. The work done was limited due to the time needed, more extensive experiments could be done, such as hyperparameters search in the pretraining phase and finetuining tasks. Moreover to get a valid comparison of the pretrained models we have to do more than one run of pretraining for each model. I am aware of the lackness of this comparisons, in future works we will do more experiments, trying to mitigate the overfitting of the reservoir models and trying to make UBERT faster, optimizing the code.
The results presented in this paper don't follow the one presented in [8], and We don't know why, we think that the motivations are different, one of those are the fact that, They use **ROBERTA** with more parameters, and not BERT as base model. More work can be done to understand better this discrepancy.
Taking apart those facts, I think those works are hopefully to start to think how the reservoir techniques and models can help in NLP field, to take down the computational resources needed to do some tasks, while mantaining the same performances, or at least without taking down it to much.
As stated in the work of Euler State Netwokrs ([3]) those models cannot do very well in the short time memory, since is a feature needed in the NLP field, we will do some experiments combining the Euler State Networks with the Echo State Networks, since they have more short time memory.
Moreover another kind of experiment that we can do is to use the Euler State Network, due to it's properties, not as component of a BERT-like model but as student model in a distillation learning scenario.

References

[1] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too

big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Claudio Gallicchio. Euler state networks. *arXiv preprint arXiv:2203.09382*, 2022.

[4] Herbert Jaeger. Echo state network. *scholarpedia*, 2(9):2330, 2007.

[5] Giovanni Puccetti, Anna Rogers, Aleksandr Drozd, and Felice Dell'Orletta. Outliers dimensions that disrupt transformers are driven by frequency. *arXiv preprint arXiv:2205.11380*, 2022.

[6] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[7] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

[8] Sheng Shen, Alexei Baevski, Ari S Morcos, Kurt Keutzer, Michael Auli, and Douwe Kiela. Reservoir transformers. *arXiv preprint arXiv:2012.15045*, 2020.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[10] John Wieting and Douwe Kiela. No training required: Exploring random encoders for sentence classification. *arXiv preprint arXiv:1901.10444*, 2019.