An exploration of semi-supervised text classification

Henrik Lien[2], Daniel Biermann[2], Fabrizio Palumbo[1], and Morten Goodwin[1,2]

[1] Artificial Intelligence Lab (AI Lab),
Institutt for informasjonsteknologi, Oslo Metropolitan University, Oslo, Norway
fabrizio@oslomet.no

[2] Centre for Artificial Intelligence Research (CAIR)
Department of ICT, University of Agder, Grimstad, Norway
{daniel.biermann,morten.goodwin}@uia.no

Abstract. Good performance in supervised text classification is usually obtained with the use of large amounts of labeled training data. However, obtaining labeled data is often expensive and time-consuming. To overcome these limitations, researchers have developed Semi-Supervised learning (SSL) algorithms exploiting the use of unlabeled data, which are generally easy and free to access. With SSL, unlabeled and labeled data are combined to outperform Supervised-Learning algorithms. However, setting up SSL neural networks for text classification is cumbersome and frequently based on a trial and error process.
We show that the hyperparameter configuration significantly impacts SSL performance, and the learning rate is the most influential parameter. Additionally, increasing model size also improves SSL performance, particularly when less pre-processing data are available. Interestingly, as opposed to feed-forward models, recurrent models generally reach a performance threshold as pre-processing data size increases.
This article expands the knowledge on hyperparameters and model size in relation to SSL application in text classification. This work supports the use of SSL work in future NLP projects by optimizing model design and potentially lowering training time, particularly if time-restricted.

Keywords: Machine Learning · Text classification · Semi-Supervised learning

1 Introduction

Text classification has many useful application areas, for example, classifying emails, medical texts, or even sentiments. Manual text classification is unfeasible due to significant time and economic costs. Artificial Intelligence (AI) techniques automate text classification tasks, making text classification cheaper and practical. Within AI, neural networks using traditional supervised learning often require a substantial amount of labeled data to achieve good text classification performance. Unfortunately, obtaining labeled data frequently requires domain-specific expertise. In contrast, getting an extensive volume of unlabeled data is often free and simple. There is a significant amount of research within the exciting area of semi-supervised learning (SSL) for text classification. The SSL technique exploits both unlabeled and labeled data to achieve improved classification performance.

Semi-Supervised Learning

The field of machine learning frequently draws a line between *supervised learning* and *unsupervised learning* [4]. Supervised learning uses a dataset containing data (x-samples) and respective labels/targets (y-samples). In contrast, unsupervised learning uses a dataset containing data (x-samples) but not labels (y-samples). *SSL* uses both supervised and unsupervised learning [6] [22]. SSL aims to increase the performance of either the supervised or unsupervised approach. It exploits knowledge from both approaches. In particular, if labeled data is challenging to obtain, using unlabeled data can be significantly useful. For successful use of SSL, three *assumptions* need to be satisfied: the smoothness assumption, the cluster assumption, and the manifold assumption [6, p. 6].

Pre-Training

Unsupervised pre-training utilizes labeled and unlabeled data within two phases. Parameters for unsupervised pre-training are static within feature extraction techniques. They can, however, be modified during the supervised fine-tuning within pre-training techniques. Before supervised learning, unlabeled data moves the decision border towards potentially more relevant areas with pre-training methods. Unsupervised pre-training of a model commonly lowers the required volume of labeled data necessary to achieve good performance. Downstream NLP tasks then became simpler and cheaper to implement. Unsupervised pre-training also makes it possible to fine-tune a model to multiple downstream tasks, reducing training time and regularly improving downstream performance. Pre-training research is therefore valuable within NLP.

Impact of Parameters on SSL Performance

It is challenging to define precise circumstances where an SSL technique is effective. It is important to highlight that unlabeled data does not always improve results [19]. Previous literature has shown decreased performance as a result of SSL, a phenomenon probably under-reported because of publication bias [22]. Several articles have investigated the use and implementation of SSL [22] [6] [18] [13] [16], but it is still unclear how to maximise its performance. This is especially true when good results are reached using purely supervised classifiers, and deploying SSL are more likely to degrade performance. There are multiple parameters that can significantly impact SSL performance within text classification. These include:

Pre-training Data Size

Raffel [17] et al. and Baevski et al. [2] show that reducing the volume of pre-training data can result in performance degradation since a large network could overfit on a small quantity of pre-training data. Raffel et al. [17] recommend utilizing a significant amount of pre-training data. Importantly, obtaining additional unlabeled data is inexpensive and straightforward.

Model size

Enlarging network size and/or training time generally improve results [17]. According to a recent paper by Bender et al. [3], the expanding scale of language models, estimated by volume of training data and parameters, has been a significant trend within NLP.

Hyperparameters

There is much literature showing that hyperparameters are very relevant for achieving good performance with SSL. Devlin et al. [8] observe that using a significant amount of data and increasing the number of hyperparameters, improves results particularly for GLUE [20] . In the RoBERTa paper [15], Liu et al. show that hyperparameters have a significant impact on SSL performance. You et al. [21] show that with 32k as batch size, BERT training time can be shortened significantly without affecting performance. Dai and Le [7] show that LSTM models can be trained and reach good performance on multiple text classification tasks, with fine adjustment of hyperparameters. It is important to highlight that some hyperparameters have bigger impact on the model performance than others. According to Goodfellow et al. [9], "The learning rate is perhaps the most important hyperparameter". However, there is not a significant volume of research exploring the impact of parameters on SSL performance for text classification.

This article explores the impact of hyperparameters, including pre-training data size and model size, on an SSL algorithm employed in a text classification task. A limited number of epochs and smaller models are used, due to hardware limitations. A program for running experiments is written based on code from an earlier project [14].

2 Methods

Structure of Experiments

Two experiment types are run for each model: Supervised learning (SL) experiments and Semi Supervised Learning (SSL) experiments. SL experiments train the models with labeled data only, without pre-training. SSL experiments pre-train the models using unlabeled data first. Then, the model is fine-tuned with labeled data.

-Feed-Forward Model

It contains an embedding layer, a dropout layer, two hidden layers, and one or multiple output layers. It uses a single output layer for the text classification task because this requires only a single output. For the pre-training task with predicting two or three masked tokens, it uses two or three output layers. These output layers are sharing hidden layers. Therefore, this model does multi-task pre-training with hard parameter sharing.

-GRU model

This model contains an embedding layer, a dropout layer, a GRU layer, a dense hidden layer followed by a ReLU activation function, and a single or two/three output layers. The output layers are similar to the output layers in the feed-forward model.

-Sequence to Sequence (Seq2seq), Classifier

It contains an encoder and a decoder. The decoder implements an attention mechanism. The encoder contains an embedding layer, a dropout layer, a bidirectional GRU layer, and the *hidden* layer. The decoder contains an embedding layer, a dropout layer, a GRU layer, an *energy* layer, and an output layer. The implemented *Seq2seq* model with attention mechanism is based on code from GitHub by aladdinpersson [1]. This model does single-task learning during pretext task training. Pretext task training uses the *Seq2seq* model, while supervised text classification training uses the *Classifier* model. Both these models contain an encoder. In contrast to the Seq2seq model, the Classifier model does not have a decoder. The Classifier achieved similar performance as the Seq2seq model during testing in a previous project [14]. Therefore, to improve training time, the Classifier model was used when possible. Additionally, in this work, we reference both the *Seq2seq* and *Classifier* model as *Seq2seq* model.

Data Handling

For pre-training, training, validation, and testing datasets are created from the original 20newsgroups [11] dataset. This data contains no headers, footers, or quotes. Unwanted characters are removed from this data. Data is split into sentences containing at the minimum 11 words. For each sentence, a sliding 10-word window iterates over the words. For each 10-word sequence, the words "*sos*" and "*[MASK]*" are inserted before the 10 words themselves. This includes the "*sos*" and "*[MASK]*" tokens in the resulting tokenizer. The eleventh word is also used, mainly because of using the codebase from the previous project [14]. Both supervised and downstream text classification training use data from the original Banking77 dataset [5]. A testing dataset is also obtained from [5] and used in experiments as validation data.

Experimental procedure

After creating the 11-word datasets from the 20newsgroups and Banking77 datasets, both supervised learning and SSL experiments are run for each model. The vocabulary used by the tokenizer is limited to words that appear at least twice in the data. A downstream task experiment initializes model weights with the best model weights obtained during pretext task training. Best model weights achieve the lowest validation loss. If not pre-trained, model weights are randomly initialized. A model trains and validates for a particular number of epochs. The pre-training uses a masking pretext task. The "*[mask]*" token replaces two or three random tokens in the sequence and the objective is to predict the masked tokens. During pre-training or validation, each batch uses dynamic masking. By randomly masking sequences in each batch, the pre-training dataset is enlarged artificially. Sequences in pre-training batches randomly mask during both training and validation. For training, the cross entropy loss is calculated and model parameters are updated using the Adam optimizer.

Hyperparameters

This article uses two baseline hyperparameter configurations for experimentation. The *Vanilla configuration*, is meant to represent a typical configuration of hyperparameters used in machine learning. The *SOTA configuration*, is partly based on hyperparameters in two GitHub projects [10] [12], based on a paper [7] by Dai and Le. The SOTA configuration masks three tokens in each ten-word sequence. This results in a masking of 30 percent for each 10-word sequence. Additionally, supervised, pre-training, and downstream training use 200 epochs each during experimentation.

Table 1 summarizes the hyperparameters used in this study. Additional controlled parameters are:
-Dataset size. The number of 10-word sequences created from the 20newsgroups dataset, to generate training, validation, and testing datasets. Used sizes range from 25k to 500k.
-Training, validation and test ratio. Ratios of 80%/10%/10% are used.
-15 Newsgroups from the 20newsgroups dataset are used.
-Length of word sequences. 10-word sequences are generated from the 20newsgroups dataset.

Model evaluation

We evaluate the performance of the models not by an absolute measure but with a relative measure called *SSL performance*. The lowest achieved text classification validation loss using supervised learning, minus the lowest achieved text classification validation loss using SSL. This measure allows us to easily see if a model benefits from using a SSL scheme or not.

$$\text{SSL performance} = L_S - L_{SSL} \tag{1}$$

In Equation 1, L_S is the text classification loss using purely supervised learning, and L_{SSL} is the text classification loss using SSL.

3 Results

Impact of hyperparameters configuration and model size on the SSL performance

Figure 1 shows the impact of hyperparameter configuration and model size on the SSL performance for each model, represented as the mean and variance of 10 simulations each. For all models with SOTA configuration, SSL performance improves compared to the Vanilla configuration. This improvement supports the hypothesis that the hyperparameters configuration significantly impacts SSL performance. Ten simulations are not a substantial number of simulations for each model, so this figure should not be observed as significantly conclusive. As expected, more extensive models improve SSL performance compared to smaller models.

Impact of increasing pre-training dataset size on the SSL performance

To test the impact of the pre-processing data size on the SSL performance we run each model configurations, both Vanilla and SOTA, with different pre-training data size. We expect that increasing the size of pre-training data leads to improved performance across all models. Additionally, based on the results observed in Figure 1 we expect bigger models to outperform smaller ones. The results of SOTA simulations are shown in Figure 2. For the feed-forward model, increasing

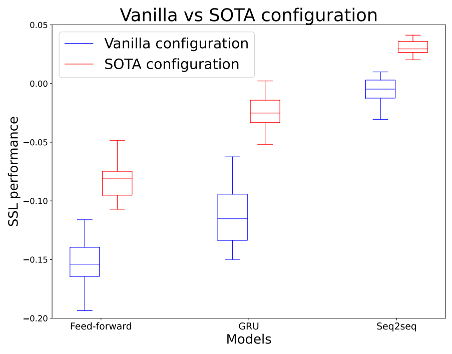

Fig. 1. Vanilla configuration versus SOTA configuration, using 25k pre-training data, 200 epochs for each training phase, and ten simulations for each configuration.

Hyperparameter	**Vanilla**	**SOTA**
Embedding size	512	256
Batch size	512	1024
Hidden size	1024	512
Dropout rate	0.0	0.2
Learning rate	0.0001	0.001
Number of masked tokens	2	3

Table 1. Baseline Vanilla configuration and SOTA configuration hyperparameters for experimentation

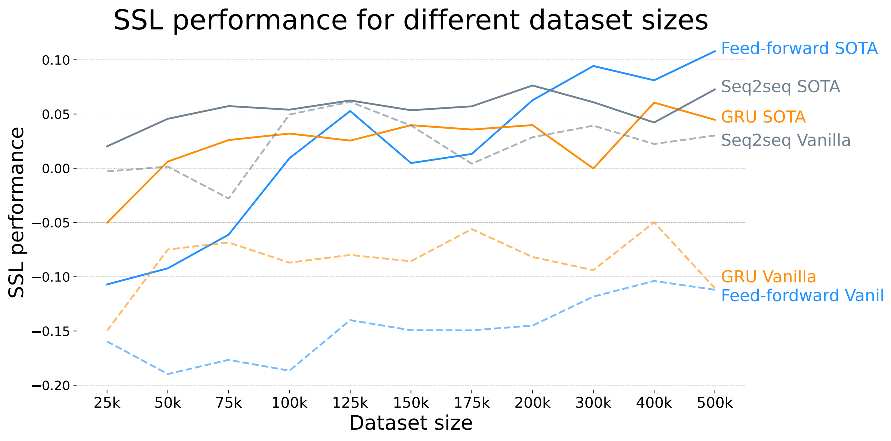

Fig. 2. Comparing pre-training data amounts, using SOTA and Vanilla configuration and 200 epochs for each training phase.

the pre-training data amount leads to an increase in SSL performance. However, for both GRU and Seq2seq models, the SSL performance reaches a plateau when more than 100k sequences are used for pre-training. This experiment shows that a feed-forward model improves SSL performance as long as pre-training data volume increases. Importantly, with a recurrent model, increasing pre-training data size beyond 100k does not improve the SSL performance significantly. This result is unexpected. Next, we investigate the SSL performance across epochs by calculating the loss function for a fix 25K sequences pre-training dataset. This analysis clearly shows that for larger models, the loss graph converges faster (data not shown in this manuscript) and reaches lower values. Both these observations support the idea that more comprehensive models learn more useful features from pre-training data per epoch. These results, taken together with Figure 2 also show that the more the loss graph converges, the less additional pre-training data improves SSL performance. Increasing pre-training data volume with SOTA configuration results in improved SSL performance for all models with different dynamics, as described in this paragraph. However, when the Vanilla configuration is tested in a similar framework, we observe a small or no impact of pre-training data size on SSL performance. This result is unexpected, particularly for the feed-forward model when compared to the SOTA configuration.

Impact of changing single hyperparameter on the SSL performance
To test the impact of each parameter on the SSL performance of the model and investigate whether the impact depends on other hyperparameters, we modify one parameter at a time while keeping the others constant. Simulations are performed with two different pre-training dataset sizes of 25K and 100K sequences. Only the 100K simulation is presented in Figure 3. In Figure 3 the impact of SSL performance is quantified across all models with the SOTA configuration as a starting point. As expected, the learning rate has a crucial impact on the results of the simulation: changing it from 0.001 to 0.0001 significantly lowers the SSL performance across all models. It is of particular interest to investigate the impact of different learning rates on the loss function of the model. In essence, when tested in a Seq2seq model, a higher learning rate (0.001) results in loss graphs converging significantly faster. Thus, indicating that the model has learned more useful knowledge during pre-training leading to improved downstream task performance. Another observation is that changing the dropout rate from 0.2 to 0.0 slightly increases the SSL performance for all models when using 100k dataset size. However, when the 25k dataset size is used (data not presented), changing the dropout rate to 0.0 significantly lowers the SSL performance for all models. This might be due to overfitting during pre-training when using only 25k data samples. This shows that the pre-training dataset size has an impact on the significance of single hyperparameters.

We last simulate the impact of changing single hyperparameters across models using the Vanilla configuration, Figure 4. Again, the learning rate shows to be an important hyperparameter also for the Vanilla configuration. However, the way the learning rate influences SSL performance in the two configurations is not the same. In the SOTA configuration, decreasing the learning rate leads to lower SSL performance, Figure 3. Therefore, it might be reasonable to expect the SSL performance to increase when increasing the learning rate. However, this is not the case. A similar decrease in SSL performance can be seen in the Vanilla configuration when increasing the learning rate from 0.0001 to 0.001, see Figure 4. This performance indicates that individual hyperparameters can not be tuned independently, and some interplay between hyperparameters exists. Another unexpected result in the Vanilla configuration is that changing the

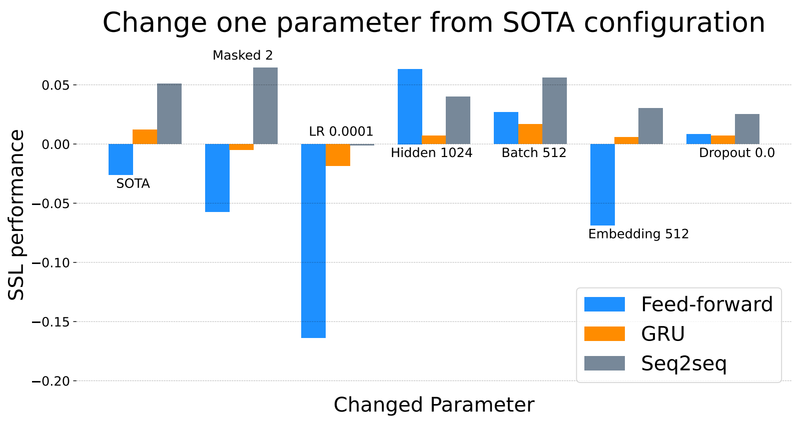

Fig. 3. Changing one parameter at a time from SOTA configuration, using 100k pre-training data and 200 epochs for each training phase

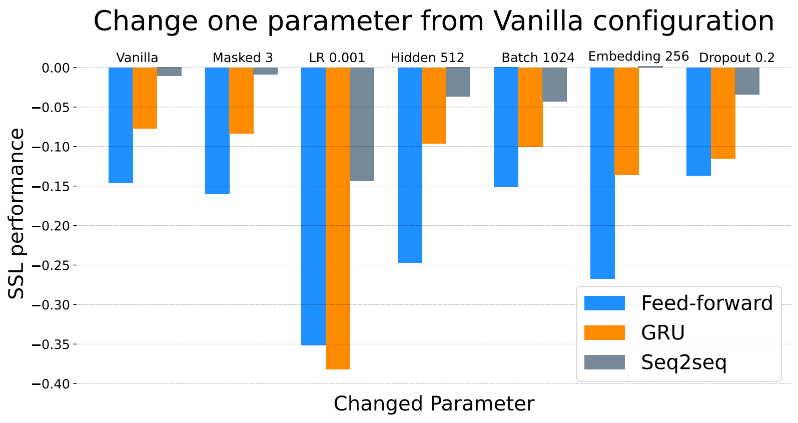

Fig. 4. Changing one parameter at a time from Vanilla configuration, using 25k pre-training data and 200 epochs for each training phase

dropout to 0.2 does not change SSL performance as significantly as it did for the SOTA configuration. This result is unexpected.

Impact of changing two hyperparameters on the SSL performance
To further understand the relationships between hyperparameters, we continue to modify two hyperparameters at a time, from SOTA configuration, and quantify the SSL performance. This tests if patterns emerge showing which combination of hyperparameters generally have the strongest impact on SSL perfor-

mance. Figure 5 shows that both the feed-forward and the Seq2seq model obtain

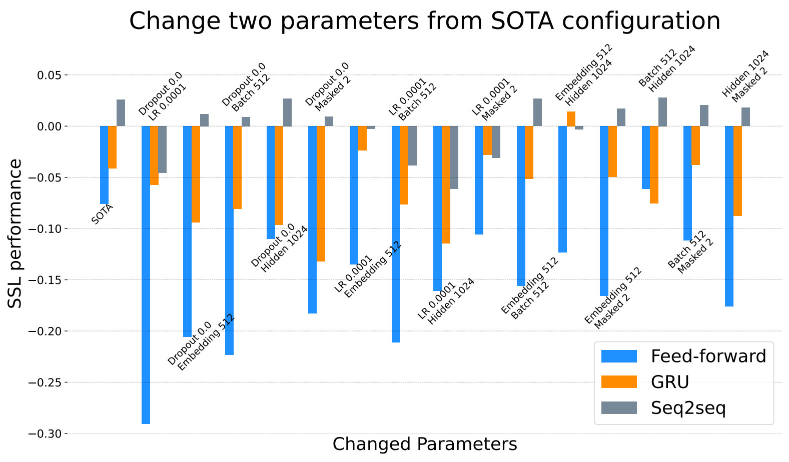

Fig. 5. Changing two parameters at a time from SOTA configuration, using 25k pre-training data and 200 epochs for each training phase while keeping everything else fixed.

the highest SSL performance when modifying batch size from 256 to 512 and hidden size from 512 to 1024. The GRU model obtains the highest SSL performance when embedding size changes from 256 to 512 and hidden size changes from 512 to 1024. The common factor here is that increasing hidden size from 512 to 1024 makes SSL performance the best for all three models. The differences in SSL performance might not exceed the performance variance. Therefore, if the same experiments are rerun, the results can change. Figure 5 also solidifies that the learning rate is a vital hyperparameter to consider for SSL. When the learning rate changes from 0.001 to 0.0001, performance decreases. Similarly, when dropout is modified from 0.2 to 0.0, SSL performance decrease.

3.1 Discussion

The hyperparameters configuration has a significant impact on the SSL performance. Based on the results presented, the hyperparameter configuration significantly impacts SSL performance at least within a fixed number of epochs. The smaller and less sophisticated feed-forward models are frequently more sensitive to hyperparameter modifications. One possible explanation is that less sophisticated models are more dependent on hyperparameters for learning effectively. Potentially, more epochs could allow smaller models to converge more during pre-training and learn more useful knowledge for the downstream classification task. Some hyperparameters have a stronger impact on SSL performance

compared to others. Particularly:
-The learning rate has a significant impact.
Increasing the learning rate can lead to a faster model adjustment to the pre-training task. Because the number of epochs is fixed at 200, increasing the learning rate results in more substantial weight modifications per update. This results in faster convergence of the model. A higher learning rate can allow the model to learn more useful features in fewer epochs, resulting in better downstream performance. However, when tested in the Vanilla configuration, a higher learning rate results in a lower SSL performance. A possible explanation can be that the combination of larger embedding size, hidden size, and no dropout may lead to overfitting on pre-training data.
-The dropout impact on SSL depends on the model.
Dropout is known for lowering overfitting and improving generalization in deep neural networks. Adding dropout results in models learning more general knowledge during pre-training, which improves downstream classification performance on validation data. Surprisingly, with 100k pre-training data, changing the dropout to 0.0 has a minor impact for the GRU and Seq2seq models, while for the feed-forward model, it has a small positive impact. Larger models with recurrent layers have more use of dropout, compared to the feed-forward model, because of a higher tendency to overfit. While for simpler models, dropout results in ignoring useful information. Additionally, if only 25K data are used, changing dropout to 0.0 results in the feed-forward model overfitting and lower performance. Using 100k pre-training data will then prevent overfitting. However, the dropout is less relevant when it comes to the Vanilla configuration. The Vanilla configuration has a lower learning rate, leading to slower learning dynamics. Increasing dropout to 0.2 additionally slows down the learning resulting in lower SSL performance within 200 epochs. The feed-forward model is the exception here, which can be a random incident. It is possible that if more epochs are used, then using a dropout of 0.2 would improve SSL performance for both Vanilla configuration and SOTA configuration.
The amount of pre-training data improves the SSL performance.
For hyperparameter configurations tuned for SSL, SSL performance improves for smaller models as pre-training data quantity increases, while larger models reach a performance threshold, at least with a fixed amount of epochs. Therefore, one should experiment with different data volumes and use as little data as possible. This can significantly save training time. One possible explanation for this is that with recurrent layers a model learns faster, and learns a larger number of useful features from pre-training data. This is because of using more parameters. Therefore, more expansive models containing more parameters require less pre-training data to learn the most useful features during pre-training. This is only speculation. It is possible that Figure 2 looks different with other hyperparameters. Experiments with additional pre-training data are considered future work. For other configurations less suited for SSL, the effect of increasing data quantity is not as significant, particularly for smaller models. This also means that the hyperparameter configuration can be more important than the

amount of pre-training data itself. Using the “wrong” configuration can severely hinder SSL performance even with large amounts of data.

Larger models improve the SSL performance.
Based on the discussion above, larger models generally show higher SSL performance than smaller models. Our experiments show exceptions to this, but these might occur due to parameter choices. Using additional pre-training data, smaller models can catch up with more extensive models regarding SSL performance. However, throughout this entire study, we do not look at absolute text classification performance. Instead, relative text classification performance is considered. In summary, extensive models show higher SSL performance compared to smaller models – particularly with a smaller volume of pre-training data.

3.2 Conclusion

There is a growing interest in SSL research caused by limited labeled data in many domains. However, the setup of SSL neural networks for text classification is cumbersome, frequently based on trial and error, with little knowledge on which setup is beneficial for SSL. Research has shown that SSL does not always improve performance compared to supervised learning. We found that the hyperparameter configuration significantly impacts SSL performance, and the learning rate has the most impact. Hence, experimenting with different hyperparameter configurations can dramatically improve SSL performance. More extensive models often improve SSL performance than smaller models, particularly with a smaller pre-training data quantity. However, as pre-training data size increases, recurrent models generally reach a performance threshold. On the other hand, smaller models can benefit from more pre-training data, especially when hyperparameter configurations are tuned for SSL. Therefore, one should generally experiment using different data volumes for all models. If aiming to achieve the best possible absolute downstream performance, larger and more sophisticated models should be used.

This article explored the impact of hyperparameters, including pre-training data size and model size, on an SSL technique for a text classification task. This exploration improves understanding of which parameters have the most impact on SSL for text classification, making it more manageable to perform SSL work for future NLP projects, particularly if time-restricted. This research also advances the understanding of model size impact on SSL for text classification, enabling a better experience of model selection for SSL designs. With this, we enhance the knowledge of parameter relations, potentially lowering the training time for SSL-based machine learning.

References

1. aladdinpersson: seq2seq_attention. https://github.com/aladdinpersson/Machine-Learning-Collection/blob/master/ML/Pytorch/more_advanced/

Seq2Seq_attention/seq2seq_attention.py (2021), [Online; accessed 3-June-2021]
2. Baevski, A., Edunov, S., Liu, Y., Zettlemoyer, L., Auli, M.: Cloze-driven pretraining of self-attention networks. arXiv preprint arXiv:1903.07785 (2019)
3. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: Can language models be too big? In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. pp. 610–623 (2021)
4. Bishop, C.M.: Pattern recognition and machine learning. springer (2006)
5. Casanueva, I., Temčinas, T., Gerz, D., Henderson, M., Vulić, I.: Efficient intent detection with dual sentence encoders. arXiv preprint arXiv:2003.04807 (2020)
6. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning (09 2006). https://doi.org/10.7551/mitpress/9780262033589.001.0001
7. Dai, A.M., Le, Q.V.: Semi-supervised sequence learning. arXiv preprint arXiv:1511.01432 (2015)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
10. chao ji: Semi-supervised-sequence-learning. https://github.com/chao-ji/semi-supervised-sequence-learning (2021), [Online; accessed 3-June-2021]
11. Lang, K.: Newsweeder: Learning to filter netnews. In: Machine Learning Proceedings 1995, pp. 331–339. Elsevier (1995)
12. Lee, D.: Transfer learning for text classification with tensorflow. https://github.com/dongjun-Lee/transfer-learning-text-tf (2018), [Online; accessed 3-June-2021]
13. Li, Y.F., Zhou, Z.H.: Towards making unlabeled data never hurt. IEEE transactions on pattern analysis and machine intelligence **37**(1), 175–188 (2014)
14. Lien, H., Ledaal, B.V.: Semi-supervised learning for text classification (2020), student project in the course IKT442: ICT Seminar 3 at UiA
15. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
16. Oliver, A., Odena, A., Raffel, C., Cubuk, E.D., Goodfellow, I.J.: Realistic evaluation of deep semi-supervised learning algorithms. arXiv preprint arXiv:1804.09170 (2018)
17. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683 (2019)
18. Singh, A., Nowak, R., Zhu, J.: Unlabeled data: Now it helps, now it doesn't. Advances in neural information processing systems **21**, 1513–1520 (2008)
19. Van Engelen, J.E., Hoos, H.H.: A survey on semi-supervised learning. Machine Learning **109**(2), 373–440 (2020)
20. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461 (2018)
21. You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., Hsieh, C.J.: Large batch optimization for deep learning: Training bert in 76 minutes. arXiv preprint arXiv:1904.00962 (2019)
22. Zhu, X.J.: Semi-supervised learning literature survey (2005)