Proceedings of SIGUL2022 @LREC2022, pages 106–111
Marseille, 24-25 June 2022

Nepali Encoder Transformers: An Analysis of Auto Encoding Transformer Language Models for Nepali Text Classification

Utsav Maskey, Manish Bhatta, Shiva Raj Bhatta, Sanket Dhungel, Bal Krishna Bal
Information and Language Processing Research Lab
Department of Computer Science & Engineering
Kathmandu University,
Dhulikhel, Nepal
bal@ku.edu.np
{um02409118, mb02407218, sb02407118, sd02407618}@student.ku.edu.np

Abstract

Language model pre-training has significantly impacted NLP and resulted in performance gains on many NLP-related tasks, but comparative study of different approaches on many low-resource languages seems to be missing. This paper attempts to investigate appropriate methods for pretraining a Transformer-based model for the Nepali language. We focus on the language-specific aspects that need to be considered for modeling. Although some language models have been trained for Nepali, the study is far from sufficient. We train three distinct Transformer-based masked language models for Nepali text sequences: distilbert-base (Sanh et al., 2019) for its efficiency and minuteness, deberta-base (P. He et al., 2020) for its capability of modeling the dependency of nearby token pairs and XLM-ROBERTa (Conneau et al., 2020) for its capabilities to handle multilingual downstream tasks. We evaluate and compare these models with other Transformer-based models on a downstream classification task with an aim to suggest an effective strategy for training low-resource language models and their fine-tuning.

Keywords: Natural Language Processing, Nepali Language, Language Modeling, Transformers, Auto Encoders

1. Introduction

The Transformer has become the go-to method for neural language modeling. It is highly parallelizable and abides by the scaling laws (i.e. performance gets better in accordance to the number of parameters, dataset size and the amount of compute) (Kaplan et al., 2020). In addition, ULMFiT (Howard & Ruder, 2018) introduced techniques that allowed neural networks to train a base language model which could then be fine-tuned on downstream tasks such as classification, text generation, etc. with much lesser data. Following these techniques, many encoder-based transformer models have achieved state-of-the-art results in text classification including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), XLM (Lample & Conneau, 2019), XLM-RoBERTa (Conneau et al., 2020), ALBERT (Z. He et al., 2018), ALBERT (Lan et al., 2019), ELECTRA (Clark et al., 2020) and DeBERTa (P. He et al., 2020).

However, these models are essentially trained on high-resource languages such as English, French, etc. and sufficient efforts and attention have not been given to low-resourced languages. This is primarily because the transformer model requires huge datasets and hence it is not straightforward and easy task for low-resource languages (Ruder, 2020).

The Nepali language belongs to the Indo-Aryan family which is written in the Devanagari script. It is the official language and lingua franca of Nepal and one of the 22 scheduled languages in India. Furthermore, it is spoken by about a quarter of the population of Bhutan and in Burma and different parts of North East India. According to the 2011 census, there are 16 million native speakers with over 9 million L2 speakers ("Nepali language - Wikipedia", 2022).

Nepali is a free word order language without upper or lower case of the characters. It is written from left to right and follows the Subject Object Verb (SOV) pattern as the sentential grammar structure. There are 33 consonant letters, 11 independent vowel letters and 10 dependent vowel signs or matras in Nepali. The consonant letters may exist independently or in conjunction with dependent symbols (matras, halanta, etc.) to form a compound letter. The halanta symbol (represented by U+094D (्) Devanagari sign Virama in Unicode) is a dependent symbol which is used to suppress the inherent vowel sign in any consonant letter and is mostly used to produce half characters in Nepali. Similarly, there are other dependent symbols including, Chandrabindu (ँ) and Shirbindu (ं) which indicates nasalization of a vowel and consonants respectively. The bisarga (ः) dependent symbol appears in some Nepali words, but they are not usually pronounced. Purna biram (।) marks the end of a sentence, similar to a full stop. The set of digits (०, १, २, ३, ४, ५, ६, ७, ८, ९) are used as numbers in Nepali. ("Nepali alphabet", 2015).

In the context of low-resource language modeling with transformers, Indic-Transformers (Jain et al., 2020) train and benchmark three languages, namely, Hindi, Bengali and Telugu on tasks including classification, POS tagging and Question Answering. Similarly, in line to this, we focus on investigating various approaches to modeling the Nepali language using contemporary transformer models.

As for the Nepali language, attempts have been made to understand the grammatical structure of the Nepali Language in the work of (Bal, 2004a; 2004b). Some notable works related to Nepali language NLP includes, summarization (Mishra et al., 2020), Named entity recognition (Maharjan G., Bal B.K., 2019), etc.

However, not much effort has been made on working with contemporary transformer models. Some encoder-based transformer models including nepaliBERT (Pudasaini, 2022) and NepaliBERT (Rajan, 2021) have been trained for Nepali, whose performance is yet to be analyzed.

In this work, we focus on training three encoder-based transformer models, DistilBERT (Sanh et al., 2019), DeBERTa (P. He et al., 2020) and XLM-R (Conneau et al., 2020) for Nepali. The objective is to find a suitable procedure for training low-resource languages like Nepali.

The contributions of our paper are as follows:

- We train an SPM, Sentence Piece Model (Kudo & Richardson, 2018) for sub-word tokenization of texts. Devanagari characters are different compared to the languages on which most transformer-trained models are trained and therefore, the development of a suitable tokenizer model should be considered. The XLM-R paper (Conneau et al., 2020) shows that they observed negligible loss in performance using SPM as compared to models trained with language-specific pre-processing. This SPM tokenizer will be used for training the language models.
- We train various encoder-based models to compare which transformer architectures are feasible for Nepali Language training.
- We present a comparison of these models by evaluating them on a downstream text-classification task. And since fine-tuning multilingual models is a popular method for modeling a low-resource language, we also reflect the performance of a multilingual model, XLM-R through a comparative study.

2. Background & Related Work

Representation learning aims to learn representations of raw data as useful information for further classification or prediction. Early attempts in this direction account to pre-trained word embeddings on a large and diverse corpus (Mikolov et al., 2013). An inductive transfer is then performed by fine-tuning on top of the learned embeddings that allowed neural networks to train on various NLP tasks. Recurrent neural networks (RNNs) along with usage of techniques including long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and gated recurrent units (GRU) (Chung et al., 2014) on top of learned representations achieved state-of-the-art on many NLP tasks.

The paper, Universal Language Model Fine-tuning (ULMFiT) (Howard & Ruder, 2018) introduced how Causal Language Modeling (CLM) can be pre-trained on neural networks as opposed to word embeddings which used only a single neural network layer for pre-training the diverse corpus. Recent methodologies, however, use Masked Language Modeling (MLM) for pre-training encoder-based transformers.

2.1 Masked Language Modeling (MLM)

As opposed to CLM, where the model attempts to predict the next sequences in a sentence, MLM attempts to predict the middle words in the sentence (Devlin et al., 2019). This ensures that the model learns contextual word representations and the learning is bi-directional. Specifically, given a sequence of text $X = \{x_i\}$, X is corrupted into $\tilde{X}$ by masking some percentile of its tokens at random and then a language model is trained to re-construct X by predicting the masked tokens $\tilde{x}$. The percentile of tokens masked on the original BERT is 15%, however in the work by (Wettig et al., 2022), they suggest that 20% masking performs better for small-sized transformer models whereas huge models favor MLM probability as high as 40%.

2.2 Auto Encoding Transformers

Transformer models are based on attention mechanisms (Vaswani et al., 2017) which consists of Encoder and/or Decoder sub-architectures. The Encoder gets good at understanding the input text and extracting its feature representations, whereas the Decoder gets good at predicting the targeted output sequences. The Encoder part of transformer architecture can independently be used as a many-to-one sequential model, where the sequence length of input tokens may vary provided that the model's output size remains constant. Such auto-encoding transformers pre-trained with language modeling objectives are the state-of-the-art models on the GLUE benchmark (Wang et al., 2018) which measures Natural Language Understanding (NLU) across several tasks of varying difficulty. Auto encoding models that use BERT-like architecture (Devlin et al., 2019) still dominate research and industry when fine-tuned on NLU tasks such as text classification, named entity recognition, and question answering.

3. Experimental Workflow

Our experiments are performed on Auto-encoding Transformers. As for training the language models, we set up a pipelined procedure consisting of data collection, tokenization, language model training and its comparative evaluation on a downstream classification task.

3.1 Nepali Text Data

With the objective of training language models from scratch, we use monolingual unlabeled Nepali texts. We gathered 13 million text sequences (phrases and paragraphs) by combining and de-duplicating three publicly available datasets: OSCAR (Suárez et al., 2020), cc100 dataset (Conneau et al., 2020) and the iNLTK dataset (Arora, 2020).

3.2 Tokenization

Tokenizing Devanagari texts differs from that of English texts due to different ways of combining consonants, vowels and vowel modifiers. For example, the compound letter, 'लु' (ल + ु) is formed by combining the free form character, 'ल' and the vowel-sign, 'ु'. However, the tokenizer used by BERT and the original Multilingual BERT removes some vowel symbols and other dependent symbols, and only the free form character remains. For example, the letter 'लु' is tokenized as:

'लु' (ल + ु) → 'ल' ('ु' is removed)

The use of Unicode normalizations causes this behavior in languages with non-Latin alphabets. When tokenizing a decomposed character sequence into multiple pieces, we may break the original meaning of the character. This creates ambiguities in the Nepali Language.

For example, the word, 'फ्लु' can be tokenized as:

	Before Tokenization	Tokenized
Decomposition:	फ्लु (फ + ् + ल + ु)	फल (फ+ ल)
Meaning:	Flu	Fruit

Table 1: Decomposition of the word, ''फ्लु''

The removal of the vowel, 'ु' and the halanta, '्' changes the original meaning of the word and causes ambiguity resulting in two different words having the same meaning.

We opted for a Sentence Piece Model (Kudo & Richardson, 2018) for training the tokenizer on the dataset that we collected. This approach is also used by the XLM-R for training multilingual models.

As for testing the tokenizers, we consider two sub-word tokenization approaches:

3.2.1 WordPiece Tokenizer

WordPiece tokenizer is used by nepaliBERT (Pudasaini, 2022) and NepaliBERT (Rajan, 2021). WorldPiece tokenization distinguishes workpieces at the start of a word from pieces starting in the middle (Song et al., 2020). The latter start with a special symbol '##' in BERT, which is called the suffix indicator. For example, the word चन्द्रागिरिमा may be tokenized as ['चन्द्रागिरि', '##मा'].

3.2.2 Sentencepiece Tokenizer (SPM)

SentencePiece tokenizer treats the input texts just as a sequence of Unicode characters. Even the whitespace is handled as a normal symbol. SentencePiece first escapes the whitespaces with a meta symbol, '_' (U+2581) and tokenizes the input into an arbitrary sub-word sequence.

Input Text : "फ्लुको कारणले हुने पहिलोनेपाली भवकृष्ण भट्टराई"	
Tokenizer	Tokenized output
Shushant/ nepaliBERT	['फल', '##को', 'कारण', '##ल', 'ह', ##न','पहिलो', '##न', '##पाली', 'भव', '##क', '##षण', 'भट', '##टर', '##ाई']
R4J4N/ NepaliBERT	['फ्लु', '##को', 'कारणले', 'हुने', 'पहिलो', '##नेपाली', 'भव', '##कृष्ण', 'भट्टराई']
Sentence Piece Model [Ours]	['_फ्लु', 'को', '_कारणले', '_हुने', '_पहिलो', 'नेपाली', '_', 'भव', 'कृष्ण', '_भट्टराई']

Table 2: Comparison of tokenizer outputs

We observe that the approach used by nepaliBERT frequently misses the dependent symbols. The NepaliBERT tokenizer performs quite well, but we choose to use the SPM tokenizer for its flexibility in generating text sequences on auto-regressive transformers. The tokenizer model is trained with a vocabulary size of 24,576 tokens. We use this tokenizer for all the language models that are trained.

3.3 MLM Training Feasibility Test

With the dataset and tokenizer developed, we train some of the popular language models and analyze the performance based on training data size, training time and computational resource constraints. We set a baseline cut-off perplexity of 54.598 (i.e. training loss of 4.0) and perform a constrained training in order to determine suitable models for training the language model.

Following models are considered for the constrained training:

- De-berta-base (P. He et al., 2020)
- Distilbert-base (Sanh et al., 2019)
- XLM-roberta (XLM-R) (Conneau et al., 2020)

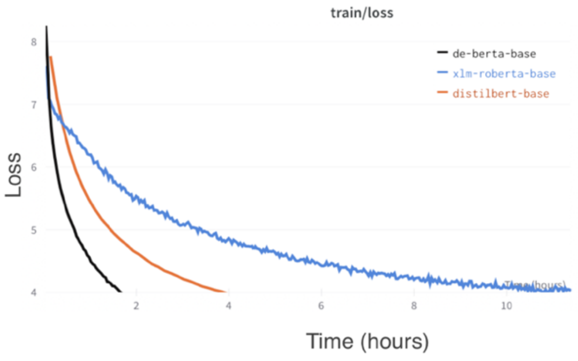


Figure 1: Training loss vs. Time

Model	Batch Size	MLM Probability	Time taken (hh:mm)	No. of training samples
distilbert	28	15%	3:59	406,000
de-berta	6	20%	**1:39**	546,000
xlm-r	1	15%	9:16	**154,000**

Table 3: Summary of the feasibility test. Comparison between the models for reaching the baseline perplexity

The DeBERTa model, despite being trained on a difficult task of MLM probability of 20%, reaches the targeted perplexity the fastest and also by a large margin. The xlm-roberta model, which is trained stochastically (with most training steps), reaches the baseline when trained with the least amount of data; but the training is noisy and the computational training requirement is massive. Therefore, we discarded xlm-roberta-base for its huge architecture and constrained computational training.

We hence decide on training two models: DeBERTa model that focuses on attaining the best performance, and the DistilBERT model which focuses on being lightweight with capabilities of on-device computations.

3.4 Language Model Pre-Training

We proceed with the training of distilbert and de-berta models for 5 epochs on the dataset that we gathered and obtain the following results:

Model	Train/loss	Batch size	Perplexity (eval)
Distilbert-base	2.6412	28	12.3802
De-berta-base	**1.9375**	6	**6.4237**

Table 4: Summary of LM training for 5 Epochs with MLM probability of 20%

In terms of perplexity, we obtain better results using the deberta model. The distilbert model, despite training much faster, produces a respectable performance. We further evaluate the performance of language models by comparing them on a downstream classification task in the next section.

4. Results and Analysis

4.1 Text Classification

The classification task performance evaluation is performed on the "16 Nepali News" dataset (Chaudhary & Sabin, 2017). The dataset consists of approximately 14,364 Nepali language news documents, partitioned (unevenly) across 16 different newsgroups: Auto, Bank, Blog, Business Interview, Economy, Employment, Entertainment, Interview, Literature, National News, Opinion, Sports, Technology, Tourism, and World.

We evaluate Nepali Language Models and compare them in terms of accuracy. The evaluation is performed with varying hyperparameters and for a number of epochs before the models tend to overfit. The following models are considered for the evaluation:

- De-berta-base [Ours]
- Distilbert-base [Ours]
- Shusant/nepaliBERT (Pudasaini, 2022)
- Rajan/NepaliBERT (Rajan, 2021)
- XLM-roberta-base (Conneau et al., 2020)

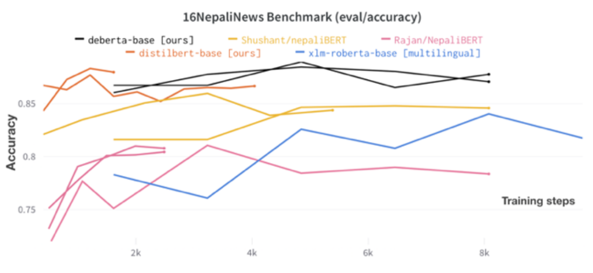


Figure 2: Evaluating language models on "16 Nepali News" Dataset. Training are performed with varying hyperparameters. Each progression in the x-axis represents an Epoch.

Model	Epoch	Train steps	Highest Accuracy
deberta-base [ours]	3	4845	**88.93%**
distilbert-base [ours]	3	**1212**	88.31%
nepaliBERT	4	3231	85.96%
NepaliBERT	6	3230	81.05%
XLM-Roberta	5	8075	84.02%

Table 5: Highest accuracy attained by the models

All the models cross the baseline accuracy of 80%. The de-berta model attains an accuracy of 88.93% which highlights the significance of training domain-adapted language models. Distilbert attains a respectable accuracy of 88.31% with the least number of training steps, which implies that the model trains the fastest for downstream tasks. The performance difference compared to the de-berta model is marginal, and being the smallest and the lightest model, it best suits a production environment with computational constraints.

We note that all models except XLM-roberta-base are domain-adapted to the Nepali language. We can see a general trend of domain-adapted models reaching their peak accuracy on the second or third epoch, whereas multilingual models prefer more training epochs. As a result, domain-adapted language models accelerate downstream task training.

5. Language Model Training and Fine-tuning Approach

We approached language model training and its finetuning with the following considerations:

5.1 Tokenizer:

Language-specific pre-processing of text benefits the training of language models. Modeling a Sentence Piece Model (SPM) tokenization performs comparably with the language-specific approach and can be used for a variety of languages. Using this approach, lesser focus may be given to the language-specific aspects and one may train language models on a language whose structure may not be familiar to them.

5.2 MLM with Less Data Resources

Considering limited data availability constraints, models are trained for multiple epochs by increasing the number of masked tokens on every preceding epoch. This progressive masking approach adds noise to the data and gradually increases the training difficulty in the later epochs. Also, de-duplicating the dataset improves the model performance.

5.3 Fine-tuning on a Downstream Task

As for fine-tuning the language model on a downstream task, our models performed optimally when trained with a learning rate of 2e-5 or 3e-5 with a linear learning rate scheduler.

6. Conclusion

In this work, we have analyzed the need and effectiveness of pre-training Transformer language models for Nepali. We focused on the language-specific aspects that are needed to be considered while modeling a low-resourced language and undertook approaches to tackle data availability constraints. We trained two auto-encoding transformer models, the DeBERTa model that focuses on attaining the best performance, and the DistilBERT model which focuses on being lightweight with capabilities for on-device computations. Our approaches are compared with other transformer models by evaluating them in terms of downstream classification accuracy and the result highlights the need of domain-adapted Language Model training on low-resourced languages.

7. Acknowledgements

This research was supported in part through computational resources provided by the Supercomputer Center Kathmandu University, which was established with equipment donated by CERN.

8. References

Arora, G. (2020). i{NLTK}: Natural Language Toolkit for Indic Languages. *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, 66–71. https://doi.org/10.18653/v1/2020.nlposs-1.10

Bal, B. K. (2004a). Structure of Nepali Grammar. *PAN Localization, Working Papers 2004-2007*, 332–396.

Bal, B. K. (2004b). A Morphological Analyzer and Stemmer for Nepali. *PAN Localization, Working Papers 2004-2007*, 324–331.

Biewald, L. (2020). Experiment Tracking with Weights and Biases. *Software available from wandb.com*. https://www.wandb.com/.

Chaudhary, A., & Sabin. (2017). 16NepaliNews Corpus. https://github.com/sndsabin/Nepali-News-Classifier

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014.*

Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *8th International Conference on Learning Representations*.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised Cross-lingual Representation Learning at Scale. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*.

He, P., Liu, X., Gao, J., & Chen, W. (2020). Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

He, Z., Bao, S., & Chung, A.C. (2018). 3D Deep Affine-Invariant Shape Learning for Brain MR Image Segmentation. *DLMIA/ML-CDS@MICCAI*.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. https://doi.org/10.18653/v1/p18-1031

Jain, K., Deshpande, A., Shridhar, K., Laumann, F., & Dash, A. (2020). Indic-transformers: An analysis of transformer language models for Indian languages. *arXiv preprint arXiv:2011.02323*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *EMNLP*.

Lample, G., & Conneau, A. (2019). Cross-lingual Language Model Pretraining. *NeurIPS*.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ArXiv, abs/1909.11942*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv, abs/1907.11692*.

Maharjan G., Bal B.K., R. S. (2019). Named Entity Recognition (NER) for Nepali. *Communications in Computer and Information Science*, *1084*(Creativity in Intelligent Technologies and Data Science).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*.

Mishra, K., Rathi, J., & Banjara, J. (2020). Encoder Decoder based Nepali News Headline Generation. *International Journal of Computer Applications*, *175*, 975–8887. https://doi.org/10.5120/ijca2020920735

Nepali language - Wikipedia. En.wikipedia.org. (2022). Retrieved 22 May 2022, from https://en.wikipedia.org/wiki/Nepali_language

Nepali alphabet. nepalilanguage.org. (2015). Retrieved 22 May 2022, from https://nepalilanguage.org/alphabet

Pudasaini, S. (2022). Pretraining Nepali Masked Language Model using BERT Architecture. *3rd International Conference on Natural Language Processing, Information Retrieval, and AI*

Rajan. (2021). *NepaliBERT.* https://huggingface.co/Rajan/NepaliBERT

Ruder, S. (2020). *Why You Should Do NLP Beyond English*. https://ruder.io/nlp-beyond-english/

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing.*

Song, X., Salcianu, A., Song, Y., Dopson, D., & Zhou, D. (2021). Fast WordPiece Tokenization. *EMNLP*.

Ortiz Suarez, P., Romary, L., & Sagot, B. (2020). A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages. *ACL*.

Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *ArXiv, abs/1706.03762*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). {GLUE}: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *International Conference on Learning Representations*.

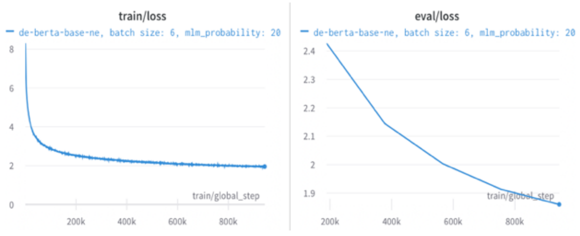
Wettig, A., Gao, T., Zhong, Z., & Chen, D. (2022). Should You Mask 15% in Masked Language Modeling? *ArXiv, abs/2202.08005*.

9. Appendix

A. Training of Language Models

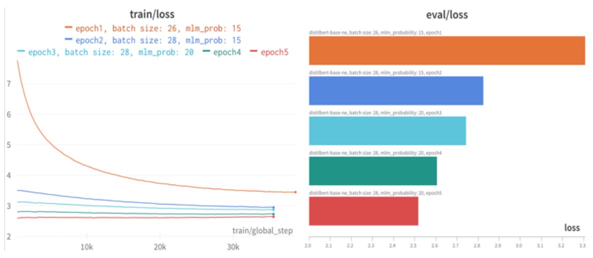
In this section we show some of the plots of pre-training the language models. Weights and Biases (Biewald, 2020) platform was used to track the training process.

A1. DeBERTa Model



A2. DistilBERT Model

Distilbert Model trained with progressive masking.


train/loss
epoch1, batch size: 26, mlm_prob: 15
epoch2, batch size: 28, mlm_prob: 15
epoch3, batch size: 28, mlm_prob: 20
epoch4
epoch5
train/global_step
10k
20k
30k
eval/loss
distilbert-base-ne, batch size: 26, mlm_probability: 15, epoch1
distilbert-base-ne, batch size: 28, mlm_probability: 15, epoch2
distilbert-base-ne, batch size: 28, mlm_probability: 20, epoch3
distilbert-base-ne, batch size: 28, mlm_probability: 20, epoch4
distilbert-base-ne, batch size: 28, mlm_probability: 20, epoch5
loss