Choose Your Transformer: Improved Transferability Estimation of Transformer Models on Classification Tasks

Lukas Garbas * **Max Ploner** *,† **Alan Akbik** *,†

* Humboldt-Universität zu Berlin
† Science Of Intelligence
<firstname>.<lastname>@hu-berlin.de

Abstract

There currently exists a multitude of pre-trained transformer language models (LMs) that are readily available. From a practical perspective, this raises the question of which pre-trained LM will perform best if fine-tuned for a specific downstream NLP task. However, exhaustively fine-tuning all available LMs to determine the best-fitting model is computationally infeasible. To address this problem, we present an approach that inexpensively estimates a ranking of the expected performance of a given set of candidate LMs for a given task. Following a layer-wise representation analysis, we extend existing approaches such as H-score and LogME by aggregating representations across all layers of the transformer model. We present an extensive analysis of 20 transformer LMs, 6 downstream NLP tasks, and various estimators (linear probing, kNN, H-score, and LogME). Our evaluation finds that averaging the layer representations significantly improves the Pearson correlation coefficient between the true model ranks and the estimate, increasing from 0.58 to 0.86 for LogME and from 0.65 to 0.88 for H-score.

1 Introduction

Current state-of-the-art approaches for classification tasks such as sequence labeling or text classification rely on pre-trained transformer language models (LMs; Devlin et al., 2018; Clark et al., 2020; He et al., 2021a). These models are fine-tuned on task-specific training data. However, with the rising availability of such pre-trained LMs (e.g. through model hubs; Wolf et al., 2019), this raises the practical challenge of choosing which pre-trained transformer to use for a specific task.

However, exhaustively evaluating all available transformer LMs is often infeasible due to the high computational costs of fine-tuning. Further, since fine-tuning is sensitive to hyperparameters such as the learning rate, a hyperparameter search would need to be executed for each candidate LM. Since the concrete choice of the pre-trained model can have a significant impact on the downstream performance, we argue that there is a need for accurate and computationally efficient *transferability estimation*. Here, a given list of candidate LMs is automatically ranked with regard to their expected performance on a downstream task.

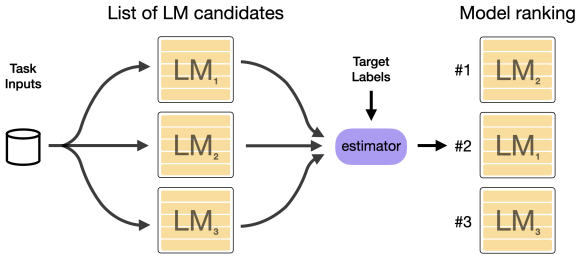
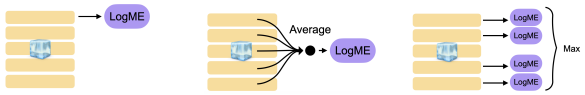


Figure 1: LMs are ranked based on the performance estimated using the encoded samples and the target labels of the downstream task.



(a) Last layer (b) Layer mean (c) Best layer

Figure 2: Illustration of different ways to aggregate information across multiple layers.

Utilizing frozen representations. One promising line of research performs this estimation by extracting static representations from the LM for all data points in the training and testing data. By excluding the parameters of the transformer from further fine-tuning (thus "freezing" the weights of the transformer), the bulk of the computational costs are avoided. Various estimators based on frozen features have been proposed. For instance, H-score (Bao et al., 2019) considers the inter-class variance and feature redundancy, while the Logarithm of Maximum Evidence (LogME; You et al.,

Findings of the Association for Computational Linguistics ACL 2024, pages 12752–12768

2021) assumes a linear transformation between representations and layers. The latter was empirically shown to be a strong baseline for task-specific ranking of LMs (Bassignana et al., 2022).

Figure 1 illustrates how performance estimators such as LogME or H-score are used: All task samples are encoded using the frozen LM. Using the target labels, a score is estimated for each of the models from which a ranking is produced.

Contributions. Building upon the insights from previous studies on transferability estimation, this work proposes methods to enhance the accuracy of such estimators. Our improvements are derived from prior findings in language model probing (Rogers et al., 2020), where it was shown that different encoder layers encode different information that can be very task-specific. They demonstrate that, given the variety of NLP datasets and language models available, relying solely on the features of the last layer (see Figure 2a) is insufficient.

Following this observation, this paper evaluates alternative strategies for extracting representations from frozen transformers with regard to their usefulness in transferability estimation. In more detail:

1. We present an analysis of transferability estimation on 6 NLP tasks using 20 LMs that vary in size and pre-training objective. For each LM and downstream task, we perform a hyperparameter search and full fine-tuning to obtain a gold ranking to compare against. We then evaluate 4 different estimators (linear probing, kNN, LogME and H-score) and compare their baseline performance to the gold ranking using Pearson correlation.

2. We investigate alternative layerwise aggregation strategies, including averaging representations across all layers (see Figure 2b) and separately identifying the best layer in each LM (see Figure 2c).

3. We present further analysis on per-layer performance of transformer LMs on downstream tasks. Our analysis shows that in frozen models, intermediary layers are often better suited for specific tasks.

Our experiments show that a simple strategy of averaging representations significantly improves transferability estimation for all considered estimators. In particular, we find that the Pearson correlation coefficient between the true model ranks and the estimate increases from 0.58 to 0.86 for LogME, and from 0.65 to 0.88 for H-score. We believe that our results indicate the viability of using the H-score in combination with layerwise mean representations for accurate and computationally efficient transformer ranking.

2 Related work

Our work builds on previous research in the fields of pre-trained model selection and automated machine learning. The following subsections give a short introduction to each of these areas.

2.1 Pre-trained Model Transferability Estimation

With the increased availability of various pre-trained models, both the natural language processing and the computer vision community began exploring approaches to estimate the performance of different encoder models without fine-tuning them on a downstream task.

H-score. Bao et al. (2019) proposed the *H-score* which estimates the transferability of features f learned on a source to a target task (with target samples X and labels Y) by considering the inter-class variance and feature redundancy. This is captured by the following equation:

$$\mathcal{H}(f) = \mathrm{tr}(\mathrm{cov}(f(X))^{-1}\mathrm{cov}(\mathbb{E}_{P_{X|Y}}[f(X)|Y]))$$

Low featured redundancy $\mathrm{tr}(\mathrm{cov}(f(X)))$ and high inter-class variance $\mathrm{tr}(\mathrm{cov}(\mathbb{E}_{P_{X|Y}}[f(X)|Y]))$ lead to a high H-score of the source feature f.

H-score was improved by Ibrahim et al. (2023) introducing adding regularization to the covariance matrix $\mathrm{cov}(f(X))$. This regularization stabilizes the inversion of the matrix which is crucial when using high-dimensional feature vectors and a small number of samples as H-score had been reported to struggle when being applied in the small data domain (Nguyen et al., 2020). After applying the regularization to H-score, it was shown to be a competitive alternative to the state-of-the-art LogME.

LogME. You et al. (2021) use the representations of frozen models $F \in \mathbb{R}^{|D| \times h}$ (i.e., embeddings with dimensionality h) to assess the transferability between the encoded inputs and their target labels $y \in \mathbb{R}^{|D|}$. To this end, LogME approximates the true probability density $p(y|F)$ by assuming a linear transformation w mapping the encoded representation to the labels ($F \rightarrow y$). A maximum

likelihood estimate could be computed by training a linear probe with optimal parameters w^* resulting in the likelihood $p(y|F, w^*)$ which, in turn, could then be used as an estimate for the suitability of features. However, since the linear model trained directly on the task data is likely to overfit, You et al. (2021) propose to use the evidence (i.e. marginalized likelihood over all values of w):

$$p(y|F) = \int p(y|F, w)p(w)dw$$

By assuming w and y to be drawn from parametrized, isotropic Gaussian distributions, this can be solved using iterative evidence maximization (You et al., 2021). The normalized logarithm of the maximized evidence (LogME) is then used to estimate the model's performance on the downstream task after fine-tuning it. To extend this to classification tasks, You et al. (2021) propose to treat them as regression problems with one-hot encoded target vectors. The per-class evidence is then averaged to retrieve a score representing the transferability to the multi-class classification problem.

LogME has been preliminarily explored within the GLUE benchmark (Wang et al., 2018; You et al., 2021). Subsequently, LogME was refined by extending the experiments to a broader spectrum of language models (7 LMs) and NLP tasks (covering 8 datasets; Bassignana et al., 2022). As an enhancement of the original work, the results show that text classification tasks benefit from utilizing the mean of all subword representations rather than relying solely on the $[CLS]$ token representation.

The previous work focused on the mapping between the features of the frozen model and their corresponding ground truth labels, specifically by not employing a trainable linear layer to reduce overfitting. However, previous works rely solely on the topmost layer features as representations of the complete model.

Other transferability estimates. Following the initial publication of the H-score, other works in the area of computer vision proposed approaches to estimate the joint distribution of pre-trained labels and target labels (Nguyen et al., 2020; Tran et al., 2019). Though fast to compute, these methods are not suited for transferring pre-trained models to classification tasks. Furthermore, these approaches also do not operate on the feature level and, therefore, cannot profit from our proposed (feature level) addition.

2.2 Layer-wise Representation Analysis

We use the term layer-wise representation analysis to describe studies that examine pre-trained models to see how their representations change through the layers and how these changes connect to different linguistic properties.

A significant part of this research was performed using the `BERT` model under the term *BERTology* (Rogers et al., 2020). It stands to reason that `BERT` encodes local information about word tokens well on its lower layers, but switches to a hierarchically-oriented encoding on higher layers (Lin et al., 2019), leading to more context-specific representations (Ethayarajh, 2019). Studies across various tasks, datasets, and methodologies broadly agree that syntactic information is predominantly found in the middle layers of `BERT` (Hewitt and Manning, 2019; Goldberg, 2019; Jawahar et al., 2019; Liu et al., 2019a).

While `BERT` has been extensively examined, there is little work on language models with different pre-training tasks. Fayyaz et al. (2021) show that given a different pre-training objective, the maximum amount of linguistic knowledge can be accumulated in different parts of the model. Specifically, they show that, unlike `BERT`, `XLNet` (Yang et al., 2019) encodes linguistic information in the earlier layers during pre-training, while `ELECTRA` tends to carry this information to the higher layers.

This indicates that solely depending on the same layer for comparisons between different models can lead to inaccuracies.

2.3 Automated Machine Learning

The field of automated machine learning (AutoML) aims to reduce the manual work required to train and deploy ML systems. Previous research has often focused on network architecture search and hyperparameter tuning (He et al., 2021b). However, current state-of-the-art NLP pipelines typically involve pre-trained language models (Saleem and Kumarapathirage, 2023). Hence, selecting the pre-trained model is an integral part of this process.

Recent research has started to acknowledge this fact. Drori et al. (2019) propose to embed dataset meta-data using an embedding network to find a pipeline that performed well on a similar task. Desai et al. (2022) propose using synthetic data and a meta-model to predict suitable tuning settings and the best-performing model out of three candidate models. Saleem and Kumarapathirage (2023) fol-

low a similar approach and test multiple models (including random forest and K-Nearest Neighbors) to predict the best performing model given a long list of meta-features. Zhang et al. (2023b) propose AutoML-GPT, a method using a GPT model to select models for fine-tuning based on datasets and model cards (Mitchell et al., 2019) which describe the dataset structure and domain and model architecture, training data and the intended use respectively. Zhang et al. (2023a) propose a similar approach they call "MLCopilot": Using an LLM to make suggestions for configuring an AutoML system.

None of these approaches uses the model's representations of the samples in the current dataset but instead rely on entirely meta-learned knowledge from other examples. While there are many merits to these types of approaches (such as having the ability to propose fine-tuning configurations), they all rely on extensively trained models and it is not clear how much they will be able to generalize to new domains and new types of models.

These approaches are largely orthogonal to the representation-based approaches and hence may complement the approaches discussed in the remainder of the paper. Model transferability estimation could either add additional information to each of these approaches or profit from a narrowed-down candidate list produced by either of these approaches. Since the scope of these approaches deviates significantly from ours, we do not include them in our evaluation.

3 Performance Estimation

The goal of model selection is to estimate the performance of various language models on a given downstream task and rank them such that the order closely aligns with the true order of their downstream performance. In an ideal scenario, for any given classification dataset D, and a list of pre-trained models $\{T_m \mid m = 1, \ldots, M\}$, there would be a method capable of ranking these models without the necessity of fine-tuning each model T on D. This approach should offer not only a speed advantage over the exhaustive fine-tuning of all potential model candidates, but also maintain sufficient accuracy to reliably identify the most suitable models.

3.1 Estimators

The straightforward approach to estimating the performance of a model is to train a linear layer on top of a frozen LM and evaluate its performance. In addition to a linear layer, we use kNN, H-score and LogME.

Linear Probing. Training a linear head requires hyperparameter search (i.e. optimizer type, learning rate, epochs, batch size) and the training can take up to 50 epochs to reach convergence. We use linear probing to confirm the results of previous work and show that LogME and H-score perform faster and are more accurate at ranking models.

K-Nearest Neighbours. kNN offers a speed advantage over linear probing due to the absence of trainable parameters. It operates with only a single hyperparameter, k. We test kNN as an alternative method to LogME. The benefit of kNN can be seen in the implementation since it is a widely supported algorithm by numerous libraries such as scikit-learn (Pedregosa et al., 2012), and can easily be used in any new model selection tool.

H-score. We use the improved version of H-score (Ibrahim et al., 2023, see Section 2.1 for more details), as it was shown to be comparable to LogME. For the sake of simplicity, we will refer to this as "H-score" from here on. The authors provide a fast implementation and show that their approach outperforms the initially published version by Bao et al. (2019).

LogME. LogME (see Section 2.1) is currently the most accurate method that can be used for model performance estimation. Since it only has two trainable parameters α and β, it can be estimated just as fast as fitting a kNN model. We report the scores for all our proposed approaches using LogME and aim to improve its current performance.

3.2 Layer-wise Feature Aggregation

We argue that the performance of just the topmost layer is not sufficient to predict the LMs full potential when it is fine-tuned. Given different pre-training objectives, LMs can have diverse layer-wise information that can strongly harm the produced ranking if only a specific layer is chosen in each of the LM candidates. To incorporate deeper layer features, we examine two ideas.

Last layer. During fine-tuning one typically attaches a new classification head to the last layer of the transformer model. Hence, it appears to be a natural choice to pick the representation produced by this last layer when estimating the performance of a given downstream task (You et al., 2021; Bassignana et al., 2022).

Layer mean. As a simple approach, we use the mean of all hidden representations. As summarized in earlier works (Rogers et al., 2020), different layers of a language model encode different aspects of linguistic information, with lower layers capturing basic syntactic patterns and higher layers encoding more complex semantic relationships. By averaging representations of all layers, we integrate a broad spectrum of linguistic features, potentially leading to a more robust and generalizable representation. In addition, different language models may have variations in how linguistic information is distributed across their layers (Fayyaz et al., 2021). Averaging the representation across all layers when comparing frozen models can normalize these differences and ensure consistency across model architectures.

Avg. of scores. As an alternative to averaging all layers into a single representation, we first estimate the performance of each layer individually and then compute the average of all estimated scores. Though this approach has the significant downside that it requires multiple computations of the estimation score, it could yield further insides into how the information across the layer should ideally be aggregated.

Best layer. To eliminate the possibility of unhelpful layers adding noise to the representation, we include a variant that searches for the layer that is estimated to have the highest performance in the model. We run a separate estimation for each layer and use the highest of the resulting scores. This should select the layer that produces the most relevant features for the task. By evaluating the best-performing layer, each model is assessed based on its potentially optimal contribution.

4 Experimental Setup

We fine-tune each model on specific downstream tasks to assess their performance. For each dataset, we arrange the models in ascending order by their fine-tuned scores to accurately determine their rankings. Then, we compare the scores of each estimator by comparing them to the true ranking. In the following, we describe the selection of language models and datasets.

	Dataset	Task	Domain	Size	|Y|
TC	Airline	Sentiment	Twitter	11.5K	2
	CoLA	Linguistic Accept.	General	10.5K	2
	SST-2	Sentiment	General	67K	2
NER	CoNLL	NER	General	22K	4
	SciNER	NER	Science	650	7
	CrossNER	NER	Science	2K	17

Table 1: Six datasets were used in this study: Three text classification tasks (TC) and three named entity recognition tasks (NER). The performance on CoLA is typically measured using the Matthews Correlation Coefficient (MCC). The other tasks are measured using a micro-averaged F1-score. |Y| refers to the number of classes or labels.

Language Models. We select 20 pre-trained language models from the Transformers Wolf et al. (2019) library's model hub, expanding upon the most recent work by Bassignana et al. (2022), which utilized 7 different models. The selection was broadened to include models that were pre-trained with other tasks than masked language modeling (MLM), such as `DeBERTa` (He et al., 2021a), `SpanBERT` (Joshi et al., 2019), and two `sentence-transformer` models: `All-MPNet-base` and `All-MiniLM` (Reimers and Gurevych, 2019). The complete list of models used in the experiments, along with their number of parameters and pre-training tasks, can be found in Table 5 in the appendix.

Datasets. We select six classification datasets for the experiments (see Table 1 for an overview). We use three datasets already used in the previous work (Bassignana et al., 2022): Airline Sentiment, SciNER (Luan et al., 2018), and CrossNER (Liu et al., 2020). Additionally, we incorporate three datasets commonly used for evaluating language models: CoLA (Warstadt and Bowman, 2019), SST-2 (Socher et al., 2013), CoNLL-03 (Sang and Meulder, 2003).

Fine-tuning. The state-of-the-art approach to achieve the best performance on most of the NLP datasets is full language model fine-tuning. We use the scores of fine-tuned models as the reference ranking. To fine-tune each model we use the Flair framework (Akbik et al., 2019).

Dataset	Size	Full FT	Probing	H-score layer mean	H-score best layer
Airline	11.5K	498s	260s	28s	40s
CoLA	10.5K	264s	96s	9s	16s
SST-2	67K	1,515s	747s	75s	103s
CoNLLL	22K	2,419s	233s	60s	130s
SciNER	2K	341s	44s	10s	28s
CrossNER	650	62s	31s	6s	19s

Table 2: Runtimes for a subset of the estimation techniques on each dataset using a single model (bert-base). We report the full duration for each of the fine-tuning runs. The number of required epochs to get the best fine-tuning perform differs between the datasets (see the Section A in the appendix). Since the runtimes of H-score and LogME are very similar, we only report it for H-score here. Similarly, the runtime for layer mean and last layer as well as best layer and avg. of scores across layers are approximately equal. In the experiments, we use a GeForce RTX 3090 GPU, 24268MiB.

Evaluation. We assess the effectiveness of each layer-wise aggregator (i.e. layer mean and best layer) given different estimators (i.e. kNN, H-score, and LogME). We evaluate each setup for ranking language models (LMs) based on their fine-tuned performances using Pearson ρ correlation coefficient as the primary metric. We report the correlation coefficients, comparing the transferability estimates of the 20 models with their fine-tuning scores. Additionally, scores using Spearman's rank correlation and Weighted Kendall's τ are provided in the appendix.

5 Results and Discussion

We report the Pearson ρ correlations between the true fine-tuned and the estimated performances in each of the setups in Table 3. We distinguish between different estimation strategies: Linear probing, kNN, H-score, and LogME. Furthermore, we report the expected correlation for each estimator, derived from different layer-wise aggregators, by averaging the outcomes across all six datasets.

Low performance when using the last layer. Due to increasing the number of models to 20, we report a decreased correlation when only the last layer is used (compared to Bassignana et al., 2022). When performing the best layer search and estimating each layer in language models, we notice that models such as `ELECTRA` and `sentence-transformers` underperform on NER tasks using the last layer compared to their deeper layers . This leads to an incorrect order of their final rankings (for detailed plots and best layer indices, see Figure 4 and Table 6 in the appendix).

Layer-aggregation improves all tested estimation methods. By selecting the best-performing layer within the encoder, we achieve an improvement in kNN's performance by 0.54, elevating its ρ value from 0.14 to 0.68. Similarly, LogME's performance is enhanced by 0.26, increasing its ρ from 0.58 to 0.84 and H-score by 0.23 (from 0.65 to 0.88).

The performance difference between the layer mean and the best layer is negligible. Specifically, when using LogME, the difference is merely 0.0086, with the layer mean surpassing the best layer in two of the six datasets (namely SciNER and CoLA). Thus, the layer mean approach is a viable alternative to selecting the best layer, offering comparable performance across datasets, with certain datasets showing benefits while for other datasets the performance difference remains insignificant.

Aggregating layer-wise representations using the layer mean also shows superiority against averaging between the scores of each layer with a difference of 0.88 vs. 0.82 for H-score, 0.86 vs. 0.80 using LogME, and 0.63 vs. 0.63 for kNN.

Since estimation with the layer mean representation is more time-efficient than identifying the optimal layer for each encoder or averaging layer-specific scores, as it requires a single estimation, the layer-mean representation appears to be the most promising.

H-score and LogME outperform the baselines. The results demonstrate the utility of the tested estimation techniques. H-score and LogME outperform the simpler linear probing and kNN approaches. The absolute difference between kNN and H-score, when utilizing their respective most effective aggregators, is 0.20 (0.18 for LogME), with Pearson correlation coefficients (ρ) of 0.68 for kNN versus 0.88 for H-score (and 0.86 for LogME).

The difference compared to linear probing is not as drastic, but still significant. H-score outperforms linear probing by 0.04. Furthermore, linear probing requires a careful selection of the training hyperparameters and is significantly slower as it usually requires multiple epochs of training.

H-score vs. LogME. While an in-depth comparison of the various estimation strategies is certainly outside the scope of this paper, we find that

Approach	Airline	CoLA	SST-2	CoNLL	SciNER	CrossNER	Avg
Linear Probing$_{\text{last layer}}$	0.6695	0.7924	0.7447	0.8292	0.3230	0.2393	0.5997
Linear Probing$_{\text{layer mean}}$	0.8806	0.8910	0.9100	**0.9408**	0.6976	0.6998	0.8366
kNN$_{\text{last layer}}$	0.1286	0.7224	-0.3888	0.3812	-0.1428	0.1265	0.1379
kNN$_{\text{layer mean}}$	0.7256	0.7549	0.3983	0.9015	0.5516	0.6596	0.6653
kNN$_{\text{best layer}}$	0.8739	0.7379	0.4235	0.8920	0.5087	0.6474	0.6806
kNN$_{\text{avg. of scores}}$	0.6938	0.8310	0.1449	0.9057	0.4839	0.7215	0.6301
H-score$_{\text{last layer}}$	0.7801	0.8525	0.5889	0.7838	0.3786	0.5045	0.6481
H-score$_{\text{layer mean}}$	0.8857	**0.9304**	0.9118	0.9298	**0.7444**	0.8761	**0.8797**
H-score$_{\text{best layer}}$	0.8860	0.9272	0.8814	0.9385	0.7039	**0.8839**	0.8701
H-score$_{\text{avg. of scores}}$	0.8326	0.8693	0.8277	0.9114	0.6353	0.8350	0.8185
LogME$_{\text{last layer}}$	0.7043	0.7804	0.5846	0.6601	0.3217	0.4331	0.5807
LogME$_{\text{layer mean}}$	0.8866	0.9052	0.9148	0.9247	0.7036	0.8466	0.8636
LogME$_{\text{best layer}}$	**0.8924**	0.8759	0.8897	0.9234	0.6365	0.8587	0.8461
LogME$_{\text{avg. of scores}}$	0.8091	0.8916	0.8295	0.9046	0.5518	0.7946	0.7969

Table 3: Pearson's ρ correlation coefficient between transferability scores and fine-tuning performances. Comparison of using different layer information together with the different estimation techniques on 3 text classification and 3 NER datasets. We report the Pearson correlation rank between the order of fine-tuned models, and the ranking estimated by using representations of frozen models. We mark the variant with the highest correlation overall in **bold** and underline the best layer-wise aggregation (for each estimator).

both H-score and LogME are very competitive with H-score showing marginally better correlation scores. Both approaches profit from the use of layer-wise representation aggregation and both methods work quite well in combination with the layer mean which we propose using.

Runtime. As shown in Table 2, estimating the performance is roughly one order of magnitude faster than computing the true performance of a model. Since the runtime of fine-tuning depends on the number of epochs, the speedup can vary drastically.

To estimate the performance, the majority of the time is spent embedding the datapoints. The actual estimation techniques only take a fraction of the time. Thus, the differences are between the different estimation techniques are minor.

Since more than 98% of the estimation's runtime is spent on embedding the datapoints, reducing the number of datapoints would linearly reduce the estimation time as well. Further research may clarify the number of task samples required to confidently estimate the performance after fine-tuning. This could offer further runtime reductions. The number of samples may depend on the estimation technique used. Additionally, one may want to compare the estimates to fine-tuning models on a subset of the data. Though the training time may be significantly reduced, we expect that using small subsets of the training data will not provide similarly robust performance estimates.

5.1 Analysis of Layer-wise Transferability

While it is commonly reported (Rogers et al., 2020; Lin et al., 2019) that the last layer of a pre-trained transformer may not yield the best performance (as it is very much tuned to the pre-training objective and hence may lack the generality that we would like from a representation), it is not clear which layer specifically contains the most relevant representation for a given model (Fayyaz et al., 2021).

Transferability estimation offers a computationally relatively cheap option to compare the layer-wise representations with an important benefit: It does not rely on a non-deterministic optimization process which relies on finding a good set of hyper-parameters.

Figure 4 in the appendix compares the estimation approaches with linear probing each layer's representation. In the case of the estimation techniques,

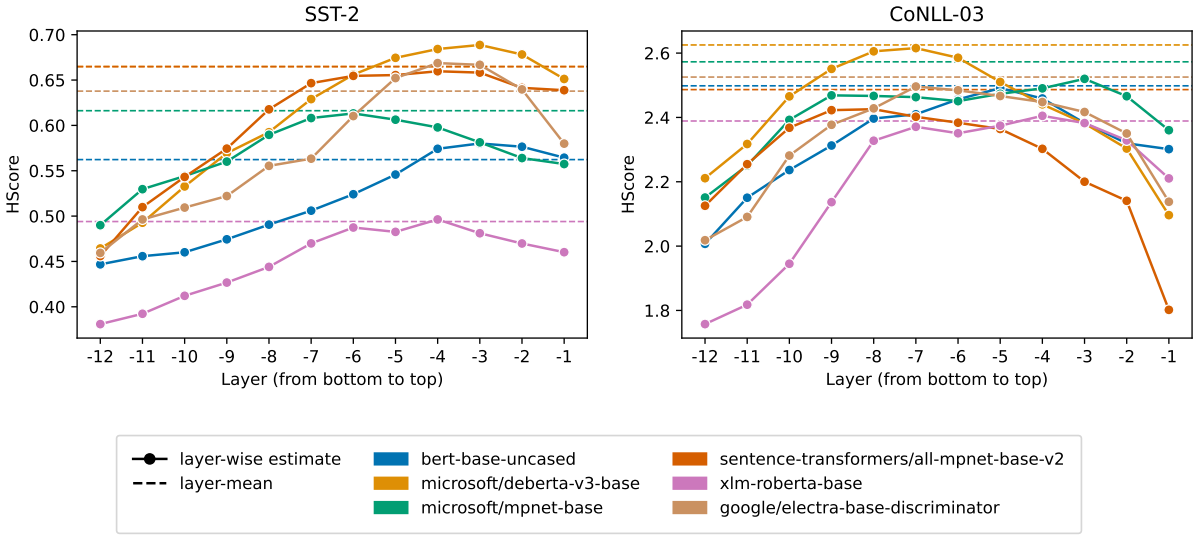


Figure 3: H-score for each of the layers in a model (on the sentence-level task SST-2 compared to the span-level task CoNLL-03): The peaks vary between the tasks but also differ from model to model. Note that the highest layer mean in SST-2 is not drawn in bold, but represents two models (`deberata-v3-base` and `all-mpnet-base-v2`) which are very close.

we plot the resulting score. For linear probing, the plot shows the test accuracy after linear probing.

The transferability significantly varies between the different layers and the peaks of expected performance vary between the models and tasks (here shown for SST-2 and CoNLL-03 in Figure 3). We hypothesize that during fine-tuning relevant features in a model can still be utilized efficiently since they are passed to the top through the residual connections within the transformer blocks (and are only scaled down on the way). The impact of layers that do not positively contribute to the downstream tasks can easily be scaled down. When using a frozen representation this is not the case.

Since the last layer rarely contains the most relevant representation, aggregating information from all layers is important for estimating the true performance once the model is fine-tuned. As shown in this work, taking the maximum of all layers' estimated score (which amounts to selecting the value of the highest peak) and using the score of the mean representation (indicated with a dashed line in Figure 3) both are sensible choices and significantly improve the estimation of the fine-tuned performance.

6 Conclusion

We propose the aggregation of layer-wise representations to enhance the performance of an existing encoder transferability estimation techniques H-score and LogME. We demonstrate that any representation aggregation method (across all layers of the transformer model) can improve the ranking of pre-trained language models. The straightforward approach to average the representations across all layers (layer mean) improves the correlation coefficient between the true model ranks and the estimate by 0.13 for H-score and 0.28 for LogME.

Searching for the layer with the best estimate is shown to perform slightly worse on average across all tasks. Furthermore, it comes with the additional cost induced by the need to compute the score for each layer individually. We therefore recommend using the layer mean variant of H-score or LogME.

Limitations

While a system designed to automatically select a suitable pre-trained model from a model hub would need to consider all included models (or at least a large subset), H-score and LogME still require a constraint candidate list. They operate on the representations produced by the frozen models and hence require encoding task samples using each of the models. If this cannot be done in the hub directly, a copy of the weights for each of the models still needs to be retrieved. Depending on the number of models, this could induce a lot of traffic.

Hence, H-score and LogME cannot be extended to complete model hubs. Instead, as suggested in the paper, they would need to be complemented

with a component that can produce candidates based on the models' metadata to automatically select models. However, it still can play a crucial role in reducing the number of candidates before fine-tuning any of the models.

Furthermore, transferability estimates are constrained to supervised tasks. E.g. H-score and LogME cannot be used to select suitable LLMs for text generation.

Ethics Statement

Anticipating ethical concerns for this research is challenging because of the wide-ranging utility of language model selection. As fine-tuning the entire space of available language models is unsustainable and unethical in terms of climate sustainability, efficient encoder pre-selection methods such as H-score and LogME provide a positive step toward tackling this problem. Our work enhances the process of efficient model selection by increasing its accuracy, thereby encouraging practitioners to avoid unnecessary fine-tuning.

Acknowledgements

We thank all reviewers for their valuable comments. Max Ploner and Alan Akbik are supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2002/1 "Science of Intelligence" – project number 390523135. Alan Akbik is further supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the Emmy Noether grant "Eidetic Representations of Natural Language" (project number 448414230).

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. 2019. An Information-Theoretic Approach to Transferability in Task Transfer Learning. In *2019 IEEE International Conference on Image Processing (ICIP)*.

Francesco Barbieri, José Camacho-Collados, Leonardo Neves, and Luis Espinosa Anke. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. *CoRR*, abs/2010.12421.

Elisa Bassignana, Max Müller-Eberstein, Mike Zhang, and Barbara Plank. 2022. Evidence > intuition: Transferability estimation for encoder selection.

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Rushil Desai, Aditya Shah, Shourya Kothari, Aishwarya Surve, and Narendra Shekokar. 2022. TextBrew: Automated Model Selection and Hyperparameter Optimization for Text Classification. *International Journal of Advanced Computer Science and Applications*, 13(9).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Iddo Drori, Lu Liu, Yi Nian, Sharath C. Koorathota, Jie S. Li, Antonio Khalil Moretti, Juliana Freire, and Madeleine Udell. 2019. AutoML using Metadata Language Embeddings.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. *CoRR*, abs/1909.00512.

Mohsen Fayyaz, Ehsan Aghazadeh, Ali Modarressi, Hosein Mohebbi, and Mohammad Taher Pilehvar. 2021. Not all models localize linguistic knowledge in the same place: A layer-wise probing on bertoids' representations. *CoRR*, abs/2109.05958.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543.

Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021b. AutoML: A Survey of the State-of-the-Art. *Knowledge-Based Systems*, 212:106622.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *CoRR*, abs/1904.05342.

Shibal Ibrahim, Natalia Ponomareva, and Rahul Mazumder. 2023. Newer is Not Always Better: Rethinking Transferability Metrics, Their Peculiarities, Stability and Performance. In *Machine Learning and Knowledge Discovery in Databases*, pages 693–709. Springer International Publishing.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *CoRR*, abs/1907.10529.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside bert's linguistic knowledge. *CoRR*, abs/1906.01698.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. *CoRR*, abs/1903.08855.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2020. Crossner: Evaluating cross-domain named entity recognition. *CoRR*, abs/2012.04373.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *CoRR*, abs/1808.09602.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 220–229.

Cuong V. Nguyen, Tal Hassner, Cédric Archambeau, and Matthias W. Seeger. 2020. LEEP: A new measure to evaluate transferability of learned representations. *CoRR*, abs/2002.12462.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2012. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how BERT works. *CoRR*, abs/2002.12327.

Shehan Saleem and Sapna Kumarapathirage. 2023. AutoNLP: A Framework for Automated Model Selection in Natural Language Processing. In *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–4.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0306050.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding.

Anh Tuan Tran, Cuong V. Nguyen, and Tal Hassner. 2019. Transferability and hardness of supervised classification tasks. *CoRR*, abs/1908.08142.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

Alex Warstadt and Samuel R. Bowman. 2019. Grammatical analysis of pretrained sentence encoders with acceptability judgments. *CoRR*, abs/1901.03438.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Kaichao You, Yong Liu, Mingsheng Long, and Jianmin Wang. 2021. Logme: Practical assessment of pre-trained models for transfer learning. *CoRR*, abs/2102.11005.

Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. 2023a. MLCopilot: Unleashing the Power of Large Language Models in Solving Machine Learning Tasks.

Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. 2023b. AutoML-GPT: Automatic Machine Learning with GPT.

A Hyperparameters for Fine-Tuning and Linear Probing

Fine-Tuning. To fine-tune each model we use the *flair* framework (Akbik et al., 2019) and an AdamW optimizer (Loshchilov and Hutter, 2019) with a linear learning rate scheduler. For each of the dataset we optimized the hyperparameters starting with previously reported settings and tried to match the performance where it has been reported. Generally we searched the learning rate in the range {2e-4, 5e-5, 7e-5, 2e-6, 5e-6} and the batch size in {4, 8, 16, 32}. Table 4 reports the hyperparameters we eventually settled on for each dataset.

In each training run, we select the model checkpoint that performs best on the development set (i.e. early stopping) and evaluate it on the test set. The scores for the SST-2 and CoLA datasets are reported on the development set, as there are no publicly available test sets. Detailed tables in this appendix (7-12) report the average score out of three runs and a standard error.

Performing a hyperparameter search for each model individually could further improve this setup. However, this was infeasible within our computational budget given the large number of models. Where individual hyperparameter searches were performed, we observed that the ideal set of parameters depends much more on the dataset than the model (given that the models we selected all have a relatively equal size). Hence, we are confident that the fine-tuned scores and, consequently, the models' ranking are fairly close to what a more extensive model-specific hyperparameter search would yield.

Dataset	Batch Size	LR	Max. epochs
Airlines	16	5e-5	10
SST-2	32	2e-5	10
CoLA	16	2e-5	10
CoNLL-03	4	5e-6	15
CrossNER	4	5e-5	30
SciNER	4	5e-5	20

Table 4: Hyperparameters used to fine-tune models on each of the datasets.

Linear Probing. For linear probing, we use the AdamW optimizer with a batch size of 16 and train for 20 epochs using a constant learning rate of 1e-3.

B Tested Language models

Table 5 contains a list of the tested models with some information on their architecture and pre-training setup. We extended the list of candidate models used by Bassignana et al. (2022), which included 7 different models. We opted to additionally include models that were pre-trained using other objectives than just Masked Language Modeling such as `DeBERTa` (He et al., 2021a), `SpanBERT` (Joshi et al., 2019), and two sentence-transformer models: All-MPNet-base and `All-MiniLM` (Reimers and Gurevych, 2019).

C Additional Results

Table 6 reports the layer with the highest estimate (LogME) for each of the models on each of the datasets. This corresponds to the position of the peak for each line in Figure 5. In Tables 7-12, we report the fine-tuned performances for each of the models as well as the LogME scores for each of the layer-wise variants. Note that the scores for the SST-2 and CoLA datasets are reported on the development set, as there are no publicly available test sets. SectionA gives a detailed description of the fine-tuning setup used to train the models.

Language Model	Publication	Pretraining Objective	# Parameters	# Layers
allenai/scibert_scivocab_cased	Beltagy et al. (2019)	MLM, NSP	110M	12
bert-base-uncased	Devlin et al. (2018)	MLM, NSP	110M	12
bert-large-uncased	Devlin et al. (2018)	MLM, NSP	340M	24
cardiffnlp/twitter-roberta-base	Barbieri et al. (2020)	MLM	110M	12
distilbert-base-uncased	Sanh et al. (2019)	Distillation, MLM	110M	6
dmis-lab/biobert-base-cased-v1.2	Lee et al. (2019)	MLM	110M	12
emilyalsentzer/Bio_ClinicalBERT	Huang et al. (2019)	MLM	110M	12
google/electra-base-discriminator	Clark et al. (2020)	RTD	110M	12
google/electra-large-discriminator	Clark et al. (2020)	RTD	340M	24
google/electra-small-discriminator	Clark et al. (2020)	RTD	14M	12
microsoft/deberta-v3-base	He et al. (2021a)	RTD	110M	12
microsoft/mdeberta-v3-base	He et al. (2021a)	RTD	110M	12
microsoft/mpnet-base	Song et al. (2020)	MLM+PLM	110M	12
roberta-base	Liu et al. (2019b)	MLM	110M	12
sentence-transformers/all-MiniLM-L12-v2	Reimers and Gurevych (2019)	MLM+PLM, CF	14M	12
sentence-transformers/all-mpnet-base-v2	Reimers and Gurevych (2019)	MLM+PLM, CF	110M	12
SpanBERT/spanbert-base-cased	Joshi et al. (2019)	SBO	110M	12
typeform/distilroberta-base-v2	Sanh et al. (2019)	Distillation, MLM	110M	6
xlm-roberta-base	Conneau et al. (2019)	MLM	110M	12
xlm-roberta-large	Conneau et al. (2019)	MLM	550M	24

Table 5: Overview of Language Models used in experiments. Pretraining tasks include MLM: Masked Language Modeling, NSP: Next Sentence Prediction, RTD: Replaced Token Detection, PLM: Permuted Language Modeling, SBO: Sentence Boundary Objective, CF: Contrastive Fine-tuning

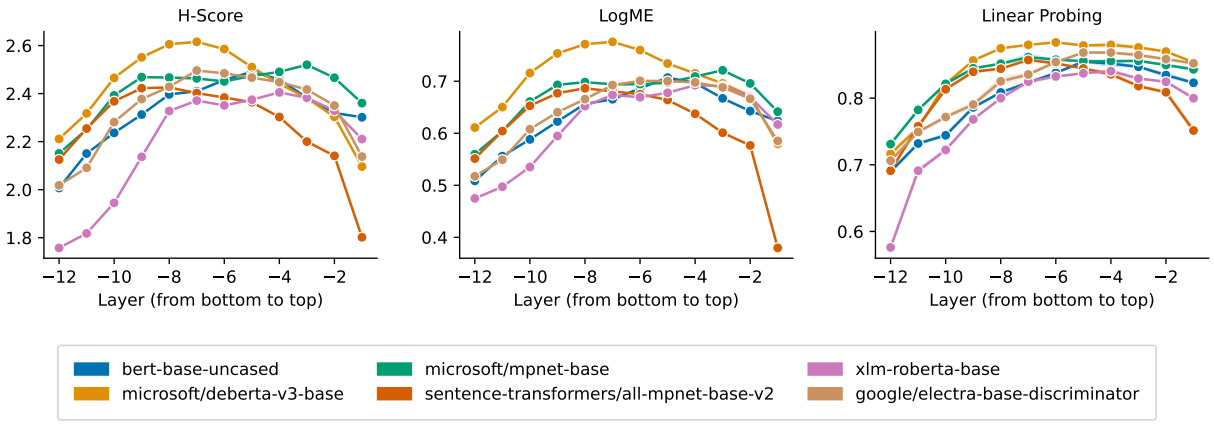


Figure 4: H-score, LogME, and F1 test scores of linear probing each layers representation on CoNLL. Since linear probing is also a means of estimating the relevance of each layer (as opposed to measuring it), we resort to comparing the different estimation techniques.

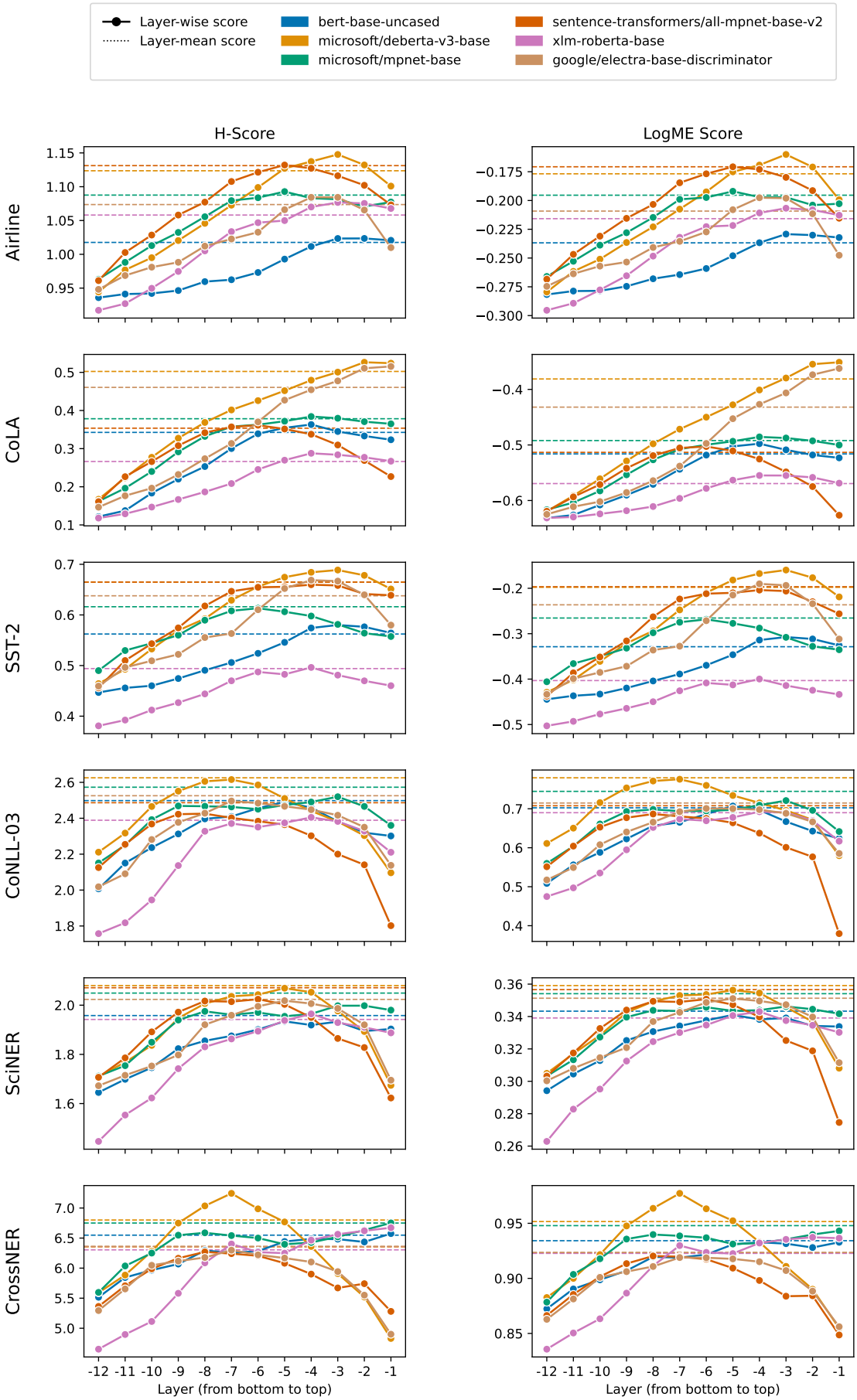


Figure 5: This plot illustrates the regularized H-score and LogME scores across each transformer layer, with a dashed line representing the layer mean performance.

Model Name	#	Airline			CoLA			SST-2			CoNLL			SciNER			CrossNER		
		LogME	HScore	kNN	LogME	HScore	kNN	LogME	HScore	kNN	LogME	HScore	kNN	LogME	HScore	kNN	LogME	HScore	kNN
allenai/scibert_scivocab_cased	12	-9	-10	-9	-3	-3	-5	-4	-4	-10	-2	-2	-2	-4	-4	-4	-1	-1	-4
bert-base-uncased	12	-3	-2	-3	-4	-4	-4	-3	-3	-10	-5	-5	-3	-5	-5	-2	-1	-1	-1
bert-large-uncased	24	-5	-4	-4	-8	-8	-8	-5	-5	-20	-9	-9	-8	-9	-8	-9	-8	-16	-6
cardiffnlp/twitter-roberta-base	12	-4	-4	-4	-3	-3	-2	-6	-6	-7	-4	-6	-2	-3	-3	-3	-8	-8	-2
distilbert-base-uncased	6	-2	-2	-2	-3	-3	-3	-2	-2	-2	-3	-3	-2	-2	-2	-1	-1	-1	-1
dmis-lab/biobert-base-cased-v1.2	12	-4	-9	-3	-4	-4	-5	-3	-3	-10	-4	-2	-2	-5	-5	-2	-2	-2	-2
emilyalsentzer/Bio_ClinicalBERT	12	-10	-10	-11	-4	-4	-5	-12	-12	-12	-5	-4	-5	-5	-5	-3	-2	-2	-2
google/electra-base-discriminator	12	-4	-3	-3	-1	-1	-1	-4	-4	-8	-6	-7	-4	-5	-5	-4	-7	-7	-4
google/electra-large-discriminator	24	-8	-8	-8	-1	-1	-1	-8	-8	-19	-15	-15	-6	-13	-13	-8	-8	-12	-9
google/electra-small-discriminator	12	-5	-5	-5	-1	-1	-1	-5	-5	-12	-8	-8	-1	-6	-6	-5	-6	-10	-1
microsoft/deberta-v3-base	12	-3	-3	-3	-1	-2	-1	-3	-3	-3	-7	-7	-5	-5	-5	-4	-7	-7	-5
microsoft/mdeberta-v3-base	12	-3	-3	-4	-1	-1	-1	-3	-3	-5	-5	-5	-5	-4	-4	-4	-5	-5	-4
microsoft/mpnet-base	12	-5	-5	-5	-4	-4	-1	-6	-6	-12	-3	-3	-6	-3	-2	-6	-1	-1	-6
roberta-base	12	-5	-5	-5	-3	-3	-1	-6	-6	-5	-4	-4	-4	-3	-8	-3	-8	-8	-3
sentence-transformers/all-MiniLM-L12-v2	12	-5	-5	-6	-7	-7	-8	-6	-6	-7	-6	-6	-6	-6	-6	-6	-6	-6	-6
sentence-transformers/all-mpnet-base-v2	12	-5	-5	-4	-6	-6	-6	-4	-4	-3	-8	-8	-7	-6	-6	-7	-8	-8	-7
SpanBERT/spanbert-base-cased	12	-5	-5	-6	-5	-5	-3	-6	-6	-11	-7	-4	-2	-7	-7	-7	-8	-8	-1
typeform/distilroberta-base-v2	6	-1	-1	-2	-2	-2	-1	-3	-3	-1	-2	-3	-2	-2	-2	-2	-1	-1	-1
xlm-roberta-base	12	-3	-3	-3	-4	-4	-3	-4	-4	-11	-4	-4	-3	-4	-4	-4	-2	-1	-4
xlm-roberta-large	24	-7	-7	-7	-6	-6	-3	-8	-8	-19	-10	-10	-7	-7	-7	-7	-10	-10	-8
Most transferable layer on average:		-4.8	-4.95	-4.85	-3.55	-3.6	-3.15	-5.05	-5.05	-9.35	-5.85	-5.75	-4.1	-5.2	-5.35	-4.55	-5	-5.75	-3.85

Table 6: Indices of the best-performing layer of each LM evaluated on 6 datasets. The index $i \in \{-1, \ldots, -n\}$ refers to transformer encoder layer, where -1 is the last layerand $-n$ is the bottom layer (i.e. the embedding layer).

Task	Language Model	LogMe				Linear	Linear	Fine-tuned
		last layer	layer mean	avg of scores	best layer	last layer	layer mean	
Airline Sentiment	allenai/scibert_scivocab_cased	–0.3106	–0.2916	–0.3001	–0.2958	78.72 ± 0.19	79.80 ± 0.36	83.36 ± 0.79
	bert-base-uncased	–0.2322	–0.2368	–0.2568	–0.2292	82.87 ± 0.20	82.84 ± 0.26	85.11 ± 0.44
	bert-large-uncased	–0.2205	–0.2203	–0.2519	–0.2176	84.07 ± 0.33	83.45 ± 0.16	86.32 ± 0.21
	cardiffnlp/twitter-roberta-base	–0.1855	–0.1800	–0.2057	–0.1759	84.90 ± 0.26	85.16 ± 0.18	85.70 ± 0.40
	distilbert-base-uncased	–0.2493	–0.2507	–0.2644	–0.2478	81.48 ± 0.23	81.44 ± 0.20	84.63 ± 0.22
	dmis-lab/biobert-base-cased-v1.2	–0.2838	–0.2669	–0.2819	–0.2790	80.09 ± 0.11	81.63 ± 0.08	84.40 ± 0.32
	emilyalsentzer/Bio_ClinicalBERT	–0.3052	–0.2865	–0.2967	–0.2894	78.58 ± 0.20	80.08 ± 0.13	82.70 ± 0.30
	google/electra-base-discriminator	–0.2476	–0.2092	–0.2346	–0.1976	81.98 ± 0.35	84.11 ± 0.33	84.97 ± 0.93
	google/electra-large-discriminator	–0.2749	–0.1681	–0.2129	–0.1566	79.81 ± 0.24	85.63 ± 0.23	86.22 ± 0.15
	google/electra-small-discriminator	–0.3300	–0.2924	–0.3075	–0.2928	77.32 ± 0.23	79.89 ± 0.23	83.33 ± 0.17
	microsoft/deberta-v3-base	–0.1993	–0.1768	–0.2105	–0.1600	83.92 ± 0.06	85.54 ± 0.18	86.86 ± 0.28
	microsoft/mdeberta-v3-base	–0.2896	–0.2112	–0.2665	–0.2022	79.30 ± 0.07	83.67 ± 0.07	85.27 ± 0.05
	microsoft/mpnet-base	–0.2028	–0.1954	–0.2158	–0.1920	83.60 ± 0.11	84.06 ± 0.14	85.18 ± 0.28
	roberta-base	–0.2018	–0.1973	–0.2186	–0.1939	84.46 ± 0.21	84.59 ± 0.08	85.27 ± 0.24
	sentence-transformers/all-MiniLM-L12-v2	–0.2614	–0.2179	–0.2432	–0.2150	83.21 ± 0.15	83.79 ± 0.18	84.74 ± 0.41
	sentence-transformers/all-mpnet-base-v2	–0.2154	–0.1707	–0.2047	–0.1707	83.93 ± 0.17	84.49 ± 0.16	85.77 ± 0.46
	SpanBERT/spanbert-base-cased	–0.2699	–0.2463	–0.2628	–0.2473	80.26 ± 0.13	82.27 ± 0.12	83.88 ± 0.12
	typeform/distilroberta-base-v2	–0.2201	–0.2251	–0.2364	–0.2201	83.59 ± 0.06	83.70 ± 0.17	84.34 ± 0.19
	xlm-roberta-base	–0.2128	–0.2158	–0.2409	–0.2067	82.64 ± 0.25	83.47 ± 0.10	84.79 ± 0.42
	xlm-roberta-large	–0.2125	–0.1912	–0.2301	–0.1791	83.08 ± 0.22	84.40 ± 0.23	86.27 ± 0.10
	AVG SCORE	–0.2463	–0.2225	–0.2471	–0.2184	81.89	83.2	84.96
	Pearson corr.	*0.7021*	*0.8765*	*0.7974*	*0.8820*	*0.6705*	*0.8711*	
	Spearman's rank corr.	*0.6544*	*0.8484*	*0.7461*	*0.8530*	*0.6446*	*0.7890*	
	Weighted Kendall's tau	*0.5340*	*0.6599*	*0.5566*	*0.6890*	*0.4852*	*0.6189*	

Table 7: Comparison between different layer aggregation methods on the Airline Sentiment dataset. Performance of linear probing and fine-tuning is measured using micro-F1.

Task	Language Model	LogMe				Linear	Linear	Fine-tuned
		last layer	layer mean	avg of scores	best layer	last layer	layer mean	
CoLA	allenai/scibert_scivocab_cased	–0.5995	–0.5978	–0.6107	–0.5925	17.22 ± 1.73	20.74 ± 1.43	38.88 ± 0.90
	bert-base-uncased	–0.5232	–0.5161	–0.5533	–0.4977	42.02 ± 0.49	40.83 ± 1.15	58.07 ± 0.25
	bert-large-uncased	–0.4956	–0.4695	–0.5400	–0.4492	46.59 ± 0.65	46.71 ± 0.34	62.01 ± 0.31
	cardiffnlp/twitter-roberta-base	–0.5056	–0.4939	–0.5319	–0.4916	38.03 ± 0.38	44.33 ± 0.63	60.37 ± 1.11
	distilbert-base-uncased	–0.5556	–0.5561	–0.5759	–0.5455	35.31 ± 0.94	34.24 ± 0.94	53.05 ± 1.56
	dmis-lab/biobert-base-cased-v1.2	–0.5774	–0.5774	–0.5944	–0.5696	26.25 ± 0.23	25.40 ± 1.00	53.78 ± 0.56
	emilyalsentzer/Bio_ClinicalBERT	–0.6120	–0.6088	–0.6161	–0.6066	12.46 ± 0.95	14.42 ± 1.09	39.59 ± 0.76
	google/electra-base-discriminator	–0.3621	–0.4319	–0.5038	–0.3622	59.33 ± 0.89	50.32 ± 1.16	67.33 ± 0.37
	google/electra-large-discriminator	–0.3140	–0.3877	–0.4618	–0.3140	60.73 ± 0.17	54.88 ± 0.25	69.17 ± 0.08
	google/electra-small-discriminator	–0.4441	–0.5075	–0.5687	–0.4440	51.09 ± 0.35	36.59 ± 1.02	58.74 ± 0.92
	microsoft/deberta-v3-base	–0.3511	–0.3812	–0.4695	–0.3511	59.88 ± 0.22	56.67 ± 0.13	71.81 ± 0.72
	microsoft/mdeberta-v3-base	–0.4102	–0.4318	–0.5513	–0.4102	50.62 ± 0.91	43.50 ± 0.18	62.47 ± 0.98
	microsoft/mpnet-base	–0.5001	–0.4921	–0.5291	–0.4855	34.79 ± 1.16	37.23 ± 1.04	62.90 ± 0.48
	roberta-base	–0.4614	–0.4830	–0.5043	–0.4507	39.89 ± 0.49	41.40 ± 0.69	62.44 ± 0.09
	sentence-transformers/all-MiniLM-L12-v2	–0.6564	–0.5852	–0.6024	–0.5708	12.33 ± 0.76	27.07 ± 0.41	47.89 ± 1.19
	sentence-transformers/all-mpnet-base-v2	–0.6265	–0.5133	–0.5533	–0.5028	23.62 ± 1.02	41.34 ± 0.17	61.84 ± 0.36
	SpanBERT/spanbert-base-cased	–0.5462	–0.5315	–0.5615	–0.5277	27.92 ± 0.91	30.83 ± 0.82	57.17 ± 0.78
	typeform/distilroberta-base-v2	–0.5100	–0.5275	–0.5463	–0.5034	33.14 ± 1.07	33.24 ± 1.73	58.05 ± 0.75
	xlm-roberta-base	–0.5688	–0.5695	–0.5908	–0.5549	16.55 ± 1.73	20.88 ± 2.28	54.14 ± 1.65
	xlm-roberta-large	–0.5309	–0.5230	–0.5648	–0.5105	22.49 ± 0.93	31.10 ± 1.05	61.97 ± 2.85
	AVG SCORE	–0.5075	–0.5092	–0.5515	–0.487	35.51	36.59	58.08
	Pearson corr.	*0.7804*	*0.9052*	*0.8916*	*0.8759*	*0.7924*	*0.8910*	
	Spearman's rank corr.	*0.8451*	*0.9609*	*0.9139*	*0.9248*	*0.7684*	*0.8797*	
	Weighted Kendall's tau	*0.7951*	*0.8920*	*0.8251*	*0.8210*	*0.7152*	*0.8249*	

Table 8: Comparison between different layer aggregation methods on the CoLA dataset. Performance of linear probing and fine-tuning is measured using Matthews correlation coefficient (MCC).

Task	Language Model	LogMe				Linear	Linear	Fine-tuned
		last layer	layer mean	avg of scores	best layer	last layer	layer mean	
SST2	allenai/scibert_scivocab_cased	–0.5127	–0.4788	–0.5008	–0.4876	78.27 ± 0.40	79.83 ± 0.23	90.54 ± 0.06
	bert-base-uncased	–0.3258	–0.3287	–0.3751	–0.3074	86.64 ± 0.06	85.09 ± 0.11	92.58 ± 0.20
	bert-large-uncased	–0.2952	–0.2961	–0.3648	–0.2683	88.02 ± 0.05	87.22 ± 0.18	93.76 ± 0.16
	cardiffnlp/twitter-roberta-base	–0.3149	–0.2713	–0.3229	–0.2634	87.27 ± 0.46	89.22 ± 0.11	94.44 ± 0.06
	distilbert-base-uncased	–0.3311	–0.3527	–0.3820	–0.3269	84.00 ± 0.17	83.60 ± 0.12	91.43 ± 0.49
	dmis-lab/biobert-base-cased-v1.2	–0.4354	–0.4223	–0.4542	–0.4274	80.45 ± 0.17	79.46 ± 0.25	91.40 ± 0.12
	emilyalsentzer/Bio_ClinicalBERT	–0.5220	–0.4855	–0.5088	–0.5002	76.20 ± 0.51	78.04 ± 0.05	90.42 ± 0.40
	google/electra-base-discriminator	–0.3118	–0.2364	–0.3057	–0.1902	89.05 ± 0.29	90.43 ± 0.17	95.20 ± 0.02
	google/electra-large-discriminator	–0.4591	–0.1948	–0.3274	–0.2366	84.00 ± 0.06	91.68 ± 0.05	96.62 ± 0.63
	google/electra-small-discriminator	–0.5443	–0.4368	–0.4729	–0.4356	78.38 ± 0.28	80.68 ± 0.06	91.53 ± 0.33
	microsoft/deberta-v3-base	–0.2190	–0.1978	–0.2641	–0.1598	90.25 ± 0.11	89.91 ± 0.11	95.98 ± 0.35
	microsoft/mdeberta-v3-base	–0.4192	–0.3522	–0.4288	–0.2754	83.49 ± 0.23	85.32 ± 0.46	93.52 ± 0.17
	microsoft/mpnet-base	–0.3354	–0.2655	–0.3192	–0.2684	86.69 ± 0.11	88.42 ± 0.23	94.72 ± 0.45
	roberta-base	–0.2985	–0.2570	–0.3086	–0.2470	87.61 ± 0.11	89.22 ± 0.22	94.83 ± 0.11
	sentence-transformers/all-MiniLM-L12-v2	–0.4199	–0.3707	–0.4065	–0.3660	83.66 ± 0.06	84.34 ± 0.06	91.42 ± 0.21
	sentence-transformers/all-mpnet-base-v2	–0.2562	–0.1966	–0.2746	–0.2039	89.39 ± 0.29	90.25 ± 0.23	94.47 ± 0.32
	SpanBERT/spanbert-base-cased	–0.3652	–0.3321	–0.3756	–0.3272	84.17 ± 0.34	87.04 ± 0.12	92.47 ± 0.15
	typeform/distilroberta-base-v2	–0.3193	–0.3194	–0.3458	–0.3080	87.44 ± 0.17	87.56 ± 0.05	92.59 ± 0.05
	xlm-roberta-base	–0.4338	–0.4032	–0.4422	–0.3998	83.78 ± 0.05	84.12 ± 0.28	92.81 ± 0.07
	xlm-roberta-large	–0.3840	–0.3176	–0.3867	–0.2918	84.81 ± 0.18	86.93 ± 0.24	94.67 ± 0.17
	AVG SCORE	–0.3751	–0.3258	–0.3783	–0.3145	84.68	85.92	93.27
	Pearson corr.	*0.5846*	*0.9148*	*0.8295*	*0.8897*	*0.7447*	*0.9100*	
	Spearman's rank corr.	*0.5789*	*0.9218*	*0.8241*	*0.9143*	*0.7424*	*0.9056*	
	Weighted Kendall's tau	*0.3981*	*0.8523*	*0.6731*	*0.7856*	*0.5343*	*0.8268*	

Table 9: Comparison between different layer aggregation methods on the SST2 dataset. Performance of linear probing and fine-tuning is measured using micro-F1.

Task	Language Model	LogMe				Linear	Linear	Fine-tuned
		last layer	layer mean	avg of scores	best layer	last layer	layer mean	
CoNLL03	allenai/scibert_scivocab_cased	0.4115	0.4166	0.3846	0.4241	67.00 ± 0.24	68.89 ± 0.05	87.58 ± 0.35
	bert-base-uncased	0.6236	0.7024	0.6350	0.7070	82.43 ± 0.04	84.09 ± 0.18	91.39 ± 0.37
	bert-large-uncased	0.5015	0.7126	0.6102	0.7334	71.83 ± 0.06	84.53 ± 0.22	92.12 ± 0.06
	cardiffnlp/twitter-roberta-base	0.6211	0.7529	0.6782	0.7293	83.37 ± 0.10	85.56 ± 0.16	92.84 ± 0.08
	distilbert-base-uncased	0.6849	0.7147	0.6643	0.7255	84.23 ± 0.33	82.70 ± 0.03	90.69 ± 0.19
	dmis-lab/biobert-base-cased-v1.2	0.3855	0.4070	0.3780	0.4080	66.30 ± 0.08	67.90 ± 0.16	87.83 ± 0.19
	emilyalsentzer/Bio_ClinicalBERT	0.3358	0.3619	0.3409	0.3544	59.53 ± 0.30	63.36 ± 0.16	85.23 ± 0.09
	google/electra-base-discriminator	0.5853	0.7147	0.6426	0.7008	85.69 ± 0.22	86.12 ± 0.15	92.55 ± 0.09
	google/electra-large-discriminator	0.5525	0.7773	0.6927	0.7705	83.37 ± 0.21	87.52 ± 0.10	94.02 ± 0.10
	google/electra-small-discriminator	0.4542	0.5012	0.4690	0.4941	77.11 ± 0.06	75.95 ± 0.09	88.63 ± 0.36
	microsoft/deberta-v3-base	0.5796	0.7794	0.7028	0.7757	84.93 ± 0.23	87.97 ± 0.05	93.86 ± 0.21
	microsoft/mdeberta-v3-base	0.5091	0.6033	0.5332	0.6204	80.66 ± 0.10	81.80 ± 0.12	93.19 ± 0.06
	microsoft/mpnet-base	0.6415	0.7446	0.6724	0.7210	84.37 ± 0.23	86.44 ± 0.08	92.88 ± 0.61
	roberta-base	0.5960	0.7497	0.6744	0.7134	83.39 ± 0.06	86.91 ± 0.07	93.47 ± 0.09
	sentence-transformers/all-MiniLM-L12-v2	0.2386	0.5777	0.5047	0.5735	69.16 ± 0.13	78.42 ± 0.22	90.57 ± 0.17
	sentence-transformers/all-mpnet-base-v2	0.3797	0.7077	0.6155	0.6865	75.68 ± 0.19	84.85 ± 0.02	91.61 ± 0.25
	SpanBERT/spanbert-base-cased	0.3637	0.4358	0.4008	0.4355	63.82 ± 0.07	70.58 ± 0.44	88.39 ± 0.50
	typeform/distilroberta-base-v2	0.6577	0.7411	0.6881	0.7287	85.49 ± 0.16	86.15 ± 0.18	92.26 ± 0.02
	xlm-roberta-base	0.6168	0.6899	0.6203	0.6924	80.86 ± 0.10	82.71 ± 0.17	92.50 ± 0.20
	xlm-roberta-large	0.6203	0.7162	0.6425	0.7033	78.09 ± 0.26	82.58 ± 0.06	93.79 ± 0.06
	AVG SCORE	0.5179	0.6403	0.5775	0.6349	77.37	80.75	91.27
	Pearson corr.	*0.6637*	*0.9258*	*0.9062*	*0.9250*	*0.8318*	*0.9415*	
	Spearman's rank corr.	*0.5113*	*0.8620*	*0.8150*	*0.7383*	*0.6860*	*0.8075*	
	Weighted Kendall's tau	*0.0910*	*0.7530*	*0.6849*	*0.6177*	*0.3840*	*0.7267*	

Table 10: Comparison between different layer aggregation methods on the CoNLL03 dataset. Performance of linear probing and fine-tuning is measured using micro-F1.

Task	Language Model	LogMe				Linear	Linear	Fine-tuned
		last layer	layer mean	avg of scores	best layer	last layer	layer mean	
SciNER (science)	allenai/scibert_scivocab_cased	0.3621	0.3756	0.3563	0.3710	46.79 ± 0.06	47.34 ± 0.16	66.59 ± 0.26
	bert-base-uncased	0.3338	0.3433	0.3271	0.3410	37.75 ± 0.37	39.22 ± 0.11	63.21 ± 0.52
	bert-large-uncased	0.3140	0.3553	0.3289	0.3564	34.59 ± 0.21	41.81 ± 0.16	66.23 ± 0.57
	cardiffnlp/twitter-roberta-base	0.3296	0.3444	0.3271	0.3394	37.09 ± 0.84	40.37 ± 0.43	65.83 ± 0.23
	distilbert-base-uncased	0.3413	0.3416	0.3304	0.3437	41.16 ± 0.27	38.55 ± 0.09	61.68 ± 0.66
	dmis-lab/biobert-base-cased-v1.2	0.3362	0.3550	0.3371	0.3543	40.81 ± 0.30	41.90 ± 0.07	64.22 ± 0.67
	emilyalsentzer/Bio_ClinicalBERT	0.3141	0.3345	0.3158	0.3279	37.98 ± 0.48	39.84 ± 0.66	65.13 ± 0.34
	google/electra-base-discriminator	0.3115	0.3513	0.3309	0.3511	40.62 ± 0.15	43.80 ± 0.35	65.82 ± 0.26
	google/electra-large-discriminator	0.3054	0.3694	0.3461	0.3655	37.22 ± 0.21	44.32 ± 0.18	66.70 ± 0.28
	google/electra-small-discriminator	0.2729	0.2882	0.2725	0.2841	31.17 ± 0.01	30.47 ± 0.49	61.04 ± 0.97
	microsoft/deberta-v3-base	0.3081	0.3591	0.3375	0.3563	38.37 ± 0.01	44.70 ± 0.71	66.48 ± 0.86
	microsoft/mdeberta-v3-base	0.2754	0.3157	0.2781	0.3101	30.27 ± 0.94	32.68 ± 0.44	65.67 ± 0.85
	microsoft/mpnet-base	0.3417	0.3541	0.3363	0.3462	39.59 ± 0.03	43.26 ± 0.08	65.08 ± 0.25
	roberta-base	0.3343	0.3519	0.3354	0.3453	37.99 ± 0.33	42.20 ± 0.05	66.31 ± 0.39
	sentence-transformers/all-MiniLM-L12-v2	0.2263	0.3180	0.2926	0.3189	19.37 ± 0.14	33.49 ± 0.33	62.20 ± 0.71
	sentence-transformers/all-mpnet-base-v2	0.2746	0.3566	0.3293	0.3507	22.71 ± 0.18	39.95 ± 0.45	65.32 ± 0.19
	SpanBERT/spanbert-base-cased	0.2902	0.3416	0.3225	0.3402	28.68 ± 0.32	39.25 ± 0.25	63.76 ± 0.28
	typeform/distilroberta-base-v2	0.3400	0.3477	0.3364	0.3488	38.26 ± 0.02	40.12 ± 0.03	64.49 ± 0.09
	xlm-roberta-base	0.3302	0.3391	0.3191	0.3430	26.25 ± 1.11	36.65 ± 0.64	64.38 ± 1.00
	xlm-roberta-large	0.3303	0.3483	0.3255	0.3454	29.70 ± 0.53	37.81 ± 0.57	65.15 ± 0.67
	AVG SCORE	0.3136	0.3445	0.3242	0.342	34.82	39.89	64.76
	Pearson corr.	*0.3217*	*0.7036*	*0.5518*	*0.6365*	*0.3230*	*0.6976*	
	Spearman's rank corr.	*0.0556*	*0.7048*	*0.5408*	*0.6150*	*0.2120*	*0.7263*	
	Weighted Kendall's tau	*0.0599*	*0.6988*	*0.5987*	*0.6448*	*0.2031*	*0.7007*	

Table 11: Comparison between different layer aggregation methods on the SciNER (science) dataset. Performance of linear probing and fine-tuning is measured using micro-F1.

Task	Language Model	LogMe				Linear	Linear	Fine-tuned
		last layer	layer mean	avg of scores	best layer	last layer	layer mean	
CrossNER (science)	allenai/scibert_scivocab_cased	0.8697	0.8732	0.8601	0.8697	35.97 ± 0.50	34.22 ± 0.65	63.91 ± 0.72
	bert-base-uncased	0.9328	0.9342	0.9153	0.9328	40.83 ± 0.04	42.18 ± 0.23	67.15 ± 0.24
	bert-large-uncased	0.8887	0.9517	0.9180	0.9446	27.03 ± 0.27	41.73 ± 0.15	71.49 ± 0.68
	cardiffnlp/twitter-roberta-base	0.9141	0.9347	0.9147	0.9315	39.27 ± 0.41	43.76 ± 0.13	68.34 ± 1.96
	distilbert-base-uncased	0.9533	0.9395	0.9273	0.9533	41.43 ± 0.33	40.20 ± 0.21	63.92 ± 0.02
	dmis-lab/biobert-base-cased-v1.2	0.8667	0.8685	0.8538	0.8724	33.55 ± 0.02	32.35 ± 0.43	64.39 ± 1.00
	emilyalsentzer/Bio_ClinicalBERT	0.8401	0.8431	0.8297	0.8415	27.50 ± 0.08	27.84 ± 0.28	57.12 ± 1.45
	google/electra-base-discriminator	0.8561	0.9236	0.8987	0.9189	33.66 ± 0.10	41.66 ± 0.29	68.63 ± 0.82
	google/electra-large-discriminator	0.8479	0.9737	0.9412	0.9804	29.34 ± 0.18	47.09 ± 0.07	69.56 ± 0.92
	google/electra-small-discriminator	0.8034	0.8194	0.8093	0.8149	22.52 ± 0.21	29.31 ± 0.48	52.55 ± 0.80
	microsoft/deberta-v3-base	0.8548	0.9516	0.9247	0.9772	36.43 ± 0.16	47.47 ± 0.18	70.91 ± 0.55
	microsoft/mdeberta-v3-base	0.8254	0.8983	0.8687	0.9205	23.55 ± 0.38	30.64 ± 0.01	68.73 ± 1.01
	microsoft/mpnet-base	0.9431	0.9479	0.9277	0.9431	23.79 ± 0.34	37.43 ± 0.05	66.55 ± 0.52
	roberta-base	0.9269	0.9492	0.9304	0.9468	35.70 ± 0.76	44.89 ± 0.32	69.53 ± 0.39
	sentence-transformers/all-MiniLM-L12-v2	0.7767	0.8550	0.8357	0.8628	17.04 ± 0.07	26.62 ± 0.33	62.57 ± 0.25
	sentence-transformers/all-mpnet-base-v2	0.8486	0.9231	0.8957	0.9203	10.06 ± 0.64	31.14 ± 0.01	64.97 ± 0.99
	SpanBERT/spanbert-base-cased	0.8338	0.8730	0.8534	0.8686	4.79 ± 0.28	26.30 ± 0.03	64.15 ± 1.34
	typeform/distilroberta-base-v2	0.9453	0.9401	0.9295	0.9453	35.78 ± 0.21	41.32 ± 0.08	66.16 ± 0.19
	xlm-roberta-base	0.9365	0.9227	0.9054	0.9373	13.96 ± 0.59	37.80 ± 0.23	66.47 ± 1.07
	xlm-roberta-large	0.9325	0.9480	0.9231	0.9687	27.34 ± 0.19	40.60 ± 0.35	72.33 ± 0.55
	AVG SCORE	0.8798	0.9135	0.8931	0.9175	27.98	37.23	65.97
	Pearson corr.	*0.4331*	*0.8466*	*0.7946*	*0.8587*	*0.2393*	*0.6998*	
	Spearman's rank corr.	*0.2602*	*0.8256*	*0.6677*	*0.7534*	*0.1910*	*0.7474*	
	Weighted Kendall's tau	*0.0135*	*0.6525*	*0.4533*	*0.5897*	*-0.0288*	*0.5492*	

Table 12: Comparison between different layer aggregation methods on the CrossNER (science) dataset. Performance of linear probing and fine-tuning is measured using micro-F1.