A Closer Look at Parameter Contributions When Training Neural Language and Translation Models

Raúl Vázquez[♠] **Hande Celikkanat**[♠] **Vinit Ravishankar**[♢] **Mathias Creutz**[♠] **Jörg Tiedemann**[♠]
[♠]Department of Digital Humanities, University of Helsinki
[♢]Language Technology Group, Department of Informatics, University of Oslo
[♠]{firstname.lastname@}@helsinki.fi [♢]vinitr@ifi.uio.no

Abstract

We analyze the learning dynamics of neural language and translation models using Loss Change Allocation (LCA), an indicator that enables a fine-grained analysis of parameter updates when optimizing for the loss function. In other words, we can observe the contributions of different network components at training time. In this article, we systematically study masked language modeling, causal language modeling, and machine translation. We show that the choice of training objective leads to distinctive optimization procedures, even when performed on comparable Transformer architectures. We demonstrate how the various Transformer parameters are used during training, supporting that the feed-forward components of each layer are the main contributors to the optimization procedure. Finally, we find that the learning dynamics are not affected by data size and distribution but rather determined by the learning objective.

1 Introduction

Neural models and Transformers in particular have achieved a great performance in almost every natural language processing (NLP) task. However, they largely remain black-box systems despite various efforts to analyze them. Much work has been devoted to understanding the dense representations that are created during training (Pavlick, 2022; Saphra, 2021) in an attempt to see whether known linguistic properties are present and how they are expressed in the model (Vulić et al., 2020; Raganato et al., 2020; Serrano and Smith, 2019). However, the intrinsic dynamics of the training procedure itself have not been analyzed in depth for highly complex network architectures. In this paper, we take a closer look into the process of parameter optimization, aiming at

- identifying the network components that contribute to the optimization of the loss function,
- revealing the effects of learning objectives and network components on training dynamics,
- detecting the impact of training data size and distribution on learning dynamics and parameter updates.

In order to enable a systematic comparison, we set out with a standard architecture that we train from scratch for each individual task. We use the Transformer architecture (Vaswani et al., 2017) with its common parametrization and size, and train it with three popular learning objectives: masked language modeling (MLM) (Taylor, 1953; Devlin et al., 2019), causal language modeling (CLM), and neural machine translation (NMT). The latter is an interesting special case as it is designed as a conditional language model in a sequence-to-sequence architecture with encoder and decoder components.

For the analysis, we use Loss Change Allocation (LCA) (Lan et al., 2019), an indicator of the contribution of individual parameters to the decrease of the total loss. Its accumulative properties provide us with a tool to perform a fine-grained analysis of the opaque optimization process of complex Transformer-based neural NLP models. We can, therefore, examine how individual layers and sub-layer components contribute to the overall loss function. This confers a unique view of the training process and how the network parameters behave during the learning procedures. Our analysis provides insight over the training dynamics broken down over the model layers and sub-layer components. On the one hand, we observe key differences in the learning dynamics across the objective arise from the nature of the loss function; depending whether being trained for gap-filling or auto-regressive prediction. On the other hand, we see clear differences linked to the nature of the sub-layer components.

We use a consistent experimental setup to enable a systematic and fair comparison between mod-

Proceedings of the 29th International Conference on Computational Linguistics, pages 4788–4800
October 12–17, 2022.

els and learning objectives. With this, we can observe all parameters when training our models from scratch. The methodology and setup are described in detail in sections 2 and 3 before we move on to our analyses and discussion of results in section 4.

2 Methodology

2.1 Neural architecture

All the models in our experiments optimize for cross-entropy loss and are based on the Transformer architecture, which currently dominates the entire field of NLP. We use a consistent setup of overall 12 layers; for the machine translation model the layers are divided into 6 each for the encoder and the decoder. Each layer is comprised of multi-headed attention and feed-forward modules that also pass information via residual connections. The essential parameters of the multi-headed attention network include the so-called key W^K, query W^Q, value W^V and out W^O parameters.[1] We refer to those parameters as the **self-attention block**. Key, query and value are used independently H times to compute a self-attention output vector, where H is the number of heads. The heads are concatenated and down-projected to the dimension of the model d_h via the out linear transformation W^O, to be passed to a two-layered feed-forward network (FFwd) creating the dense-layer representation. If we obviate the biases, dropout and normalization steps, we describe the interaction of the trainable parameters (the W parameter matrices) in one layer of the Transformer as follows:

$$\Phi = X\ W_i^{\Phi}\ ; \quad \Phi = K, Q, V \tag{1}$$

$$Z_i = softmax(\frac{QK^T}{\sqrt{d_k}})V\ ; \quad i = 1:H$$

$$Z = [Z_1, \ldots, Z_H]\, W^O \tag{2}$$

$$FF = W_2\ RelU(W_1 Z) \tag{3}$$

All our models are set to $d_h = 512$, $d_k = 64$ and $H = 8$, as is common practice. The decoder layers used in the MT systems use an additional encoder-decoder attention module in which the keys and queries are computed using the last encoder layer state, while the values come from the module's input signal.

[1] W denote the parameter matrices, as in Alammar (2018)

2.2 Training objectives

Machine Translation In the traditional encoder-decoder approach to neural MT, the model predicts words in the target sentence, word by word, i.e., given a source sentence $X = (x_1, \ldots, x_{T_x})$, a target sentence $Y = (y_1, \ldots, y_{T_y})$ and a model parametrized by θ, at each step the model learns to provide estimates of the conditional probability distribution $P(y_i|X, y_1, \ldots y_{i-1}, \theta)$. For this, we train the standard Transformer for performing translation and then analyze its encoder and decoder.

Causal Language Modeling CLMs estimate the probability of a word given the previous words in a sentence. Formally, the model is trained with inputs $X = (x_1, ..., x_{i-1})$ and outputs $Y = (x_i)$, to estimate $P(x_i|x_1, ..., x_{i-1}, \theta)$. We follow Radford and Narasimhan (2018) and use stacked Transformer decoder layers as models.

Masked Language Modeling The MLM objective is designed for predicting randomly masked tokens from an input sentence. For our experiments, we adopt the RoBERTa (Liu et al., 2020b) implementation of the MLM objective (Devlin et al., 2019). Namely, we sample 15% of the tokens to be predicted and replace the corresponding input token by [MASK] 80% of the time, with a random token 10% of the time and the other 10% is left unchanged. This is done with dynamic masking, so we generate the masking pattern every time we feed a sequence.

For a sentence $(x_1, \ldots, x_T)$, where token x_i is replaced with [MASK], the input to the model is $X = (x_1, \ldots, x_{i-1}, [\text{MASK}], x_{i+1}, ..., x_T)$ and the output $Y = (x_i)$, by estimating $P(x_i|X, \theta)$.

2.3 Loss Change Allocation

We use Loss Change Allocation (LCA) (Lan et al., 2019) because it allows for a fine-grained analysis of each network component. It measures the contributions of each parameter to the change in the loss function at each gradient update by decomposing the components of an approximate path integral along the training trajectory using a Runge-Kutta integrator (Runge, 1895; Kutta, 1901).

To compute the change in loss due to a parameter $\varphi \in \mathbb{R}^K$ update from step t to $t+1$, the LCA uses the first order approximation for the change in loss during the $(t+1)$-th training step

$$L(\varphi_{t+1}) - L(\varphi_t) \approx \sum_{i=0}^{K-1} (\nabla\varphi L(\varphi_t))^{(i)} (\varphi_{t+1}^{(i)} - \varphi_t^{(i)}),$$

where $\eta^{(i)}$ represents the i-th entry of any vector η, and $\nabla_\varphi L(\varphi_t)$ is the gradient of the loss with respect to φ evaluated at φ_t.

This formulation makes it possible to aggregate the LCA over higher-level breakdowns because the sum of all individual components equals the total change in loss. This also holds for layer- and sub-layer accumulations as each contribution has the same fundamental units as the loss: *nats* for models that optimize for cross-entropy loss, as is our case.

The LCA measure allows us to identify loss decreases at a per-parameter, per-timestep level. Put simply, a negative LCA at a given parameter update translates into that parameter being beneficial for the optimization process because the LCA is negative when that parameter's component of the gradient is negative. Conversely, positive LCA is "hurting" the learning process, for instance caused by noisy mini-batches where the gradient points in the wrong direction or has too large a step size for an irregular loss landscape (Lan et al., 2019; Jastrzębski et al., 2019; Xing et al., 2019). In other words, the raw value of the LCA is an intuitively interpretable statistic that allows analyzing the training dynamics.

3 Experimental setup

Models. We apply the common Transformer *base* settings of Vaswani et al. (2017). For consistency, we train all the models using the same toolkit, fairseq (Ott et al., 2019). We record the LCA at every parameter update and refer the reader to Appendix A for a details on model hyper-parameters.

Data. In our main experiments we use the English - German portion of Europarl (Koehn, 2005). We download, preprocess and split it using OpusTools (Aulamo et al., 2020). We use 2.5k sentences for dev and test, leaving a total of 1.9M sentences in the training data. For all systems we learn and apply byte-pair-encoding (BPE) segmentation (Sennrich et al., 2016; Kudo and Richardson, 2018). For the MT systems we use a 32k vocabulary shared among source and target, while for the LMs we train monolingual models, also with 32k vocabulary.

We perform additional experiments to investigate how different training signals may affect the LCA, and hence the validity and limitations of our conclusions. For these, we use data in different languages (English, Estonian, German, Finnish), domains (Europarl, WMT[2]), and of different sizes (700k, 2M and 5M utterances). Details about all experiments' data and model settings can be found in Appendices A and B.

4 Analyzing the training dynamics and parameter contributions

In this section we systematically dissect the development of the training process using the LCA indicator. In section 4.1, we look at the overall picture by aggregating the contributions throughout the entire training process. The aggregated figures showcase differences that originate from the training tasks and contrast the monolingual LMs with the sequence-to-sequence MT model. In section 4.2 we zoom into the iterative training procedure to further analyze the dynamics of learning network parameters over time. In section 4.3 we analyze the individual attention heads and in section 4.4 we look at the parameters that *hurt* the optimization process.

4.1 Layer-wise contributions

In Figure 1 we depict the LCA aggregated over all parameters within each layer component and summed over all time steps. The plots reveal particular characteristics for the components of each model, evidencing that the training dynamics are highly dependent on nature of the sub-layers. In particular, the difference in the order of magnitude across the components, indicates that the optimization process relies heavily on the feed-forward components.

Across modeling objectives, the clearest difference can be seen in the shapes of the **self-attention block**; the key, query, value and out parameters in eq. (1) and (2). The MLM shows a concave trend, where the main contributions are on the last layers and, to some degree, on the first two layers, while the middle layers display minor contributions. For CLM, contributions are higher for layers 1 and 6-10, with a tendency to be more evenly distributed among all layers. It contrasts with the MLM by emphasizing the penultimate layers, not the final ones, and shows a drop in the final layer contributions. We refer to this bimodal shape as *cup-and-handle shaped* from here on. In the MT setup, it is easy

[2]Including paracrawl, rapid 2016, europarl-v7, news-commentary-v13 and commoncrawl

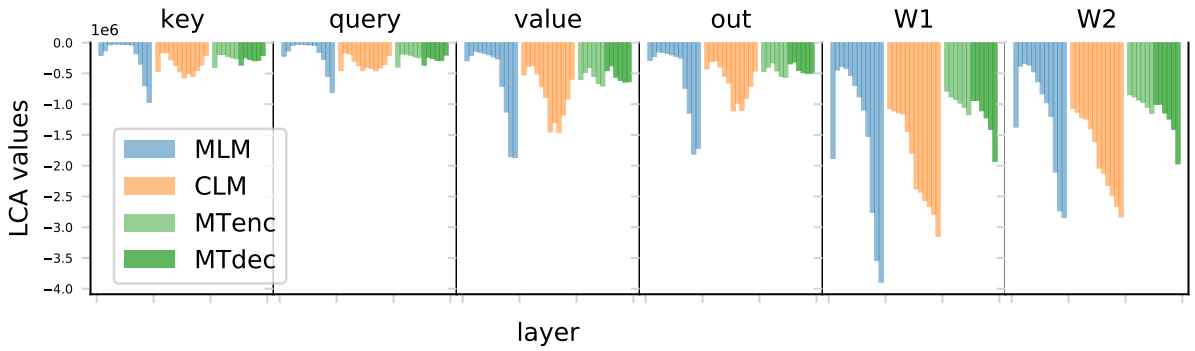

Figure 1: LCA aggregated over all training steps results for individual components of different layers. Each bar location in the x-axis denotes a separate layer.

to identify the transition from the encoder, and we see that the key-query pair behave similarly, as do the value-out pair. The encoder's key-query behaviour peaks at the first layer, and falls drastically at the second layer, showing a slight increase afterwards. Meanwhile, the value-out couple show a more clear concave form with prominent initial and final layers. The decoder's key-query pair show a cup-and-handle shape, and the value-out LCA is similar but does not drastically decrease for the last layer.

The **feed-forward block** components, W_1 and W_2 in eq. (3), show a higher order of magnitude and a similar shape for all models, with the highest accumulated LCA value in the final layer. However, MLM stands out by having strong contributions in the fist layer and a steep increase towards the final layers. Unlike the other models, the MLM shows the same trend for all the sub-layer components. The CLM shows monotonic increases over all layers, with the particularity that layers 4-7 show sharper jumps – noticeably, Layer 7 is where we observe the change in trend for the attention-block components. For the MT model, we again see the separation between encoder and decoder with the highest contributions at the end layers of each module. The encoder shows linear increase, while the decoder behaves more like the CLM model with moderate contributions initially and high increase towards the last layer.

Discussion. Under our controlled experimental setup, it is reasonable to assume that the specific behaviour observed in the learning dynamics stem from the particularities of each task. Specifically, the MLM performs the cloze task: has access to all tokens in the sentence and decodes a few, non-sequential tokens. We hypothesize that this forces MLM to compose output representations simultaneously, leading to a higher amount of information flowing into upper layers. This is consistent with previous research stating that MLMs recreate token identity in later layers, and more contextualized representations (Voita et al., 2019a; Ethayarajh, 2019). In contrast, the CLM has access to previous tokens only and generates the following one. The left-to-right decoding, may also be the reason for the shape of the attention parameters. The model sees only the *past* context, and hence depends more on the middle layers. Some support for this hypothesis comes from Ethayarajh (2019), showing that CLMs produce more context-specific representations as early as layers 4–5, presumably to predict the next word more accurately. Also, Voita et al. (2019a) demonstrate that when advancing across the layers of the [CLM] network, the system loses information about input and accumulates information about output, that is, the context in the CLM case. Moreover, middle layers potentially help learning more fine-grained syntactic information on MLMs (Tenney et al., 2019), a behavior that may become more prominent for CLMs due to its nature, although we leave the testing of this hypothesis for future work.

For the MT model, the task at hand also dictates the observed behaviour. By design, the encoder is more similar to an MLM because it has access to all tokens in the source side, while the decoder has more similarities to a CLM. We observe that the training dynamics behave accordingly – especially the self-attention parameters, with the slight differences arising perhaps from translation being a more complex task (see Voita et al. (2021) for a discussion on how NMT is a composition of several sub-tasks). MT also requires a higher capacity from the model. The encoder–decoder bottleneck triggers this even further making clear why we,

as Zhu et al. (2020), observe peaks at the transition from encoder layers to decoder in W_1 and W_2. The encoder transfers information to the next stage where the decoder starts its generative task.

Finally, we observe higher-magnitude contributions from the feed-forward block for all models may be due to the differences in the update frequencies. This was noted by Zhu et al. (2020) between sparse and dense layer, but this argument can be extended to the difference in feed-forward and attention blocks contributions, since the attention block updates are relative to the token they score from the input (see eq. 1), whereas feed-forward block updates are relative to all tokens. In accordance with Mickus et al. (2022), who show the high relative importance of the feed-forward components in the token representations, our results also suggest that when analyzing Transformer-based models and their representations the feed-forward modules should not be discarded in favor of the more readily interpretable multi-headed attention components.

4.2 The learning dynamics over time

So far, we have pointed out the time-collapsed behaviour in the self-attention and the feed-forward components of each layer. In this section, we show the particular dynamics of the LCA values over the course of training (Figure 2), analyzing the parameters in two broad groups: the feed-forward parameters W_0 and W_1 from eq. (3), and the self-attention block parameter matrices W^{Φ} for the `key`, `query` and `value` parameters from eq. (1). Since the LCA presents a high variance during training, making the learning process fuzzy, as noted by Lan et al. (2019), the trends in Figure 2 were obtained using moving averages with a window of 1k steps.

All models show that the largest LCA values are observed early during training, denoting a fast initial learning phase, after which the gradient descent learning dynamics transitions to a much slower exploration phase, in line with the findings of Feng and Tu (2021). The plots also reveal the convergence property (Raghu et al., 2017) in action, by which earlier layers tend to converge faster. However, the MLM presents a distinctive behavior in the initial and final layers. The embedding layer, as well as layers 1, 10–12 contribute substantially to the overall loss optimization especially in the beginning of the training runs. Moreover, these layers present oscillations for much longer and with larger magnitudes. In contrast, the middle layers do not show substantial activity throughout the entire training process in MLM. These parameters converge early during training, particularly those in the self-attention block. CLM and MT parameter contribution patterns display many similarities in shape across almost all layers and blocks. MT displays a smoother convergence than CLM, especially for layers 7–10. The most significant contributions in CLM clearly come from layers 7–9 in the attention block and the layers 7–12 of the feed-forward block. These higher activations in the middle layers also exhibit the most striking differences to the MT model.

Discussion. We observe that some of the parameters settle remarkably early, with very minor contributions to the loss decrease after an initial phase. This is in line with Lan et al. (2019), who, after analyzing the noisiness of the training process, suggest to freeze parameters that harm the optimization process since it sheds light on the connection between LCA and how parameters are optimized for the task at hand. In other words, some parameters are not that relevant after the initial training phase, where the loss decreases drastically. On the contrary, the MLM later layers take a significantly longer time to settle. Feng and Tu (2021) show that in SGD optimization with mislabeled data points, the parameters go through a late retrofitting phase in which they try to learn to account for the mislabeled data. In our case, we do not have a set of correct-incorrect labels, but we do the random masking for the cloze task on-the-fly, implying that the training data contains a lot of examples with easy-to-learn rules, and some rarer harder-to-generalize exceptions. We draw a comparison to such retrofitting in the observed oscillations of the MLM initial and final layers, since MLM mostly induces learning at independent lexical level (as encoded in the initial embedding layer) and the most contextualized layers at the end of the self-attention chain.

As noted in section 4.1, the essential operational difference across objectives can explain the behavioral similarities of CLM and MT in comparison to MLM. The latter extensively uses contextualized information to predict lexical gaps and, therefore, requires *strong word embeddings* and a careful connection of *contextual clues in both directions*. CLM and MT, on the other hand seem to require more iterative knowledge from all layers forcing more

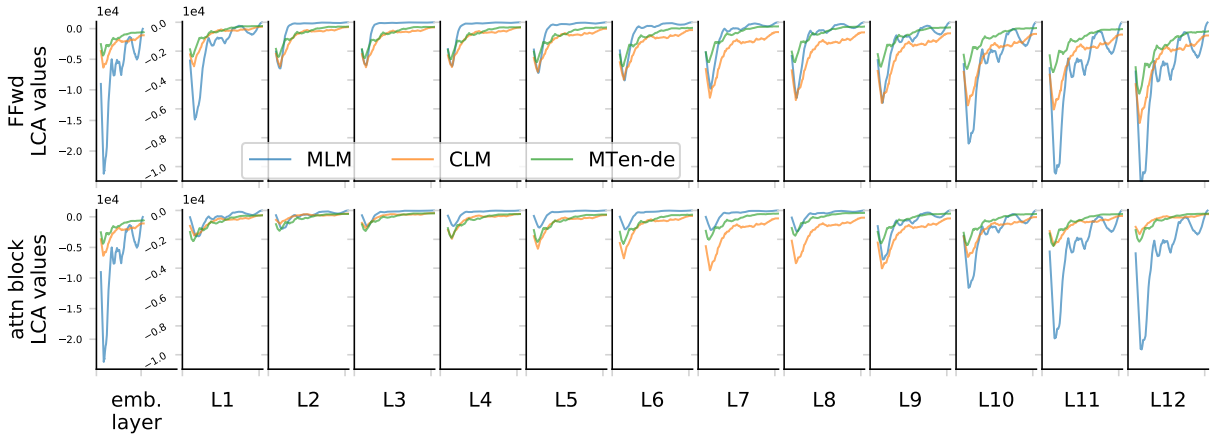

Figure 2: Training dynamics trends decomposed over time for the three learning objectives. The embeddings layer plot is the same on both rows, and it shows a different scale for visualization purposes.

adjustments throughout the entire network.

When focusing on the *difference between attention and feed-forward components*, we observe that, compared to the attention parameters, the feed-forward components of CLM- and MT-objective models seem to converge at a much later stage and have a higher contribution to the loss decrease, particularly at layers 10–12. This backs up the observations made by Wang and Tu (2020), where self-attentive components were empirically found to be less important for machine translation. For reference, the mechanism by which these feed-forward components operate is described by Geva et al. (2021). Previous studies have also concluded that attention in MT relies heavily on fixed positional connections (Raganato et al., 2020; Voita et al., 2019b). This means that their contribution to reducing the loss is limited once the essential attention patterns have been established and learned.

4.3 The training dynamics of individual attention heads.

Aggregating over all attention heads as we have done so far could hide variations present across them. Therefore, in Figure 3 we plot the contributions of each individual head over the entire training period, for each of the self-attention block parameter matrices W^{Φ} from eq. (1). First, we observe some redundancy in that many of the differences between contributions measured across heads of the same layer are negligible. Second, we also see that some layers' heads range from light green to dark violet (see e.g., MLM-1, -9 & -12, CLM-3 & -7, MT-5 & -7). These simultaneous behaviours can be tied to the idea of attention heads specialization (Voita et al., 2019b), by which some heads serve specific purposes. The observations also provide support for the observed redundancy in attention heads, which allows to effectively prune models without compromising prediction quality (Michel et al., 2019; Zhang et al., 2021). Moreover, most of the layers presenting no clear differences highlights the similarity in learning dynamics across all of the attention heads. This is consistent with findings from Clark et al. (2019), who argued that the apparent redundancy in BERT's attention heads is the result of attention dropout.

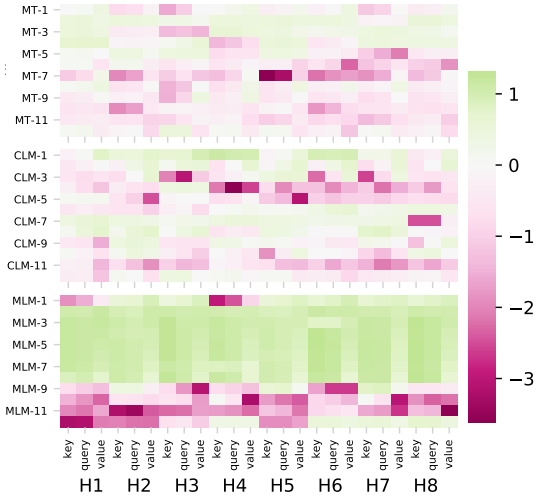

Figure 3: Contribution from individual attention heads, presented as normalized LCA values. Normalization is performed over the entire heatmap.

4.4 Noise in the optimization process

Using LCA to analyze the optimization process provides an additional insight to it, by identifying if there are parameters that *hurt* the training process,

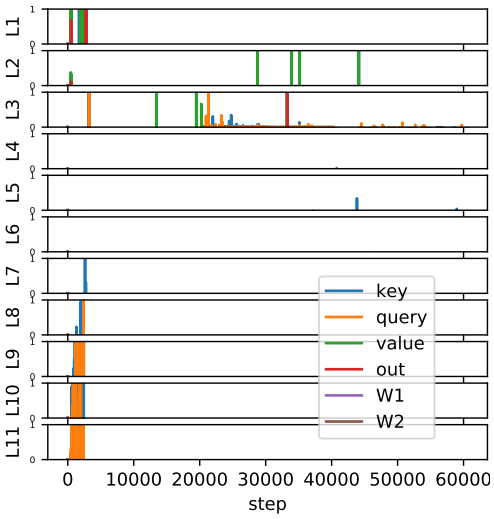

Figure 4: An overview of the MLM training process. To focus on the positive LCA we only show the interval [0,1]. We observe that positive LCA in later stages of training is present almost exclusively in attention block parameters, on layers 2–5, consistent with Fig. 1.

causing the total loss to increase instead of decrease. We check whether there are any such parameters in our models, and if so, at which phase during training they hurt the loss optimization. Lan et al. (2019) argue that the training process is notoriously noisy and they even recognize that some full layers hurt the training process.

	CLM	MLM	MT
emb.	2	0	2
key	29	1565	64
query	24	2055	64
value	24	24	130
out	24	6	113
W1	24	0	59
W2	24	0	131
Last step w/positive	1072	59727	2673
Overall max.	5.58 (key)	68.18 (value)	95 (W2)

Table 1: Summary of parameters that hurt the training process. We show the number of times a block of parameters has positive LCA, the last training step where this happens, and the magnitude of the parameter with the highest positive LCA value.

In our experiments, we rarely observe this phenomenon for the CLM and MT objectives. The *loss-hurting behavior* only seems to occur at the beginning phase of training, specifically during the first couple of thousand steps, which corresponds to at most 3% of the total time. Moreover, the magnitude of such parameters is also rather low, in the order of $1e1$. However, it should be noted that, since we are observing the component-wise aggregated contributions of parameters, we expect the effect to be "neutralized" by highly negative parameters within the corresponding components, resulting in the relatively low sums.

A different behavior is again shown by the MLM-objective (Figure 4). The MLM objective presents several positive LCA values occurring at much later stages of training. We interpret this as evidence of a comparatively noisier optimization process for the MLM objective. This is consistent with section 4.2, where we see that the MLM parameters converge much later. We depict the positive LCA values from different layers of MLM in Figure 4 to understand if such harmful contributions are specific to single layers, or are distributed over the entire network. The results show fairly clear localization to certain components of the lower layers especially for the later stages of training.

We emphasize that, contrary to findings in Lan et al. (2019), in none of our experiments do we observe a full layer or sub-layer damaging the training process. This might be due to the nature of the difference in the task, architectures, and the size of the networks used in their analysis. [3]

4.5 Effect of data size and distribution

To see the robustness of our findings against varying training datasets, we replicated the experiments presented so far using training data of different sizes (700k, 2M and 5M), domains (Europarl and WMT) and languages (English, German, Finnish and Estonian).[4] A summary of results is presented in Figure 5, with detailed plots for each setting available in the appendix B.2, Figure 6.

These additional experiments show consistent training-dynamic trends, regardless of the immediate training data properties. We observe two differences. First, the learning dynamics of the Estonian MLM system, based on smaller data, are the most different.[5] We do not observe a clear concave shape in the attention block components as for the other languages. Also, we see that the initial layers are more heavily responsible for the loss optimization. Trained with smaller data, the Estonian MLM still presents small contributions of the parameters in

[3] The authors analyze networks trained for image processing, using convolutional NNs with 5, 9 and 20 layers.

[4] For all the experiments, we use the same models and hyper-parameters (see Appendix A)

[5] We mark it with a dashed line and Figure 6 in the appendix shows the individual plots

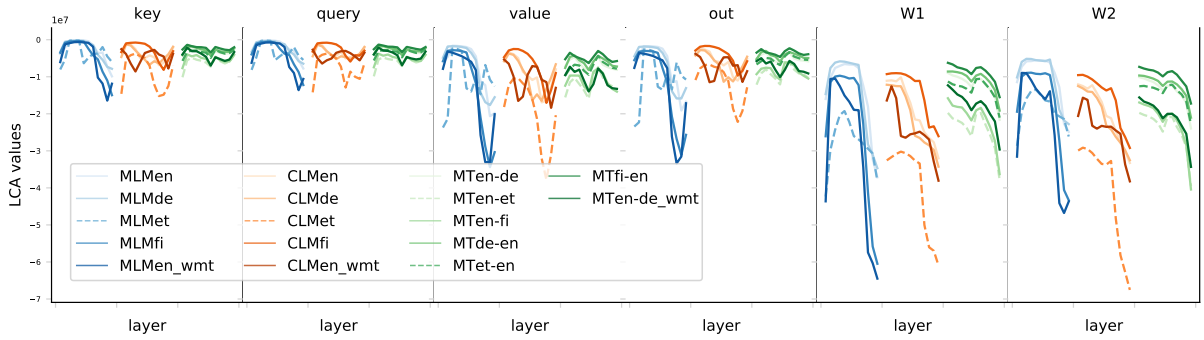

Figure 5: LCA aggregated over all training steps results for individual components of different layers. Each color corresponds to a loss function and each shade to a different model.

the middle layers, a characterizing behavior seen in all of the MLM systems we trained. Second, we observe that the CLM objective for English and German have a higher response in relatively early layers of the attention components, when compared to Finnish and Estonian. However note that the general CLM trend towards peaking contribution in penultimate layers exists for all four languages. The other cases seem very robust in their trends to the different settings, and do not present qualitatively differences in their behavior.

We interpret the observed stability of the results not only as a way to verify their validity, but also as an indication that the parameters' contributions to the optimization process are determined mainly by the choice of architecture and objective function and not so much by the training signal in the data. In particular, we argue that the training dynamics we observe are independent of the particular syntactic structure of different languages. On top of this, variations on data domains and size of the training data, show how the parameters' contributions to the change in loss are not tied to the semantic information of the training data. On the contrary, changes in the trends consistently correlate with the objective functions, over all those different settings.

5 Related work

The role of the inner representations, and internal behaviors of neural models in NLP have been extensively studied (Rogers et al., 2021; Pavlick, 2022). Relevant directions investigate, for instance, the geometry of the learned representational spaces (Ethayarajh, 2019; Vázquez et al., 2021), probing the learned representations with simple classifiers (Vulić et al., 2020; Apidianaki and Garí Soler, 2021), and the attention patterns at each step in the sequence (Voita and Titov, 2020; Raganato et al., 2020). Regarding the attention mechanism, while in the past, analyses have tended to focus on the role of the self-attention mechanism as an explanatory mechanism (Jain and Wallace, 2019; Serrano and Smith, 2019), recent work has been more nuanced about whether this is even necessary (Bastings and Filippova, 2020). Wiegreffe and Pinter (2019) argue that the attentive components are not inherently separable from the rest of the model; this has sparked analyses of attention models not directly involving attention weights. Kobayashi et al. (2020) analyse vector norms and find reasonable word alignment, while Geva et al. (2021) show that feed-forward components act as key-value stores. Ravishankar et al. (2021) analyse the effect of freezing different components on the interpretability of the attention mechanism, and Mickus et al. (2022) use a novel decomposition of Transformer embeddings to isolate the impact of the network components, showing that multi-head attention accounts for a small proportion of the embeddings.

Most such works tend to test and analyse fixed model checkpoints While there is value to analyzing these models over the course of the training process, much work in this direction emerged from outside the NLP community; e.g., LCA was first proposed and tested using only vision classification datasets (Lan et al., 2019), and it has been used since in NLP to analyze NMT training dynamics (Zhu et al., 2020). Saphra and Lopez (2019) present another early work on the analysis of the training dynamics of LMs; Kaplan et al. (2020) investigate the language modeling loss on model architecture, size, amount of training data and computing power. Their work was key for the development of large-scale pretrained models like GPT-3 (Brown et al., 2020) and Switch-Transformers (Fedus et al.,

2021). Liu et al. (2020a) analyze the training dynamics of Transformer-based models based on the stability of the gradients, leading them to propose an adaptive model initialization method.

6 Conclusions

In this work, we take a closer look at the learning dynamics of Transformer-based models using the loss change allocation (LCA) indicator. Specifically, we analyze the training dynamics for (1) masked language modeling, (2) causal language modeling, and (3) machine translation training objectives. We find that the training dynamics seem to be delimited by the training objective being either a gap-filling task or an auto-regressive prediction task, despite the similarities that the models present. In particular, the different layers of each model behave accordingly to the training objective used. We also show that the feed-forward blocks contribute more than the attention blocks to the decrease in loss during training, a finding that strengthens the relative importance of the former, and invites more research to understand this currently more obscure component.

We also show that these dynamics are robust against changes in the size and distribution of the training data. The overall trends stay the same with different domains and languages reassuring that our findings are not just an artefact of a specific setup. In future work, we would like to study the training dynamics of systems trained with different optimizers and regularization techniques, such as, attention dropout or data augmentation. We would also like to investigate the influence of noisy training data on the training process. For this, we can apply controlled shuffling to the training data to influence the noise level and gradually remove syntactic information from the training signal.

Acknowledgments

This work is part of the FoTran project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement № 771113).

References

Jay Alammar. 2018. The illustrated transformer.

Marianna Apidianaki and Aina Garí Soler. 2021. ALL dolphins are intelligent and SOME are friendly: Probing BERT for nouns' semantic properties and their prototypicality. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 79–94, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mikko Aulamo, Umut Sulubacak, Sami Virpioja, and Jörg Tiedemann. 2020. OpusTools and parallel corpus diagnostics. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3782–3789. European Language Resources Association.

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, page 1877–1901. Curran Associates, Inc.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 55–65, Hong Kong, China. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961.

Yu Feng and Yuhai Tu. 2021. Phases of learning dynamics in artificial neural networks in the absence or presence of mislabeled data. *Machine Learning: Science and Technology*, 2(4):043001.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. 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 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amost Storkey. 2019. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86, Phuket, Thailand.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

W. Kutta. 1901. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Zeit. Math. Phys.*, 46:435–53.

Janice Lan, Rosanne Liu, Hattie Zhou, and Jason Yosinski. 2019. Lca: Loss change allocation for neural network training. *Advances in Neural Information Processing Systems*, 32:3619–3629.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020a. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5747–5763, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Roberta: A robustly optimized bert pretraining approach.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in neural information processing systems*, volume 32. Curran Associates, Inc.

Timothee Mickus, Denis Paperno, and Mathieu Constant. 2022. How to dissect a muppet: The structure of transformer embedding spaces.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Ellie Pavlick. 2022. Semantic structure in deep learning. *Annual Review of Linguistics*, 8(1):447–471.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training. .

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 556–568, Online. Association for Computational Linguistics.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Vinit Ravishankar, Artur Kulmizev, Mostafa Abdou, Anders Søgaard, and Joakim Nivre. 2021. Attention can reflect syntactic structure (if you let it). In *Proceedings of the 16th Conference of the European*

Chapter of the Association for Computational Linguistics: Main Volume, pages 3031–3045, Online. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

C. Runge. 1895. Ueber die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178.

Naomi Saphra. 2021. Training dynamics of neural language models.

Naomi Saphra and Adam Lopez. 2019. Understanding Learning Dynamics Of Language Models with SVCCA. *Proceedings of the 2019 Conference of the North*, pages 3257–3267.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Wilson L. Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NeurIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Raúl Vázquez, Hande Celikkanat, Mathias Creutz, and Jörg Tiedemann. 2021. On the differences between BERT and MT encoder spaces and how to address them in translation tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 337–347, Online. Association for Computational Linguistics.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019a. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.

Elena Voita, Rico Sennrich, and Ivan Titov. 2021. Language modeling, lexical translation, reordering: The training process of NMT through the lens of classical SMT. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8478–8491, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019b. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240, Online. Association for Computational Linguistics.

Wenxuan Wang and Zhaopeng Tu. 2020. Rethinking the Value of Transformer Components. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6019–6029, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. 2019. A walk with SGD: How SGD explores regions of deep network loss?

Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Qun Liu, and Maosong Sun. 2021. Know what you don't need: Single-shot meta-pruning for attention heads. *AI Open*, 2:36–42.

Conghui Zhu, Guanlin Li, Lemao Liu, Tiejun Zhao, and Shuming Shi. 2020. Understanding learning dynamics for neural machine translation. *CoRR*, abs/2004.02199.

A Model hyper-parameters

We train all our models using the fairseq-toolkit [6], using a single Nvidia V100 GPU for 48 hours. In Table 2, we present a summmary of the set of hyper-parameters used in all cases.

Parameter	Value
num. layers	12 [7]
att. heads	8
embeddings dim.	512
ffwd hidden dim.	2048
update-freq	16
precision	fp16
total-num-update	125000
warmup-updates	8000
lr-scheduler	polynomial decay
lr	0.0005
optimizer	adam (2015)
adam-betas	(0.9, 0.98)
adam-eps	1e-06
weight-decay	0.01
clip-norm	0.0
dropout	0.1
attention dropout	0.0
activation-fn	relu
tokens-per-sample	512
max-tokens-per-sample	2048

Table 2: Set of hyper-parameters shared across all our models

B Additional experiments

B.1 Data

In Table 3 we summarize the size of the datasets used in our all experiments. We use the parallel datasets when training MT systems, while for the monolingual LM objective, we use only the target side of the corpus. Specifically for English, we use the source sentences of the En-De datasets.

In a nutshell, the experiments presented in section 4 use the En-De portion of the Europarl dataset, and we test the performance with that test split and the newstest2018. For the experiments referred in section 4.5 we train monolingual LMs in English, German, Finnish and Estonian and bilingual MT systems for En-De, En-Fi, and En-Et, as well as the inverse translation directions De-En, Fi-En and Et-En using the respective Europarl data B.2

Dataset	Languages	# Sentences		
		Train	Val.	Test
Europarl	En-De	1.92M	2.5K	2.5K
	En-Fi	1.91M	2.5K	2.5K
	En-Et	0.7M	2.5K	2.5K
WMT-2018	En-De	5M	3K	-
newstest2018	En-De	-	-	2.9K
	En-Fi	-	-	3.0K
	En-Et	-	-	2.0K

Table 3: Size of the train, validation and test splits for the datasets used in all experiments.

B.2 Results

Figure 6 shows the aggregated LCA values in additional experiments using four different languages and language pairs.

[6] https://github.com/pytorch/fairseq

[7] For the MT models we use 6 layers for each, encoder and decoder

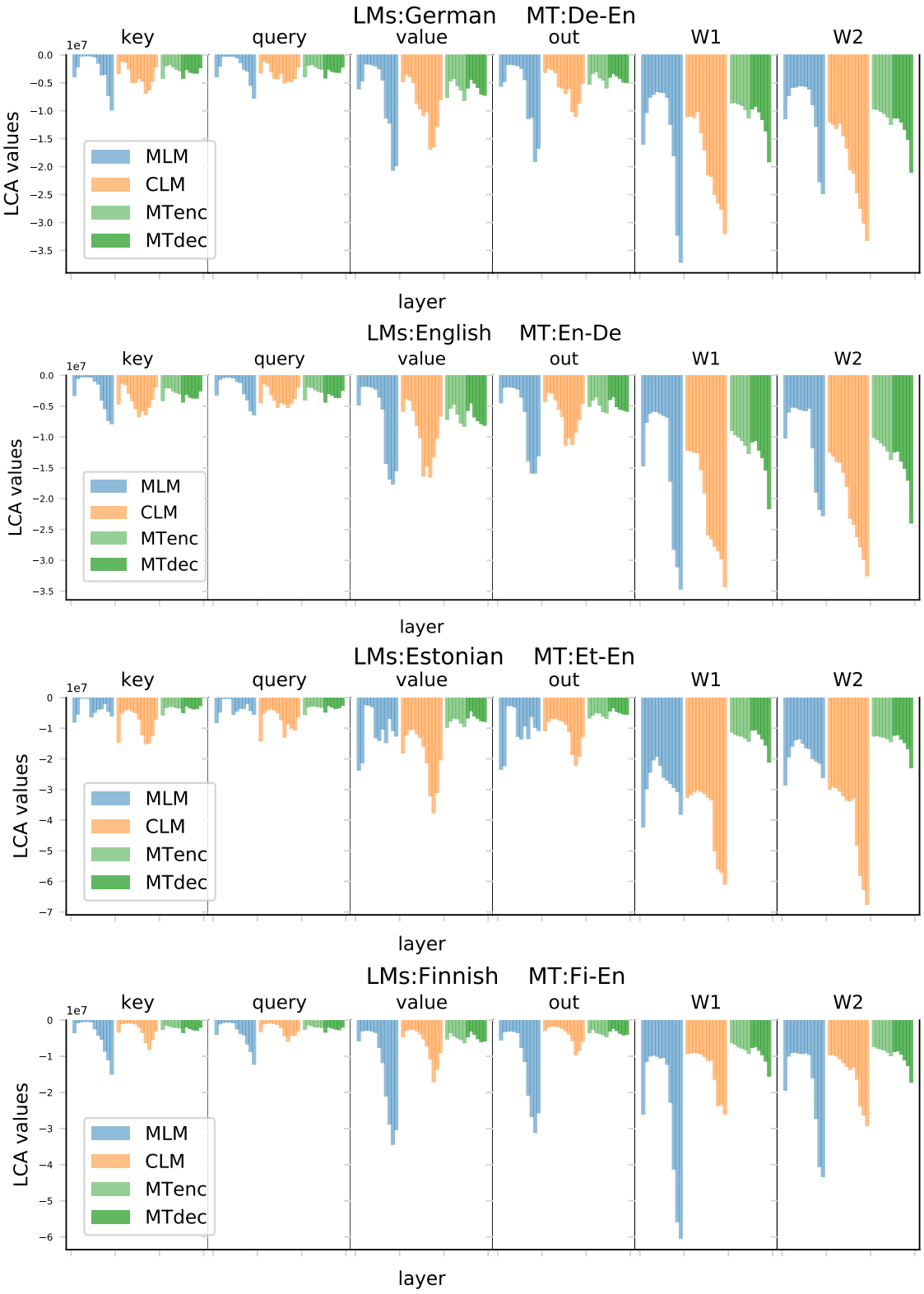

Figure 6: LCA aggregated over all training steps for the individual components of each Transformer layer. Results for all monolingual LMs and MT experiments en-de (with WMT data), de-en, et-en, fi-en.