Addressing the Representation Bottleneck in Neural Machine Translation with Lexical Shortcuts

First Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

The transformer is a state-of-the-art neural translation model that uses attention to iteratively refine lexical representations with information drawn from the surrounding context. Lexical features are fed into the first layer and propagated through a deep network of hidden layers. We argue that the need to represent and propagate lexical features in each layer limits the model's capacity for learning and representing other information relevant to the task. To alleviate this bottleneck, we introduce gated shortcut connections between the embedding layer and each subsequent layer within the encoder and decoder. This enables the model to access relevant lexical content dynamically, without expending limited resources on storing it within intermediate states. We show that the proposed modification yields consistent improvements on standard WMT translation tasks and reduces the amount of lexical information passed along the hidden layers. We furthermore evaluate different ways to integrate lexical connections into the transformer architecture and present ablation experiments exploring the effect of proposed shortcuts on model behavior.

1 Introduction

Since it was first proposed, the transformer model (Vaswani et al., 2017) has quickly established itself as a popular choice for neural machine translation, where it has been found to deliver state-of-the-art results on various translation tasks (Bojar et al., 2018). Its success can be attributed to the model's high parallelizability allowing for significantly faster training compared to recurrent neural networks (Chen et al., 2018), superior ability to perform lexical disambiguation, and capacity for capturing long-distance dependencies on par with existing alternatives (Tang et al., 2018).

Recently, several studies have investigated the nature of features encoded within individual layers of neural translation models (Belinkov et al., 2017, 2018). One central finding reported in this body of work is that, within current architectures, different layers prioritize different information types. As such, lower layers appear to predominantly perform morphological and syntactic processing, whereas semantic features reach their highest concentration towards the top of the layer stack. One necessary consequence of this distributed learning is that different types of information encoded within input representations received by the translation model have to be transported to the layers specialized in exploiting them.

Within the transformer encoder and decoder alike, information exchange proceeds in a strictly sequential manner, whereby each layer attends over the output of the immediately preceding layer, complemented by a shallow residual connection. For input features to be successfully propagated to the uppermost layers, the translation model must therefore store them in its intermediate representations until they can be processed. By retaining lexical content, the model is unable to leverage its full representational capacity for learning new information from other sources, such as the surrounding sentence context. We refer to this limitation as the representation bottleneck.

To alleviate this bottleneck, we propose extending the standard transformer architecture with lexical shortcuts which connect the embedding layer with each subsequent self-attention sub-layer in both encoder and decoder. The shortcuts are defined as gated skip connections, allowing the model to access relevant lexical information at any point, instead of propagating it upwards from the embedding layer along the hidden states.

We evaluate the resulting model's performance on multiple language pairs and varying corpus sizes, showing a consistent improvement in trans-

lation quality over the unmodified transformer baseline. Moreover, we examine the distribution of lexical information across the hidden layers of the transformer model in its standard configuration and with added shortcut connections. The presented experiments provide quantitative evidence for the presence of a representation bottleneck in the standard transformer and its reduction following the integration of lexical shortcuts. While our experimental efforts are centered around the transformer, the proposed components are compatible with other multi-layer NMT architectures.

The contributions of our work are therefore as follows:

1. We propose the use of lexical shortcuts as a simple strategy for alleviating the representation bottleneck in neural machine translation models.

2. We demonstrate significant improvements in translation quality across multiple language pairs as a result of equipping the transformer with lexical shortcut connections.

3. We report a positive impact of our modification on the model's ability to perform word sense disambiguation.

4. We conduct a series of ablation studies, showing that shortcuts are best applied to the self-attention mechanism in both encoder and decoder.

2 Proposed Method

2.1 Background: The transformer

As defined in (Vaswani et al., 2017), the transformer is comprised of two sub-networks, the encoder and the decoder. The encoder coverts the received source language sentence into a sequence of continuous representations containing translation-relevant features. The decoder, on the other hand, generates the target language sequence, whereby each translation step is conditioned on the encoder's output as well as the translation prefix produced up to that point.

Both encoder and decoder are composed of a series of identical layers. Each encoder layer contains two sub-layers: A self-attention mechanism and a position-wise fully connected feed-forward network. Within the decoder, each layer is extended with a third sub-layer responsible for attending over the encoder's output. In each case,

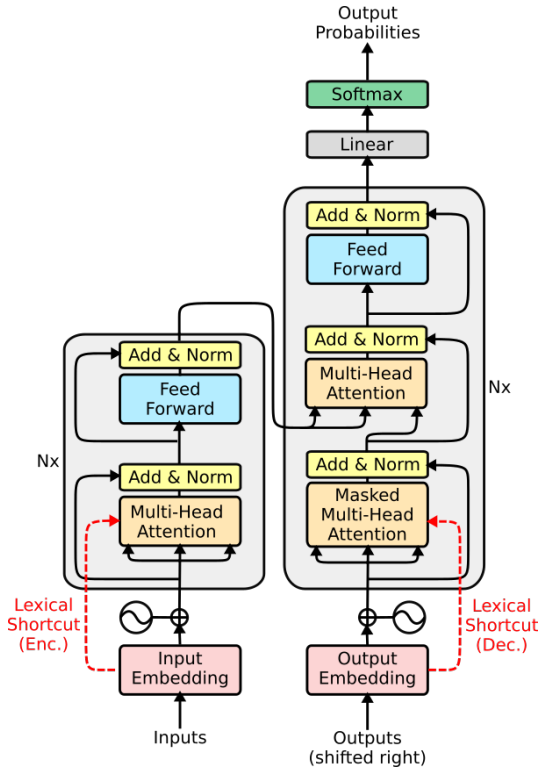

Figure 1: Integration of lexical shortcut connections into the overall transformer architecture.

the attention mechanism is implemented as multi-head, scaled dot-product attention, which allows the model to simultaneously consider different context sub-spaces. Additionally, residual connections between neighboring layers are employed to aid with signal propagation.

In order for the dot-product attention mechanism to be effective, its inputs first have to be projected into a common representation sub-space. This is accomplished by multiplying the input arrays H^S and H^T by one of the three weight matrices K, V, and Q, as shown in Eqn. 1-3, producing attention keys, values, and queries, respectively. In case of multi-head attention, each head is assigned its own set of keys, values, and queries with the associated learned projection weights.

$$Q = W^Q H^S \quad (1)$$

$$K = W^K H^T \quad (2)$$

$$V = W^V H^T \quad (3)$$

In case of encoder-to-decoder attention, H^T corresponds to the final encoder states, whereas H^S is the context vector generated by the preceding self-attention sub-layer. For self-attention, on the other hand, all three operations are given the output of the preceding layer as their input. Eqn.

4 defines attention as a function over the projected representations.

$$Attention(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (4)$$

To prevent the magnitude of the pre-softmax dot-product from becoming too large, it is divided by the square root of the total key dimensionality d_k. Finally, the translated sequence is obtained by feeding the output of the decoder through a softmax layer and sampling from the produced distribution over target language tokens.

2.2 Lexical shortcuts

Given that the attention mechanism represents the primary means of establishing parameterized connections between the different layers within the transformer, it is well suited for the re-introduction of lexical content. We achieve this by adding gated connections between the embedding layer and each subsequent self-attention sub-layer within the encoder and the decoder, as shown in Figure 1.

To ensure that lexical features are compatible with the learned hidden representations, the retrieved embeddings are projected into the appropriate latent space, by multiplying them with the layer-specific weight matrices $W_l^{K^{SC}}$ and $W_l^{V^{SC}}$. We account for the potentially variable importance of lexical features by equipping each added connection with a binary gate inspired by the Gated Recurrent Unit (Cho et al., 2014). Functionally, our lexical shortcuts are therefore reminiscent of highway connections proposed in (Srivastava et al., 2015).

$$K_l^{SC} = W_l^{K^{SC}} E \quad (5)$$
$$V_l^{SC} = W_l^{V^{SC}} E \quad (6)$$
$$K_l = W_l^K H_{l-1} \quad (7)$$
$$V_l = W_l^V H_{l-1} \quad (8)$$

$$r_l^K = \text{sigmoid}(K_l^{SC} + K_l + b_l^K) \quad (9)$$
$$r_l^V = \text{sigmoid}(V_l^{SC} + V_l + b_l^V) \quad (10)$$

$$K_l' = r_l^K \odot K_l^{SC} + (1 - r_l^K) \odot K_l \quad (11)$$
$$V_l' = r_l^V \odot V_l^{SC} + (1 - r_l^V) \odot V_l \quad (12)$$

After situating the outputs of the immediately preceding layer H_{l-1} and the embeddings E within a shared representation space (Eqn. 5-8),

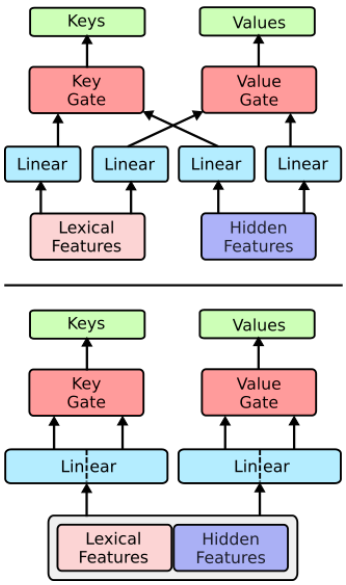

Figure 2: Modified attention inputs. Top: lexical shortcuts, bottom: lexical shortcuts + feature-fusion. Dashed lines denote splits along the feature dimension.

the relevance of lexical information for the current attention step is estimated by comparing lexical and latent features, followed by the addition of a bias term b (Eqn. 9-10). The respective attention key arrays are denoted as K_l^{SC} and K_l, while V_l^{SC} and V_l represent the corresponding value arrays. The result is then fed through a sigmoid function to obtain the lexical relevance weight r, used to combine both sets of features by calculating their weighted sum (Eqn. 11-12), where $\odot$ denotes element-wise multiplication. Next, the obtained key and value arrays K_l' and V_l' are passed to the multi-head attention function instead of the original K_l and V_l.

In an alternative formulation of the model, we concatenate E and H_{l-1} before the initial linear projection, splitting the result in two halves along the feature dimension and leaving the rest of the shortcut definition unchanged. This reduces Eqn. 5-8 to Eqn. 13-14, and enables the model to select relevant information by directly inter-relating lexical and hidden features. As such, both K_l^{SC} and K_l encode a mixture of embedding and hidden features, as do the corresponding value arrays. We refer to this step as 'feature-fusion'[1]. Figure 2 provides a high-level illustration of how lexical in-

[1]While this arguably diminishes the contribution of the gating mechanism towards feature selection, preliminary experiments have shown that replacing gated shortcuts with residual connections (He et al., 2016) causes the transformer to stagnate at 0 validation-BLEU. Similarly, using feature-fusion alone resulted in untrainable models.

formation is integrated into the attention inputs.

$$K_l^{SC}, K_l = W_l^K[E; H_{l-1}] \quad (13)$$

$$V_l^{SC}, V_l = W_l^V[E; H_{l-1}] \quad (14)$$

Other than the immediate accessibility of lexical information, one potential benefit afforded by the introduced shortcuts is the improved gradient flow during back-propagation. As noted in (Huang et al., 2017), the addition of skip connections between individual layers of a deep neural network results in an implicit 'deep supervision' effect (Lee et al., 2015), which aids the training process. In case of our modified transformer, this corresponds to the embedding layer receiving its learning signal from the model's overall optimization objective as well as from each layer it is connected to, making the model easier to train.

3 Experiments

3.1 Training

To evaluate the efficacy of the proposed approach, we re-implement the transformer model and extend it by applying lexical shortcuts to each self-attention layer in the encoder and decoder. Our code is publicly available to aid the reproduction of the reported results.[2] Details regarding our model configurations, data pre-processing, and training setup are given in the appendix (A.1-A.2).

3.2 Data

We investigate the potential benefits of lexical shortcuts on 5 WMT translation tasks: German → English (DE→EN), English → German (EN→DE), English → Russian (EN→RU), English → Czech (EN→CS), and English → Finnish (EN→FI). Our choice is motivated by the differences in training data size as well as by the typological diversity of the target languages.

To make our findings comparable to related work, we train EN↔DE models on the WMT14 news translation data which encompasses ∼4.5M sentence pairs. EN→RU models are trained on the WMT17 version of the news translation task, consisting of ∼24.8M sentence pairs. For EN→CS and EN→FI, we use the respective WMT18 parallel training corpora, with the former containing ∼50.4M and the latter ∼3.2M sentence pairs.

Throughout training, model performance is validated on newstest2013 for EN↔DE, newstest2016 for EN→RU, and on newstest2017 for EN→CS and EN→FI. Final model performance is reported on multiple tests sets from the news domain for each direction.

[2]Link withheld to preserve anonymity.

3.3 Translation performance

The results of our translation experiments are summarized in Tables 1-2. To ensure their comparability, we evaluate translation quality using sacreBLEU (Post, 2018). As such, our baseline performance diverges from that reported in (Vaswani et al., 2017). We address this by evaluating our EN→DE models using the scoring script from the tensor2tensor toolkit[3] (Vaswani et al., 2018) on the tokenized model output, and list the corresponding BLEU scores in the first column of Table 1.

Our evaluation shows that the introduction of lexical shortcuts consistently improves translation quality of the transformer model across different test-sets and language pairs, outperforming transformer-BASE by 0.5 BLEU on average. With feature-fusion, we see even stronger improvements, yielding total performance gains over transformer-BASE of up to 1.4 BLEU for EN→DE (averaging to 1.0), and 0.8 BLEU on average for the other 4 translation directions.

We furthermore observe that the relative improvements from the addition of lexical shortcuts are substantially smaller for transformer-BIG compared to transformer-BASE. One potential explanation for this drop in effectiveness is the increased size of the latent representations the wider model is able to learn. It is possible that the larger hidden state size of transformer-BIG widens the representation bottleneck, thus reducing the benefits of dynamic lexical access.

It is also worth noting that transformer-BASE, when equipped with lexical connections, performs comparably to the standard transformer-BIG, despite containing ∼2/3 of its parameters and being only marginally slower to train than our transformer-BASE implementation. An overview of model sizes and training speed is provided in the supplementary material (A.1).

Concerning the examined language pairs, the average increase in BLEU is lowest for DE→EN (0.6 BLEU) and highest for EN→RU (1.1 BLEU). While we do not have conclusive evidence for why this is the case, one possible explanation could be the difference in language topology. Of the tar-

[3]https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/get_ende_bleu.sh

Model	newstest2014 (tokenized)	newstest 2014	newstest 2015	newstest 2016	newstest 2017	newstest 2018	test mean
transformer-BASE	27.3	25.8	28.5	33.2	27.3	40.4	31.0
+ lexical shortcuts	27.6	26.1	29.5	33.3	27.5	41.1	31.5
+ feature-fusion	**28.3**	**26.8**	**29.9**	**34.0**	**27.7**	**41.6**	**32.0**
transformer-BIG	28.7	27.2	30.1	**34.0**	28.1	41.3	32.1
+ lexical shortcuts + feature-fusion	**29.4**	**27.8**	**30.3**	33.2	**28.4**	41.3	**32.2**

Table 1: BLEU scores for the EN→DE news translation task.

Model	DE→EN		EN→RU		EN→CS		EN→FI	
	newstest 2014	newstest 2017	newstest 2017	newstest 2018	newstest 2015	newstest 2018	newstest 2015	newstest 2018
transformer-BASE	31.1	32.3	27.9	24.2	23.4	21.1	18.7	14.0
+ lexical shortcuts	31.3	32.3	28.4	24.9	24.1	21.4	19.5	14.5
+ feature-fusion	**31.7**	**32.9**	**28.9**	**25.3**	**24.3**	**21.6**	**19.8**	**14.8**

Table 2: Effect of lexical shortcuts on translation performance for different language pairs.

get languages we consider, English is the morphologically weakest one, where individual words do not carry much inflectional information. As such, features encoded by (sub-)words in isolation may be less useful for the translation task than the larger sentence context aggregated within the hidden states. We expect this to result in a less pronounced representation bottleneck with fewer lexical features propagated across the network, diminishing the contribution of added shortcuts.

To further investigate the role of lexical connections within the transformer, we perform a thorough analysis of the models' internal representations and learning behaviour. The following analysis is based on the model incorporating lexical shortcuts as well as feature-fusion, due to its superior performance.

4 Analysis

4.1 Representation bottleneck

The proposed approach is motivated by the hypothesis that the transformer retains lexical features within its individual layers, which limits its capacity for learning and representing other types of relevant information. Direct connections to the embedding layer alleviate this by providing the model with access to lexical features at each processing step, reducing the need to propagate them along hidden states. To investigate whether this is indeed the case, we perform a probing study, where we estimate the amount of lexical content present within each hidden state of the encoder and decoder.

We examine the internal representations learned by our models by modifying the probing technique introduced in (Belinkov et al., 2017). Specifically, we train a separate lexical classifier for each layer of a frozen translation model. Each classifier receives hidden states extracted from the respective transformer layer[4] and is tasked with reconstructing the sub-word corresponding to the position of each hidden state. Encoder-specific classifiers learn to reconstruct sub-words in the source sentence, whereas classifiers trained on decoder states are trained to reconstruct target sub-words.

The accuracy of each classifier on a withheld test set is assumed to be indicative of the lexical content encoded by the associated transformer layer. We expect classification error to be low if the evaluated representations predominantly store information propagated upwards from the embeddings at the same position and to increase proportional to the amount of information drawn from

[4]We treat the output of the feed-forward sub-layer as that layer's hidden state.

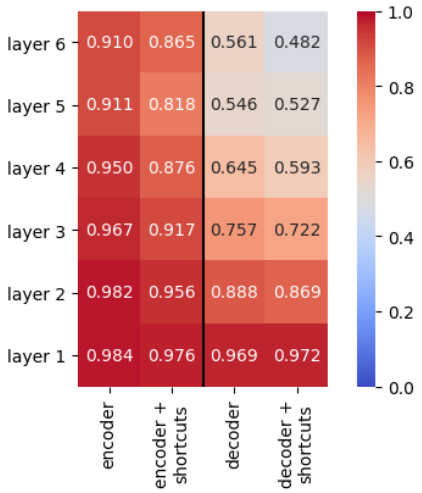

Figure 3: Layer-wise lexical probe accuracy of transformer-BASE for EN→DE (newstest2014).

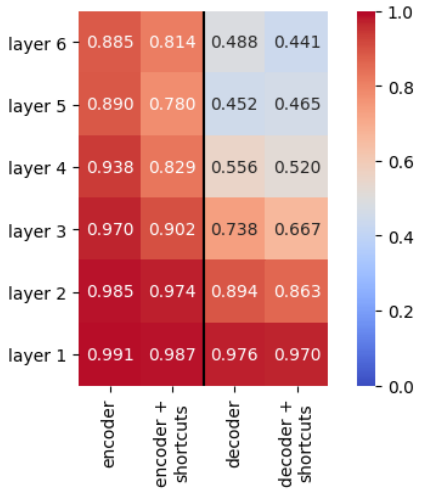

Figure 4: Layer-wise lexical probe accuracy of transformer-BASE for EN→RU (newstest2017).

the surrounding sentence context. Figures 3 and 4 show accuracy scores obtained for each layer of the base transformer and its variant equipped with lexical shortcut connections.

Based on the classifier results, it appears that immediate access to lexical information does indeed alleviate the representation bottleneck by reducing the extent to which (sub-)word-level content is retained across encoder and decoder layers. The effect is consistent across multiple language pairs, supporting its generality. Additionally, to examine whether lexical retention depends on the specific properties of the input tokens, we track classification accuracy conditioned on part-of-speech tags and sub-word frequencies. While we do not discover a pronounced effect of either category on classification accuracy, we present a summary of our findings as part of the supplementary material for future reference (A.3).

Another observation arising from this analysis is that the decoder retains fewer lexical features beyond its initial layers than the encoder. This may be due to the decoder having to represent information it receives from the encoder in addition to target-side content, necessitating a lower rate of lexical feature retention. Even so, by adding shortcut connections we can increase the dissimilarity between the embedding layer and the subsequent layers of the decoder, indicating a noticeable reduction in the retention and propagation of lexical features along the decoder's hidden states.

A similar trend can be observed when evaluating layer similarity directly, which we accomplish by calculating the cosine similarity between the embeddings and the hidden states of each transformer layer. Echoing our findings so far, the addition of lexical shortcuts reduces layer similarity relative to the baseline transformer for both encoder and decoder. The corresponding visualizations are also provided in the appendix (A.3).

Model	newstest 2017	newstest 2018	test mean
transformer-SMALL	25.2	37	28.6
+ lexical shortcuts	25.7	38	29.3
+ feature-fusion	**25.7**	**38.5**	**29.6**

Table 3: BLEU scores for small EN→DE models; 'test mean' denotes the average of all test-sets in table (1).

If the absolute amount of lexical information propagated along layers is independent of model size, shortcuts may benefit smaller models more. We put this hypothesis to test by scaling down the standard transformer, halving the size of its embeddings, hidden states, and feed-forward sub-layers. Table 3 shows that, on average, improvements to translation quality are comparable for the small and standard transformer (1.0 BLEU for both). A possible explanation is that halving the embedding size effectively halves the amount of lexical features the model can access and propagate upwards from the embedding layer, leaving the overall width of the bottleneck unchanged. Nonetheless, the exact interaction between the scale of a model and the information encoded in its hidden states remains to be fully explored. Interestingly, basic lexical shortcuts are more effective in the small model, whereas feature-fusion offers

more benefit to the standard model, implying that their relative contributions may be complementary and dependent on model size.

Overall, the presented experiments support the existence of a representation bottleneck in NMT models as one explanation for the efficacy of the proposed lexical shortcut connections.

4.2 Shortcut variants

Until now, we focused on applying shortcuts to self-attention as a natural re-entry point for lexical content. However, previous studies suggest that providing the decoder with direct access to source sentences can improve translation adequacy, by conditioning each translation step on the relevant source words (Nguyen and Chiang, 2017).

To investigate whether the proposed method can confer a similar benefit to the transformer, we apply shortcut connections to decoder-to-encoder attention, replacing or adding to shortcuts feeding into self-attention. Formally, this equates to fixing E to E^{enc} in Eqn. 5-6. As can be seen from Table 4, while integrating shortcut connections into the decoder-to-encoder attention improves upon the base transformer, the improvement is smaller than when we modify self-attention. Furthermore, combining both methods yields worse translation quality than either one does in isolation, indicating that the observed improvements are not complementary. We therefore conclude that lexical shortcuts are most beneficial to self-attention.

Model	newstest 2017	newstest 2018	test mean
transformer-BASE	27.3	40.4	31
+ self-attn. shortcuts	**27.7**	**41.6**	**32**
dec-to-enc shortcuts	27.6	40.7	31.5
+ self-attn. shortcuts	27.7	40.5	31.4
non-lexical shortcuts	27.1	40.6	31.3

Table 4: BLEU scores of shortcut types in EN→DE models; 'test mean' averages over test-sets in table (1).

A related question is whether the encoder and decoder benefit from the addition of lexical shortcuts to self-attention equally. We explore this by disabling shortcuts in either sub-network and comparing the so obtained translation models to one with intact connections. Figure 5 illustrates that best translation performance is obtained by enabling shortcuts in both encoder and decoder. This also improves training stability, as compared to the decoder-only ablated model. The latter may be explained by our use of tied embeddings which receive a stronger training signal from shortcut connections due to 'deep supervision', as this may bias learned embeddings against the sub-network lacking improved lexical connectivity.

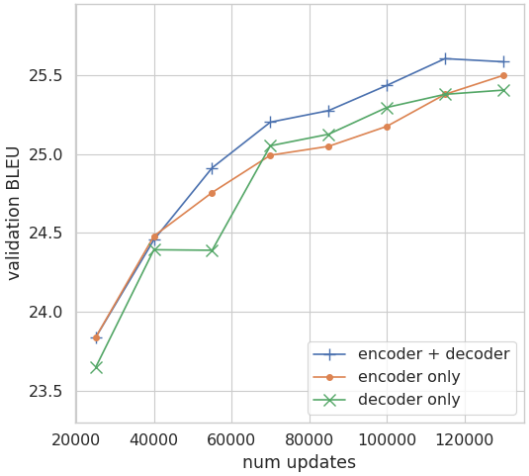

Figure 5: Effect of disabling shortcuts in either sub-network on validation BLEU.

While adding shortcuts improves translation quality, it is not obvious whether this is predominantly due to improved accessibility of lexical content, rather than increased connectivity between network layers, as suggested in (Dou et al., 2018). To isolate the importance of lexical information, we equip the transformer with non-lexical shortcuts connecting each layer n to layer $n-2$, e.g. layer 6 to layer 4.[5] As a result, the number of added connections and parameters is kept identical to lexical shortcuts, while lexical accessibility is reduced. Test-BLEU reported in Table 4 suggests that while non-lexical shortcuts improve over the baseline model, they perform noticeably worse than lexical connections. Therefore, the increase in translation quality associated with lexical shortcuts is not solely attributable to a better signal flow or the increased number of trainable parameters.

4.3 Word-sense disambiguation

Beyond the effects of lexical shortcuts on the transformer's learning dynamics, we are interested in how widening the representation bottleneck affects the properties of the produced translations. One challenging problem in translation which in-

[5]The first layer is connected to the embedding layer, as there is no further antecedent.

tuitively should benefit from the model's increased capacity for learning information drawn from sentence context is word-sense disambiguation.

We examine whether the addition of lexical shortcuts aids disambiguation by evaluating our trained DE→EN models on the *ContraWSD* corpus (Rios et al., 2017). The contrastive dataset is constructed by paring source sentences with multiple translations, varying the translated sense of selected source nouns between translation candidates. A competent model is expected to assign a higher probability to the translation hypothesis containing the appropriate word-sense.

While the standard transformer provides a very strong baseline for the disambiguation task, we nonetheless see improvements as a result of adding direct connections to the embedding layer. While our baseline model reaches an accuracy rating of 88.8%, equipping it with lexical shortcuts further improves the score to 89.5%.

5 Related Work

Within recent literature, proposals have been made for how the standard transformer architecture may be extended, including adaptive model depth (Dehghani et al., 2018), layer-wise transparent attention for the effective training of deeper models (Bapna et al., 2018), and a more effective exploitation of features learned by the deep network (Dou et al., 2018). Our investigation bears strongest resemblance to the latter, concurrent work by introducing additional connectivity to the model. However, rather than establishing new connections between layers indiscriminately, we explicitly seek to facilitate the accessibility of lexical information throughout the model, as this reduces the need to represent and propagate lexical features along hidden states, thereby increasing the model's capacity for learning novel information. As a result, our proposed shortcut connections are sparser, simpler, and more efficient.

Another line of research from which we draw inspiration concerns itself with the analysis of the internal dynamics and learned representations within deep neural networks (Karpathy et al., 2015; Shi et al., 2016; Qian et al., 2016). Here, (Belinkov et al., 2017) and (Belinkov et al., 2018) serve as our primary points of reference by providing a thorough and principled investigation of the extent to which neural translation models are capable of learning linguistic properties from raw text. While the role of lexical features in NMT has not received widespread attention, (Nguyen and Chiang, 2017) note that improving accessibility of source words by the decoder benefits translation quality in low-resource settings.

Our view of the transformer as a model learning to refine input representations through the repeated application of attention is consistent with the iterative estimation paradigm introduced in (Greff et al., 2016). According to this interpretation, given a stack of connected layers sharing the same dimensionality and interlinked through highway or residual connections, the initial layer generates a rough version of the stack's final output, which is iteratively refined by successive layers, e.g. by enriching localized features with information drawn from the surrounding context. The results of our probing studies support this analysis of the transformer, further suggesting that different layers not only refine input features but also learn entirely new information given sufficient capacity, as evidenced by the decrease in similarity between embeddings and hidden states with increasing model depth.

6 Conclusion

In this paper, we have proposed a simple yet effective method for widening the representation bottleneck in the transformer by introducing lexical shortcuts. Our modified models achieve up to 1.4 BLEU (0.9 BLEU on average) improvement on 5 standard WMT datasets, at a small cost in computing time and model size. Our analysis suggests that lexical connections are useful to both encoder and decoder, and remain effective when included in smaller models. Moreover, the addition of shortcuts noticeably reduces the similarity of hidden states to the initial embeddings, indicating that dynamic lexical access aids the network in learning novel, diverse information. We also performed ablation studies comparing different shortcut variants and demonstrated that one effect of lexical shortcuts is an improved WSD capability.

The presented findings offer new insights into the nature of information encoded by the transformer layers, supporting the iterative refinement view of feature learning. In future work, we intend to explore other ways to better our understanding of the refinement process and to help translation models learn more diverse and meaningful internal representations.

References

Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. *arXiv preprint arXiv:1808.07561*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Ondej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (wmt18). In *Proceedings of the Third Conference on Machine Translation*, pages 272–307, Belgium, Brussels. Association for Computational Linguistics.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. *arXiv preprint arXiv:1810.10181*.

Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. 2016. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*.

Barry Haddow, Nikolay Bogoychev, Denis Emelin, Ulrich Germann, Roman Grundkiewicz, Kenneth Heafield, Antonio Valerio Miceli Barone, and Rico Sennrich. 2018. The university of edinburghs submissions to the wmt18 news translation task. In *Proceedings of the Third Conference on Machine Translation*, pages 403–413, Belgium, Brussels. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*, 2, page 3.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570.

Toan Q Nguyen and David Chiang. 2017. Improving lexical choice in neural machine translation. *arXiv preprint arXiv:1710.01329*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Analyzing linguistic knowledge in sequential model of sentence. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 826–835.

Annette Rios, Laura Mascarell, and Rico Sennrich. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. In *Proceedings of the 2nd Conference on Machine Translation, Copenhagen, Denmark*.

Danielle Saunders, Felix Stahlberg, Adria de Gispert, and Bill Byrne. 2018. Multi-representation ensembles and delayed sgd updates improve syntax-based nmt. *arXiv preprint arXiv:1805.00456*.

Helmut Schmid. 1999. Improvements in part-of-speech tagging with an application to german. In *Natural language processing using very large corpora*, pages 13–25. Springer.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? a targeted evaluation of neural machine translation architectures. *arXiv preprint arXiv:1808.08946*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*.

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 *Advances in Neural Information Processing Systems*, pages 5998–6008.

A Supplementary Material

A.1 Training details

The majority of our experiments is conducted using the transformer-BASE configuration, with the number of encoder and decoder layers set to 6 each, embedding and attention dimensionality to 512, number of attention heads to 8, and feed-forward sub-layer dimensionality to 2048. We tie the encoder embedding table with the decoder embedding table and the pre-softmax projection matrix to speed up training, following (Press and Wolf, 2016). All trained models are optimized using Adam (Kingma and Ba, 2014) adhering to the learning rate schedule described in (Vaswani et al., 2017). We set the number of warm-up steps to 4000 for the baseline model, increasing it to 6000 and 8000 when adding lexical shortcuts and feature-fusion, respectively, so as to accommodate the increase in parameter size.

We also evaluate the effect of lexical shortcuts on the larger transformer-BIG model, limiting this set of experiments to EN→DE due to computational constraints. In this more expensive configuration, the baseline model employs 16 attention heads, with attention, embedding, and feed-forward dimensionality doubled to 1024, 1024, and 4096. Warm-up period for all big models is set to 16,000 steps. For our probing experiments, the classifiers used are simple feed-forward networks with a single hidden layer consisting of 512 units, dropout (Srivastava et al., 2014) with $p = 0.5$, and a ReLU non-linearity.

Model	# Parameters	Words / sec.
transformer-BASE	65,166K	29,698
+ lexical shortcuts	71,470K	26,423
+ feature-fusion	84,053K	23,601
transformer-BIG	218,413K	10,215
+ feature-fusion	293,935K	6,769

Table 5: Model size and training speed of the compared transformer variants.

All models are trained concurrently on four Nvidia P100 Tesla GPUs using synchronous data parallelization. Delayed optimization (Saunders et al., 2018) is employed to simulate batch sizes of 25,000 tokens, to be consistent with (Vaswani et al., 2017). Each transformer-BASE model is trained for a total of 150,000 updates, while our transformer-BIG experiments are stopped after 300,000 updates. Validation is performed every 4000 steps, as is check-pointing. Training base models takes ∼43 hours, while the addition of shortcut connections increases training time up to ∼46 hours (∼50 hours with feature-fusion). Table 5 details the differences in parameter size and training speed for the different transformer configurations. Parameters are given in thousands, while speed is averaged over the entire training duration.

Validation-BLEU is calculated using multi-bleu-detok.pl[6] on a reference which we pre- and post-process following the same steps as for the models' inputs and outputs. All reported test-BLEU scores were obtained by averaging the final 5 checkpoints for transformer-BASE and final 16 for transformer-BIG.

A.2 Data pre-processing

We tokenize, clean, and truecase each training corpus using scripts from the Moses toolkit[7], and apply byte-pair encoding (Sennrich et al., 2015) to counteract the open vocabulary issue. Cleaning is skipped for validation and test sets. For EN↔DE and EN→RU we limit the number of BPE merge operations to 32,000 and set the vocabulary threshold to 50. For EN→CS and EN→FI, the number of merge operations is set to 89,500 with a vocabulary threshold of 50, following (Haddow et al., 2018)[8]. In each case, the BPE vocabulary is learned jointly over the source and target language, which necessitated an additional transliteration step for the pre-processing of Russian data[9].

A.3 Probing studies

Cosine similarity scores between the embedding layer and each successive layer in transformer-BASE and its variant equipped with lexical shortcuts are summarized in Figures 6-7.

For our fine-grained probing studies, we evaluated classification accuracy conditioned of part-of-speech tags and sub-word frequencies. For the former, we first parse our test-sets with TreeTagger (Schmid, 1999), projecting tags onto the constituent sub-words of each annotated word. For frequency-based evaluation, we divide sub-words

[6] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu-detok.perl

[7] https://github.com/moses-smt/mosesdecoder

[8] We do not use synthetic data, which makes our results not directly comparable to theirs.

[9] We used 'Lingua Translit' for this purpose.

into ten equally-sized frequency bins, with bin 1 containing the least frequent sub-words and bin 10 containing the most frequent ones. We do not observe any immediately obvious, significant effects of either POS or frequency on the retention of lexical features. While classification accuracy is notably low for infrequent sub-words, this can be attributed to the limited occurrence of the corresponding transformer states in the classifier's training data. Evaluation for EN→DE models is done on newstest2014, while newstest2017 is used for EN→RU models. Figures 8-15 present results for the frequency-based classification. Accuracy scores conditioned on POS tags are visualized in Figures 16-23.

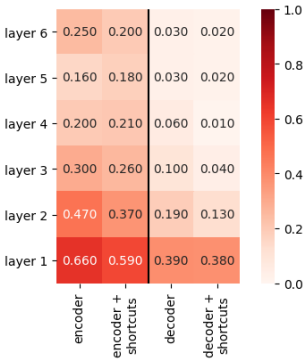

Figure 6: Cosine similarity of transformer-BASE for EN→DE (evaluated on newstest2014).

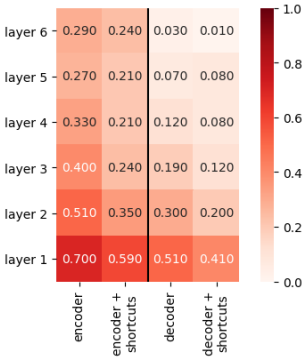

Figure 7: Cosine similarity of transformer-BASE for EN→RU (evaluated on newstest2017).

We also investigated the activation patterns of the lexical shortcut gates. However, despite their essential status for the successful training of transformer variants equipped with lexical connections, we were unable to discern any distinct patterns in the activations of the individual gates, which tend to prioritize lexical and hidden features to an equal degree regardless of training progress or (sub-)word characteristics.

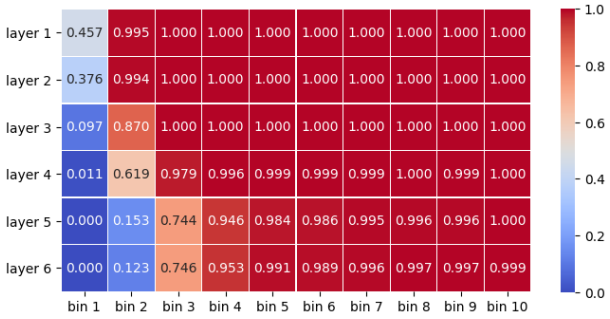

Figure 8: Frequency-based classification accuracy on states from the EN→DE encoder.

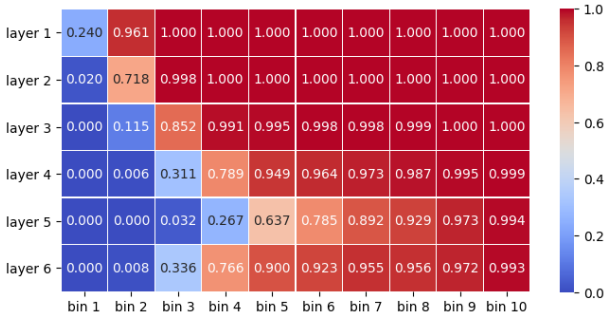

Figure 9: Frequency-based classification accuracy on states from the EN→DE encoder + lexical shortcuts.

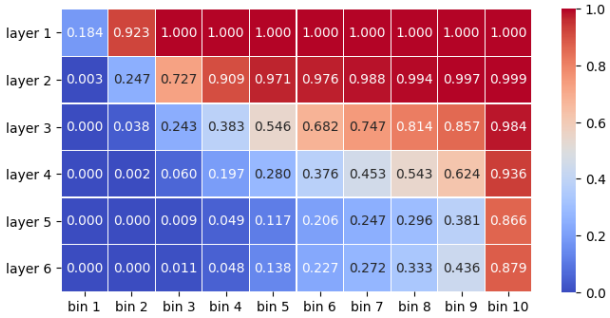

Figure 10: Frequency-based classification accuracy on states from the EN→DE decoder.

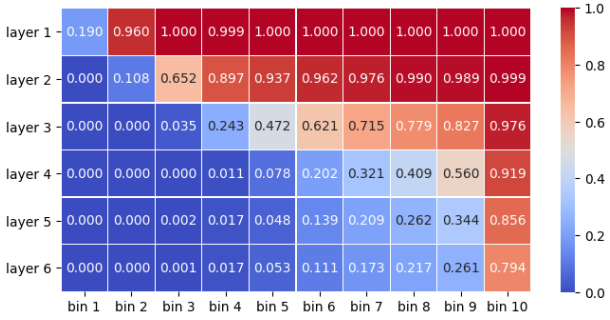

Figure 11: Frequency-based classification accuracy on states from the EN→DE decoder + lexical shortcuts.

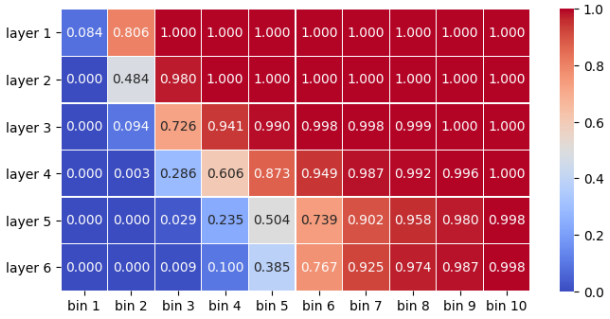

Figure 12: Frequency-based classification accuracy on states from the EN→RU encoder.

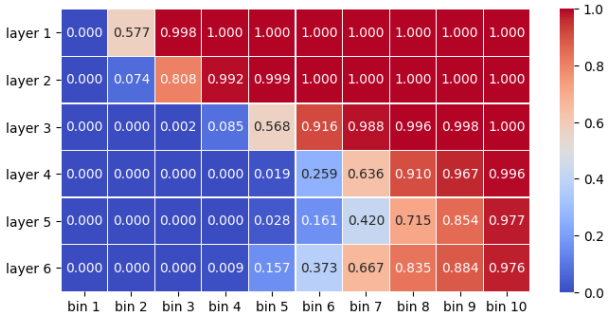

Figure 13: Frequency-based classification accuracy on states from the EN→RU encoder + lexical shortcuts.

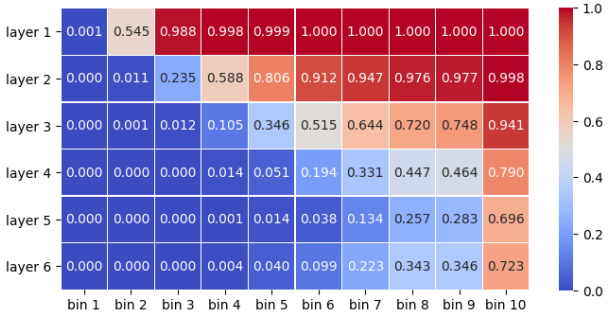

Figure 14: Frequency-based classification accuracy on states from the EN→RU decoder.

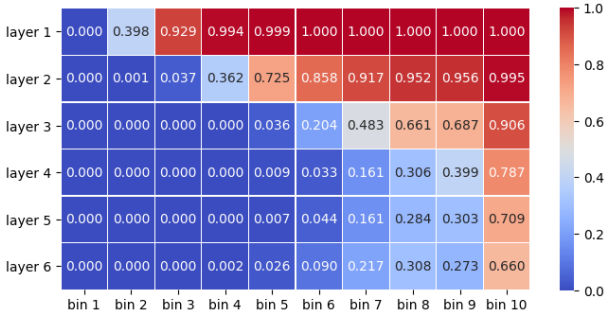

Figure 15: Frequency-based classification accuracy on states from the EN→RU decoder + lexical shortcuts.

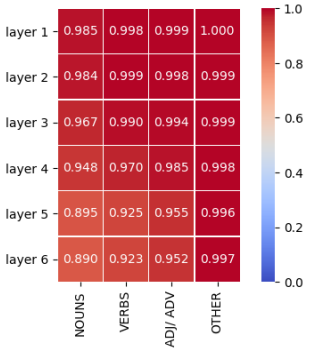

Figure 16: POS-based classification accuracy on states from the EN→DE encoder.

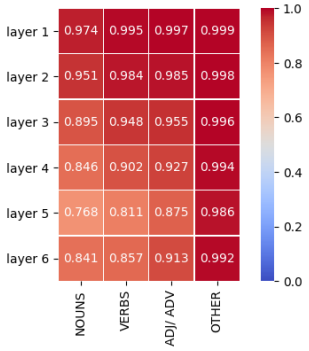

Figure 17: POS-based classification accuracy on states from the EN→DE encoder + lexical shortcuts.

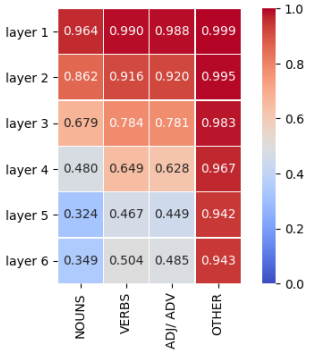

Figure 18: POS-based classification accuracy on states from the EN→DE decoder.

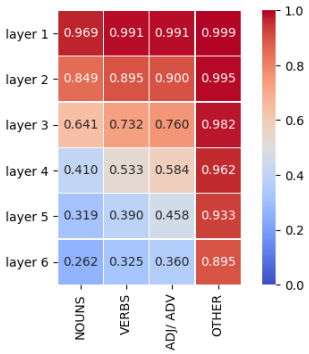

Figure 19: POS-based classification accuracy on states from the EN→DE decoder + lexical shortcuts.

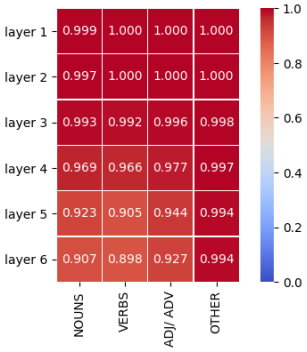

Figure 20: POS-based classification accuracy on states from the EN→RU encoder.

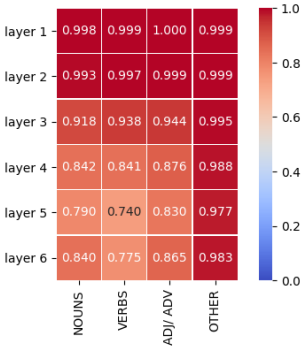

Figure 21: POS-based classification accuracy on states from the EN→RU encoder + lexical shortcuts.

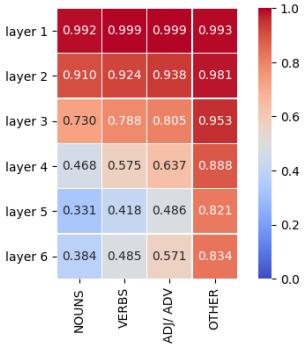

Figure 22: POS-based classification accuracy on states from the EN→RU decoder.

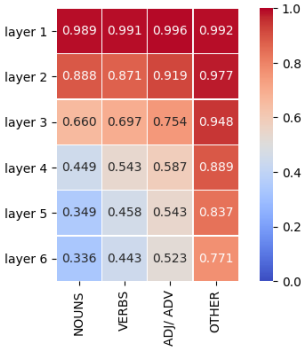

Figure 23: POS-based classification accuracy on states from the EN→RU decoder + lexical shortcuts.