On the Sub-layer Functionalities of Transformer Decoder

Yilin Yang*
Oregon State University
yilinyang721@gmail.com

Longyue Wang
Tencent AI Lab
vinnylywang@tencent.com

Shuming Shi
Tencent AI Lab
shumingshi@tencent.com

Prasad Tadepalli
Oregon State University
tadepall@oregonstate.edu

Stefan Lee
Oregon State University
leestef@oregonstate.edu

Zhaopeng Tu
Tencent AI Lab
zptu@tencent.com

Abstract

There have been significant efforts to interpret the encoder of Transformer-based encoder-decoder architectures for neural machine translation (NMT); meanwhile, the decoder remains largely unexamined despite its critical role. During translation, the decoder must predict output tokens by considering both the source-language text from the encoder and the target-language prefix produced in previous steps. In this work, we study how Transformer-based decoders leverage information from the source and target languages – developing a universal probe task to assess how information is propagated through each module of each decoder layer. We perform extensive experiments on three major translation datasets (WMT En-De, En-Fr, and En-Zh). Our analysis provides insight on when and where decoders leverage different sources. Based on these insights, we demonstrate that the residual feed-forward module in each Transformer decoder layer can be dropped with minimal loss of performance – a significant reduction in computation and number of parameters, and consequently a significant boost to both training and inference speed.

1 Introduction

Transformer models have advanced the state-of-the-art on a variety of natural language processing (NLP) tasks, including machine translation (Vaswani et al., 2017), natural language inference (Shen et al., 2018), semantic role labeling (Strubell et al., 2018), and language representation (Devlin et al., 2019). However, so far not much is known about the internal properties and functionalities it learns to achieve its superior performance, which poses significant challenges for human understanding of the model and potentially designing better architectures.

Recent efforts on interpreting Transformer models mainly focus on assessing the encoder representations (Raganato et al., 2018; Yang et al., 2019; Tang et al., 2019a) or interpreting the multi-head self-attentions (Li et al., 2018; Voita et al., 2019; Michel et al., 2019). At the same time, there have been few attempts to interpret the decoder side, which we believe is also of great interest, and should be taken into account while explaining the encoder-decoder networks. The reasons are threefold: (a) the decoder takes both source and target as input, and implicitly performs the functionalities of both alignment and language modeling, which are at the core of machine translation; (b) the encoder and decoder are tightly coupled in that the output of the encoder is fed to the decoder and the training signals for the encoder are back-propagated from the decoder; and (c) recent studies have shown that the boundary between the encoder and decoder is blurry, since some of the encoder functionalities can be substituted by the decoder cross-attention modules (Tang et al., 2019b).

In this study, we interpret the Transformer decoder by investigating when and where the decoder utilizes source or target information across its stacking modules and layers. Without loss of generality, we focus on the *representation evolution*[1] within a Transformer decoder. To this end, we introduce a novel sub-layer[2] split with respect to their functionalities: *Target Exploitation Module* (TEM) for exploiting the representation from translation history, *Source Exploitation Module* (SEM) for exploiting the source-side representation, and *Information Fusion Module* (IFM) to combine representations from the other two (§2.2).

Further, we design a universal probing scheme

*Work done when interning at Tencent AI Lab.

[1]By "evolution", we denote the progressive trend from the first layer till the last.

[2]Throughout this paper, we use the terms "sub-layer" and "module" interchangeably.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4799–4811

to quantify the amount of specific information embedded in network representations. By probing both source and target information from decoder sub-layers, and by analyzing the alignment error rate (AER) and source coverage rate, we arrive at the following findings:

- SEM guides the representation evolution within NMT decoder (§3.1).
- Higher-layer SEMs accomplish the functionality of word alignment, while lower-layer ones construct the necessary contexts (§3.2).
- TEMs are critical to helping SEM build word alignments, while their stacking order is not essential (§3.2).

Last but not least, we conduct a fine-grained analysis on the information fusion process within IFM. Our key contributions in this work are:

1. We introduce a novel sub-layer split of Transformer decoder with respect to their functionalities.
2. We introduce a universal probing scheme from which we derive aforementioned conclusions about the Transformer decoder.
3. Surprisingly, we find that the de-facto usage of residual FeedForward operations are not efficient, and could be removed in totality with minimal loss of performance, while significantly boosting the training and inference speeds.

2 Preliminaries

2.1 Transformer Decoder

NMT models employ an encoder-decoder architecture to accomplish the translation process in an end-to-end manner. The encoder transforms the source sentence into a sequence of representations, and the decoder generates target words by dynamically attending to the source representations. Typically, this framework can be implemented with a recurrent neural network (RNN) (Bahdanau et al., 2015), a convolutional neural network (CNN) (Gehring et al., 2017), or a Transformer (Vaswani et al., 2017). We focus on the Transformer architecture, since it has become the state-of-the-art model on machine translation tasks, as well as various text understanding (Devlin et al., 2019) and generation (Radford et al., 2019) tasks.

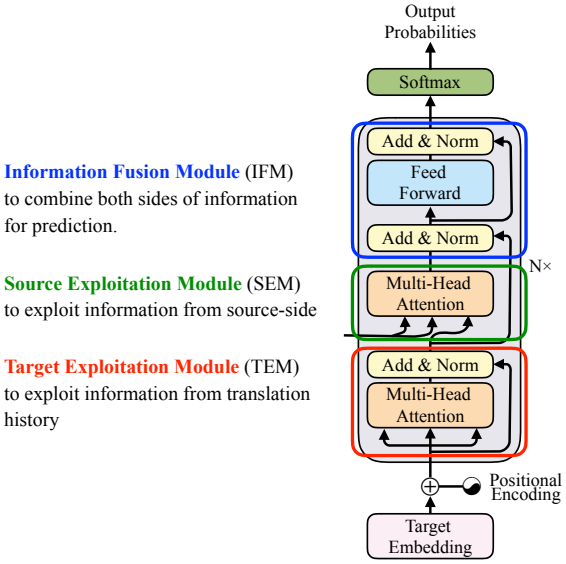


Figure 1: A sub-layer splitting of Transformer decoder with respect to their functionalities.

Specifically, the decoder is composed of a stack of N identical layers, each of which has three sub-layers, as illustrated in Figure 1. A residual connection (He et al., 2016) is employed around each of the three sub-layers, followed by layer normalization (Ba et al., 2016) ("Add & Norm"). The first sub-layer is a self-attention module that performs self-attention over the previous decoder layer:

$$\mathbf{C}_d^n = \textsc{Ln}\big(\textsc{Att}(\mathbf{Q}_d^n, \mathbf{K}_d^n, \mathbf{V}_d^n) + \mathbf{L}_d^{n-1}\big)$$

where $\textsc{Att}(\cdot)$ and $\textsc{Ln}(\cdot)$ denote the self-attention mechanism and layer normalization. $\mathbf{Q}_d^n$, $\mathbf{K}_d^n$, and $\mathbf{V}_d^n$ are query, key and value vectors that are transformed from the (n-1)-th layer representation $\mathbf{L}_d^{n-1}$. The second sub-layer performs attention over the output of the encoder representation:

$$\mathbf{D}_d^n = \textsc{Ln}\big(\textsc{Att}(\mathbf{C}_d^n, \mathbf{K}_e^N, \mathbf{V}_e^N) + \mathbf{C}_d^n\big)$$

where $\mathbf{K}_e^N$ and $\mathbf{V}_e^N$ are transformed from the top encoder representation $\mathbf{L}_e^N$. The final sub-layer is a position-wise fully connected feed-forward network with ReLU activations:

$$\mathbf{L}_d^n = \textsc{Ln}\big(\textsc{Ffn}(\mathbf{D}_d^n) + \mathbf{D}_d^n\big)$$

The top decoder representation $\mathbf{L}_d^N$ is then used to generate the final prediction.

2.2 Sub-Layer Partition

In this work, we aim to reveal how a Transformer decoder accomplishes the translation process utilizing both source and target inputs. To this end,

we split each decoder layer into three modules with respect to their different functionalities over the source or target inputs, as illustrated in Figure 1:

- *Target Exploitation Module* (TEM) consists of the self-attention operation and a residual connection, which exploits the target-side translation history from previous layer representations.
- *Source Exploitation Module* (SEM) consists only of the encoder attention, which dynamically selects relevant source-side information for generation.
- *Information Fusion Module* (IFM) consists of the rest of the operations, which fuse source and target information into the final layer representation.

Compared with the standard splits (Vaswani et al., 2017), we associate the "Add&Norm" operation after encoder attention with the IFM, since it starts the process of information fusion by a simple additive operation. Consequently, the functionalities of the three modules are well-separated.

2.3 Research Questions

Modern Transformer decoder is implemented as multiple identical layers, in which the source and target information are exploited and evolved layer-by-layer. One research question arises naturally:

> **RQ1**. How do source and target information evolve within the decoder layer-by-layer and module-by-module?

In Section 3.1, we introduce a universal probing scheme to quantify the amount of information embedded in decoder modules and explore their evolutionary trends. The general trend we find is that higher layers contain more source and target information, while the sub-layers behave differently. Specifically, the amount of information contained by SEMs would first increase and then decrease. In addition, we establish that SEM guides both source and target information evolution within the decoder.

Since SEMs are critical to the decoder representation evolution, we conduct a more detailed study into the internal behaviors of the SEMs. The exploitation of source information is also closely related to the inadequate translation problem – a key weakness of NMT models (Tu et al., 2016). We try to answer the following research question:

> **RQ2**. How does SEM exploit the source information in different layers?

In Section 3.2, we investigate how the SEMs transform the source information to the target side in terms of alignment accuracy and coverage ratio (Tu et al., 2016). Experimental results show that higher layers of SEM modules accomplish word alignment, while lower layer ones exploit necessary contexts. This also explains the representation evolution of source information: lower layers collect *more* source information to obtain a global view of source input, and higher layers extract *less* aligned source input for accurate translation.

Of the three sub-layers, IFM modules conceptually appear to play a key role in merging source and target information – raising our final question:

> **RQ3**. How does IFM fuse source and target information on the operation level?

In Section 3.3, we first conduct a fine-grained analysis of the IFM module on the operation level, and find that a simple "Add&Norm" operation performs just as well at fusing information. Thus, we *simplify* the IFM module to be only one Add&Norm operation. Surprisingly, this performs similarly to the full model while significantly reducing the number of parameters and consequently boosting both training and inference speed.

3 Experiments

Data To make our conclusions compelling, all experiments and analysis are conducted on three representative language pairs. For English⇒German (En⇒De), we use WMT14 dataset that consists of 4.5M sentence pairs. The English⇒Chinese (En⇒Zh) task is conducted on WMT17 corpus, consisting of 20.6M sentence pairs. For English⇒French (En⇒Fr) task, we use WMT14 dataset that comprises 35.5M sentence pairs. English and French have many aspects in common while English and German differ in word order, requiring a significant amount of reordering in translation. Besides, Chinese belongs to a different language family compared to the others.

Models We conducted the experiments on the state-of-the-art Transformer (Vaswani et al., 2017), and implemented our approach with the open-source toolkit FairSeq (Ott et al., 2019). We fol-

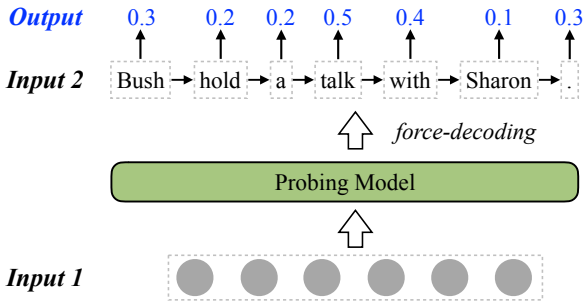


Figure 2: Illustration of the information probing model, which reads the representation of a decoder module ("Input 1") and the word sequence to recover ("Input 2"), and outputs the generation probability ("Output").

low the setting of Transformer-Base in Vaswani et al. (2017), which consists of 6 stacked encoder/decoder layers with the model size being 512. We train our models on 8 NVIDIA P40 GPUs, where each is allocated with a batch size of 4,096 tokens. We use Adam optimizer (Kingma and Ba, 2015) with 4,000 warm-up steps.[3]

3.1 Representation Evolution Across Layers

In order to quantify and visualize the representation evolution, we design a universal probing scheme to quantify the source (or target) information stored in network representations.

Task Description Intuitively, the more the source (or target) information stored in a network representation, the more probably a trained reconstructor could recover the source (or target) sequence. Since the lengths of source sequence and decoder representations are not necessarily the same, the widely-used classification-based probing approaches (Belinkov et al., 2017; Tenney et al., 2019b) cannot be applied to this task. Accordingly, we cast this task as a generation problem – evaluating the likelihood of generating the word sequence conditioned on the input representation.

Figure 2 illustrates the architecture of our probing scheme. Given a representation sequence from decoder $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_M\}$ and the source (or target) word sequence to be recovered $\mathbf{x} = \{x_1, \ldots, x_N\}$ the recovery likelihood is calculated as the perplexity (i.e. negative log-likelihood) of forced-decoding the word sequence:

$$PPL(\mathbf{x}|\mathbf{H}) = \sum_{n=1}^{N} -\log P(x_n|x_{<n}, \mathbf{H}) \quad (1)$$

[3] More implementation details are in Sec A.1.

The lower the recovery perplexity, the more the source (or target) information stored in the representation. The probing model can be implemented as any architecture. For simplicity, we use a one-layer Transformer decoder. *We train the probing model to recover both source and target sequence from all decoder sub-layer representations.* During training, we fix the NMT model parameters and train the probing model on the MT training set to minimize the recovery perplexity in Equation 1.

Task Discussion The above probing scheme is a general framework applicable to probing any given sequence from a network representation. When we probe for the source sequence, the probing model is analogous to an auto-encoder (Bourlard and Kamp, 1988; Vincent et al., 2010), which reconstructs the original input from the network representations. When we probe for the target sequence, we apply an attention mask to the *probing decoder* to avoid direct copying from the input of translation histories. Contrary to source probing, the target sequence is never seen by the model.

In addition, our proposed scheme can also be applied to probe linguistic properties that can be represented in a sequential format. For instance, we could probe source constituency parsing information, by training a probing model to recover the linearized parsing sequence (Vinyals et al., 2015). Due to space limitations, we leave the linguistic probing to future work.

Probing Results Figure 3 shows the results of our information probing conducted on the heldout set. We have a few observations:

- The evolution trends of TEM and IFM are largely the same. Specifically, the curve of TEM is very close to that of IFM shifted up by one layer. Since TEM representations are two operations (self-attn. and Add&Norm) away from the previous layer IFM, this observation indicates TEMs do not significantly affect the amount of source/target information. [4]

- *SEM guides both source and target information evolution.* While closely observing the curves, the trend of layer representations (i.e. IFM) is always led by that of SEM. For example, as the PPL of SEM transitions from

[4] TEM may change the order or distribution of source/target information, which are not captured by our probing experiments.

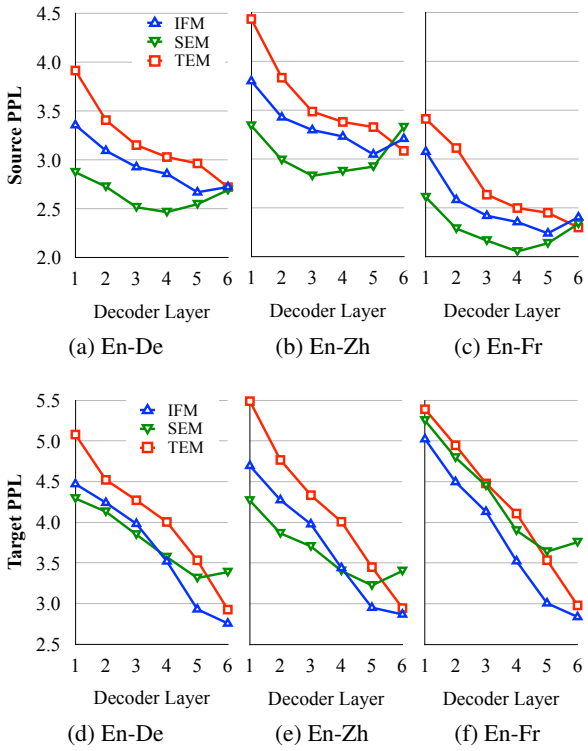


Figure 3: Evolution trends of source (upper panel) and target (bottom panel) information embedded in the decoder modular representations across layers. Lower perplexity ("PPL") denotes more information embedded in the representations.

decreases to increases, the PPL of IFM slows down the decreases and starts increasing as an aftermath. This is intuitive: in machine translation, source and target sequences should contain equivalent information, thus the target generation should largely follow the lead of source information (from SEM representations) to guarantee its adequacy.

- For IFM, the amount of target information consistently increases in higher layers – a consistent decrease of PPL in Figures 3(d-f). While source information goes up in the lower layers, it drops in the highest layer (Figures 3(a-c)).

Since SEM representations are critical to decoder evolution, we turn to investigate how SEM exploit source information, in the hope of explaining the decoder information evolution.

3.2 Exploitation of Source Information

Ideally, SEM should *accurately* and *fully* incorporate the source information for the decoder. Accordingly, we evaluate how well SEMs accomplish the expected functionality from two perspectives.

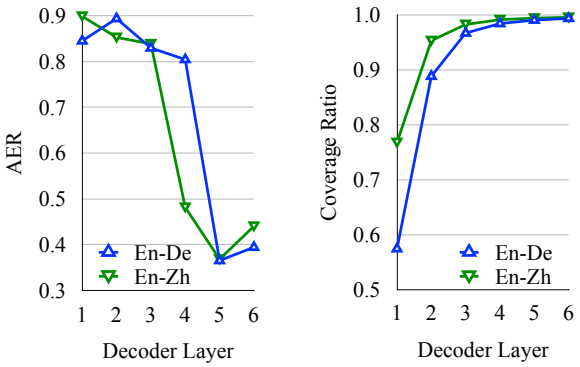


Figure 4: Behavior of the SEM in terms of (a) alignment quality measured in AER (the lower, the better), and (b) the cumulative coverage of source words.

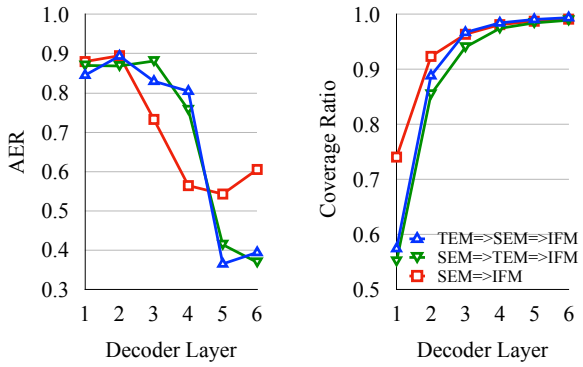
Word Alignment. Previous studies generally interpret the attention weights of SEM as word alignments between source and target words, which can measure whether SEMs select the most relevant part of source information for each target token (Tu et al., 2016; Li et al., 2019; Tang et al., 2019b). We follow previous practice to merge attention weights from the SEM attention heads, and to extract word alignments by selecting the source word with the highest attention weight for each target word. We calculate the alignment error rate (AER) scores (Och and Ney, 2003) for word alignments extracted from SEM of each decoder layer.

Cumulative Coverage. Coverage is commonly used to evaluate whether the source words are fully translated (Tu et al., 2016; Kong et al., 2019). We use the above extracted word alignments to identify the set of source words A_i, which are covered (i.e., aligned to at least one target word) at each layer. We then propose a new metric *cumulative coverage ratio* $C_{\leq i}$ to indicate how many source words are covered by the layers $\leq i$:

$$C_{\leq i} = \frac{|A_1 \cup \cdots \cup A_i|}{N} \quad (2)$$

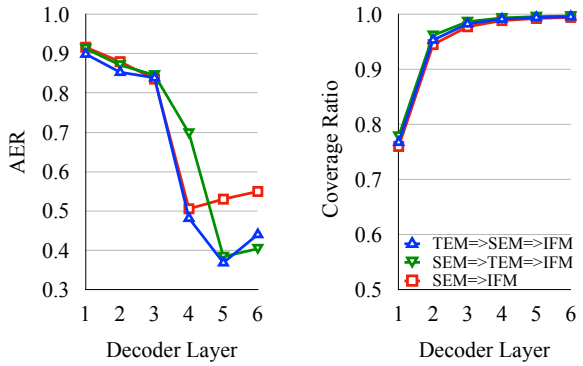
where N is the number of total source words. This metric indicates the completeness of source information coverage till layer i.

Dataset We conducted experiments on two manually-labeled alignment datasets: RWTH En-



(a) Alignment (b) Cumulative Coverage

Figure 5: Effects of the stacking order of TEM and SEM on the En-De dataset.



(a) Alignment (b) Cumulative Coverage

Figure 6: Effects of the stacking order of TEM and SEM on the En-Zh dataset.

De[5] and En-Zh (Liu and Sun, 2015). The alignments are extracted from NMT models trained on the WMT En-De and En-Zh dataset.

Results Figure 4 demonstrates our results on word alignment and cumulative coverage. We find that the *lower-layer SEMs focus on gathering source contexts* (rapid increase of cumulative coverage with poor word alignment), while *higher-layer ones play the role of word alignment* with the lowest AER score of less than 0.4 at the 5th layer. The 4^{th} layer and the 3^{rd} layer separate the two roles for En-De and En-Zh respectively. Correspondingly, they are also the turning points (PPL from decreases to increases) of source information evolution in Figure 3 (a,b). Together with conclusions from Sec. 3.1, we demonstrate the general pattern of SEM: SEM tends to cover more source content and gain increasing amount of source information up to a turning point of 3^{rd} or 4^{th} layer, after which it starts only attending to the most relevant source tokens and contains decreasing amount of total source information.

[5] https://www-i6.informatik.rwth-aachen.de/goldAlignment

Decoder	En-De	En-Zh	En-Fr
TEM⇒SEM⇒IFM	27.45	32.24	40.39
SEM⇒TEM⇒IFM	27.61	33.62	40.89
SEM⇒IFM	22.76	30.06	37.56

Table 1: Effects of the stacking order of decoder sublayers on translation quality in terms of BLEU score.

Depth	En-De	En-Zh	En-Fr	Ave.
6	27.45	32.24	40.39	33.36
4	27.52	31.35	40.37	33.08
12	27.64	32.50	40.44	33.53

Table 2: Effects of various decoder depths on translation quality in terms of BLEU score.

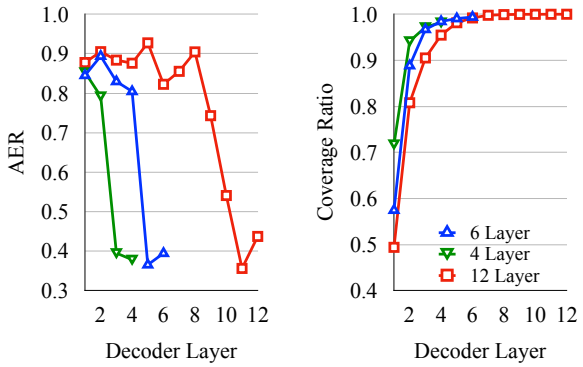
TEM Modules Since TEM representations serve as the query vector for encoder attention operations (shown in Figure 1), we naturally hypothesize that TEM is helping SEM on building alignments. To verify that, we remove TEM from the decoder ("SEM⇒IFM"), which significantly increases the alignment error from 0.37 to 0.54 (in Figure 5), and leads to a serious decrease of translation performance (BLEU: 27.45 ⇒ 22.76, in Table 1) on En-De, while results on En-Zh also confirms it (in Figure 6). *This indicates that TEM is essential for building word alignment.*

However, reordering the stacking of TEM and SEM ("SEM⇒TEM⇒IFM") does not affect the alignment or translation qualities (BLEU: 27.45 vs. 27.61). These results provide empirical support for recent work on merging TEM and SEM modules (Zhang et al., 2019).

Robustness to Decoder Depth To verify the robustness of our conclusions, we vary the depth of NMT decoder and train it from scratch. Table 2 demonstrates the results on translation quality, which generally show that more decoder layers bring better performance. Figure 7 shows that SEM behaves similarly regardless of depth. These results demonstrate the robustness of our conclusions.

3.3 Information Fusion in Decoder

We now turn to the analysis of IFM. Within the Transformer decoder, IFM plays the critical role of fusing the source and target information by merg-

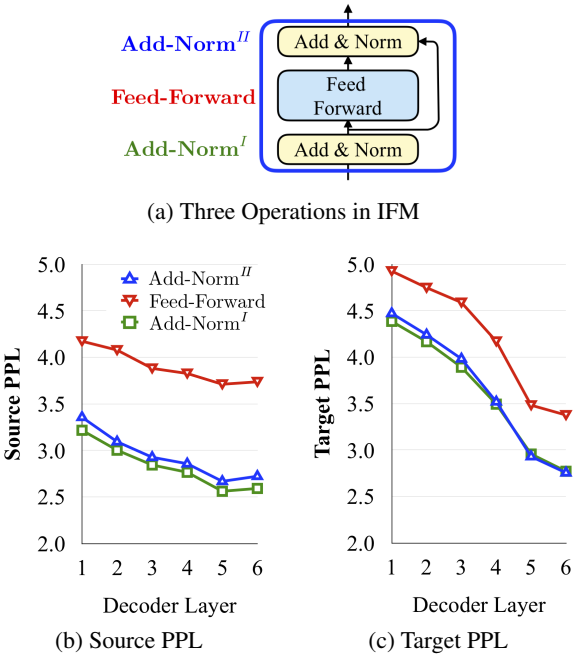


(a) Word Alignment (b) Cumulative Coverage

Figure 7: Effects of decoder depths on SEM behaviors on the En-De task.

Model	Self-Attn.	Enc-Attn.	FFN
Base	6.3M	6.3M	12.6M
Big	25.2M	25.2M	50.4M

Table 3: Number of parameters taken by three major operations within Transformer Base and Big decoder.[6]



(a) Three Operations in IFM (b) Source PPL (c) Target PPL

Figure 8: Illustration of (a) three operations within IFM, and (b,c) the source and target information evolution within IFM on En-De task.

ing representations from SEM and TEM. To study the information fusion process, we conduct a more fine-grained analysis on IFM at the operation level.

Fine-Grained Analysis on IFM As shown in Figure 8(a), IFM contains three operations:

- *Add-Norm*^{I} linearly sums and normalizes the representations from SEM and TEM;
- *Feed-Forward* non-linearly transforms the fused source and target representations;
- *Add-Norm*^{II} again linearly sums and normalizes the representations from the above two.

IFM Analysis Results Figures 8 (b) and (c) respectively illustrate the source and target information evolution within IFM. Surprisingly, Add-Norm^{I} contains a similar amount of, if not more, source (and target) information than Add-Norm^{II}, while the Feed-Forward curve deviates significantly from both. This indicates that the residual Feed-Forward operation may not affect the source (and target) information evolution, and one Add&Norm operation may be sufficient for information fusion.

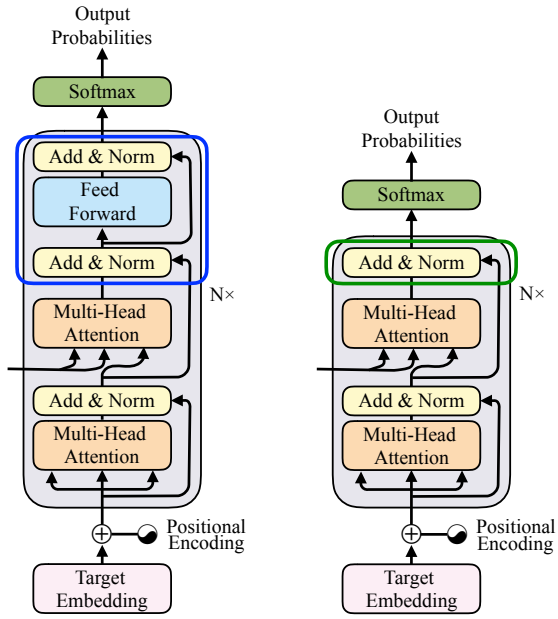
Simplified Decoder To empirically demonstrate whether one Add&Norm operation is already sufficient, we remove all other operations, leaving just one Add&Norm operation for the IFM. The architectural change is illustrated in Figure 9(b), and we dub it the "*simplified decoder*".

Simplified Decoder Results Table 4 reports the translation performance of both architectures on all three major datasets, while Figure 10 illustrates the information evolution of both on WMT En-De. We find the simplified model reaches comparable performance with only a minimal drop of 0.1-0.3 BLEU on En-De and En-Fr, while observing 0.9 BLEU gains on En-Zh.[7] To further assess the translation performance, we manually evaluate 100 translations sampled from the En-Zh test set. On the scale of 1 to 5, we find that the simplified decoder obtains a fluency score of 4.01 and an adequacy score of 3.87, which is approximately equivalent to that of the standard decoder, i.e. 4.00 for fluency and 3.86 for adequacy (in Table 5).

On the other hand, since the simplified decoder drops the operations (FeedForward) with most parameters (shown in Table 3), we also expect a significant increase on training and inference speeds. From Table 4, we confirm a consistent boost of both training and inference speeds by approximately 11-14%. To demonstrate the robustness, we also confirm our findings under Transformer big settings (Vaswani et al., 2017), whose results are

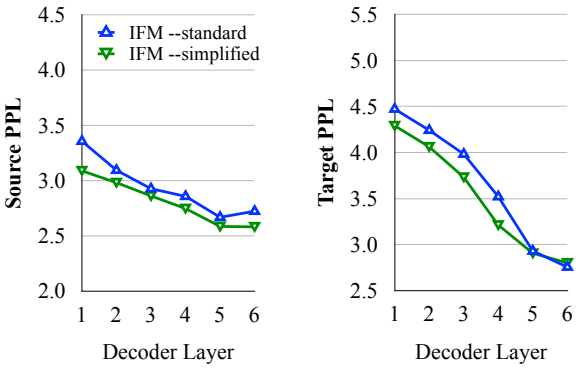
[6] As a comparison, the total number of parameters in Base and Big models are 62.9M and 213.9M respectively on En-De.

[7] Simplified models are trained with the same hyper-parameters as standard ones, which may be suboptimal as the number of parameters is significantly reduced.



(a) Standard (b) Simplified

Figure 9: Illustration of (a) the standard decoder, and (b) the simplified decoder with simplified IFM.



(a) Source PPL (b) Target PPL

Figure 10: Comparison of IFM information evolution between the standard and simplified decoder on En-De.

	Decoder	BLEU	#Train.	#Infer.
En-De	Standard	27.45	63.93K	65.35
	Simplified	27.29	71.08K	72.93
	△	*-0.16*	*+11.18%*	*+11.60%*
En-Zh	Standard	32.24	32.49K	38.55
	Simplified	33.15	36.59K	54.06
	△	*+0.91*	*+12.62%*	*+40.23%*
En-Fr	Standard	40.39	68.28K	58.97
	Simplified	40.07	76.03K	67.23
	△	*-0.32*	*+11.35%*	*+14.01%*

Table 4: Performance of the simplified Base decoder. "#Train" denotes the training speed (words per second) and "#Infer." denotes the inference speed (sentences per second). Results are averages of three runs.

shown in Section A.2. The lower PPL in Figure 10 suggests that the simplified model also contains consistently more source and target information across its stacking layers.

Our results demonstrate that a single Add&Norm is indeed sufficient for IFM, and the simplified model reaches comparable performance with a significant parameter reduction and a noticable 11-14% boost on training and inference speed.

4 Related Work

Interpreting Encoder Representations Previous studies generally focus on interpreting the encoder representations by evaluating how informative they are for various linguistic tasks (Conneau et al., 2018; Tenney et al., 2019b), for both RNN models (Shi et al., 2016; Belinkov et al., 2017; Bisazza and Tump, 2018; Blevins et al., 2018) and Transformer models (Raganato et al., 2018; Tang et al., 2019a; Tenney et al., 2019a; Yang et al., 2019). Although they found that a certain amount of linguistic information is captured by encoder representations, it is still unclear how much encoded information is used by the decoder. Our work bridges this gap by interpreting how the Transformer decoder exploits the encoded information.

Interpreting Encoder Self-Attention In recent years, there has been a growing interest in interpreting the behaviors of attention modules. Previous studies generally focus on the self-attention in the encoder, which is implemented as multi-head attention. For example, Li et al. (2018) showed that different attention heads in the encoder-side self-attention generally attend to the same position. Voita et al. (2019) and Michel et al. (2019) found that only a few attention heads play consistent and often linguistically-interpretable roles, and others can be pruned. Geng et al. (2020) empirically validated that a selective mechanism can mitigate the problem of word order encoding and structure modeling of encoder-side self-attention. In this work, we investigated the functionalities of decoder-side attention modules for exploiting both source and target information.

Interpreting Encoder Attention The encoder-attention weights are generally employed to interpret the output predictions of NMT models. Recently, Jain and Wallace (2019) showed that atten-

Model	Fluency	Adequacy
Standard (Base)	4.00	3.86
Simplified (Base)	4.01	3.87

Table 5: Human evaluation of translation performance of both standard and simplified decoders on 100 samples from En-Zh test set, on the scale of 1 to 5.

tion weights are weakly correlated with the contribution of source words to the prediction. He et al. (2019) used the integrated gradients to better estimate the contribution of source words. Related to our work, Li et al. (2019) and Tang et al. (2019b) also conducted word alignment analysis on the same De-En and Zh-En datasets with Transformer models[8]. We use similar techniques to examine word alignment in our context; however, we also introduce a forced-decoding-based probing task to closely examine the information flow.

Understanding and Improving NMT Recent work started to improve NMT based on the findings of interpretation. For instance, Belinkov et al. (2017, 2018) pointed out that different layers prioritize different linguistic types, based on which Dou et al. (2018) and Yang et al. (2019) simultaneously exposed all of these signals to the subsequent process. Dalvi et al. (2017) explained why the decoder learns considerably less morphology than the encoder, and then explored to explicitly inject morphology in the decoder. Emelin et al. (2019) argued that the need to represent and propagate lexical features in each layer limits the model's capacity, and introduced gated shortcut connections between the embedding layer and each subsequent layer. Wang et al. (2020) revealed that miscalibration remains a severe challenge for NMT during inference, and proposed a graduated label smoothing that can improve the inference calibration. In this work, based on our information probing analysis, we simplified the decoder by removing the residual feedforward module in totality, with minimal loss of translation quality and a significant boost of both training and inference speeds.

5 Conclusions

In this paper, we interpreted NMT Transformer decoder by assessing the evolution of both source and target information across layers and modules. To this end, we investigated the information functionalities of decoder components in the translation process. Experimental results on three major datasets revealed several findings that help understand the behaviors of Transformer decoder from different perspectives. We hope that our analysis and findings could inspire architectural changes for further improvements, such as 1) improving the word alignment of higher SEMs by incorporating external alignment signals; 2) exploring the stacking order of SEM, TEM and IFM sub-layers, which may provide a more effective way to transform information; 3) further pruning redundant sub-layers for efficiency.

Since our analysis approaches are not limited to the Transformer model, it is also interesting to explore other architectures such as RNMT (Chen et al., 2018), ConvS2S (Gehring et al., 2017), or on document-level NMT (Wang et al., 2017, 2019). In addition, our analysis methods can be applied to other sequence-to-sequence tasks such as summarization and grammar error correction, whose source and target sides are in the same language. We leave those tasks for future work.

[8]We find our results are more similar to that of Tang et al. (2019b). Also, our results are reported on the En⇒De and En⇒Zh directions, while they report results in the inverse directions.

Acknowledgments

Tadepalli acknowledges the support of DARPA under grant number N66001-17-2-4030. The authors thank the anonymous reviewers for their insightful and helpful comments.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

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

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*.

Arianna Bisazza and Clara Tump. 2018. The lazy encoder: A fine-grained analysis of the role of morphology in neural machine translation. In *EMNLP*.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep rnns encode soft hierarchical syntax. In *ACL*.

Hervé Bourlard and Yves Kamp. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*.

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. *ACL*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What You Can Cram into A Single $&!#* Vector: Probing Sentence Embeddings for Linguistic Properties. *ACL*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *IJCNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

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

Denis Emelin, Ivan Titov, and Rico Sennrich. 2019. Widening the representation bottleneck in neural machine translation with lexical shortcuts. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Xinwei Geng, Longyue Wang, Xing Wang, Bing Qin, Ting Liu, and Zhaopeng Tu. 2020. How Does Selective Mechanism Improve Self-Attention Networks? In *ACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Shilin He, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael Lyu, and Shuming Shi. 2019. Towards understanding neural machine translation with word importance. In *EMNLP*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *NAACL*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Xiang Kong, Zhaopeng Tu, Shuming Shi, Eduard Hovy, and Tong Zhang. 2019. Neural machine translation with adequacy-oriented learning. In *AAAI*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. *EMNLP*.

Xintong Li, Guanlin Li, Lemao Liu, Max Meng, and Shuming Shi. 2019. On the word alignment from neural machine translation. In *ACL*.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *AAAI*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *NeurIPS*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*.

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 *NAACL-HLT*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *arXiv*.

Alessandro Raganato, Jörg Tiedemann, et al. 2018. An analysis of encoder representations in transformer-based machine translation. In *BlackboxNLP Workshop*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. In *AAAI*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *EMNLP*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *EMNLP*.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2019a. Encoders help you disambiguate word senses in neural machine translation. In *EMNLP*.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2019b. Understanding neural machine translation by simplification: The case of encoder-free models. In *RANLP*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical nlp pipeline. In *ACL*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *ICLR*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *ACL*.

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 *NIPS*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*.

Longyue Wang, Zhaopeng Tu, Xing Wang, and Shuming Shi. 2019. One model to learn both: Zero pronoun prediction and translation. In *EMNLP*.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *EMNLP*.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. 2020. On the Inference Calibration of Neural Machine Translation. In *ACL*.

Baosong Yang, Jian Li, Derek Wong, Lidia S. Chao, Xing Wang, and Zhaopeng Tu. 2019. Context-Aware Self-Attention Networks. In *AAAI*.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Assessing the ability of self-attention networks to learn word order. In *ACL*.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Improving deep transformer with depth-scaled initialization and merged attention. In *EMNLP*.

A Additional Results

A.1 Implementation Details

All transformer models are selected based on their loss on validation set, while evaluated and reported on the test set. For En-De and En-Fr models, we used newstest2013 as validation set and newstest2014 as test set. For En-Zh models, we used newsdev2016 as validation set and newstest2017 as test set.

All three datasets follow the prepossessing steps from FairSeq[9], which uses Moses tokenizer[10], with a joint BPE of 40000 steps, while does not include lower-casing nor true-casing.

All models are evaluated with a beam size of 10. Before evaluating the BLEU score, we apply a postprocessing step, where En-De and En-Fr generations apply compound word splitting[11], and En-Zh generations apply Chinese word splitting (into Chinese characters). All generations are then evaluated with Moses *multi-bleu.perl* script[12] against the golden references.

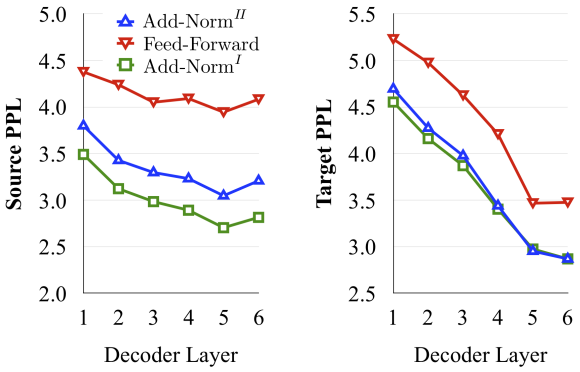
A.2 Transformer Big Results

We also compare the performance of the standard and simplified decoder under Transformer Big setting. Big models are trained on 4 NVIDIA V100 chips, where each is allocated with a batch size of 8,192 tokens. Other training schedules and hyper-parameters are the same as standard (Vaswani et al., 2017). Also, our Transformer Base models are all trained with full precision (FP32), while Big models are all trained with half precision (FP16) for faster training.

Transformer Big results are shown in Table. 6. We could observe a more severe BLEU score drop with a more significant speed boosting under Big setting. This is very intuitive, compared to Base setting, the simplified decoder drops more parameters, while still trained under the same schedule as standard, thus escalating the training discrepancy. Unfortunately due to the resource limitation, we could not afford hyper-parameter tuning for Transformer.

	Decoder	BLEU	#Train.	#Infer.
En-De	Standard	28.66	103.7K	74.3
	Simplified	28.20	125.2K	90.5
	△	*-0.46*	*+20.7%*	*+21.8%*
En-Zh	Standard	34.48	71.3K	30.5
	Simplified	34.35	82.6K	46.0
	△	*-0.13*	*+15.8%*	*+50.8%*
En-Fr	Standard	42.48	113.8K	65.7
	Simplified	42.19	138.1K	80.9
	△	*-0.29*	*+21.4%*	*+23.1%*

Table 6: Performance of the simplified Big decoder. "#Train" denotes the training speed (words per second) and "#Infer." denotes the inference speed (sentences per second).



(a) Source PPL (b) Target PPL

Figure 11: Illustration of the source and target information evolution within IFM on En-Zh.

A.3 Additional En-Zh and En-Fr Plots

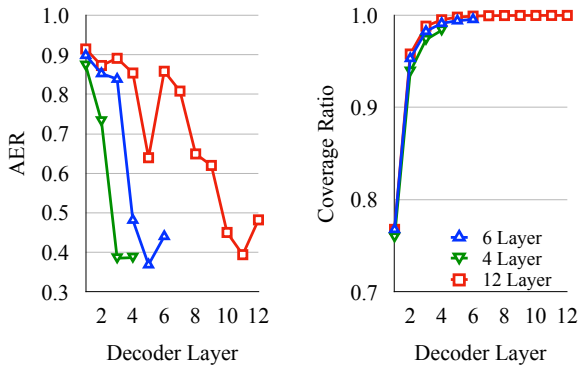
All experiments are conducted on three datasets (En-De, En-Zh and En-Fr), where we have similar findings. Due to space limits, we mainly demonstrate results on En-De task in our paper. In this section, we provide additional results on En-Zh and En-Fr if applicable.

[9] https://github.com/pytorch/fairseq/blob/master/examples/translation/prepare-wmt14en2de.sh

[10] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/mosestokenizer/tokenizer.py

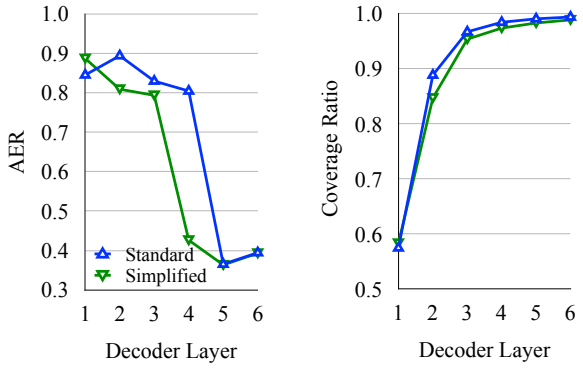
[11] https://gist.github.com/myleott/da0ea3ce8ee7582b034b9711698d5c16

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



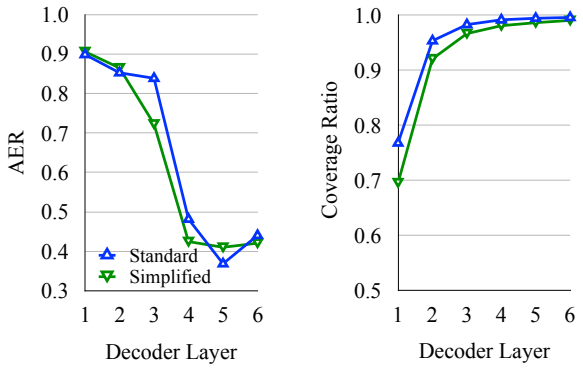
(a) Word Alignment (b) Cumulative Coverage

Figure 12: Effects of decoder depths on SEM behaviors on En-Zh.



(a) Word Alignment (b) Cumulative Coverage

Figure 13: Comparison between standard and simplified model on SEM behaviors on En-De.



(a) Word Alignment (b) Cumulative Coverage

Figure 14: Comparison between standard and simplified model on SEM behaviors on En-Zh.