Competition: Hindi to English Machine Translation System

Kushagra Pandey
20111028
{kushagrap20}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

In this report, we present an overview of our Hindi to English Machine Translation system developed as a part of the competition for CS779. In particular, we propose a Dual Attention mechanism and a Fused Attention Mechanism, developed as a part of the competition, and compare the proposed approaches with a traditional Transformer model on an internal validation set to demonstrate the utility of our approach. Our best model (Dual-Attention Transformer) achieved a METEOR score of 0.421, a BLEU score of 0.1428 on the final test set, and a rank of 4 on the final leaderboard.

1 Competition Result

Codalab Username: K_20111028
Final leaderboard rank on the test set: 4
METEOR Score wrt to the best rank: 0.421 (Best: 0.428)
BLEU Score wrt to the best rank: 0.1428 (Best: 0.1489)
Link to the CoLab/Kaggle notebook: https://colab.research.google.com/drive/1Y8CQOUBg1oCtz4q4q_yiUItmXy47VzNs?usp=sharing

2 Problem Description

In this report, we tackle the problem of Hindi to English Machine Translation. As the name suggests, the goal is to translate a given Hindi sentence into an English sentence. In particular, we adopt a Neural Machine Translation [1][2][3] approach to this problem, where we primarily make use of Self-attention [3] based Encoder-Decoder architectures to train a model to translate from Hindi to English sentences.

3 Data Analysis

We describe several aspects of the training data used for model development and test data used for final inference in this section.

3.1 Corpus Statistics Analysis

The raw training data provided for the competition consists of 102322 parallel Hindi-English sentences. We performed an initial analysis to get an estimate of the average sentence size of the training data. The mean sentence length for Hindi and English sentences in the training data were around 12.77 and 11.93 tokens, respectively. However, to get a complete picture of the length of Hindi and English sentences, we also visualized a histogram of sentence lengths. The visualization is shown in Figure 1. From the visualization, it is clear that most sentences in the parallel training corpora have sentence lengths well below 100. However, the maximum sentence length encountered for Hindi and English was 596 and 446 tokens, respectively. This analysis played an important part in choosing the maximum sequence length during model training.

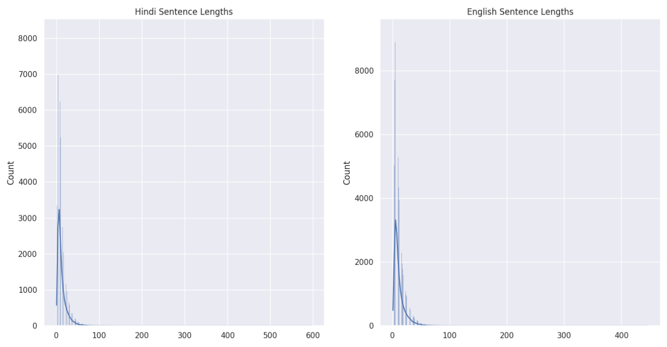


Figure 1: An illustration of the distribution of sentence lengths in the training corpus used in the competition. Most sentences in the corpus are less than 100 tokens in length.

3.2 Noise analysis

During the training phase, noise in the data can significantly affect the model's performance on downstream tasks like inference. In order to get an estimate of the amount of noise present in the data, we mainly performed three types of analysis, namely: detecting the presence of unwanted tokens, presence of contractions, and presence of accents (especially in English sentences.). The results of all these analyses for both training and test data are summarized below.

3.2.1 Presence of unwanted tokens

In order to get an estimate of noise in the form of unwanted tokens present in the sentences, we performed simple filtering for Hindi and English sentences. For Hindi data, we filtered out sentences containing tokens not belonging to the Hindi Unicode character range. Punctuation marks were excluded from the filtering process. Similarly, for English data, we filtered out sentences not containing alphabets, digits, or punctuations. Out of the 102322 sentences in the training data, 53960 Hindi sentences were noisy, while 43687 English sentences were noisy.

3.2.2 Presence of contractions

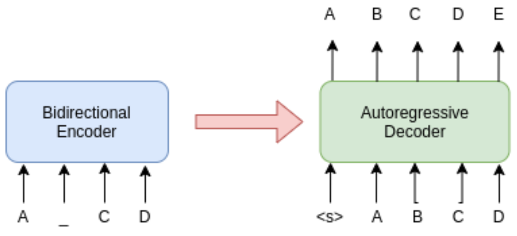
Contractions constitute common English phrases like "I've, we've, I'll" etc. Expanding these short phrases can help in reducing the vocabulary size during training. Hence, we performed rule-based contraction detection for English sentences in the corpus and found 29552 sentences containing contractions.

3.2.3 Presence of accents

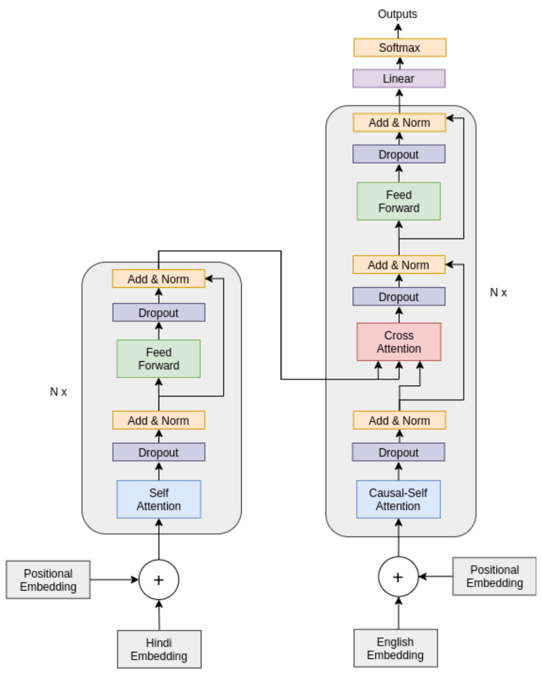
Accents are a part of dialect concerning local pronunciation. The presence of accents also adds noise to the data, leading to increased vocabulary size. We filtered out English sentences in the training data containing accents. We found 741 sentences in the training data containing some accented language. Hence, the number of such accented sentences is small compared to the size of the training corpus and can also be ignored during training.

4 Model Description

In this section, we present a brief overview of different models used during different phases of the competition.



(a)



(b)

Figure 2: a) Illustration of BART model schematics b) Illustration of the Standard Transformer network architecture. Nx denotes the number of stacks used.

4.1 Phase 1

During phase 1 of the competition, we primarily experimented with smaller variants of pre-trained BERT [4] models. More specifically, we experimented with three miniature BERT-variants [5], namely BERT-Mini, BERT-Small, and BERT-Medium. Using these smaller variants instead of using a full BERT model was made primarily due to two reasons. Firstly, we wanted to investigate the impact of training progressively larger models on the NMT task. Secondly, smaller variants would help with rapid prototyping given our restricted computational budget. The pre-trained models for these variants were obtained using the Huggingface transformers library [6].

During our internal validation testing, we noticed that an increase in model capacity led to increased model performance on the Machine translation task. BERT-Medium performed the best out of all the three variants on our internal cross-validation set. This observation was not surprising as BERT-Medium is the largest among the variants we tried.

4.2 Phase 2

During phase 2, we tried to explore Seq2Seq models pre-trained on different pretext tasks. The intuition behind this decision was that pre-trained Seq2Seq models might offer better performance than using a pre-trained BERT model, which was not initially pre-trained in a Seq2Seq setting. Therefore, we chose a pre-trained Bidirectional and Autoregressive Transformer (BART) [4] model for finetuning our NMT system for this phase. BART uses a Seq2Seq architecture, where the encoder is a bi-directional encoder as in BERT, and the decoder is an autoregressive transformer as proposed in GPT [5]. BART is trained during the pretraining phase by providing corrupted inputs to the encoder and minimizing the reconstruction (cross-entropy) loss between the decoder's output and the original input. [4] explore several choices for denoising the input like Token Masking, Token deletion and Sentence permutation. An overview of the BART model is as shown in Figure 2a For phase 2, we used a pre-trained BART model with six encoder and six decoder layers with a hidden embedding size of 768. The pre-trained models for these variants were also obtained using the Huggingface transformers library [6]

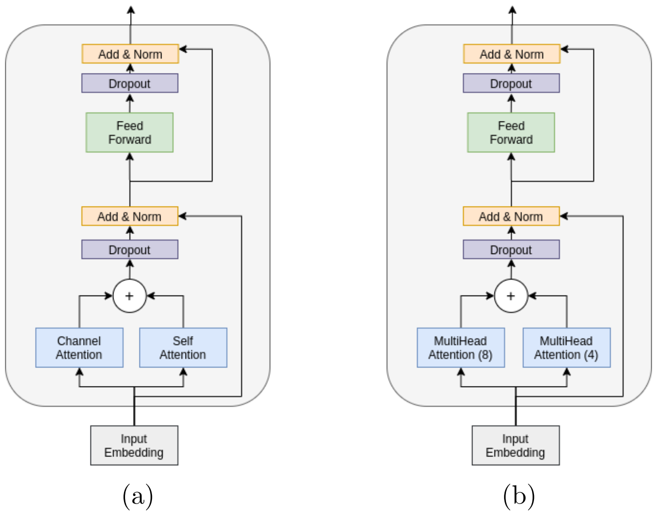


Figure 3: a) Illustration of the Dual-Attention Encoder Layer b) Illustration of the Fused-Attention Encoder Layer (fusion using 4 and 8 heads)

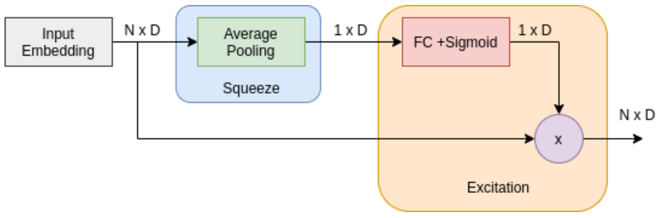


Figure 4: Illustration of the Squeeze-Excitation module used in computing Channel-wise Attention

4.3 Phase 3 and Beyond

In phase 3, we mostly experimented with different attention mechanisms in the standard Transformer architecture [3]. Figure 2b shows the standard Transformer model architecture. The architecture consists of an Encoder-Decoder architecture with N encoder stacks. The output of the last encoder stage is provided as an input to each of the N decoder stacks. We augmented this Encoder-Decoder model with two types of attention mechanisms which are described in more details as follows:

4.3.1 Dual-Attention Mechanism

The main idea behind a Dual-Attention formulation is to perform Channel-wise attention, which recalibrates the individual feature dimensions of the input embedding, in addition to performing Multi-Head Attention across word tokens. The outputs of both the MultiHead Attention and the Channel Attention modules are added and passed to subsequent layers. Figure 3a provides an overview of the Dual Attention Encoder layer.

We use the Squeeze-Excitation (SE) formulation [7] to perform Channel-wise Attention. The SE formulation consists of two main stages: Squeeze and Excitation. In the Squeeze stage, the input embeddings are average-pooled to aggregate spatial context. In the Excitation stage, the aggregated context is transformed through a linear layer, followed by a sigmoid activation to produce channel-wise attention weights. Each feature dimension of the input embeddings is then scaled by the resulting weights, thus producing channel-wise recalibrated feature embeddings. Figure 4 illustrates the methodology of the Squeeze-Excitation module.

To summarize, the Channel Attention module in the Dual-Attention Encoder layer architecture (Figure 3a) uses the SE module illustrated in Figure 4. A Dual-Attention Encoder Layer then replaces the standard Encoder Layer in the Transformer network.

4.3.2 Fused-Attention Mechanism

The main idea behind a Fused Attention formulation is to independently perform MultiHead Attention using different heads and fuse each attention module's outputs using simple addition. Figure 3b provides an overview of the approach. As shown in Figure 3b, the input embeddings are independently passed through two MultiHead modules containing 4 and 8 heads, respectively. The outputs of these modules are then fused (added) and passed to the next layer.

The primary motivation for the approach stems from the fact that the number of heads used in MultiHead Attention is mainly empirical. Combining information from multiple attention modules can help mitigate the need for making this choice. Secondly, using a large number of heads in the MultiHead Attention module can lead to reduced performance on downstream tasks [3]. Hence, in this approach, we can fuse information from a large number of heads without explicitly increasing the number of heads in a single MultiHead module, which can be another advantage of the approach.

4.3.3 Key learnings

We tried both types of attention mechanisms discussed previously and found that using the Encoder-only Dual-Attention Mechanism outperformed the Standard Transformer on our internal validation set. However, using the Dual-Attention framework in both the Encoder and the Decoder led to a slight decrease in performance. More surprisingly, using the Fused Attention Mechanism in the Encoder and the Decoder gave less performance than the Standard Transformer. Overall, we found that the Transformer model with a Dual-Attention mechanism in the Encoder gave the best results on our internal validation set and was thus used to make predictions for the final test set. More details on the model specification can be found in Section 5.2 and on results can be found in Section 6.2

4.4 Model Loss functions

We used the standard cross-entropy loss between the decoder output predictions and the ground-truth English sentences throughout all phases. The predictions at the padding positions were not used to update the model parameters during loss computation.

4.5 Inference

During phases 1 and 2, we used the Beam search-based (with five beams) decoding during inference. However, in our initial experiments, we found that the performance difference between Beam search and Greedy decoding was significantly less due to which we used only Greedy decoding during Phase 3 and beyond.

5 Experiments

5.1 Data Pre-processing

For the Hindi data, initial pre-processing was performed by normalizing Hindi sentences. We used the standard IndicNLP Normalizer for this task. We then removed all unwanted tokens (see Section 3.2.1) from the sentences to generate a clean Hindi corpus. We lower the text for English data, followed by contraction resolution and removal of accents and unwanted characters. It is worth noting that this kind of pre-processing was motivated by the Corpus Statistics analysis and Corpus Noise analysis presented in Section 3.2.

The next step is to learn a vocabulary on the clean Hindi and English sentences. For phases 1 and 2, we used the WordPiece [8] and ByteLevelBPE [9] tokenizers respectively. However, due to implementation issues, we resorted to a simple word-level tokenizer for both Hindi and English text in Phase 3. We trained separate tokenizers for Hindi and English text, which yielded vocabulary sizes of 40887 tokens and 33369 tokens, respectively. The unique tokens [SEP], [CLS], [PAD], and [UNK] denote text end, text beginning, padding, and unknown tokens, respectively.

For internal validation, we split the given training corpus into independent train and Val sets. We used a train-Val split of 95%-5% to generate a validation set of 5071 parallel sentences leaving the

Table 1: List of common hyperparameters for all models used during Phase 3

Hyperparameter	Value
N	6
embedding_dim	512
ff_size	2048
n_heads	8
Positional embedding	sinusoid
Dropout	0.1
Norm	LayerNorm [10]
Activation	ReLU

Table 2: Internal Validation Set results for Phase 3 models

Model	METEOR
Trans	0.485
DA-Trans	**0.487**
FA-Trans_4_8	0.458
Trans-L	**0.501**

remaining sentences for training the models. The sentences were batched together with a max_length of 64, and padding was applied to match the length of the longest sentence in the mini-batch.

5.2 Model Architectures

In this section, we discuss the model architecture specifications used during Phase 3. Refer to Sections 4.1 and 4.2 for details about model specification used in Phases 1 and 2, respectively. Table 1 lists the hyper-parameters for the Standard Transformer model used during Phase 3. Here, N stands for the number of encoder and decoder stacks, *embedding_dim* denotes the size of the word-embedding, ff_size denotes the size of the feedforward module in the encoder stack, and *n_heads* signifies the number of heads used in Multi-Head Attention in the encoder and the decoder. [3] was used as a reference for hyperparameter selection.

Table 2 lists the performance of different Phase 3 models on the internal validation set. Here, $Trans$ denotes a Standard transformer architecture (hyperparameters in 1), $Trans - L$ denotes a large transformer with an embedding_size of 1024, $DA - Trans$ denotes a standard Transformer using Dual-Attention mechanism only in the Encoder, and $FA - Trans\_4\_8$ denotes a standard Transformer architecture using Fused-Attention mechanism in the Encoder and the decoder with a combination of 4 and 8 heads in the Fused Attention Module.

As can be observed from the results, using a more significant embedding dimension results in better performance. Moreover, the $DA - Trans$ variant performs the best among other standard Transformer variants. From these results, we chose the large version of the $Trans$ and the $DA - Trans$ variant for the final test-set evaluation

5.3 Optimizer

We used the Adam [11] optimizer during training with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e - 08$. For varying the learning rate throughout training, we used a Poly-learning rate scheduler [12]. The learning rate for the t_{th} epoch is given by:

$$lr(t) = lr(0) * (1 - \frac{t}{num\_epochs})^{\alpha} \tag{1}$$

We used $\alpha = 0.9$ during training. The initial learning rate was set to 0.0001

5.4 Regularization

We apply Dropout [13] during different stages of the encoder and the decoder as shown in Figure 2b. We used a dropout rate of 0.1 for all dropout layers in all the variants.

Table 3: Dev set results for models in all Phases.

Model	Phase	BLEU	METEOR	Rank
BERT-Medium	1	0.0349	0.301	6
BART	2	**0.0830**	**0.301**	3
Std. Transformer	3	0.0613	0.253	2

Table 4: Test set results

Model	BLEU	METEOR
Trans-L	0.1409	0.412
DA Trans-L	**0.1428**	**0.421**

5.5 Training schedule

We trained our models in the Kaggle environment, which provides a single NVIDIA P100 GPU. Phase 1 and 2 models were fine-tuned for 20 epochs. Standard models in Phase 3 were trained for 20 epochs, and larger models (with 1024 embedding size) were trained for 25 epochs from scratch. For Phase 3 models, batch size was set to 32, resulting in about 3010 steps per epoch. Each step took about 0.2 seconds for standard models, while each step in the larger models took about 0.4 seconds. The last available checkpoint during training was utilized for inference.

6 Results

6.1 Dev set results

The result of different models used during different phases of the competition are shown in Table 3. One interesting aspect of the results in 3 is that, as expected, BART performed better than BERT-Medium in terms of the BLEU score but showed no improvement in the METEOR scores. Overall, the BART model performed the best among all the models in different phases. This can be expected because we used a pre-trained Seq2Seq model like BART to finetune our NMT model in Phase 2, which is more likely to give better performance than training a Standard Transformer model from scratch (in Phase 3). Moreover, we used Subword tokenizers [8][9] for tokenizing text in Phases 1 and 2. In contrast, we used a simple Word-level tokenizer in Phase 3, leading to more OOV words during test time. We hypothesize that using a subword tokenizer to train the Standard Transformer model would have given better results.

However, we should be slightly cautious in interpreting the dev-set results as the models were evaluated using different dev-sets in different phases of the competition.

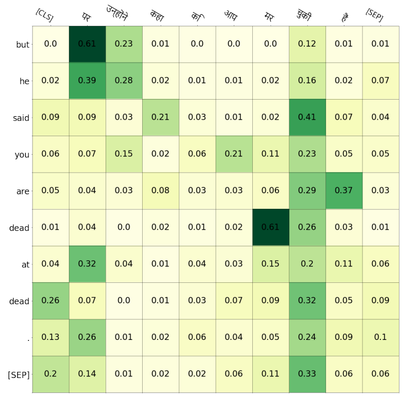
6.2 Test Set Results

Table 4 shows the test set results of different models. Phase-1 and Phase-2 models were not evaluated on the test-set given the logistical requirements of the competition. Also, due to the poor performance of the Fused-Attention mechanism on our internal validation set, it was not evaluated on the final test set. As shown in Table 4, the Dual-Attention mechanism-based transformer outperforms the standard transformer, both in terms of BLEU and METEOR scores. As hypothesized earlier, this result should be expected given that the Dual-Attention mechanism calibrates the features spatially and along the channel dimensions.

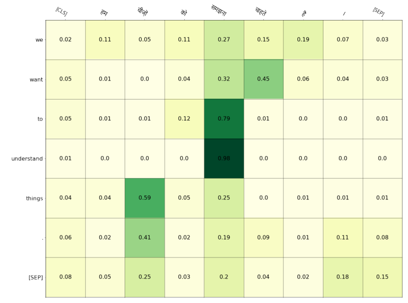
7 Error Analysis

In this section, we present a brief error analysis of our best-performing model (Dual-Attention based Large Transformer) on our internal validation set. Figures 5a and 5b show an attention map visualization between a Hindi and an English translation using greedy-decoding from the final layer of the decoder. In general, we would expect that the generated English words would most likely attend to their analogous Hindi words in the sentence. As shown in Figure 5a, the generated word *dead* places the most weight on the corresponding Hindi word. Similarly, the generated word *but* places the most weight on the starting Hindi word.

Figure 5b shows a similar trend where words like *understand* place a very high attention weight on the corresponding Hindi word. Hence, we can say that our model can understand the context and can



(a)



(b)

Figure 5: Cross-Layer Attention map visualizations between a Hindi sentence and it's generated counterpart. (Best viewed when zoomed-in)

capture statistical dependencies between words reliably. However, a closer look at our two examples reveals some problems with our model, which we state below. It is worth noting that we base our observations after analyzing many such translations, only two of which are shown here for brevity.

1. Our best-performing model could not resolve the ambiguity in the generation of the punctuation character ".". As can be noticed in Figure 5b, our model places more weight on other Hindi tokens than on the Poorna Virama ("|") character during "." generation.
2. We also observed token duplication in the generated text during inference. Figure 5a shows one such instance, where the phrase including the word "dead" is generated twice. However, using n-gram repeat penalties as proposed in [14] and [15] can alleviate this problem.
3. We also found that our model faced problems generating translations for longer Hindi text, indicating that our model might be biased towards translating shorter sentences. Another reason for this problem can be our model's inability to capture long-term dependencies. However, we believe that using more sophisticated attention-mechanisms as proposed in [16] can alleviate this problem.

8 Conclusion

In this report, we presented an overview of the different approaches we used to train an NMT model for Hindi to English Machine Translation. More notably, we presented a Dual-Attention formulation and a Fused-Attention formulation as two variants of the standard Transformer architecture and showed that the Dual-Attention variant outperforms the standard Transformer architecture on the final competition test set. However, the inferior performance of the Fused-Attn formulation was surprising. For future work, we plan to investigate more into the error analysis for this formulation.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[4] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019.

[5] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[6] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online), pp. 38–45, Association for Computational Linguistics, Oct. 2020.

[7] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2019.

[8] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, 2012.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2018.

[10] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[12] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," 2017.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, p. 1929–1958, Jan. 2014.

[14] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017.

[15] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," 2017.

[16] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020.