Document-Level Machine Translation with Hierarchical Attention

A Project Report

Presented to

Dr. Chris Pollett
Dr. William Andreopoulos
Dr. Thomas Austin
Department of Computer Science
San Jose State University

In Partial Fulfillment
Of the Requirements of the Class
Spring 2023: CS 298

By

Yu-Tang Shen

March 2023

ABSTRACT

Machine translation (MT) aims to translate texts with minimal human involvement, and the utilization of machine learning methods is pivotal to its success. Sentence-level and paragraph-level translations were well-explored in the past decade, such as the Transformer and its variations, but less research was done on the document level. From reading a piece of news in a different language to trying to understand foreign research, document-level translation can be helpful.

This project utilizes a hierarchical attention (HAN) mechanism to abstract context information making document-level translation possible. It further utilizes the Big Bird attention mask in the hope of reducing memory usage. The results from the experiments showed that the HAN models produced readable translations and had an average BLEU score of 0.75 (0.67 for full attention HAN, and 0.82 for Big Bird attention), whereas the Transformer model failed to comprehend the large input and had a score of 0.22 on the same dataset.

Keywords – Hierarchical attention model (HAN), machine translation (MT), neural machine translation (NMT)

TABLE OF CONTENTS

TABLE OF FIGURES

I. INTRODUCTION

Machine translation (MT) has broken the language barriers between people using different languages [1]. Applications that utilize MT such as Google Translate have become essential tools for people comprehending content in foreign languages. Yang, Wang, and Chu [2] and Stahlberg [3] demonstrated that neural machine translation (NMT), especially attentional models, can produce satisfying translations. The attention model obtained better results with a faster training process compared to other NMT models [4] [5], and it had been deployed in commercial products like Google Translate. In experiments before this project, it was confirmed that the attentional models outperformed prior state-of-the-art MT techniques such as statistical machine translation (SMT). In contrast to the successful results on sentence-level and paragraph-level translation, much fewer experiments and research had been done on document-level translation. Paragraph-level translation can sometimes produce confusing results, such as wrong pronoun interpretations and inconsistent words, making the translated text hard to comprehend. Therefore, an NMT model that produced comprehensive document translations is wanted.

Contexts play an important role in MT since one word can have different meanings under different contexts, for example, the word “date” refers to two unrelated ideas in “I bought some dates in the grocery store” and “I have a date tomorrow”. MT has evolved from rule-based machine translation (RBMT) to statistical machine translation (SMT) and finally to NMT for a better interpretation of the context. RBMT ties tokens

(vocabularies or phrases) to a single translation, so it can fail to produce correct translations under different contexts. SMT determines the translation based on statistics of the original language, for instance, if 90% of the time the word “date” meant appointment, an SMT model would interpret “date” as appointment 90% of the time, which multiple incorrect translations could appear in a paragraph describing the fruit. NMT models not only translate each token based on their lexical meaning but extract the context from the relationships between tokens to refine the outputs. NMT models can better identify context, and consequently, the results show better readability and coherence [4] [5].

However, as Sun et. al. [6] stated, many NMT models translate input sequences individually, neglecting long-range context in documents leading to incorrect translations such as inconsistency in words and wrong interpretation of pronouns. Thus, there was a need for a new NMT model that was aware of long-range contexts.

This project aims to utilize hierarchical attention (HAN) to provide context information during the translation process, where two variations of HAN are presented. The first design adopts the attention mechanism illustrated by Vaswani et. al. [4], and in spite of it failing to properly translate digits in the initial implementation, it eventually generates adequate Chinese translation from English documents. The second implementation utilizes the Big Bird attention mask aiming to reduce memory usage. The model successfully translated the documents, but the memory usage was not reduced. It is speculated that the built-in masking function doesn’t optimize toward the Big Bird attention pattern.

This project report is organized as follows: Section II first introduces the fundamental machine translation technologies to highlight the bottlenecks each method faced. The proposed designs to overcome the inability to resolve long-range context are presented in Section III, including a hierarchical attention model with a full attention mechanism and another one with the Big Bird [7] attention mechanism. The implementations of those designs are shown in Section IV. Section V gives the result of both designs under English to Chinese and Chinese to English translations. Section VI concludes the project, and Section VII lists future improvements that can be done.

II. FUNDAMENTAL MACHINE TRANSLATION TECHNIQUES

1. Machine translation types

All machine translation technologies can be categorized into the following three types or mixtures of them, direct translation, indirect translation, and transfer translation. These approaches describe how source languages (SL) are mapped to target languages (TL).

Direct translation maps SL directly to TL, which is a succinct way to describe the relationships, but the number of relationship sets grows exponentially: for translation between N languages, $N(N-1)$ sets of relationships are required. For example, bidirectional translations between Chinese (*zh*), English (*en*), Spanish (*es*), and German (*de*) require 12 sets of rules: *zh-en, zh-es, zh-de*, *en-es*, *en-de*, *es-de*, and the other 6 sets of complementary rules (*en-zh*, *es-zh*, etc.)

Indirect translation adds an additional interlingua (IL) layer between two languages so that fewer sets of rules are required for multi-language MT: SL is first mapped to IL, which preserved the semantic information, and TL is generated from IL. With this approach, merely $2N$ sets of rules are required for translation between N languages. To translate the four languages listed in the previous paragraph, 8 rules are required: *zh-IL, en-IL, es-IL, de-IL, IL-zh, IL-en, IL-es*, and *IL-de*, where the interlingua language *IL* describes the context information.

Indirect translation requires the interlingua to abstract context information in multiple languages, and it can be difficult for one interlingua to accomplish such information extraction. Transfer translation attaches an extra layer of IL from the indirect translation schema, so the two layers of IL act as abstractions of SL and TL respectively. This approach resolved the challenge of synthesizing different TL with the same piece of IL.

2. Rule-based Machine Translation

RBMT uses a set of rules to perform translation, and the difference between swapping every word in the SL into the TL was more complex rules, such as re-ordering, could be specified in RBMT [8]. For instance, a rule to bring postnominal adjectives before nouns can be specified for English to Chinese translation as follows.

He is the person with a British accent.	他是那個有英國口音的人
Mary is the lady in a white dress.	瑪莉是那位穿白裙子的女士

The yellow highlighted texts (postnominal adjectives) and the green highlighted nouns are reordered.

The advantage of RBMT is the results are deterministic and predictable: once the rules were set, the outputs are determined. This makes the rules calibration easier: one can trace the rules that produce unexpected results and make changes accordingly. Unfortunately, such determinism is also its weakness: the only way to produce flexible and correct translations for the same word under different contexts is to specify all the rules under all situations, which is impractical. Moreover, rule calibration is labor-intensive and requires a strong understanding of SL and TL.

3. Statistical Machine Translation

SMT [9] provides flexibility to interpret words into different translations and reduces the workload of listing rules. The core value of SMT is to select the best candidate among all the possible translations based on known statistics, including the frequency of each token in SL, the popularity of each candidate in TL, the probability of each candidate following the current output, etc. A greedy SMT algorithm can be abstracted as follows:

```
1 translate(S)
2  result = φ
3  for s in S
4     candidates = getCandidate(s)
5     bestCandidate, bestScore = φ, -1
6     for candidate in candidates
7        if(probability(s, candidate) > bestScore)
8           bestCandidate = candidate
9           bestScore = probability(s, candidate)
10       end if
11    end for
12    result = result + bestCandidate
13 end for
14 return result
```

The `getCandidate` function on line 4 retrieves all the possible translations for the source language vocabulary `s`, and the `probability` functions on lines 7 and 9 return the probability of the first argument being translated into the second argument. The `translate` function iterates through all the vocabulary `s` in the SL sentence `S` to get the most probable TL word and joins the TL words into sentences.

Although SMT improves the translation quality, SMT falls short of translating composite terms, especially those requiring reordering the output.

4. Neural Machine Translation

Sequence-to-sequence (seq2seq) models improve the capability of deciphering composite terms to produce correct translations. The term seq2seq refers to models that take an input sequence of arbitrary length to produce an output sequence of unspecified sizes, such as recurrent neural network (RNN) [10], long short-term memory (LSTM) [10], and attention model [4].

RNN mitigates the drawbacks of RBMT and SMT. Neco and Forcada [11] tackled MT with an RNN, where no assumptions on the output text were made and each bit of the original text was translated with context information attained from previous elements in the sequence. By utilizing RNNs, human involvement is reduced since RNNs can identify the context to produce proper translations. Nonetheless, RNNs suffer from the vanishing gradient problem [12], which occurs when the associated terms are too separated.

Hochreiter [12] adopted LSTM to alleviate the vanishing gradient problem in RNN. LSTM memorizes context information in a separate space such that necessary context distant from the current bit can be included to determine the fittest translation. Vathsala and Holi [13] experimented on one million Twitter posts with LSTM, and the results proved LSTM had outperformed SMT.

Bahdanau, Cho, and Bengio [14] added the attention mechanism to an RNN, mimicking human translators: cutting a long sentence into smaller fragments and processing each of them. The term “attention” illustrates the correlation between two

words in the input sequence; attentions are computed for each term in the input sequence such that the model translates based on the current word and the context information from words having strong correlations to the current word.

Vaswani et. al. [4] claimed “attention is all you need” and introduced the Transformer model. This model is constructed with attention mechanisms but without recurrent layers as shown in Figure 1. Further details will be presented in Section III.2.

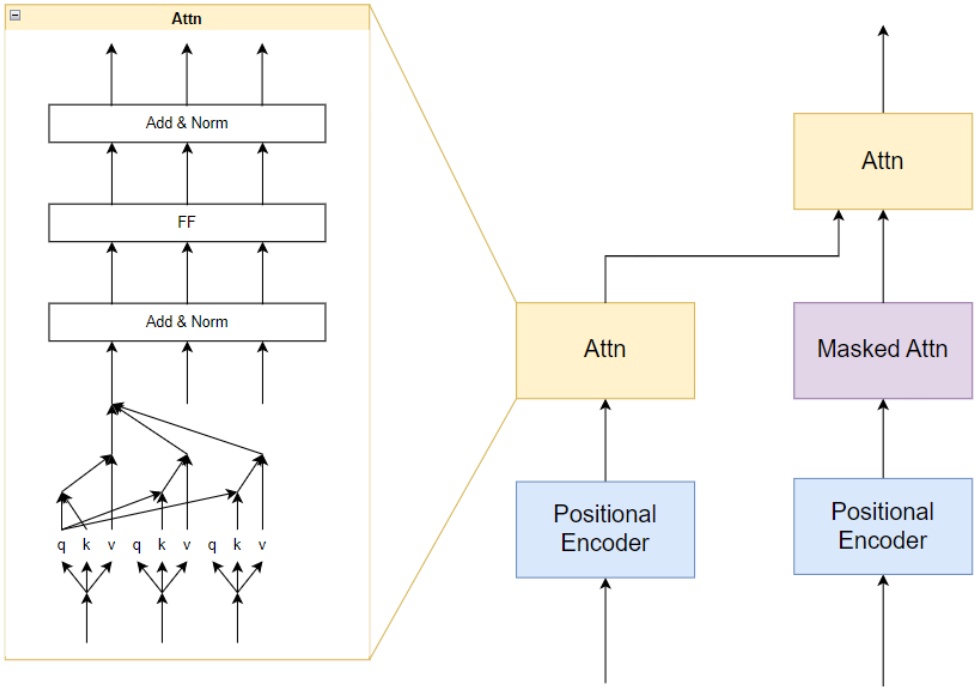


Figure 1 The Transformer

The Transformer model thoroughly considers context information to enhance translation qualities, and it has a more comprehensive interpretation of the underlying contexts than RNN and LSTM models [4]. It queries the tokens in front and behind the current token, the number of tokens it queries is also referred to as the window size. The

window size w is typically limited to reduce memory usage: for each token, w number of attention scores need to be computed, leading to an overall $O(w^2)$ memory usage. Current attention-based MT models commonly bound the input sequence length to around 1,000 words. Although it is adequate for paragraph-level translation, models that can handle longer input texts, such as documents, are needed.

III. Hierarchical Attention Model Design

A hierarchical attention (HAN) model including two variations was designed and implemented in this project. As shown in Figure 2, there are two additional kinds of layers in this design compared to the Transformer: the autoencoder layer and the HAN layer. The autoencoder layer summarizes the sentences in documents, and the HAN layer takes the sentence summaries and the input tokens to derive the appropriate output.

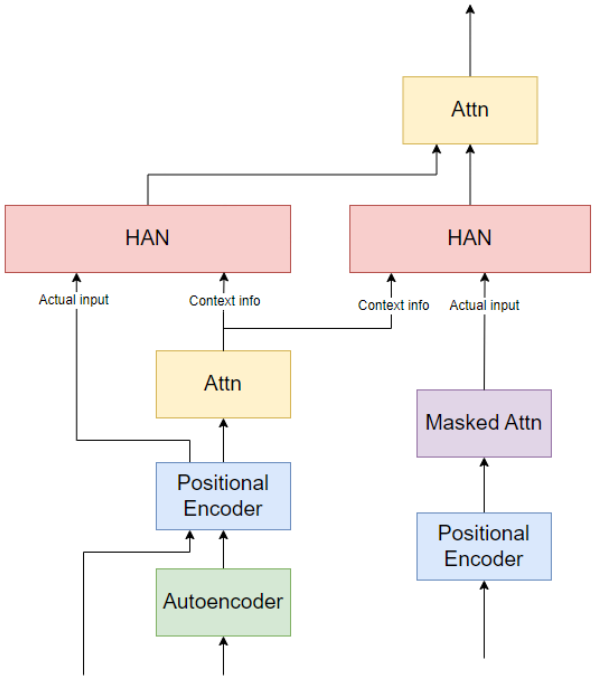


Figure 2 HAN model design

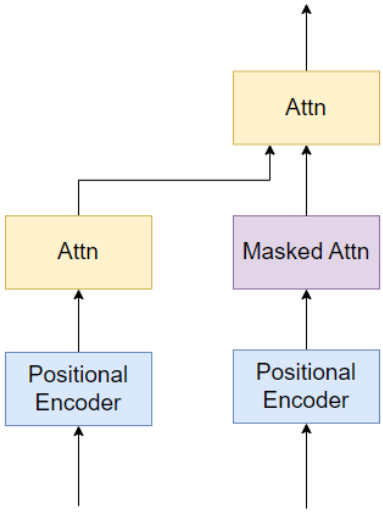


Figure 3 Abstracted design for the Transformer

1. *Autoencoder Layer Design*

The autoencoder layer was designed to summarize the sentences in documents. To precisely segment each sentence, in other words, to separate sentences with punctuations,

is inefficient because the model would have to accommodate different sentence lengths. Instead of accurately slicing sentences and obtaining the summary from them, a one-dimensional convolutional layer was used. The convolutional layer was set to have a filter size of S and a stride of $S/2$. Although this slices some sentences incorrectly, the overlap should retain necessary summary information. The integer S is a hyperparameter that corresponds to the mean sentence length of the documents.

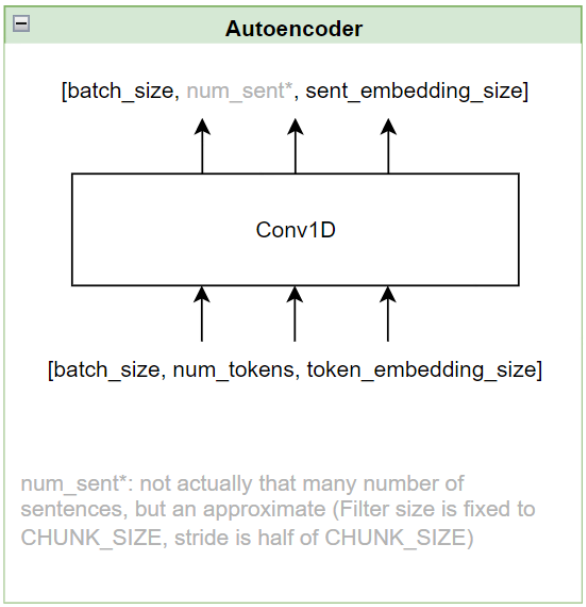


Figure 4 Autoencoder layer design

2. *Attention Layer Design*

This project implements the attention layers as described by Vaswani et. al. [4]. *Query*, *key*, and *value* are extracted from each token in the following way:

Query: $q_i = W_q a_i$

Key: $k_i = W_k a_i$

Value: $v_i = W_v a_i$

where W are trainable weights for different functions and a are inputs tokens to be parsed

After getting the *Q*, *K*, and *V* matrices (where the *i*-th item in each matrix corresponds to q_i, k_i, and v_i), the attention scores are computed as $Attention(Q, K, V) = QK^T V$. The attention scores will then be summed with the original activation a_is and the result will be layer normalized [15]. The next step is to feed the outputs into a feed-forward network and then add the layer-normalized results from the previous step and layer-normalize the results again.

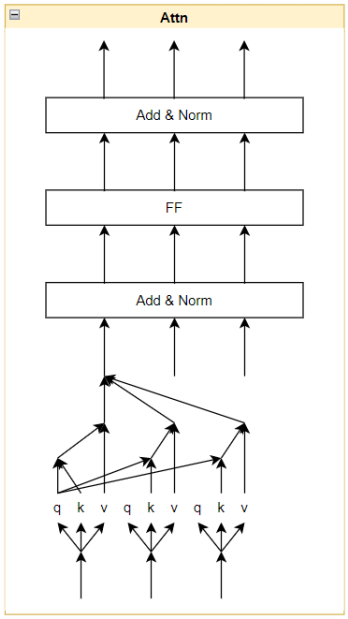


Figure 5 Attention layer design

3. *HAN Layer Design*

With the autoencoder layer summarizing the sentences in documents, the goal of a hierarchical attention layer is to extract information based on each token with additional summarized contexts. As shown in Figure 6, the layer performs self-attention to the tokens themselves and to other tokens to obtain the attention scores; additionally, it adds the attention score from querying to the context information, i.e., sentence summaries, to the outcomes.

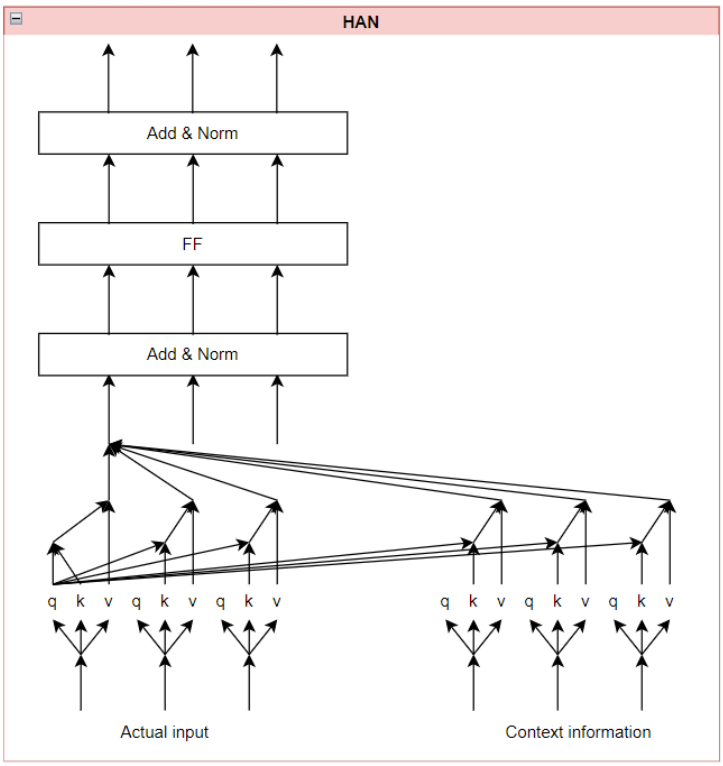


Figure 6 HAN layer design

The difference in how the attention scores are calculated between the Transformer [4] and the HAN layer in this report is shown in Table 1, where Q, K, V corresponds to

the *query*, *key*, and *value* matrices and those subscripted with *s* denotes the sentence summary matrices obtained from the autoencoder layer.

The attention layer in the Transformer	HAN layer in this report
$Attention(Q, K, V) = QK^T V$	$Attention(Q, K, V, Q_s, K_s, V_s) = QK^T V + Q_s {K_s}^T V_s$

Table 1 Difference between the Transformer and the HAN layer in this report

From Figure 2, one can see that the information extracted from the autoencoder layer is not directly passed into the HAN layer and the tokens go through a positional encoding layer before they enter the HAN layer. Since the attention layer doesn't perceive each token's position (there are no recurrent nodes nor convolution nodes), Vaswani et. al. [4] showed that adding positional information to each token in the sequence helped the attention layers to recognize the relative order of the tokens. Therefore, positional encodings are added to the input sequences.

4. Big Bird HAN Layer Design

The Big Bird attention model [7] shows that without calculating all the attention scores between all the input nodes, the attention model can still obtain satisfying results. In this project, the Big Bird attention pattern was tested for its impact on translation quality.

The Big Bird attention pattern, as shown in Figure 7, combined three patterns to construct the attention mask. In the Transformer [4] model, each input token queries all the tokens in the input sequence to obtain the final attention score. The Big Bird model

merges random attention, window attention, and global attention for its attention mask. The random attention pattern selects arbitrary two tokens in the input sequence to perform attention. The window attention chooses the adjacent tokens to calculate attention scores. The global attention attends to each token with the first two tokens in the input sequence.

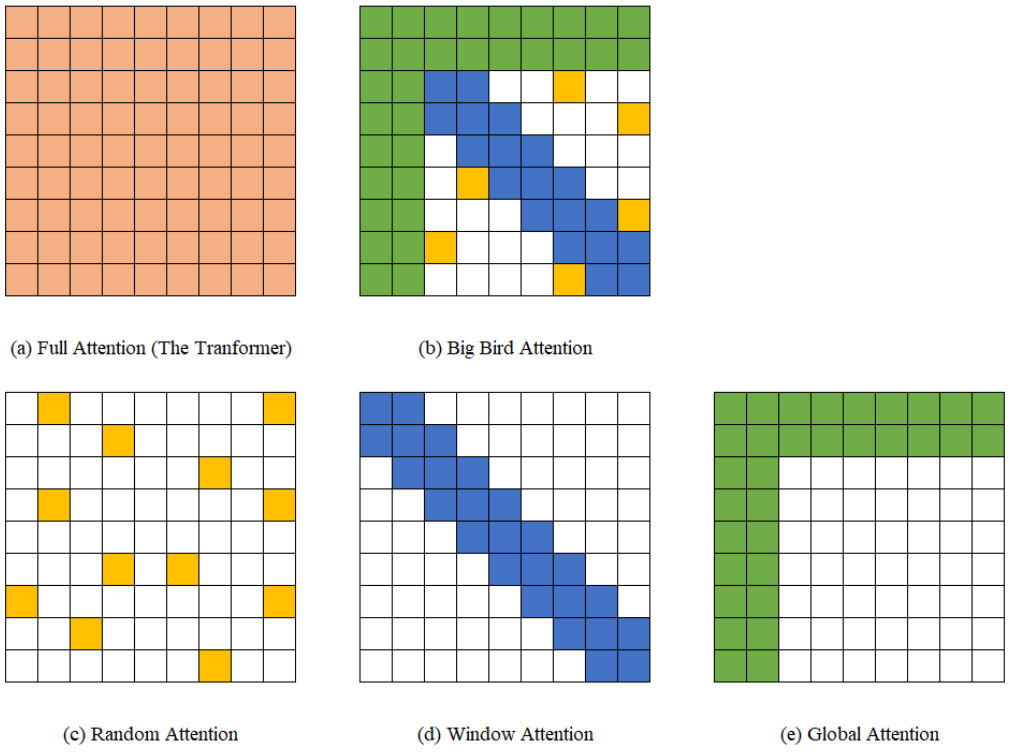


Figure 7 Big Bird attention pattern

IV. Hierarchical Attention Model Implementation

This project implemented the design shown in Figure 2 with TensorFlow [16] and Keras [17] libraries in Python.

1. Dataset and Preprocessing

This project used the News Commentary dataset in OPUS [18], word vectors in Fasttext [19] library, word tokenization from spaCy [20] library, and SciPy [21] library for deciphering output word vectors.

The data pipeline is shown in Figure 8: the raw data, i.e., human-readable news, was passed into a spaCy tokenizer; the Fasttext word vector object returned the word vectors for each token; the HAN model then took a sequence of word vectors and outputs another sequence of word vectors; finally, a SciPy k-d tree looked for the closest word for each word vector in the output sequence.

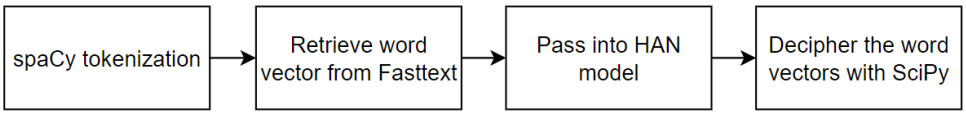


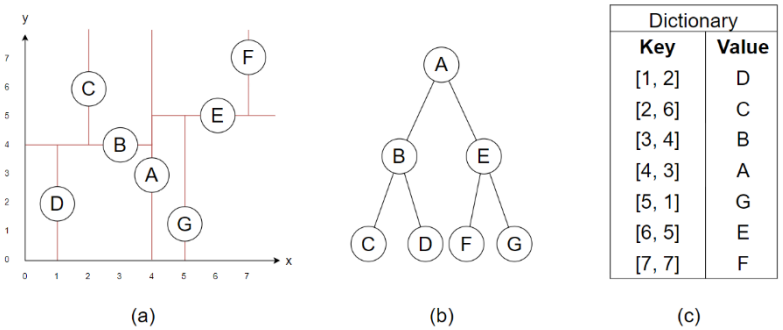
Figure 8 Data pipeline

In most of the MT projects, the word vector can be optimized during the model training stage to align with the training dataset, which can produce better results assuming the application environment is analogous to the training dataset. However, due to hardware restrictions, this project fed the HAN model with constant word vectors,

meaning that the word vectors remained the same over the whole training process. In the chosen dataset, there are 60,000 unique English tokens and 113,000 unique Chinese tokens, and the vocabulary space for both languages required too much memory for the hardware that this project was conducted.

During the preprocessing stage, all documents were transformed into $N \times 300$-sized matrices, where N is the number of tokens in each document and 300 is the default word vector dimension in Fasttext [19].

The model needs to decode the word vectors to obtain human-readable texts, so an algorithm to search for the closest word vector to the predicted ones is needed. A k-dimensional tree (k-d tree) is a binary tree that bisects k-dimensional vectors on each dimension as shown in Figure 9(a) and (b) such that searching for a vector has a time complexity of $O(log\ n)$ [22]. In this project, a k-d tree is used to store the word vectors, and a dictionary holds the vectors as keys and the words as the values as shown in Figure 9(c). To decode a word vector, it searches along the tree until a leaf node is reached or a non-leaf node matches the predicted vector, and it then retrieves the corresponding word from the dictionary.



Dictionary	
Key	**Value**
[1, 2]	D
[2, 6]	C
[3, 4]	B
[4, 3]	A
[5, 1]	G
[6, 5]	E
[7, 7]	F

Figure 9 Sample k-d tree (k=2) and dictionary

a. The digit issue

During the experiment stage, the model translated most words correctly but the digits, which will be presented in Section V.1. In order to overcome this issue, the numbers were split into separate tokens where each token contained exactly one digit.

2. Model Implementation

As described in Section III, there are three proprietary functions in this project: the autoencoder layer, the HAN layer, and the Big Bird attention mask. In addition to the three functions, a new loss function was also wanted.

a. Autoencoder layer

The autoencoder layer summarizes the input document on a sentence level, and the output should remain the same size on the second dimension such that it matches the second dimension of the input sequence. Therefore, a 1-dimensional convolutional layer was deployed. The pseudo-code for the autoencoder layer can be described as follows.

```
1 class Autoencoder():
2     def __init__(CHUNK_SIZE):
3         self.cnn = Conv1D(kernel_size=CHUNK_SIZE,
4                           strides=CHUNK_SIZE // 2)
5     def call(x):
6         x = self.cnn(x)
7         return x
```

b. HAN layer

The HAN layer implementation is similar to the Transformer attention layer except for adding the sentence summary information to the attention scores. The pseudo-code for a Transformer attention layer is listed in the following.

```
 1 class Attention():
 2    def __init__():
 3        self.mha = MultiHeadAttention()
 4        self.layernorm = LayerNormalization()
 5        self.add = Add()
 6    def call(x):
 7        attn_output = self.mha(query=x, value=x,key=x)
 8        x = self.add([x, attn_output])
 9        x = self.layernorm(x)
10        return x
```

The HAN layer can be described as follows. The difference is an additional attention layer, `HAN.context_attn`, used to calculate $Q_s K_s^T V_s$ listed in Section III.3. On line 9, the `HAN.context_attn` object uses the input token `x` to query the context information `context` and multiply it with the context information `context`. On line 13, the attention scores, including those performed on the tokens themselves, `attn_output`, and those attending to context information, `context_output`, are added to the original activation `x`.

```
 1 class HAN():
 2    def __init__():
 3        self.mha = MultiHeadAttention()
 4        self.context_attn = MultiHeadAttention()
 5        self.layernorm = LayerNormalization()
 6        self.add = Add()
 7    def call(x, context):
 8        attn_output = self.mha(query=x, value=x,key=x)
 9        context_output = self.context_attn(
10                                query=x,
11                                value=context,
```

```
12                                      key=context)
13        x = self.add([x, attn_output, context_output])
14        x = self.layernorm(x)
15        return x
```

c. Big Bird attention mask

This project utilized the TensorFlow [16] built-in functions to boost the mask generation efficiency during training, as the function within the library can take advantage of its graph optimization speedup.

The following pseudo-code describes the mask-generating process. The function takes the `shape` of the needed mask and the percentage of random attention wanted. The function generates a random [0, 1] matrix on line 2, fills the diagonal for window attention from lines 3 to 8, and sets the global attention on lines 9 and 10.

```
 1 def generate_mask(shape=[-1, -1], random_ratio=0.2):
 2    mask = random(shape = shape) < random_ratio
 3    for i in [0, shape[0]-1]:
 4        for j in [0, shape[1]-1]:
 5            mask[i][j]   = True
 6            mask[i+1][j] = True
 7            mask[i][j+1] = True
 8    mask[-1][-1] = True
 9    mask[0:2][:] = True
10    mask[:][0:2] = True
11    return mask
```

d. Loss and accuracy

As the HAN model doesn't output a sequence of tokens and instead generates a sequence of word vectors, a new loss function is required. Mean

squared error is selected as the loss function; because the final k-d tree deciphering process searches for the closest word vector to the output, the goal is to minimize the distance between the output word vector and the correct word vector.

Given the first word vector from HAN output is y and the first item from the ground truth is v; the distance between the first predicted word and the first word in the ground truth is $\sqrt{(y-v)^2}$. The mean squared error between y and v is $\frac{1}{\dim(y)}\sum_{i=0}^{\dim(y)}(y_i - v_i)^2$, where y_i and v_i are the i-th dimension of y and v. Since $\sqrt{(y-v)^2} \propto \frac{1}{\dim(y)}\sum_{i=0}^{\dim(y)}(y_i - v_i)^2$, it is reasonable to adopt mean squared error as the loss function.

Accuracy describes the fraction that a model predicts correctly, and it gives the developers an idea of whether a model is improving or not. In this project, the accuracy metric is defined as follows. The function takes the predicted sequence of word vectors, the corresponding truth, and a threshold. On line 2, the function performs an element-wise subtraction and checks if the absolute value of the difference is less than the threshold. The function then retrieves the number of elements in the word vectors on line 3 and eventually returns the percentage of matching elements in all word vectors.

```
1 def accuracy(pred, truth, threshold):
2    match = abs(pred-truth) < threshold
3    num_val = truth.shape[0] * truth.shape[1]
4    return float(sum(match)) / float(num_val)
```

V. HIERARCHICAL ATTENTION MODEL EXPERIMENTS

1. Training configurations

a. Hardware and software

The models in this project were trained on Google Colaboratory PRO[1] with Nvidia A100 40GB GPU, and its default image with Tensorflow version 2.12.0.

b. Parameter configuration

The hyperparameters are configured as follows.

Autoencoder kernel size	16
Big Bird random attention ratio	0.3
Number of training epochs	40
Accuracy element difference threshold	0.1
Attention layer number of heads	8
Number of attention layers	4
Dropout rate	0.1

The autoencoder kernel size was set to 16 since the average length of sentences in the dataset is around 16. The number of heads for attention layers represents how many independent sets of attention scores would be computed.

[1] https://research.google.com/colaboratory/

2. *Results*

a. English to Chinese HAN translation

In the initial experiment, the numbers were not split into separate digits and the word vector dimension was reduced to 100. The red highlighted numbers in Figure 10 were translated incorrectly, and a stuttering effect shown in yellow highlighted text in Figure 10 also occurs throughout the prediction. The translation in Figure 10 had a BLEU [23] score of 0.44, which can be considered an understandable translation.

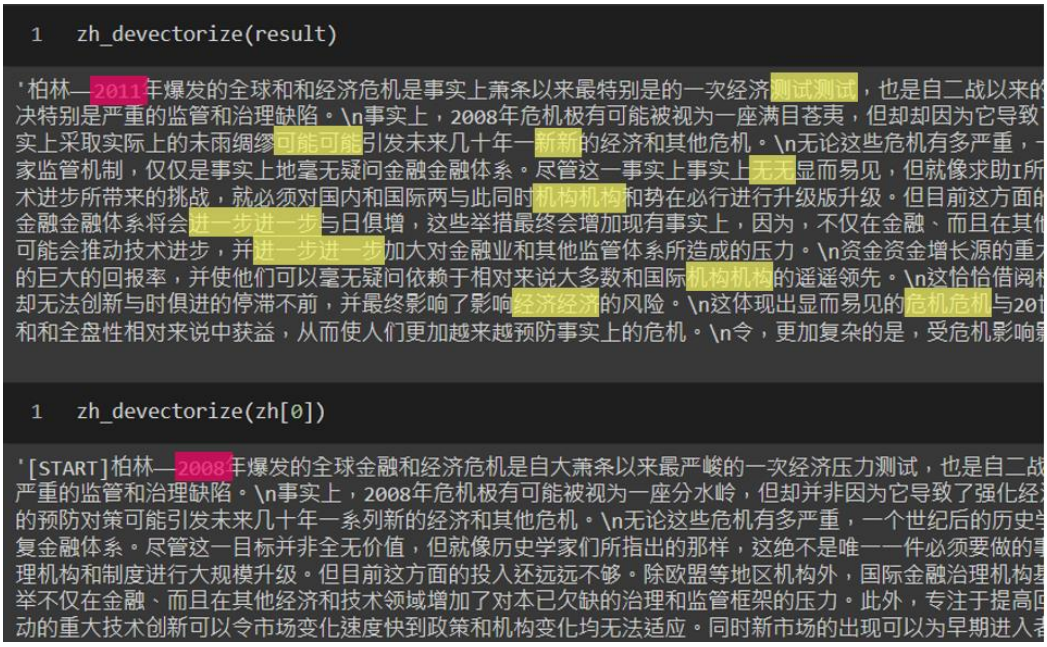


Figure 10 Translation result (wrong digits highlighted in red and stuttering effect highlighted in yellow)

The model reached 80% accuracy after 100 epochs in this configuration as shown in Figure 11. Splitting the digits and unfolding the word vector dimension

were experimented with in the following stages aiming to improve the translation quality.

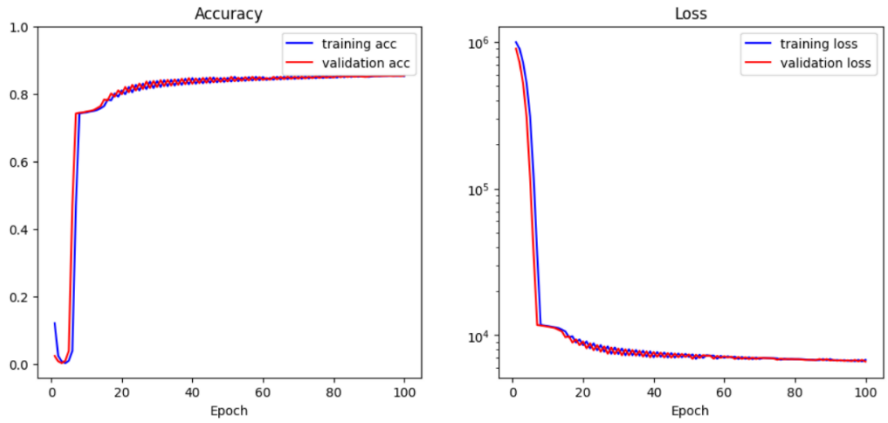


Figure 11 Accuracy and loss from the initial experiment

By splitting the numbers into a series of single digits, the model translated the numbers correctly. By unfolding the word vector dimensions to 300, the stuttering effect was greatly reduced and the sample translation in Figure 12 obtained a BLEU [23] score of 0.9.

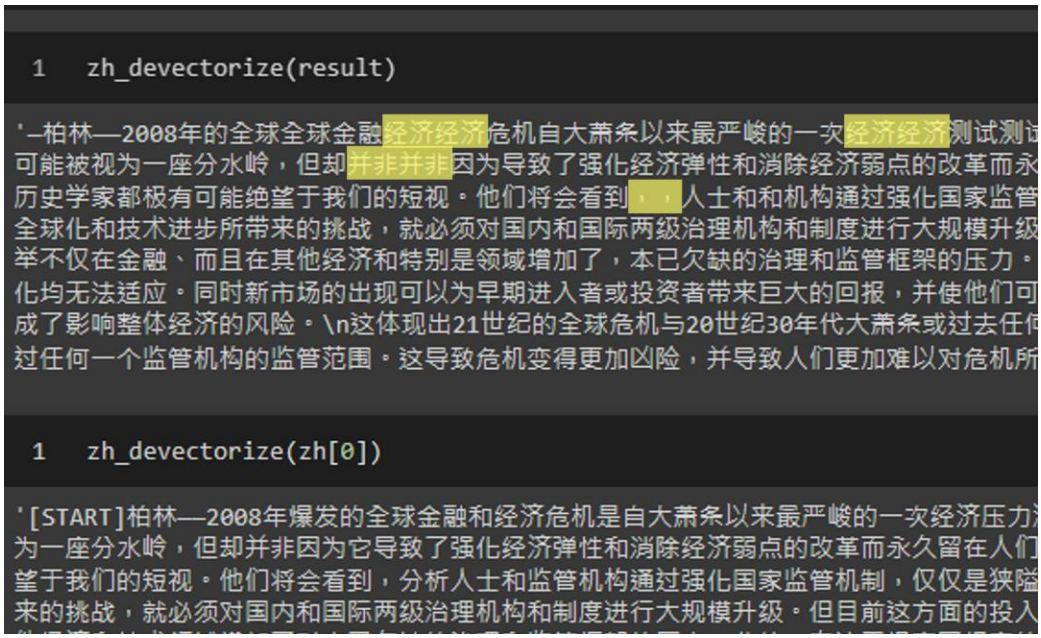


Figure 12 HAN result with split digits and unfolded word vectors

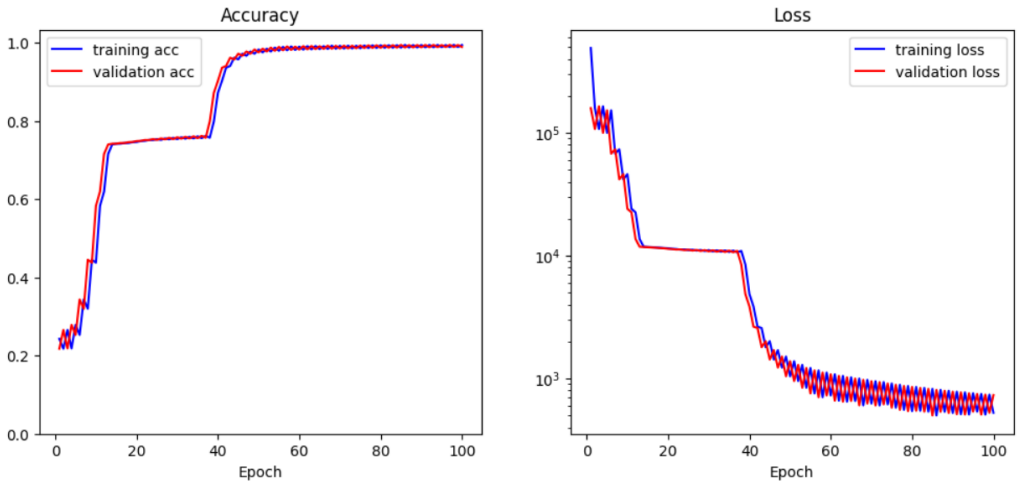


Figure 13 HAN accuracy and loss with split digits and unfolded word vectors

b. English to Chinese Big Bird HAN translation

As Manzil et. al. [7] presented, the Big Bird mask should consume less RAM resources and have a better efficiency to get an equivalent or even better translation quality.

Since the configuration listed in Section V.1.b. consumed all the RAM on the machine and there was no documentation pointing out if there was memory swapping in the background for Colaboratory notebooks, the model complexity was reduced to better observe the RAM usage. As shown in Figure 14, the RAM usage was identical and the training time difference was negligible. The entire attention mechanism was rebuilt in the original Big Bird [7] implementation, but this project adopted the built-in attention mask in the Keras [17] library, and the lack of optimization toward the Big Bird attention pattern could contribute to the absence of an efficiency boost in this experiment.

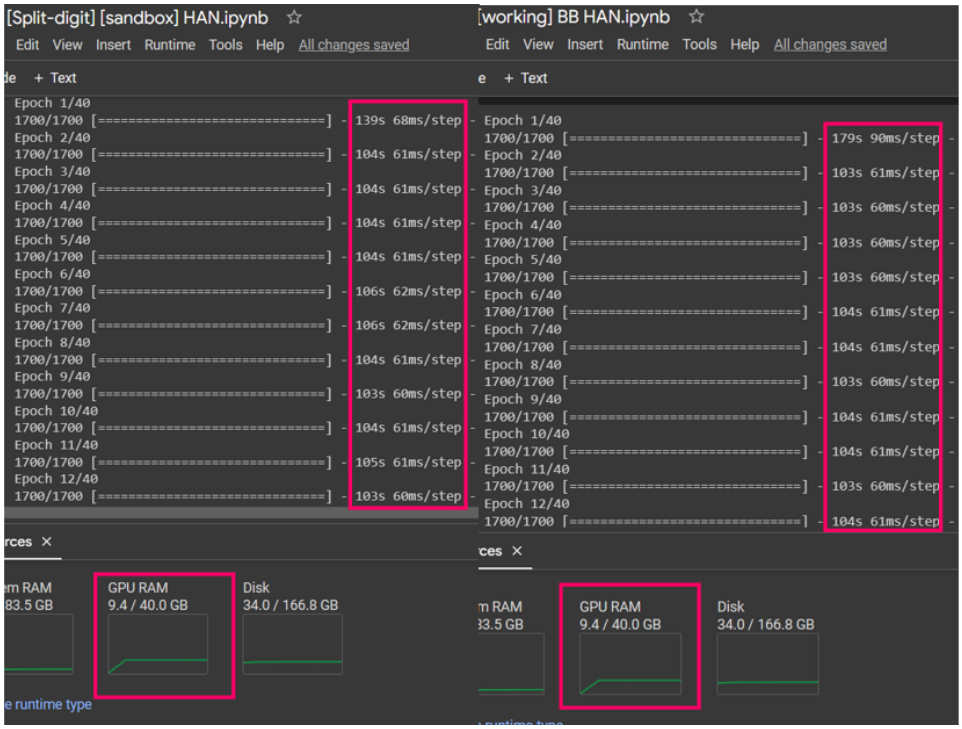


Figure 14 Full-attention HAN (on the left) and Big Bird attention HAN (on the right) performance comparison

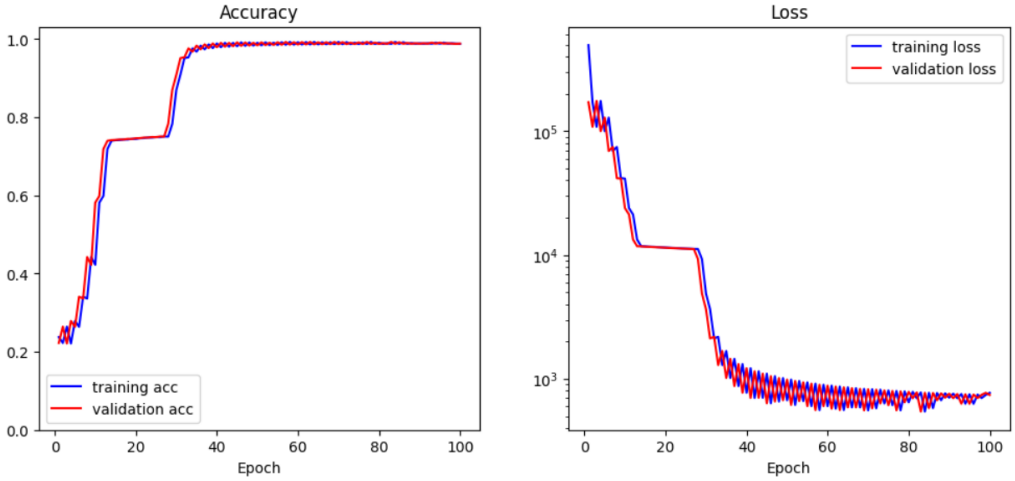


Figure 15 EN-ZH Big Bird HAN accuracy and loss

The stuttering effect remained in the Big Bird attention model, and the sample translation in Figure 16 had a BLEU score of 0.86.

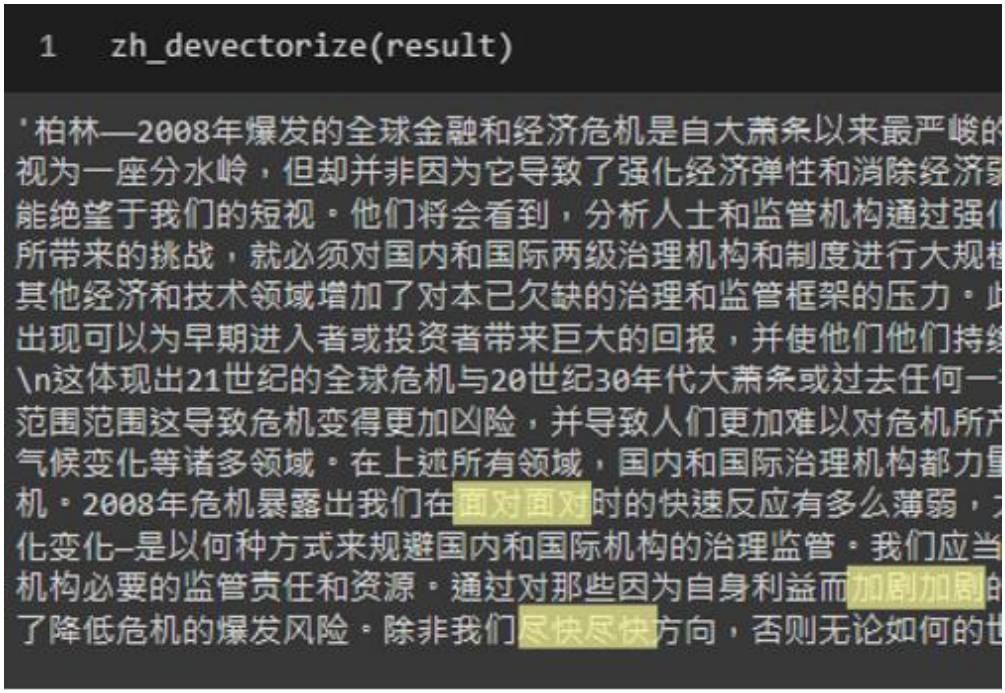


Figure 16 EN-ZH Big Bird HAN result

c. Chinese to English Big Bird HAN translation

As stated in Section IV.1. the input vocabulary space in this experiment was larger than the previous ones: 113k compared to 61k unique vocabularies. Therefore, the model complexity needed to be reduced to avoid out-of-memory error, where there were two layers of attention each with four attention heads.

Figure 17 shows the model eventually reached 95% accuracy after 40 epochs.

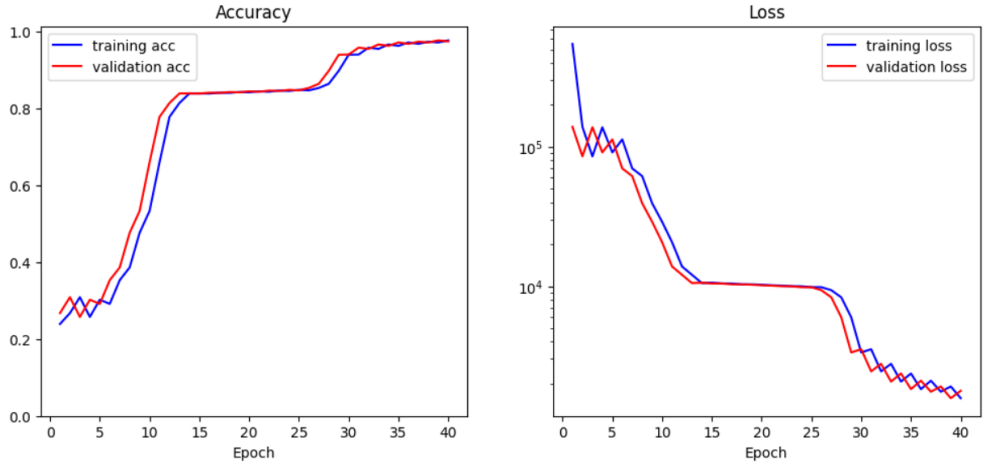


Figure 17 ZH-EN Big Bird HAN accuracy and loss

The sample translation in Figure 18 with a 0.79 BLEU score showed that the model failed to correctly translate several categories, including city names, new line characters, and some other vocabulary. Since previous experiments didn't fail to translate these entities, it is speculated that these failures are caused by reducing model complexity. Again, the stuttering effect remained in this model as the model structure is analogous to the previous experiment.

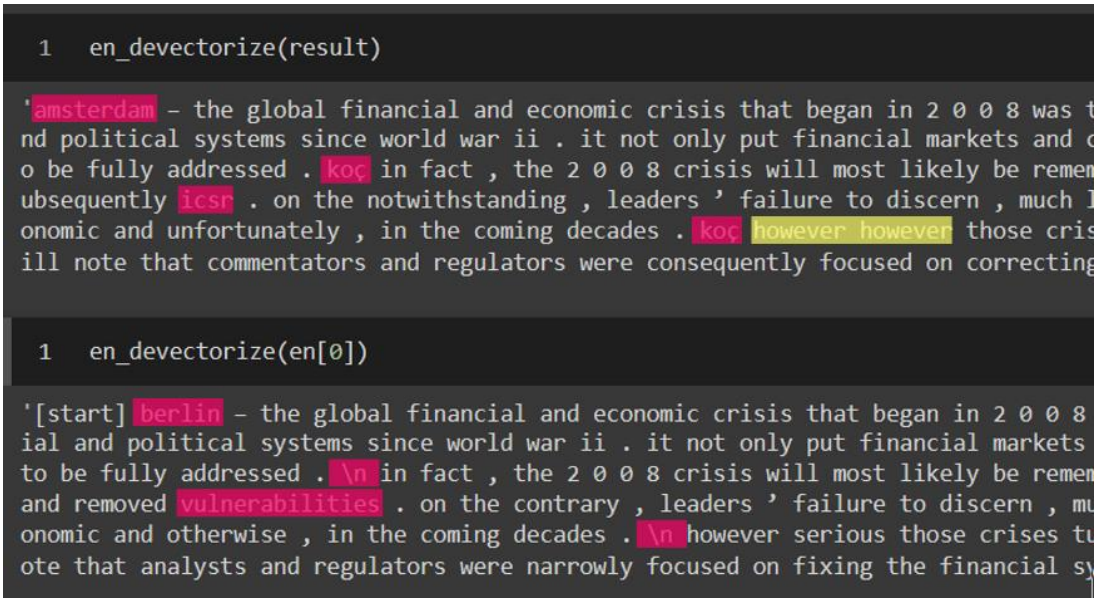


Figure 18 ZH-EN Big Bird HAN result (wrong translations were highlighted in red and stutter was highlighted in yellow)

d. Model comparison

Four models were tested in this project: HAN with gathered numbers and reduced word vector (HAN), HAN with split digits and unfolded word vector (HAN-SD), Big Bird HAN for English to Chinese translation (BB-HAN-EN_ZH), and Big Bird HAN for Chinese to English translation (BB-HAN-ZH_EN). The performance comparison is shown in Table 2.

Model	Training Configuration		Results		
	Attention layers	Attention heads	Validation Accuracy	BLEU score	Training cost (ms/step)
HAN	4	8	0.8	0.44	172
HAN-SD	4	8	0.92	0.9	169
BB-HAN-EN_ZH	4	8	0.96	0.86	171
BB-HAN-ZH_EN	2	4	0.95	0.79	81

Table 2 Model comparison

VI. CONCLUSION

This project proposed utilizing hierarchical attention models to translate documents, and the results in previous sections demonstrated the feasibility of utilizing HAN for document translation tasks.

The Transformer (full-attention) HAN model makes translating documents as a whole possible, while the Transformer couldn't translate the same piece of document as a whole but in chunks. The model first stumbled on correctly translating the numbers, which was later solved by splitting numbers into single digits. The model eventually yields satisfactory results despite the stuttering effect occurring occasionally.

Two experiments were conducted on the Big Bird HAN model: English to Chinese and Chinese to English translation tasks. The two experiments produced successful translations, but the expected efficiency boost was not found in the experiments.

VII. Future Work

Although the experiments showed satisfying results, due to the hardware limitations stated in Section IV.1., the models couldn't optimize the word vectors to better suit the use case in this project. This could be viewed as a way to remedy the stuttering effect happening in the current models.

Simple post-editing rules can be applied to the models to improve the translation readability: in Section V.2.c., some wrong translations followed a pattern, for instance, koç corresponded to the new line character, and those errors could be solved by implementing post-editing rules. Also, in languages like English, the stuttering effect could also be alleviated with post-editing rules, because repetition of words is less common in the language.

Although the Big Bird HAN model was expected to have better efficiency than the full-attention HAN model, such an effect was not observed in this project. As stated in Section V.2.b., the author speculated that it was due to the lack of optimization toward the peculiar attention pattern. With an optimized masking algorithm, the model could achieve a similar boosting effect shown in the Big Bird [7] paper.

REFERENCES

[1] W. Xiong and Y. Jin, "A new Chinese-English machine translation method based on rule for claims sentence of Chinese patent," in *2011 7th International Conference on Natural Language Processing and Knowledge Engineering*, 2011.

[2] S. Yang, Y. Wang and X. Chu, "A Survey of Deep Learning Techniques for Neural Machine Translation," *arXiv preprint arXiv:2002.07526*, pp. 11-21, 2020.

[3] F. Stahlberg, "Neural Machine Translation: A Review," *Journal of Artificial Intelligence Research*, vol. 68, pp. 343-418, 2020.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, pp. 11-21, 2017.

[5] W. He, Y. Wu and X. Li, "Attention Mechanism for Neural Machine Translation: A survey," in *2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2021.

[6] Z. Sun, M. Wang, H. Zhou, C. Zhao, S. Huang, J. Chen and L. Li, "Rethinking Document-level Neural Machine Translation," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

[7] Z. Manzil, G. Guru, D. Avinava, A. Joshua, A. Chris, O. Santiago, P. Philip, R. Anirudh, W. Qifan, Y. Li and A. Amr, "Big bird: Transformers for longer sequences," *Adv. Neural Inf. Process. Syst.,* vol. 33, pp. 17283-17297, 2020.

[8] W. J. Hutchins, "Machine translation: A brief history," in *Concise history of the language sciences*, Elsevier, 1995, pp. 431-445.

[9] E. Charniak, K. Knight and K. Yamada, "Syntax-based language models for statistical machine translation," in *Proceedings of Machine Translation Summit IX ,* 2003.

[10] I. Sutskever, O. Vinyals, Q. V. Le and I. Sutskever, "Sequence to sequence learning with neural networks," *Adv. Neural Inf. Process. Syst.,* vol. 27, 2014.

[11] R. P. Neco and M. L. Forcada, "Asynchronous translations with recurrent neural nets," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, 1997.

[12] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 6, no. 02, pp. 107-116, 1998.

[13] M. Vathsala and G. Holi, "RNN based machine translation and transliteration for Twitter data," *International Journal of Speech Technology,* vol. 23, no. 3, pp. 499-504, 2020.

[14] D. Bahdanau, K. Cho and Y. Bengio, "Neural machine translation by jointly learning to

align and translate," *arXiv preprint arXiv:1409.0473,* 2014.

[15] J. L. Ba, J. R. Kiros and G. E. Hinton, "Layer Normalization," *arXiv preprint arXiv:1607.06450,* 2016.

[16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,* 2015.

[17] F. Chollet and others, *Keras,* 2015.

[18] J. Tiedemann, "Parallel data, tools and interfaces in OPUS," in *Proceedings of the 8th International Conference on Language Resources and Evaluation*, 2012.

[19] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, "Enriching Word Vectors with Subword Information," *arXiv preprint arXiv:1607.04606,* 2016.

[20] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017.

[21] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson,

K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt and S. 1. Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods,* vol. 17, p. 261–272, 2020.

[22] J. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM,* vol. 18, no. 9, pp. 509-517, 1975.

[23] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, "BLEU: A Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics,* USA, 2002.