Smooth Embedding and Word Sampling Research Based on Transformer Pointer Generation Network

Meiwei Zhang and Yichang Wu, Yihui Pang, Hao Wen, and Jiaxin Wang

***Abstract*—Textual reasoning and abstraction, which both take in a long text and generate a short digest, are widely implemented in the area of Natural Language Processing (NLP). See *et al*. pioneer the Seq2Seq and Pointer Generation Network structure to address the summarisation task. Later, Transformer model, a successor of Seq2Seq was developed. However, research on the impact of word frequency on text-generated tasks is not adequate. In this paper, we propose two methods to evaluate the effect of word frequency: Smooth Embedding and Word Sampling. The experiments witness the improvement of Smooth Embedding performance. On the contrary, Word Sampling fails to meet our expectation. It increases the sensitivity of noise, which is a symbol of over-fitting.**

***Index Terms*—Smooth embedding, text generation, sequence to sequence model, pointer generation network, attention mechanism.**

I. Introduction

Textual reasoning and automatic summarisation aim to get a short sentence which covers the general meaning of long text. Basically, extractive and abstractive approaches are the main methods for textual reasoning and summarisation tasks. Extractive method cannot understand the text and generate ideas like humans [1]. Abstractive method is more suitable for understanding and reasoning tasks. A successful abstractive algorithm should demonstrate the core information in a sentence as the way natural language does.

After 2017, Transformer model started to be used in some kinds of Natural Language Processing (NLP) tasks. And Transformer gets better performance compared with Seq2Seq in some generation tasks [2]. Therefore, this paper deploys Transformer instead of Seq2Seq before PointerGeneration Network.

Abstractive method, however, has significant shortcomings - the content inaccuracy and repetition caused by Out-Of-Vocabulary (OOV) words. They lead to the inefficient decoding, although we can use attention or selfattention mechanism to get the weights between inputs and outputs.

Manuscript received October 12, 2019; revised August 7, 2020.

Meiwei Zhang and Yichang Wu are with Athlone Institute of Technology, Athlone, Ireland (e-mail: zhangmwchris@gmail.com, w.yichang@research.ait.ie).

Yihui Pang is with Chinese Academy of Agricultural Science, Beijing, China (e-mail: YihuiPang_543@outlook.com).

Hao Wen and Jiaxin Wang are with the University of Electronic Science and Technology of China, Chengdu, China (e-mail: wh2015uestc@gmail.com, wangjiaxin966@gmail.com).

The assumption for self-attention mechanism is that all the words in our vocabulary hold the same weight. However, the most frequent words provide less information than the rare words [3]. Accordingly, rare words should be emphasized during text understanding and automatic summarisation.

In this paper, we propose two methods to evaluate the influence resulted from frequency of words. The first approach smooths the high frequency words through embedding. This strategy sharing the idea of TF-IDF in information retrieval [4] By Smooth Embedding, the high frequency words, such as "a, the, to, for", will decrease their weights during encoding and decoding. Another solution adapts Word Sampling in the decoding stage. Transformer use self-attention mechanism to get "relationship" between input words and output words by context matrix. It will be multiplied with the original words' distribution. Finally, we implement our two approaches along with a baseline, i.e. standard Transformer Pointer-Generation Network. Three models are evaluated with the Baidu Auto Online Consultation Summarisation 2019 dataset.

II. Background

Automatic summarisation experiences great progress recently. Early research has focused on extraction methods [2], which prioritise each sentence's importance score from the original document. Scoring methods mainly depend on frequency and stochastic topic models [5].

General methods of sentence selection include Maximum Margin Correlation (MMR), Integer Linear programming (ILP). In recent years, direct prediction using neural networks has been proven to be effective [6]. Given the relative importance of sentences in a selected set, the extractive method has the advantage of retaining the original information more completely, especially guaranteeing the coherence of each sentence, but it is suffering from redundant information. On the other hand, with the rapid development of Deep Learning technology over the years, abstractive method has been evolving. RNNbased structure is successfully applied to the area of NLP, including but not limited to syntax Analysis, text summarisation, dialogue system. nallapati *et al*. created encoder-decoder structure and attention mechanism to address summary tasks and obtained good results in 2015 [7]. Later, researchers extend the work and combine other features into the model to get better results. For example, graph-based attention mechanism proposed by Tan and Wan improves the generalization of the model [8]. Li *et al*. add history proposal as the latent variable into encoder to extract complex substructure [9]. Nallapati *et al*. use replication mechanism [7] to solve OOV problems [10].

doi: 10.18178/ijmlc.2021.11.3.1045

Regarding the internal correlation research between sentences, we can see that self attention is one of the representatives. Self-attention has been applied to many aspects such as NLP, CV, and speech processing, and has achieved good results.

Self Attention is very different from the traditional Attention mechanism: the traditional Attention is based on the hidden state of the source and target to calculate Attention, and the result is the dependence between each word on the source and each word on the target relationship. But Self Attention is different. It is carried out on the source side and the target side. Only the Self Attention related to the source input or the target input itself captures the dependency between the words on the source side or the target side; The self Attention obtained is added to the Attention obtained on the target end to capture the dependency between words and words on the source end and the target end. Therefore, self Attention is better than the traditional Attention mechanism. One of the main reasons is that the traditional Attention mechanism ignores the dependency between words in the source or target sentence. In contrast, self Attention can not only get the dependence relationship between the words and words of the source and target ends, and the dependence relationship between the words and words of the source or target end itself can also be effectively obtained.

Let's take a look at a translated example "I arrived at the bank after crossing the river". Does the bank refer to the bank or the river bank? This requires us to contact the context. When we see the river, we should know that the bank is very large. Probability refers to the river bank. In RNN, we need to process all the words from bank to river step by step, and when they are far apart, the effect of RNN is often poor, and because of its sequential processing efficiency is also low. Self-Attention uses the Attention mechanism to calculate the association between each word and all other words. In this sentence, when the word bank is translated, the word river has a higher Attention score. Using these Attention scores, you can get a weighted representation, and then put it into a feedforward neural network to get a new representation, which takes into account contextual information. As shown in the figure below, the encoder reads in the input data, and uses the self-attention mechanism of superimposed layers to obtain a new representation of each word that takes into account the context information. Decoder also uses a similar Self-Attention mechanism, but it not only looks at the previously generated output text, but also the output of the attend encoder.

For self-attention, the three matrices *Q*(Query), *K*(Key), and *V*(Value) all come from the same input. First, we need to calculate the dot product between *Q* and *K*, and then in order to prevent the result from being too large, Will be divided by a scale$\sqrt{d_k}$, where d_k is the dimension of a query and key vector. Then use the Softmax operation to normalize the result to a probability distribution, and then multiply it by the matrix *V* to get the weighted summation. This operation can be expressed as

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

But the level of self attention is the embedding level, not from the frequency of words to consider, which is the reason for this paper.

A. Word Embedding

Word Embedding method converts words as a continuous vector which captures the lexical and semantic characteristics of words in a dimensional space [11]. Socher *et al.* get internal representation from text neural network model [12]. Our work is most directly related to continuous vector. For most NLP tasks, Blacoe *et al.* define each word as a dimension fixed vector which can be static or dynamic, and then compose word embedding [13].

B. Word Sampling

According to Word2Vec, Negative Sampling is an auxiliary method of hierarchical softmax [14]. Skip-Gram model in Word2Vec, aims to find the positive word through Ngram words [11]. But how to select negative words from the rest, Word2Vec proposes Negative Sampling [11], which starts with building word distribution within sentence. To get the distribution, Word2Vec defines a sampling rate for each word which is associated with word's frequency [15]. The "negative samples" are selected using a "unigram distribution", where more frequent words are more likely to be chosen. For instance, suppose you have entire training corpus as a list of words, you choose 5 negative samples by picking randomly from the list. This is expressed by the following equation:

$$P(w_i) = \frac{f(w_i)}{\sum f(w_j)} \quad (2)$$

Mikolov *et al.* tried a number of variations of this equation, and the best practice is depicted in following equation [16]:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum f(w_j)^{3/4}}$$

III. Model

A. Transformer Model

Up to now, Transformer, a new neural architecture based on Seq2Seq, introduces Self-Attention mechanism to encoder and decoder. Self-Attention utilizes multi-head attention mechanism that provides Transformer dependencies regardless of their distance in input and output sentence. One upmost advantage that Transformer has is supporting parallel processing, thus reducing training time. Transformer can produce groundbreaking results in Machine Translation [2]. Sanjabi *et al.* also prove that Transformers perform better than various Seq2Seq models in NLP tasks [17].

B. Pointer-Generation Network

Abigail and Liu proposed the Pointer Generator Network which is a hybrid model combining Attention-based Seq2Seq model and Pointer Network [18]. The model is capable of generating a new word from a fixed vocabulary and coping a word from the original word literature. In addition, Coverage Mechanism is deployed to solve duplicate problems. Formula 3 produces P_{gen}, which indicates the possibility of generating a new word, taking context vector h^*_t, decoder hidden state S_t, decoder input X_t into consideration. Generation probability P_{gen} is deduced by a linear layer and sigmoid function

$$P\_gen = \sigma(W_{h*}^T * h_t^* + W_s^T * S_t + W_x^T * X_t + b_{ptr}) \quad (4)$$

Except h^*_t, S_t, X_t, the rest are trainable parameters.

As Formula 4 depicted, the word distribution $P(w)$ takes in the vocabulary distribution P_{vocab} and attention distribution a^t_i and weighted by P_{gen}, $1-P_{gen}$ respectively.

$$P(w) = P_{gen} * P_{vocab}(w) + (1 - P_{gen}) \sum a^t_i \quad (5)$$

A Coverage Mechanism, introduced in the Pointer Generator Network, reduces duplication issues. Coverage vector c^t, i.e. total attention distribution, records the coverage of received words from the attention mechanism. The coverage mechanism also performs additional calculations in loss function to inhibit repeated words. The whole calculation processes are as the following:

$$c^t = \sum \sigma^{t'} \quad (6)$$

$$e^t_i = V^T tanh(W_h * h_i + W_s * S_t + W_c * c^t_i + b_{attn}) \quad (7)$$

$$covloss_t = \sum min(\sigma^t_i, c^t_i) \quad (8)$$

$$Loss = \frac{1}{T} * \sum[-logP(w^*_t) + \mu \sum covloss_t] \quad (9)$$

C. Smooth Embedding

Given a sentence= [$word_1$, $word_2$, $word_3$, …, $word_j$], and Z($word_i$) representing the frequency of $word_i$ in our corpus, we calculate P($word_i$) as the smooth frequency of $word_i$ in Formula 9, where α is the smoothing factor. For most of various text similarity tasks, this method provides much better performance than the unweighted methods, such as complex surveillance methods (RNN and LSTM model [3]). This method is used in various types of corpora to calculate the vector of sentences.

$$P(w) = \alpha / (\alpha + Z(word_i)) \quad (10)$$

$$W_{vec} = W_{vec} * P(w) \quad (11)$$

If we change the parameter α, the different results are as below:

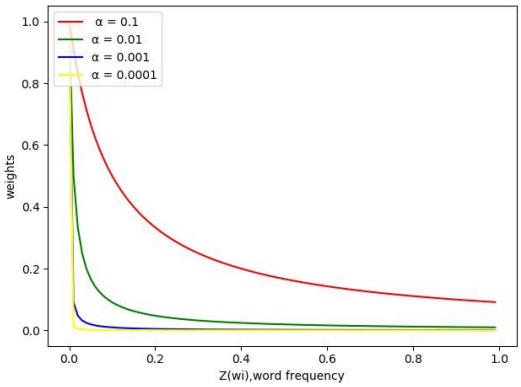

Fig. 1. Smooth frequency.

From Fig. 1, we conclude:

- if α = 0.1 and the word frequency is 0.5, the smooth results will be reduced to about 0.2, which means the elements of words vector will decrease 80% accordingly.
- As α decreases, the smooth results drop sharply. The model goes under-fitting.
- In the contrary, with the rise of α, the effect of smooth diminishes.

D. Decoder Word Sampling

According to Vaswani [19], Transformer utilizes selfattention which focus on the word's relation within inputs instead of between inputs and outputs. Considering that, we propose to adapt word sampling during decoding, which takes the original importance of input words into consideration.

$$P(word_i) = \left(\frac{Z(word_i)}{\beta} + 1\right)^{\frac{1}{2}} * \frac{\beta}{Z(word_i)} * \frac{1}{2} \quad (12)$$

$$weight_j = \frac{P(word_j)}{\sum P(word_j)} \quad (13)$$

where $weight_j$ should satisfy the constraint:

$$1 = \sum weight_j \quad (14)$$

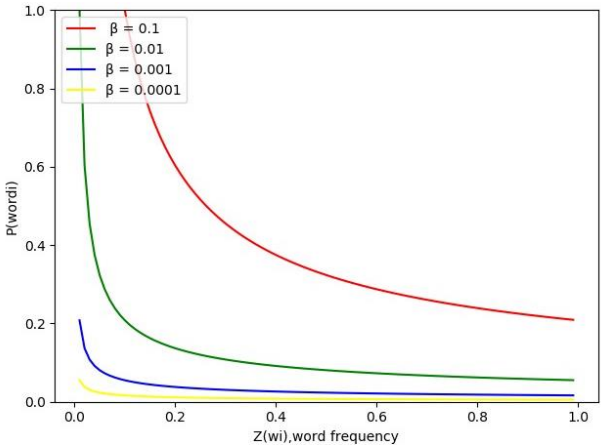

Fig. 2. Smooth in word sampling.

$P_{vocab}(w)$ represents the probability of each input word being selected at each position in output sentence:

$$P_{vocab}(w) = \begin{bmatrix} w_{11} & \cdots & w_{j1} \\ \vdots & \ddots & \vdots \\ w_{i1} & \cdots & w_{ji} \end{bmatrix} * \begin{bmatrix} weight_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & weight_j \end{bmatrix} \quad (15)$$

Fig. 2 demonstrates the effect of word frequency in Sampling. On the horizontal axis is $Z(w_i)$, the frequency of word i. On the other side, vertical axis indicates $P(w_i)$, probability of being kept in sentence. By altering the sampling factor β, we perform 4 experiments. From them we conclude:

- When β = 0.001 and $z(w_i)$<=0.00089, $P(w_i)$ = 1.0. It means the rare words like name or address will be sampled.
- As β decreases, the word's probability of being sampled will drop shapely. As a result, the matrix in Formula 14 will turn sparse which represents underfitting.
- In the contrary, with the rise of β, Word Sampling mechanism becomes invalid.

To implement the Decoder Word Sampling, we construct the Transformer model as Fig. 3.

IV. Experiments

1) Dataset: Our dataset, coming from an automotive consultancy, is render from Baidu Paddle AI community. It contains the consumer inquires along with the experts' advices. There are 130,000 samples which contains 20 million car owners' questions covering almost all types of

vehicles on the market and various car problems. In our experiment, we use 80 percents of the dataset as our training dataset, 10 percents as our validating dataset, 10 percents as our testing dataset. The dataset length information is illustrated at the table below.

2) Implementation: To evaluate our proposals, we choose Transformer pointer-generation network as our baseline, comparing with Smooth Embedding and Decoder Word Sampling approaches.

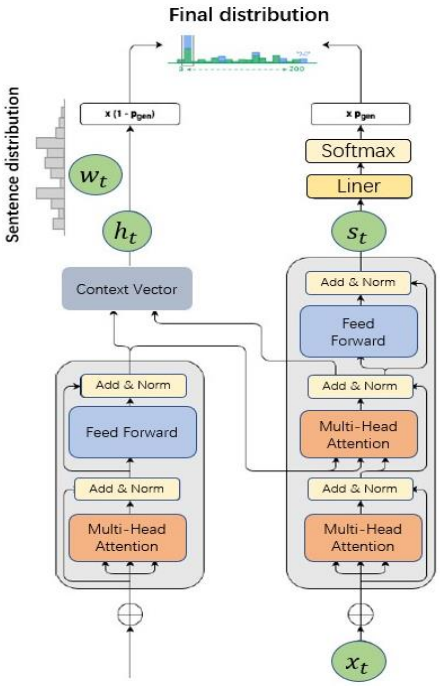

Fig. 3. Decoder sampling structure.

TABLE I: DATASET LENGTH INFORMATION

Length	Training Set		Validating Set		Testing Set	
	inp	outp	inp	outp	inp	outp
Max	1035	91	988	82	902	88
1 Dev	170	26	174	27	164	29
2 Dev	500	50	445	51	432	54
3 Dev	1000	83	789	80	861	82

Notes. The table shows dataset information: Dev, inp, outp means Deviation, the input sentence, the summary results respectively. We assume that the length of sentence fits Gaussian Distribution.

After data cleaning, we select top 50K words as our original vocabulary. Then we use the corpus to train our Word2Vector model so as to build the static embedding for the neural network. The dimension of word vector is set to 256 which matches the decoder Gated Recurrent Units (GRU) numbers. According to the Table I. We set maximum encoding length i.e. the maximum length of the input sequence to 400, and the decoding length i.e. the output sequence length is 50, and our model is trained by Adam Optimizer with learning rate of 0.001. All the code is implemented by *TensorFlow* 1.14.0 on an *NVIDIA TITAN XP* GPU.

3) Evaluation: In our experiments, we apply Recall-Oriented Understudy for Gisting Evaluation (ROUGE) as evaluation criterion. ROUGE is widely used in text generation research field for content evaluation. ROUGE calculates the recall rate of N-grams or words between the model-generated and the human summary. In each experiment, we calculate ROUGE-1, ROUGE-2, ROUGE-L scores respectively.

$$Rouge_n(c) = \frac{\sum s \in ref \sum gram_{n \in s} match(gram_n)}{\sum s \in rs \sum gram_{n \in s} count(gram_n)} \quad (16)$$

Besides, we also randomly exhibit two samples to give an explicit overview.

4) Experimental Results and Analysis: The experimental results are aggregated in Table II and III.

TABLE II: ROUGE RESULTS

Models	ROUGE-1	ROUGE-2	ROUGE-L
Baseline	36.21	16.49	17.56
Smooth Embedding	38.35	17.41	18.57
Word Sampling	35.1	15.64	16.84

The higher a ROUGE is, the better the model performs. As we can see in Table II, Smooth Embedding scores higher than the Baseline, 2.14 percents, 0.92 percent, 1.01 percent respectively. On the contrary, Word Sampling doesn't work as we expect. After reviewing the whole process of Word Sampling, we find that the sensitivity of noise increases while we try to suppress the high frequency words. It is a symbol of over-fitting.

V. CONCLUSION

Since research on the impact of word frequency on text-generated tasks is not adequate, we propose two methods to evaluate this effect: Smooth Embedding and Word Sampling. Smooth Embedding balances the embedding through word frequency. The final results show that smooth embedding improve summary's informativeness. Word Sampling complements self-attention mechanism theoretically. However, the actual abstract results are not effective as Smooth Embedding. It increases the sensitivity of noise while we try to suppress the high frequency words, which indicates over-fitting.

TABLE III: TESTING RESULTS DETAILS

Document: The right front door wheel fender is recessed, and the right door cannot be opened. The technician said: Hello. This is caused by deformation and extrusion, and it can only be opened after going to a repair shop for sheet metal. Sheet metal is done and then painted. The cost is about 600 yuan. Owner said: OK, the door cannot be opened due to deformation and squeezing Original Summary: After doing the sheet metal paint repair, it can be opened. **Baseline**: Construction may be caused by sheet metal paint. **Smooth Embedding+Baseline**: It is expensive to make sheet metal paints. It takes UNK to do sheet metal painting. Follow UNK. **Encoder word sampling**: The background is optimistic that it will do sheet metal spray painting. Generally, the repair shop needs UNK.
Document: Dongfeng Fengshen h30cross 2013 automatic transmission clock multi-function display how to call up the fuel consumption and range mileage interface, the technician said: Hello, the adjustment button of this model is on the dashboard, the owner said: Hello, how to adjust specifically the technician said: After turning on the key switch, the meter lights up, press the button Original Summary: Turn on the key switch and press the meter button. **Baseline**: It is recommended that you use a dedicated diagnostic computer for resetting. Reading fuel consumption can open the functions on the dashboard. **Smooth Embedding+Baseline**: See the picture method. The test picture has been sent to you to test whether there is a dashboard upload to focus on the future. **Encoder word sampling**: For dynamic wind adjustment, just press and hold the button on the instrument panel to adjust the instrument automatically.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

For this paper, Meiwei Zhang and Yichang Wu contribute to the structure designing and paper writing. Hao Wen, Jiaxin Wang and Yihui Pang set up the experimental environment, help to preprocess the data at the first stage and carefully review this paper.

AIT Software Research Institute helped us provide hardware environment and some guidance throughout the experiment.

References

[1] X. Jiang, P. Hu, L. Hou, and X. Wang, "Improving pointer-generator network with keywords information for chinese abstractive summarization," in *Proc. CCF International Conference on Natural Language Processing and Chinese Computing*, Springer, 2018, pp. 464–474.

[2] J. Baan, M. ter Hoeve, M. van der Wees, A. Schuth, and M. de Rijke, "Do transformer attention heads provide transparency in abstractive summarization?" *arXiv preprint arXiv:1907.00570*, 2019.

[3] S. Arora, Y. Liang, and T. Ma, "A simple but toughto-beat baseline for sentence embeddings," 2016.

[4] D. Hiemstra, "A probabilistic justification for using tf× idf term weighting in information retrieval," *International Journal on Digital Libraries*, vol. 3, no. 2, pp. 131–139, 2000.

[5] J.-g. Yao, X. Wan, and J. Xiao, "Recent advances in document summarization," *Knowledge and Information Systems*, vol. 53, no. 2, pp. 297–336, 2017.

[6] Z. Cao, F. Wei, W. Li, and S. Li, "Faithful to the original: Fact aware neural abstractive summarization," in *Proc. Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequenceto-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[8] J. Tan, X. Wan, and J. Xiao, "From neural sentence summarization to headline generation: A coarse-tofine approach," *IJCAI*, pp. 4109–4115, 2017.

[9] P. Li, W. Lam, L. Bing, and Z. Wang, "Deep recurrent generative decoder for abstractive text summarization," *arXiv preprint arXiv:1708.00625*, 2017.

[10] L. Wang, J. Yao, Y. Tao, L. Zhong, W. Liu, and Q. Du, "A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization," *arXiv preprint arXiv:1805.03616*, 2018.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[12] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, "Grounded compositional semantics for finding and describing images with sentences," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 207–218, 2014.

[13] W. Blacoe and M. Lapata, "A comparison of vectorbased representations for semantic composition," in *Proc. the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, 2012, pp. 546–556.

[14] M. U. Gutmann and A. Hyvärinen, "Noisecontrastive estimation of unnormalized statistical models, with applications to natural image statistics," *Journal of Machine Learning Research*, vol. 13, pp. 307–361, 2012.

[15] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," *arXiv preprint arXiv:1511.08198*, 2015.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[17] N. Sanjabi, "Abstractive text summarization with attention-based mechanism," Master's thesis, Universitat Politècnica de Catalunya, 2018.

[18] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *arXiv preprint arXiv:1704.04368*, 2017.

[19] 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*, pp. 5998–6008, 2017.

Meiwei Zhang was born in Zi Gong, Sichuan Province in China in 1991. He received the B.S degree in the University of Electronic Science and Technology of China in 2015, and now studying the master of software engineering in natural language processing at Athlone Institute of Technology in Ireland. Since 2015, he had worked in AI companies like IBM and Avaintec for three years, focusing on the deep learning in NLP. His research interests include natural language understanding, recommender system, data mining.

Yichang Wu was born in Tianmen, Hubei Province, China in 1990. He was awarded the degree of bachelor of engineering in 2012, majoring in thermal energy and power engineering. After being as an electrical engineer for 4 years, he was enrolled as a postgraduate researcher in Athlone Institute of Technology fully funded by Government of Ireland (GOI) International Education Scholarship. His research interests involve blockchain, trusted computing, machine learning.

Hao Wen was born in Chongqing, China in 1993. He received the B.S degree in the University of Electronic Science and Technology of China in 2015, and now studying the master in Statistics in the University of Electronic Science and Technology of China. Since 2019, he has been an internship in the Financial Technology Department of China Construction Bank. His research interests include data mining, statistical machine learning and high-dimensional data analysis.

Jiaxin Wang was born in Ruzhou, Henan Province in China in 1997. She received the B.S degree in Henan University in China in 2019, and now studying master in the University of Electronic Science and Technology of China. Her research interests include applied mathematics, statistics and complex network.

Yihui Pang was born in Yun Cheng, Shanxi Province in China in 1995. She received the B.A degree in the Central University of Finance and Economics of China in 2017, and now studying the master of library and information science in knowledge discovery at Agricultural Information Institute of Chinese Academy of Agricultural Science in China. Between 2017 and 2019, she worked at a Financial Technology company and accumulated a lot of expertise in AI and data mining. Her research interests include knowledge discovery, text sentiment analysis, data visualization.