Proceedings of CONF-CDS 2025 Symposium: Data Visualization Methods for Evaluatio
DOI: 10.54254/2755-2721/2025.PO24517

# *Beyond Attention: Evaluation of the Capabilities, Challenges, and Future of Transformer Models for Natural Language Generation*

**Zhehao Xie**

*School of Automation, Shanghai University of Electric Power, Shanghai, China*
*xzh1184699381@gmail.com*

***Abstract.*** The coming of the Transformer architecture marks a really important time in natural language processing (NLP). It basically changed the way we do sequence-to-sequence tasks. This paper looks back at the development that led to the Transformer and makes clear its main principles, especially the self-attention mechanism that helps with advanced modeling of long-range dependencies compared to the previous recurrent architectures. This study checks out different uses of the Transformer in text generation. It looks at how well it works and the problems it has in areas like making long texts (for example, creative writing), code generation, and multilingual translation. Through large-scale pre-training, Transformer-based models like GPT and BERT have gotten top results. But there are still big problems. These include the quadratic complexity of self-attention. This makes it less efficient for long sequences. Big models have high computing costs and are hard to understand. There's exposure bias in the generation process and problems making sure it's controllable, consistent, and evaluating the quality of the generated text. To make this industry grow in the future, it's really necessary to solve these limits and the related ethical problems.

***Keywords:*** Transformer model, Natural Language Processing (NLP), Text Generation, Self-Attention Mechanism, Large Language Models (LLMs)

## 1. Introduction

Natural Language Processing (NLP) has seen huge growth, from the early rule-based systems to statistical methods, and now to the stage of deep learning. The first appearance of the Transformer architecture in 2017 was a big step forward. It became the main model fast because it was so good at getting context information through the self-attention mechanism. Models using this architecture often have billions of parameters and are pre-trained on a lot of text collections. They do really well in all kinds of NLP tasks, especially in text generation. But there are still many problems and difficulties. In the traditional training ways, making high-quality and connected long texts has problems linked to limited feedback signals. Looking for controllable generation that mixes the wanted things like style or topic with fluency and creativity is still a big research challenge. The built-in computational complexity, hard-to-understand parts, and possible biases of big Transformer

DOI: 10.54254/2755-2721/2025.PO24517

models bring up big technical and ethical problems. This study wants to take a full look at how Transformer models are used in different text generation tasks, like in multilingual situations, code generation, and long creative writing. We'll check out the specific problems in these areas and deal with the current limits of the Transformer architecture, like controllability and efficiency. The aim of this review is to find the key areas for future research, especially focusing on making strong evaluation methods, making the text better and more controlled, making the model more efficient, and thinking about ethical effects. So, this will help the reliable growth of Transformer-based text generation technology.

## 2. Overview of the transformer model

### 2.1. The evolution of natural language processing

The development of natural language processing (NLP) has gone through three main stages: the symbolic time (about from the 1950s to the 1980s), the coming of statistical methods (from the 1990s to the early 2000s), and the deep learning revolution (from 2010 up to now). The hand-made rules that experts made were the base of the early NLP efforts, including the machine translation experiment by Georgetown University and IBM in 1954. Rule-based systems were hard to keep up and were "weak" when dealing with unexpected inputs because of their complexity, flexibility, vagueness, and the dynamic nature of language. The change to statistical and machine learning ways let computers learn patterns directly from a lot of language collections. But N-gram models, Hidden Markov Models (HMMs), and Probabilistic Context-Free Grammars (PCFGs) were used for marking parts of speech and analyzing sentences, but they couldn't catch long-range dependencies [1].

Around 2010, neural networks, especially deep learning, started to completely change NLP. Word embeddings gave dense vector representations for words, and recurrent neural networks (RNNs) and their kinds like LSTMs and GRUs were good at dealing with sequential text data. Recently, attention mechanisms and the Transformer architecture, especially its self-attention mechanism, have been very disruptive. Self-attention is good at catching the dependencies between any two words in a sentence, no matter how far apart they are, and it can improve the understanding of long texts a lot. The current top large language models (LLMs), like GPT and Gemini, are mainly based on this Transformer architecture.

### 2.2. Fundamental principles of the transformer model

#### 2.2.1. Model architecture

Transformer is a kind of neural network architecture made for sequence-to-sequence tasks, like machine translation. It doesn't use the traditional recurrent layers and convolutional layers. It only depends on the attention mechanism. As can be seen from Figure 1, it mainly has an encoder and a decoder.

The encoder: It's built as a stack of N same layers. Each layer has two sub-layers: the multi-head self-attention mechanism and the position-wise feed-forward network. Residual connections are used on the output of each sub-layer, and then there's layer normalization.

The decoder: It's also made up of N layers that are the same. Besides the two sub-layers in the encoder, the decoder also has a third sub-layer: the encoder-decoder attention mechanism, which pays attention to the output of the encoder. It's very important that the self-attention sub-layer in the

DOI: 10.54254/2755-2721/2025.PO24517

decoder is masked to stop positions from paying attention to the ones that come after. This keeps the autoregressive property that's needed for generation. Just like the encoder, each sub-layer in the decoder uses residual connections and layer normalization [2].

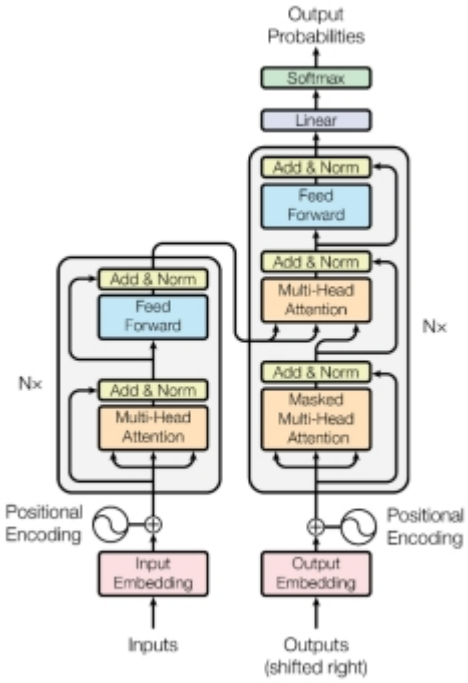


Figure 1: The transformer - model architecture [2]

### 2.2.2. Self-attention mechanism

The self-attention mechanism is the heart of the Transformer architecture. It lets the model make sequence representations by looking at how important different parts of the input sequence are. The model uses a scaled dot-product attention mechanism. For every position in the input sequence, the query (Q), key (K), and value (V) vectors are made through linear transformations from their vector representations. To calculate the attention weights, we take the dot product of the query vector and each key vector, normalize it by the square root of the key vector dimension (dk), and then use the Softmax algorithm. These weights show how important each position is compared to all the other positions. So, we use these attention weights to do a weighted sum of the value vectors to get the self-attention output for that specific position.

The multi-head attention mechanism is used to catch information from multiple representation subspaces at the same time. This means using different learned linear projections to project the queries, keys, and values h times. Then, the attention mechanism is carried out at the same time in each prediction round. The final attention output is connected and projected through an extra linear transformation to make the final multi-head attention output.

## 2.3. The significance and role of the transformer model in NLP

When the Transformer model showed up, the field of natural language processing (NLP) had a big change. This architecture was first put forward by Vaswani and others in their 2017 work. It quickly and deeply changed the main ways in this field. Transformer-based systems, like the models GPT and BERT, have done a lot better than earlier top technologies, like recurrent neural networks

DOI: 10.54254/2755-2721/2025.PO24517

(RNNs), including their variations like LSTMs and GRUs, in a wide range of uses. This big success is mostly because the Transformer can deal with the key problems that RNNs have. Specifically, these problems include the challenges of doing parallel computing tasks and the always-present vanishing gradient problem. This problem stops us from catching the long-distance relationships in sequences effectively [3] .

Transformer and the things that come from it have not only become the main architecture and base of modern NLP, but also made the example of large-scale unsupervised pre-training and then fine-tuning for specific tasks popular. In this example, models first get general language knowledge from a lot of unlabeled text data. Then, they can quickly get used to different downstream tasks by using a relatively small amount of labeled data. This lowers the difficulty of making special NLP applications a lot [4].

## 3. Applications of transformer in diverse text generation tasks

### 3.1. Transformer applications in long text generation

#### 3.1.1. Concept and challenges of long text generation

Long text generation is all about making long, connected, and full-of-meaning text samples. It's super important for apps like machine translation, chat systems, and image captions. But the current generation models, especially the ones based on Generative Adversarial Networks (GANs), often have trouble making text sequences that are longer than 20 words. This limits how useful they are. One big problem comes from the limits of the feedback signal from the discriminator in GANs. In the normal GAN setup, the binary (true/false) feedback is usually only there when the whole generated text is done. This makes the signal rare. Also, this one single signal for the whole text doesn't have much info in the middle about how the text is being made. So, it can't really help with how the grammar and meaning flow. Even though improvements like feature matching have shown some hope for shorter texts, the feedback signals are still only available at the end of the sequence. So, getting past these limits from the rare and often not-so-helpful feedback signals is a key problem to solve if we want high-quality long text generation [5].

#### 3.1.2. Transformer applications in creative writing (novel/script generation)

Large Language Models (LLMs) based on the Transformer architecture have made big steps in creative writing. Let's take NovelAI as an example. This platform gives AI help for making text and images. It helps users write, make up stories, and even create virtual friends. By using a fine-tuned GPT version, NovelAI is really good at making text that's both connected and full of imagination.

In the area of interactive story-telling, "AI Dungeon" has a lot of fans. This free online text adventure game has several modes, including OpenAI's GPT. It makes storylines and interactions based on what players type in. This lets players make their own adventures and work with the AI system to make up stories.

#### 3.1.3. Case study analysis

Marco and his team looked closely at how NovelAI makes text, especially its originality and inner value. [6] Their study showed that even though the platform can make creative writing, to get the best results, users often need to change its parameters by hand. This finding shows clearly the

DOI: 10.54254/2755-2721/2025.PO24517

conflict between being creative and having a good structure. This is something we often see in systems based on the Transformer model.

Specifically, NovelAI, which runs on the Transformer architecture, has a clear tendency to use common sayings and repeat patterns. But users can control this to some extent. Research shows that NovelAI can balance making creative content and having a good chat. This example shows how hard it is to set common standards to judge the "value" of creative works, especially the ones made by AI text generators.

#### 3.1.4. Current challenges

Right now, using Transformer models for creative things like writing novels and scripts has at least three big problems.

First, judging the creative output of AI is a hard problem. It's mixed up with complicated ethical and cultural things and might even touch on deep philosophy. Also, the way humans tell if something is creative is, by nature, subjective. It's affected by things like cultural background, education, and personal ideas. Making better and more complete ways to judge is still a main goal for researchers [7].

The second big problem is the balance that's often seen in Transformer models between being creative and staying consistent when making text. Even though big language models usually make text that's clear and connected, sometimes they aren't very new or surprising. Making AI systems stay really creative and imaginative while also making the text make sense is a big technical problem.

Finally, how users see this content is limited by what Marco and others call "author bias". That means AI-generated text is seen differently from text written by humans. This might stop AI-based writing help tools from being widely accepted and used. So, making users more confident and open to AI-generated content is another important thing to think about.

### 3.2. Transformer applications in emerging text generation tasks

#### 3.2.1. Code generation

The Transformer architecture model is getting more and more recognition because of its big potential in the field of code intelligence. It's made big progress in how code is made and understood. Actually, research results show that the big-scale iterations of these Transformer models can solve complex programming challenges effectively, like the ones you see in competitions. They do this by building full functional code based on the natural language specs given [8].

At the same time, these models have been put into integrated development environments (IDEs) to give AI-powered suggestions for code completion. By looking at the code situation right then, these systems suggest relevant methods and API calls [9]. This makes developers a lot more efficient. Taking this further, Svyatkovskiy and others used Transformer technology to make code completion better. They made it work for a wide range of programming languages and different general situations. The way their team designed it showed they were good at predicting sequences of code parts that weren't limited (not just simple method names or parameter values). They could even put together full, grammatically right lines of code. This gave developers really detailed help [10].

Basically, the things we found from these studies say that the Transformer design is really good at understanding the fine structure and small differences in meaning of software code at the same time.

DOI: 10.54254/2755-2721/2025.PO24517

It can also make advanced code things. This brings a new and innovative stage to software development methods.

### 3.2.2. Multilingual text generation

The coming of the Transformer-based framework is a big step forward in making text for many languages. It lets one model handle translation tasks for different language pairs. A simple but powerful strategy is to use the normal Transformer setup and add a special token at the start of the input string to say what the output language should be. How well this works shows that one universal framework can meet the needs of multilingual neural machine translation (NMT). Really importantly, it also makes it possible for zero-shot translation. That's when you translate between language pairs that the model wasn't trained on specifically [11]. Building on this first success, later academic research has made these Transformer-based systems able to handle over a hundred different languages at the same time. A lot of experiments have looked at the trade-offs between translation quality, the number of languages supported, and the model capacity needed in these big, usually English-centered multilingual settings [12]. Analyzing these "in the wild" scale systems has shown key challenges. These include very uneven data distribution across languages, the complicated mix of positive transfer learning and negative interference, the hard job of making good vocabularies for different writing systems, and performance problems because the model capacity is limited. It's worth noting that deeper Transformer models have been found to be better at reducing interference than wider ones [13]. To make English-centered translation better, researchers are working on new data mining strategies (like using "bridge languages") and combining dense scaling with language-specific sparse parameters in the Transformer framework. These efforts are to make true many-to-many translation and make the direct translation quality between non-English language pairs a lot better [14].

## 4. Challenges and future directions for transformer in text generation

### 4.1. Current challenges for transformer in text generation

Even though the Transformer architecture is the base of modern text generation, its wide use has brought a few built-in technical problems. First, the core self-attention mechanism needs more computing time and memory as the sequence length goes up by a lot. This feature makes it way less efficient when dealing with really long sequences. This limit stops it from being used directly in tasks like summarizing a lot of documents or translating at the document level. These limits have made people look for better ways, like using sparse attention, linear attention methods (like Longformer), and efficient execution frameworks (like Flash attention). Also, training and using the really big Transformer architectures takes a lot of computing power and energy. This costs a lot of money and makes people worried about its effect on the environment (like carbon emissions and water use), so making the models more efficient has become really important for research.

At the same time, the complicated ways these models work make them like a "black box" that's hard to understand. It's hard to say why the model makes a certain text or how it makes its predictions. This makes it hard to trust the model, makes debugging hard, and stops it from being used for sure.

When it comes to text generation, the usual autoregressive training methods have what's called exposure bias. This means the data distribution in the training stage (using real tokens) is different from the one in the inference stage (depending on the tokens the model makes by itself in the earlier

DOI: 10.54254/2755-2721/2025.PO24517

steps). As the generation goes on, this can cause mistakes to add up. In the end, even though we know there's a link between how big the model is and how well it works, the huge size of Transformer models and all their parameters cause problems. These problems are about using these parameters well and making the model easy to change, especially when fine-tuning. This has made people come up with ways to fine-tune the models better with less parameters (like LoRA and prefix tuning). The goal is to make the model work better but change only a small part of it, so it can be changed with not many resources. All these technical problems are at the front of Transformer research now and make people keep coming up with new ideas for the design, training, and making them more efficient [15].

### 4.2. Controllability and quality assessment of generated text

Figuring out the quality of the text that's made and how much control we have over its features, especially in controllable text generation (CTG), is a very important but hard job. The key to this assessment is two main things: how true the generated content is to the control things we set (like safety rules, feelings, topics, style), and how well the system keeps the basic text things (like being smooth, making sense, being different, being useful), even when it follows the control rules. A really good CTG system can balance these two well [16].

The ways to evaluate can be divided into ones that are automatic and ones that are led by people. Automatic evaluation depends on things that can be measured. For example, using classifiers to see if it follows the feeling or topic rules, and using things like perplexity to see how well the language flows. But these computer-based measures often can't catch all the small and personal things about natural language. So, having people evaluate is still really important. These reviewers look at the text output based on things like how controllable it is, how smooth it is, how relevant it is, how consistent it is, and the overall quality. Recently, evaluation strategies that use big language models (LLMs) have come up. This means using these big language models as evaluators to try and balance the big scale we can do with automatic measures and the detailed look of human evaluations.

### 4.3. Future prospects

The important chances for future development are in the natural language generation methods based on transformers. Even though the existing models have had great success in a lot of applications, there are still big problems when it comes to manageability, being consistent over a long time, and making the generated text make sense. The main goal of the research that's going on right now has to be to go for really good technology to find and use the context information that's in long talks or a lot of documents. Also, integrating strategies like reinforcement learning or adversarial training frameworks in a systematic way has the potential to make these models even better, especially when it comes to making different features better, like the depth of information in the generated content or the range of how different it can be.

The new areas for the research that comes next also include special text generation and tasks that need to put together multimodal information flows – for example, making text descriptions based on the visual parts of charts or images – which shows a productive direction for academic research. These types of apps need models that have two functions: not just understanding text but also being good at putting together and connecting data from a lot of sources. While going for these goals, getting over the problems related to data is still something that needs immediate and serious attention. Building strong and diverse datasets with better deep semantic information is a must for making model training methods better and making them work for new inputs. From the point of

DOI: 10.54254/2755-2721/2025.PO24517

view of Hoque and others [17], a bunch of possible dangers – like twisting the facts, deep-seated biases, and spreading misleading stuff through text that's made by algorithms – need to be looked at carefully in future academic studies, and active safety measures should be made. This effort basically shows the deep ethical responsibility that comes with using these powerful technologies.

## 5. Conclusion

This paper gives an analysis of the Transformer architecture model, talks about how it's changed and makes clear the self-attention mechanism, which makes natural language generation a lot better by dealing with long-range context information well. It looks at how the Transformer model is used in key text generation areas in a systematic way, especially long creative writing, automatic code generation, and multilingual translation, and points out the big successes and the ongoing challenges. The current analysis shows that although the models based on transformers have been really effective, there are still some limits that need to be dealt with.

The modern Transformer models have big problems when it comes to text generation in several areas. First, the need for a lot of computing resources for the self-attention mechanism limits how well it works for long sequences. Second, the high costs of training and using large-scale architectures are made worse by their "black box" nature that's not easy to understand, which makes it hard to explain. Third, autoregressive generation techniques have exposure bias. Finally, there are big technical challenges when it comes to making sure the generated text is under control and stable and being able to evaluate the quality of the content reliably.

Future research should focus on making transformer variants that don't need a lot of computing and new training examples. Making the generated content more under control, more accurate, and more consistent with the context, especially in complex and long stories, is an important area of research. Also, we can expect progress in multimodal content generation, making personalized text, and developing more exact and reliable ways to evaluate. Future research should also put ethical issues first, like making the model biases less and stopping bad uses, to help with responsible innovation in transformer-based text generation technologies.

## References

[1] Chai, Y., Jin, L., Feng, S., & Xin, Z. (2024). Evolution and advancements in deep learning models for natural language processing. Applied and Computational Engineering, 77, 144-149.
[2] Vaswani, A., Shazeer, N., et al. (2017). Attention is all you need. Advances in neural information processing systems, 30.
[3] Patwardhan, N., Marrone, S., & Sansone, C. (2023). Transformers in the real world: A survey on NLP applications. Information, 14(4), 242.
[4] Gillioz, A., Casas, J., Mugellini, E., & Abou Khaled, O. (2020). Overview of the Transformer-based Models for NLP Tasks. In 2020 15th Conference on computer science and information systems (FedCSIS). IEEE, pp. 179-183.
[5] Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., & Wang, J. (2018). Long text generation via adversarial training with leaked information. In Proceedings of the AAAI conference on artificial intelligence. Vol. 32, No. 1.
[6] Marco, G., Gonzalo, J., & Rello, L. Transformers Can Outperform Humans in Short Creative Writing Tasks. Available at SSRN 4673692.
[7] Kusmiatun, A., Efendi, D., et al. (2024, May). The Power of AI in Generating Novel Based and Impactful Character Development for Fiction Story. In 2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). IEEE, pp. 1555-1561.
[8] Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., ... & Vinyals, O. (2022). Competition-level code generation with alphacode. Science, 378(6624): 1092-1097.
[9] Svyatkovskiy, A., Zhao, Y., Fu, S., Sundaresan, N. (2019, July). Pythia: Ai-assisted code completion system. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 2727-

DOI: 10.54254/2755-2721/2025.PO24517

2735.
[10] Svyatkovskiy, A., Deng, S. K., Fu, S., & Sundaresan, N. (2020, November). Intellicode compose: Code generation using transformer. In Proceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pp. 1433-1443.
[11] Johnson, M., Schuster, M., et al. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. Transactions of the Association for Computational Linguistics, 5, 339-351.
[12] Aharoni, R., Johnson, M., & Firat, O. (2019). Massively multilingual neural machine translation. arXiv preprint arXiv: 1903.00089.
[13] Arivazhagan, N., Bapna, A., Firat, O., Lepikhin, D., Johnson, M., Krikun, M., ... & Wu, Y. (2019). Massively multilingual neural machine translation in the wild: Findings and challenges. arXiv preprint arXiv: 1907.05019.
[14] Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., ... & Joulin, A. (2021). Beyond english-centric multilingual machine translation. Journal of Machine Learning Research, 22(107): 1-48.
[15] Becker, J., Wahle, J. P., Gipp, B., & Ruas, T. (2024). Text generation: A systematic literature review of tasks, evaluation, and challenges. arXiv preprint arXiv: 2405.15604.
[16] Liang, X., Wang, H., Wang, Y., Song, S., Yang, J., Niu, S., ... & Li, Z. (2024). Controllable text generation for large language models: A survey. arXiv preprint arXiv: 2408.12599.
[17] Hoque, E., & Islam, M. S. (2025, February). Natural language generation for visualizations: State of the art, challenges and future directions. In Computer Graphics Forum, Vol. 44, No. 1, pp. e15266.