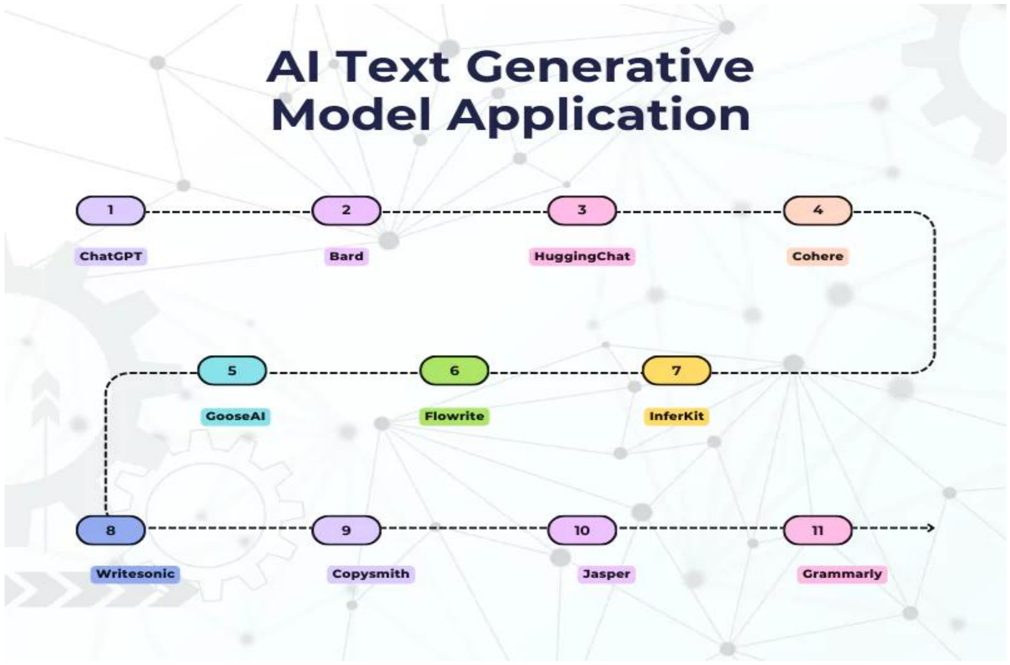
Text Generation

1. Introduction to Text Generation

Text generation is a core task in Natural Language Processing (NLP) that focuses on automatically producing human-like text using computational models. The goal is to generate coherent, contextually relevant, and grammatically correct sentences or paragraphs based on a given input or prompt. Text generation plays a crucial role in modern AI applications such as chatbots, virtual assistants, machine translation, content creation, summarization, and storytelling.

Historically, text generation began with simple rule-based systems, where predefined grammar rules and templates were used to generate sentences. As data availability and computational power increased, statistical and neural approaches became dominant, allowing machines to learn language patterns directly from data.



2. Importance and Applications of Text Generation

Text generation has become essential due to its wide range of real-world applications:

• **Chatbots and Virtual Assistants** – Generate natural responses in conversations.

• **Content Creation** – Automatically create articles, blogs, marketing copy, and reports.

• **Machine Translation** – Generate translated text in a target language.

• **Text Summarization** – Produce concise summaries from long documents.

• **Story and Poetry Generation** – Creative writing assistance.

• **Code Generation** – Automatically generate programming code from natural language.

These applications save time, improve productivity, and enhance user experience.

3. Early Approaches to Text Generation

3.1 Rule-Based Text Generation

Rule-based systems use handcrafted grammar rules, templates, and dictionaries to generate text. For example: "Hello, {name}. Welcome to {place}."

Advantages:

• High control over output

• Grammatically correct

Limitations:

• Not scalable

• Lacks flexibility

• Cannot handle unseen cases

3.2 Statistical Language Models

Statistical models assign probabilities to sequences of words.

N-Gram Models

An n-gram model predicts the next word based on the previous *n-1* words.

Example (Bigram): P("machine" | "learning")

Limitations:

• Data sparsity

• Cannot capture long-term dependencies

4. Neural Network Based Text Generation

The introduction of neural networks revolutionized text generation by enabling models to learn complex language patterns.

4.1 Recurrent Neural Networks (RNNs)

RNNs process text sequentially, maintaining a hidden state that captures previous context.

Advantages: • Handles variable-length input

Limitations:

• Vanishing gradient problem

• Poor long-term dependency handling

4.2 LSTM and GRU

Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) solve RNN limitations by introducing gating mechanisms.

Benefits:

• Capture long-range dependencies

• Improved stability

5. Sequence-to-Sequence (Seq2Seq) Models

Seq2Seq models consist of an encoder and a decoder.

• **Encoder** converts input text into a context vector

• **Decoder** generates output text word by word

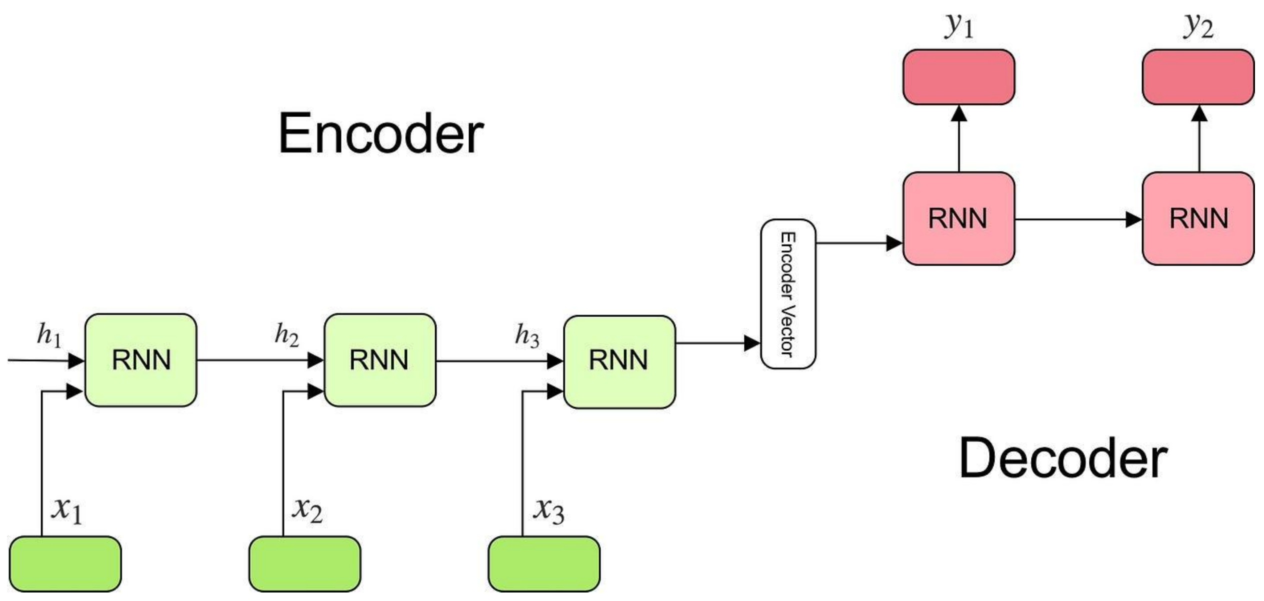
Applications:

• Machine translation

• Chatbots

• Text summarization

Attention mechanisms further improved Seq2Seq models by allowing the decoder to focus on relevant parts of the input.



6. Transformer-Based Text Generation

Transformers replaced RNNs with self-attention mechanisms, enabling parallel processing.

6.1 Self-Attention Mechanism

Self-attention allows models to weigh the importance of different words in a sentence.

Advantages:

• Captures global context

• Faster training

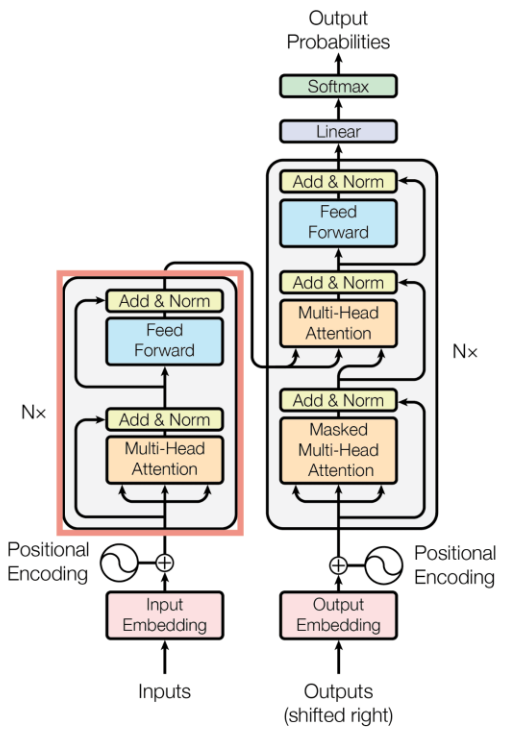
6.2 Pre-trained Language Models

Transformers trained on large corpora can generate highly fluent text.

Key characteristics:

• Context-aware generation

• Transfer learning capability



7. Training Process for Text Generation Models

7.1 Data Collection

Large text corpora such as books, articles, and web data are used.

7.2 Text Preprocessing

• Tokenization

• Lowercasing

• Removing noise

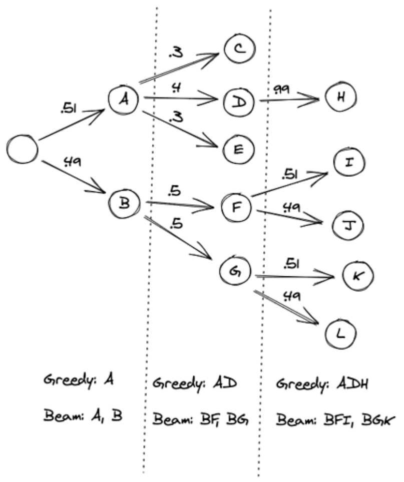
• Vocabulary creation

7.3 Model Training

Models are trained to predict the next token using loss functions like Cross-Entropy Loss.

8. Text Generation Strategies

8.1 Greedy Search



Selects the most probable word at each step.

8.2 Beam Search

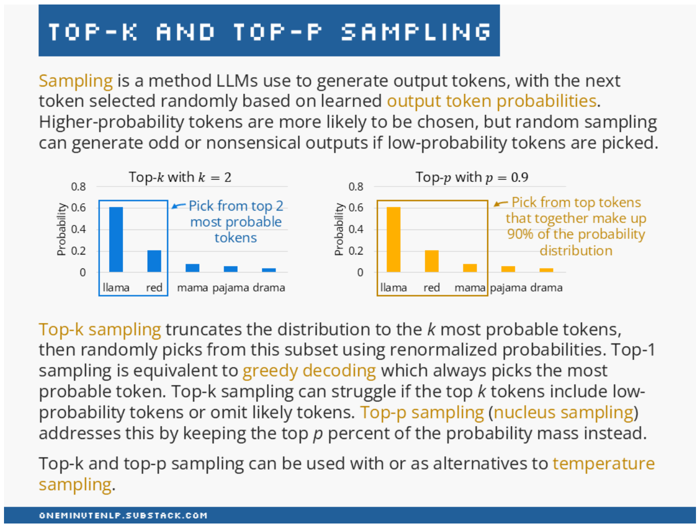
Maintains multiple candidate sequences to improve output quality.

8.3 Sampling Methods

• Top-k Sampling

• Top-p (Nucleus) Sampling

These methods balance creativity and coherence.



TOP-K AND TOP-P SAMPLING

Sampling is a method LLMs use to generate output tokens, with the next token selected randomly based on learned output token probabilities. Higher-probability tokens are more likely to be chosen, but random sampling can generate odd or nonsensical outputs if low-probability tokens are picked.

Top-k with $k = 2$
Probability
0.8
0.6
0.4
0.2
0
Pick from top 2 most probable tokens
llama
red
mama
pajama
drama
Top-p with $p = 0.9$
Pick from top tokens that together make up 90% of the probability distribution

Top-k sampling truncates the distribution to the *k* most probable tokens, then randomly picks from this subset using renormalized probabilities. Top-1 sampling is equivalent to greedy decoding which always picks the most probable token. Top-k sampling can struggle if the top *k* tokens include low-probability tokens or omit likely tokens. Top-p sampling (nucleus sampling) addresses this by keeping the top *p* percent of the probability mass instead.

Top-k and top-p sampling can be used with or as alternatives to temperature sampling.

ONEMINUTENLP.SUBSTACK.COM

9. Evaluation of Text Generation

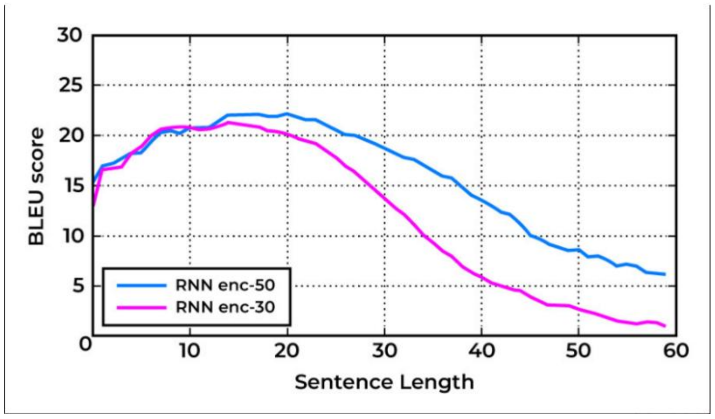
Evaluating generated text is challenging.

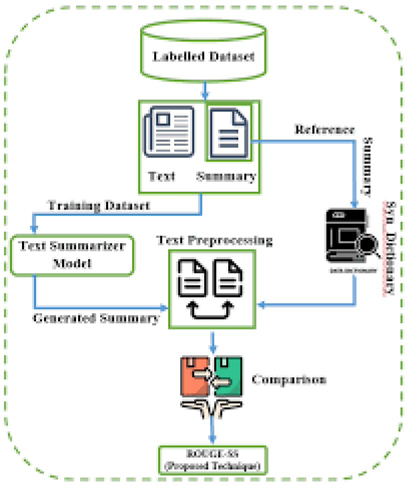
Automatic Metrics

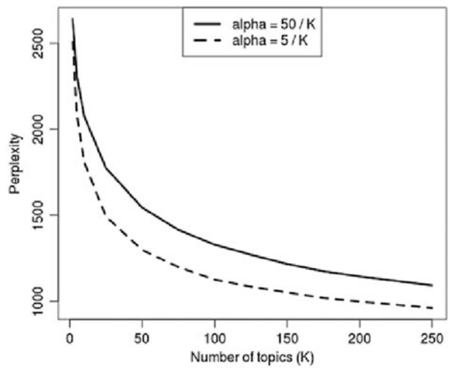
• BLEU

• ROUGE

• Perplexity







Human Evaluation

• Fluency

• Relevance

• Coherence

10. Challenges and Ethical Considerations

Challenges

• Hallucinations

• Bias in generated text

• Lack of factual accuracy

Ethical Issues

• Misinformation

• Plagiarism

• Responsible AI usage

11. Future Directions in Text Generation

• Multimodal generation (text + image + audio)

• Better factual grounding

• More controllable generation

• Efficient and smaller models