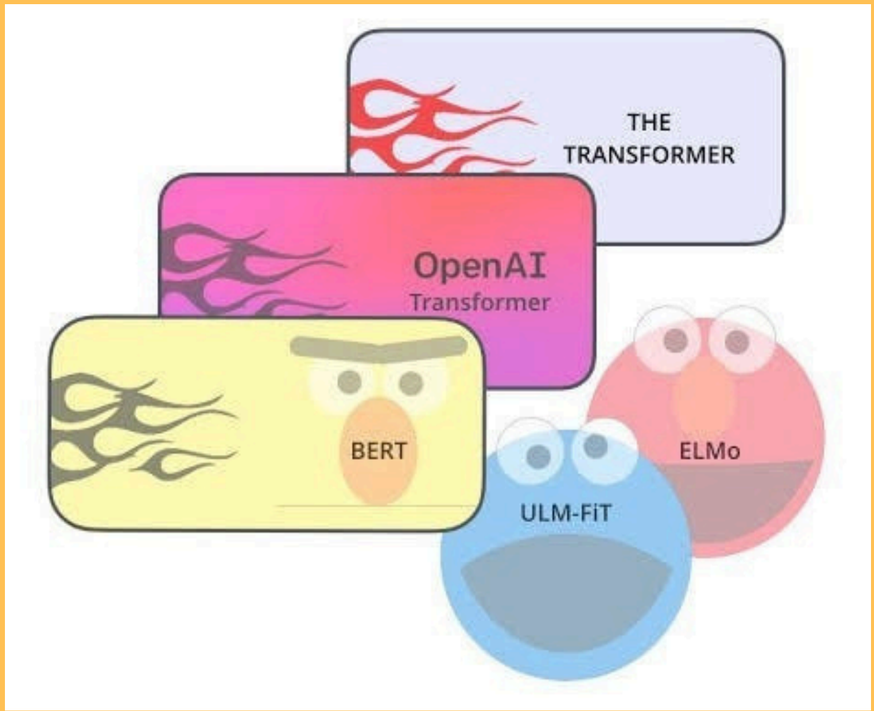
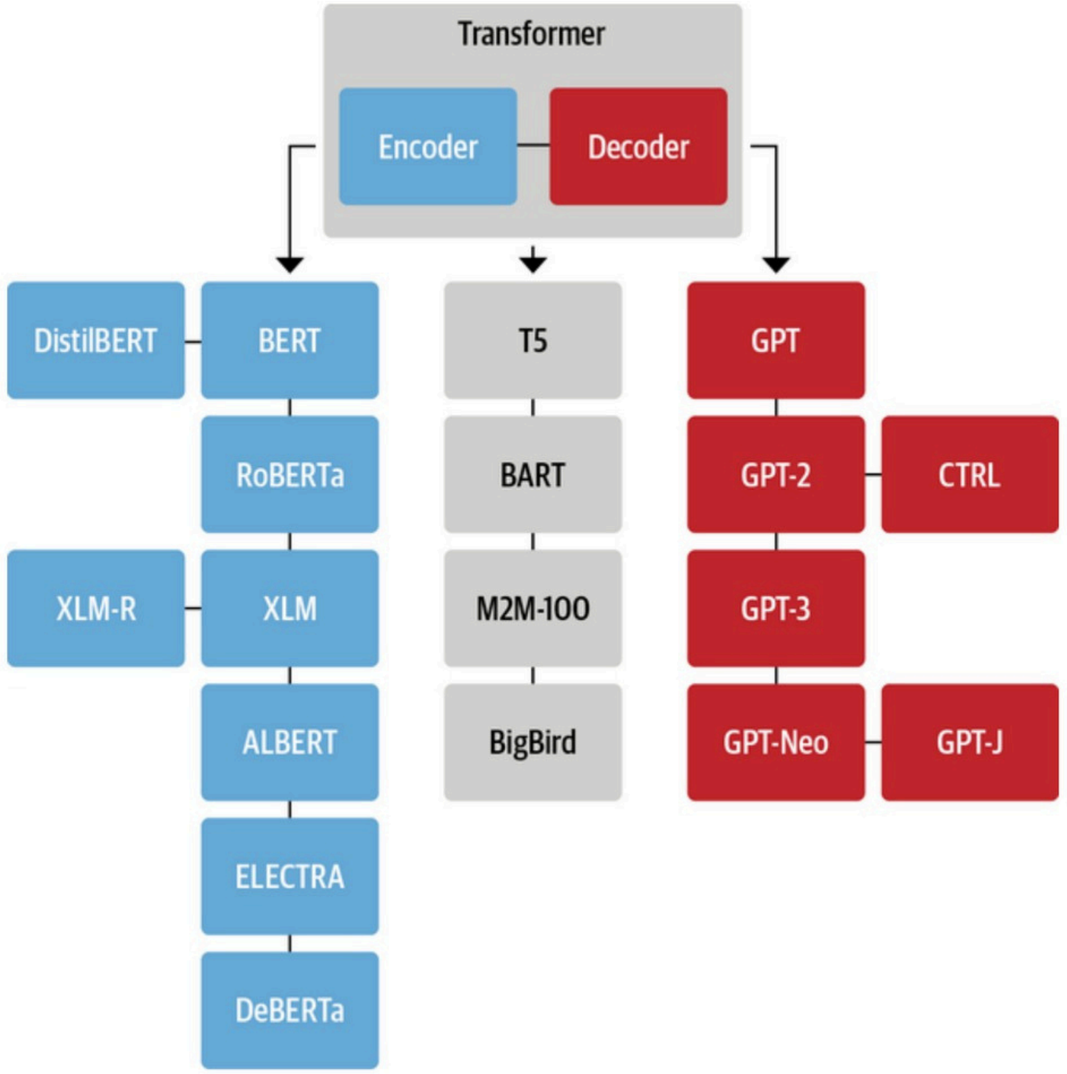
RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

MEET THE TRANSFORMERS



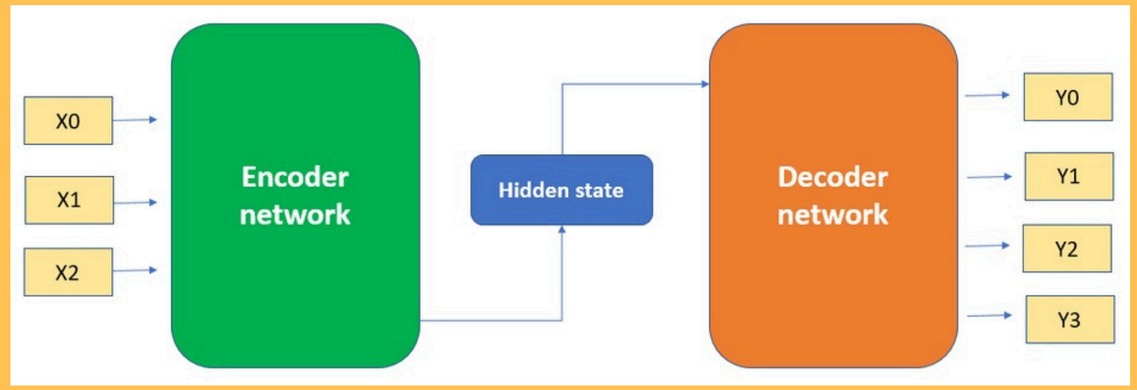
RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE TRANSFORMER TREE OF LIFE



RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE ENCODER BRANCH



- Encoder-only models are mainly used for understanding text, known as Natural Language Understanding (NLU).
- These models focus only on processing input text, without generating output sequences.
- Tasks like question answering, text classification, and named entity recognition are typically handled by encoders.
- The first major encoder model was BERT, which led to many advanced variants.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE ENCODER BRANCH

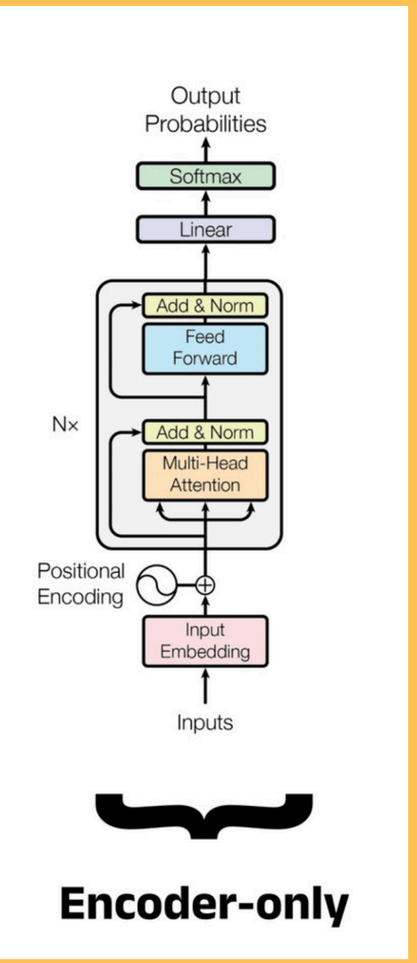
BERT – The First Encoder Model

- BERT introduced the idea of learning bidirectional context by masking words in the input and predicting them.
- It uses two main training tasks: masked language modeling (MLM) and next sentence prediction (NSP).
- BERT became a breakthrough model, achieving top performance on many language understanding tasks.

MLM: Masking some words and predicting them.
NSP: Predicting if one sentence follows another.

DistilBERT – A Smaller BERT

- DistilBERT is a lightweight version of BERT that retains most of its performance while being smaller and faster.
- It is trained using a technique called knowledge distillation, where it learns from a larger teacher model (BERT).
- DistilBERT is suitable for environments with limited resources and latency requirements.

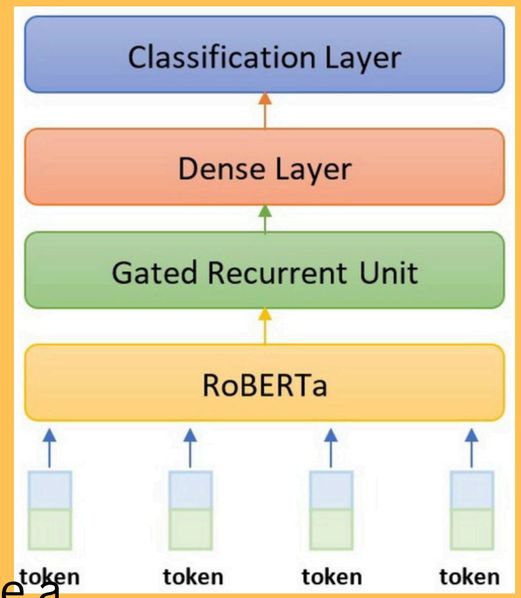


RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE ENCODER BRANCH

RoBERTa

- RoBERTa improves upon BERT by training on more data and longer batches without using the NSP task.
- This modified pretraining strategy leads to better generalization and performance across benchmarks.
- RoBERTa is known for being a strong baseline in many NLP tasks.



XLM and XLM-RoBERTa

- XLM is a multilingual model that supports multiple languages using both MLM and translation-based objectives.
- It introduced Translation Language Modeling (TLM) for learning from multilingual text.
- XLM-RoBERTa scaled up the training data massively using Common Crawl to create a 2.5 TB multilingual dataset.
- XLM-RoBERTa removed the translation-based loss but achieved better performance, especially for low-resource languages.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE ENCODER BRANCH

ALBERT

- ALBERT reduces memory usage by separating embedding size from hidden size, and by sharing parameters across l aye r s.
- It replaces the NSP task with a sentence order prediction task that checks if sentences are in the correct order.
- These changes help ALBERT perform well while using fewer parameters than BERT.

ELECTRA

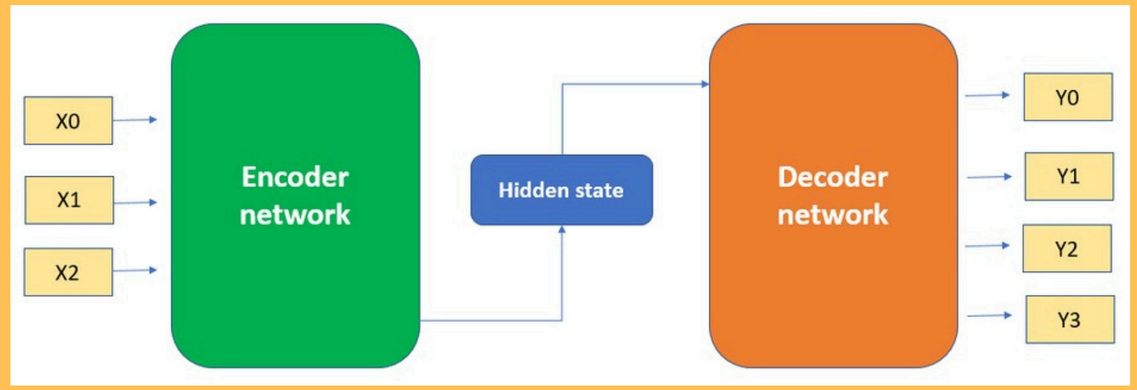
- ELECTRA introduces a more efficient training method where a generator replaces some words, and a discriminator identifies them.
- Unlike BERT, which updates only masked tokens, ELECTRA trains on all tokens, making it 30 times more efficient.
- The discriminator model is used for downstream tasks, just like BERT.

DeBERTa

- DeBERTa represents each word using two vectors—one for meaning and one for relative position in the sentence.
- This separation helps the model understand the relationship between nearby words more clearly.
- DeBERTa also adds absolute positions before the final layer to help in decoding tasks.
- It is the first model to beat human performance on the SuperGLUE benchmark.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE DECODER BRANCH



- Decoder-only models focus on generating text, and are trained to predict the next word based on previous words.
- These models are widely used in text generation, storytelling, summarization, and even coding tasks.
- OpenAI has led much of the development in this area, starting with GPT.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

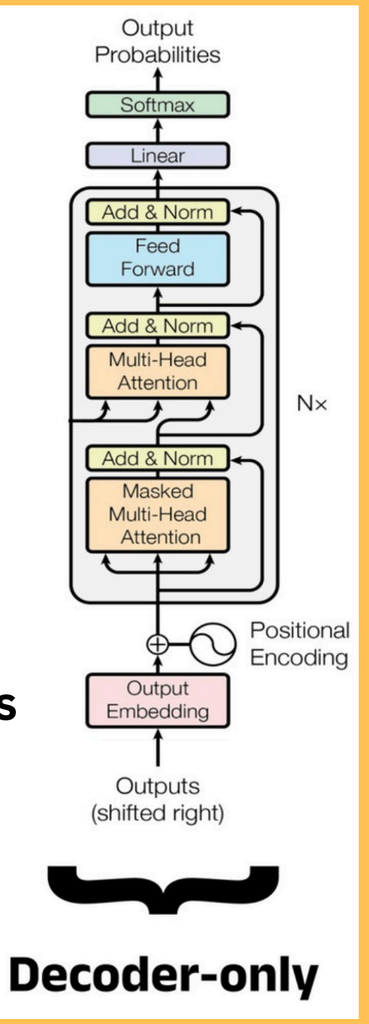
THE DECODER BRANCH

GPT and GPT-2

- GPT introduced a powerful method of pretraining a decoder using a simple objective: predicting the next word.
- GPT-2 scaled this approach with a larger dataset and model size to generate coherent and realistic text.
- Due to potential misuse, GPT-2 was released gradually, starting with smaller versions.

CTRL and GPT-3

- CTRL allows users to control the style or topic of generated text using special tokens at the beginning of input.
- GPT-3 scaled up the GPT-2 model to 175 billion parameters, leading to massive improvements in quality and learning.
- GPT-3 can perform tasks from just a few examples, showing few-shot learning ability.

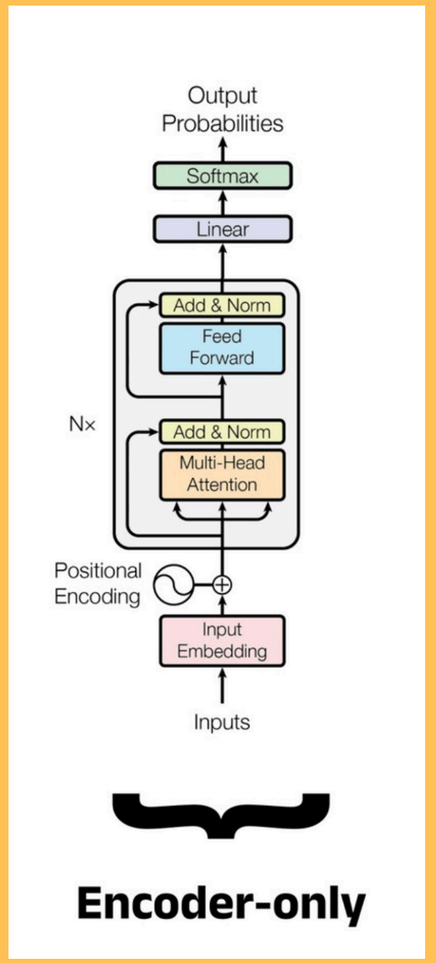


RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE DECODER BRANCH

GPT-Neo and GPT-J-6B

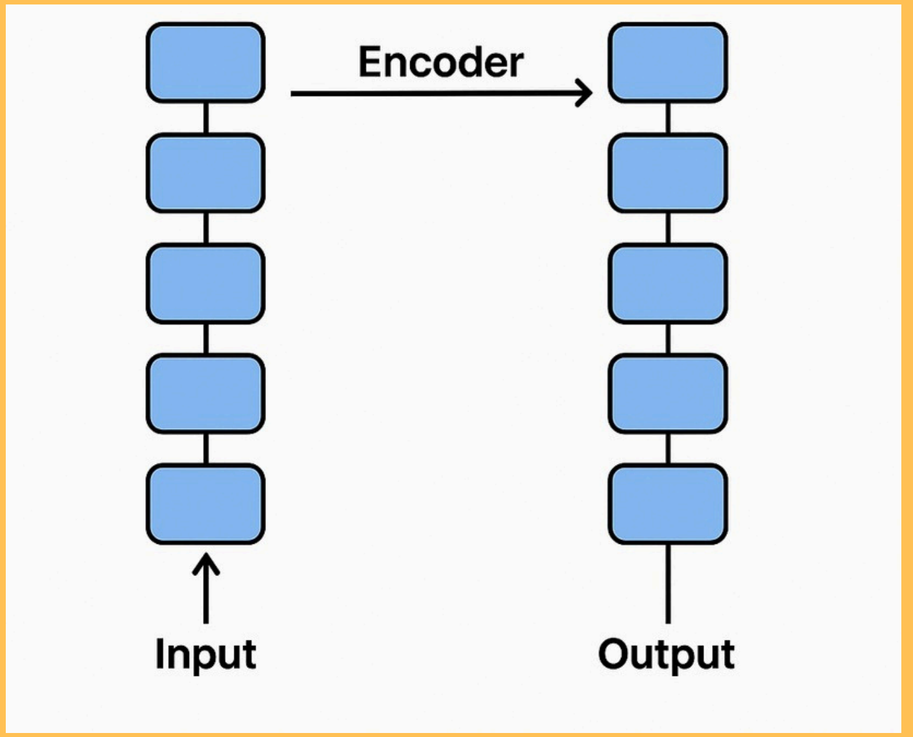
- GPT-Neo and GPT-J are open-source alternatives to GPT-3 developed by the EleutherAI community.
- These models are smaller (up to 6 billion parameters) but competitive with smaller GPT-3 versions.
- They make powerful generative models accessible to the public and researchers.



RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE ENCODER–DECODER BRANCH

- Encoder-decoder models use two stacks: one to understand input and another to generate output. These models are well-suited for translation,
- summarization, and tasks involving both input and output text.
  They combine the benefits of both encoders and decoders.



RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

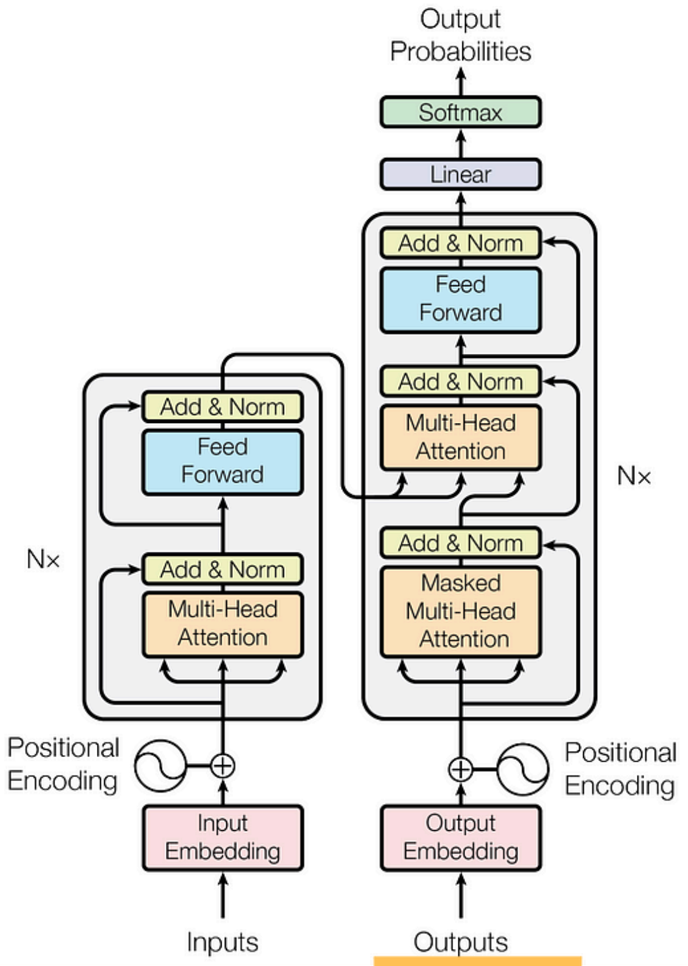
THE ENCODER–DECODER BRANCH

BERT

Encoder

GPT

Decoder



RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

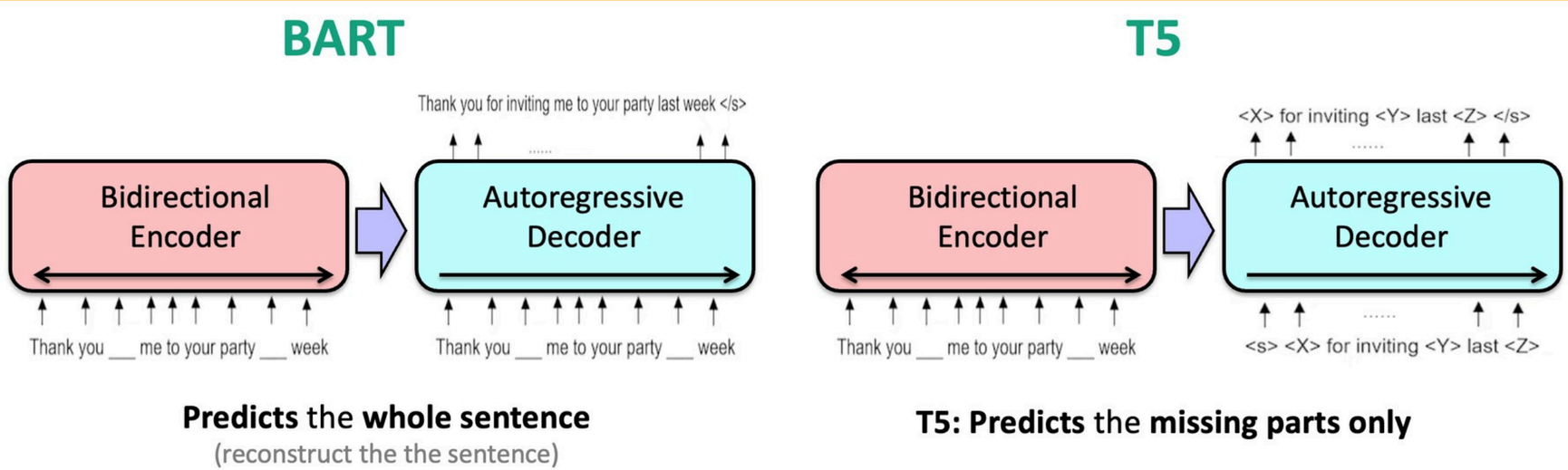
THE ENCODER–DECODER BRANCH

T5

- T5 reformulates all NLP tasks as a text-to-text problem, turning everything into an input-output text format.
- It uses both encoder and decoder components and is trained on a large dataset called C4.
- T5 achieved state-of-the-art performance on several benchmarks, including SuperGLUE.

BART

- BART blends the training ideas of BERT and GPT by corrupting input text and training the model to fix it.
- It uses transformations like masking, sentence shuffling, and token deletion to challenge the model.
- BART is flexible and effective for both understanding and generating text.



RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

THE ENCODER–DECODER BRANCH

M2M-100

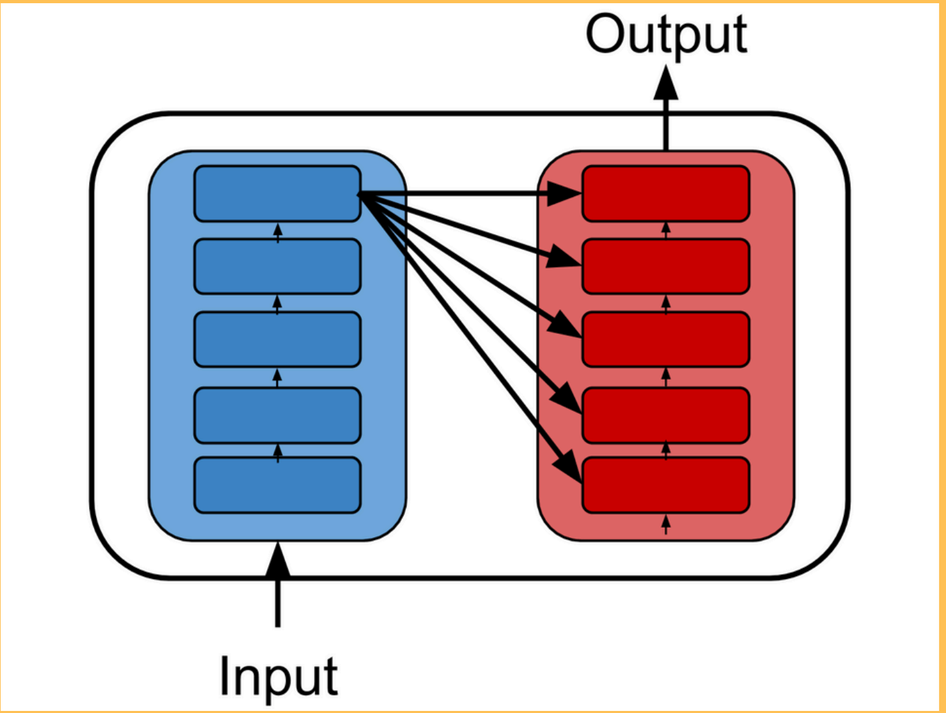
- M2M-100 is the first model that can directly translate between 100 languages without relying on English as a pivot.
- It uses prefix tokens to indicate the source and target languages during translation.
- This allows high-quality translation, especially for underrepresented languages.

BigBird

- BigBird addresses the limitation of short input length in traditional transformers by using sparse attention.
- This allows the model to handle much longer sequences—up to 4096 tokens instead of just 512.
- BigBird is especially useful for tasks like document summarization where long text matters.

CONCLUSION

- Transformer architectures have grown into three major branches: encoders, decoders, and encoder–decoders.
- Each branch has introduced models that improve performance, efficiency, and multilingual support.
- These models are available on the Hugging Face Hub and can be fine-tuned for specific NLP tasks.
- The journey of transformer models continues, pushing the boundaries of what AI can understand and generate.



RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TEXT GENERATION

Text generation refers to the process by which models predict the next word/token based on prior context. It's a sequential prediction task, where the model uses previous tokens to generate the next.

Real-World Applications:

- Chatbots (e.g., ChatGPT)
- Machine Translation (e.g., Google Translate)
- Text Summarization
- Code Generation
- Story Writing

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TEXT GENERATION - CHALLENGES

1. **Coherence:**
   - The logical flow of ideas — the text should be on-topic and make logical sense across sentences.
   - Language models generate one token at a time without true understanding. So if early tokens go off-track, the rest follows.

     Example:

   - **Prompt:** "India is known for"
     - ❌ Incoherent Output: "India is known for elephants is winter happy sky delicious data."
     - ✅ Coherent Output: "India is known for its rich cultural heritage, historical landmarks, and diverse cuisine."

2. **Repetition:**
   - LM might repeat words or phrases unnecessarily, especially in long texts. The model sometimes assigns high probability to the same phrase repeatedly.

     Example:

     - ❌ "The food was good. The food was good. The food was good."
     - ✅ "The food was delightful, especially the desserts, which were exceptionally well made."

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TEXT GENERATION - CHALLENGES

3. Diversity:

- Generate different valid outputs for the same prompt, especially in creative tasks.

Example:

- Prompt: “Tell me a Story”
  - Version 1: “there lived a lonely dragon in a distant cave.”
  - Version 2: “a brave knight set out on an epic journey.”

4. Fluency:

- Proper grammar, syntax, and word usage.
- Model must follow syntactic rules while making semantic sense.

Example:

- ❌ “They go to shop because raining bus.”
- ✅ “They went to the store despite the heavy rain.”

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

GREEDY SEARCH DECODING

Greedy decoding is a method to convert these probability distributions into actual words (tokens).
Here's how it works:

- At timestep 1, the model outputs a list of probabilities for all words in the vocabulary.
- Greedy decoding picks the word (token) with the highest probability.
- This chosen token is then fed back into the model to predict the next word.
- This process repeats until an end-of-sequence token is generated or a max length is reached.

Equation:

$$\hat{y}_t = \underset{y_t}{\operatorname{argmax}}\ P\left(y_t | y_{<t}, \mathbf{x}\right).$$

$\hat{y}_t$: This is the **predicted word (token)** at time step t.

$\arg\max_{y_t}$: This means **we are choosing the value of** y_t (the next word) that **gives the highest probability**.

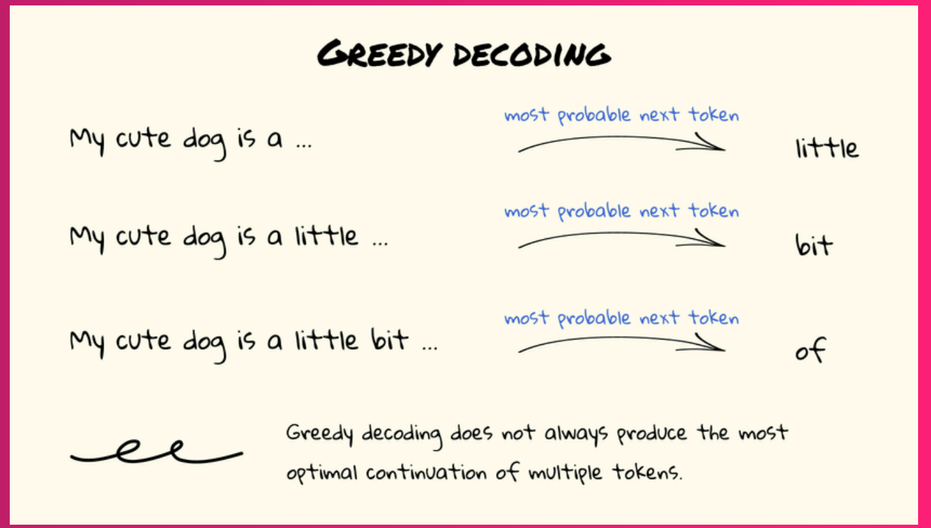
$P(y_t \mid y_{<t}, \mathbf{x})$: This is the model's probability of choosing the word y_t, given:

- All the previously generated words $y_{<t}$ (i.e., words before step t)
- The input $\mathbf{x}$ (e.g., a question or source sentence)

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

GREEDY SEARCH DECODING

Example : Let's say the model gives these choices for the first word:"I" → 70%,"You" → 20%,"They" → 10%
Greedy decoding picks: "I" (because 70% is the highest)
Next step, it gives:"am" → 80%,"was" → 10%,"will" → 10%
Greedy picks: "am"
Final sentence so far: "I am"



Pros of Greedy Decoding

- Simple to implement
- Fast – only one path is followed, no search needed

Cons of Greedy Decoding

- Can get stuck in loops or produce repetitive outputs Not
- NOT optimal

Alternatives to Greedy Decoding

- Beam Search – explores multiple possible sequences simultaneously
- Top-k Sampling – randomly selects from the top k tokens
- Top-p (nucleus) Sampling – samples from the smallest set of words whose total probability exceeds p

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

BEAM SEARCH DECODING

- Beam search is a smarter version of greedy decoding.
- Instead of picking only the most probable word at each step (like greedy decoding), it keeps track of the top b options (called beams) and explores multiple possible sequences.
- Beam Search keeps the best few sentence paths at each step, helping models choose the most likely full sentence instead of just the next best word.

Beam Search Step-by-Step:

1. Start with input: "I am"
2. The model gives probabilities for next tokens: "a", "not", "very", etc.
3. Instead of choosing only the best (like greedy), beam search keeps the top b options.
4. For each of these, get next-step predictions, and again keep top b sequences overall.
5. Repeat until end of sentence (max length or EOS token).

Method	Pros	Cons
Greedy	Fast, simple	May miss better sequences
Beam Search	Better results (usually)	Slower, can be repetitive, memory use ↑

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

BEAM SEARCH DECODING

Equation:

$$\log P(y_1, ...y_t|\mathbf{x}) = \sum_{t=1}^{N} \log P(y_t|y_{<t}, \mathbf{x}).$$

The total log-probability of generating a sequence y1,y2,...,yty_1, y_2, ..., y_ty1,y2,...,yt given input x\mathbf{x}x is the sum of the log-probabilities of generating each token yty_tyt, one at a time, given all the previous tokens y<ty_{<t}y<t and the input x\mathbf{x}x.

NOTE :

- **INSTEAD OF USING RAW PROBABILITIES, WE USUALLY USE LOG-PROBABILITIES, WHICH ARE JUST THE LOG (LOGARITHM) OF THOSE PROBABILITIES.**
- **PARTIAL HYPOTHESES : INCOMPLETE SENTENCE FRAGMENTS BEING BUILT**
- **B (BEAM WIDTH) :NUMBER OF POSSIBLE SEQUENCES TO TRACK AT EACH STEP**

Advantages:

- Produces more coherent and globally optimal outputs compared to greedy decoding.
- Reduces the likelihood of premature or suboptimal sequence termination.

Limitations:

- May generate repetitive or generic text.
- Higher beam widths result in increased latency and memory usage.
- Does not incorporate randomness, which limits diversity in generated outputs.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

BEAM SEARCH DECODING

Example:

Beam Search (beam width = 2):

Instead of choosing just one word, Beam Search keeps 2 good choices at every step

How it works:

- Start with "I"
- Check next best 2 words: "love", "eat"
- → Beams: "I love", "I eat"

  For each of these, check next best 2 words again:
- "I love" → "pizza", "cake"
- "I eat" → "pasta", "rice"

  Now we have 4 options:
- "I love pizza"
- "I love cake"
- "I eat pasta"
- "I eat rice"

  Pick the top 2 based on total score.
- Repeat until the sentence ends!

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

WHAT IS SAMPLING IN LANGUAGE MODELS?

Sampling is a technique used to generate text from a language model like GPT.

When a model is generating the next word (token), it produces a probability distribution over the entire vocabulary.
This distribution tells us how likely each word is to come next.

- The model does not always pick the word with the highest probability (that would make text boring and repetitive).
- Instead, we sample from this distribution to introduce some randomness and creativity.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TOP-K SAMPLING

→ **What it is ?**

Only the Top K most probable tokens are considered for generation. All others are ignored, no matter their probability.

→ **Why it's used**

To eliminate low-probability words and focus only on the most likely candidates.

→ **How it works**

1. Sort all possible tokens by their predicted probabilities.
2. Select the top K tokens with highest probability.
3. Normalize their probabilities (so they sum to 1).
4. Randomly select the next word from only these K tokens.

→ **When to Use Top-K Sampling**

Use it when you want a balance between randomness and relevance. It makes the output

- More focused than random sampling
- More creative than greedy (max-probability) decoding

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TOP-K SAMPLING

→ **Limitations and Considerations**

- **Hyperparameter Tuning**
  - Low K → Repetitive but safe output
  - High K → Diverse but riskier text
- **Not Adaptive**
  - K stays fixed across all contexts. It doesn't adjust based on how confident the model is.
- **Example**
  - K = 5 → Only 5 most likely tokens are considered every time, regardless of context.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TOP-K SAMPLING

Assume this is the model's predicted probability distribution for the next word:

TokenProbability.

Example: Top-K Sampling (K = 3)

- **Step 1:** Keep Top 3 highest probability tokens:
  - "cat" (0.30), "dog" (0.25), "mouse" (0.20)
- **Step 2:** Normalize these probabilities:
  - Total = 0.30 + 0.25 + 0.20 = 0.75
  - New probabilities:
    - "cat" = 0.30 / 0.75 ≈ 0.40
    - "dog" = 0.25 / 0.75 ≈ 0.33
    - "mouse" = 0.20 / 0.75 ≈ 0.27

- **Result**: The model randomly selects one of these three based on these normalized probabilities.

Tokens like "car", "tree", "book", and "pizza" are completely ignored.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TOP-P SAMPLING (NUCLEUS SAMPLING)

-> What it is

Instead of picking a fixed number of top tokens (like in Top-K), Top-P selects the smallest number of top tokens whose cumulative probability ≥ p.

-> Why it's used

To make the token selection adaptive to the model's confidence.

If the model is confident, fewer tokens are needed. If not, more tokens are allowed.

-> How it works

1. Sort all tokens by probability (from high to low).
2. Start adding token probabilities until the sum is just greater than or equal to p (e.g., 0.9).
3. Randomly choose the next word from this group.

-> When to Use Top-P Sampling

Use when you want adaptive and context-sensitive generation. Works well in

- Creative writing
- Conversational AI
- Storytelling

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TOP-P SAMPLING (NUCLEUS SAMPLING)

Limitations and Considerations

- **Computational Cost:**
  - Needs sorting + cumulative sum → slightly slower than Top-K.
- **Hyperparameter Sensitivity**
  - Low p (e.g., 0.7) → High randomness
  - High p (e.g., 0.95) → Safer, more predictable text
- **Example**
  - P = 0.9 → Choose the smallest set of top tokens whose combined probability ≥ 90%.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

TOP-P SAMPLING (NUCLEUS SAMPLING)

Example: Top-P Sampling (P = 0.85)

- Step 1: Sort tokens by probability
  - "cat" (0.30), "dog" (0.25), "mouse" (0.20), "car" (0.10), "tree" (0.08)...
- Step 2: Add probabilities until sum ≥ 0.85
  - "cat" + "dog" + "mouse" = 0.30 + 0.25 + 0.20 = 0.75
  - Add "car" (0.10) → total = 0.85
- Selected tokens
  - "cat", "dog", "mouse", "car"
- Step 3: Normalize these 4:
  - Total = 0.85
  - Normalized probabilities:
    - "cat" ≈ 0.30 / 0.85 ≈ 0.35
    - "dog" ≈ 0.25 / 0.85 ≈ 0.29
    - "mouse" ≈ 0.20 / 0.85 ≈ 0.24
    - "car" ≈ 0.10 / 0.85 ≈ 0.12
- Result: The model randomly selects one of these four tokens.
- Unlike Top-K, this method adaptively selected 4 tokens to reach the target probability p = 0.85.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

SUMMARIZATION

- Text Summarization Pipelines
- Summarization Baseline:
  - GPT-2
  - T5
  - BART
  - PEGASUS

NLP FLIPPED CLASS

Done By:
Ankush - IRV22AI008
Rajyalakshmi - IRV22AI041
Safiya - IRV22AI048

INTRODUCTION

What is Summarization:

Summarization is the task of shortening a text document by retaining only the most essential information.

Types of Summarization:

- **Extractive**: Selects important sentences or phrases directly from the source text.
- **Abstractive**: Generates new sentences that convey the core ideas, like how humans summarize.

Use Cases:

News articles, research paper summaries, legal documents, customer service, meeting notes.

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Department of Artificial Intelligence and Machine Learning

TEXT SUMMARIZATION PIPELINE

1. Input Preprocessing:

a. Tokenization
b. Lowercasing
c. Removal of noise (optional)

2. Encoding the Input:

a. Use pretrained language models (transformers)

3. Model Processing:

a. Encoder-only (for extractive)
b. Encoder-decoder (for abstractive)

4. Decoding:

a. Beam search / Greedy / Top-k / Top-p sampling

5. Post-processing:

a. Detokenization
b. Trimming or formatting the output

6. Evaluation:

a. ROUGE, BLEU scores

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

SUMMARIZATION PIPELINE IN TRANSFORMERS

Hugging Face Pipeline Example:

```
from transformers import pipeline
summarizer = pipeline("summarization")
summary = summarizer("Your long text here...", max_length=150,
min_length=40, do_sample=False)
```

- Uses pretrained models like T5, BART, PEGASUS under the hood.
- Abstracts away tokenization, model loading, and decoding.

GPT-2 FOR SUMMARIZATION

- **Type**: Decoder-only Transformer
- Not originally trained for summarization
- **Fine-tuning Required:** Needs supervised training on summarization datasets (e.g., CNN/DailyMail)
- **Approach:**
  - Concatenate prompt + article
  - Model generates the next tokens as the summary
  - Often less accurate than encoder-decoder models for summarization
- **Strengths:** Coherent and fluent text generation
- **Weaknesses:** Tends to generate overly verbose or off-topic summaries

T5 (TEXT-TO-TEXT TRANSFER TRANSFORMER)

- **Architecture**: Encoder-Decoder
- **Google (2019)**: Unified framework where every NLP task is framed as text-to-text
- **Strengths:**
  - Flexible
  - Good at both extractive and abstractive tasks
  - Works well across tasks with transfer learning
- **Limitations:**
  - Large model size
  - Slow inference in real-time scenarios

BART
(BIDIRECTIONAL AND AUTO-REGRESSIVE TRANSFORMERS)

- **Facebook AI (2019)**
- **Architecture:** Encoder-decoder
- **Training Objective:** Corrupted text reconstruction
- **Noise types:** token masking, deletion, sentence permutation
- **Fine-Tuned For:**
  Summarization tasks like CNN/DailyMail
- **Strengths:**
  Strong performance on abstractive summarization
  Robust to noisy input
- **BART vs T5:**
  BART is a denoising autoencoder; T5 uses text-to-text format
  Both are top choices, but BART is slightly better at reconstructive tasks.

PEGASUS

(PRE-TRAINING WITH EXTRACTED GAP-SENTENCES FOR ABSTRACTIVE SUMMARIZATION)

- **Google Research (2020)**
- Specialized for summarization tasks
- **Training Strategy:**
  Mask entire sentences (gap-sentences) that are likely to be summary-worthy
  Model learns to generate those from remaining text
- **Architecture:** Encoder-decoder (like T5/BART)
- **Strengths:**
  Outperforms BART and T5 on summarization benchmarks
  Summaries are closer to human-written ones
- **Limitations:**
  Requires high computational resources
  Not suitable for general NLP tasks (more narrow)

Model Comparison Table

Model	Architecture	Pretraining Objective	Strength	Limitation
GPT-2	Decoder-only	LM objective (next-token)	Fluent generation	Not ideal for summarization
T5	Encoder-Decoder	Text-to-Text with span masking	Flexible, multi-task	Slower inference
BART	Encoder-Decoder	Denoising (reconstruct input)	Strong abstraction	Memory-heavy
PEGASUS	Encoder-Decoder	Gap-sentence prediction	Summarization-specific	Needs powerful hardware

EVALUATION METRICS

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**
  ROUGE-1: unigram overlap
  ROUGE-2: bigram overlap
  ROUGE-L: longest common subsequence

- **BLEU (mainly for translation)**
  Less commonly used in summarization

- **Human Evaluation:**
  Coherence, fluency, relevance, and grammaticality

CONCLUSION

- Summarization is a key task in NLP with many real-world applications
- Transformer models like T5, BART, and PEGASUS have significantly improved abstractive summarization
- Choice of model depends on use-case:

  GPT-2: general generation

  T5: flexible multi-task

  BART: robust, denoising

  PEGASUS: high-quality summaries