Encoding-Decoding in NLP - what is it all about?

Microsoft Bing : Generated and published : May 23, 2025

An advanced large language model tells me that in machine learning and language modeling in the context of encoding and decoding of written text, there are three common architectures that are used:

the encoder-only method, the decoder-only method, and the encoder-decoder method.

According to you, is it correct, what the other LLM tells me?

Yes, that description is largely correct. In the field of NLP and machine learning, especially with transformer-based architectures, we typically group models into these three categories:

- 1. **Encoder-Only Models:** These models focus exclusively on processing the input text to generate rich representations. They're often employed in tasks such as sentiment analysis, text classification, and named entity recognition. A prime example is BERT, which uses a bidirectional encoder to understand context from both left and right sides of a token.
- 2. Decoder-Only Models: In this architecture, the model generates text in an auto-regressive manner—that is, it predicts the next token based on the tokens that came before it. This approach is ideal for tasks like text generation and completion. GPT models typify this design, where the sole focus is on producing coherent and contextually appropriate text sequences.
- 3. Encoder-Decoder Models: Also known as sequence-to-sequence models, these architectures use an encoder to process and summarize the input sequence and a decoder to generate the output sequence. They are especially valuable in tasks that require transformation from one representation to another, such as machine translation, text summarization, and even certain forms of question answering. The original Transformer model by Vaswani et al. is a classic example that introduced this paradigm.

Each of these architectures has its own strengths and is chosen based on the nature of the task at hand. For example, encoder-only models excel at understanding and representing input text, decoder-only models shine in generative tasks, and encoder-decoder models are ideal for tasks requiring a transformation from input to output.

This taxonomy has served as a helpful guide in the design and application of NLP models, and while there are hybrid approaches and continual innovations that sometimes blur the lines between these categories, the fundamental distinction remains a solid way to understand how these models function and where they perform best.

Curious to dive deeper? We could explore how training objectives differ among these architectures or look into recent innovations that blend these paradigms for more advanced tasks.



To clarify this further, please briefly explain the direct advantage and obvious disadvantage of the:

- encoder-only method compared to the decoder-only method and the encoder-decoder method, then of the
- decoder-only method compared to the encoder-only method and the encoder-decoder method, then of the
- encoder-decoder method compared to the encoder-only method and then of the
- encoder-decoder method compared to the decoder-only method.

Below is a concise comparison highlighting the main advantage and disadvantage for each method when contrasted with the others:

1. **Encoder-only Method**

- **Advantage compared to Decoder-only and Encoder-Decoder:**

Encoder-only models process input bidirectionally, meaning they can capture the full context from both directions across the entire sequence. This results in deep, well-rounded representations, making them highly effective for interpretation tasks such as text classification, sentiment analysis, or named entity recognition.

- **Disadvantage compared to Decoder-only and Encoder-Decoder:**

The architecture lacks a built-in generative mechanism. Since there is no explicit decoder component, encoder-only models are inherently unsuited for any task requiring text generation, such as machine translation or creative story writing.

2. **Decoder-only Method**

- **Advantage compared to Encoder-only and Encoder-Decoder:**

Decoder-only models excel at autoregressive generation. By focusing solely on generating text token by token (and using left-context), they are optimized for tasks like language modeling, text completion, and any free-form generation where sequential consistency and creativity are prioritized.

- **Disadvantage compared to Encoder-only and Encoder-Decoder:**

Their reliance on a unidirectional (left-to-right) context makes them less effective at fully understanding or representing the overall context of long input texts. This can be a significant limitation for tasks where a comprehensive grasp of the input (especially from both directions) is critical, or where a clear separation between comprehension and generation (as in sequence-to-sequence tasks) is needed.

3. **Encoder-Decoder Method Compared to Encoder-only**

- **Advantage:**

The encoder-decoder architecture is purposefully designed for sequence-to-sequence tasks (e.g., translation or summarization). The encoder is dedicated to building a thorough representation of the entire input, while the decoder transforms this representation into the desired output. This clear division enables improved handling of tasks where the output must be directly tied to a given input.

- **Disadvantage:**

The additional decoder brings extra complexity and computational overhead. In scenarios where the task only requires deep input understanding (without the need for generating a sequence), a dedicated encoder-only model is more efficient and simpler to train and deploy.

4. **Encoder-Decoder Method Compared to Decoder-only**

- **Advantage:**

By decoupling the input processing (encoder) from the text generation (decoder), the encoderdecoder architecture can more effectively align and transform inputs into outputs. This makes it particularly adept at tasks where precise mapping between source and target is crucial, such as in machine translation or text summarization, leading to controlled and contextually accurate generation.

- **Disadvantage:**

The separated components increase the model's complexity and resource demands. For tasks centered purely on free-form text generation where the entire focus is on predicting the next token (with a relatively straightforward context), the unified nature of decoder-only models can be more computationally efficient and easier to use.

Resume: Each architecture carries its intrinsic strengths and trade-offs, and the choice among them largely depends on the specific demands of the task at hand—whether you need deep input contextualization, efficient text generation, or a well-coordinated transformation between input and output sequences.

Would you like to explore how recent model innovations blend these paradigms or dive into examples of tasks best suited for each approach?



b Let's break down the task fit for each architecture:

1. **Encoder-Only Models**

Best Suited For:

- **Understanding and Representation Tasks:** These models shine when the primary challenge is to deeply understand a text. They're perfect for tasks that require robust comprehension rather than generation.

- **Examples:**

- **Text Classification:** Assigning categories like sentiment or topic.
- **Named Entity Recognition (NER):** Identifying proper nouns and other key elements in text.
- **Extractive Question Answering:** Spotting precise answers directly from a passage.

Why They Work Well:

Being bidirectional (as in models like BERT), they examine the entire context around a word, which leads to richer representations for understanding tasks.

2. **Decoder-Only Models**

Best Suited For:

- **Generative Tasks:** These models are adept at creating text one token at a time in an autoregressive manner. They're the go-to choice when the challenge is about generating fluent, coherent, and sometimes creative text.

- **Examples:**

- **Language Modeling:** Predicting the next word in a sequence (core to models like GPT).

- **Text Completion and Story Generation:** Where the model continues or expands upon a provided prompt.

- **Dialogue Systems:** Offering conversational, context-aware responses in a chat setting.

Why They Work Well:

Their design is optimized for sequential generation, maintaining consistency and natural flow, although they may struggle with capturing long-range contextual dependencies in a bidirectional manner.

3. **Encoder-Decoder Models**

Best Suited For:

- **Transformation/Sequence-to-Sequence Tasks:** These architectures are tailor-made when the task demands converting one sequence into another. By separating the understanding (encoder) and generation (decoder) phases, they offer strong alignment between input and output.

- **Examples:**

- **Machine Translation:** Converting text from one language to another.
- **Summarization:** Distilling long texts into concise summaries.
- **Paraphrasing/Style Transfer:** Rewriting text while preserving its essential meaning.

- **Conditional Text Generation:** Where output is directly influenced by a given input sequence (e.g., transforming a question into an answer).

Why They Work Well:

The architecture allows for a detailed understanding of the input before beginning generation. This separation is crucial for tasks that require both a deep grasp of the source and a finely tuned output that accurately reflects it.

Resume: Each architecture offers distinct trade-offs. Encoder-only models prioritize comprehension, decoder-only models emphasize fluid generation, and encoder-decoder models balance both for tasks that need a precise transformation of input to output.

If you're curious about how some of these models are adapted in hybrid setups or the latest research in blending these paradigms, we can dive into recent innovations next!

Thank you. This may all seem a little theoretical, however.

What I'd like to know is how are these architectures used in real-world applications?

b Let's explore how these architectures translate into real-world applications:

Encoder-Only Models in Real-World Settings

Applications:

Encoder-only models, such as BERT, are primarily leveraged for tasks that require deep language understanding rather than generation. In the industry, these models are widely used for:

- **Text Classification and Sentiment Analysis:** Helping companies analyze customer feedback or social media posts to determine sentiment, detect spam, or automatically categorize content.

- **Named Entity Recognition (NER) and Information Extraction:** Extracting key details from documents—for instance, pulling out names, dates, or locations from legal or medical records.

- **Search and Ranking:** Enhancing search engines by understanding query context, leading to more relevant results. For example, Google integrated encoder-based models to better understand the intent behind search queries.

Benefit in Use:

Their ability to process text in a bidirectional manner provides a nuanced understanding of language, making them exceptionally good at tasks where context and semantic relationships are critical.

Decoder-Only Models in Real-World Settings

Applications:

Decoder-only models, like those used in the GPT series, are built for generating text. They come into play when creating fluent and coherent text is the primary goal:

- **Conversational Agents and Chatbots:** Many modern chatbots rely on decoder-only architectures to generate real-time responses in customer service interactions or virtual assistants.

- **Content Generation and Storytelling:** They are often seen in applications generating creative content such as news articles, marketing copy, or even fictional stories.

- **Code Completion/Assistance:** Developer tools, like GitHub Copilot, use these models to predict and suggest code snippets, enhancing programming productivity.

- **Text Completion and Auto-suggestions:** From email drafting to online writing assistants, these models help users complete sentences or ideas based on a given prompt.

Benefit in Use:

Their autoregressive nature **allows these models to generate extended, coherent sequences**, making them adept at keeping the narrative consistent and creative.

Encoder-Decoder Models in Real-World Settings

Applications:

The encoder-decoder setup is particularly valued when the task involves transforming one sequence into another. This paradigm is central to:

- **Machine Translation:** Services like Google Translate use these models for converting text from one language to another, maintaining alignment and context from source to target.

- **Summarization and Paraphrasing:** Whether summarizing lengthy documents into concise briefs or rephrasing content, encoder-decoder models provide the architecture necessary to map and transform input into a coherent output.

- **Speech Recognition and Captioning:** In systems that convert spoken language into text or generate captions for images and videos, an encoder-decoder framework effectively captures the input (audio or visual signal) and produces an accessible textual output.

- **Question Answering Systems:** Certain QA systems use the encoder to understand the context of a query and the decoder to generate a specific, tailored answer.

Benefit in Use:

The clear separation between understanding (via the encoder) and output generation (via the decoder) makes these architectures especially powerful for tasks that require mapping or transformation of input data to a different output modality.

Resume: Each of these architectures is optimized for different types of problems. Encoder-only models excel where deep interpretation is essential, decoder-only models lead in creative and sequential generation, and encoder-decoder models strike a balance for tasks needing a refined transformation from input to output.

Would you be interested in examples of how these models enhance customer experiences or the latest innovations that blend these paradigms for more complex applications?

Yes, I would be interested in examples of how these models enhance customer experiences or the latest innovations, blending these paradigms for more complex applications. A diagram would be useful here, would it not. So, could you specify this by means of a diagram?







- Encoder-Only Models: These models are engineered to deeply understand and represent input text. They are ideal for tasks where comprehension is key—such as sentiment analysis, classification, named entity recognition, and enhancing search ranking by grasping the full context of the query.
- Decoder-Only Models: With a focus on generating text one token at a time, decoder-only models excel in creative and free-form generation. This makes them perfect for building conversational agents, completing text prompts, generating stories, and even suggesting code for programming assistance.
- Encoder-Decoder Models: Built for mapping one sequence to another, these models shine in tasks requiring a clear transformation of the input. They power machine translation, summarization, paraphrasing, and other applications where the output must closely relate to the provided input.